

Autonomic Computing Technologies for Cyber-Physical Systems

Ingeol Chun*, Jeongmin Park**, Wontae Kim*, Woochun Kang*, Haeyoung Lee*, Seungmin Park*

**Embedded SW Research Dept., S/W & Content Research Lab., Electronics and Telecommunications Research Institute (ETRI),
138 Gajeongno, Yuseong-gu, Deajon, 305-700, KOREA*

*** Dept. of Software Engineering, Dongyang Technical College
62-160, Kochuk-dong, Seoul, 152-714, KOREA*

igchun@etri.re.kr, jmpark@ece.skku.ac.kr, {wtkim, wchkang, haelee, minpark}@etri.re.kr

Abstract— Cyber-physical systems (CPS) are characterized by extremely tight integration of and coordination between computational and physical resources. Embedded computers and networks in the cyber world monitor and control the physical world. But it is impossible to make systems suitable to all situations because the physical world has much uncertainty and uncontrollable condition. The interactive mechanism between the cyber world and the physical world is a key domain for innovation and a major challenge to building Cyber-Physical systems. As the most application fields of CPS are the national infrastructure such as Smart Rescue System, Intelligent Building System, Smart Grid, Smart Highway and etc., all cyber-physical systems “must be dependable, secure, safe, efficient and operate in real-time. They must also be scalable, cost-effective and adaptive.” [1]

In this paper, we focus on the dependability of CPS to make high reliable systems.

Keywords—CPS, Autonomic Computing, Self-Healing System

I. INTRODUCTION

As embedded computer systems are embodied in mission critical systems and time critical systems such as traffic control and safety, automotive systems, medical device and systems, energy conservation, critical infrastructure control (ex. power transmission systems, water resources and communication systems), military systems, avionics and so on, embedded systems have always been held to a higher reliability and predictability standard than general-purpose computing systems. Especially, the physical world that embedded systems operate in is not entirely predictable. That is, embedded systems will not operating in a controlled environment so that it must be robust and reliable to unexpected conditions and adaptable to system failure.

To make robust and reliable systems, the first consideration is to understand the characteristics of the physical world. The physical world has much uncertainty and uncontrollable condition. But in general development

methodology, uncertainty and uncontrollable condition are not allowed. Moreover all of the user requirements and the system states are well-defined and described in requirement specification document. That makes it impossible to make systems suitable to all situations in the physical world. Hence new approach of the system design and the system development methodology are strongly needed.

In this paper, we focus on making robust and reliable system to deal with uncertainty and uncontrolled condition in the physical world.

II. WHAT IS CYBER PHYSICAL SYSTEMS?

Cyber-Physical Systems (CPS) is the integration systems between computation and physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa[2]. The interactive mechanism between the cyber world - that includes embedded systems, network, sensors actuators and etc. - and the physical world is a key domain for innovation and a major challenge to building Cyber-Physical systems.

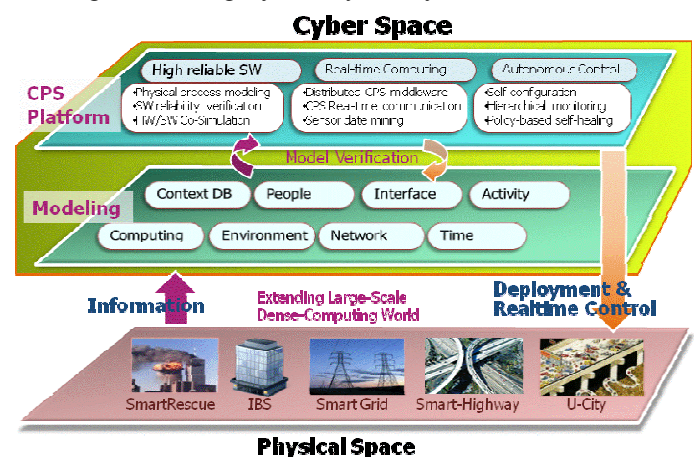


Figure 1. Architecture of Cyber-Physical Systems

We depict the architecture of Cyber-Physical Systems in Figure 1. As the most application fields of CPS are the national infrastructure such as Smart Rescue System, Intelligent Building System, Smart Grid, Smart Highway and etc., all cyber-physical systems “must be dependable, secure, safe, efficient and operate in real-time. They must also be scalable, cost-effective and adaptive.” [1]

We classified major characteristics of the CPS as below.

- High reliability : users do not expect their vehicles, medical device and critical infrastructure to crash or reboot. They hope that their systems must be robust and reliable.
- Real-time : the time critical system like cyber physical systems interacts with each other to achieve a particular application objective within a specific time frame. Users want their systems to operate in response to input on time.
- Autonomics : embedded systems are operating under the controlled environment. If a problem occurs, it will be fixed using predefined problem resolution. But cyber physical systems are working in uncertainty and uncontrollable condition so that systems don't know what problem happens. If the systems are faced with unexpected problems, it stops in a panic. To avoid crash, CPS is to be adaptable to unexpected condition. Autonomics would enable CPS to achieve the required objectives by policy-based decision making.

In the remainder of this paper, we focus on autonomic computing technologies for making CPS.

III. RELATED WORKS

Autonomic Computing(AC) is an approach to address the complexity and evolution problems in software system. A software system that operates on its own or with a minimum of human interference according to a set of rules is called autonomic. The term autonomic derives from human body's autonomic nervous system, which controls key functions without conscious awareness or involvement[3]. Autonomic Computing is not a new research area but deals with software agent, fault-tolerant computing, distributed and real-time computing, artificial intelligence, control theory and etc.

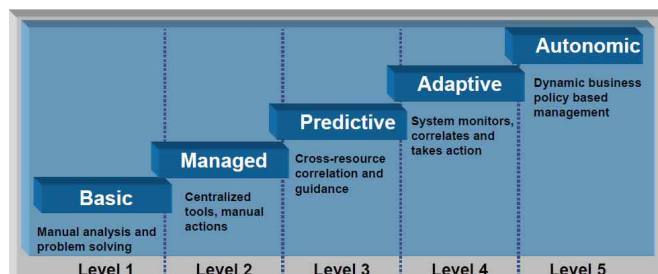


Figure 2. Autonomic, Policy-driven System Level of IBM

Many researches about autonomic computing are progressing. IBM started the autonomic computing initiative in 2001 to build self-managing computing systems to overcome the rapidly growing complexity problem. IBM defined the essence of autonomic computing systems as self-management that consists of four aspects as they are now and would be with autonomic computing[7].

- Self-configuration : autonomic systems will configure themselves automatically in accordance with high-level policies - representing business-level objectives, for example - that specify what is desired, not how it is to be accomplished.
- Self-healing : autonomic systems will detect, diagnose, and repair localized problems resulting from bugs or failures in software and hardware.
- Self-optimization : autonomic systems will continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient in performance or cost.
- Self-protecting : autonomic systems will protect themselves from malicious attacks and inadvertent cascading failures.

IBM applied the research result to more than 75 products such as DB2 Configuration Advisor, Tivoli Risk Manager, Smart planet, etc[4]. IBM defines autonomic, policy-driven system level as shown in Figure 2. Also many other companies including Microsoft, Hewlett Packard, Sun, Dell, Hitachi, Electronic Data Systems(EDS), etc. have started related researches.

The self-healing approach for the autonomic computing has been applied to the target system based on context-awareness, calm computing[5] and autonomic computing[6]. Basic strategies for self-healing adopt tactics such as restarting the system or replacing the faulty component. Previous studies show how systems can recover from an abnormal state to a normal state by having an awareness of the external resource environments(CPU, RAM, Bandwidth, etc). There are component-based self-healing frameworks that design additional modules for their ability to generating self-healing at the software design phase in order to reinforce reliability of the target system. In this section, we describe some studies about self-healing approaches and analyze their features and weaknesses.

A. Architecture-based Self-healing approaches

Architecture-based self-healing approaches [4, 7, 8] use the architectural model as a basis for system reconfiguration. They have a view of the external environment to monitor, analyze, and reconfigure the target system. A set of probes are deployed in the target system or physical environment and announce aspects of the actual system by means of a probe bus. The data are monitored by a gauge. Gauges translate and analyze data by searching logical events mapped to physical events using a structural model. The control layer monitors whether an action succeeded or not by logging good, suspicious, and bad events. Effectors interact with the target system to tune and restart the

system based on the adaptation plan. These proposed approaches model resource environments and processes based on an external view founded on ADL (Architecture Description Language). Architecture-based self-healing approaches for external environments have benefits and drawbacks as follows.

- It is sufficient to generate probes that monitor the target system based on an external architecture model. They need not to modify target system based on information collected by probes about the external resource environments.
- They support self-healing application development without internal information on target system. Some of external self-healing approaches are well attention.
- When developing self-healing software without internal information on target system, developers and analysts, they cannot but generate self-healing code not to affect the original system. This is performance based self-healing approach. Thus, it is difficult to determine the internal state of the component because the approach focuses on performance of the external environment. It is also difficult to develop a variety of self-healing strategies without internal information on target system.

B. Component-based self-healing

Component-based self-healing [9, 10] focuses on the internal status of software by adding the capability that supports self-healing at the design phase to make software. The service layers and the healing layers in the systems are designed separately. This approach models the state of normal actions of the service layer in a system. If an anomaly is detected, the service layer will restart, initialize, and reconfigure the system by using blocking and freezing. Component-based self-healing systems have advantages and disadvantages as listed below.

- They are able to reconfigure components in the service layer in component-based architecture through the state model that model behaviors of tasks in the service layer. Also, after reconfiguration, it provides a standard to judge whether or not the component can perform operations normally
- However, as these focused on events related to internal states, they cannot determine internal status problems due to lack of resource. Hence, it cannot heal resource anomalies.

C. Log-based self-healing

Log-based self-healing approaches [11, 12] suggest a self-healing of the system by auto-correcting event logs generated from various types of software (webserver, database system, etc.). Log-based self-healing systems analyze errors about the internal status of target system by converting a log event to a common base event. The logs are generated based on a common format and are analyzed by an autonomic manager. The healing system performs self-healing strategies when

encountering anomalies through a symptom service and policy engine. The benefits and drawbacks of this approach are listed below.

- It is useful to search for related problems based on log events provided by the system when an error event is logged. It also can easily restructure a database and be useful for bottom-up analysis because the contents of the log are recorded at runtime.
- Monitoring methods have difficulty to detect the source of the problem if the event has yet to be logged. Because it focuses on problems after the event, it is difficult to model constraints for monitoring, analyzing, and healing based on the internal view of system if the event is unlogged. Additional effort is needed to determine the level of problem determination in the target system. It is difficult to analyze whether the problem caused by one component influences another one from only the contents of the log.

A strict definition of reliability relates the system implementation to its specification. That is, the system is behaving reliably if its behaviour is consistent with that defined in the specification. System failures are inevitable, but the disruption caused by failure can be minimized if the system can be repaired. In order for that to happen, it must be possible to diagnose the problem, access the component that has failed and make changes to fix that component. However, Self-healing in software is enhanced when the organization using the system has access to source code and has the skills to make changes to it [13].

Unfortunately, this is becoming increasingly uncommon as we move towards system development using third-party, black-box components. Therefore, this paper assumes that the software developer and the self-healing software developer are different. The self-healing software developer has a high burden to analyze the target system as follows:

- It has burden to analyze the target system because it is necessary to analyze constraints for detecting the internal and external statuses of the target system.
- It has burden to analyze the dependence between components related to the abnormally functioning component.

The aim of this paper is to explore the autonomic computing technology and how to apply this technology to CPS as shown in Figure 3.

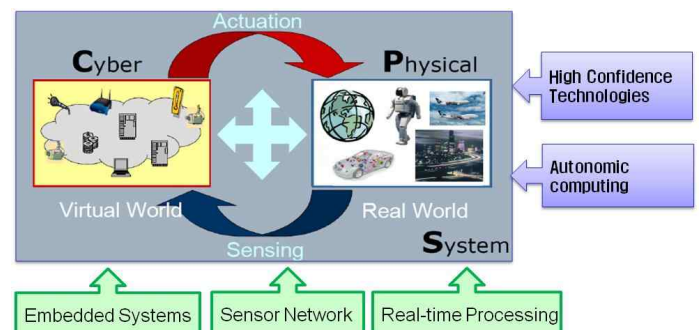


Figure 3. Autonomic Computing in CPS

IV. AUTONOMIC COMPUTING CYCLE OF CPS

As mentioned earlier, the software systems of CPS must be robust and reliable. We decided that autonomic computing technologies are the alternatives of providing robustness and reliability. To embody autonomous in software system, we propose Autonomic Computing Cycle of CPS (ACCC) as shown in Figure 4. The description of ACCC implies three aspects of self-management – self-configuration, self-healing, self-optimization, while self-protect is deal with in distributed CPS middleware.

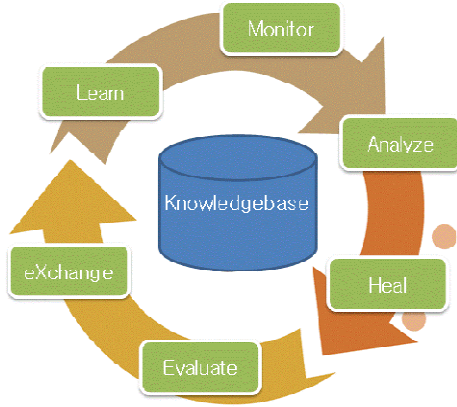


Figure 4. Autonomic Computing Cycle of CPS(ACCC)

ACCC is represented in closed-loop form consisting of six steps. The first step is to monitor various data from multiple sensors in systems. The monitoring system runs according to the rule defined in a knowledgebase (that supports decision making in other steps). It provides the statistical analysis result related to system performances such as CPU usage, memory allocation, processes information in execution, network latency, etc. and it examines whether any behavior inconsistency between a desired system defined in rules and the current system exists or not[14].

The second step is to analyze the types and the severity of the risk using monitored data. According to the result of the risk analysis, appropriate set of a problem report and a healing policy is selected. If any set doesn't match to this situation, new set is created in knowledgebase and will become complete during learning step.

Healing step is a problem resolution step. For removing inconsistency, the current system is repaired according to healing policy.

Evaluating step is to measure the ability of the healing system. The ability is defined as the performance ratio of the healed system that calculated as following equation.

$$PRH = \frac{\sum_{i=1}^n P(S_{hea}(i))}{\sum_{j=1}^t P(S_{ori}(j))} * 100$$

Where,

PRH is the performance ratio of a healed system.

P is the performance function.

S_{hea} is the service of a healed system.

S_{ori} is the service of an original system.

The original system performance is defined as various types such as response time, throughput, available function rate, network delay, etc. in healing policy.

Exchanging step is to exchange a healed system with an original system. The step is executed in the case that PRH is greater than the Performance Rate of the current system. PRC is calculated as follow.

$$PRC = \frac{\sum_{i=1}^n P(S_{cur}(i))}{\sum_{j=1}^t P(S_{ori}(j))} * 100$$

Where,

PRC is the performance ratio of a current system.

S_{cur} is the service of a current system.

Because all healed system do not have more ability than original systems, a healed system is not always substituted.

The final step is to learn a healing policy. If the Healing Success Ratio is lower than predefined threshold, healing policies are not suitable to the current system. In that case, learning procedures are activated to modify healing policy. HSR is calculated as follow.

$$HSR = \frac{NHE}{NOP} * 100$$

Where,

HSR is the healing success ratio.

NHE is the number of healed system exchange preformed.

NOP is the total number of healing operation performed.

For learning a healing policy, many learning algorithm is applied such as genetic algorithm, neural network, bayesian network, fuzzy theory, etc.

V. AUTONOMIC COMPUTING SUPERVISOR

To give autonomies in CPS objects, we propose Autonomic Computing Supervisor(ACS) that operates according to Autonomic Computing Cycle of CPS(ACCC) as shown in Figure 5. As ACS is implemented into each CPS objects, both each CPS object and the cluster of CPS objects can provide seamless service without user intervention.

ACS gathers massive distributed data from widespread sensors and monitors systems. Based on gathered data and knowledge in knowledgebase, it decides whether the system falls into abnormal condition or not then constructs and executes plans to achieve the management goals in accordance with the policies and rules in effect. Knowledgebase acts as a repository of knowledge such as the system status, the predefined abnormal condition and the error status, the healing policy and strategy, machine-learning procedure and

environmental information, which can be utilized by ACS. Knowledge can be obtained from multiple sources: from system modelling tool, peer systems and users, observation, past experience or can be obtained using machine-learning procedure.

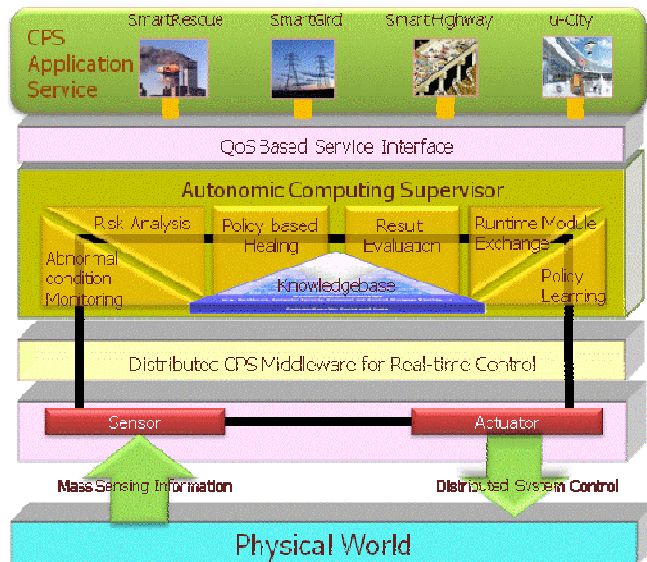


Figure 5. Autonomic Computing Supervisor in CPS Runtime Environment

Autonomic computing Supervisor acts in accordance with following six steps.

A. Abnormal Condition Monitoring

This is the first step of the Autonomic Computing Cycle of CPS. In this step, the Autonomic Computing Supervisor monitors the status of CPS objects based on the massive collected sensing data from widespread sensors. Data can be acquired by polling at specified time intervals or can be collected asynchronously when needed. The various types of the collected data such as sound, seismic, light, video, audio, temperature, system data (CPU, memory, process, etc.) are transformed to a single type and the data fusion is operated. ACS compares the transformed data with the fault model in the knowledgebase to detect an abnormal condition. If the abnormal condition occurs, the second step – Risk Analysis – is activated.

B. Risk Analysis

The Autonomic Computing Supervisor analyzes the types and the severity of the risk using fused data generated in the previous step. ACS has the ability to consult historical data and to compare them with current status to select a risk type. There will be various risks however the seamless service providing is the most important feature in the high reliable systems like CPS. ACS makes an inference from the risk model then selects appropriate healing policy in the knowledgebase.

C. Policy based Healing

Autonomic Computing Supervisor selects a predefined healing strategy from the healing policy. A healing policy has several healing strategies including the healing procedures and the healing method.

To minimize the service suspended time and prevent the false healing during healing operation, we propose DHX (Duplicate-Heal-eXchange) mechanism. ACS does not stop malfunctioned systems to repair directly but duplicates them and repairs duplicated systems for the seamless service providing. Even if malfunctioned components are out of order, it may provide a little service.

D. Result Evaluation

After healing operation, ACS evaluates the healing efficiency quantitatively using the goal model and the evaluation model in the knowledgebase. As healed components are not always suitable to all conditions, ACS recognizes more preferred systems between original systems and healed systems.

E. Runtime-Module eXchange

If the healed systems got a higher performance - that is, PRH is greater than PRC as described in previous section, ACS exchanges original systems into healed systems. On the other hands, ACS does not exchange the original systems.

F. Policy Learning

After healing, ACS calculates Healing Success Ratio as defined in previous section. If the HSR is lower than predefined threshold, machine-learning procedures are activated to modify healing policy. First of all, the learner determines which type of problem is. We classify the problems into 3 types - modeling mistake, inference error, and strategy mismatch.

- Modeling mistake : in the system modeling phase, users have no concern about specific situation so that risks or healing policies about that situation do not exists in the knowledgebase.
- Inference error : ACS makes wrong inference from risk analysis. It makes wrong healing policy selected.
- Strategy mismatch : in the system modeling phase, users model this abnormal condition but the healing strategy and the healing method used in the DHX mechanism is not match to current situation.

To solve the problems, the learner performs machine-learning procedures using such an appropriate algorithm as genetic algorithm, neural network, bayesian network, fuzzy theory, etc. It adds new risk or healing policy, updates the risk analysis algorithm or modifies the healing strategy and the healing method suitable to current situation for next time. According to this machine-learning algorithm, ACS may find best solution gradually.

VI. CONCLUSIONS

In this paper, we present the autonomic computing technologies for Cyber-Physical Systems which is a rising research challenges. To satisfy the robust and reliable characteristics of CPS, we decided to apply autonomic computing technologies. We present Autonomic Computing Cycle of CPS(ACCC) and Autonomic Computing Supervisor(ACS) to implement autonomic computing technology in CPS. The CPS objects that embody the proposed technology have capable of providing seamless service without user intervention.

At this time, the prototype of Autonomic Computing Supervisor is developed and we go on additional tests for stabilizing that. In the future work, we focus on improving the efficiency and the reliability of Autonomic Computing Technologies for CPS.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of NIPA, [Development of an Embedded Software Convergence Foundation]

REFERENCES

- [1] <http://varma.ece.cmu.edu/cps/CFP.htm>
- [2] Edward A. Lee, Cyber Physical Systems: Design Challenges, International Symposium on Object/Component/Service-Oriented Real-time Distributed Computing (ISORC), May 6, 2008, Orlando, FL, USA
- [3] IBM Research. Autonomic Computing, <http://www.research.ibm.com/autonomic>, 2002
- [4] David Garlan, Bradley Schmerl, "Model-based adaptation for self-healing systems," in Proceedings of the 1st Workshop on Self-Healing Systems, pp. 27-32, Nov., 2002.
- [5] Mark Weiser, "The Computer of 21st Century," Scientific American, 265(3), pp.94-104, Sep., 1991.
- [6] Jeffrey O.Kephart David M. Chess IBM Thomas J. Watson Research Center, "The Vision of Autonomic Computing," *IEEE Computer*, Vol.36, No.1, pp.41-50, Jan., 2003.
- [7] David Garlan, Shang-Wen Cheng, An-Cheng Huang, Bradley Schmerl, Peter-steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure," *IEEE Computer*, Vol.37, No.10, pp.46-54, Oct., 2004.
- [8] David S. Wile and Alexander Egyed, "An Externalized Infrastructure for Self-Healing Systems," in Proceedings of the 4th Working IEEE/IFIP Conference on Software Architecture, pp.285-288, Sep., 2004.
- [9] Michael E. Shin, "Self-healing components in robust software architecture for concurrent and distributed systems," *Science of Computer Programming*, Vol.57, No.1, pp.27-44, Jul., 2005.
- [10] Michael E. Shin, and Jung Hoon An "Self-Reconfiguration in Self-Healing Systems," in Proceedings of the 3th IEEE International Workshop on Engineering of Autonomic & Autonomic Systems, pp.89-98, Mar., 2006.
- [11] IBM, Autonomic Computing, "IBM's Perspective on the State of Information Technology," <http://www.ibm.com/industries/government/doc/content/resource/thought/278606109.html>.
- [12] Jeongmin Park, Giljong Yoo, Chulho Jeong, and Eunsoek Lee, "Self-Management System based on Self-healing Mechanism," LNCS 4238, pp.372-382, Sep., 2006.
- [13] Ian Sommerville, "Software Engineering 8th Edition", Addison-Wesley, 2007.
- [14] Nelly Delgado, Ann Q. Gates, and Steve Roach, Members, "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools", *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, pp. 859-870, Dec., 2004.