REACTIVE, DISTRIBUTED AND AUTONOMIC COMPUTING ASPECTS OF AS-TRM

E. Vassev, H. Kuang, O. Ormandjieva, J. Paquet
Department of Computer Science and Software Engineering, Concordia University, EV 3.165, 1455 de Maisonneuve West, Montreal, Quebec, H3G 1M8, Canada

Keywords: Autonomic Computing, Distributed Computing, Reactive Systems, Software Architecture.

Abstract: The main objective of this research is a rigorous investigation of an architectural approach for developing and evolving reactive autonomic (self-managing) systems, and for continuous monitoring of their quality. In this paper, we draw upon our research experience and the experience of other autonomic computing researchers to discuss the main aspects of Autonomic Systems Timed Reactive Model (AS-TRM) architecture and demonstrate its reactive, distributed and autonomic computing nature. To our knowledge, ours is the first attempt to model reactive behavior in the autonomic systems.

1 INTRODUCTION

Autonomic computing is a new research area led by IBM Corporation, which area concentrates on making complex computing systems smarter and easier to manage (Kephart, Chess, 2003, Horn, 2001, Ganeck, Corbi, 2001). The main characteristic of autonomic computing is self-management, i.e., self-monitoring of its own use and quality in the face of changing configurations and external conditions, based on automatic problem-determination algorithms. Many of autonomic systems concepts imitate self-regulatory model of human autonomic system; thus autonomic computer systems are envisioned to combine the following seven characteristics: self-configuring, self-healing, self-optimizing self-protecting, self-defining, contextually aware and anticipatory (Kephart, Chess, 2003, Horn, 2001). The first four characteristics listed above are considered to be the core characteristics of an autonomic computer system (McCann, Huebscher, 2004).

However, according to our best knowledge, autonomic computing technology has not been applied to model and develop real-time reactive systems, which systems have high demand for autonomic computing technology to remove the complexity of modeling and development. With autonomic behavior, real-time reactive systems will be more self-managed to themselves and more adaptive to their environment.

Research Goals. The long-term research goals for this project are: 1) modeling of distributed autonomic reactive components along with their relationships; 2) modeling of the qualitative properties constraining systems' behavior, such as reliability. In order to achieve our research goals, we need to: 1) develop an appropriate formal framework for autonomic distributed real-time reactive systems that leverages their modeling, development, integration and maintenance, as well as for continuous self-monitoring of their quality formalism to support distributed autonomic behavior, and 2) build corresponding architecture along with communication mechanism to implement distributed autonomic behavior. In this paper we describe the architecture and the communication mechanism for of Autonomic Systems Timed Reactive Model (AS-TRM) by revealing its reactive, distributed and autonomic computing nature.

Our Approach. We add reactiveness to the autonomic components' behavior thus allowing them to communicate and synchronize with the environment while fulfilling their tasks. The novelty of our approach consists in combining the advantages of both the formal representation of reactive components in TROM formalism (Achuthan, 1995), and the autonomy of components in agent-oriented paradigm. Our objectives include

Vassev E., Kuang H., Ormandjieva O. and Paquet J. (2006).
REACTIVE, DISTRIBUTED AND AUTONOMIC COMPUTING ASPECTS OF AS-TRM.
In *Proceedings of the First International Conference on Software and Data Technologies*, pages 196-202
DOI: 10.5220/0001313601960202

an extension of TROM as an Autonomic System Timed Reactive Model (AS-TRM) to include the specification of distributed reactive components along with their relationships, and the non-functional properties constraining systems' behavior.

Fig. 1 illustrates the concept of the Reactive Autonomic System AS-TRM. To our knowledge, ours is the first attempt to model reactiveness in autonomic systems.

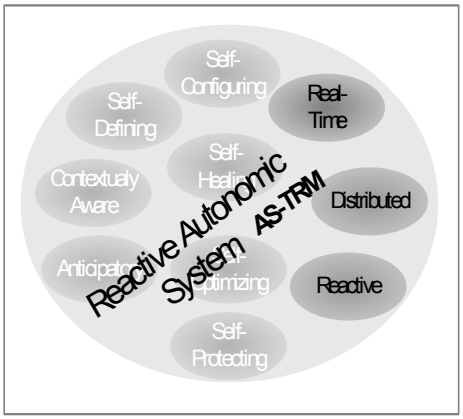

Figure 1: The characteristics of visionary Reactive Autonomic System AS-TRM.

This paper is organized as follows: The AS-TRM architecture fulfilling the requirements of both autonomic and reactive behavior is introduced in section 2. Section 3 presents the structure of the AS-TRM communication system. In section 4, we review the related work. Finally, we present our conclusion and future work.

2 AS-TRM ARCHITECTURE

This section provides the comprehensive conceptual view of the AS-TRM architecture; it is intended to capture and convey the significant architectural decisions, which serve as a foundation for the further design and implementation. We focus on the structural and the dynamic view, as well as on the specific characteristics of AS-TRM to discuss its reactive, distributed and autonomic aspects.

Our AS-TRM architecture builds upon extending the TROM formalism (Achuthan, 1995) for modeling reactive systems. Reactive systems are the computer systems that continuously react to their physical environment, continually sensing and responding to the environment, at the speed determined by the environment. Reactive autonomic systems have infinite behavior and must satisfy the following two important requirements for reactiveness:

- stimulus synchronization: the process is always able to react to stimulus from the environment;
- response synchronization: the time elapsed between a stimulus and its response is acceptable to the relative dynamics of the environment so that the environment is still receptive to the response.

The TROM formalism for developing real-time reactive systems is briefly introduced below.

2.1 TROM

Real-time reactive systems are some of the most complex systems, so the modeling and development of real-time reactive systems becomes very challenging and difficult work. The TROM formalism (Achuthan, 1995) for real-time reactive systems developed at Concordia University is a powerful tool for dealing with complexity issues in developing such systems. The TROM formalism is a three-tier formal model (Achuthan, 1995). This three-tier structure describes system configuration, reactive classes, and relative Abstract Data Types (see Fig.2).

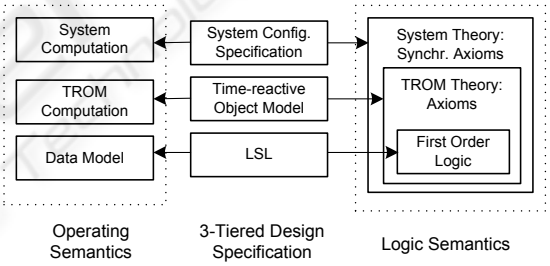

Figure 2: TROM Formal Model.

The lowest tier is the Larch Shared Language (LSL) tier, which specifies the Abstract Data Types used in the reactive classes (Achuthan, 1995). The middle tier is specifying the reactive classes named Generic Reactive Classes (GRCs). A GRC is a hierarchical finite state machine augmented with ports, attributes, logical assertions on the attributes, and time constraints. The upper-most tier is the System Configuration Specification, which models the collaboration between the reactive classes and their communication through port links (Achuthan, 1995). As a layered model, each upper tier communicates only with its immediate lower tier. The independence between the tiers makes the modularity, reuse, encapsulation, and hierarchical decomposition possible.

On the other hand, autonomic computing is the new research area, which focuses on developing complex computing system smarter and easier to manage. The goal of our work is to extend the current TROM formalism to AS-TRM to include the specification of distributed reactive autonomic components along with their relationships, and the qualitative properties constraining the system's behavior.

2.2 AS-TRM

AS-TRM can be considered as TROM with extended autonomic behavior, and the autonomic functionalities are those creating the autonomic behavior. Autonomic functionalities can be implemented locally, using locally maintained measurements and knowledge. The autonomic behavior can be implemented among the members within a peer group through sharing measurements and knowledge of the group.

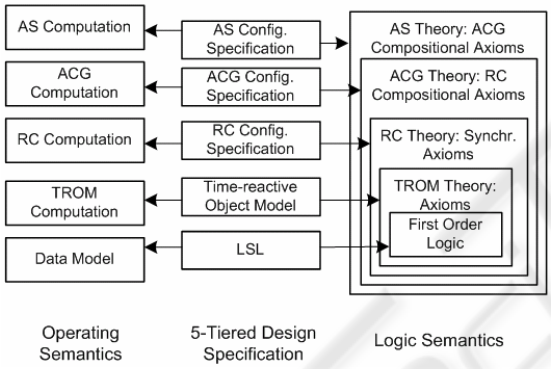

Figure 3: AS-TRM Formal Model.

AS-TRM extends the TROM formal model by adding more tiers (see Fig. 3) and including the specifications for a time-reactive component (RC), an autonomic group of synchronously interacting RCs (ACG) and an autonomic system (AS) that consists of asynchronously communicating ACGs.

RC. This newly added tier encapsulates TROM objects (the TROM formalism's second tier) into an AS-TRM reactive component. The synchronous interaction between the RCs allow for realization of a reactive task. The communication between RC and its upper tier ACG is realized through an interface and is asynchronous (see section 3.1).

ACG. AS-TRM Autonomic Group of RCs is a set of synchronously communicating RCs cooperating in fulfillment of a group task. Each ACG can accomplish a complete real-time reactive task independently. The self-monitoring behavior at the ACG tier and the asynchronous interaction between ACG and the RCs is realized by an ACG Manager (AGM). The responsibilities of an AGM include the following:

- Continuous monitoring of the ACG quality level required by the evolving nature of the peer group
- AGM monitors the behavior of the synchronously communicating RCs and analyzes the correctness of their functionalities according to the policies.
- AGM receives diagnostic messages from RCs and sends back treatment messages to them;
- AGM unplugs the broken RCs from their group and plugs them back when they are ready;
- AGM automates the initialization and maintenance according to evolving group configuration and changes in the run time;
- AGM encapsulates any data under control of this group and manages all data shared either between RCs or between the group and other groups.

The reactive behavior is modeled at the RC and ACG tiers. We model the environmental objects communicating with the system as RCs, and incorporate them into the ACG fulfilling the corresponding reactive task.

The self-healing and self-optimizing autonomic behavior can be implemented on peer group level as the following aspects:

- Automating backup of policies for the group;
- Knowledge and resource sharing within the group;
- Execution time optimization according to the empirical data.

AS. Autonomic System (AS) tier is abstracting a set of asynchronously communicating ACGs. The self-monitoring behavior and the asynchronous interaction between AS and the ACGs is realized by the Global Manager (GM). The responsibilities of the Global Manager (GM) include the following:

- Continuous monitoring of the AS quality level required enduring the safety of the autonomic system;
- GM verifies user access according to the security policies and different level privileges defined among GM, ACGs, RCs, and environment;
- GM monitors the behavior of the ACGs and analyzes whether they work correctly according to the policies;
- GM receives diagnostic messages from ACGs and sends back treatment messages to them;
- GM receives a request for updating the compositional rules for ACGs and synchronization axioms among RCs from the user, and forwards the updates to the AGMs.

The self-protecting, self-configuring, self-optimizing, and self-healing autonomic behavior can be implemented at the AS level as the following aspects:

- automating user access support;
- automating the configuration for users;
- knowledge and resource sharing within the system;
- automating the backup of the policies for the entire system.

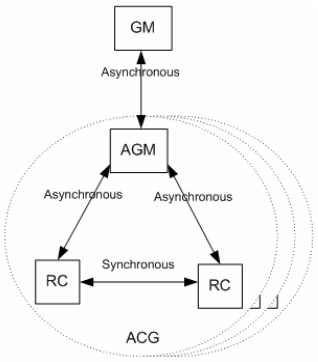

Figure 4: AS-TRM Architecture.

2.3 Characteristics of AS-TRM

Below we summarize the autonomic characteristics of the AS-TRM architecture shown in Fig. 4 (based on a figure in (Bantz et al, 2003)) in addition to real-time and reactive those inherited from the TROM formalism (Achuthan, 1995):

- AS-TRM is self-managed: it can monitor its components (internal knowledge) and its environment (external knowledge) by checking the status from them, so that it can adapt to changes that may occur, which may be known changes or unexpected changes;
- AS-TRM is distributed: the components within AS-TRM can collaborate to complete a common real-time reactive task distributively;
- AS-TRM is proactive: it can initiate changes to the system;
- AS-TRM is evolving: a) the policies of each RC can be changed in the run time according to the changes of requirements; b) the composition rules of the RCs within corresponding peer group can be changed in the run time; c) the synchronization axioms among the RCs within corresponding peer group can be changed in the run time.

2.4 Architecture of AS-TRM

Our architectural goal is to capture the above-mentioned characteristics of AS-TRM. The architecture of AS-TRM (see Fig. 4) is based on the tiers of the AS-TRM formal model, and consists of Reactive Components (RCs), AS-TRM Component Group Manager (AGM), and Global Manager (GM), which are connected to each other at the local, peer group, and system levels.

At the peer group level, which is also the AS-TRM Autonomic Group of RCs (ACG) level, every AGM interacts and shares knowledge as well as information with its RCs; it receives information (policies) from its superior (Global Manager) and implements them with its own resources. The autonomic behavior at this level is a result of peer knowledge-sharing, getting local agreement, and acting locally on that knowledge. Fig. 5 is another architectural view of AS-TRM.

ACG architecture. An ACG consists of an AGM and a set of managed RCs. An AGM consists of a collection of intelligent agents which are responsible for the autonomic behavior of self-configuring, self-healing, self-optimizing, as well as self-protecting, and a replicator for replicating the states of the RCs within the ACG. The intelligent agents in the AGM can communicate one another through the Autonomic Signal Channel. Each managed RC communicates its events and other measurements with the AGM. According to the input received from the RCs, the AGM makes the decisions based on the policies, facts, and rules (stored in the ACG repository) and communicates the instructions with corresponding RCs.

Anatomy of GM and RC. A GM consists of a set of intelligent agents which are responsible for the autonomic behavior of self-configuring, self-healing, self-optimizing, as well as self-protecting, and a replicator for replicating the states of the ACGs within the AS-TRM system. The intelligent agents in the GM can communicate each other through the Autonomic Signal Channel. Every ACG communicates its events and other measurements with the GM. According to the input received from the ACGs, the GM makes the decisions based on the policies, facts, and rules (stored in the AS repository) and communicates the instructions with corresponding ACGs.

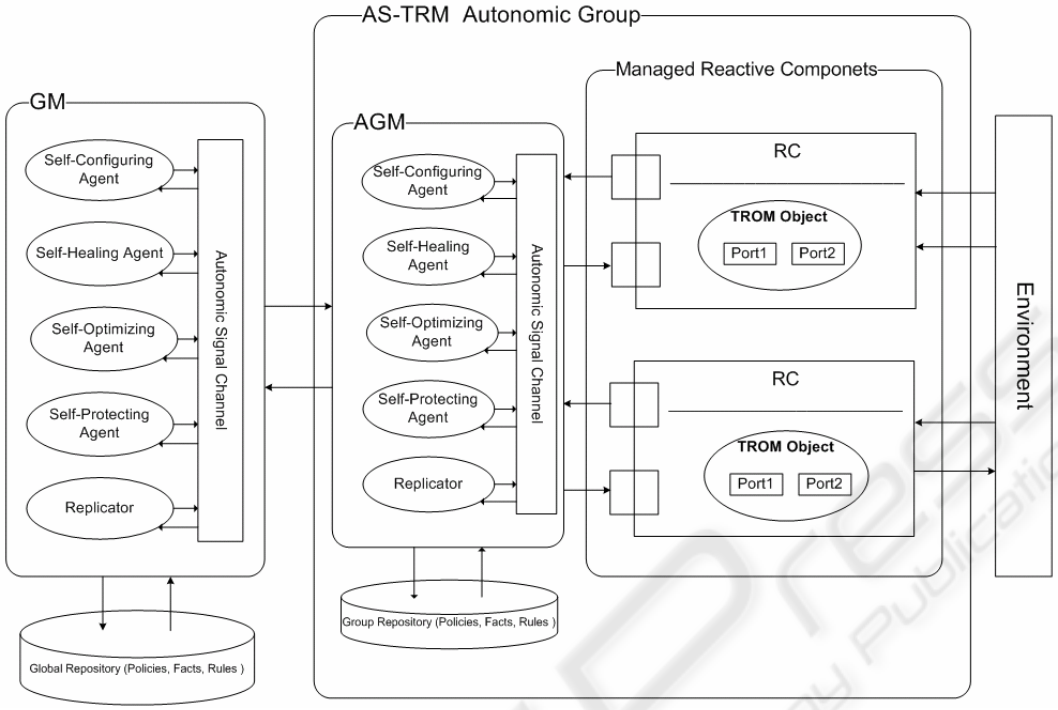

Figure 5: Anatomy of AS-TRM.

3 AS-TRM COMMUNICATION SYSTEM

The AS-TRM Communication System (ACS) is an autonomous messaging system in the AS-TRM that exposes interfaces for both synchronous and asynchronous message-delivery services. By virtue of its architecture, the ACS is an application of the Demand Migration Framework (DMF) (Vassev, 2005) which extends the DMF architecture by adding new features for adaptation to the autonomic computing needs. The ACS architecture provides two means of communication among AS-TRM entities (RC, AGM and GM) – asynchronous and synchronous (see section 2.3). Asynchronous communication was inherited from DMF centralized message-persistent asynchronous communication, and synchronous communication is a variant of peer-to-peer communication (Vassev, 2005). The former takes place between the RCs and the AGMs, and between the AGMs and the GM. Peer-to-peer communication takes place between RCs. Fig. 6 depicts the layered architecture of the ACS derived from the DMF. The architectural ACS model consists of four layers – Message Space (MS), Message Space Proxies (MSPs), Transport Agents (TAs) and Peer-to-Peer Transport Agents (P2PTAs). While the MS, MSP and TA layers are derived directly from the DMF (Vassev, 2005), the P2PTA layer is an ACS extension that addresses synchronous communication issues. The ACS inherently relies on MS, MSPs and TAs to "form architecture applicable to asynchronous communication systems, where the messages are delivered in a demand-driven manner" (Vassev, 2005). The MS incorporates a persistent storage mechanism for all the messages exchanged asynchronously in the AS-TRM. The MS in turn incorporates an Object Query Language (OQL) (Emmerich, 2000) for querying the stored messages. On top of this, we have the MSP presentation layer. There is a single MS and multiple MSPs in the model, each MSP being associated with a TA.

The TAs (see the dark grey segments named TA in Fig. 6) form a migration layer (Vassev, 2005) for transporting messages asynchronously to and from the RCs, AGMs and the GM that adhere to the TAs' interface. TAs are "independent stand-alone components able to carry objects over the machine boundaries" (Vassev, 2005). We use them to migrate messages from one node to another. The TAs provide a transparent form of migration. On top of the TA layer, we have the P2PTA layer (see the

white segments wrapping the TA layer and bordered with a dashed circle). P2PTAs provide an alternative means of communication, which is synchronous point-to-point communication. The RCs use the P2PTAs for direct synchronous communication.

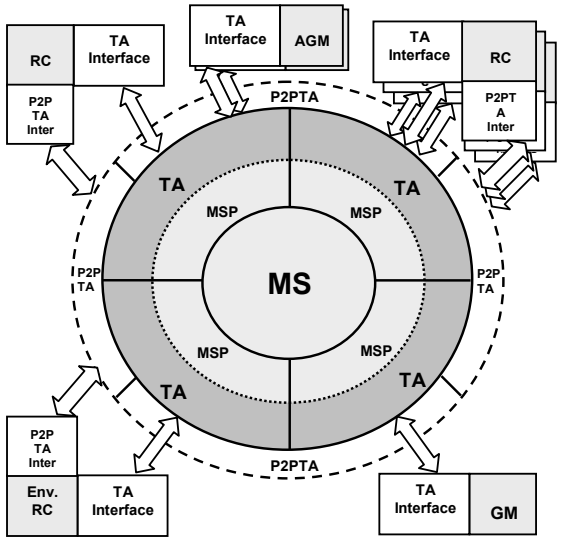

Figure 6: AS-TRM Communication System.

3.1 Messaging

The messages communicated via the ACS fall into two major groups – heartbeat and regular messages. The heartbeat messages are used for self-monitoring, and they provide a summary of the component state. The RCs send proactively and regularly their state to the associated AGM, and the AGMs send their state to the GM. The regular messages are the AS-TRM work and configuration messages. In addition, each AS-TRM message has a priority, recognizable by the transport agents. This helps messages with higher priority to be delivered first. From the functional perspective, the ACS addresses:

- asynchronous broadcasting;
- canceling messages;
- asynchronous and synchronous sending and receiving heartbeat and regular messages;
- asynchronous sending and receiving regular messages to and by a specified node.

3.2 Autonomic Features

The AS-TRM Communication System extends the DMF (Vassev, 2005) by implying some autonomic computing features like self-protection, self-optimization and self-configuration (see section 2). Some of the autonomic computing features are addressed by the DMF architecture (Vassev, 2005). The core components – MS and TAs, work in autonomous and independent mode. Hence, the ACS inherently consists of autonomous elements. In order to make the ACS' components autonomic, we extend their autonomous architecture by adding to them a management unit (see Fig. 7). The management unit (MU) controls and monitors the associated ACS' unit. Hence, each ACS' component is a peer of autonomous units – a management unit and ACS work unit (WU). The MU performs control functions over the WU, which simply performs its work duty and reports proactively its state to the MU. The last can decide to shut down and/or restart the former if there is no state report received for an efficient amount of time.

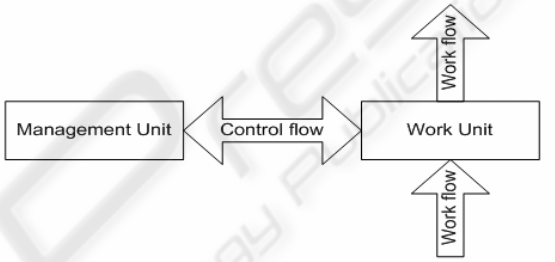

Figure 7: ACS' Components Architecture.

The ACS' autonomic features are:

Self-protection. Only communication-trusted end-points are able to communicate via ACS. The ACS exposes an integrated security mechanism that prevents unauthorized access. This autonomic feature is inherited from the DMF architecture.

Self-configuration. The ACS is a distributed system with hot-plugging (Vassev, 2005) features. For example, the TAs are able to discover available MS and plug into the ACS. This autonomic feature is inherited from the DMF architecture.

Self-healing. The ACS' components can be restarted by the embedded management unit. The ACS addresses at least one delivery semantics (Vassev, 2005), which prevents messages sent asynchronously to be lost. That allows the restarted component to continue from the point it stopped.

4 RELATED WORK

An autonomic system may contain many autonomic components that communicate and negotiate with each other and other types of resources within or outside of system boundaries. This is referred to as autonomic manager collaboration. The architectural concepts for autonomic systems are mainly based on

IBM Corporation's blueprints and on the on-going research for autonomic computing conducted at the IBM's laboratories (IBM, 2003, IBM 2004, IBM, 2005). The three blueprints are overviews of the basic concepts, constructs and behaviors for building autonomic capability into computer systems. The autonomic component architecture relies on the technique of feedback control optimization based on forecasting models, which technique facilitates the self-management features of an autonomic system.

In (Yuan-Shun, 2005), a new model-driven scheme for autonomic management is presented. This scheme can better allocate resources by using reliability models to predict and direct the distribution of monitoring efforts. If certain services or components are predicted to have high reliability at a particular time, then there is no need for intensive monitoring during that period, but those with low reliability require intensive monitoring.

Our research considerably differs from the related work in this area for the reason that we target the modeling of both reactiveness and autonomicity in distributed systems.

5 CONCLUSIONS AND FUTURE WORK

The research work reported in this paper is our first step towards developing a formal framework for developing distributed reactive autonomic components along with their relationships, and the qualitative properties such as reliability and safety constraining the behavior of the system. Particularly, it addresses: 1) the extension of the existing TROM formalism for modeling real-time reactive systems to AS-TRM formalism for supporting autonomic behavior; 2) the characteristics of AS-TRM for determining the requirement specification, design, and implementation of AS-TRM; and 3) the architecture and communication mechanism of AS-TRM for implementing autonomic as well as real-time reactive functionalities. This paper describes only the architecture aspects of AS-TRM, i.e. it does not deal with load balancing and efficiency aspects, as these are part of our future work on AS-TRM.

One of the most important aspects of autonomic systems is their self-management – a feature requiring formal mechanism for self-diagnosis of the AS-TRM system's quality status. The evolving nature of the AS-TRM requires continuous monitoring of the quality levels to evaluate the risk of deploying a change on the configuration of the AS-TRM system, and to diagnose potential safety hazards in AS functionality. We are investigating means for achieving continuous quality assessment of the evolving AS-TRM. We intend to develop and analyze algorithms and negotiation protocols for conflicting quality requirements, and determine what bidding or negotiation algorithms are most effective.

REFERENCES

Achuthan, R., 1995. A Formal Model for Object-Oriented Development of Real-Time Reactive Systems. *Ph.D. Thesis*, Department of Computer Science, Concordia University, Montreal, Canada.

Bantz, D.F. et al, 2003. Autonomic Personal Computing. *IBM Systems Journal*, Vol 42, No 1, pp. 165-176.

Emmerich, W., 2000. *Engineering Distributed Objects*. Baffins Lane, Chichester, Wiley.

Ganeck, A.G., Corbi, T.A., 2003. The Dawning of the autonomic computing era. *IBM Systems Journal,* Vol 42, No 1, pp.5-18.

Horn, P., 2001. Autonomic Computing: IBM's Perspective on the State of Information Technology. *IBM Manifesto.*

IBM, 2003. An architectural blueprint for autonomic computing.

IBM 2004. An architectural blueprint for autonomic computing.

IBM 2005. An architectural blueprint for autonomic computing.

Kephart, J.O., Chess, D.M., 2003. The Vision of Autonomic Computing. *IEEE Computer*, pp.41-50.

McCann, J. A., Huebscher, M.C., 2004. Evaluation issues in Autonomic Computing. *Proceedings of the International Workshop on Agents and Autonomic Computing and Grid Enabled Virtual Organizations (AAC-GEVO'04),* pp.21-24.

Vassev E., 2005. General Architecture for Demand Migration in the GIPSY Demand-Driven Execution Engine. *Masters Thesis*, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada.

Yuan-Shun Dai, 2005. Autonomic Computing and Reliability Improvement. *Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pp. 204-206.