Ambiance: Adaptive Object Model-based Platform for Macroprogramming Sensor Networks

Reza Razavi
University of Luxembourg,
FSTC, LUXEMBOURG
razavi@acm.org

Kirill Mechitov,
Sameer Sundresh, Gul Agha
UIUC, IL, USA
{mechitov, sundresh, agha@cs.uiuc.edu}

Jean-François Perrot
Université Pierre et Marie Curie
LIP6, Paris, FRANCE
jean-francois.perrot@lip6.fr

Abstract

A pervasive computing system requires the integration of computation and communication with physical objects in the environment. The *Ambiance* project is building a platform for Macroprogramming pervasive systems based on an Adaptive Object-Model; the platform provides a meta-level architecture to analyze the execution context and customize the services it provides.

Categories and Subject Descriptors D.1.5 [**Programming Techniques**]: Object-oriented Programming – metalevel architectures; D.3.2 [**Programming Languages**]: Language Classifications – Specialized application languages; D.3.3 [**Programming Languages**]: Language Constructs and Features – Concurrent programming structures; D.3.4 [**Programming Languages**]: Processors – Code generation, Runtime environments.

General Terms Design, Languages, Verification, Reliability.

Keywords Sensor Networks, Actors, Adaptive Object-Models

1. Ambient Intelligence and Sensor Networks

Progress in *Ambient Intelligence* (AmI) technologies such as Micro-Electro-Mechanical Systems (MEMS), smart materials, dust networks, and bio-inspired software will enable the '*invisible*' incorporation in our surrounding environment of billions of sensing, actuating, computation and communication components in an *AmI infrastructure*. For example, PARC's *Smart Matter* project pursues a novel synthesis of these technologies to create intelligent programmable surfaces, particles, and materials that can sense, reason, and interact with their environment, and dynamically adjust their fundamental properties such as shape, aspect, and even load-bearing strength.

The AmI infrastructure will enable novel applications and new work practices in many fields including scientific observation and experimentation, monitoring (environment, health, etc.), medical diagnosis, and personalized services as described in [1].

A key *enabling technology* for AmI is networks of large numbers of wirelessly connected small, low-powered computers with very limited processing, memory, sensing, actuation, and communication ability. Such a system is called a *Wireless Sensor Network* (WSN), and each node a *mot*e. Moreover, motes are prone to failure (for example, if they run out of energy) and communication between them is unreliable. Programming such networks requires addressing those limitations. Unfortunately, current methods for WSN programming lead developers to mix high-level concerns such as quality of service requirements, for instance timeliness, reliability, application logic, adaptivity, with low-level concerns like resource management, synchronization, communication, routing, data filtering and aggregation. This makes developing software for WSNs a costly and error-prone endeavor, even for expert programmers.

We propose to simplify sensor network programming by developing *Ambiance*, a platform which supports *macroprogramming* by non-professional programmers. The idea of macroprogramming is to abstract away low-level concerns and to "*allow application designers to write code in a high-level language that captures the operation of the sensor network as a whole. The global program could then be compiled into a form that executes on individual nodes*" [2]. We propose to minimize the required programming knowledge to empower ordinary users to interact with the network. Moreover, the macroprogramming environment must provide decentralized execution under some form of control which ensures that global properties are satisfied. The control may be centralized, or the properties may result from an emergent behavior of the network.

2. State of the Art in Macroprogramming

A survey of the current solutions in the literature reveals a variety of approaches. These solutions include *a spreadsheet approach* [3], *ActorNet* [4], *EnviroSuite* [5], and *Regiment* [2]. Although many of these approaches are powerful, none of them provide the language abstractions that are required for macroprogramming as outlined above. For example, the spreadsheet approach, although targeting non-programmer end-users, uses an Excel spreadsheet to represent the layout of nodes and insert their functionality in the spreadsheet; queries are resolved by a logic program which generates a composition of services, where a service is a *.Net* component. In principle, this is quite general, however, it fixes a particular grid based naming scheme and does not allow for the definition of arbitrary groups of nodes and operations over such groups. ActorNet provides support for actor-based, mobile agents. In response to a query, a number of mobile agents may be created to coordinate in addressing the task. EnviroSuite proposes *environmentally immersive programming*, an object-based programming model in which individual objects represent physical elements in the external environment. In both cases, the actors and objects must be created explicitly to provide a service. Behavioral specifications are not in terms of groups of objects. Protocols to support operations over groups of objects and protocols to implement such specifications may not be reused. Most current solutions are designed for professional application programmers. Examples of *macroprogramming* languages in other domains include *PDL* for fault-tolerant distributed systems [6] and *explicit connectors* for coordination of distributed active objects [7].

OOPSLA'06 October 22–26, 2006, Portland, Oregon, USA.
ACM 1-59593-491-X/06/0010..

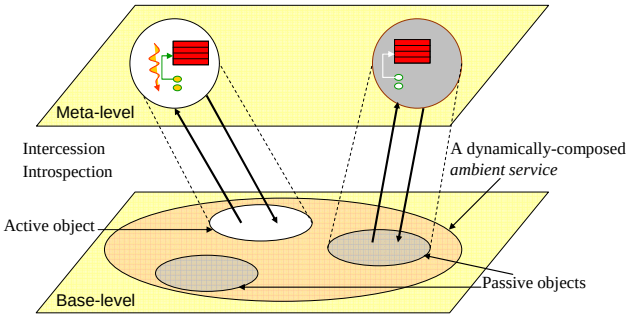

Figure 1: Meta-level and hybrid (passive and active objects) architecture of the *Ambiance* platform.

3. Adaptive Object Model-based Architecture

Adaptive Object-Models (AOMs) are meta-level architectures that enforce separation of concerns, and in particular the separation of business logic from technical aspects. Non-programmer experts customize, at run-time, the AOM by specifying its object-model (state and behavior), using a business-related language. [8]

In order to support macroprogramming of WSNs, we propose to extend this architectural style to translate high-level specifications of global behavior into both meta-objects and meta-actors which control and customize the runtime behavior of passive and active application objects (Figure 1). These meta-objects are dynamic, they have the capability to observe the application objects and the environment (*introspection*), and to customize their own behavior by analyzing these observations (*intercession*). Moreover, our architecture allows meta-objects to modify their behavior in more fundamental ways, for example, if the meta-objects are endowed with learning mechanisms.

Our architecture supports an open concurrent system – requests may come in asynchronously from the environment. A visual language is used for the high-level definition of structures and behaviors. *Dart* [9], a meta-level object-oriented framework for task-specific, artifact and activity-driven behavior modeling (Figure 2), offers the reifications needed for explicitly representing these specifications. Each specification is translated into a group of meta-actors which implement protocols to meet it. For example, in an AmI *parking lot* application, a vehicle identification query may be decomposed in terms of a set of meta-actors. These meta-actors control the concurrent and distributed collection of data by actors in the nodes of a sensor network and analyze the data. Note that the analysis may trigger further data collection, or reuse already collected data. The system being developed also provides a type system for a partial semantic check to ensure that queries are well-formed, for example, that a query does not result in a graph of node invocations which is cyclic.

4. Status of Our Implementation

Our implementation is building on our previous work, namely ActorNet and Dart (both described above). A first prototype, called *AmItalk*, was built as an extension to the *Mobidyc* [10] ecology simulation system. Mobidyc is an AOM used by ecologists to model ecosystems (individual and social behavior of animals, plants, etc.) and to study their evolution using simulations. AmItalk extends Mobidyc with a web interface that generates XML representations of the expert models, which are then interpreted by Mobidyc. Ambiance will generalize the capabilities of

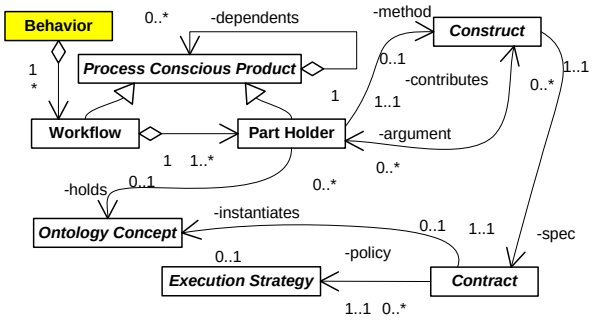

Figure 2: The core design of *Dart* in UML (class diagram).

AmItalk to arbitrary systems by replacing Mobidyc with ActorNet and its runtime engine. Specifically, queries will produce meta-actors which control the life-cycle of an actor. Such a lifecycle involves activation and registration, request management, application logic, and result dissemination. A type system for Dart is also being written in *Maude* (http://maude.cs.uiuc.edu). Once the core system is in place, we will work with experts in domains to which sensor networks are applicable, such as civil engineering, to experimentally assess and refine the Ambiance architecture.

5. Acknowledgments

This work is set in the framework of *Åmbiance*, a project in the field of Ambient Intelligence funded by the University of Luxembourg (R1F105K04). We would also like to acknowledge the valuable collaboration of A. Cardon, N. Bouraqadi, Ch. Dony, V. Ginot, J. Malenfant, M. Malvetti, R. Johnson, and Y. Kwon.

6. References

[1] IST Advisory Group. *Ambient intelligence: from vision to reality*, September 2003.

[2] Mainland, G., Kang, L., Lahaie, S., Parkes, D.C., and Welsh, M. *Using Virtual Markets to Program Global Behavior in Sensor Networks*. SIGOPSEW04, Belgium, September 2004.

[3] Woo, A., Seth, S., Olson, T., Liu J., and Zhao, F.: *A Spreadsheet Approach to Programming and Managing Sensor Networks*. IPSN'06, SPOTS Track, Nashville, TN, April 2006.

[4] Kwon, Y. M., Sundresh, S., Mechitov, K., Agha, G. *ActorNet: An Actor Platform for Wireless Sensor Networks.* AAMAS, 2006.

[5] Luo, L., Abdelzaher, F., He, T., Stankovic, J.A. *EnviroSuite: An Environmentally Immersive Programming Framework for Sensor Networks*. ACM TECS, 2006.

[6] Sturman, D., Agha, G. A Protocol Description Language for Customizing Failure Semantics. *13th Symposium on Reliable Distributed Systems* (SRDS), pp 148-157, October 1994.

[7] Ducasse, S., Günter, M. *Coordination of Active Objects by Means of Explicit Connectors*. DEXA Workshop 1998.

[8] Yoder, J.W., Johnson, R.E. *The Adaptive Object-Model Architectural Style*. WICSA3, pp.3-27, Montréal, CA, 2002.

[9] Razavi, R.: "*Outils pour les Langages d'Experts – Adaptation, Refactoring et Réflexivité*". Ph.D. Thesis, Paris 6, 2001.

[10] Ginot, V., Le Page, C., Souissi, S: A multi-agents architecture to enhance end-user individual-based modeling. Ecological Modeling 157, pp.23-41, 2002.