PerAdaNewsroom

10.2417/1200809.0651

Construct: towards a community middleware for pervasive computing

Simon Dobson, Graeme Stevenson, Graham Williamson, Stephen Knox, Matthew Stabeler, Lorcan Coyle, Steve Neely, and Paddy Nixon

Pervasive computing requires a substantial investment in terms of infrastructure, both for sensors and software. We are developing a software platform to provide a scalable, extensible, standards-based, open-source basis from which to explore adaptive pervasive systems.

Pervasive computing provides a means of broadening and deepening the use of information technology in society. The tools and techniques of pervasive systems can be used to simplify interaction with web sites, provide advanced location-sensitive services for people on the move, and support all aspects of citizens' life in the community.

Integrating IT services into everyday life requires that we can sense the environment in which services are to be offered, and tailor the offering according to changes in that environment. People are not automata, however, and will often perform the same activity in slightly different ways. Moreover, the ways in which we sense their actions are inherently error-prone and imprecise, and we may make a number of different observations, from different sensors or information sources, of the same events. People's support needs also change over time. These triple problems of *situation identification*, *context fusion* and *behavioural evolution* constitute the major challenges to building robust pervasive services.

Whilst it has proved to be reasonably straightforward to implement individual *applications* for pervasive systems, for example tour guides [1] or healthcare [2], it has proved considerably more difficult to build pervasive *systems* in which a dynamic population of services share infrastructure, sensing, inferencing and capabilities. Each new system requires a considerable investment of time (to acquire expertise) and money (to create the necessary infrastructure), which act as barriers to entry. We aim to reduce these barriers and simplify the construction of extensible, long-lived pervasive systems.

We have developed our system, Construct, by identifying the "best of breed" techniques used for implementing pervasive systems successfully. We have collected these together into a middleware platform on which sensors and services may be deployed. Construct provides a uniform framework for situation identification and context fusion, while providing transparent data dissemination and node management [3].

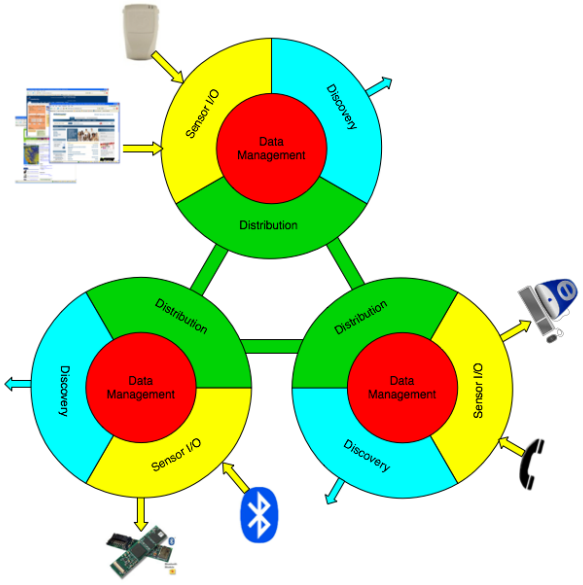

Figure 1. *Construct nodes form a network that disseminates data using gossiping*

Construct's basic architecture (figure 1) consists of a distributed collection of nodes accessed by services and sensors. Construct regards all data sources as sensors: physical sensors for temperature, pressure, location and so on, plus "virtual" sensors accessing digital and web resources. A sensor injects information into Construct's RDF triple store [4], which provides upper ontologies for common abstractions [5]. This model means that different sensors can be used to sense the same information: location may be sensed directly from RFID or Ubisense, inferred from diary or proximity information, and so on, but can be accessed by services using a common data model. Services query the triple store using the standard SPARQL query language.

Continued on next page

Construct does not provide remote access to sensors: instead, sensor data is disseminated around the network using a combination of node discovery and gossiping. In the standard configuration, Construct nodes discover each other using the Zeroconf protocol [6] and exchange data between them by gossiping [7] – both of these components are customisable. Gossiping means that nodes randomly synchronise their triple stores. Whilst this can lead to substantial "background" communications traffic, it increases the robustness of the system (since a node failure will not cause sensed data to be lost) and can improve responsiveness (since all queries are performed locally). There are interesting issues of the timeliness of data that are a topic of current research. Taken together, these approaches mean that developers need never deal with service discovery and other distributed systems issues, since these are abstracted into the middleware. The Construct architecture is completely modular, allowing us to (for example) experiment with adaptive gossiping [8], piggybacking and other protocols.

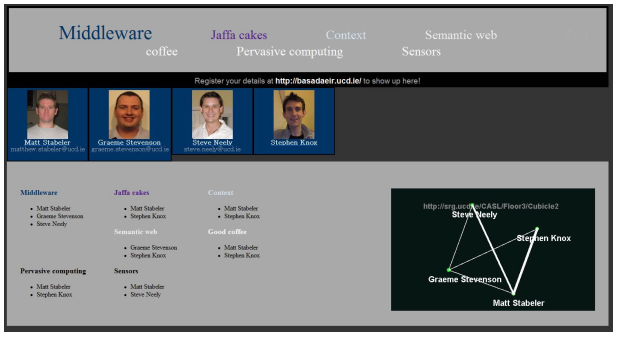

Figure 2. *Basadaeir integrates physical sensing, digital information and social networks*

We have developed a suite of sensors including Ubisense and Bluetooth location, diary and web-scraping for online information. An example service is Basadaeir (the Irish word for "matchmaker"), which aims to provide a context-sensitive display of information of interest to a group of people. We use Bluetooth sensing of individual's mobile phones to detect their presence near the display, and fuse this with information about their research interests, available papers and events, and joint research projects. The resulting display (figure 2) will show research information relevant to the overlap of the individuals present – a combination of location sensing, digital information and social networking.

Construct provides a platform that we hope will be useful both to researchers and practitioners. By providing a basis on which to develop and exchange components, we aim to simplify the development of systems and convergence towards best practices and design patterns. This in turn will help the community to develop the techniques needed to realise truly pervasive services.

Beta-test releases of Construct and various sensors are available from `http://www.construct-infrastructure.org` under LGPL.

This work is partially supported by Science Foundation Ireland under grant numbers 04/RPI/1544, "Secure and Predictable Pervasive Computing," 03/CE2/I303-1, "Lero: the Irish Software Engineering Research Centre," and 05/RFP/CMS0062, "Towards a Semantics of Pervasive Computing."

Author Information

Simon Dobson
Graeme Stevenson, **Graham Williamson**, **Stephen Knox**, **Matthew Stabeler**, **Lorcan Coyle**, **Steve Neely** and **Paddy Nixon**
Systems Research Group
School of Computer Science and Informatics
UCD
Dublin, Ireland

UCD's Systems Research Group consists of five academic staff and over thirty researchers and graduate students. SRG's research focuses on the design and implementation of pervasive and adaptive systems. Projects cover middleware, applications development, mathematical models of adaptive behaviour, software engineering, visualisation and sensor design. The group is supported by national, international and industrial sources, and publishes extensively in leading journals and conferences.

References

1. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, *Developing a context-aware electronic tourist guide: some issues and experiences* pp. 17–24, 2000.
2. G. Hayes, G. Abowd, J. Davis, M. B. andMaria Ebling, and E. Mynatt, *Opportunities for pervasive computing in chronic cancer care* **5013**, pp. 262–279, 2008.
3. L. Coyle, S. Neely, G. Stevenson, M. Sullivan, S. Dobson, and P. Nixon, *Sensor fusion-based middleware for smart homes*, **International Journal of Assistive Robotics and Mechatronics 8** (2), pp. 53–60, 2007.
4. O. Lassila and R. Swick, *Resource Description Framework model and syntax specification* tech. rep., World Wide Web Consortium, 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
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* 2006.
6. D. Steinberg and S. Cheshire, **Zero configuration networking: the definitive guide**, O'Reilly Media, 2005.
7. P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, *Epidemic information dissemination in distributed systems*, **IEEE Computer 37** (5), pp. 60–67, 2004.
8. S. Anawar, L. Coyle, S. Dobson, and P. Nixon, *Context delivery in ad hoc networks using enhanced gossiping algorithms* **LNCS 4272**, 2006.

Continued on next page