

Multi-Agent Autonomic Architectures for Quality Control Systems

Gilda Pour

College of Engineering
San Jose State University
San Jose, CA, U.S.A.

Abstract - *Quality control is crucial to the success of any business and organization. Due to the ever-increasing complexity and mission-critical nature of quality control, development, deployment, and management of information systems for quality control overwhelms the capabilities of software developers and system administrators. This has motivated our research, co-sponsored by IBM and HP, to explore development of multi-agent autonomic architecture for quality control systems. We have designed a new multi-agent autonomic architecture for quality control systems. This paper presents the new multi-agent autonomic architecture, and discusses our latest research results and our plans for enhancement of this research project. Background information is also provided in this paper.*

Keywords: Autonomic architectures, multi-agent architecture, quality control.

1 Introduction and Motivation

Quality control is a necessary and crucial process in any business and organization. It is a set of operational techniques and activities that are required to fulfill requirements for quality. It involves techniques that monitor and analyze the managed and eliminate causes of unsatisfactory performance at all stages of the quality loop.

Due to the ever-increasing complexity and mission-critical nature of quality control, development, deployment, and management of information systems for quality control overwhelms the capabilities of software developers and system administrators.

Autonomic computing is emerging as a viable approach to address the issue of ever-increasing complexity in development, deployment and management of computing systems [1-5]. Autonomic computing aims at creating next-generation computing systems with the ability to self-manage. Self-management requires the ability to self-configure, self-optimize, self-protect, and self-heal. Autonomic computing is in its infancy. Development of autonomic architectures remains an open research issue.

This has motivated our research, co-sponsored by IBM and HP, to explore development of multi-agent autonomic architecture for quality control systems. The main goal is to develop autonomic architectures to allow complex computing systems to monitor managed resources, analyze the measured data, and use the information to develop and execute plans for managing the system and resources.

In this project, we have developed a multi-agent approach to designing autonomic architecture, and design multi-agent autonomic architecture for quality control systems using this multi-agent approach.

The remainder of this paper is organized as follows. Section 2 discusses the multi-agent approach for autonomic system development. Section 3 presents our multi-agent autonomic architecture for quality control systems. It also discusses prototype development. Section 4 provides concluding remarks and our plans for enhancement of this research project.

2 Multi-Agent Autonomic Systems

Multi-agent development has emerged as a viable approach to meet the autonomic system requirements--autonomy, adaptability, intelligence, goal-oriented interaction, collaboration, and flexibility. Using multi-agent approach, real-world problems can be modeled in the form of autonomous, interacting agent components. Agent components (a.k.a. agents) refer to next-generation software components--specialized software components exhibiting a combination of several of the characteristics, such as autonomy, adaptability, context-awareness, collaboration, intelligence, persistence, and mobility [6].

Multi-agent approach supports a decentralized model that does not suffer from shortcomings of centralized systems, such as resource limitations, critical failures, and performance bottlenecks. It also allows the interactions and interoperation of non-agent-based systems (e.g. legacy systems) with multi-agent systems through special agent proxies. It enhances the overall system performance, particularly in terms of adaptability, reusability, reliability,

extensibility, maintainability, flexibility, robustness, responsiveness, and computational efficiency [6][9][10].

3 The Multi-Agent Architecture for Autonomic Quality Control Systems

We have designed autonomic architecture for quality control systems as a layered architecture. Figure 1 shows our high-level design of intelligent control loop for autonomic quality control systems. At the core of an autonomic system is an intelligent control loop.

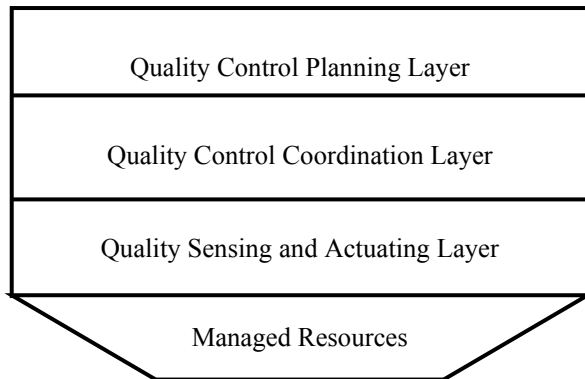


Figure 1. Design of Intelligent Control Loop for Autonomic Quality Control Systems

The key element of an autonomic system is an intelligent control loop, which enables the system components to communicate and collaborate with one another, and also enables high-level management tools to regulate the system and mask inherent system complexities from users [1-3]. Using high-level policies, the intelligent control loop in an autonomic computing system performs the following four main functions:

- Quality Monitoring
- Quality Analyzing
- Quality Planning
- Quality Plan-Executing

Figure 2 shows our design of the architecture of a subsystem in the intelligent control loop of an autonomic computing system. As shown in Figure 2, a subsystem includes different sets of agent components, such as information receiver agents, knowledge receiver agents, processor agents, and reporter agents.

Figure 3 illustrates our multi-agent architecture design for key subsystems of autonomic systems. The emphasis is on the control loop design. Each layer is designed as a set of subsystems. Each subsystem is designed as a set of collaborating agent components with well-defined goals as well as skills and competence to perform a set of specific tasks.

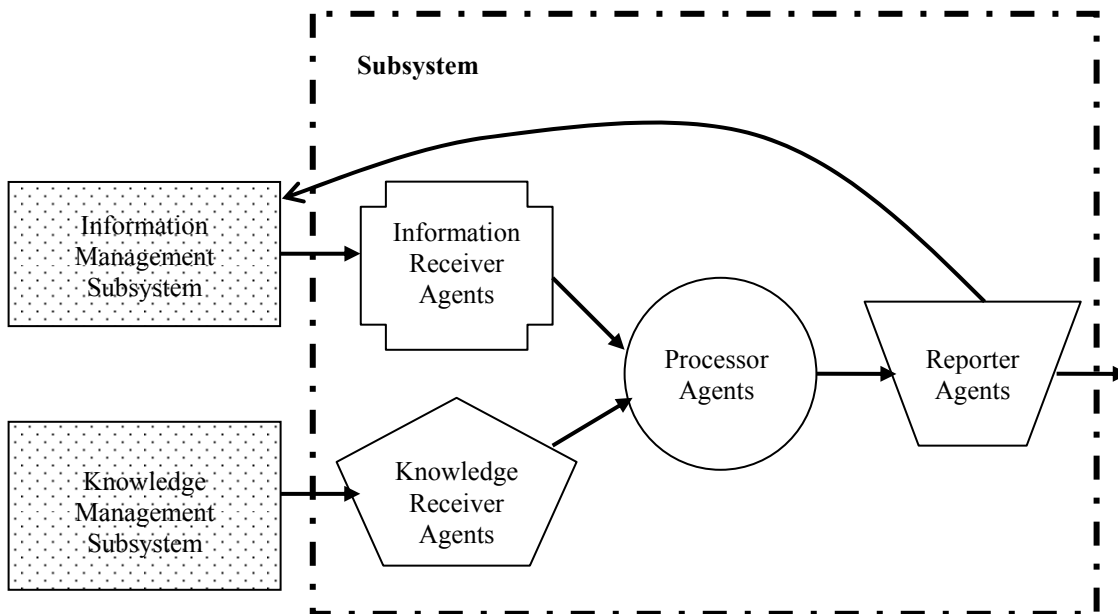


Figure 2: The Architecture of a Subsystem in the Intelligent Control Loop of an Autonomic System

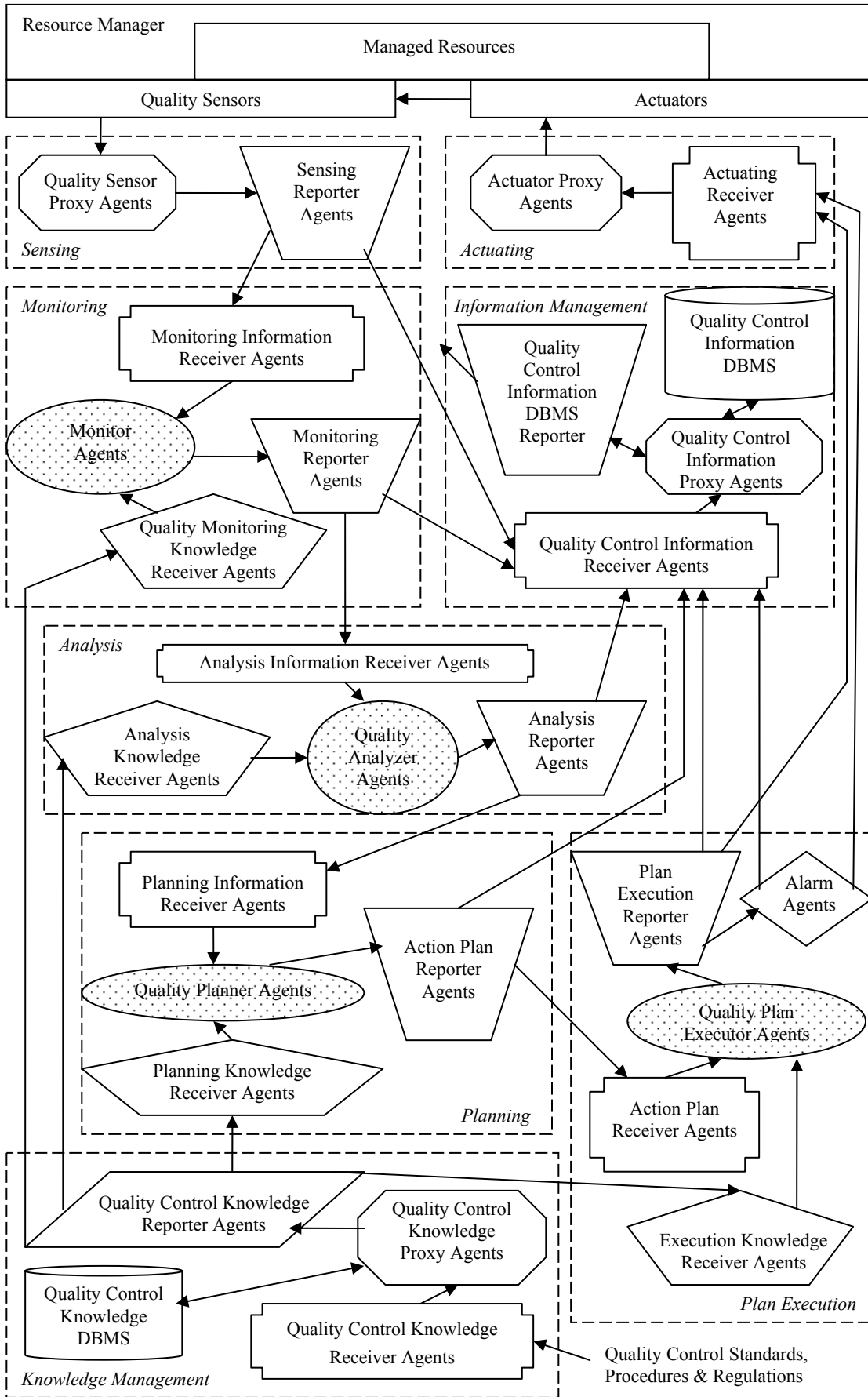


Figure 3. Design of Multi-Agent Autonomic Architecture for Quality Control Systems

3.1 Quality Sensing and Actuating Layer

Quality Sensing and Actuating layer (Layer 1) comprises two key subsystems:

- Quality Sensing Subsystem
- Quality Actuating Subsystem.

Quality sensing and actuating layer includes reactive agents that operate locally. These reactive agents are autonomous and suitable for performing fast control and pre-programmed self-management actions requiring immediate actions.

As Figure 3 illustrates, Quality Sensing subsystem consists of quality sensors, quality sensor proxy agents, and sensing reporter agents. Quality sensors collect and report measurements of interest. A quality sensor is represented by a quality sensor proxy agent in an autonomic system. Sensing reporter agents manage the interactions of quality sensing subsystem with other subsystems in the autonomic system.

As Figure 3 illustrates, Quality Actuating subsystem consists of actuators, actuator proxy agents, and actuating receiver agents. Actuators are mechanisms or commands that autonomic systems use to change or control the environment. An actuator is represented by an actuator proxy agent in an autonomic system. Actuating receiver agents manage the interactions of quality sensing subsystem with other subsystems in the autonomic system.

3.2 Quality Control Coordination Layer

Quality control coordination layer (Layer 2) is responsible for ensuring consistency in the system. Agents in quality control coordination layer have heuristic knowledge. They receive measured values and alarms from quality sensing and actuating layer; identify those that are urgent, critical, or resource-intensive, and require report triggers events to quality control planning layer; update the system's model of real-world as needed; monitor, analyze, and check the plans/commands received from the quality control planning layer to ensure every quality plan is in accordance with the system's current status of real-world model, and if it does not match, triggers the quality control planning layer to modify the plan/command; develop a list of actual control signals based on the plan/command received from quality control planning layer if the plan/command is in accordance with the system's current status of real-world model, and communicate it with the appropriate components in layer 1; and ensure consistency between layers 1 and 3 of the autonomic system.

Quality control coordination layer includes subsystems for quality knowledge management, quality information management, quality monitoring, quality analysis, quality plan execution, environment model

update, event filtering, event identification, workload management, fault detection management, quality plan execution, resource discovery, operation management, and environment management.

As Figure 3 shows, Quality Information Management subsystem includes:

- Quality Information Database Management System (DBMS)
- Quality Information Proxy Agents
- Quality Information DBMS Receiver Agents
- Quality Information DBMS Reporter Agents

Quality information management subsystem is in charge of managing the quality information generated by other subsystems, and keeps a historical record of the quality information and states in the control quality system.

Quality information DBMS stores quality information received from reporter agents of other subsystems, such as quality sensing and actuating, monitoring, analysis, planning, or plan execution subsystems.

Quality information proxy agents represent quality information in the Quality Information DBMS. They handle requests intended for the associated information, and manage the interactions of DBMS with the other parts of the multi-agent autonomic system.

Quality Information DBMS receiver agents are in charge of receiving quality information generated by monitoring, analysis, planning and plan execution subsystems.

Quality information reporter agents are in charge of reporting to authorized components the quality information stored in Quality Information DBMS.

As Figure 3 shows, Quality Knowledge Management subsystem includes:

- Quality Knowledge Database Management System (DBMS)
- Quality Knowledge Proxy Agents
- Quality Knowledge Receiver Agents
- Quality Knowledge Reporter Agents

Quality Knowledge DBMS stores quality standards, procedures and regulations (a.k.a. knowledge in this paper). A quality knowledge proxy agent represents a standard, procedure or regulation in Quality Knowledge DBMS. It also handles requests intended for the associated quality standard, procedure or regulation, and manages its interactions with other parts of the autonomic system.

Quality knowledge receiver agents obtain new standards, procedures and regulations from appropriate sources outside of the autonomic system, and send them to Quality Knowledge DBMS through quality knowledge proxy agents.

Quality knowledge reporter agents send the requested standards, procedures, and regulations to quality knowledge receiver agents in other subsystems (e.g. monitoring, analysis, planning, and plan execution).

As Figure 3 shows, Quality Monitoring subsystem includes:

- Quality Monitoring Knowledge Receiver Agents
- Quality Monitoring Information Receiver Agents
- Quality Monitor Agents
- Quality Monitoring Reporter Agents

Quality monitoring knowledge receiver agents obtain quality monitoring standards, procedures and regulations through quality knowledge reporter agents, and then package and send them to quality monitoring agents.

Quality monitoring information receiver agents obtain measured values through quality sensing reporter agents, and then package and send them to quality monitoring agents.

Quality monitor agents use quality monitoring standards, procedures and regulations to vigilantly monitor the autonomic system to ensure the measured quality values received by quality monitoring subsystem are valid (e.g. not corrupted), and then package and send verified information to quality monitoring reporter agents.

Quality monitoring reporter agents package and send the verified quality information to quality information DBMS receiver agents as well as quality analysis information receiver agents.

As Figure 3 shows, Quality Analysis subsystem includes:

- Quality Analysis Knowledge Receiver Agents
- Quality Analysis Information Receiver Agents
- Quality Analyzer Agents
- Quality Analysis Reporter Agents

Quality control analysis knowledge receiver agents obtain analysis standards, procedures and regulations from quality knowledge reporter agents; then package and send them to quality analyzer agents.

Quality analysis information receiver agents obtain information from monitoring reporter agents, and then package and send the quality information to quality analyzer agents.

Quality analyzer agents analyze the received information in accordance to the received quality analysis standards, procedures and regulations; and then package and send quality analysis results (including the severity of risks associated with a detected problem) to quality analysis reporter agents.

Quality analysis reporter agents package and send the analysis results to quality information DBMS receiver agents as well as quality planning information receiver agents in layer 3.

As Figure 3 shows, Plan Execution subsystem includes:

- Quality Control Execution Knowledge Receiver Agents
- Quality Control Action Plan Receiver Agents
- Quality Control Plan-Executor Agents
- Quality Control Action Reporter Agents
- Quality Alarm Agents

Quality execution knowledge receiver agents obtain quality plan execution standards, procedures and regulations through quality knowledge reporter agents, and then package and send them to quality plan executor agents.

Quality action plan receiver agents obtain quality action plans generated and reported by the quality planning subsystem located in Layer 3, and then package and send them to quality plan executor agents.

Quality plan executor agents manage the execution of the quality action plans using the associated quality plan execution standards, procedures and regulations.

Quality control action reporter agents generate required reports for appropriate control authorities and/or systems, actuators, quality alarm agents (when needed), and quality information DBMS receiver agents. The system reconfiguration is carried out through actuators.

Quality alarm agents generate time-stamped alarm messages to notify appropriate authorities, systems and actuators depending upon the severity of the problem detected by the system. Quality alarm agents also notify quality information DBMS receiver agents of the alarm activation to ensure the quality alarm is recorded in the quality information DBMS.

3.3 Quality Control Planning Layer

Quality control planning layer (Layer 3) consists of executive subsystems in charge of autonomic planning. The executive subsystems are intended to collaborate in planning to address key issues, such as robustness, dependability, self-healing, adaptability, security.

Quality control planning layer includes quality planning subsystem and a set of other executive subsystems in charge of major activities such as risk management, security management, (re)configuration planning, and reinstallation planning. In quality control planning layer, cognitive agents collaborate to make executive decisions and quality plans, and issue commands at the system level.

As Figure 3 shows, Quality Planning subsystem includes:

- Quality Planning Knowledge Receiver agents
- Quality Planning Information Receiver Agents
- Quality Planner Agents,
- Quality Action Plan Reporter Agents.

Quality planning knowledge receiver agents obtain quality planning standards, procedures and regulations from knowledge reporter agents, and then package and sent them to quality planning agents.

Quality planning information receiver agents obtain quality planning information from quality analysis reporter agents in layer 2, and then package and send them to quality planning agents.

Quality planner agents develop plans using the received planning information, standards, procedures and regulations, and then package. They also send quality action plans to quality action plan reporter agents.

Quality action plan reporter agents package and send actions quality plans to quality information DBMS receiver agents as well as quality action plan receiver agents.

3.4 Other Subsystems in Autonomic Systems

Different sets of agent components called utility agents support utility functions required in an autonomic system. The list of utility agents include digital signature for verification agents. These agents are in charge of verifying digital signatures for authentication purpose.

A series of specialized brokers and recommenders, and service agents (e.g. name service) are also included in the autonomic architecture to support federated yellow page registry and dynamic agent creation and discovery.

3.5 Prototype Development

We have designed agents as abstract agent components with a set of variation points to extend the benefits of agent components to the entire software; in particular, to significantly enhance the flexibility, adaptability, and extensibility of the enterprise distributed system [6-8].

For prototype development, we have chosen widely-accepted open source technologies such as Java Agent Development Framework (JADE) agent toolkit [9] [10], XML schema (<http://www.w3.org/XML/Schema>), SOAP (<http://www.w3.org/TR/soap/>), and Java-based application server known as JBoss (<http://www.jboss.org>).

Agent components communicate via asynchronous message passing, using a loosely-formatted, text-based communication language. Agent communication languages allow agents to exchange information required for their interactions, such as their collaboration, cooperation, and competition. A message includes a header, such as “from” and “to”, a message type, an expected message protocol, a variable-length body, and ontology.

FIPA (<http://www.fipa.org/>) specifies a number of high-level message types such as “Query”, “Inform”, and “Notify.” FIPA/JADE AMS (white pages) and Facilitator agents (yellow pages) are used for registration and lookup in the system. The capabilities to register dynamically with Facilitator agents allow new agents with new functions to be launched dynamically and incrementally in this system. We have design a new QualityControl ontology and QualityControlOperation language. This is currently being enhanced to support agent interactions in the multi-agent systems for quality control.

4 Concluding Remarks

We have developed multi-agent approach for autonomic architecture design, and used the approach to develop multi-agent autonomic architecture for quality control systems. Adopting agent-oriented software engineering approach for autonomic system development, and also representing sensors, actuators, and both the knowledge and information database management systems by proxy agent components have increased the degree of system adaptability. The multi-agent autonomic architecture can dynamically adapt itself not only to the changes in the number and types of sensors and quality information and knowledge database management systems, but also to the changes in the standards, procedures and regulations for all different phases of quality control process (e.g. quality monitoring, analysis, planning, and plan execution).

The results of this research serve as the foundation for further research to develop multi-agent autonomic architectures, particularly for quality control in different application domains.

Other key issues that need to be addressed in development of autonomic architectures include system consistency in dealing with various models, measurements obtained through various sensors, actions to be taken by

various actuators, and various goals, policies, decisions, and metrics throughout the system.

We are in the process of extending our research to incorporate mobility into the autonomic quality control system. We plan to use mobile agents, and enhance the demonstration prototype using Java-based mobile agent technologies such as JADE-LEAP, J2ME, CLDC, and MIDP. This approach is based on coupling Java 2 Platform, Micro Edition (J2ME) technologies, such as Connected Limited Device Configuration (CLDC) (<http://java.sun.com/products/cldc/>) and the Mobile Information Device Profile (MIDP) (<http://java.sun.com/products/midp/>) in the front-tier. J2ME (<http://java.sun.com/j2me/>) provides an application environment that specifically addresses the needs of mobile phones, pagers, personal digital assistants (PDAs). Lightweight Extensible Agent Platform (LEAP) is combined with JADE to allow agents run on lightweight devices [11]-[14]. CLDC defines the base set of application programming interfaces and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream PDAs. CLDC coupled with MIDP provides a solid Java platform for developing applications for mobile and wireless devices with limited memory, processing power, and graphical capabilities (e.g. cell phones and PDAs) [11]-[14].

5 Acknowledgements

The author gratefully acknowledges the co-sponsorship of IBM and Hewlett-Packard Company.

6 References

- [1] IBM Autonomic Computing White Paper, "Automating Problem Determination: First Step Toward Self-Healing Computing Systems," 2003. http://www-306.ibm.com/autonomic/pdfs/Problem_Determination_WP_Final_100703.pdf
- [2] J. Kephart and D. Chess, "The Vision of Autonomic Computing," IEEE Computer, Vol. 36, No. 1, pp. 41-50, 2003.
- [3] G. Pour, "Exploring Multi-Agent Architectures for Autonomic Systems," Pervasive Computing, Vol. 2. No. 1, 2005.
- [4] M. Waldrop, "Autonomic Computing: The Technology of Self-Management," 2003. <http://www.thefutureofcomputing.org/Autonom2.pdf>
- [5] S. Cheng, A. Huang, D. Garlan, B. Schmerl, and P. Steenkiste, "An Architecture for Coordinating Multiple Self-Managing Systems," IEEE/IFIP Conference on Software Architecture, 2004.
- [6] M. Griss and G. Pour, "Accelerating Development with Agent Components," Cover feature article, IEEE Computer, Vol. 34, No. 5, 2001.
- [7] G. Pour, "Web-Based Multi-Agent Architecture for Distributed Collaborative Software Engineering," Applied Computing, Vol. 1, No. 3, 2004.
- [8] G. Pour, "Internet-Based Multi-Agent Architecture for Service Delivery," Internet Computing, Vol. 1, No. 5, 2004.
- [9] Bellifemine, F. et al., "JADE: A FIPA-Compliant Agent Framework," Proceedings of Practical Applications of Intelligent Agents and Multi-Agents. pp. 97-108, 1999.
- [10] D. Cowan, M. Griss, and B. Burg, "BlueJADE – A Service for Managing Software Agents," Hewlett-Packard Technical Report HPL-2001-296, 2001.
- [11] G. Pour and N. Laad, "Mobile Agent Architecture for Mobile Systems for Telehealth and Telemedicine," Telehealth Series, To appear, 2006.
- [12] G. Pour, "Exploring Mobile-Agent-Based Architectures for M-Commerce Applications," E-Commerce Series, Vol. 1, No. 2, 2004.
- [13] G. Pour, F.C. Yao and C.C. Yu, "A Mobile Agent-Based Architecture for Mobile Systems Supporting Distributed Software Project Management," IEEE Publication series on Software, Telecommunications & Networks, Vol. 5, No. 1, 2004.
- [14] G. Pour and N. Laad, "Enhancing the Horizons of Mobile Computing with Mobile Agent Components," IEEE Computer Society Press, To appear, 2006.