Self-Adaptive System Development Method for Smart Control Systems in CPS

In-geol Chun, Jeong-min Park, Won-tae Kim

Embedded SW Research Department, Electronics and Telecommunications Research Institute (ETRI)

218 Gajeongno, Yuseong-gu, Deajon, 305-700, Korea

igchun@etri.re.kr, jmpark23@etri.re.kr, wtkim@etri.re.kr

Abstract— **The human desire to develop high reactive, safety, precise and convenient control system makes smart machinery equipped with computing elements appear on the scene. Especially in order to get free of household affairs, the necessity of home service robot - that is a good example of control systems - has been increased. Home service robot is a robot used for household chores on behalf of human. We expect that every home will at least possess and use more than one home service robot in the near future. However the operating environment of home service robots - like house, yard, garage, and so on - has much uncertainty and uncontrollable conditions, so that it is impossible to make robots suitable to all situations. In this paper, to achieve user satisfaction and overcome abnormal situation, we propose a self-adaptive system development method that make home service robots dependable, secure, safe, and efficient, and operating in real-time. To apply this approach, home service robots could be more intelligent in the adaptation.**

Keywords—**Autonomic Computing, Home Service Robot, Intelligent system, Self-Adaptive System, Smart Control System**

I. Introduction

As the needs of the smart machinery are increased, numerous computing systems are embodied in legacy systems. Especially mission-critical systems and safety-critical systems like Cyber-Physical Systems(CPS) including robotic surgery, 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, have always been held to a higher reliability and predictability standard than general-purpose computing systems so that the huge necessity of intelligence arise. The operating environment for general-purpose computing systems is strict and fixed, while the physical world in which these control systems operate is not entirely predictable. That is, the control systems will not be operating in a controlled environment, so that they must be robust and reliable to unexpected conditions and adaptable to system failure.

In this paper, we propose a self-adaptive system development method to make a control system smart, dependable, secure, safe, and efficient, and operating in real-time. The self-adaptive system is a system that has some special capabilities to reason about its state and environment, and adapt itself at runtime automatically and dynamically in response to changes[1]. The first consideration to make a self-adaptive system is to understand the characteristics of the operational environment. Most of the operational environment has much uncertainty and uncontrollable conditions. But considering the general development method, uncertainty and uncontrollable conditions are not allowed[2]. Moreover, all of the user requirements and the system states must be well-defined and described in requirement specification documents. For that reason, we propose new systems development method based on the self-adaptation technologies. To prove the proposed approach, we try to develop the smart home service robot(S-HSR) based on the self-adaptive systems development method. S-HSR is used for moving burden, cleaning room, showing educational or entertainment videos, detecting fire alarms or monitoring an intruder. Also it can help sick and infirm people and provide helpful functions on behalf of human[3]. It moves around everywhere at home to achieve predefined missions. But the operational space of the robot like a house is not entirely predictable, so that some capabilities to overcome unpredictable condition are strongly needed.

The remainder of this paper is organized as follows. In section 2, we show the related work to look into prior researches and section 3 introduces the self-adaptive system development method. Finally in section 4, to prove the proposed approach, we make S-HSR smart and reliable to unexpected conditions and adaptable to abnormal situation. Then we show that the feasibility and efficiency of the proposed approach in case study.

II. Related Works

There are many definition of self-adaptive system but commonly used notion is a system that is able to evaluate and adjust their behaviour in respond to their perception of environment and the system itself[4]. The primary reason why we need self-adaptive device is the increasing cost of handling the complexity of software systems to achieve their goals. In past, software design is responsible for handling complexity and achieving quality goals. However, in recent years, there has been an increasing demand to handle these problems at runtime. Researchers in this self-adaptive system aim to incorporate adaptation mechanisms into software systems that adjust various artifacts or attributes in response to changes in the self and in the context of a software system[5]. In this section, we look into the related researches for fundamental studies. An architecture-based adaptation framework named Rainbow is proposed by Garlan[6]. The rainbow framework consists of an adaptation infrastructure and system-specific adaptation knowledge. The adaptation infrastructure implements self-adaptive capabilities to monitor, detect, decide, and act, based on the adaptation knowledge. Mukhija and Glinz propose a Contract-based Adaptive Software Architecture(CASA) framework that supports both application-level and low-level (for example, middleware) adaptation actions through an external adaptation engine[7]. Dynamic adaptation on the CASA framework is achieved via runtime reconfiguration of the components of an application, according to the adaptation policy specified in the application contract. DEAS propose a framework for identifying the objectives, and analyzing alternative ways of how these objectives can be met[8]. The DEAS framework defines the design method for the system that supports all or some of these alternative behaviors using goal models. MADAM proposes modelling tools and middleware that facilitate adaptive application development for mobile computing by representing architecture models at runtime[9]. A middleware of this project provides the dynamic adaptation of component-based applications, and a model-driven development methodology that is based on abstract adaptation models and corresponding model-to-code transformations. The Middleware(M-Ware) project addresses distributed applications subject to performance constraints when moving and operating on large data volumes[10]. The M-ware provides agility (adapting application components and then, dynamically deploying new components and change component structures), resource-awareness, runtime management and openness in distributed applications. Multi-Level Intrusion Detection System(ML-IDS) detects network attacks by inspecting and analyzing the traffic using an autonomic computing concept to automate control and management. This automation allows ML-IDS to detect network attacks and proactively protect the operating system against them.

III. Self-Adaptive System Development Method

A. Develop System Model

To develop a system model is the starting point to make self-adaptive software. We have selected ECML(ETRI CPS Modeling Language) as a system modeling language[11]. ECML is developed by ETRI(Electronics and Telecommunications Research Institute) to model a hybrid system that includes both continuous and discrete properties. First of all, developer designs a system model and defines the normal status, then specifies them into the system model. In order to specify a normal status and its condition, we defines the normal status constraint(NSC) in the extension of ECML. The NSC is inserted in a behavior model of ECML to specify the normal status and the normal condition as shown in Figure 1. Each state of CPS behavior model may have the NSC that describe as follows.

$$NSCname[NSCondition] \tag{1}$$

In equation (1), the NSC name is a unique name of the NSC and the NSC condition is the condition that describes normal status.

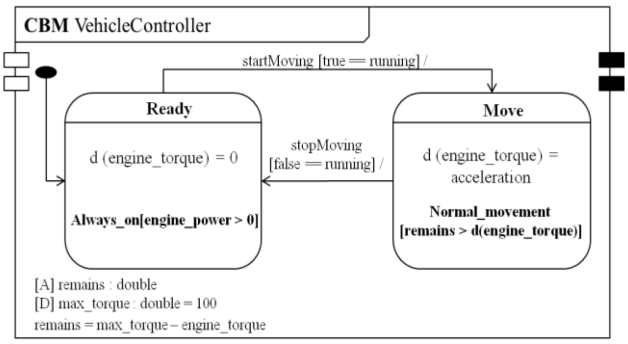
CBM VehicleController
startMoving [true == running] /
Ready
d (engine_torque) = 0
Always_on[engine_power > 0]
stopMoving [false == running] /
Move
d (engine_torque) = acceleration
Normal_movement [remains > d(engine_torque)]
[A] remains : double
[D] max_torque : double = 100
remains = max_torque – engine_torque

Figure 1. CPS Behavior Model of ECML

Next, the fault monitor(FM) is inserted in CPS structure model of ECML to specify system components where failures may occur as shown in Figure 2. Each structure of a structure model may have FM that describes the fault status.

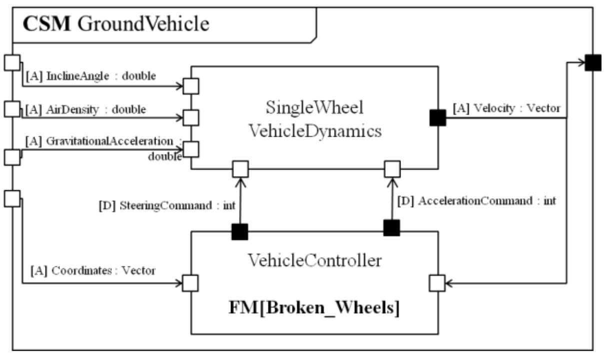
CSM GroundVehicle
[A] InclineAngle : double
[A] AirDensity : double
[A] GravitationalAcceleration : double
SingleWheel VehicleDynamics
[A] Velocity : Vector
[D] SteeringCommand : int
[D] AccelerationCommand : int
[A] Coordinates : Vector
VehicleController
FM[Broken_Wheels]

Figure 2. CPS Structure Model of ECML

B. Implement Self-Adaptive System

Generally internal and external adaptation approaches are used to implement a self-adaptive system. However, they are not suitable for S-HSR because of various user requirements, the physical size of a system and the mutual intercommunication between outer systems. For that reason, we used the hybrid adaptation approach to implement cooperative S-HSR[2]. The architecture of the hybrid adaptation approach consists of the adaptable software and the adaptation agent. The adaptable software consists of the main function and the SAManager thread that achieves the internal adaptation cooperating with the adaptation agent. The adaptation agent gathers massive distributed data from widespread sensors and the SAManager in adaptable software. Such 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, and system information (CPU, memory, process, etc.) are transformed into a single type through data fusion mechanism. If the self-adaptive software encounters an anomaly, it constructs and executes adaptation plans to achieve its objective in accordance with the policies and rules in effect. The knowledgebase acts as a repository including system status, the predefined normal status constraints (NSC) and conditions, faults, error status, the healing strategies, machine-learning procedures and environmental information. Knowledge can be obtained from multiple sources such as system models, peer systems and users, observation, past experience or can be obtained using the machine-learning procedure.

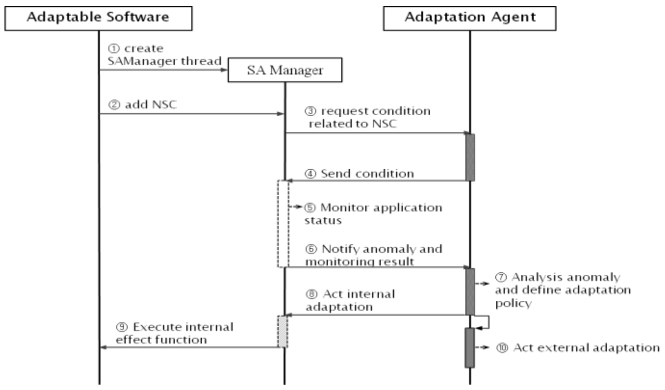
Adaptable Software
Adaptation Agent
① create SAManager thread
SA Manager
② add NSC
③ request condition related to NSC
④ Send condition
⑤ Monitor application status
⑥ Notify anomaly and monitoring result
⑦ Analysis anomaly and define adaptation policy
⑧ Act internal adaptation
⑨ Execute internal effect function
⑩ Act external adaptation

Figure 3. Sequence Diagram of Self-Adaptive System

The sequence diagram of self-adaptive system is shown in Figure 3. First of all, the adaptable software invokes the SAManager thread when it starts. The SAManager monitors the main function and executes the internal adaptation. The adaptable software adds NSC to the NSC_list of the SAManager, when the adaptable software meets “Add_NSC” command. Then the SAManager and the adaptation agent cooperate to monitor the status of the adaptable software and detect anomalies. If detected, the adaptation agent analyzes how anomaly occurs, then decides an adaptation policy and executes adaptation processes.

IV. Case Study of the Proposed Approach

In order to provide feasibility and effectiveness of proposed approach, we developed the prototype of S-HSR in Figure 4.

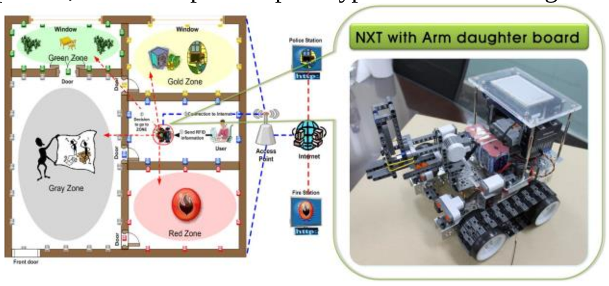

Figure 4. Operational Environment of Smart Home Service Robot

This prototype has many hardware and software components such as ultra-sonic sensor, color/light/sound sensor, compass sensor, step motor, web camera, service and monitoring software, and self-adaptation agent. As mentioned in our previous research[2], we collect customer requirements and desired functions firstly. Then we design the system model in which NSCs and FMs are specified and make the Adaptation Strategy Table(AST) according to the adaptation capability of a target system. Finally we generate the adaptation knowledgebase that includes the Normal Status Table(NST), the Fault Monitor Table(FMT) and the Adaptation Policy Table(APT) derived from the system model. The NST consists of category, NSC name, CBM name, CPM name, state name and condition to specify the normal status of a target system. The FMT consists of category, FM name, CSM name, substructure name, fault description to designate the potential structure of a target system. The APT is made based on the FMT and the AST for specifying an adaptation policy by developers. The AST has been defined according to the adaptation capability of a target system and consists of strategy name, action and type. The AST has two types of the adaptation strategy – internal and external type. Internal type is that adaptation processes are executed in the adaptable software. On the other hand, external type is that adaptation processes are executed by the adaptation agent outside of the adaptable software.

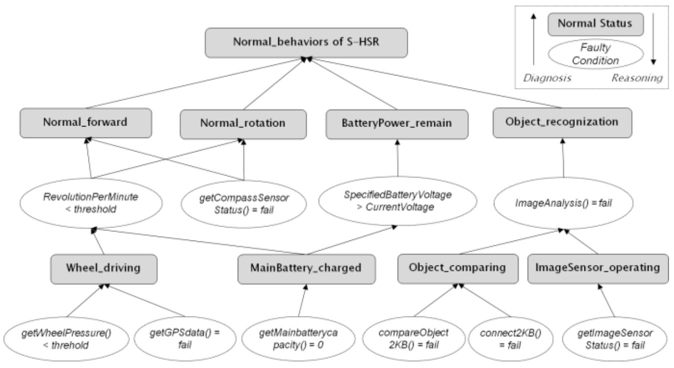

Figure 5. Example of Status-Fault Relation Graph for Smart Home Service Robot

After above processes, the adaptation knowledgebase is applied to the target system for the runtime adaptation. During the runtime, we propose the Status-Fault relation Graph(SFG) to analyze the current system status as indicated in Figure 5. SFG is generated by the adaptation agent dynamically and used to the failure analysis method that finds a faulty state of a system using Boolean logic to combine a series of lower-level events. SFG is composed of the normal status and the faulty condition. The normal status is the user desirable status of the system and the faulty condition is the criteria to judge the status of the system. For example, “Normal_forward” status is judged by the combination of the faulty condition “RevolutionPerMinute < threshold” and ”getCompassSensor Status() = fail”. In the same manner, the faulty condition “RevolutionPerMinute < threshold” is composed of the normal status “Wheel_driving” and “MainBattery_charged”.

This Status-Fault relation graph analysis method is generally used in the field of safety engineering to determine the probability of a safety hazard quantitatively.

In this case study, we show how the self-adaptive robot is adapted to unpredictable situation. We assume that S-HSR goes around satisfying customer requests in one’s house then meets a sudden-appeared obstacle. This robot is developed to operate in a pre-defined flat space that has no sudden obstacles and only has a few adaptive functions because of the lack of the memory capacity. The robot patrols and monitor the house for performing its objective (NST {NSC_name : Object_recognization, State_name : Vehicle Controller/Tarcking/SensingObject, Conditioin : Dectected _Ojbect = normal}) in the normal condition as shown in Figure 6. If the robot encounters any object – that is, the robot is in the abnormal status-, the adaptation agent of this robot tries to analyze the detected object and decides how to act against it. In the case of the pre-defined object such as people, wall, door, furniture and so on, the robot operates its own jobs. On the other hand - that is, the condition (NST {Condition : Detected_Object = normal}) is violated - the robot starts an analysis process to decide the type of the occurred fault using a heuristic algorithm and a neuro fuzzy algorithm based on granular-neuro-evolutionary computing (FMT {FM_name : UnknownObject, Strucutre _name : GroundVehicle/Vision_Sonsor, Fault_status : image Analysis() = fail}). Then referring to the APT, the robot builds the adaptation plan (APT{Policy_name : Upgrade SW, FM_name : UnknownObject, Strategy_name : SW Update}) and executes the adaptation strategy (AST {Strategy _name : SWUpdate, Action : Update current SW to new SW, Type : External}). Finally the robot applying new SW can, therefore, avoid or remove the obstacle.

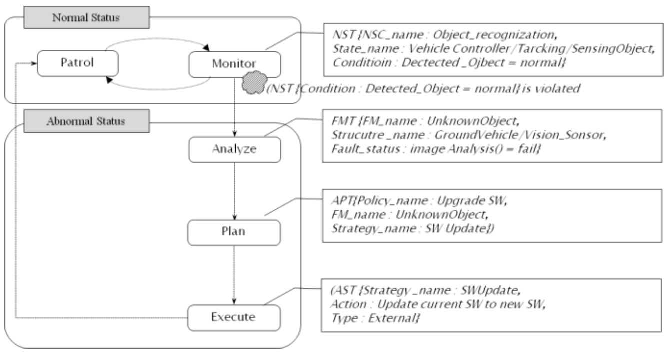
Normal Status
Patrol
Monitor
NST {NSC_name : Object_recognization, State_name : Vehicle Controller/Tarcking/SensingObject, Conditioin : Dectected _Ojbect = normal}
(NST {Condition : Detected_Object = normal} is violated
Abnormal Status
Analyze
FMT {FM_name : UnknownObject, Strucutre _name : GroundVehicle/Vision_Sonsor, Fault_status : image Analysis() = fail}
Plan
APT{Policy_name : Upgrade SW, FM_name : UnknownObject, Strategy_name : SW Update})
Execute
(AST {Strategy _name : SWUpdate, Action : Update current SW to new SW, Type : External}

Figure 6. Example of Adaptation Processes

V. Conclusions

The self-adaptive system is an adaptive system that has special capabilities of evaluating and adjusting their behaviour in respond to their perception of environment and the system by itself. Especially the self-adaptation is to handle uncertainty and unpredictable situation of its operational environment. It is very useful technology to make a proactive and high reliable system that is an emergent research challenges.

In this paper, we proposed the self-adaptive system development method to make a control system dependable, secure, safe, and efficient, and operating in real-time. In case study, we developed the prototype of a smart home service robot based on the proposed approach to satisfy user requirements, overcome the system failure, and then provide seamless service to user. If the smart home service robot faces on abnormal condition, it doesn’t stop operating but can achieve the self-adaptation process to modify its behavior. That is, it can provide its services continuously without user intervention and ensure the feasibility and efficiency of the proposed approach. In future work, we are planning to achieve additional tests and verification process to enhance the proposed approach.

Acknowledgment

This work was supported by the IT R&D Program of MSIP/KEIT [10035708, "The Development of CPS(Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"]

References

[1] Betty H.C. Cheng, Rog´erio de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, “Software Engineering for Self-adaptive devices: A Research Roadmap”

[2] Ingeol Chun, Jeongmin Park, Hyeyoung Lee, Wontae Kim, Seungmin Park and Eunseok Lee, “An agent-based self-adaptation architecture for implementing smart device in smart space”, Telecommunication Systems, Vol. 52, no. 4, Apr. 2013

[3] Bill Gates, *A Robot in Every Home*, Scientific American Magazine, Jan. 2007

[4] Laddaga R., *Self-adaptive software*, Technical Report. 98-12. DARPA BBA, 1997.

[5] Salehie, M., and Tahvildari, L. (2009). “Self-adaptive software: Landscape and research challenges”, *ACM Transaction on Autonomous and Adaptation Systems*. vol. 4, no. 2, 2009

[6] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P., “Rainbow: Architecture-based self-adaptation with reusable infrastructure”, *IEEE Computing*, vol. 37, vo. 10, pp. 46–54, 2004

[7] Mukhija, A., and Glinz, M., “Runtime adaptation of applications through dynamic recomposition of components”, *In Proceedings of the International Conference on Architecture of Computing Systems*, pp. 124–138, 2005.

[8] Lapouchnian, A., Liaskos, S., Mylopoulos, J., and Yu, Y., “Towards requirements-driven autonomic systems design”, *In Proceedings of the Workshop on Design and Evolution of Autonomic Application Software*, pp. 1–7, 2005.

[9] Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., and Gjorven, E. , “Using architecture models for runtime adaptability”, *IEEE Software, pp.* 62–70, 2006.

[10] Kumar, V., Cooper, B., Cai, Z., Eisenhauer, G., and Schwan, K., “Middleware for enterprise scale data stream management using utility-driven self-adaptive information flows”, *Cluster Computing*, vol. 10, no. 4, pp. 443–455, 2007

[11] Ingeol Chun, Jinmyoung Kim, Haeyoung Lee, Wontae Kim, Seungmin Park and Eunseok Lee, “Faults and Adaptation Policy Modeling Method for Self-adaptive Robots”, Communications in Computer and Information Science, 2011, Volume 150, 156-164.

In-geol Chun received his Ph.D., M.S. and B.S. degrees in Electrical and Computer Engineering from SungKyunKwan University, Korea, in 2010, 1997 and 1995 respectively. He is currently a senior member of engineering staff in Electronics and Telecommunication Research Institute (ETRI), Korea from 1998 and also an adjunct professor in University of Science & Technology (UST) from 2012. His research interests are Cyber-Physical Systems (CPS), Autonomic Computing, Agent-oriented intelligence system, Embedded systems and Software Engineering.

Jeong-min Park received his Ph.D. and M.S. degrees in Department of Computer Engineering from Sungkyunkwan University, Korea, in 2009 and 2005, respectively, and his B.S. degree in Computer Engineering from Korea Polytechnic University, in 2003. He is currently a senior member of engineering staff in ETRI, Korea. His research interests include Cyber-Physical System (CPS), Autonomic Computing and Software Engineering.

Won-tae Kim received his B.E., M.E., and Ph.D. degrees in Electronic Engineering from Hanyang University, Seoul, Korea in 1994, 1996, and 2000, respectively. From Jan. 2001 to Feb. 2005, he was CTO of Rostic Technologies, a venture company which developed advanced mobile technologies. He joined ETRI (Electronics and Telecommunication Research Institute), the major national research institute of Korea, in March 2005. He is the team director of CPS (Cyber-Physical Systems) Research Team in Dept. of Embedded SW from Aug. 2010. His main research areas are CPS, RT Middleware, Autonomic Control, and High confidential Computing.