Developing intelligent sensor networks —a technological convergence approach

Emil Vassev, Mike Hinchey, Paddy Nixon

Publication date

01-01-2010

Published in

SESENA '10 Proceedings of the 2010 ICSE Workshop on Software Engineering for Sensor Network Applications;pp. 66-71

Developing Intelligent Sensor Networks — A Technological Convergence Approach

Emil Vassev
Lero—the Irish Software Engineering Research Centre, University College Dublin, Ireland
emil.vassev@lero.ie

Mike Hinchey
Lero—the Irish Software Engineering Research Centre, University of Limerick, Ireland
mike.hinchey@lero.ie

Paddy Nixon
Lero—the Irish Software Engineering Research Centre, University College Dublin, Ireland
paddy.nixon@lero.ie

ABSTRACT
We present a technological convergence approach to developing sensor networks capable of self-management. We use ASSL (Autonomic System Specification Language) to formally develop autonomous intelligent sensor nodes and DMF (Demand Migration Framework) to connect these nodes in a sensor network. ASSL provides constructs for modeling special self-management policies that drive the sensor nodes' behavior and control the communication mechanism provided by DMF.

Categories and Subject Descriptors
I.2.2 [**Automatic Programming**]: Program synthesis, Program transformation; D.2.11 [**Software Architectures**]: Domain-specific architectures; D.3.2 [**Programming Languages**]: Language classifications - *concurrent, distributed and parallel languages, very high-level languages*; D.2.10 [**Software Engineering**]: Design - *methodologies*; I.5.5 [**Implementation**]: Special architectures

General Terms
Algorithms, Performance, Design, Reliability, Experimentation

Keywords
sensor networks; distributed systems; architecture; ASSL; DMF

1. INTRODUCTION
Sensor networks [1] employ the latest in computing and communication technology to provide a sensing, computing, and communication mechanism that helps us observe and act on events occurring in physical and cyber infrastructures. Such networks operate over sensors collecting and processing data in diverse domains, such as air quality control, weather forecasting, traffic control, security and surveillance applications, etc. Although there have been great advances in the field of sensor networks (cf. Section 2), the development of resource-efficient sensor networks able to adapt to situations in order to improve their efficiency is still a challenging task. Such a "smart" behavior requires "intelligent" sensor nodes able not only to sense the environment but also to reason and collaborate with other sensor nodes in the network. Such sensor networks (SNs) we term *intelligent sensor networks* (ISNs).

This research aims at building ISNs capable of self-management. We consider such systems to be autonomic systems (ASs) [2] employing self-management by virtue of special policies driving the network in question in critical situations. In general, the AS paradigm draws inspiration from the human body's autonomic nervous system. The idea is that software systems can manage themselves and deal with dynamic requirements, as well as unanticipated changes, automatically, just as the human body does, through self-management based on high-level objectives [3]. Our approach to the development of autonomic systems is ASSL (Autonomic System Specification Language), which is an initiative promoting formal specification, validation, and code generation of ASs within a framework [4, 5].

In order to build intelligent sensor nodes exhibiting AS features, we draw upon our experience with the ASSL framework. Note that with ASSL we successfully built ASs, such as prototypes simulating both the NASA ANTS [6] and the NASA Voyager [7] missions. Moreover, in order to connect these intelligent sensor nodes in an ISN, we refer to a special networking mechanism called DMF (Demand Migration Framework) [8]. Neither ASSL nor DMF were originally developed for the purpose of building ISNs, but the combination of both allows for this successful technological convergence.

The rest of this paper is organized as follows. In Section 2, we review work related to intelligent networks such as 1) adaptable networks employing certain intelligent behavior; 2) energy-aware sensor networks employing energy-management algorithms; and 3) agent-based ISNs incorporating self-management features. In Section 3, we present preliminaries in terms of technologies, which must be introduced to the reader first. In this section, we briefly introduce the nature of sensor networks together with a brief introduction to ASSL and DMF. In section 4, we present our approach to the development of ISNs using ASSL and DMF. We also present a case study demonstrating how our approach can be applied for developing an ISN useful for home automation. Finally, Section 5 provides brief concluding remarks and a summary of future research plans..

2. RELATED WORK

One of the important aspects of any SN is the underlying network mechanism. By their nature, SNs are distributed networks with multiple nodes exchanging messages (cf. Section 3.1). Moreover, often network nodes can be used as re-transmitters and thus, there may be multiple routing paths used to deliver a message from a source to a destination. This problem is tackled by so-called ad-hoc networks employing special adaptive routing protocols. Such networks decide on-the-fly the most appropriate route considering different factors such as: current network status, performance measures, cost of transmission over a given route, reliability of a path, time of transmission, etc.

Considerable work has been done on routing protocols in ad-hoc networks. For example, routing protocols for mobile wireless networks are discussed in [9, 10, 11]. Another example is special routing algorithms based on game theory developed by Altman et al. [11].

Another aspect of SN intelligence is energy management. Practice has shown that energy efficiency appears to be of crucial importance for both performance and reliability of any energy-independent (battery-driven) SNs. Algorithms for energy awareness and management have been developed, where network intelligence is implemented at the level of single node or/and at the global level of the entire network [12].

An approach to ISNs providing autonomic behavior is described in [13]. Similar to our approach, in that work an ISN is realized through the use of multi-agent architecture and self-management behavior.

In [14] an agent oriented programming paradigm for the development of intelligent sensor networks is presented. The proposed architecture for ISNs consists of autonomous intelligent agents that interact with other agents over special high-level communication protocol implementing a special declarative high-level agent communication language.

In our approach, we do not aim at efficient routing algorithms, although such can be implemented with ASSL as a global network-level behavior. Instead, using ASSL we develop intelligent autonomous units embedding sensors and driven by self-management policies. Moreover, we may use ASSL to specify global self-management policies, those working at the network level and forming global network-level intelligence. Therefore, an ASSL-developed ISN usually employs an intelligent behavior at both the unit level and the network level. Moreover, in our approach, the networking mechanism exposes a centralized topology and is independent of sensor nodes. This makes an ISN both reliable and efficient, since its network nodes are volunteers and any node can be easily replaced by new one, without interrupting the entire network.

3. PRELIMINARIES

3.1 Sensor Networks

In general, a sensor network is composed of sensor nodes connected to other sensor nodes. The network connection usually is wireless. Sensor nodes usually rely on a routing protocol to communicate with other nodes not directly connected to the first. Moreover, usually a sensor node has limited computational power and storage space. This is due to the fact that in most cases, sensor networks rely on batteries where high-performance hardware cannot be efficiently supplied with energy [1, 15].

In our approach, sensor nodes are considered to have enough computational power to run both a JVM and the Java-implemented self-management control software generated with ASSL. Note that ASSL generates Java code [4] and the employed DMS [8] (a DMF instance) is JINI-based [15], which is a Java application as well.

3.2 ASSL

Although ASSL is dedicated to autonomic computing [2], with this work we demonstrate how it can be used for the development of sensor networks with self-management capabilities. In this subsection, we present the ASSL specification model by emphasizing special features that make the framework suitable for the development of ISNs.

3.2.1 ASSL Specification Model

ASSL is based on a specification model exposed over hierarchically organized formalization tiers. The ASSL specification model is intended to provide both infrastructure elements and mechanisms needed by an AS (autonomic system), or in this case by an ISN.

Table 1. ASSL multi-tier specification model

AS		**AS Service-Level Objectives**
		AS Self-Management Policies
		AS Architecture
		AS Actions
		AS Events
		AS Metrics
ASIP		**AS Messages**
		AS Channels
		AS Functions
AE		**AE Service-Level Objectives**
		AE Self-Management Policies
		AE Friends
	AEIP	**AE Messages**
		AE Channels
		AE Functions
		AE Managed Elements
		AE Recovery Protocols
		AE Behavior Models
		AE Outcomes
		AE Actions
		AE Events
		AE Metrics

Each tier of the ASSL specification model is intended to describe different aspects of the AS under consideration, such as special *service-level objectives, policies, interaction protocols, events, actions*, etc. This helps to specify an AS at different levels of abstraction imposed by the ASSL tiers (cf. Table 1).

The ASSL specification model considers the ASs as being composed of special autonomic elements (AEs) interacting over interaction protocols, whose specification is distributed among the ASSL tiers. Note that although ASSL allows for specification and code generation of interaction protocols, the latter cannot be used as an ISN networking mechanism, because ASSL currently does not generate distributed systems. Instead, it generates multithreaded systems with embedded messaging. Here, we rely 1) on ASSL to specify and generate sensor nodes in the form of AEs; and 2) on DMF to implement the needed networking mechanism, which connects the nodes together.

Table 1 presents the multi-tier specification model of ASSL. As shown, it decomposes an AS in two directions:

1) into levels of functional abstraction;
2) into functionally related tiers (sub-tiers).

With the first decomposition (cf. first column in Table 1), an AS is presented from three different perspectives depicted as three main tiers:

1) *AS Tier* forms a general and global AS perspective exposing the architecture topology, general system behavior rules, and global actions, events, and metrics applied to these rules.
2) *ASIP Tier* (AS interaction protocol) forms a communication perspective exposing a means of communication for the AS under consideration.
3) *AE Tier* forms a unit-level perspective, where an interacting sets of the AS's individual components is specified. These components are specified as AEs with their own behavior, which must be synchronized with the behavior rules from the global AS perspective.

It is important to mention that the ASSL tiers are intended to specify different aspects of the AS in question but it is not necessary to employ all of them in order to model an ISN. Thus, to specify a simple ISN, we need to specify a single AE per sensor node providing the self-management control software controlling the node's sensors and the communication with other AEs. Moreover, self-management policies must be specified to provide self-management behavior at the level of AS (the AS tier) and at the level of AE (AE tier). Note that this rule is implied by the fact that all the ASSL specifications must be AC-driven, i.e., based on self-management [2].

In the following sub-subsection, we present some of the ASSL constructs needed to specify an ISN.

3.2.2 Self-management Policies

The self-management behavior of an AS (or ISN), is specified with ASSL self-management policies (cf. the appropriate tiers in Table 1). These policies are specified with special ASSL constructs termed *fluents* and *mappings*:

- A fluent is a state where an AS enters with fluent-activating events and exits with fluent-terminating events.
- A mapping connects fluents with particular actions to be undertaken.

Self-management policies are driven by events and actions determined in a deterministic manner, similar to finite state machines. For the purpose of ISN development, self-management policies may be specified to control the network sensors and the sending and receiving of messages. Moreover, both network-level (at the AS tier) and node-level (at the AE tier) self-optimizing policies may be specified as already mentioned. Self-management policies may be used to control the communication in real-time systems bounded by deadlines, where the deadline may be a particular time or time interval, or may be the arrival of some event. Thus, ASSL may specify real-time ASs where events can be used to trigger different policies intended to solve problems when the deadline cannot be met.

3.2.3 ASSL Events

ASSL aims at event-driven autonomic behavior. Hence, to specify self-management policies driving the sensor nodes of an ISN, we need to specify appropriate events. Here, we rely on the reach set of event types exposed by ASSL. To specify ASSL events, one may use logical expressions over service-level objectives (SLO), or may attach events to metrics (cf. Section 3.2.4), other events, actions, time, and messages. Moreover, ASSL allows for the specification of special conditions that must be stated before an event is prompted.

3.2.4 ASSL Metrics

For an ISN, one of the most important success factors is the ability to sense the environment and to react on sensed events. Here, together with the reachable set of events, ASSL imposes metrics as a means of determining dynamic information about external and internal points of interest. Although four different types of metrics are allowed [4], for the needs of ISN development the most important are the so-called resource metrics intended to measure special managed resource quantities. Note that a *managed resource* (cf. Section 3.2.5) can be a controlled sensor. In such a case, an ASSL metric is linked with a network sensor.

3.2.5 Managed Resources

An AE typically controls a managed resource specified with ASSL in the form of managed elements [4]. A managed element (ME) generally is a functional unit (hardware or software) controlled by an AE. In an ASSL-developed INS, a ME presents a controlled sensor (or a group of sensors). In order to understand how an ASSL-developed INS works, it is important to understand the AE-ME relationship. Note that an AE monitors and interacts with its MEs.

In ASSL, a ME is specified with a set of special interface functions intended to provide control functionality over the same. ASSL provides an abstraction of a ME through specified interface functions. With the framework we can specify and generate the interface controlling a ME, but not the implementation of this interface in that controlled ME. Thus, when developing an ISN, the generated interfaces must be implemented by the appropriate sensor drivers.

3.3 DMF

DMF (Demand Migration Framework) [8] is a generic scheme for migrating information in the form of messaging objects, in a heterogeneous and distributed environment determined by both information senders and information recipients (both termed as communication nodes). The framework provides a context for performing migration activities, where the migrated messaging objects encapsulate both behavior and data. In general, DMF may be used to derive a generic architecture for the implementation of special family of Demand Migration Systems (DMSs) [8].

3.3.1 Rationale
Originally, DMF was developed in [8] to define a generic framework for object-migration in a heterogeneous and distributed environment. By applying DMF, we can design a variety of DMSs conforming to a set of requirements described by the following elements [8]:

- *platform interoperability* – deals with process-machine boundaries and diversification of different hosting platforms (connects nodes running on Linux, Solaris, Windows, and Mac-OS platforms);
- *"at least once" delivery semantics* – ensures that no object can be delivered to the wrong recipient and must be delivered at least once;
- *asynchronous communication* – communication nodes run independently and have no synchronized lifecycles;
- *no prioritization* – both objects and communication nodes are served equally by the system;
- *secure communication*;
- *fault-tolerant migration*;
- *hot-plugging* – communication nodes are "volunteers" in the communication process.

3.3.2 DMF Architectural Model for ISNs
DMF defines a layer-structured architecture. Figure 1 depicts an ISN-elaborated variant of that architecture. As shown, the largest circle depicts an ISN as composed of AEs controlling sensor nodes. The double-lined inner circle depicts DMF. Here, AEs (autonomic elements) are communication nodes, and DMF is a communication intermediate between them. DMF consists of two main functional layers called Demand Dispatcher (DD) and Migration Layer (ML) respectively. DMF relies on these two functional layers to form an asynchronous message-persistent communication protocol where messages are permanently stored and delivered upon request.

DD (depicted by a bold-lined circle; cf. Figure 1) is an object-based storage mechanism able to dispatch messaging objects to their recipients. ML (depicted as a dark grayed layer on top of DD) is the layer performing object migration from an AE to another AE. ML makes the communication in a heterogeneous distributed environment possible. In addition, ML emphasizes the use of special kind of agents called transport agents (TAs), which are based on distributed technologies. To use DMF as a communication protocol, the AEs of an ISN must adhere to the special interface defined by DMF TAs. Moreover, the DD (demand dispatcher) layer establishes a context of demand propagator that consists of two layers—Demand Space (DS) and Presentation Layer (PL) (cf. Figure 1). The DS layer defines a context of internal object-based storage mechanism. PL is an abstract layer on top of DS that makes all the DS functionality transparent and generic.

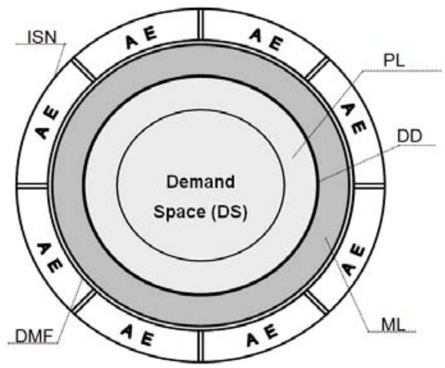

Figure 1. DMF - Demand Migration Framework

3.3.3 DMS for ISNs
The communication protocol of an ISN is an instance of DMF termed DMS (Demand Migration System). In our approach, we rely on the DMS described in [8]. This DMS defines TAs based on JINI [15] and termed JINI TAs. Here, the AEs controlling the sensors of an ISN adhere to the JINI TA interface.

4. BUILDING INTELLIGENT SENSOR NETWORKS?
We consider ISNs as sensor networks composed of sensors employing an event-driven behavioral mechanism that helps the network react to changes in the environment or in the network structure (e.g., a node is down). We build an ISN using ASSL to specify and generate intelligent sensor nodes as AEs. Next, we connect the generated AEs in a network by using a special DMS instantiated from DMF and exposing JINI TAs (cf. Figure 2).

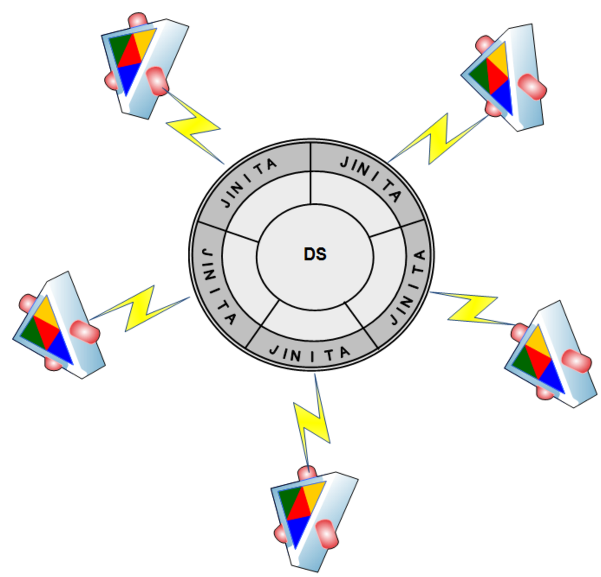

Figure 2. ISN's sensor nodes communicate via a DMS

Figure 2 depicts a conceptual model of our network. As shown, the AEs controlling network sensors are connected in a network through a JINI-based DMS. Note that the network topology is centralized. The DMS stores the messages sent by AEs and delivers them to the recipient AEs when the latter are available. Moreover, in order to use the DMS, each AE connects to a JINI TA via a special interface. Although this is not depicted in Figure 2, note that a single JINI TA can be shared by multiple AEs.

4.1 Steps

The steps of building an ISN with ASSL and DMF are as follows:

1) Specify the AEs in terms of self-management behavior and ME using ASSL.
2) Generate the Java implementation of the AEs with ASSL.
3) Connect the generated AEs with the appropriate sensors through the generated ME interfaces.
4) Install the JINI DMS in place.
5) Connect the JINI TAs with the AEs through the generated ME interfaces.
6) Run the JINI DMS.
7) Run the ISN's AEs.

4.2 Case Study

In the course of this project we used ASSL and DMF to build an ISN for home automation. Our first step was to automate the living room of a house. Here, we used ASSL to specify four different AEs (autonomic elements) composing the ISN for home automation:

- *Light AE* – controls the lights in the living room. This AE uses light sensors to determine the level of brightness in the living room and uses the light switch to turn on/off the lights. Moreover, this AE communicates with the Motion AE to determine when and where there is motion in the room, which may prompt turning lights on.
- *Voice AE* – controls the microphones in the living room. This AE detects and recognizes speech. It communicates with the Light AE and with the Door AE to perform voice commands, such as "turn lights on/off" or "open/close door".
- *Motion AE* – controls motion detectors to sense the living room for motion. It zones the living room and detects where the motion is taking place and how many moving objects are there. Communicates with the Door AE and with the Light AE.
- *Door AE* – controls the door to open or close the same. It communicates with the Motion AE, e.g., when motion is detected towards the door, the Door AE opens the door automatically.

For each one of these AEs we specified policies and MEs. The following ASSL fragments present a partial specification of the Voice AE. The first specification presents a self-management policy. This policy determines the behavior of the AE when speech is detected. Events are specified to initiate and terminate fluents within this policy. Here, as shown, when speech is detected (via the controlled microphones) the AE starts this policy in an attempt to recognize a voice command. If such is recognized, it will be propagated to the Door AE (if it is "open/close door") or to the Light AE (if it is "turn on/off lights") through the DMS run in place using a JINI TA.

```
AESELF_MANAGEMENT {
  OTHER_POLICIES {
    POLICY MANAGE_VOICE_COMMAND {
      FLUENT inSpeach {
        INITIATED_BY { EVENTS.speechDetected }
        TERMINATED_BY { EVENTS.commandRecognized,
                                   EVENTS.commandNotRecognized }
      }
      FLUENT inCommandRecognized {
        INITIATED_BY { EVENTS. commandRecognized }
        TERMINATED_BY { EVENTS.commandProcessed }
      }
....
      MAPPING { CONDITIONS { inSpeach }
                DO_ACTIONS { ACTIONS.recognizeCommand } }
      MAPPING { CONDITIONS { inCommandRecognized }
                DO_ACTIONS { ACTIONS.processCommand } }
....  }}}
```

To control both the microphones and the JINI TA (transport agent), the Voice AE specifies two MEs (managed elements) determining the control interface for each one. The following ASSL specification snippet presents the specified MEs.

```
MANAGED_ELEMENTS {
  MANAGED_ELEMENT MICROPHONES {
    INTERFACE_FUNCTION speechDetected { RETURNS { Boolean } }
    INTERFACE_FUNCTION retrieveCommand { RETURNS { String } }
  }
  MANAGED_ELEMENT JINI_TA {
    INTERFACE_FUNCTION sentMessage {
      PARAMETERS { ISNMessage oMessage } }
    INTERFACE_FUNCTION receiveMessage {
      RETURNS { ISNMessage } }
  } }
```

The specified ME interfaces help the Voice AE detect speech, retrieve a voice command from the detected speech, and send and receive messages. Note that Light AE controls not only the sensors (microphones) but also the TA (JINI TA) allowing this AE to communicate through the DMS run in place.

4.3 Test Results

In this case study, we specified and generated the four AEs composing the ISN for home automation. We also put together the generated AEs (autonomic elements) and the JINI DMS. However, we did not use real sensors. Instead, we simulated sensing the home automation environment with ASSL events specified to simulate sensor activity. Note that events trigger the specified policies by initiating appropriate fluents (cf. the AESELF_MANAGEMENT ASSL specification sample above). Thus, with such events we were able to simulate speech detection, voice command recognition, and other sensor-related events.

Table 2 shows some of the multiple experiments we performed with the ASSL-developed ISN prototype. The results (extracted from the generated log records) demonstrated that, under simulated conditions, the run-time behavior of the ISN strictly followed the ASSL-specified self-management policies. Moreover, the ISN's AEs were able to exchange messages through the JINI DMS run in place.

Table 2. Experiments with the ISN Prototype

Test Case	Simulated Conditions	Results
speech detection (Voice AE)	voice command "open door"	called interface function MICROPHONES.speechDetected; occurred event speechDetected; activated fluent inSpeach;

correct voice command (Voice AE)	voice command "open door"	test case "speech detection"; performed action recognizeCommand; called interface function MICROPHONES.retrieveCommand; called interface function WIRELESS_NETWORK.sendMessage; occurred event commandRecognized; terminated fluent inSpeach;
incorrect voice command (Voice AE)	voice command "open window"	test case "speech detection"; performed action recognizeCommand; called interface function MICROPHONES.retrieveCommand; occurred event commandNotRecognized; terminated fluent inSpeach;
open door (Voice AE & Door AE)	voice command "open door"; door closed	test case "correct voice command"; occurred event mustOpenDoor; activated fluent inOpenDoor; performed action openDoor; called interface function DOOR.isDoorOpen; called interface function DOOR.open; occurred event doorOpened; terminated fluent inOpenDoor;

5. CONCLUSION AND FUTURE WORK

We have demonstrated how ASSL, a tool for formal specification of autonomic systems, and DMF, a distributed computing framework, can be used together to develop ISNs (intelligent sensor networks) with self-management capabilities. In our approach, we use ASSL to specify behavioral policies provided by special AEs (autonomic elements) intended to control sensors via special MEs (managed elements). We assume ISNs composed of AEs, which are specified with suitable ASSL specification constructs, and subsequently, their Java implementation is automatically generated. Moreover, in our approach we use a JINI-based DMS (demand migration system) to connect the generated AEs. This DMS provides a networking protocol needed by an ASSL-developed ISN, where the ISN's AEs use special JINI TAs (transport agents) to communicate. As a proof of concept, we have successfully built an ISN for home automation, where sensors are simulated with special events.

Future work is concerned with further ISN experiments and development by including hardware attached to the control software generated by the ASSL framework. Moreover, we intend to build ISN prototypes incorporating self-managing policies such as self-healing, self-protecting, and self-adapting. This will help us to investigate and develop ISNs able to automatically detect and fix performance problems, e.g., by switching to alternative sensors.

ACKNOWLEDGMENT

This work was supported in part by Science Foundation Ireland grant 03/CE2/I303_1 to Lero—the Irish Software Engineering Research Centre.

REFERENCES

[1] Cook, D.J. and Das, S.K. (ed.). 2004. Smart Environments: Technologies, Protocols, and Applications. John Wiley, New York.

[2] Murch, R. 2004. Autonomic Computing: On Demand Series. IBM Press, Prentice Hall.

[3] Horn, P., 2001. Autonomic Computing: IBM's Perspective on the State of Information Technology. Technical Report. IBM T. J. Watson Laboratory.

[4] Vassev, E. 2008. Towards a Framework for Specification and Code Generation of Autonomic Systems. Doctoral Thesis. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada.

[5] Vassev, E. and Hinchey, M. 2009. ASSL: A Software Engineering Approach to Autonomic Computing. IEEE Computer. 42, 6 (June 2009), pp. 90–93.

[6] Vassev, E., Hinchey, M., and Paquet, J. 2008. Towards an ASSL Specification Model for NASA Swarm-Based Exploration Missions. In Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008) - AC Track. ACM Press, pp. 1652–1657.

[7] Vassev, E. and Hinchey, M. 2009. Modeling the Image-Processing Behavior of the NASA Voyager Mission with ASSL. In Proceedings of 3rd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'09), IEEE Computer Society, pp. 246-253.

[8] Vassev, E. 2005. General Architecture for Demand Migration in the GIPSY Demand-Driven Execution Engine. M. Sc. Thesis. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, ISBN 0494102969.

[9] Perkins, C. and Bhagwat, P. 1994. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In Proceedings of SIGCOMM. ACM Press, pp. 234–244.

[10] Das, S., Perkins, C., and Royer, E. 2000. Performance Comparison of Two On-demand Routing Protocols for Ad-hoc Networks. In Proceedings of INFOCOM 2000, IEEE Computer Society, 3–12.

[11] Altman, E., Basar, T., Jimenez, T., and Shimkin, N. 2002. Competitive routing in networks with polynomial costs, IEEE Trans, Automat. Control, 47, 1, pp. 92–96.

[12] Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M. B. 2002. Energy-aware Wireless Microsensor Networks. IEEE Signal Processing Magazine, IEEE Computer Society, 19, 2, pp. 40–50.

[13] Marsh, D., Tynan, R., O'Kane, D., and O'Hare, G. 2004. Autonomic Wireless Sensor Networks. Engineering Applications of Artificial Intelligence, Elsevier.

[14] Karlsson, B., Bäckström, O., Kulesza, W., Axelsson, L. 2005. Intelligent Sensor Networks - an Agent-Oriented Approach, In Proceedings of Workshop on Real-World Wireless Sensor Networks (REALWSN'05), Sweden.

[15] Flenner, R. 2001. JINI and JavaSpaces Application Development, Sams Publishing, Indianapolis.