THESIS FOR DOCTOR OF PHILOSOPHY DEGREE

Agent-based Design Mehtod for Evolutional Systems

発展型システムにおけるエージェント指向設計手法に関する研究

Kinoshita Laboratory

Department of Computer and Mathematical Sciences

Graduate School of Information Sciences

Tohoku University

B1ID1002　Wenpeng Wei

January 2014

Contents

List of Figures

List of Tables

Acknowledgment

I would like to express my sincere and utmost gratitude to Professor Tetsuo Kinoshita for the significant and helpful advice and support through my research period over 5 years. As my supervisor, his many constructive advises and guidance were the key to the successful completion of my PhD research and thesis. In every stage of this research, precious advice and comments given have improved.

I would also like to extent many appreciation to my thesis committee members, Professor Michitaka Kameyama and Professor Ayumi Shinohara, whose constructive comments helped to improve this thesis.

I would also like to thank Associate Professor Gen Kitagata for his significant opinions and comments.

Likewise, I would like to deeply thank the staffs of the Kinoshita Lab who provided me with assistance during the development of the ideas in this thesis, and for helpful comments on the text. I thank Dr. Kazuto Sasai, Dr. Hideyuki Takahashi, Dr. Johan Sveholm, and Dr. Khamisi Kalegele for the many constructive discussions we have had.

I would like to deeply thank Dr. Akiko Takahashi from Sendai National College of Technology for her support and help to this research.

I would like to thank Ms. Ami Konno, Laboratoty secretary, for helping me focusing on my work.

I would also like to thank all my friends and colleagues who contributed to my work in so many ways.

I also very much appreciate the full scholarship award which I received from the Japanese

Ministry of Education, Culture, Sports, Science and Technology.

Lastly, I am grateful to the support and courage I have been receiving from my family.

Chapter 1 Introduction

1.1 Background

1.1.1 Internet and Open Systems

Networks of computers have been rapidly developed in the last decade. With the increment and development of networks, software systems become more and more complex and open[20]. For an instance, consider the software that is designed to be distributed in the unstable and unpredictable environment like the Internet. The bandwidth of the connections among those components are changing over time. In some extreme situation there might be no connection at all. The construction of the system also changes dynamically while the system is running. Some components are probably offline. Some might be replaced by new versions. There is almost no way to predict the situation of the environment and the construction of that kind of open system.

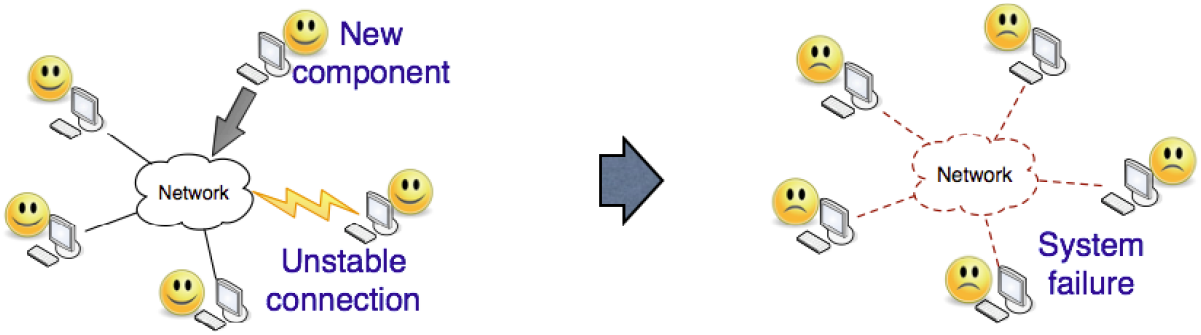


Fig. 1.1: System failures due to unstable environment

In practice, it becomes obviously difficult to design and to control the behavior of the system.

To design a open system, in general the designer is required to decide the behavior of the system in all of the situations that the system is possible to face while running. However, it is clearly difficult, if not impossible, to predict those changing situations in design process. By the reasons mentioned above, the design and development of open systems are still an open challenge in the real world.

Therefor, a methodology that provides appropriate support for engineering large-scale open systems is significant and useful[11].

1.1.2 Self-* Concept from Automatic Computing Community

As one possible solution to the problem about open systems that is mentioned in the last section, Automatic Computing community introduces the Self-* concept to provide a approach of archiving systems, which are more intelligent and autonomous. The corresponding concepts can be categorized into three different layers as follows[18].

Primitive Level This level contains the underlying primitive properties of Self-* concept. For example, self-awareness, which means that the system is aware of its self states and behaviors, and context-awareness, which means that the system is aware of its operational environment.

Major Level This level contains four major properties.

Self-configuring is the capability of autonomous and dynamic reconfiguration.

Self-healing is the capability of discovering, diagnosing and reacting to disruptions.

Self-optimizing is the capability of managing performance and resource allocation.

Self-protecting is the capability of detecting security breaches and recovering from their effects.

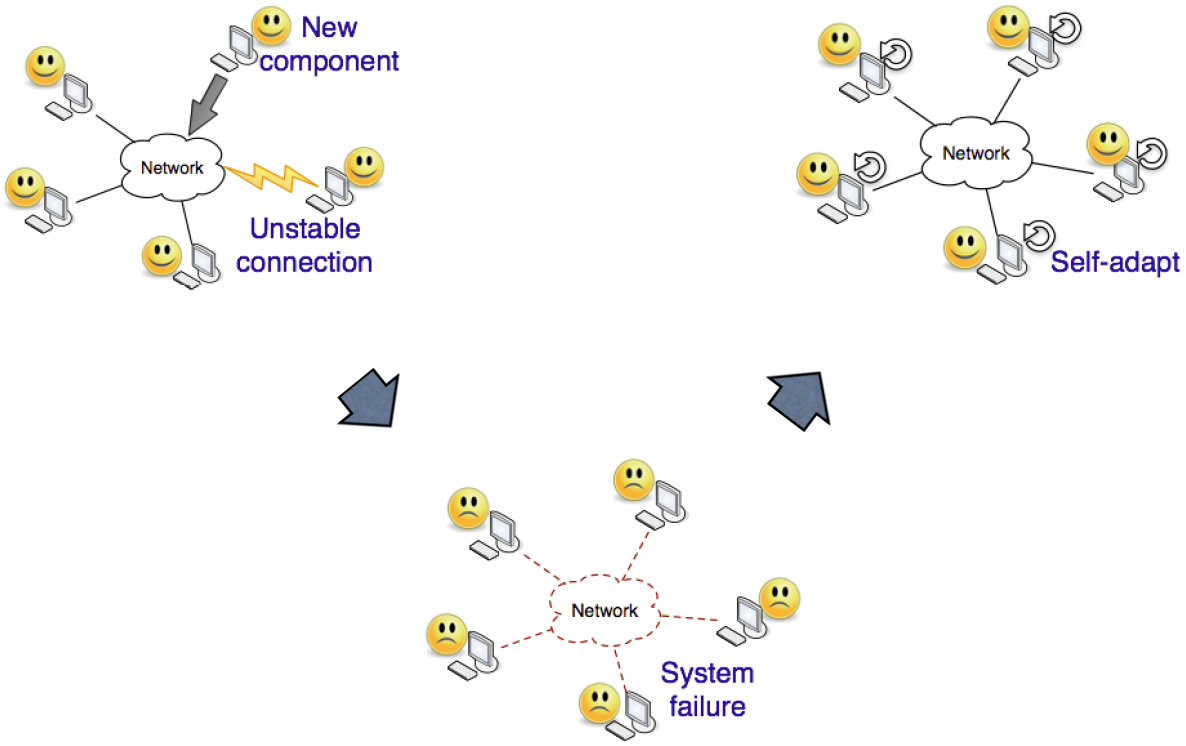


Fig. 1.2: Self-adaptive system recovers from system failure

General Level This level contains global properties of self-adaptive software. The self-adaptiveness is a top-down approaches, with which the system is supposed to change itself to adapt to the changing operational environment or changing requirement.

The self-adaptive property is actually a general solution from Autonomic Computing community for the problems around open systems mentioned last section. However, as a general approach, it has some short comes for particular situations. For an instance, self-adaption is able to recover the system after environment or requirement changes, but is not able to avoid the negative effect of them. Those short comes make self-adaptive system not suitable for the mission critical applications such as financial applications and security applications.

1.1.3 Evolutional System

As a solution for the problems self-adaptive systems have, Evolutional System is introduced in our previous research.

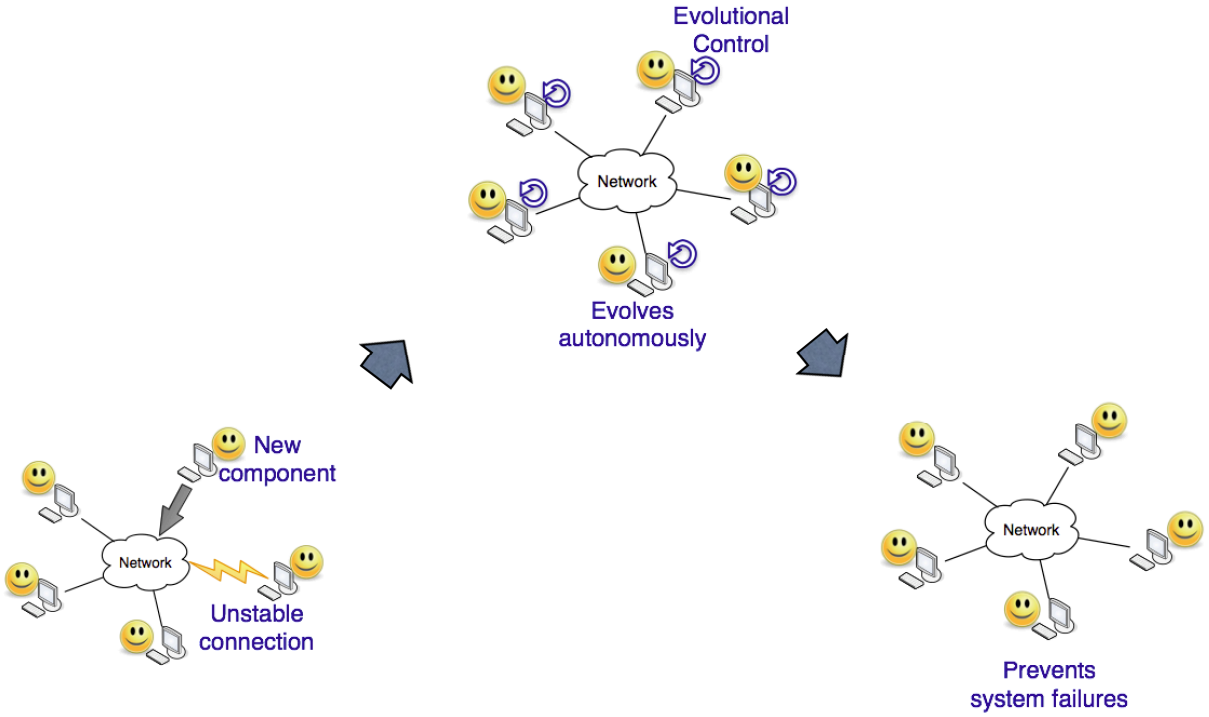


Fig. 1.3: EAS prevents system failure

An Evolutional System(ES) is an intelligent system with the following characters.

- An ES evolves autonomously and dynamically over time.
- An ES improves itself in many aspects such as stability and effectiveness continuously.
- An ES features unique properties that over self-* systems such as structure evolution, potential improvement and deterioration resistant.

Specifically, deterioration resistant means the system ability to autonomously prevent the negative effect of undesired changes of system structure or system function before those changes actually arise. In contrast to the self-adaptive system, that unique property makes ES an ideal solution for those mission critical applications, which is difficult for self-adaptive approach.

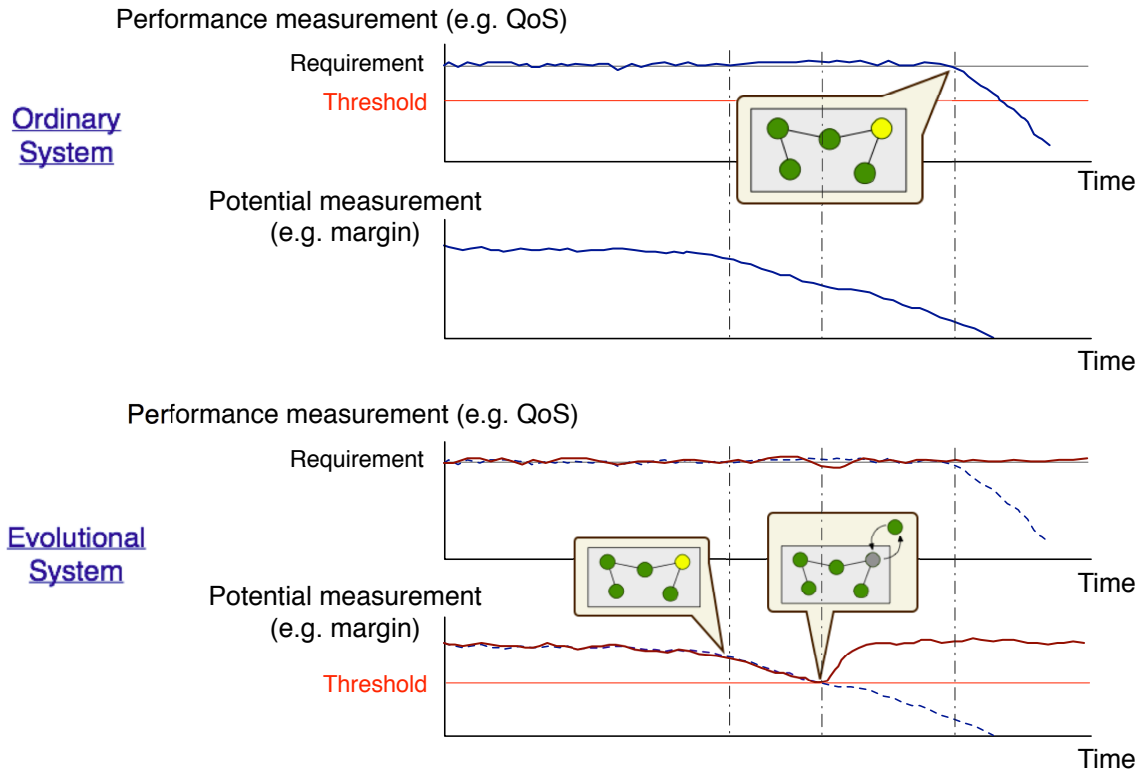


Fig. 1.4: EAS operation images

The conceptual architecture of an ES is shown in Fig. 1.5.

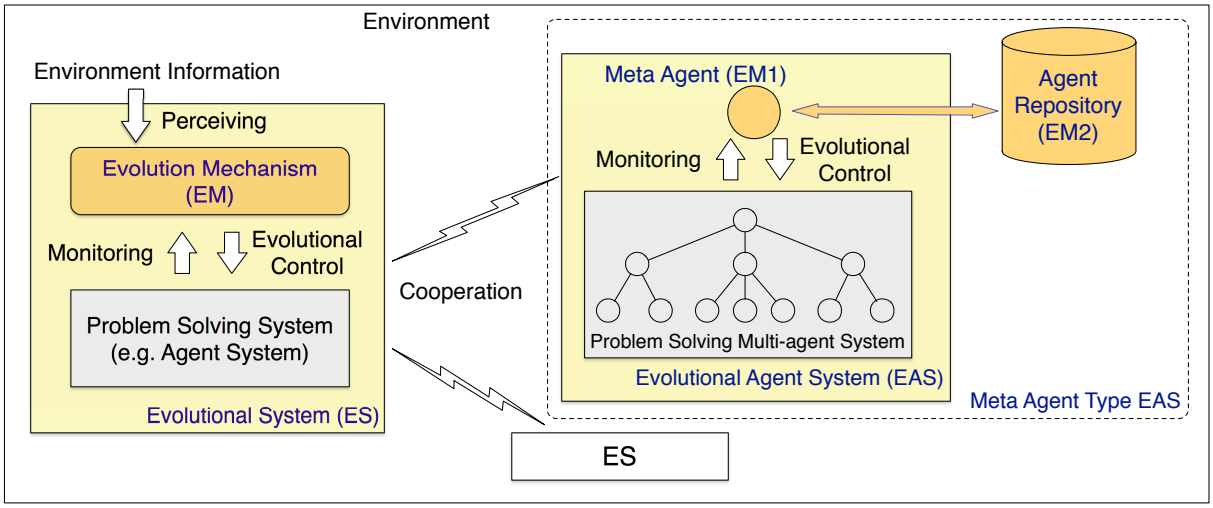


Fig. 1.5: The conceptual architecture of an ES

The problem solving system is the ordinary system dealing with the system requirement. For

an instance, it could be an ordinary plain agent system. Besides that, the additional Evolution Mechanism is introduced. The Evolution Mechanism monitors the problem solving system and perceives from the operational environment of the system. Then it judges the situation of system by using all the parameters it monitored. If there is any possible undesired change, the Evolution Mechanism adjusts or re-organizes the problem solving system to change the character and behavior of the system to avoid the negative effect of those changes even before those changes actually arise.

It is obviously that the key point of design an Evolutional System is to know how to design the Evolution Mechanism. Nevertheless, those additional complexity and flexibility of the ES appears as serious challenges for engineering them properly.

This research focuses on the design of Evolution Mechanism towards the realization of deterioration resistant property of Evolutional Systems.

1.1.4 Agent-Oriented Software Engineering

We choose to use software agent and multi-agent system(MAS) as the paradigm for the design of Evolutional Systems. It has good enough advantage to decide to use them.

Firstly, multi-agent system is the actually standard paradigm for distributed information processing systems. Featuring autonomous and sociable, software agents are ideal first class for the design of intelligent autonomous system distributed over networks.

Secondly, through the development over decades, there a number of mutual design methodologies to not only adopt but also to learn from. The legacy from existing research is specifically valuable and useful. Some of the famous multi-agent system design methods can be categorized into two types.

- General purpose design methods for MAS

  The Gaia methodology[28], the Tropos methodology[5] and the MaSE methodology[8]

- Special purpose design methods for MAS
  - For open systems and system adaption: the ADELFE methodology[3]
  - For telecommunication applications: the MESSAGE methdodology[6]

Although there are such a number of mutual design methodology for multi-agent system, there is still no specific design method or tools for Evolutional Systems yet. Therefor, the remain of this research aims to provide a agent-based design method for Evolutional Systems.

1.2 Objectives

1.2.1 Research Objectives

This research aims for the reliable and effective realization of Evolutional System that features the deterioration resistant property. The key points of this research objective is as the follows.

Reliable the Evolutional System must meet all design requirements and performs stably

Effective the development cost for the Evolutional System must stay low even with the extra flexibility

Deterioration resistant the Evolutional System must be able to autonomously prevent the effect of undesired changes of system structure or system function before those changes actually arise

Towards that objective, this research focuses on the design method and tools for Evolution Mechanism, which performs Evolutional Control of the Evolutional System.

1.2.2 Challenges

Focusing on the design of the Evolutional Mechanism for performing Evolutional Control, challenges appear as the follows.

(C1) To know how to perform Evolutional Control by changing the behavior and construction of the problem solving system reliably and effectively. This challenge can be described by two sub challenges as following.

(C1.1) Keeping the problem solving system construction flexible yet controllable.

C(1.2) Keeping the development cost of the system low even with the extra flexibility.

(C2) To know to decide the appropriate timing of performing Evolutional Control before actual undesired changes arise without extra design burden.

1.2.3 Proposals

Against to the challenges mention last section, we propose a agent-based design method for Evolutional Systems as the follows.

Proposal to challenge (C1) This proposal can be described by two sub proposals as following.

Proposal to challenge (C1.1) (P1) Organization controlling architecture for agent-based Evolutional Systems

Proposal to challenge (C1.2) (P2) Protocol design method for agent-based Evolutional Systems

Proposal to challenge (C2) (P3) Autonomous knowledge construction method for Evolutional Control

The details of each proposal are discussed in the following chapters.

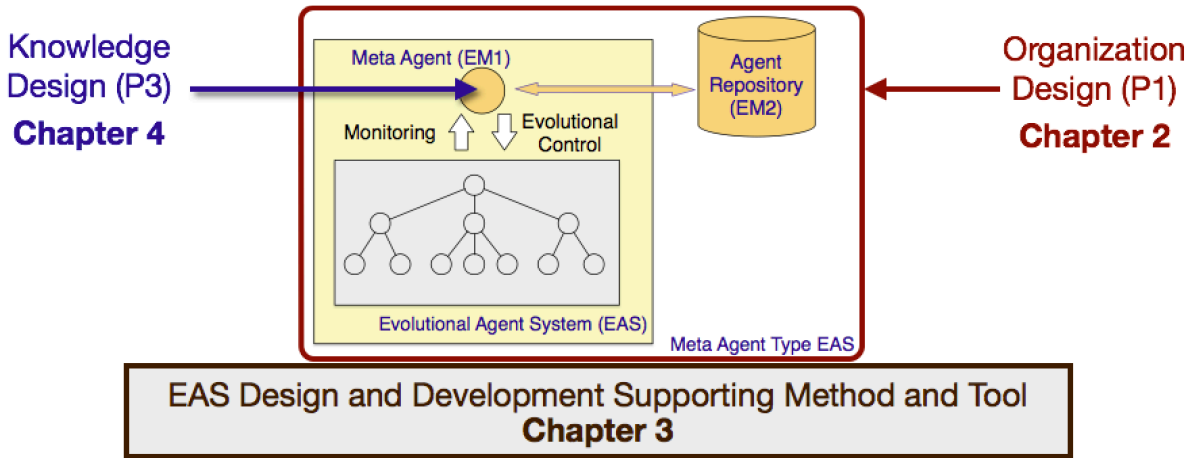


Fig. 1.6: Position of proposals and chapters in this research

1.3 Summary

As the background of this research, it becomes common that information processing systems are distributed over vary network environment. For this kind of distributed systems, the activity properties of them change irregularly depending on the changes or failures of system components as also as the changes of network situations. The unstable changing of activity properties is one of the main reasons, which are responsible for negatively effecting the quality of services. In general, it is extremely difficult to consider all the possible changes the system will face to in the design process. Presently, many research focusing on the judgment of abnormal situation of systems and the recover solutions are under taken.

On the other hand, the research about cooperation distributed system call Evolutional System is introduced. By constructing the system as multi-agent systems, Evolutional System approach proposes the methodology that performs preventing processing in advance before the effects of system changes arise by monitoring the activity properties of the system, which affect system behaviours. An appropriate method for design and development of Evolutional System is important. In this paper, we provides a design method based on multi-agent system, which is described in the following 3 chapters.

Research Objective

This research aims for the reliable and effective realization of Evolutional Systems(ES) that features the deterioration resistant property.

Challenges

(C1) To know how to perform Evolutional Control by changing the behavior and construction of the problem solving system reliably and effectively. This challenge can be described by two sub challenges as following.

(C1.1) Keeping the problem solving system construction flexible yet controllable.

C(1.2) Keeping the development cost of the system low even with the extra flexibility.

(C2) To know to decide the appropriate timing of performing Evolutional Control before actual undesired changes arise without extra design burden.

Proposals

(P1) Organization controlling architecture for agent-based Evolutional Systems

(P2) Protocol design method for agent-based Evolutional Systems

(P3) Autonomous knowledge construction method for Evolutional Control

Chapter 2 Organization Controlling Architecture for Agent-based Evolutional Systems

2.1 Overview

With the rapid development and popularization of smart portable devices, it is becoming common to use various network services on portable devices such as smart phones nowadays such as navigation services. However, those portable devices are usually used in unstable and dynamic environments with certain limitations such as variation in network bandwidth. In addition to that, not only the limitations of the environments, but also the limitations of the devices themselves such as effects of limited battery power on user experiences. For instance, users of multimedia sharing services like YouTube have to always keep an eye on the battery charge of their devices while enjoying the contents to make sure the playback will not be suspended.

A number of researches trying to improve the situation mentioned above have been undertaken in recent years. Those research efforts can be divided into two mainstreams. One of them is the works that attempt to reduce local resource consumption of the portable devices by optimizing local processing, for instance, a smart phone application manager trying to extend battery life by exiting background processing. Due to the lack of the ability to control service provisions outside the portable devices, those works show only limited effect in the case of that the numerous system resources are consumed by the services which are provided by independent service providers. The other one is the efforts on the service provider side to prepare

optimized services based on the situations of users and their devices. Nevertheless, it is obviously difficult, if not impossible, to predict all possible situations and prepare suitable services for all users in advance.

To solve the problems mentioned above, this research aims to provide appropriate services to portable devices flexibly by considering the situations of users and their devices while enjoying the services. The remainder of this paper is organized as follows. Related works and main challenges are discussed in the next section. In Section 3, the proposal is presented in details. An experimental system is described and the results discussed in Section 4. Last but not least, Section 5 presents important conclusions.

2.2 Related Works and Problems

2.2.1 Related Works

There are quite a number of researches aiming at reducing the energy consumption of portable devices. A mechanism which sleeps the processors of portable devices for short periods to save battery power is suggested in Brakmo et al [4]. While in Qiu et al [17], an algorithm of task scheduling to aggressively reduce energy consumption is provided. Both approaches mentioned above are limited in the portable devices and lack for the ability to control service provision when portable devices are using network services.

On the other hand, automatic and dynamic construction of services according to the situations of service users and environments has been an active research area for a long time. In the last decade, a number of researches on autonomous service provisioning systems have been trying to provide appropriate services depend on the situations of the systems. Users of those systems are required to construct and to control the service provisioning system dynamically to receive adequate services.

Regarding multi-agent based autonomous service provision systems, methods for construct-

ing the autonomous overall system using a combination of software agents that collaborate with each other autonomously have been proposed[10, 26]. Those methods construct agent organizations with mutual interactions among autonomous agents to achieve autonomous system characteristics. Depending on QoS degradation and environmental changes, the systems change parameters and reconstruct the agent organization during service provisioning. The QoS degradation and environment changes serve as triggers in those systems which function reactively.

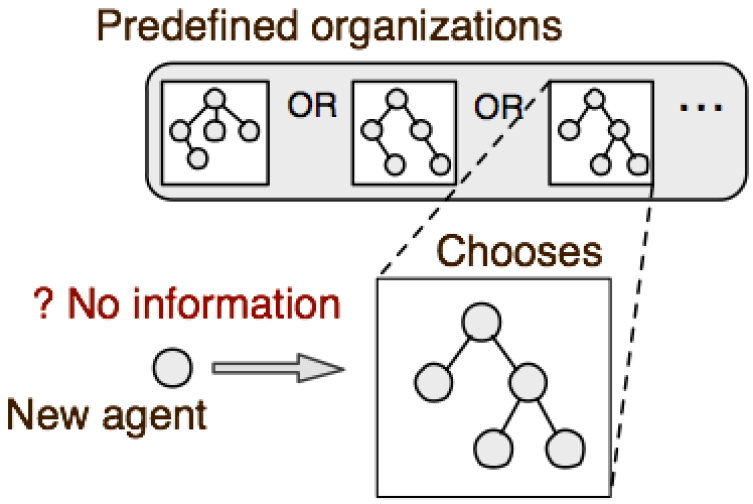


Fig. 2.1: Static organization design

An agent organization model to construct a system that recovers autonomously when trouble occurs is reported in Oyenan et al [16]. The knowledge for the control of agents is required by the reported organization model: the condition of the agent organization have to be monitored. A multi-agent system is constructed systematically by operating agents autonomously under the model. In case some agent is aborted or some trouble which does not satisfy the system's objective during service provision, the reconstruction of agent organization is performed to recover service provision autonomously. Nonetheless, service suspension is difficult to avoid by this model.

A multi-agent system architecture incorporates a policy to manage applications, resources and service provision is proposed in Tesauro et al [27], thereby achieving means to construct

systems based on user requirements, to recover from system failures and to optimize systems. However, in case that the system handles every system component using the same policy, the overall system becomes overloaded easily.

There are also a number of researches aiming to construct multi-agent systems which are able to adapt to changeable unstable environments. Disparate models for that purpose have been introduced in various publications [12, 19]. The former provides a meta-model focusing on organizational concepts such as roles in a multi-agent system. The latter gives a concrete model for the actual modeling agent organizations of the application systems. In those approaches, the adaption of agent behavior is realized by changing the roles which agents play dynamically after some predefined conditions are perceived. In Luckey et al [15], a useful tool to defined those important predefined conditions is provided. By extending standard UML use cases, the proposed Adapt Cases enable the explicit modeling of those conditions with domain specific means early in the design phase of software engineering process. It is noteworthy that the approaches mentioned above perform adaptive actions after the system falls into some undesirable situations. Even if the systems recover after that, user experiences are noticeably affected especially under an unstable environment such as wireless network.

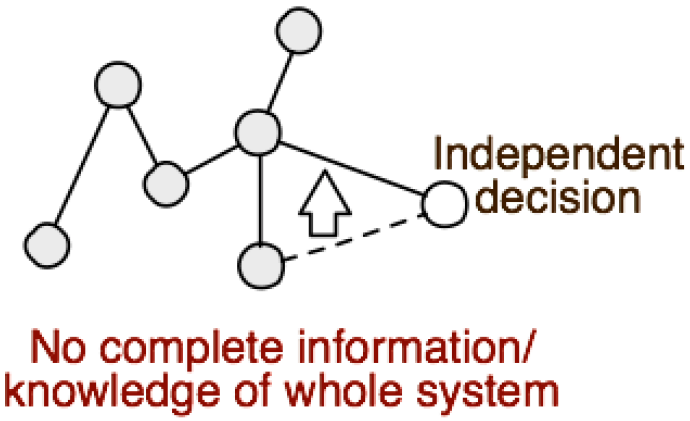


Fig. 2.2: Dynamic organization design

Addressing to the problems mentioned above, in our previous researches[22, 23], a method that is able to predict system troubles in order to maintain service provision by adopting the

Evolutional Agent System(EAS)[21] has been proposed. Instead of monitoring undesired conditions of the applications as the triggers for reconstruct agent organizations, the proposed approach focuses on the characteristic of agent behavior and application requirements to obtain the current system potential as system margin for providing required services. Based on the system margin, the proposed approach controls multi-agent system proactively considering the "control possibility" and "control effect" for reducing the decrement of QoS to the absolute minimum while providing services by reconstructing agent organizations. This approach avoids QoS deterioration effectively for systems which consist of immobile platforms. Therefore, it is natural to apply the succeed approach to portable devices. However, there are several challenges due to the unique limitations of portable devices mentioned above.

2.2.2 Problems of Adopting Existing Approaches

For portable devices which are used in unstable environments, it is obvious that autonomous service provisioning systems are suitable for users to experience adequate quality. While adapting our previous experience to portable devices, we meet the challenges that can be described by the following problems.

Without considering system power storage and other local resources, device battery life is shortened by additional control processing. The extra control process reduces user experiences while enjoying provided services due to the limited local resources.

Different from immobile devices, portable devices have certain limitations. Those limitations have to be considered while applying EAS concept to systems which involve mobile devices. The limited battery capacity is one of those limitations. In an EAS, besides to the agent system which provides the required services, there is an extra evolution mechanism to improve the whole system. It is obvious that the extra computation of the evolution mechanism consumes extra energy of the system. That might not be an serious issue for immobile devices but for the devices with limited power supply it means shorter battery life which is an undesirable

condition. Another limitation is the limited processing power of the portable devices. On those devices, extra controlling process consumes additional computation resources and effects user experiences.

Therefore, against those challenges we propose a new design of energy-consumption-aware EAS for portable devices. The proposed design customizes and extends previous EAS model to overcome the limitations of portable devices. In brief, the proposal provides solutions to the challenges mentioned above as follows.

A service provision control mechanism considering device energy consumption and local resources. In addition to QoS parameters, local resources parameters are also taken into account while service is being provided to reduce the influence to device battery life. A system architecture which considers user experiences during service provision. Instead of users and their devices, software agents on service provider side deal with the extra controlling.

Instead of performing control process of evolution mechanism on the portable devices, the proposed design performs those process using immobile platforms of the whole system as much as possible to minimize the effect on portable devices. The details of the proposal are described in next section.

2.3 Proposal

2.3.1 Proposal Overview

The overview of the proposal is described as the following.

Proposal

Organization controlling architecture for Agent-based Evolutional Systems.

Proposed solutions

1. Controllable construction of agent organization

   This solution makes it possible to constructs agent organization dynamically while running according to user requirement. It also provides ability to be controlled while re-organization.

2. System architecture based on Evolutional Agent System model

   This solution provides system design to support the realisation of deterioration resistant property of Evolutional Agent System(EAS).

2.3.2 Controllable Construction of Agent Organization

In this section the details of the controllable construction of agent organization are described. The idea of this solution is shown in Figure.2.1.

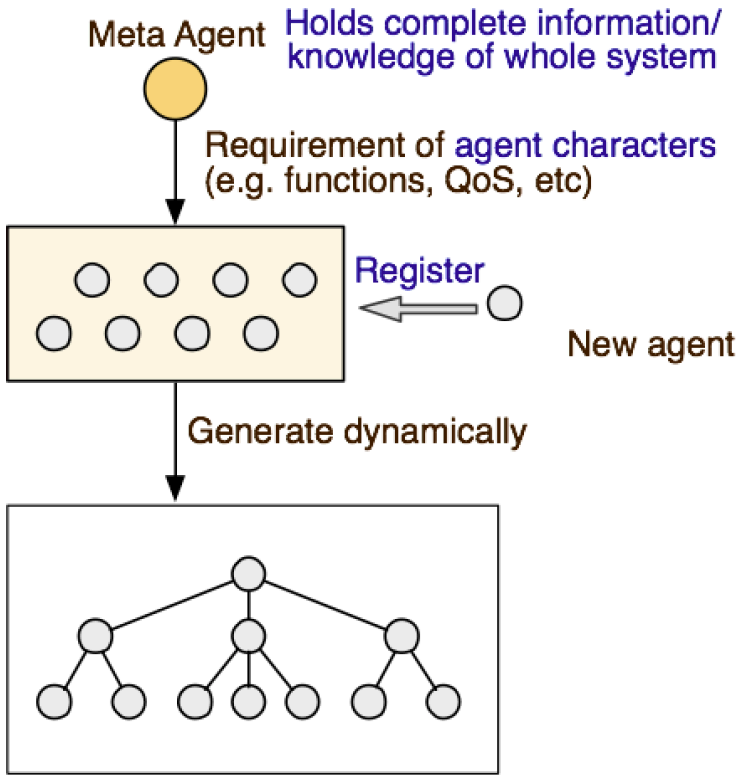


Fig. 2.3: Controllable construction of agent organization

In general, the organization of a system is either controlled in a centralized fashion or in a decentralized way. For the former, there is usually a controller in the system, which is holding all the organization candidate information. While there is a need to be reorganized, the controller chooses the most appropriate organization depending on the strategy of the whole system and the situation the system is facing. The centralized approach usually achieves best result for a closed system thanks to that all the information of the system is hold by the controller. However for the open system, there is always a chance that unexpected component comes to the system, the information of which is not available for the controller. That makes the controller losing the full control of the system. In that case, the reorganizing fails due to the uncertain center control.

For the latter, there is no such a center controller in the system. Instead of it, each agent decides which one is going to be communicated depending on its own goals and decision making. In this case, usually each agent in the system only holds part of the information of the system. For that reason, it is extremely difficult, if not impossible, to make the whole system reorganize to achieve some certain system level goals. In other words, it is hard to control at system level.

2.3.3 System Architecture based on EAS Model

A Evolutional Agent System (EAS) is a multi-agent system which has the ability to control and reconstruct itself actively to recover QoS after analyzing the system environment, not merely passively responding to changes. We applied the EAS concept to construct an autonomous service provision system in the previous research[22]. Based on that successful experience, in this research we customize the EAS model from [22] by explicitly modeling energy consumption related characteristics of agent behavior and environment to overcome the limitations of portable devices. The customized EAS model is described in a syntactic manner as the following.

Definition 1. An Evolutional Agent System, denoted as an EAS, is designed based on ne, R, S, E and P;

$$EAS =< ne, R, S, E, P > .$$

Where ne signifies the name of the EAS, R denotes a set of requirements given to EAS, S is the EAS structure, E stands for the environment in which EAS operates, and P is the EAS property.□

The EAS usually consists of several agent workplaces and one agent repository running on distributed platforms. Detailed elements of an EAS in this research are described as the following definitions.

Definition 2. R includes $req_k(k = 1, 2, ..., N_r)$ and req_k is determined as a required service function $req\text{-}f_k$ and the required quality $req\text{-}f\text{-}q_k$ of the $req\text{-}f_k$, as shown below.

$$R = \{req_k | req_k =< req\text{-}f_k, req\text{-}f\text{-}q_k >; k = 1, ..., N_r\}$$

Here, N_r is the total number of the required functions.□

Definition 3. S is an organization of an EAS. It consists of AG, STR and $AL\text{-}AG$. S is

$$S =< AG, STR, AL\text{-}AG >,$$

where AG denotes a set of EAS member agents, STR stands for the structure of an agent organization and $AL\text{-}AG$ signifies a set of alternate agents which member agents in AG can be replaced by.□

Firstly, AG includes $ag_i(i = 1, ..., N_a)$ which is determined with the following: identifier na_i ; a set of services $Sv\text{-}ag_i$ which can be provided by the ag_i .

$$AG = \{ag_i | ag_i =< na_i, Sv\text{-}ag_i >; i = 1, ..., N_a\}$$

Here, N_a is the total number of EAS member agents. Member agents AG are problem solving agents which run in agent workplaces to provide required services.

$Sv\text{-}ag_i$ includes sva_i, which is determined as a provided function $func_i$,the provided quality $func\text{-}q_i$ of the $func_i$ and relative energy consumption $func\text{-}ec_i$ of providing $func_i$ as follows:

$$Sv\text{-}ag_i = \{sva_i | sva_i =< func_i, func\text{-}q_i, func\text{-}ec_i >; \\ i = 1, ..., N_s\}.$$

Here, N_s is the total number of functions agent ag_i has.

Secondly, $AL\text{-}AG$ includes $ag_j (j = 1, ..., N_{alt})$ as the following.

$$AL\text{-}AG = \{ag_j | ag_j =< na_j, Sv\text{-}ag_j >; j = 1, .., N_{alt}\}$$

Here, N_{alt} is the total number of the alternate agents. Alternate agents $AL\text{-}AG$ are stored in the agent repository which is running on top of a immobile platform such as a PC server.

Finally, STR includes $rel_{ij} (i, j = 1, ..., N_a)$ which denotes the relation r_{ij} between ag_i and ag_j. Here, ag_j and ag_i are included by AG, r_{ij} is included by Rel which is the set of inter-agent relations defined in EAS, respectively.

The STR is

$$STR = \{rel_{ij} | rel_{ij} =< r_{ij}, ag_i, ag_j >; \\ ag_i, ag_j \in AG; r_{ij} \in Rel\}.$$

All the agents in the EAS are stored as alternate agent in the agent repository at first. After a requirement is issued to the EAS, appropriate agents are initiated from the agent repository to the agent workplaces as member agent to construct agent organizations which provide the user required services. Those member agents are selected in a design process as the following.

Definition 4. In a design process, the EAS is feasible and S is determined when all requirements and functions are compared and matched. It can be described by the following expression:

$$
\begin{aligned}
(\forall k) req_k &= < req\text{-}f_k, req\text{-}f\text{-}q_k > \in R, \\
(\exists i, j) ag_i &= < na_i, Sv\text{-}ag_i > \in AG; \\
sva_j &= < func_j, func\text{-}q_j, func\text{-}ec_j > \in Sv\text{-}ag_i; \\
req\text{-}f_k &= func_j; \\
req\text{-}f\text{-}q_k &= func\text{-}q_j.
\end{aligned}
$$

Using this expression, S is determined.□

By that design process, agents that have necessary functions to realize the user requirements are selected as member agent. After the process, the EAS is ready to provide required services using the agent organization constructed by member agents AG.

P is the operating property of the EAS. It is determined by the evolutional mechanism of the EAS. If the system can recover its performance, then the evolutional mechanism operates the system actively and dynamically based on the operating status of the multi-agent system and the information of system environment. In this study, we design the evolutional mechanism as the knowledge and abilities of a meta-agent(mag). It manages P as internal information. The details of evolutional control based on system margin are described in [22]. In addition to that, to overcome the certain limitations of portable devices, the hardware information of distributed platforms is also necessary as follows.

Definition 5. E is the information of the behavioral environment of the EAS system. In this research, E consists of the hardware information HW_i of each distributed platform:

$$E = \{HW_i | i = 1, ..., N_{dp}\},$$

where N_{dp} is the total number of distributed platforms.□

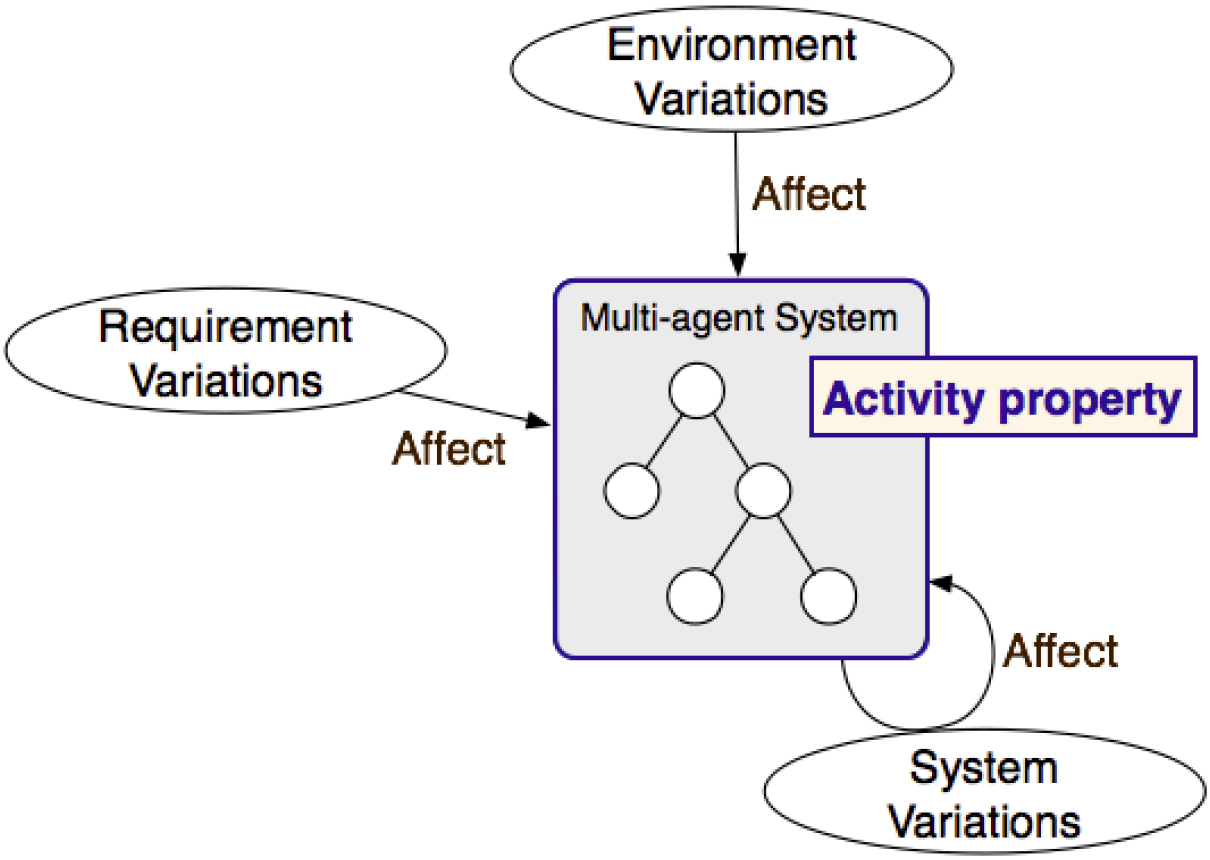


Fig. 2.4: Activity Property P in EAS model

With that information, the EAS is able to be aware of the energy consumption and potential of the whole system as discussed in the next section.

To design an energy consumption aware control mechanism, we propose the Energy Continuance Index ECI as an indicator of the estimated continuance of system energy storage for each distributed platform. ECI is one of the detectable properties of EAS which are represented by the set P.

By calculating the energy consumption related data which is obtained from platform hardware, it is possible to be aware of the impact of the current service on the energy storage of the platform. ECI is defined as a numerical value to measure that impact which is directly proportional to current battery charge of the platform and is inversely proportion of the energy consumption speed.

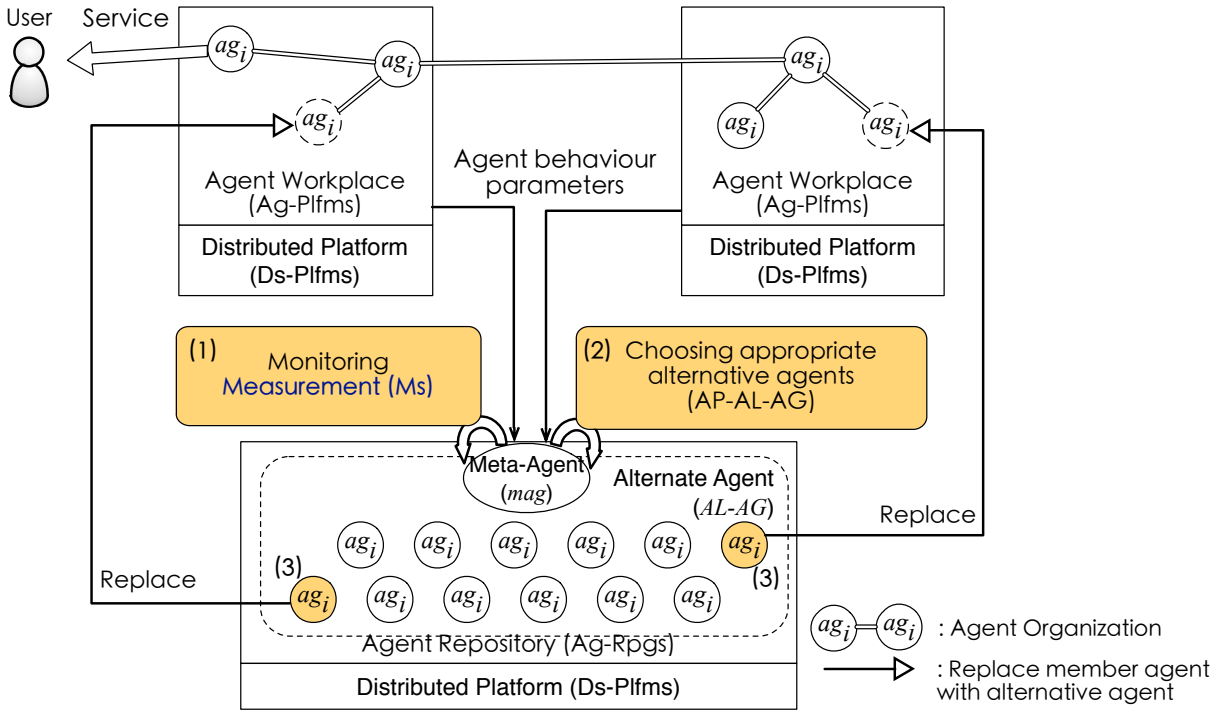


Fig. 2.5: System architecture based on EAS model

Definition 6. The Energy Continuance Index ECI is obtained using platform hardware information HW_i as the follows.

$$ECI = \{eci | eci =< eng\text{-}strg_i, ECF_i >; eng\text{-}strg_i \in HW_i; \\ ECF_i \subseteq HW_i; i = 1, ..., N_{dp}\}$$

where $eng\text{-}strg_i$ denotes the current energy storage of the platform, ECF_i is a set of energy consumption factors of the platform as follows:

$$ECF_i = \{ecf_{ij} | ecf_{ij} \in HW_i; j = 1, ..., N_f\},$$

where N_f is the number of energy consumption factors of the platform.□

In this study, for simpilicity we define eci as the following.

$$eci_i = \frac{N_f \times eng\text{-}strg_i}{\sum_{j=1}^{N_f}(\alpha_{ij} \times ecf_{ij})},$$

where α_{ij} is the weight of each factor and N_f is the total number of ecf_{ij}. The range of α_{ij} is $(0, 1]$ and for each ecf_{ij} a specific α_{ij} is defined. $eng\text{-}strg_i$ is a numerical value which represents energy storage, the battery charge for instance, in the range of $[0, 1]$. ecf_{ij} denotes the energy consumption factor which influences the energy storage using a value from $[0, 1]$. Therefore, eci_i is a numerical value which is larger than or equal to 0, where a value of 0 shows that the platform has no energy in the storage to operate anymore, while a value of positive infinity means that the platform has minimal energy consumption and is expected to provide maximal battery life that hardware allows. In practice, $eng\text{-}strg_i$ and ecf_{ij} can be various over disparate platforms. A concrete instance is shown in next section.

ECI is updated and monitored frequently at regular intervals during service provision. The architecture of EAS with energy consumption awareness is presented in 2.5. The agent workplace is the behavioral platform which contains the member agents ($ag_i \in AG$) to provide user required services. The alternate agents ($ag_i \in AL\text{-}AG$) are managed by the meta-agent (mag) in the agent repository which constructs agent organizations in response to user requirements. mag also retrieves platform specific hardware information HW_i from each distributed platform and keeps updating and monitoring ECI.

Using the architecture described above, the agent behavior to archive energy consumption awareness is designed as the follows. During service provision, if a ECI degradation, i.e. acute energy consumption which may seriously affect system power storage, is detected by mag, it dynamically reconstructs the agent organization by replacing corresponding member agents with alternative agents which have equivalent functions with same QoS and lower relative energy consumption as described next.

Definition 7. The agent organization S is reorganized if

(1) ECI degradation is detected:

$$ec_i + \Delta ec_i < \theta,$$

(2) appropriate alternative agent is available:

$$\begin{aligned}
(\exists i) ag_i &\in AG, \\
(\exists j) ag_j &\in AL\text{-}AG, \\
&\text{where} \\
(\forall k) sva_k &= < func_k, func\text{-}q_k, func\text{-}ec_k > \in Sv\text{-}ag_i, \\
(\exists q) sva_q &= < func_q, func\text{-}q_q, func\text{-}ec_q > \in Sv\text{-}ag_j; \\
func_k &= func_q; \\
func\text{-}q_k &= func\text{-}q_q; \\
func\text{-}ec_k &> func\text{-}ec_q.
\end{aligned}$$

Here, θ is the threshold to decide ECI degradation.□

By implementing those models as behavior knowledge of software agents in the EAS, energy consumption awareness is realized. An application example is described in the next section.

2.4 Experiment and Evaluation

2.4.1 Experiment Overview

The objective of this experiment is to verify if proposed architecture could dynamically reorganize by considering system situation.

The scenarios of the experiment is as the follows. Implement a experiment system streaming live video to a portable device, which is supposed to consider the energy consumption of the portable device at first. The system using proposed architecture is expect to scale down the

video by re-organization to reduce the energy consumption of the portable device while battery consumes too fast on it.

The object of comparison is the system without proposed architecture using the same initial organisation through the experiment. The parameter of comparison is the battery life of streamed video play back at a full charge.

2.4.2 Experiment System Design

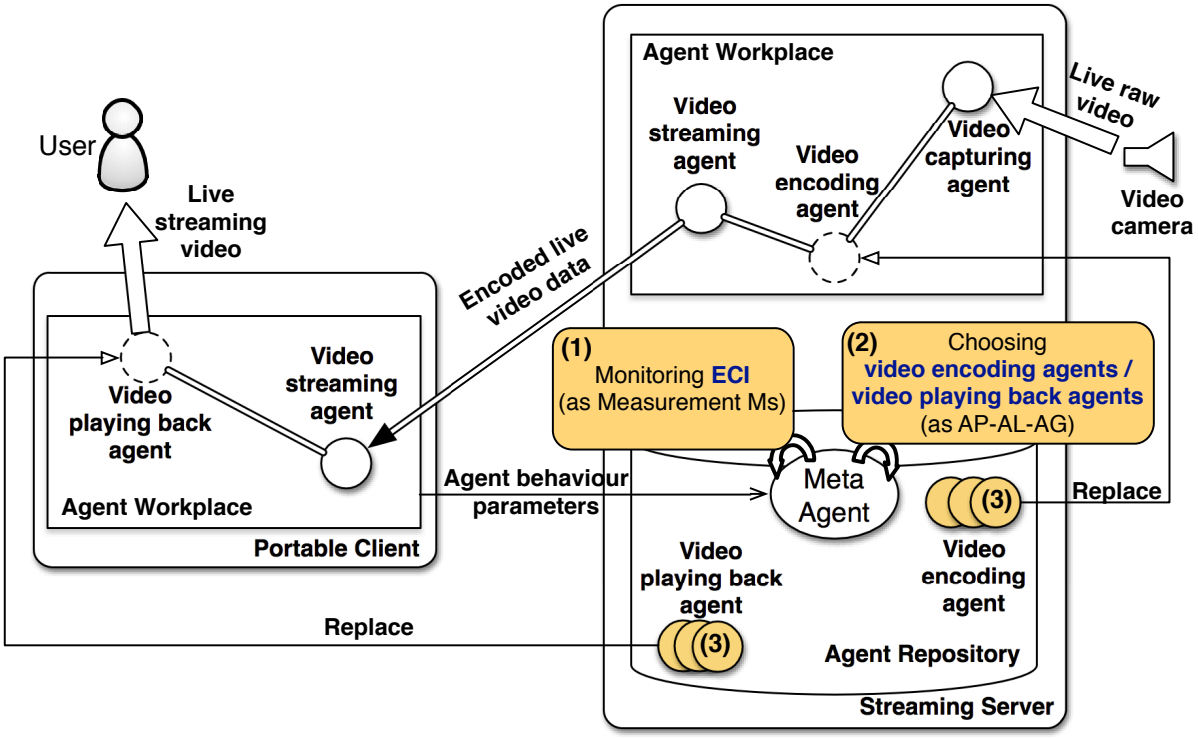


Fig. 2.6: Experiment system design for organization controlling architecture

To evaluate our proposal, we applied it to a Flexible Multimedia Network Middleware (FMNM)[24], which is an agent-based multimedia communication system providing adequate multimedia communication services dynamically based on user requirements. In this research, the experimental application contains a live video server and one or more portable devices as clients. The server captures live video with a web camera and encodes the video while streaming it to the

clients to play it back. As the previous research of FMNM, the quality of the provided service in terms of video frame rate(fps) and video resolution are used as QoS parameters. Every QoS parameter is implemented as a component of design specifications and the parameters of user requirement's function and its quality are implemented as $req\text{-}f$ and $req\text{-}f\text{-}q$. The agents in the system are as follows:

video capturing agent The agent controls the web camera and passes the video raw data from the camera to the video encoding agent. The controller and driver of the web camera on the video server is implemented as the knowledge of this video capturing agent.

video encoding agent The agent encodes the video raw data using different video encoders with variable parameters to pass to the video streaming agent. All the video encoders available on the video server are wrapped and implemented as the knowledge of this video encoding agent.

video streaming agent The agent controls network communication to send and to receive encoded video data from video encoder agent to video playing back agent. The networking related controllers and drivers of both server and client are implemented as the knowledge of this video streaming agent.

video playing back agent The agent receives encoded video data from video streaming agent and decodes the data using corresponding video decoder to play back the video on the display. All the video decoders and the video player on the client are implemented as the knowledge of this video playing back agent.

In addition to that, each of those agents also has the knowledge for communication with each other. Specifically, for the video encoding agent and the video playing back agent, there are disparate agents for different video encoders/decoders with varying parameters. The agents for each type have the same functions but with different internal implementations. They perform the same task such as encoding a video with different QoS and energy consumption depending

on the concrete encoder/decoder implementation and parameters. During system initiation, user requires a multimedia communications service which is able to play a remotely captured live video on a Android tablet device at a required frame rate and resolution as required QoS. The system selects appropriate member agents from the agent repository according to the design process defined in Definition 4. In this process, the agents combination which is able to provide required services at best QoS is selected to construct agent organization.

2.4.3 Implementation of Experiment System

After constructing the agent organization which is able to provide user required service, the experimental prototype system continuously provides streaming services for a live video captured from a web camera of a PC to an Android tablet through wireless network. For the Android tablet, in addition to the conventional QoS parameters such as video frame rate, local resource parameters are also involved as HW. In this prototype system, those local resource parameters include the following: CPU load(%), memory usage(%), display brightness(%), wireless network signal strength(%), network traffic(kB) and battery charge(%).

$$HW =< \textit{cpu-load}, \textit{mem-usg}, \textit{disp-brt}, wifi\text{-}sigl, nw\text{-}traf, \textit{batt-chrg} >$$

During the required service provision, the meta agent mag in the agent repository receives HW update from the table every second and use it to calculate the energy continuance index ECI of the tablet. For the use case in this experiment, the battery power of the tablet is mainly used in three parts as the following.

1. Network part: to receive video data through the wireless network.

2. Processor part: to decode the video data using CPU calculation.

3. Display part: to playback the video on the display.

Specifically, for the network part, the power consumed for the wireless communication is inversely proportional to the signal strength of the networks. More power is needed for a connection with relatively weak signal. In summary, there are three factors of energy consumption ecf of the tablet. Also the tablet is the only portable device in this experimental system for the calculation of ECI. Therefore, ECI of the system is defined as follows.

$$ECI = \frac{3 \times \textit{batt-chrg}}{\textit{cpu-load} + \textit{disp-brt} + f\textit{-scale}(\frac{\textit{nw-traf}}{\textit{wifi-sigl}})}$$

Here, $f\text{-}scale$ is a function to scale the network part factor to the same numerical range of the other factors to guarantee they have the same influence on the ECI. For the simplicity, we use 1 as the weight α for all the factors. In this way, the mag is able to be aware of the energy continuance index of the portable tablet device.

If an ECI degradation of the system is detected by the mag, it looks for alternative agents which have the same required functions as member agents but consume relative less energy in the set $AL\text{-}AG$ according to Definition 7. The relative energy consumption of a certain function is represented by $func\text{-}ec$ element of sva. For example, encoding video in high resolution consumes more resources and more power than encoding video in low resolution. In this study, the value of $func\text{-}ec$ is 1 for encoding video in full resolution and 0.8 for 80% resolution.

In case appropriate alternative agent exists, corresponding member agent is replaced by it to recover the ECI. I.e. to reduce the energy consumption of the portable device in order to recovery expected battery life. In this process, as a trade off to reducing the rate of energy consumption , usually agents perform required tasks with lower QoS are selected. However, the reduced QoS is still able to satisfy user requirements otherwise the reorganization will not be performed.

The experimental application described above is implemented using ADIPS/DASH agent

framework[13] and Java language. The live video server is running on a Unix-based desktop PC while the clients are implemented using tablets running Android OS. The server and clients are connected using wireless network.

Fig. 2.7: Snapshot of the experiment system for organization controlling architecture

2.4.4 Experiment Results

To evaluate the feasibility and effectiveness of our proposal, using the experimental application described above, two comparison experiments have been performed.

This experiment is set up to evaluate the feasibility of the proposal. As shown in Figure 2.7, a live video captured by a web camera of a PC is continuously streaming to an unplugged Android tablet using the multimedia communication service based on FMNM. The requirements to the service are live video streaming at over 15 fps and 0.25 times resolution. The tablet starts to use the service and plays the video stream when it has 80% battery charge and stops when the battery charge is less than 10%. The exactly identical experiments are repeated for the conventional method from previous work which only considers QoS parameters and proposed method of this research. The time spend in that operation is recorded as system battery life which is the main evaluation aspect. The local resource parameters such as battery voltage and battery charge are also collected during the service provision to help analyzing experiment results. The comparison of portable device battery life is shown in 2.1. The detailed information of battery is presented in 2.8 and 2.9, where the x-axis shows time and y-axis shows battery voltage, battery charge and ECI of the tablet ECI, respectively.

Table 2.1: Result of experiment 1: comparison of portable device battery life

	conventional method	our proposal
battery life (hours)	3.78	6.74
increment	n/a	78%

First of all, as shown in 2.1, comparing to the conventional method which does not consider energy consumption of the portable devices, our proposal significantly saves battery power to increase battery life of the device. In the experiment using conventional method, it takes 3.78 hours to drain the tablet battery power from 80% to 10% during the multimedia communication

service provision. By using our proposal, for providing the same service with maintaining the same QoS represented by video frame rate, the same battery energy lasts for 6.78 hours which is a 78% increased battery life.

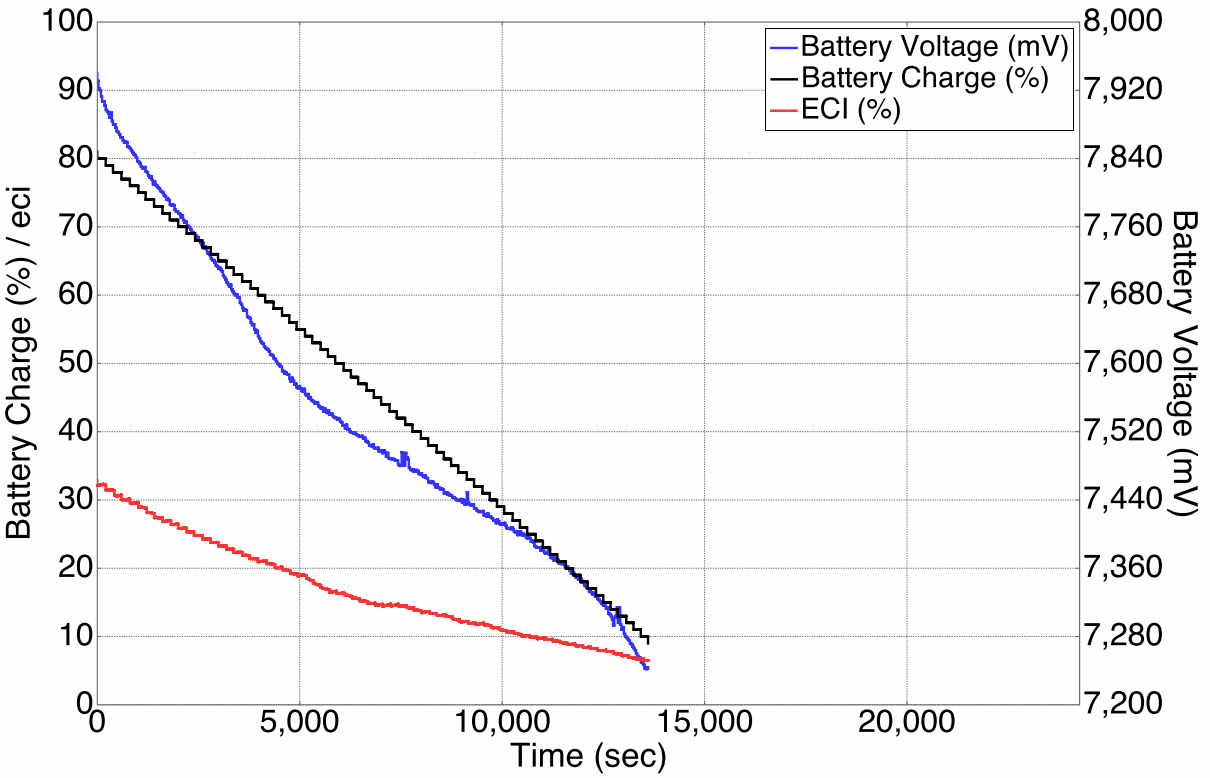


Fig. 2.8: Result of experiment: the ECI and battery information of conventional method

The detailed battery information and the ECI obtained from the experiment using the conventional method is shown in Fig. 2.8. During the service provision, with the decrement of the battery power which is represented by battery voltage, the battery charge decrease linearly and fast. The ECI decreases rapidly after service provision starts and keeps low during the experiment.

2.4.5 Evaluation

In the experiment using proposed method, we set the ECI threshold θ to 20 as the trigger to reorganize the service provision agent organization. After service provision starts, with the

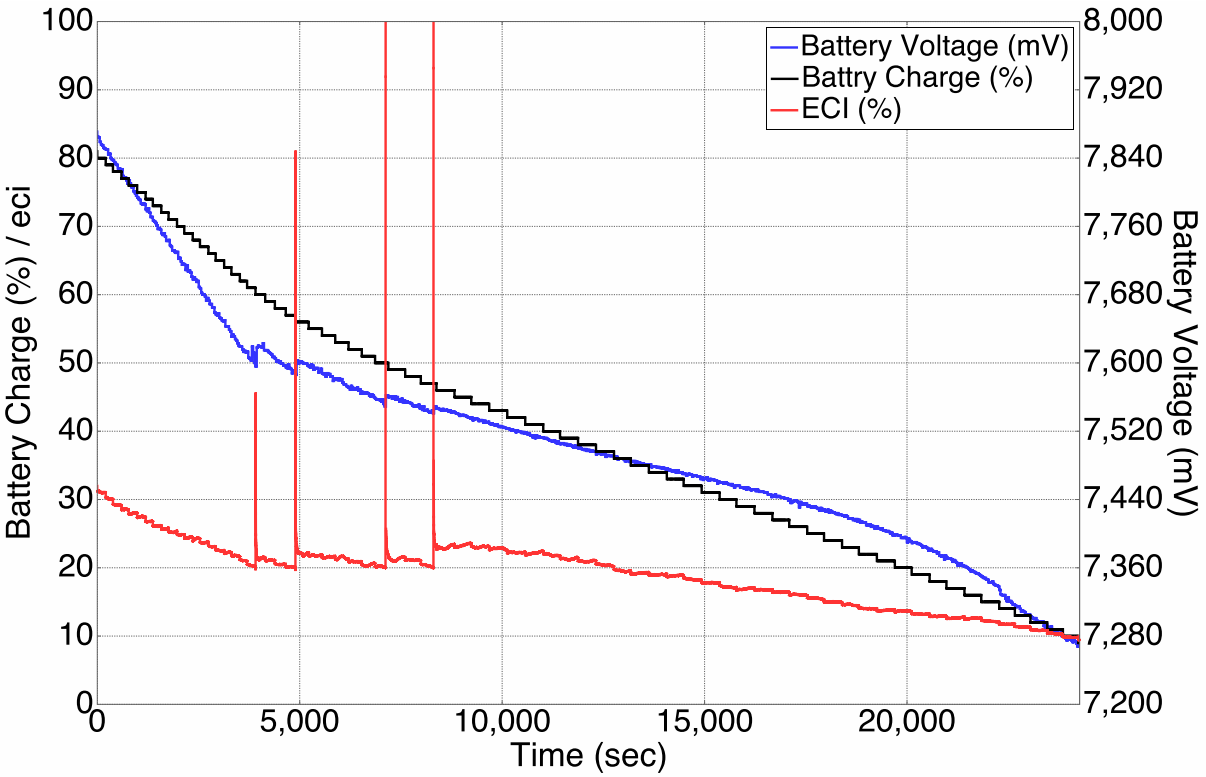


Fig. 2.9: Result of experiment: the ECI and battery information of proposed method

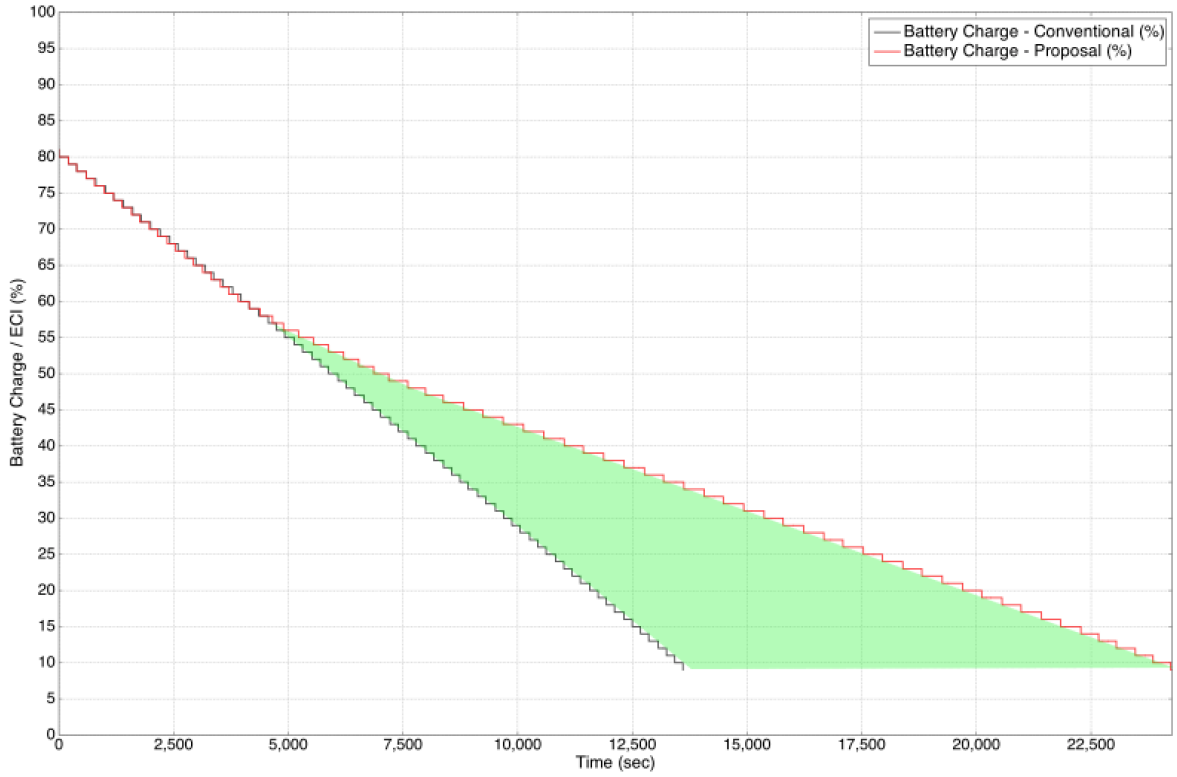


Fig. 2.10: Comparison of battery life

decrement of the battery voltage, the battery charge decreases linearly and the ECI decreases rapidly as the same as the experiment using convectional method.

However, when the ECI reaches the threshold, the reorganization is performed by replacing the video encoding agent in member agents using alternate agent which uses another video encoder. The new video encoding agent encodes video with 75% less resolution and requires less energy to provide the multimedia communication service with the required QoS. After the reorganization, a recovery of the battery voltage is confirmed from the result figure. The ECI recovers over the threshold and the battery charge decreases at a lower rate which is represented by the slope of the battery charge curve.

After the first reorganization, the ECI reaches the threshold again and system reorganizes one more time to reduce the energy consumption of the portable device. The reorganization repeats four times in the experiment to keep the ECI over the threshold to save the battery

energy of the tablet. When the ECI reaches the threshold at the fifth times, there is no available alternative agent, which is able to provide the same service with the required QoS, in the agent repository. Therefore no reorganization is performed and the ECI falls down under the threshold and battery charge decrease linearly.

It is noteworthy that in the process of reorganization, there is the trade-off between battery life and QoS of the provided service. The balance of that trade-off is controlled by the ECI threshold θ. The higher θ is, the earlier reorganization is performed i.e. the more energy consumption is reduced and more QoS is reduced and the other way round. In practice, service providers are encouraged to simulate multiple times to investigate the appropriate θ values for particular applications.

In this research, the result of experiment shows that the system functions correctly as designed. The rate of energy consumption is reduced by the control of the evolution mechanism autonomously before the battery charge of the tablet actually reaches low level. The QoS is maintained at the level which is able to satisfy requirements to the system. Therefore, proposed design is feasible for the system which contains portable devices as the example.

2.5 Summary

Facing the situation that network services are being used on portable devices under unstable environment with variable limitations, this research aimed to provide adequate services to the portable devices flexibly by considering the situations of users and their devices while enjoying the services. As described in this paper, we proposed a new design of energy-consumption-aware Evolutional Agent System for portable devices. The proposal considers platform specific parameters of local resources in addition to QoS parameters to control the service provision by autonomous agent reorganizations. Moreover, to evaluate the feasibility and the effectiveness of our proposal, we implemented a multimedia communication system based on the proposed design. By the experiment results, through comparison to the conventional method, we con-

firmed that our proposal is able to provide adequate multimedia communication services with minimum effect to user experiences.

Chapter 3 Protocol Design Method for Agent-based Evolutional Systems

3.1 Overview

Multi-agent systems are proved very effective for solving the problems related to distribute information processing and simulations in the past few years. Many researches make great efforts in those areas. For an instance, a multi-agent based approach for providing communication services in ubiquitous computing environment is proposed. Pu introduces an agent-based negotiation service with the combination of ontology. An agent-based computing model and its application for care-support services are proposed.

However, the development of multi-agent system still remains difficult. Addressing to that restriction, this research proposes a cooperation protocol design method to improve the efficiency of multi-agent system development. Based on the repository mechanism, reusable protocol templates can be applied for disparate multi-agent applications. Furthermore, a graphical design tool with automatic code generation functions is developed to support the proposal.

The remainder of this chapter is organized as follows. In Section 2, we introduce the related researches of protocol design for multi-agent systems and the repository-based agent framework. In Section 3, the proposed design method is described in details. The applications and evaluations are presented in Section 4. Finally, we conclude this paper in Section 5.

3.2 Related Works and Problems

3.2.1 Related Works

There are many different multi-agent system development methodologies have been proposed in order to support the development of agent systems. In the common sense of software engineering, multi-agent system development contains six different states in the whole lifecycle: Requirement Analysis, Design, Implementation, Testing, Deployment and Maintenance.Supported development lifecycle coverages are different depend on different multi-agent methodologies. While GAIA[28], MaSE[8], Prometheus and Troops[5] are supporting Requirement Analysis and Design phases, ADELFE[3], INGENIAS and PASSI are supporting one more phase to Implementation. However, the other phases of Testing, Deployment and Maintenance are out of support by those methodologies. There are also researche of comparing agent-oriented methodologies have been done.

3.2.2 Problems of Adopting Existing Method

One of the problems of those methodologies is the limitation to the specific agent implementation frameworks. Some methodologies are requiring specific agent natures to adapt. The simple comparison of agent nature requirements are shown in *Figure 2.2.* The cells with red background are the requiring agent natures that DASH dose not support. Those requirements make it impossible to adapt those methodologies to DASH to support multi-agent system development. Another problem is the lack of Implementation phase coverage of the other methodologies. The multi-agent development in DASH could not be fully supported without the Implementation phase supporting methodology.

3.3 Proposal

3.3.1 Proposal Overview

In this section, we would discuss our proposal of cooperation design method for repository-based agent framework in details. The proposed method would be described below by three different aspects which are related closely. *Figure 3.1* gives the overview of the proposal.

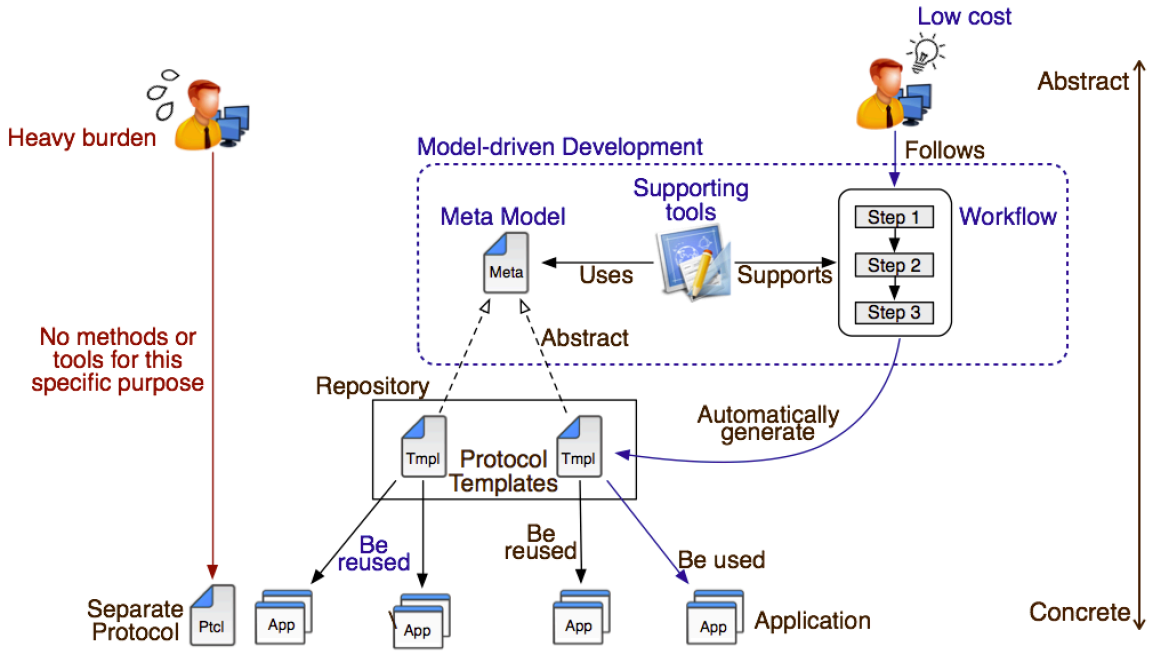


Fig. 3.1: Overview of the proposal

3.3.2 Repository-based Reusing Mechanism for Cooperation Protocol

For practical using the repository mechanism to improve the high level software reuse, we would like to introduce the protocol template mechanism. The protocol templates are the communication pattern abstract to well-used agent cooperation protocols. Developers of the protocol templates could make different protocol templates for different cooperation protocols and submit them to the repository to form a protocol library. The other multi-agent system developers who want to use the functions of the protocols just need to simply import the protocol templates they would like to use from the repository and implement the interfaces of the templates to use them.

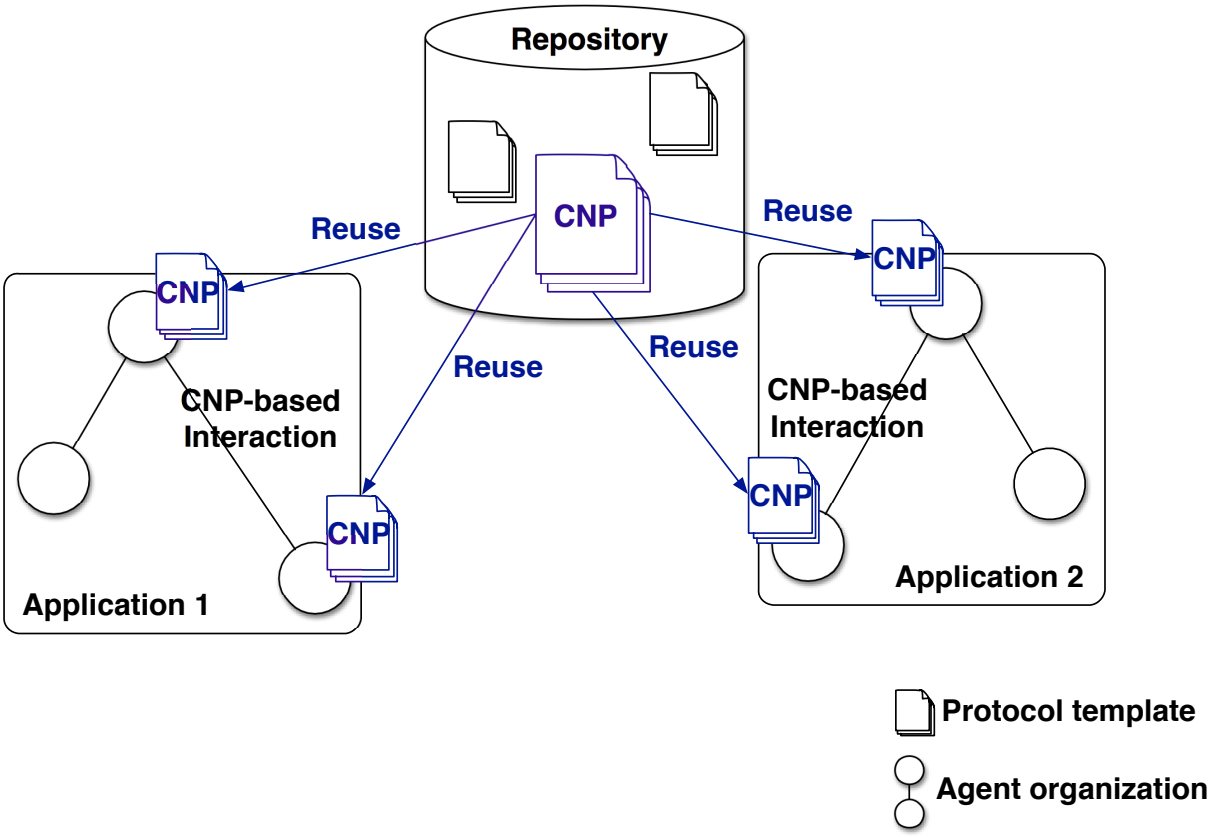


Fig. 3.2: Protocol templates using repository

Developer A (export developer) designed and implemented three different protocol templates and submitted them to the repository for future reuse. In the other side, developer B (non-

expert developer) who hopes use the functions of a cooperation protocol just imported desired protocol template to his multi-agent system project without re-design and re-implementation of it. To support the expert developers, we would like to introduce protocol design workflow for DASH, protocol template design method and supporting tools with protocol designer (expert developer) support functions which would be described separately in the following section 3.2, 3.3, and 4.2, respectively. While for the non-expert developers, we also prepared supporting functions for the protocol users which would be described in section 4.1 later.

3.3.3 Tools Supported Design Work Flow for Protocol Template

The protocol design workflow is the guideline and instructions for design protocol templates. The proposed workflow is described in *Figure 3.3*.

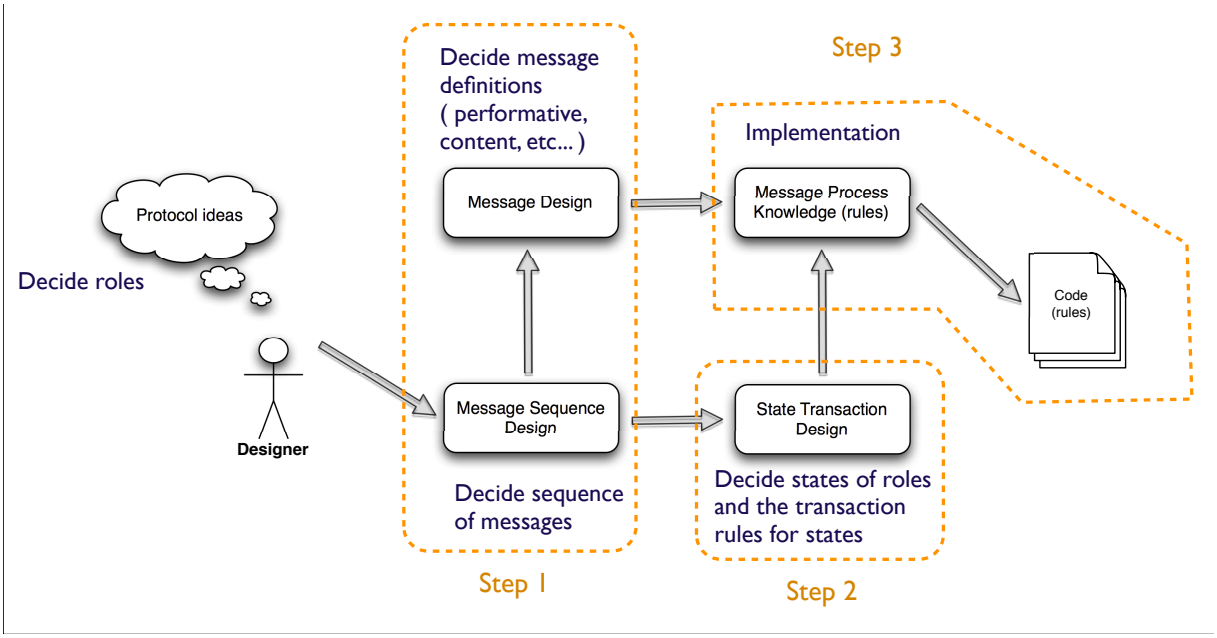


Fig. 3.3: Protocol Design Workflow for DASH

As shown in *Figure 3.3*, the protocol template design activity would be started from the **Protocol Ideas** of protocol designer, where the roles of the protocol should be decided. After that, the sequence of the messages which would be used in this protocol should be decided in

the next **Message Sequence Design** phase. Once the sequence of the messages are defined, the developer should give the detail definitions of the messages such as the performative, content, etc... in the **Message Design** phase. And also the states of each roles of the protocol and the transition rules between states should be decided in the **State Transition Design** phase following. While all those design phase finished, the developer would implement the designed protocol template in the phase of **Message Process Knowledge (rule)** to get the final code (rules). The proposed workflow has been divided into three different step as shown in *Figure 3.2.* Phase **Message Sequence Design** and **Message Design** are contained by **Step 1**, while **State Transition Design** phase in **Step 2** with the rest in **Step 3**. We would propose the design method follow those design steps in the following section.

Protocol Template Design Method

In this section, we would like to describe the proposed protocol template design method for the steps of the proposed protocol design workflow which has been introduced in last section. The products and supporting tools for each step is listed in *Table 3.1.*

Step	Product	Tool Support
1. Role and Sequence Design	Sequence Diagram	General UML Design Tool
2. State Design	DASH State Diagram	Proposed Graphical Design Tool
3. Implementation	Code (*.rset files)	Code Generation Function

Table 3.1: Steps, products and supporting tools of proposed design method

The **Sequence Diagram** would be the product of **Role and Sequence Design** step. And There are a number of general UML design tools to support the design of it. While for the **State Design** step, **DASH State Diagram** should be generated. We would introduce a graphical design tool which is designed for this purpose to support the design of this step in section 4.2. And for the **Implementation** step in which final codes are the product, the code generation

function of the graphical design tool would definitely help the developers. We would like to introduce each step in details in the following subsections.

Step 1, Role and Sequence Design

The roles and message sequences of the protocol template would be decided by developer in this step using common UML sequence diagrams with specific notations for the template characters. An example of Contract Net Protocol template has been shown in *Figure 3.4*.

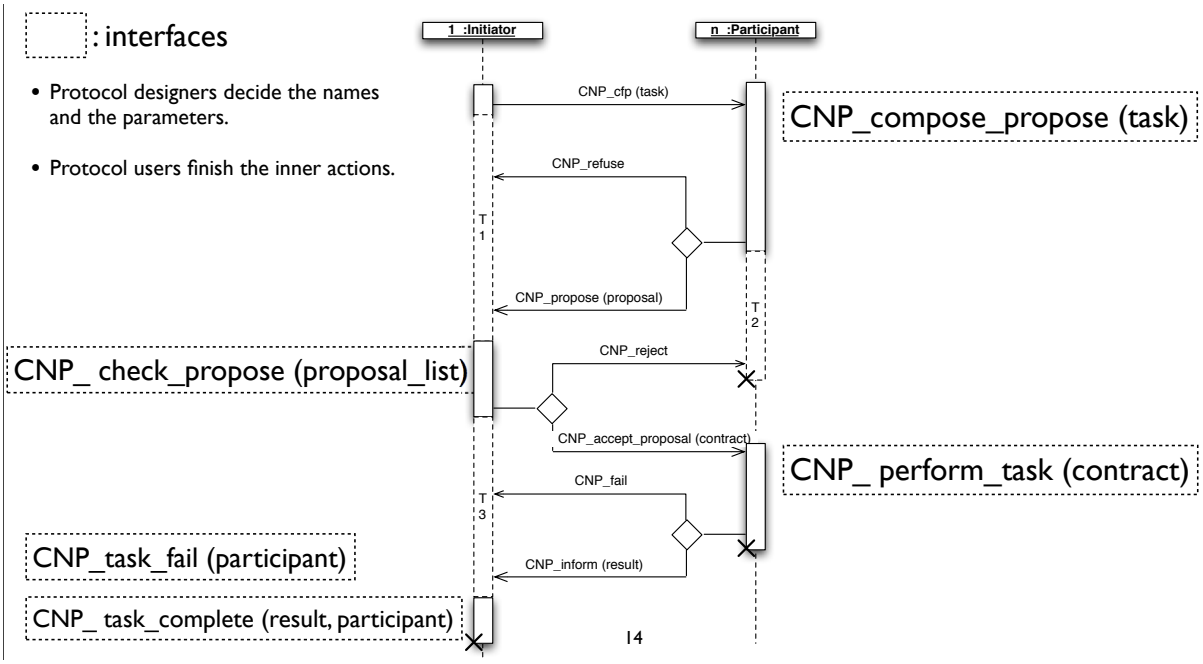


Fig. 3.4: Sequence Diagram

The specific nations for DASH protocol templates in the sequence diagram are surrounded with dashed lines in *Figure 3.4* which are called **interface**. The interface is just like the definition of functions, which are decided by the protocol designers in ways of the names and the parameters. The body of the interfaces would be left blank to the protocol users to finish the inner actions to meet their application domain specific requirements.

In the example of Contract Net Protocol template above, two different roles of Initiator and Participant communicate using seven different kind of type messages to archive the goals of the

protocol. Five interfaces with parameters have been also defined by the protocol developer for the protocol users to archive the application domain specific requirements. Once the sequence diagram is defined, the development could entry the next step of State Design which we will describe in next subsection.

Step 2, State Design

For each role of the protocol template, the DASH State Diagram should be designed in this step by the designer depend on the sequence diagram which has been designed in the last step. The DASH State Diagram is formed by six different basic components which are shown in *Figure 3.5.*

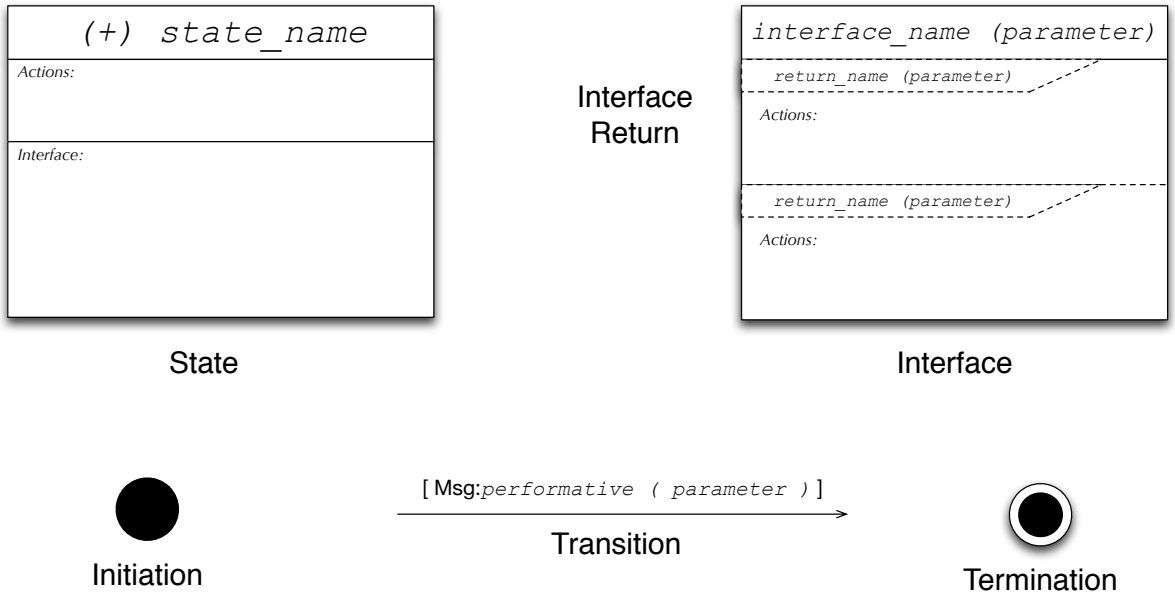


Fig. 3.5: DASH State Diagram Component

The DASH State Diagram would be always started from the **Initiation** state and ended to the **Termination** state. The else states of the roles would be described by the **State** part with the specific state names. The"+" in front of the state name decide if a new thread would be created for a particular state. And the Actions part of the state describe the actual actions

would be performed in the state. The state would contain one optional **Interface** which must be defined with a name and none or more parameters. The interface would be always invoked with the parameters after all the actions of the state has been performed by the system. One or more **Interface Return** components would be contained by an interface. The interface return components must be defined by interface return names and optional parameters and actions. Finally, all the states must be connected by the **Transition** with the optional message labeled.

With the basic components which have been described above, developers could easily compose them to construct complex DASH State Diagram. One example of the Role Participant of Contract Net Protocol template has been shown in *Figure 3.6.*

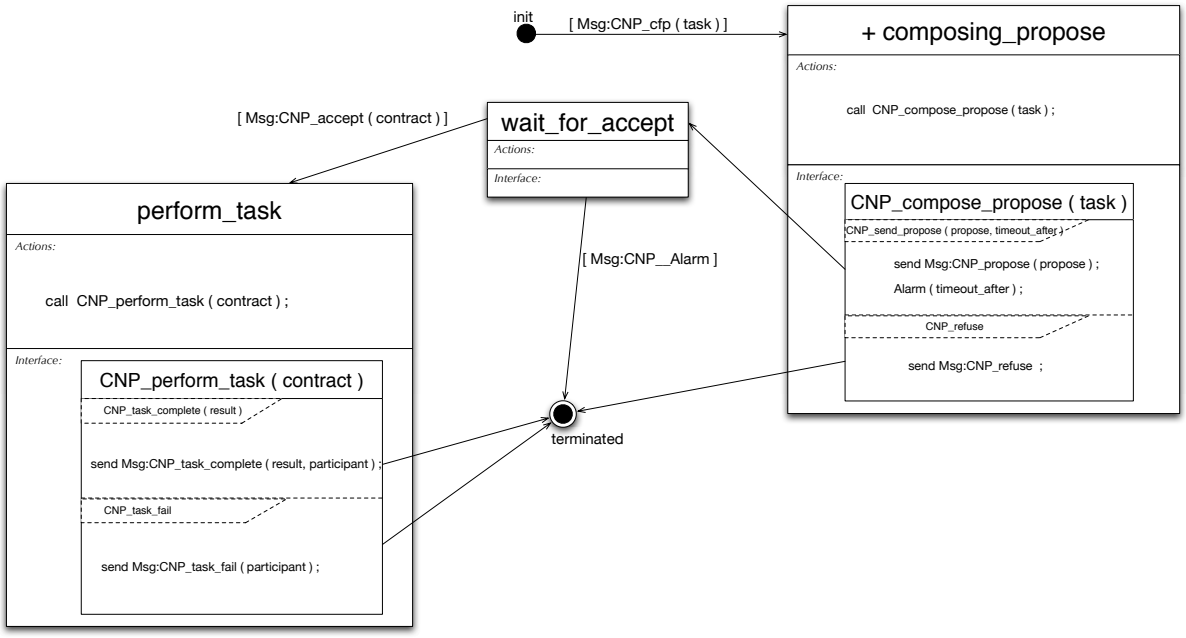


Fig. 3.6: DASH State Diagram of Role Participant of Contract Net Protocol Template

In the example there are three different states besides the initiation and the termination states. The initiation state would transfer to the *composing_propose* state in a new created thread on receiving a message with the performative of *CNP_cfp* and parameter of *task*. The interface *CNP_compose_propose* would be invoked in the state *composing_propose* to give the protocol template users chance to decide wether propose or not using interface return *CNP_send_propose*

or *CNP_refuse*. If user chooses to refuse, the message with performative of *CNP_refuse* would be sent and the thread turns into termination state. While the user chooses to propose, the message with performative of *CNP_propose* would be sent and an alarm would be set and the thread turns into the state of *waiting_for_accept*. If the alarm goes time out without receiving the accepted message, the thread goes to terminate. On receiving message with performative of *CNP_accept*, the thread transfers to the state of *perform_task* to perform the contracted tasks by invoking the interface of *CNP_perform_task*. Finally, after performing the tasks, the thread turns into termination.

As the example we have been explained above, the developers could use the components to construct any kind of DASH State Diagram to describe the state transitions of the roles of the protocol templates. Once this step has been finished, the developer could implement the designed transitions in the next step of Implementation which would be described in next subsection.

Step 3, Implementation

In this step developers should convent the designed DASH State Diagram into DASH rule set codes. To archive that goal, we would like to introduce a meta model of the DASH State Diagram so that the implementation activity could be described by the mapping between models and codes. The meta model of the DASH State Diagram has been shown in *Figure 3.7*.

According to the meta model, the **StateDiagram** class contains the **State** class and the **Transition** class as components. And the Transition class has State class type properties named "source" and "target" which points to the source state and the target state of the transition. **Interface** class with **InterfaceReturn** class in it with a component could be contained by the State class, which could also contain **Action** class as components.

Using the proposed meta model, the DASH State Diagram could be modeled and mapping to the DASH rule set codes. The mapping keeps one Transition object to one rule which has the

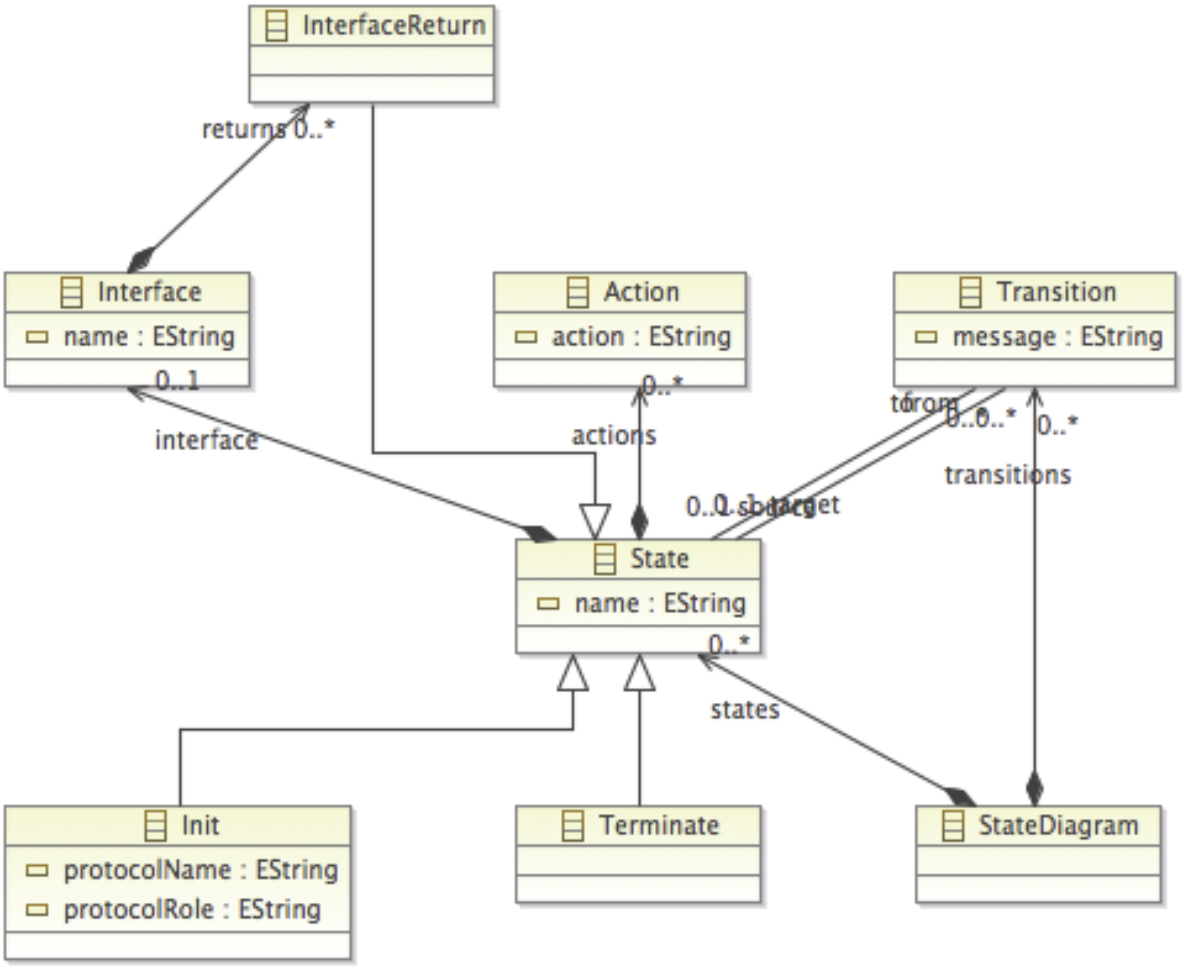


Fig. 3.7: Meta Model of the DASH State Diagram

conditions from the message property of the Transition object and the name of State object of source property. The actions of the rule are constructed by the Action properties of the target State object of the Transition object. And example of the State Diagram-model-code mapping has been shown in *Figure 3.8* and *Figure 3.9*.

The diagram-model mapping has been shown in *Figure 3.8* with the example of the part of Contract Net Protocol template. The transition from initiation to the *composing_propose* state has been mapped to the transition object with the source property which pointed to the state object with the name property of "init" and the target property which pointed to the state object with the name property of "composing_propose".

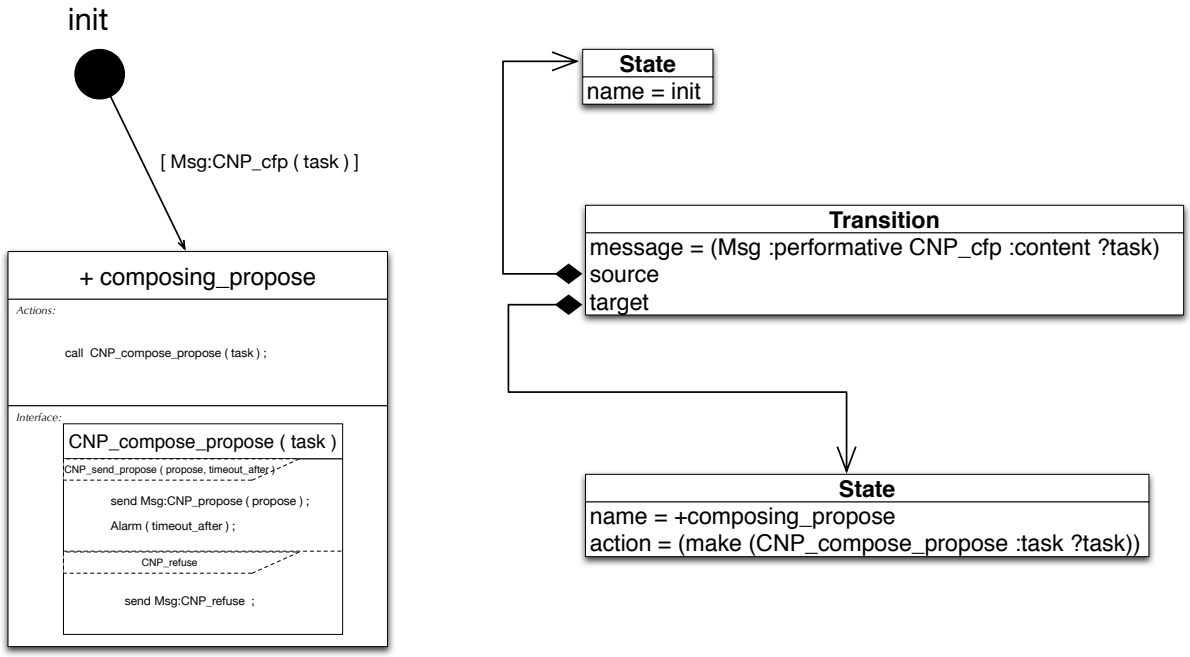


Fig. 3.8: DASH State Diagram to Model Mapping

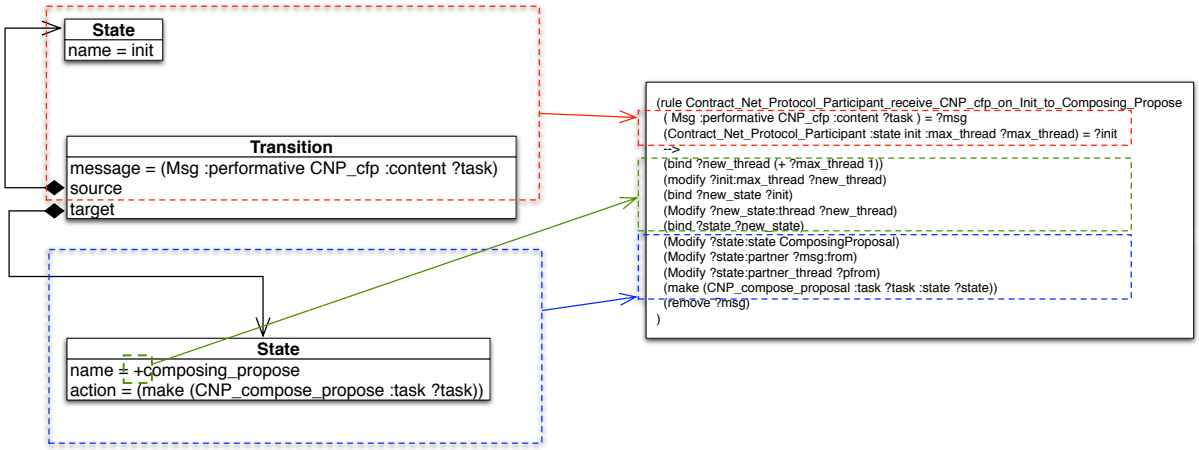


Fig. 3.9: DASH State Diagram Model to DASH Rule Set Code Mapping

The model-code mapping has been shown in *Figure 3.9* with the same example. The conditions of the rule would be constructed by the transition object and the source state of it. And the action part of the rule would be constructed by the action properties of the target state of the transition. The "+" notation in front of the name of the state would construct the codes of

creating new thread.

With the proposed diagram-model-code mapping in this step, the developers could easily implement the designed DASH State Diagram into reusable DASH rule set codes. Further more, we also provide the graphical design tool for the diagram design with the function of code generation to support the development of protocol templates which would be introduced in the next chapter.

3.4 Experiment and Evaluation

3.4.1 Experiment Overview

To validate the functionality and feasibility of our proposal, we developed a multi-agent applications. The agent communication protocols of the applications are designed and generated by the proposed protocol design method and tools.

A microgrid system is a private small-scale power grid, which is typically composed of distributed generation system (DGs), distributed storage devices (DSs), and loads. Usually a human operator is required for efficient and economical microgrid operation. This system aims a multi-agent system for autonomous microgrid operation by appending intelligent software agents to the components of a microgrid. Instead of a human operator, Microgrid Operation Control Center (MGOCC) agent collects information such as power supply and demand from the other power unit agents in the system and makes plans for operation. In this research, we focus on the protocol design aspect.

The CNP Template is used in the system for information exchange and coordination. The MGOCC agent plays role Initiator and the other power unit agents play role Participant of the CNP Template. The MGOCC agent broadcasts the CNP_cfp message to collect the power information such as power demand of a load. Then an operation plan is created according to the power information collected. The plan is assigned to the power unit agents as CNP task

contracts through CNP_accept_proposal messages. Participant agents perform the tasks and report the results to the MGOCC agent. In this way, the system performs appropriate operation autonomously.

3.4.2 Experiment Results

The snap shot of the running graphical design tool has been shown in *Figure 3.10* with the DASH State Diagram of Role Participant of the Contract Net Protocol template as an example.

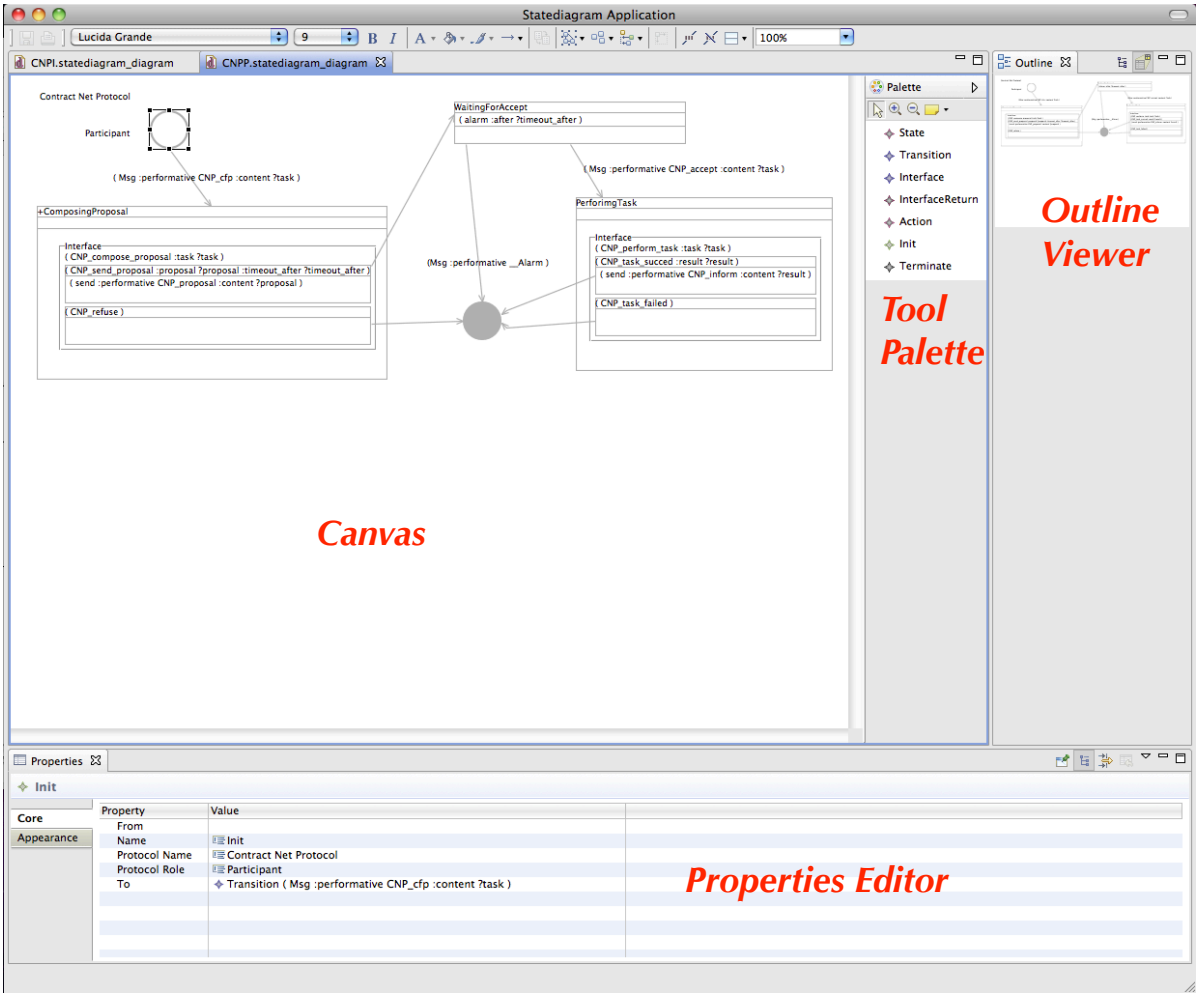


Fig. 3.10: User interface of the graphical design tool for the DASH State Diagram design

The tabbed **Canvas** has been placed in the main place of the tool in which user could design

the diagram by choose different component from the **Tool Palette** besides the canvas. At the bottom of the tool the **Properties Editor** could be used to easily edit the properties of the components. Finally, the right side **Outline Viewer** could give the developer a full perspective of the whole diagram. The detailed images of Tool Palette and Properties Editor could be seen in *Figure 3.11*.

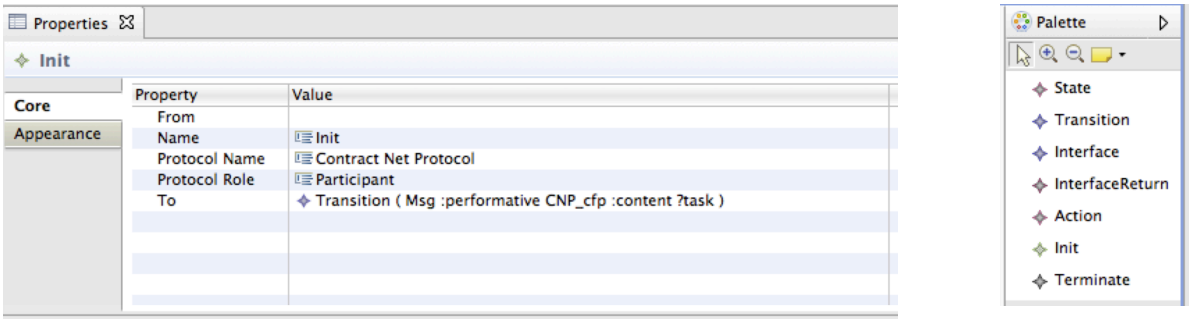


Fig. 3.11: Detailed images of Tool Palette and Properties Editor

Once the design of the DASH State Diagram has been finished, the developers could use the "Generate Dash Rule Set" menu from the popup menu of the init component of the diagram to generate the DASH rule set codes. The menu has been shown in *Figure 3.12*

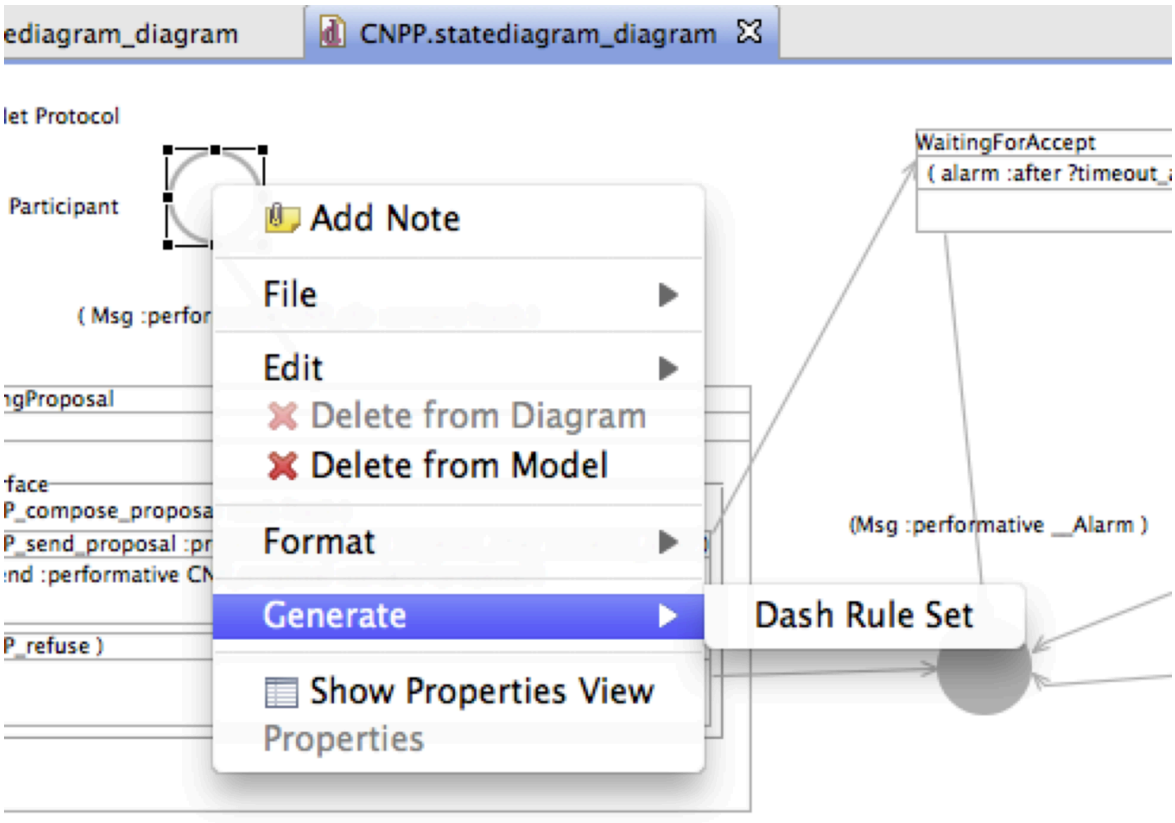


Fig. 3.12: "Generate Dash Rule Set" menu

The example of output DASH rule set code file could be seen in *Figure 3.13.* And the whole content of the file could be found in Appendix B.

Contract Net Protocol (CNP) is a well known task sharing protocol that is used for task allocation in multi-agent systems and consists of a collection of nodes or software agents that form the contract net. Each node on the network can, at different times or for different tasks be

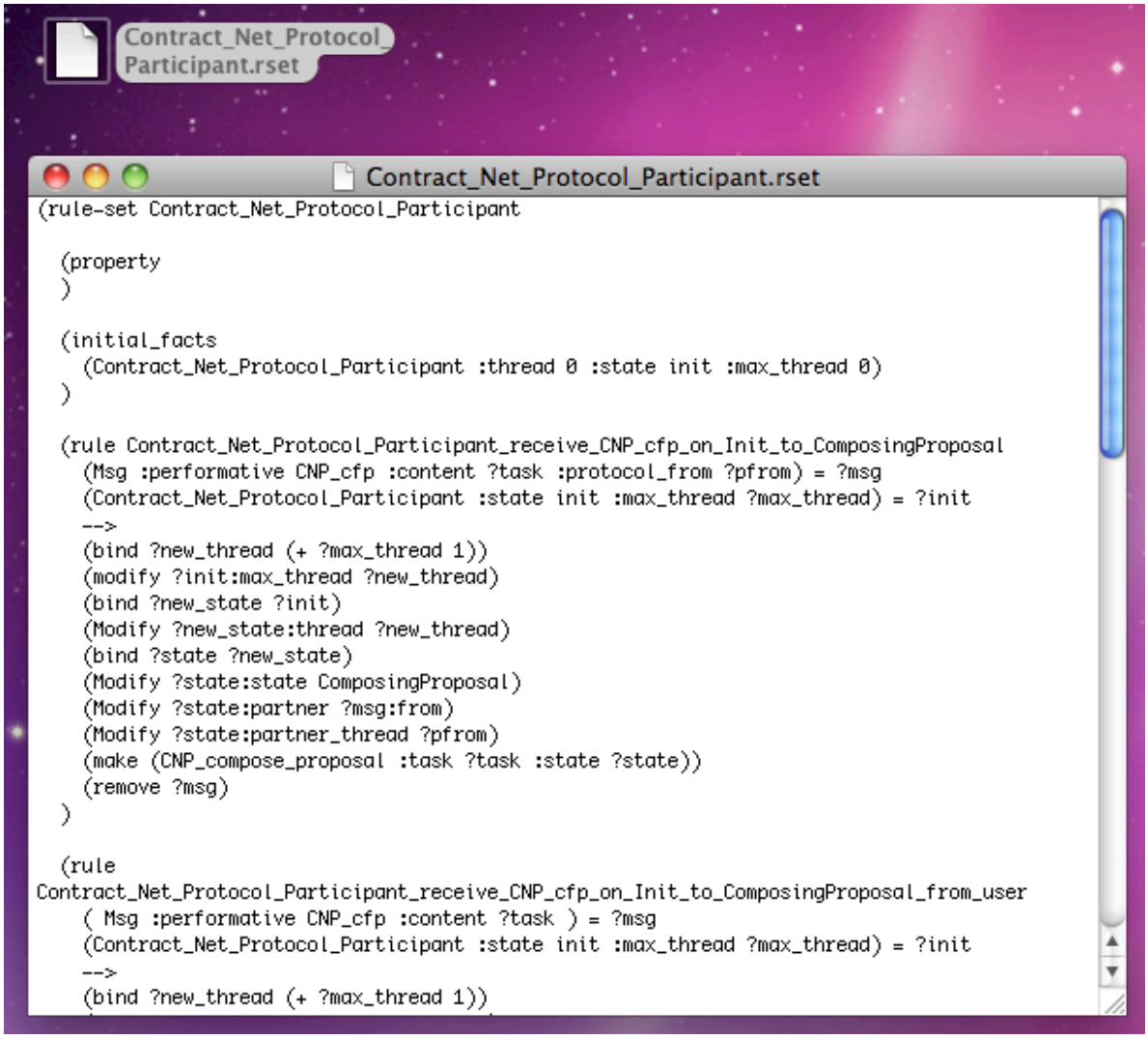


Fig. 3.13: DASH rule set code file

a manager or a contractor.*Figure 3.14* gives a abstract of the Contract Net Protocol.

The DASH State Diagrams which has been designed using the proposed design method and DASH Rule Set files which has been generated by the supporting tool has shown in *Figure 3.15.*

The new implemented MicroGrid Control System using the generated protocol templates has been running well as designed, which verified the validity of the generated codes. The snap of running system has been shown in *Figure 3.16.*

Comparing to the old implementation which were all written by developers, the experiment

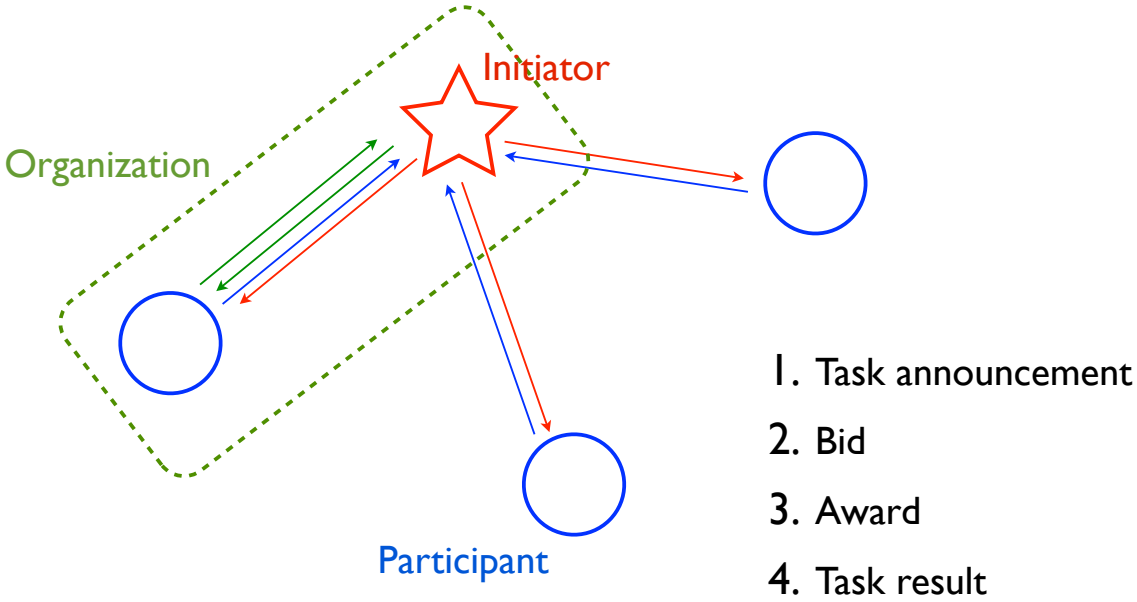


Fig. 3.14: Contract Net Protocol

implementation using the proposed design method and supporting tool which had only small parts need to be written by developers. The example of code comparisons has been shown in *Figure 3.17*.

3.4.3 Evaluation

To evaluate the coding work reducing effect of the proposed design method, the comparison of lines of code between the old implementation and proposed design method has been performed. The result of the comparison could be seen in *Table 3.2*.

According to the result table, about half coding work has been reduced using the proposed design method comparing to the old implementation. The problem of heavy coding work had been solved by the proposed design method.

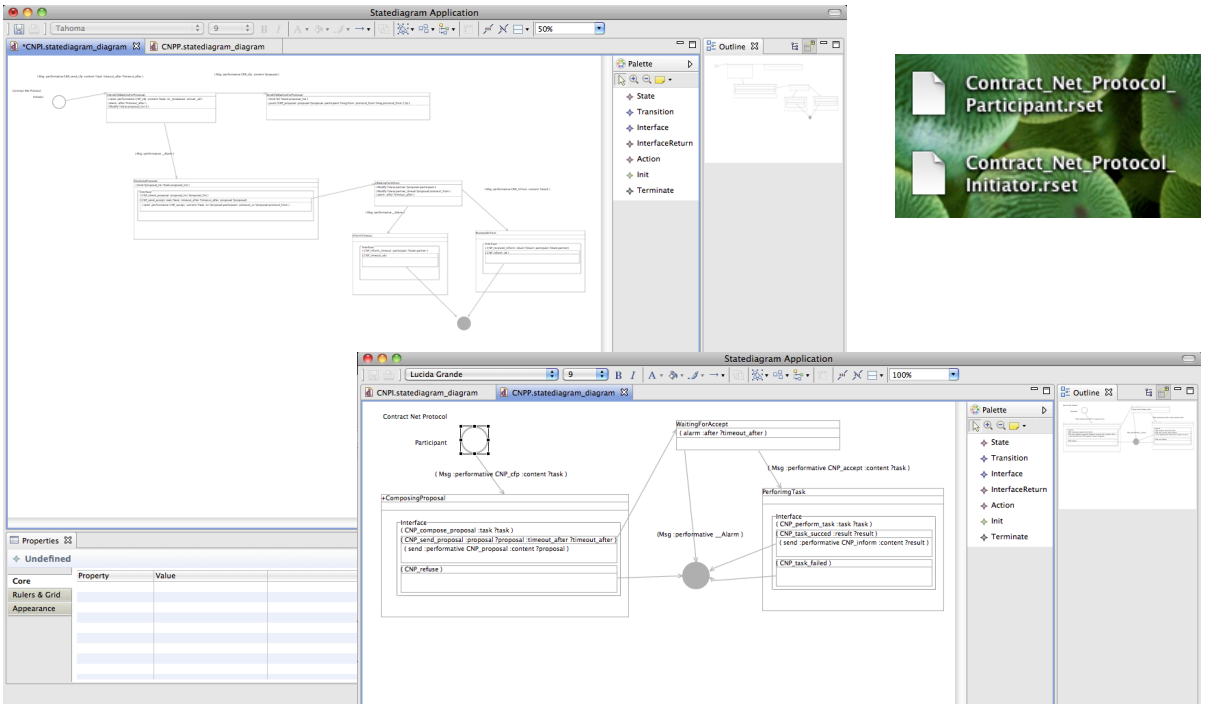


Fig. 3.15: DASH State Diagrams and DASH Rule Set files

	MGOCC Agent	LOAD Agent	DG Agent
Lines of codes of old implementation	298 lines	114 lines	136 lines
Lines of codes of proposed method	129 lines	58 lines	62 lines
Coding work reduction	56.7%	49.1%	54.4%

Table 3.2: The result of the comparison of lines of code between the old implementation and proposed design method

3.5 Summary

Research Purpose

This research has been done to support the multi-agent system development in DASH.

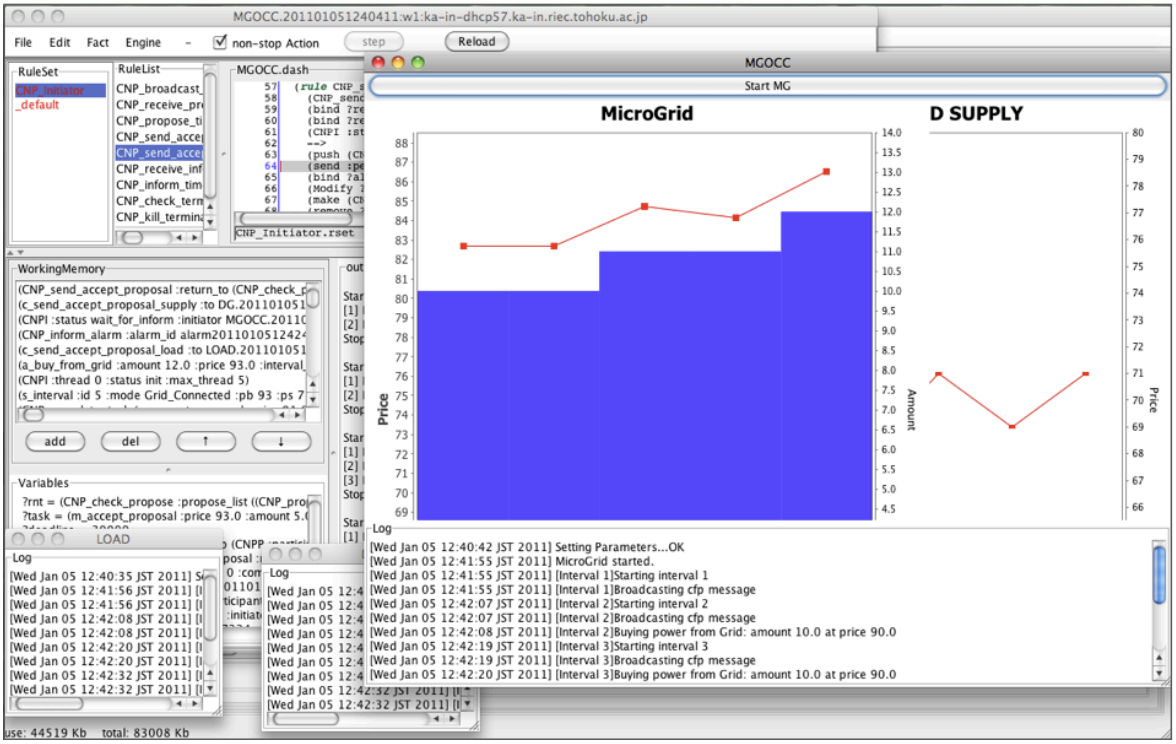


Fig. 3.16: MicroGrid Control System using the generated protocol templates

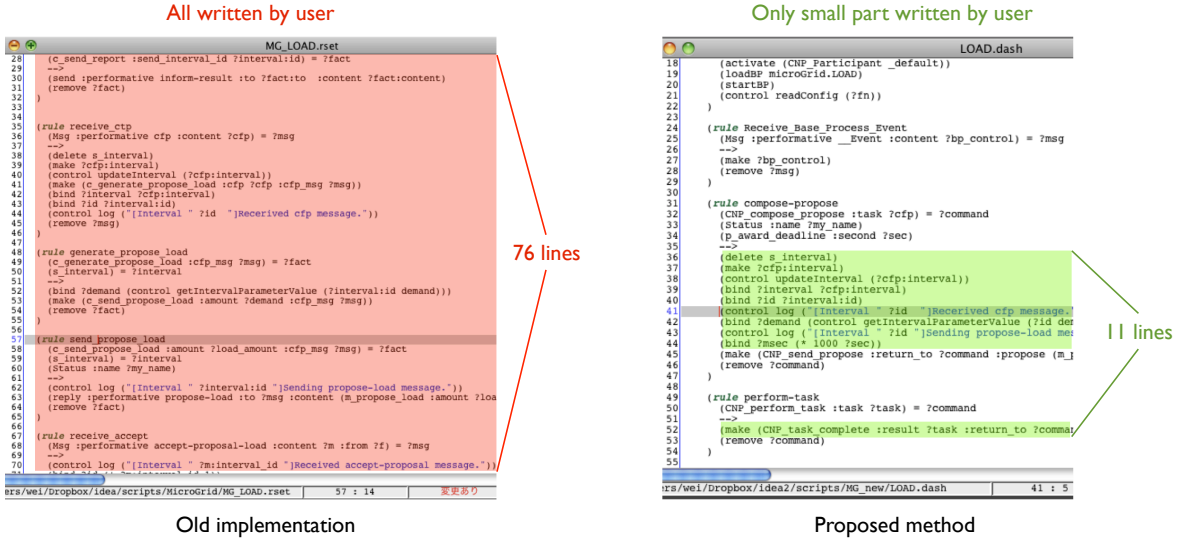


Fig. 3.17: Code comparisons between old implementations and proposed method

Problem

The problem of DASH is the lack of easy-to-use, effective development methodology.

1. Difficult for non-expert developers
2. Heavy coding work due to the low level reusability.

Proposal

To solve that problem of DASH, the proposal of cooperation protocol design method for repository-based agent framework has been proposed.

Result

1. The supporting functions and the proposed design method successfully reduced the difficulty of design and implementation of protocol templates for repository-based multi-agent framework.
2. The proposed design method effectively reduced coding work by practical using design level software reuse.

Chapter 4 Autonomous Knowledge Construction Method for Evolutional Control

4.1 Overview

Toward the effective realisation of Evolutional System with deterioration resistant property, this chapter describes the issues about extracting and construction of control knowledge for agent organization controlling in EAS. The challenge appears as that how to deciding the appropriate timing of performing Evolutional Control before actual undesired changes arise without extra design burden. Here, we proposes autonomous knowledge construction method for Evolutional Control.

In the other words, by utilising machine learning technique to the monitoring results of system activity property, the collection and management of control knowledge, which are performed manually in present, are able to be finished automatically. And simulation experiment proves the feasibility of our proposal. This is a significant result toward to the design and development of EAS for practice usages.

The remains of this chapter is as the follows. Section 2 discusses the related works of this chapter. Section 3 describes the proposal in details. Following section 4 describes and analysis the simulation experiment and results. Finally section 5 gives a summary to the chapter.

4.2 Related Works and Problems

In this section, the related works are reviewed carefully. Also, the problems of adopting existing approaches are discussed after that.

4.2.1 Related Works

The related works can be categorized into two disparate mainstreams. One of them is to directly monitor and perceiving the system characters at system global level and to react against to the monitoring result. Naturally, this kind of approach is widely used by self-adaptive systems. For an instance, an extension[15] to UML Use Case[9][1] is introduced for modeling the situation described above. In this approach, some predefined conditions are stored in the system while designing. The whole system is continuously monitored in the aspects of those predefined conditions while running. Right after any of the conditions are met, according predefined actions are performed to try to recover the system from failure. This is a quite straight forward approach which usually is able to achieve the original goal of the system. That is also the reason of wide adaption of this approach in the self-adaptive community. However, obviously this approach lacks the ability to prevent any effect of undesired situations of the system before those situations actually happen.

On the other hand, some of our previous research show unique direction of solutions to the problems. We turn our focus from the global measurement of the system to the relationship between the behavior of the system and it. This approach is described with according example application[25]. In the previous research, a multimedia communication system is used as the example application. The QoS of the multimedia communication service is the global measurement of the experimental system. Unlike the conventional method that monitors the QoS and acts after the QoS falls, our approach focuses on the system behaviors and parameters those effect QoS possibly. Those behaviors and parameters show the potential ability of maintain current QoS. By figuring out the relationship between the QoS(i.e. system global measurement)

and the related system behaviors and parameters(i.e. system potential measurement), it is able to foreseen the QoS failure of the system through perceiving those behaviors and parameters before QoS failure actually happens. Therefore it is able to be prevented by performing recovering action(e.g. tuning the system and/or re-organizing) before QoS fails. The possibility of applying this approach is proved by our previous experiment. Nevertheless, to understand the relationship between system global measurement and potential measurement relies on the application domain knowledge and is extremely difficult even for the application domain experts.

4.2.2 Problems of Adopting Existing Method

Towards the effective realization of the Evolutional System with the deterioration resistance property, the adoption of existing methods is discussed. However, we notice certain problems for the existing method mentioned in last section. For the approach of direct perceiving and reacting, the lack of the ability to avoid deterioration of systems is unacceptable. Because of that the global measurement of systems usually appears random, it is hard to predict for avoiding system deterioration. This is also the main reason of the absent of the ability for avoiding system failure of self-adaptive systems. Even it is possible to recover the system itself after certain system failure, this approach is not suitable for the systems those cannot afford any kind of system failure(mission critical systems, e.g. financial systems and medicine related systems). For the other approach which is presented in our previous research, it is possible to prevent certain system failure before the effect appears. However the difficulty of explaining the relationship between the system measurement and system behavior parameters restricts the applying of this approach. Those kind of hidden relationship in a complex system such as multi-agent systems is extremely hard to be discovered manually. The developers are forced to construct those kind of knowledge manually with the extra burden and difficulty.

To improve this situation, we propose the autonomous knowledge construction method for evolutional control. The details of the proposal are described in the following section.

4.3 Proposal

4.3.1 Proposal Overview

The proposal of autonomous knowledge construction method for evolutional control can be divided into two solutions. The first one is indirect estimation of system global measurement. Instead of monitoring system global measurement directly, this solution uses the system local behavior parameters to estimate the system global measurement.

The second one is knowledge construction utilizing machine learning. Instead of manually discovering the relationship between system global measurement and local parameters, this solution utilizes machine learning technology to discover the hidden knowledge automatically.

4.3.2 Indirect Estimation of System Global Measurement

The details of the indirect estimation of system global measurement is described in this section. The main idea of this solution is shown in Figure 4.1.

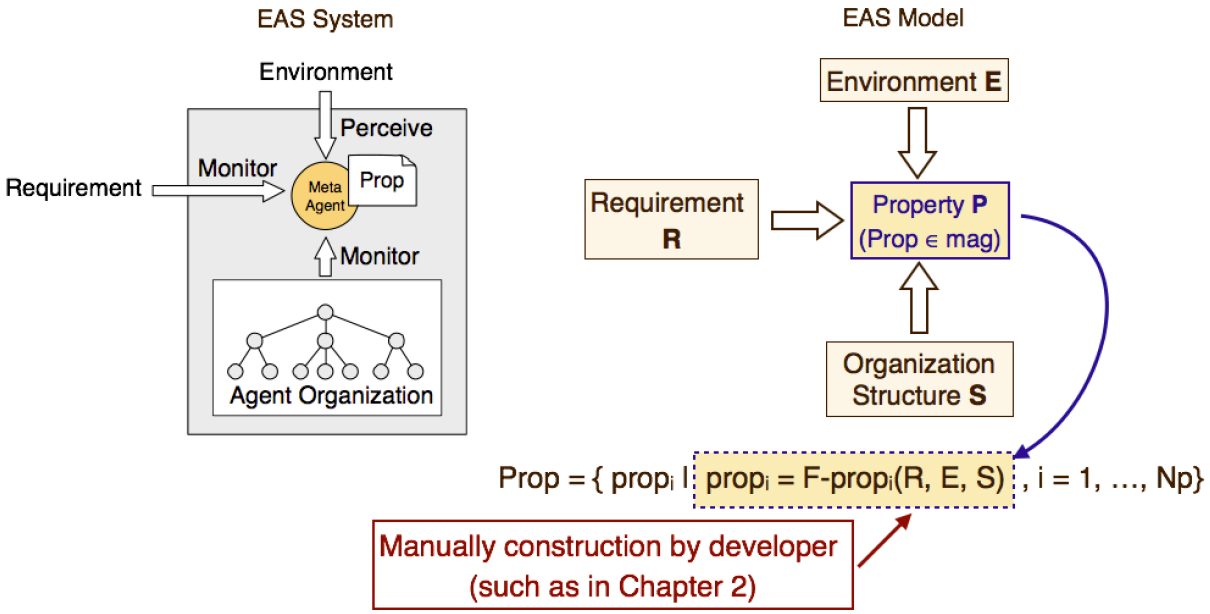


Fig. 4.1: Indirect estimation of system global measurement

To explain this solution, there are two very important facts about multi-agent system must be cleared. The first one is that although the most of multi-agent systems are barely complicated system, the single agent in the system remains relative simple. This also is one of the motivations of the multi-agent systems. For engineering large scaled complex systems, using top-down approach to break the whole system down to simple software agents makes it becomes relative easier to develop and to maintain. For that reason, usually single agents in multi-agent systems behaves following simple rules to achieve their goals. Those rules are written by human beings and are of course readable to human beings. It becomes possible to predict the behavior of a single agent of the whole system once knowing the rules it following and the situation it facing.

Nevertheless, even with the well regulated single agent behavior, the global measurement of the system still appears random. This is because of the second fact about the multi-agent systems, which is that the whole system level behavior of multi-agent systems is decided by complicated interactions among member agents. In other words, the system level performance and behavior of multi-agent system is decided dynamically by the unpredictable interactions of member agents autonomously. That means, there is nearly no way to predict or foresee the global measurement of the multi-agent systems by profiling itself.

In the EAS model described in the last chapter, the function for calculating activity properties of EAS is introduced. The function using requirement R, Environment E, and Organization Structure S as parameters to calculate Activity Property Prop. The reason is the fact that activity properties of the EAS are usually affected by the requirement, environment, and the EAS itself. Developers are only required to provide the implementation of the function to calculate the activity properties and to choose one from them as the measurement of the Evolution Control. Then the application system will be constructed as defined in the EAS model. The function is manually constructed by developers using their knowledge and experiences about the application domain. This brings noticeable burden to developers.

4.3.3 Knowledge Construction Utilizing Machine Learning

As discussed in last section, to extract the relationship knowledge between system global measurement and local parameters is the critical point for Evolutional Systems. However, to finish it manually is an extremely difficult job. Specifically, for the large scaled systems, the amount of information of local parameter overwhelms the capacity of human can deal with at the same time. Not mention to the confusing interactions among member agents. Even for experienced experts fulfilled with domain related knowledge, the task appears ultimate difficult.

On the other hand, machine learning technology shows another possibility for solutions. It is widely used for discovering hidden knowledge among a large amount of data. Therefor in this research, we decide to utilize machine learning technology for extraction and construction of control knowledge. The knowledge construction architecture utilizing machine learning is shown in Figure 4.2.

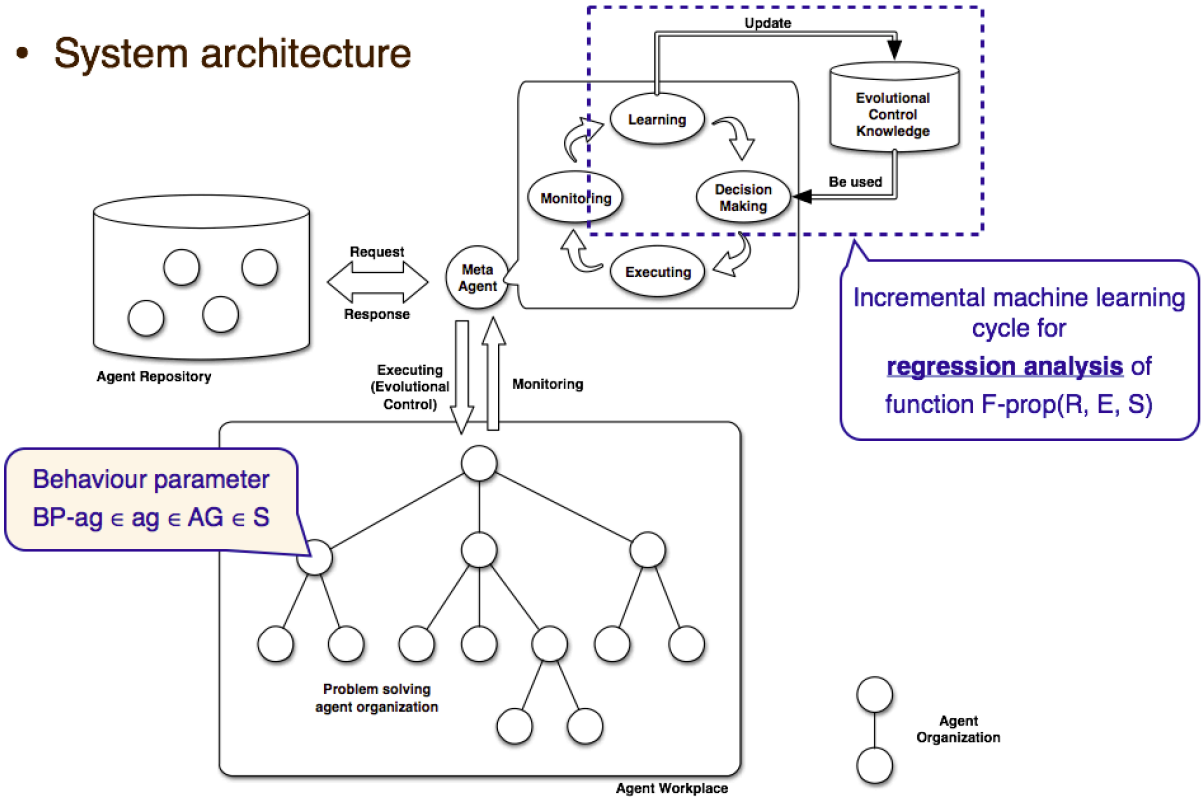


Fig. 4.2: Knowledge construction architecture utilizing machine learning

In our previous research, the meta agent presents the Evolution Mechanism, which monitors the agent organization and control the reorganization of it. After obtains member agent local parameters during monitoring phase, there is a newly appended learning phase follows. In the learning phase, all the data about local parameter and global measurement are used by machine learning engine to update and maintain the Evolutional Control Knowledge. Regression analysis is used for finding the relationship between activity properties and requirement, environment, and structure of EAS. The incremental machine learning cycle is utilized to construct the knowledge of implementation of the function of calculating activity properties. After the learning phase, in the decision making phase, the updated Evolutional Control Knowledge is used to decide whether to reorganize. The flow chart of decision making phase is shown in Figure 4.3.

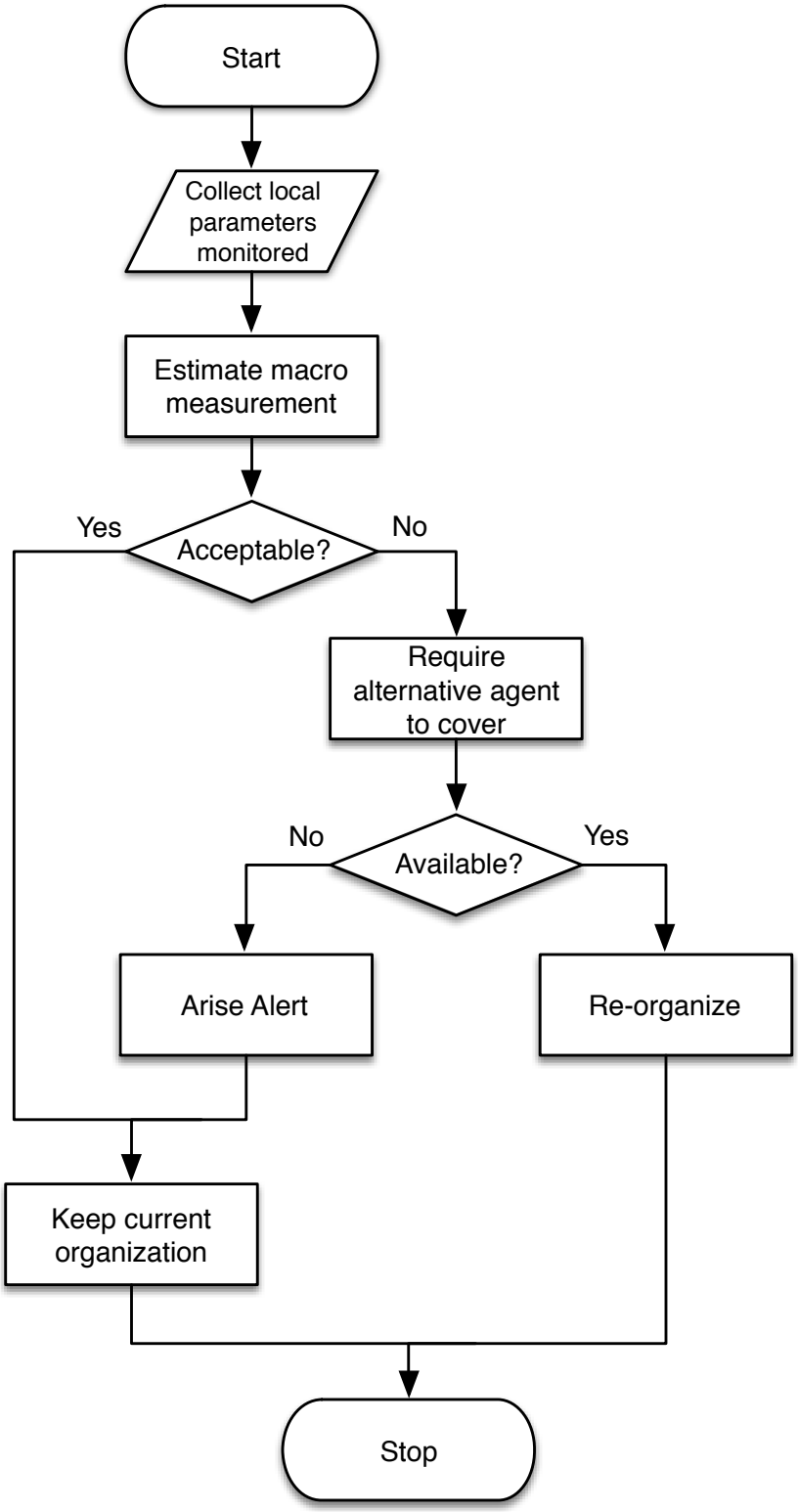


Fig. 4.3: Flow chart of decision making process of Meta-Agent

Right after the process starts, the data of local parameters is collected for the estimation of global measurement using the same machine learning engine. If the result of estimation is acceptable, then keep the current organization and terminate the process. If the result is not acceptable, then alternative agent is required to cover this deterioration. The meta agent reorganize the organization in the case of that alternative agent is found. If these is no available alternative agent, the meta agent arises an alter for human administrator and keeps the current organization and terminate the process.

4.4 Experiment and Evaluation

4.4.1 Experiment Overview

To verify our proposal, an experiment is performed. The overview of the experiment is shown as the follows.

Experiment objectives

To verify if proposed knowledge construction method is able to autonomously accumulate control knowledge to avoid system failure.

Experiment scenarios

Experiment method

In this experiment, an implementation of a agent-based MicroGrid control simulator using proposed autonomous knowledge construction method is performed at first. In the experiment system, power generators are controlled by agent to balance power supply and power demand in the grid and to prevent power overload failure. Experiment sys-

tem is supposed to autonomously increase power supply before possible power overload failure to avoid it.

Object of comparison

Conventional system which simply increase power supply all the time.

Parameter of comparison

Times of power overload failure and amount of wasted power.

4.4.2 Experiment System Design

In general, a MicroGrid operates as the figure following.

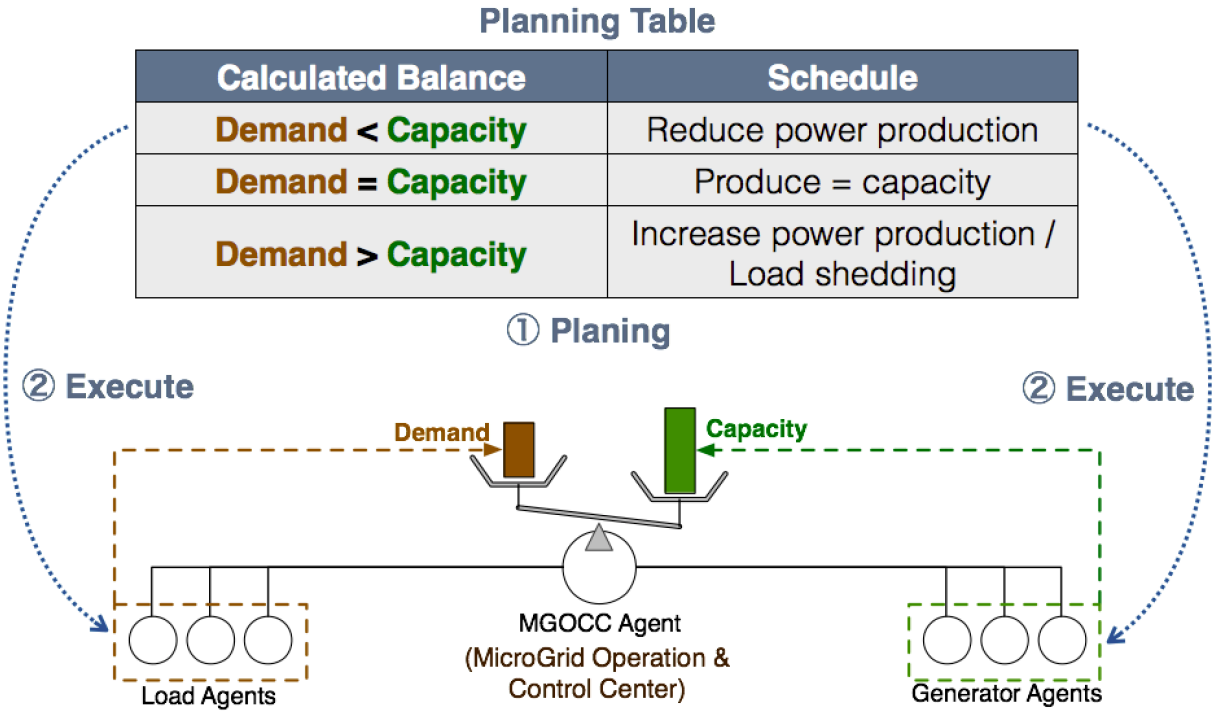


Fig. 4.4: General MicroGrid operation

The system has a control center called MGOCC (MicroGrid Operation and Control Center), which collects information from power loads and generators and makes plans to be executed by the them. Power loads inform the MGOCC the power demand, while generators inform the

capacity of supply. Then the MGOCC makes plan according to the planning table to maintain power balance in the grid by reducing power production in the case of low demand or increasing power production in the case high demand. Once the plan is made, it is executed by the loads and generators. All the processes are performed in cycle, which are repeated through microgrid operation.

Another important character of MicroGrid is the procedure of operation as showing in this figure.

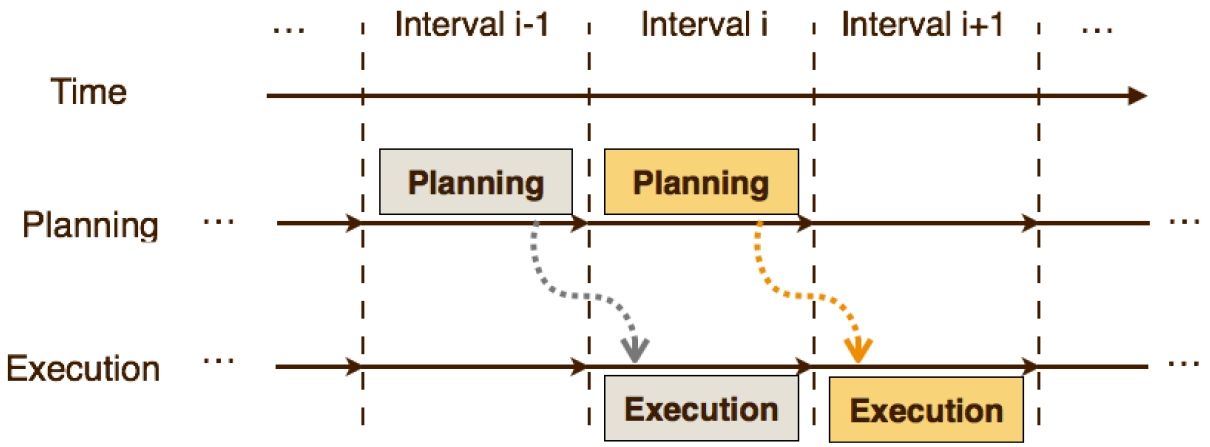


Fig. 4.5: Procedure of MicroGrid operation

To make sure the control to the power units in the grid is constant, the planing and execution are performed in parallel. This is realized by dividing the grid operation into several intervals of time. The plan made in interval i is executed in interval i+1 as shown in the figure. So a whole operation cycle is during 2 intervals. While the execution of current plan, the plan for next interval is just being made.

In the ideal world, using the strategy described above, the grid is always able to keep the power balance. That means the generators are controlled to produces just the right amount of power which load consumes. However, in the real world it is not the case. To make the simulation more real and practical, a price-based adjustment is made. All the generator has certain cost for produce power. The information of the cost is also collected by the MGOCC to

decide the price of the power while selling it to loads. The power price is the average cost of power generator produced.

The important point is, loads are able to decide their actual power consumption depending on the power price and their budget. This actual consumption is not needed to be the same as the scheduled consumption, which the load informed MGOCC. The budget of load is decided by the power price in last interval. In other words, if power price become expensive, loads are free to reduce purchasing. On the other hand, loads is possible to consumes more power than the amount they informed MGOCC if the power price turns out cheap. In result, this price-based adjustment introduces the danger of power grid overload failure into the simulation experiment.

In the case of power demand overload, the system is designed to increase power supply in two ways. The design of the experiment system is shown in Figure 4.6.

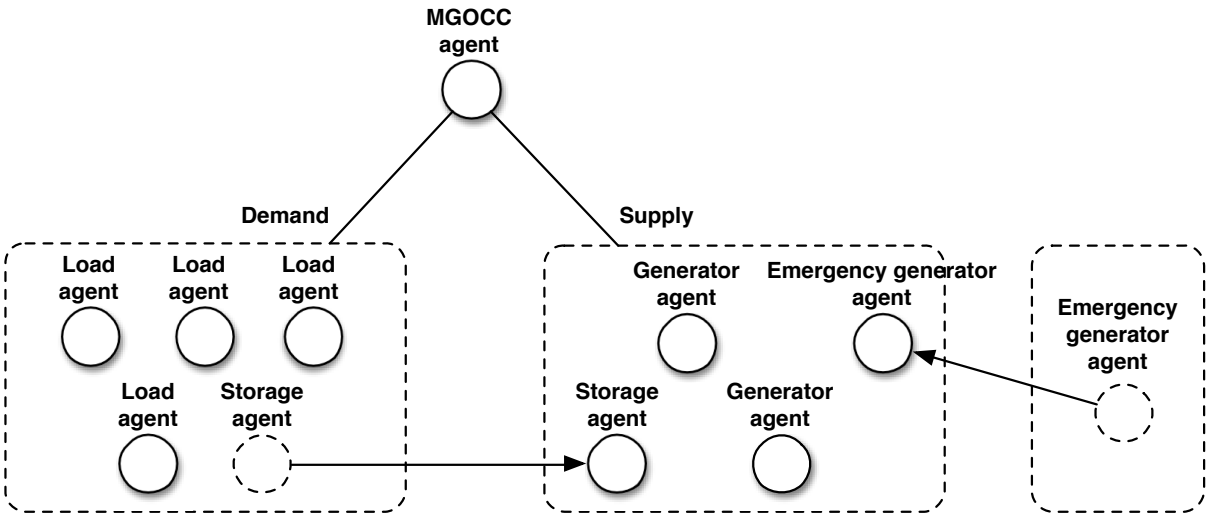


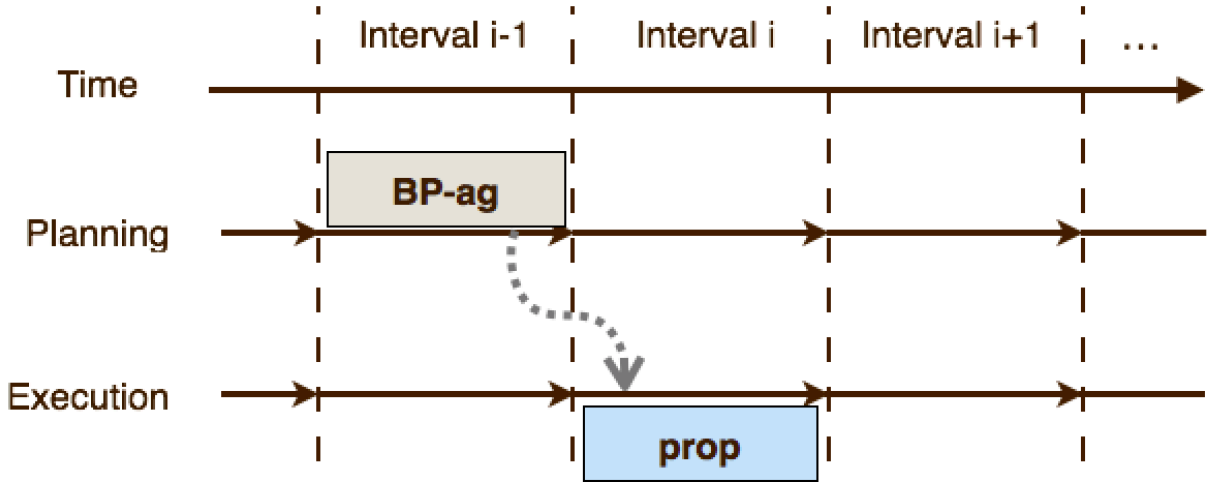
Fig. 4.6: Design of experiment system

There are three types of agents in the system. The MGOCC agent is the controller of the grid system. It collects information, calculates the power balance and make schedule for the other units. The demand type of agents includes Load agent and Storage agent. The supply type of agents includes Generator agent, Storage agent and Emergency generator agent. In the case of power demand overload, the system reorganized in two ways as following:

1. Using storage agent as power supply.

2. Initiating emergency generator agent as power supply.

The learning process is shown in the next figure.



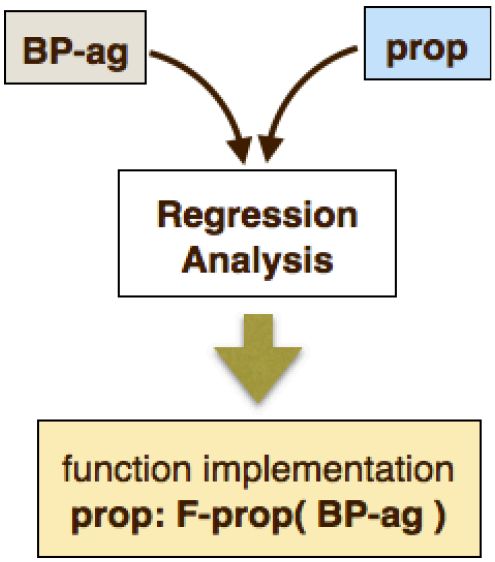
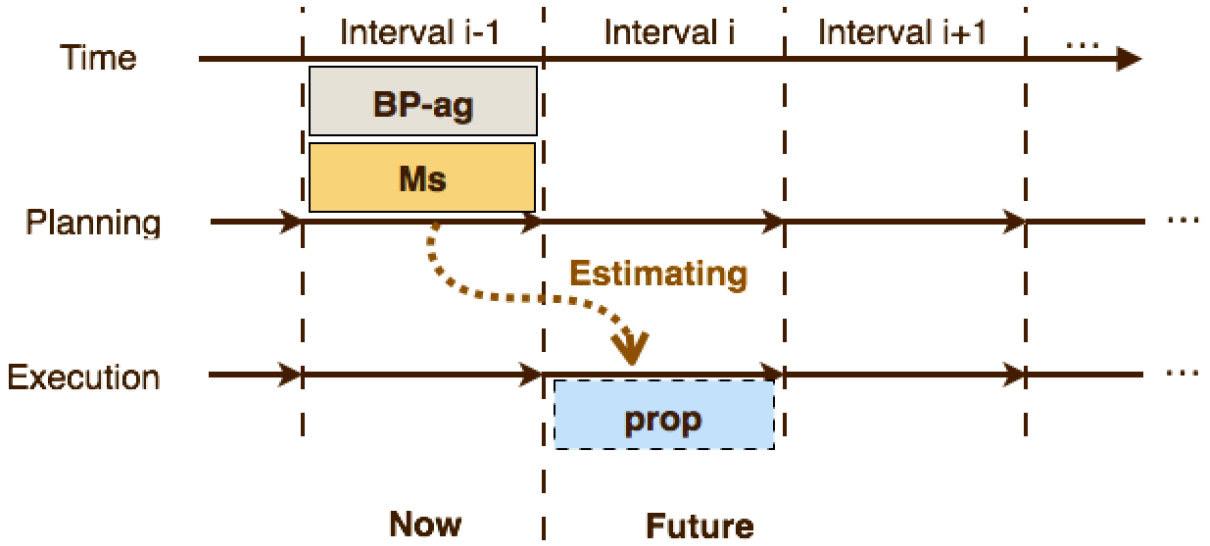


Fig. 4.7: Learning process

The activity property in this experiment is the actual power balance in the grid. The behavior parameters are all the parameters of power units such as demand of power load, capacity and cost of generator, and price of power. For an instance, at interval i-1, all the behavior parameters are collected by the meta agent from the member agents. Based on those information, the MGOCC makes plan for the power units. At interval i, the plan is executed by the grid and the actual power balance at that time is also monitored by meta agent as the activity property. Then the behavior parameters and activity property are used for the regression analysis to construct the implementation of the property calculation function. This learning process is performed for every interval of the grid operation to accumulate the knowledge to construct accurate implementation of the function. At the same time, the estimation process is performed in parallel.

While in the estimation process, only BP-ag is used as input as shown in this figure.

For example, at interval i-1, power unit parameters are monitored by meta-agent as behavior parameters as the same in the learning process. Once the behavior parameters are collected, they are used as input parameters to the activity property calculation function, which is the result of learning process. The function takes power unit parameters as input and calculates the estimated power balance in the case of that the plan based on the current power unit parameters is executed in the next interval. This estimated result is used as the measurement of the system to evaluate the danger of power grid overload in the future. Therefor, if the measurement is under zero, it means the meta agent is aware overload danger in next interval. As a result, the meta agent reorganizes the power grid to increase the power production to prevent the possible system failure.



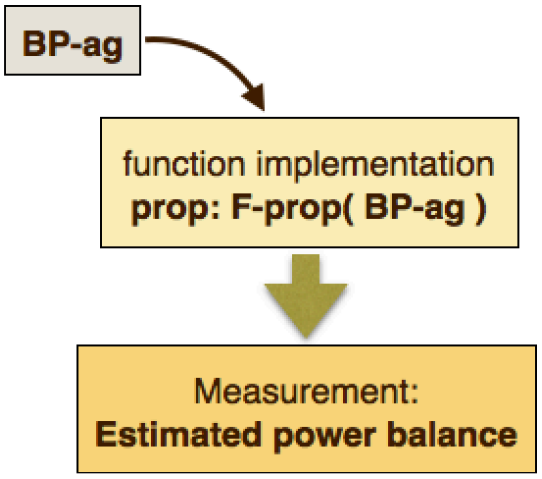


Fig. 4.8: Estimation process

4.4.3 Experiment Results

The experiment results are discussed in this section.

The power balance serial of scheduling as required is shown in Figure 4.5. It is noticed

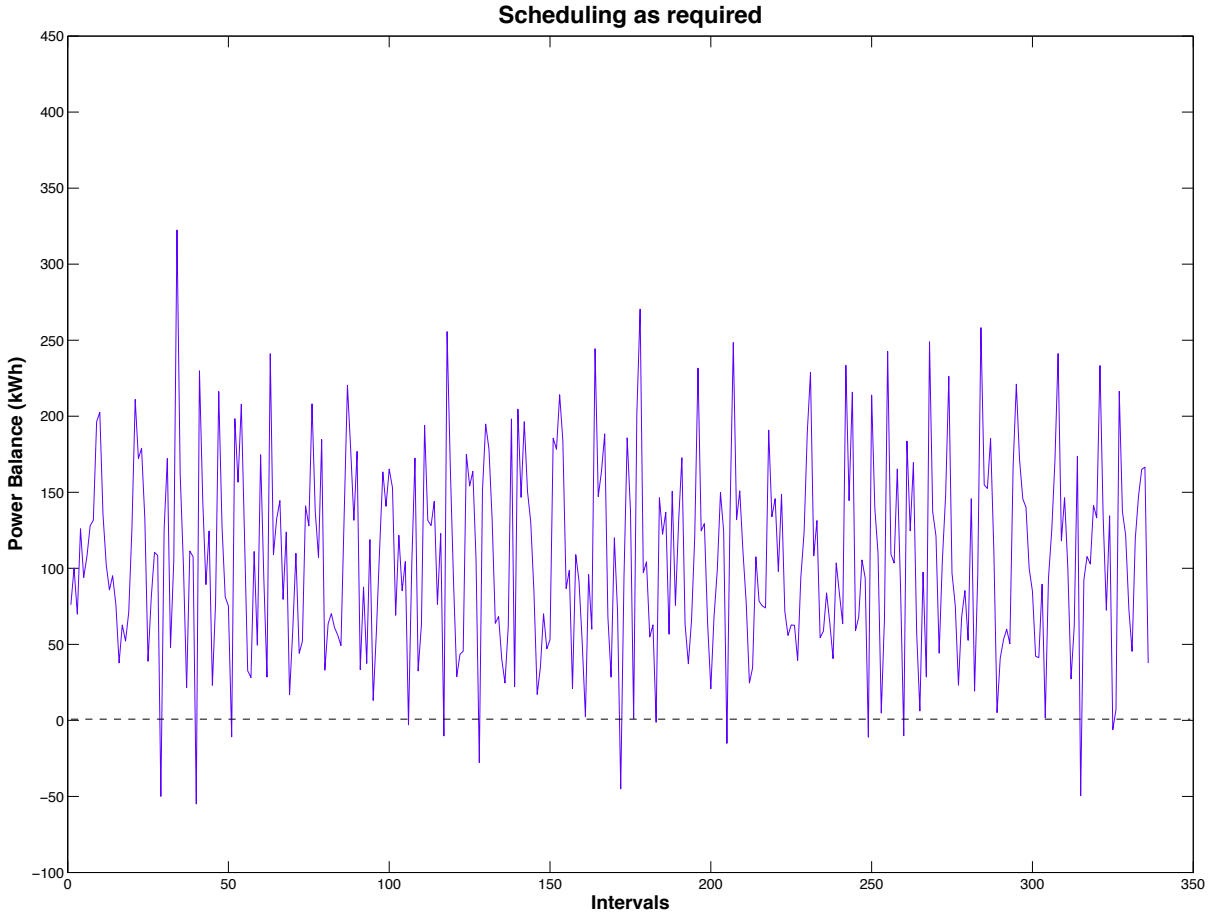


Fig. 4.9: Scheduling as required

that although the system reorganized power supply agent after overload failure to recover every times, the grid still experienced many times of power demand overload failures. It will be unacceptable for real world application that the power grid is unstable like this.

In practice, the generators are usually scheduled to produce more power than required to prevent overload failure. For this simulation experiment, the generators are scheduled to produce 40% more power than the load required.

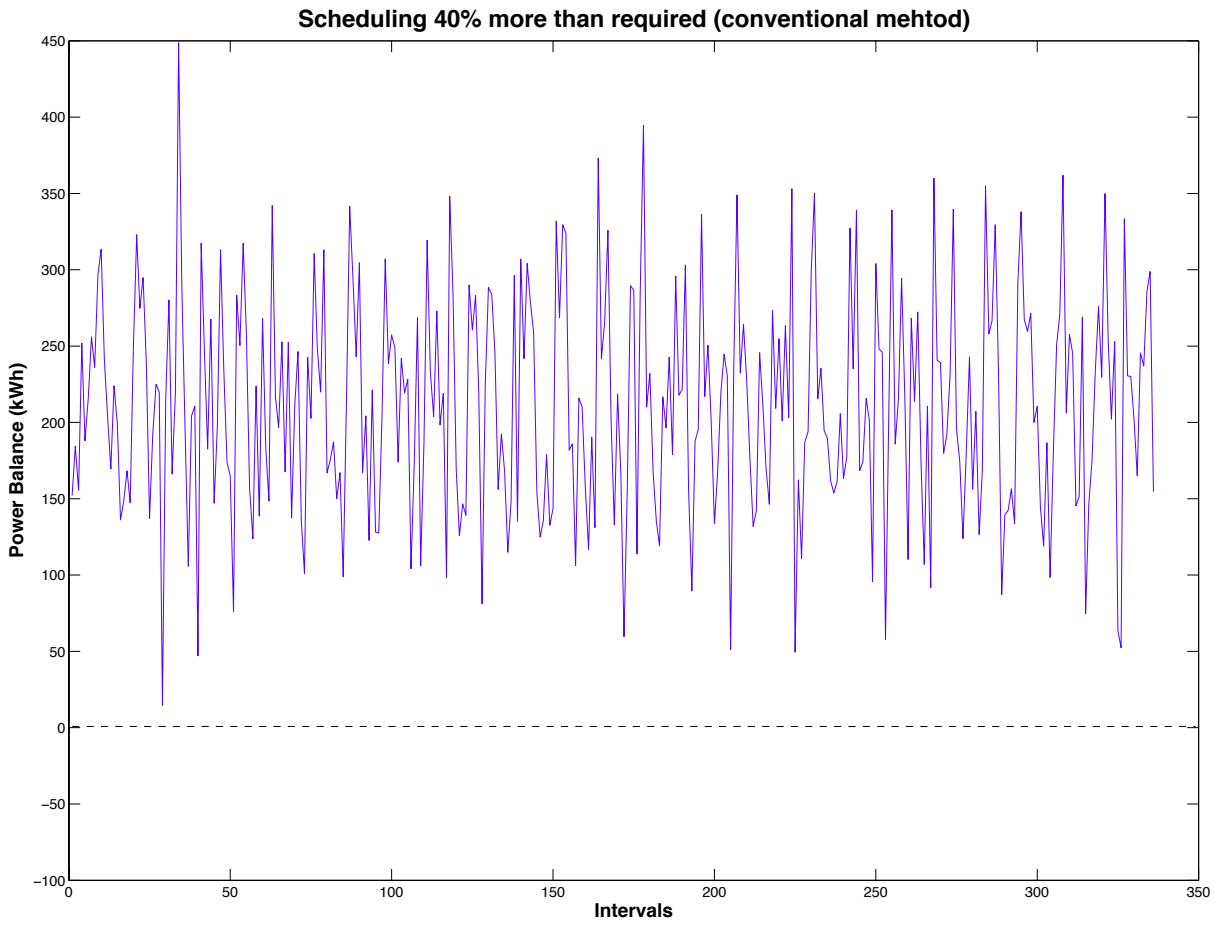


Fig. 4.10: Scheduling more all the time

As a result of preparing more power than loads actually need, there is no overload failure during the simulation. However, this method wastes a significant amount of energy which becomes more and more valuable nowadays. Especially for the large scaled gird, the waste of energy must not be ignored.

On the other hand, the following figure shows the power balance while the grid is scheduled by the EAS.

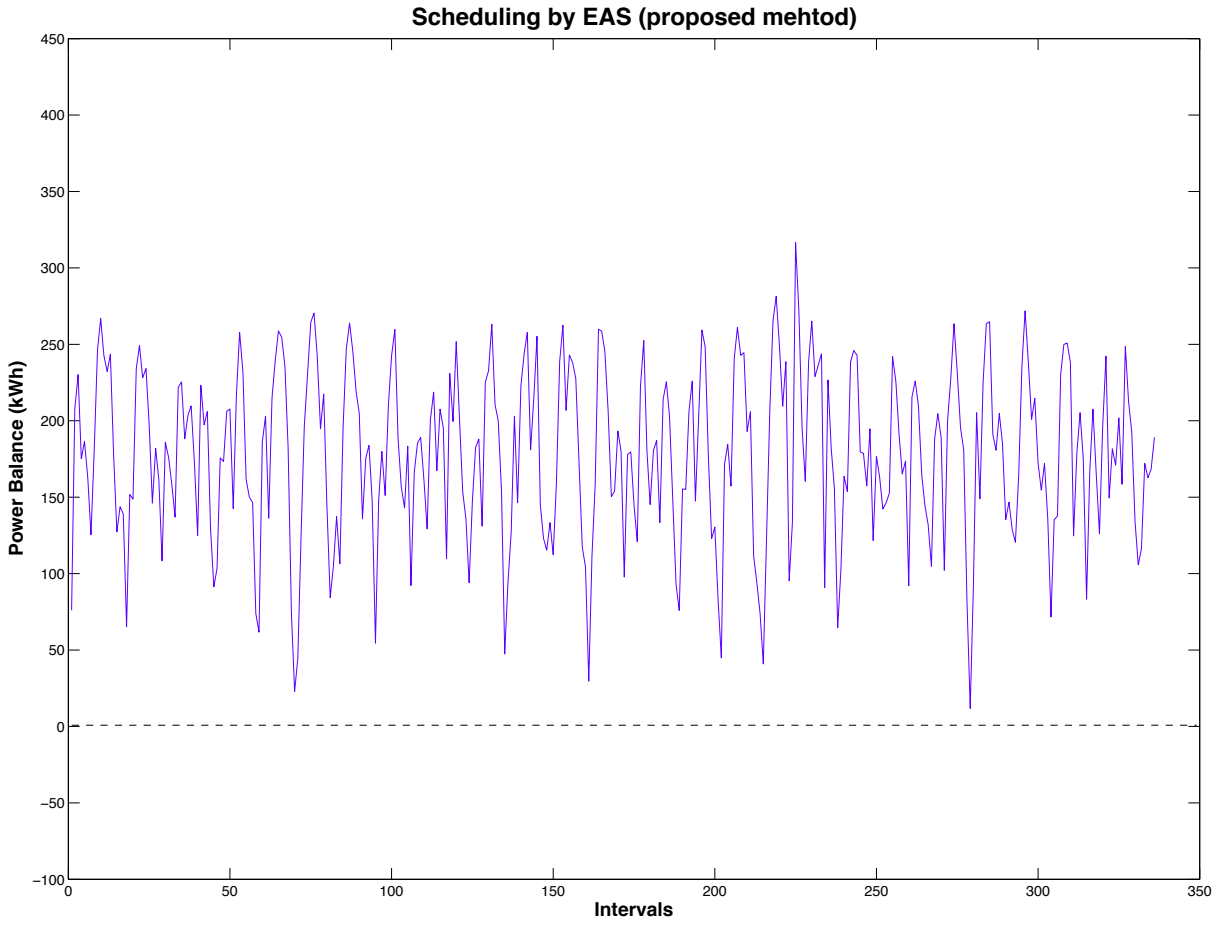


Fig. 4.11: Scheduling by EAS

It is obvious that on overload failures occurs thanks to the reorganizations performed while meta agent predicts the possible overload failure before the failures actually arise.

In the aspect of wasted power, the comparison of the amount of wasted power between proposed EAS based control and conventional manual control is shown in the following figure.

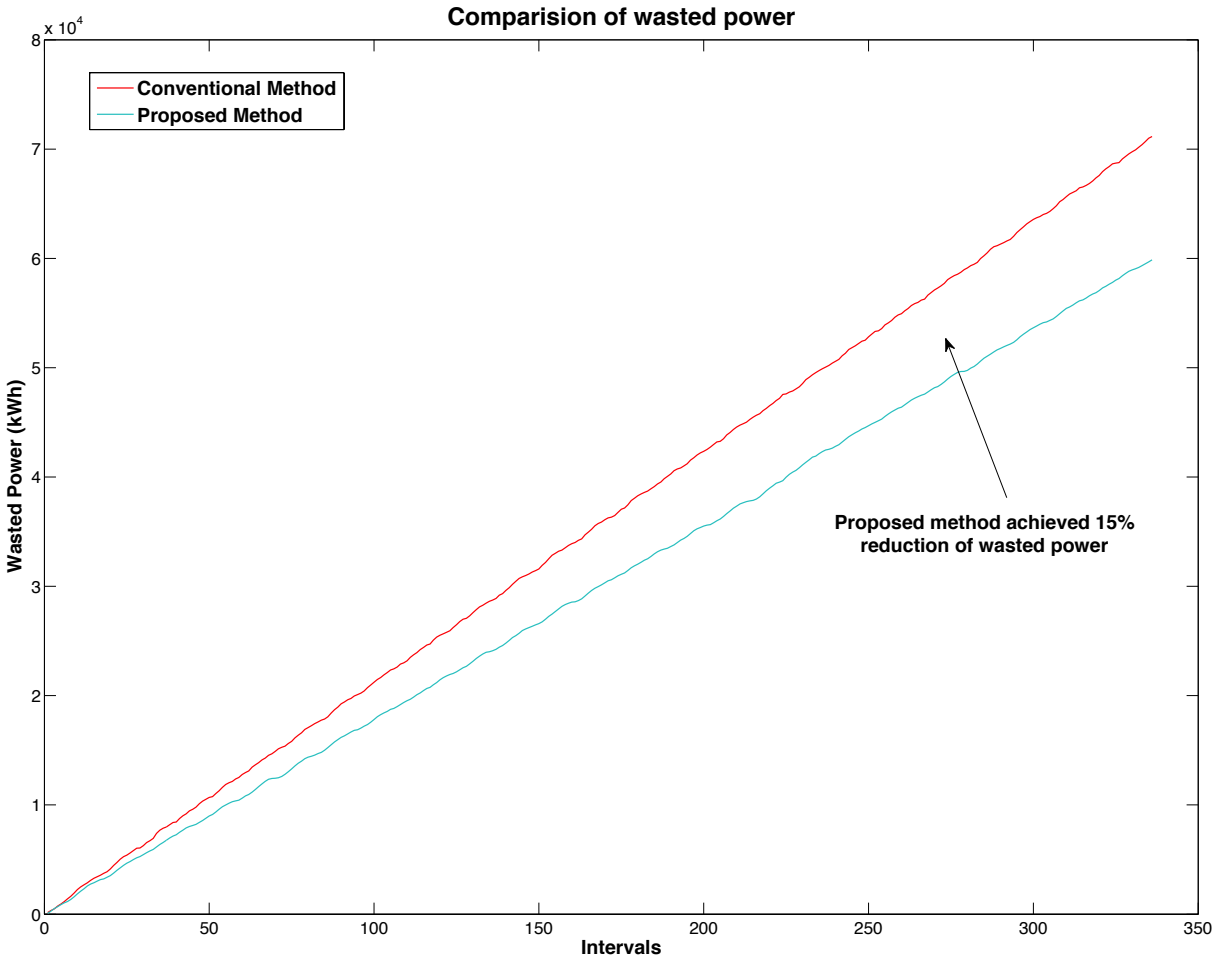


Fig. 4.12: Comparison of wasted power

As shown in the figure, proposed method is managed to reduce wasted power successfully by 15%.

4.4.4 Evaluation

To verify if proposed knowledge construction method is able to autonomously accumulate control knowledge to avoid system failure, this experiment is performed. The results of the experiment shows the experiment system using proposed method is able to re-organise before overload actually happens to prevent failure. What is important is that through the experiment there is no need to manually input knowledge about Evolutional Control explicitly. That means proposed knowledge construction method is possible to extract knowledge autonomously from the running system. Furthermore, the knowledge proposed method discovered successfully prevent system failures, which is the original goal of the proposal. Therefor the proposal is totally feasible for Evolutional Systems.

4.5 Summary

In this chapter, toward the effective realisation of Evolutional System with deterioration resistant property, the research on control knowledge construction is discussed. Challenges appears as the problems of how to decide the appropriate timing of performing Evolutional Control before actual undesired changes arise without extra design burden. The existing approach either lacks the ability to avoid deterioration of systems or adds extra burden and difficulty of manual knowledge construction. To improve that situation, we propose autonomous knowledge construction method for Evolutional Control. The proposal features indirect estimation of system global measurement and constructs control knowledge utilizing machine learning autonomously. The experiment result shows that proposed autonomous knowledge construction method is able to decide the appropriate timing of performing Evolutional Control without extra design burden to help the effective realisation of Evolutional System featuring deterioration resistant property.

Chapter 5 Conclusion

5.1 Conclusions

As the background of this research, it becomes common that information processing systems are distributed over vary network environment. For this kind of distributed systems, the activity properties of them change irregularly depending on the changes or failures of system components as also as the changes of network situations. The unstable changing of activity properties is one of the main reasons, which are responsible for negatively effecting the quality of services. In general, it is extremely difficult to consider all the possible changes the system will face to in the design process. Presently, many research focusing on the judgment of abnormal situation of systems and the recover solutions are under taken.

On the other hand, the research about cooperation distributed system call Evolutional System is introduced. By constructing the system as multi-agent systems, Evolutional System approach proposes the methodology that performs preventing processing in advance before the effects of system changes arise by monitoring the activity properties of the system, which affect system behaviours. An appropriate method for design and development of Evolutional System is important. In this paper, we provides a design method based on multi-agent system, which is described in 3 parts.

In chapter 2, we propose operation controlling architecture for agent-based Evolutional Systems. The operations and activities of Evolutional System are formalised and modelled in Evolutional Agent System(EAS) model. Based on the EAS model, we develop the EAS architecture. By the experiment of using energy consumption rate as measurement, the feasibility of

proposed model is proved. This is a brand new approach for organization design for Evolutional Systems.

In chapter 3, we propose protocol design method for agent-based Evolutional Systems. In details, we introduce protocol template mechanism, which enables the communication protocol reuse among different applications. Furthermore, based on model-driven development principle, we propose a tool supported design work flow utilising newly proposed meta model of protocol template for supporting the design and development of EAS. The comparison experiment shows proposal reduces almost half development cost against to conventional method using general purpose tools. This is useful for the effective design and development of EAS.

In chapter 4, we proposes autonomous knowledge construction method for Evolutional Control. In the other words, by utilising machine learning technique to the monitoring results of system activity property, the collection and management of control knowledge, which are performed manually in present, are able to be finished automatically. And simulation experiment proves the feasibility of our proposal. This is a significant result toward to the design and development of EAS for practice usages.

In summary, we proposes an original design method based on multi-agent system for Evolutional Systems in this paper, which is a significant contribution to agent-oriented software engineering and information sciences.

5.2 Contributions

The first contribution of this research is that a feasible agent-based design method for Evolutional Systems is provided for reliable and effective realisation

As the second contribution, using the provided design method, effective ES realisation with deterioration resistance provides solutions for the problem of preventing unpredictable system failures, which distributed open system usually have.

This research also provides a possible solution for mission critical industry applications, such

as financial systems, autonomous network management system, etc.

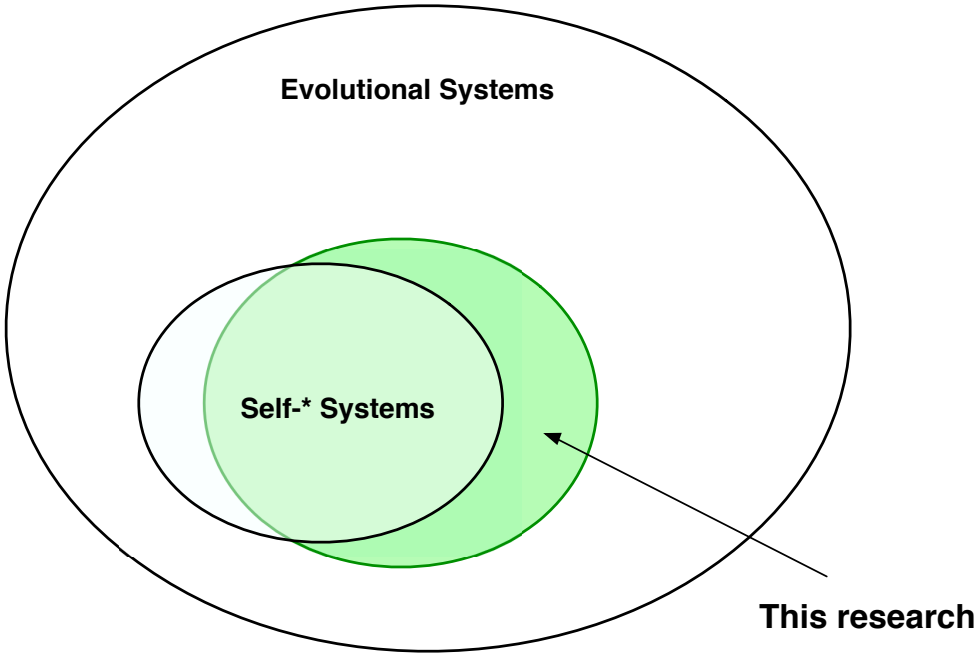


Fig. 5.1: Contribution of this research

5.3 Future Works

The essence characters of autonomous knowledge construction are still unclear, more work for discovering them through long time span experiment is needed.

The effective of autonomous knowledge construction method for Evolutional Control is still possible to be improved by using disparate machine learning method(e.g. neural network deep learning).

The work about the design and implementation of general supporting tools (e.g. common framework, development environment, etc) are necessary and welcomed.

Publications

Journals

1. Wenpeng Wei, Hideyuki Takahashi, Takahiro Uchiya, Tetsuo Kinoshita, “Cooperation Protocol Design Method for Repository-based Multi-agent Applications”, International Journal of Software Science and Computational Intelligence, Vol.5, No.2, pp1-14, 2013.

2. Wenpeng Wei, Akiko Hatakashi, Tetsuo Kinoshita, “Design and Evaluation of Energy-consumption-aware Evolutional Agent System for Portable Devices”, Journal of Information Processing. (Conditional Accepted)

3. Wenpeng Wei, Tetsuo Kinoshita, “ Autonomous Control Knowledge Construction Using Machine Learning for Evolutional Agent System ”(In Preparation)

4. Hak-Man Kim, Wenpeng Wei, Tetsuo Kinoshita, ” A New Modified CNP for Autonomous Microgrid Operation based on Multiagent System, ” Journal of Electrical Engineering & Technology, Vol.6, No.1, pp139-146, 2011.1.

Proceedings

1. Shota Kotato, Hideyuki Takahashi, Wei Wenpeng, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, ”User-Oriented Information Delivery System based on Autonomous Cooperation of Heterogeneous Contents,” Proc. of the 2nd International Workshop on Smart Technologies for Energy, Information and Communication (STEIC2013), pp.105-112, 2013. 8.

2. Wenpeng Wei, Akiko Takahashi, Tetsuo Kinoshita, ”Design of Energy-consumption-aware Evolutional Agent System for Portable Devices,” Proc. of the 12th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC2013), pp.254-259, 2013. 7.

3. Shota Kotato, Aki Asanuma, Wenpeng Wei, Hideyuki Takahashi, Tetsuo Kinoshita, ”Interactive Information Delivery System based on Active Information Resources,” Proc. of the 12th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2013), pp.247-250, 2013. 6.

4. Wenpeng Wei, Aki Asanuma, Shota Kotato, Hideyuki Takahashi, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, ”User-oriented Autonomous Contents Delivery System based on Active Information Resources,” MoMuC2012-57,AN2012-59,USN2012-68(2013-1), pp.85-86, 2013. 1.

5. Akiko Takahashi, Mitsuru Abe, Wenpeng Wei, Tetsuo Kinoshita, ”Proactive Control Method based on System Margin in Evolutional Agent System,” Proc. of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pp.64-68, 2012.12.

6. Wenpeng Wei, Aki Asanuma, Taishi Ito, Hideyuki Takahashi, Kazuto Sasai, Gen Kitagata, Tetsuo Kinoshita, ”Design of Cooperation Scheme of Active Information Resource for Heterogeneous Contents,” Proc. of the 1st International Workshop on Smart Technologies for Energy, Information and Communication (IW-STEIC2012), pp.81-87, 2012.10.

7. Akiko Takahashi, Mitsuru Abe, Noriyuki Horikawa, Wenpeng Wei, Tetsuo Kinoshita, ”Design and Evaluation of System Margin in Evolutional Multiagent System,” Proc. of the 1st International Workshop on Smart Technologies for Energy, Information and Communication (IW-STEIC2012), pp.69-79, 2012.10.

8. Wenpeng Wei, Hideyuki Takahashi, Takahiro Uchiya, Tetsuo Kinoshita, ”Repository-based Methodology of Cooperation Protocol Design for Multi-agent System,” Proc. of The 11th International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC2012), pp.283-288, 2012.8.

9. Wenpeng Wei, Hideyuki Takahashi, Takahiro Uchiya, Tetsuo Kinoshita, ”Cooperation Protocol Design Method based on Repository Mechanism for Multiagent System,” Proc. of the Joint Agent Workshop & Symposium 2011 (JAWS2011), pp.1-6, 2011.10.

Bibliography

[1] Unified Modeling Language (UML). http://www.uml.org/.

[2] Ebrahim Al-Hashel, BalaM. Balachandran, and Dharmendra Sharma. A Comparison of Three Agent-Oriented Software Development Methodologies: ROADMAP, Prometheus, and MaSE. In Bruno Apolloni, RobertJ. Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems SE - 111*, volume 4694 of *Lecture Notes in Computer Science*, pages 909–916. Springer Berlin Heidelberg, 2007.

[3] Carole Bernon, Marie-Pierre Gleizes, Sylvain Peyruqueou, and Gauthier Picard. ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents World III SE - 12*, volume 2577 of *Lecture Notes in Computer Science*, pages 156–169. Springer Berlin Heidelberg, 2003.

[4] Lawrence S. Brakmo, Deborah A. Wallach, and Marc A. Viredaz. μSleep: a technique for reducing energy consumption in handheld devices. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services - MobiSYS '04*, page 12, New York, New York, USA, June 2004. ACM Press.

[5] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

[6] Giovanni Caire, Wim Coulier, Francisco Garijo, Jorge Gómez-Sanz, Juan Pavón, Paul Kearney, and Philippe Massonet. The MESSAGE methodology. *Methodologies and Soft-*

ware Engineering for Agent Systems The Agent-Oriented Software Engineering Handbook, 11:177–194, 2004.

[7] KhanhHoa Dam and Michael Winikoff. Comparing Agent-Oriented Methodologies. In Paolo Giorgini, Brian Henderson-Sellers, and Michael Winikoff, editors, *Agent-Oriented Information Systems SE - 6*, volume 3030 of *Lecture Notes in Computer Science*, pages 78–93. Springer Berlin Heidelberg, 2004.

[8] Scott A DeLoach. The mase methodology. In *Methodologies and software engineering for agent systems*, pages 107–125. Springer, 2004.

[9] D.G. Firesmith. Use case modeling guidelines. *Proceedings of Technology of Object-Oriented Languages and Systems - TOOLS 30 (Cat. No.PR00278)*, 1999.

[10] Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277–296, March 2000.

[11] Thomas Juan, Adrian Pearce, and Leon Sterling. ROADMAP: extending the gaia methodology for complex open systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1*, pages 3 – 10, New York, New York, USA, July 2002. ACM Press.

[12] Thomas Juan and Leon Sterling. A meta-model for intelligent adaptive multi-agent systems in open environments. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS '03*, AAMAS '03, page 1024, New York, New York, USA, 2003. ACM Press.

[13] Tetsuo Kinoshita and Kenji Sugawara. ADIPS Framework for Flexible Distributed Systems. In Toru Ishida, editor, *Multiagent Platforms*, chapter Chapter 2, pages 18–32. Springer Berlin / Heidelberg, Berlin, Heidelberg, 1999.

[14] Graham Low, Ghassan Beydoun, Brian Henderson-Sellers, and Cesar Gonzalez-Perez. Towards Method Engineering for Multi-Agent Systems: A Validation of a Generic MAS Metamodel. In Aditya Ghose, Guido Governatori, and Ramakoti Sadananda, editors, *Agent Computing and Multi-Agent Systems SE - 22*, volume 5044 of *Lecture Notes in Computer Science*, pages 255–267. Springer Berlin Heidelberg, 2009.

[15] Markus Luckey, Benjamin Nagel, Christian Gerth, and Gregor Engels. Adapt Cases: Extending Use Cases for Adaptive Systems. In *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11*, page 30, New York, New York, USA, May 2011. ACM Press.

[16] Walamitien H. Oyenan and Scott A. DeLoach. Design and Evaluation of a Multiagent Autonomic Information System. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pages 182–188. IEEE, November 2007.

[17] Meikang Qiu, Zhi Chen, Laurence T. Yang, Xiao Qin, and Bin Wang. Towards Power-Efficient Smartphones by Energy-Aware Dynamic Task Scheduling. In *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, pages 1466–1472. IEEE, June 2012.

[18] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):1–42, May 2009.

[19] Candelaria Sansores and Juan Pavón. An adaptive agent model for self-organizing MAS. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, pages 1639–1642. International Foundation for Autonomous Agents and Multiagent Systems, May 2008.

[20] Munindar P Singh. Agent-based abstractions for software development. *Methodologies and Software Engineering for Agent Systems*, pages 5–18, 2004.

[21] A. Takahashi, T. Suganuma, T. Abe, Y. Iwaya, and T. Kinoshita. A behavioral characteristics model for flexible distributed system. In *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, volume 1, pages 6 pp.–280. IEEE, 2006.

[22] Akiko Takahashi, Mitsuru Abe, Noriyuki Horikawa, Wenpeng Wei, and Tetsuo Kinoshita. Design and Evaluation of System Margin in Evolutional Multiagent System. In *The 1st International Workshop on Smart Technologies for Energy, Information and Communication*, pages 69–79, 2012.

[23] Akiko Takahashi, Mitsuru Abe, Wenpeng Wei, and Tetsuo Kinoshita. Proactive Control Method Based on System Margin in Evolutional Agent System. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 64–68. IEEE, December 2012.

[24] Akiko Takahashi and Tetsuo Kinoshita. Configuration and control design model for an agent based Flexible Distributed System. *Web Intelligence and Agent Systems*, 9(2):161–178, January 2011.

[25] Akiko Takahashi and Tetsuo Kinoshita. Dynamic Control and Construction Method for Multiagent Systems Based on an Evolutional Agent System. *International Journal of Energy, Information and Communications*, 4(2):1–20, 2013.

[26] Akiko Takahashi, Takuo Suganuma, and Tetsuo Kinoshita. Dynamic Construction Scheme of Multimedia Processing Components Based on Multiagent Framework. *IPSJ Journal*, 45(2):366–376, February 2004.

[27] Gerald Tesauro, David M. Chess, William E. Walsh, Rajarshi Das, Alla Segal, Ian Whalley, Jeffrey O. Kephart, and Steve R. White. A Multi-Agent Systems Approach to Autonomic Computing. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 464–471. IEEE Computer Society, July 2004.

[28] Michael Wooldridge, NicholasR. Jennings, and David Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

Appendix A

Multi-Agent System Methodologies Comparison Result Tables

Phases	ROADMAP	MaSE	Prometheus
System specification	Detailed	Medium	Detailed
Analysis	Detailed	Detailed	Detailed
Architectural design	Abstract, high-level	Detailed	Detailed
Detailed design	Not exists (Architecture)	Not exists (Architecture)	Detailed (BDI agents)

Fig. A.1: Illustrates the scale of the details within each development phase[2]

Concept	ROADMAP	MaSE	Prometheus
Autonomy	Medium	Medium	High
Mental attitudes	Uses knowledge schema.	Agents do not have to be intelligent	Agents are intelligent agents. BDI agents.

Fig. A.2: Presents the measure of agent concept that each methodology support[2]

Criteria	ROADMAP	MaSE	Prometheus
Clear notation	Strongly agree	Strongly agree	Strongly agree
Ease of learning	Strongly agree	Strongly agree	Agree
Ease of use	Agree	Strongly agree	Agree
Adaptability	Strongly agree	Strongly agree	Agree
Traceability	Strongly agree	Agree	Strongly agree

Fig. A.3: Shows the scale of the modeling criteria within each methodology[2]

Property	ROADMAP	MaSE	Prometheus
Openness	High	Low	Medium
Environment	High	Medium	Medium
Abstraction	High	High	High
Traceability	High	High	High
Modelling	Medium	High	High
Complexity	Low	Low	Medium
Ease of use	Easy, requires some	Easy	Complicated,
Limitations	Lack of richer notations	Lack of	Highly
Language	Low	Medium	High
Reusability	High	Medium	Medium

Fig. A.4: Compares the properties of the methodologies[2]

Phases	ROADMAP	MaSE	Prometheus
System Specification			Stakeholders Scenarios diagram
Analysis	Environment, Knowledge, Goal, Role, Revised role model, Social model	Use cases, Goal hierarchy, Sequence, Concurrent task diagram	Goal overview, Role, Data coupling diagrams
Architectural Design	Agent l, Service, Acquaintance model	Agent classes, Conversations	Agent acquaintance, System Overview Agent Descriptors Protocols
Detailed Design		Agent's internal architectures, Deployment diagram	Process , Agent Overview Diagrams, and Capacity, Capability overview, Event, Data, and Plan

Fig. A.5: Illustrates the available activities in each development phase[2]

Methodology	Application
ROADMAP	Coarse-grained computational, complex, open systems
MaSE	Heterogeneous multi agent systems
Prometheus	Intelligent (BDI) agents' systems

Fig. A.6: Illustrates the type of the system domain that each methodology is suitable for[2]

Methodology	Development Tool
ROADMAP	REBEL is a tool for building Goal Models and Role Models during the analysis stage.
MaSE	AgentTool, able to do printing, verification on developing system, and generating a skeleton code in java.
Prometheus	Prometheus Design Tool (PDT), which is able to do cross checking, saving diagrams as pictures, JDE (JACK Development Environment) generates JACK code.

Fig. A.7: Summarizes the toolkits that are available for each methodology[2]

	GAIA	TROPOS	MAS-COMMONAKADS	PROMETHEUS	PASSI	ADELFE	MASE	RAP	MESSAGE	INGENIAS
Development lifecycle	**Iterative within each phase but sequential between phases**	**Iterative and incremental**	**Cyclic risk-driven process**	**Iterative across all phases**	**Iterative across and within all phases (except for coding and deployment)**	**Rational Unified Process (RUP)**	**Iterative across all phases**	**RUP**	**RUP**	**Unified software development process**
Coverage of the lifecycle	**Analysis and Design**	**Analysis and Design**	**Analysis and Design**	**Analysis and Design**	**Design and Implementation**	**Analysis, Design and Implementation**	**Analysis and Design**	**Analysis and Design**	**Analysis and Design**	**Analysis, Design and Implementation**
Development perspecti	**Top-down**	**Top-down**	**Hybrid**	**Bottom-up**	**Bottom-up**	**Top-down**	**Top-down**	**Hybrid**	**Hybrid**	**Hybrid**
Application domain	**Independent (business process management, GIS, traffic simulation)**	**Independent (e-business systems, knowledge management, health IS)**	**Independent (Flight reservation, automatic control)**	**Independent (holonic manufacturing, online bookstore)**	**Independent (distributed robotics applications, online bookstore)**	**Dependent - adaptive systems (adaptive intranet system, timetabling system)**	**Independent (distributed planning, database integration system, computer virus immune system, automatic control)**	**Dependent – distributed organizational IS (supply chain management, enterprise resource planning)**	**Independent (network management , operational support systems, knowledge management systems)**	**Independent (collaborative information filtering, personal computer management , robot battles)**
Size of MAS	**<= 100 agent classes**	**Not specified**	**Not specified, but possibly any size**	**Any size**	**Not specified**	**Not specified, but possibly any size**	**<= 10 agent classes**	**Any size**	**Not specified, but possibly any size**	**Not specified, but possibly any size**
Agent nature	**Heterogeneous**	**BDI-like agents**	**Heterogeneous**	**BDI-like agents**	**Heterogeneous**	**Adaptive**	**Not specified but possibly heterogeneous**	**Reactive agents**	**Heterogeneous**	**Agents with goals and states**
Support for verification and validation	**No**	**Yes**	**Mentioned but no explicit steps/guidelines provided**	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**Mentioned as future enhancement**	**Yes**
Ease of understanding of the process	**High**	**High**	**High**	**High**	**High**	**High**	**High**	**High**	**High**	**High**
Usability of the methodology	**Medium**	**Medium**	**Medium**	**High**	**High**	**High**	**High**	**Medium**	**Medium**	**High**
Refinability	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Approach towards MAS developm ent	**a. OO b. RO (OrO)**	**a. i* modelling framework b. NRO**	**a. KE b. NRO**	**a. OO b. NRO**	**a. OO b. RO**	**a. OO b. NRO**	**a. OO b. RO (GO)**	**a. OO b. RO**	**a. OO b. RO (GO and BO)**	**a. OO b. RO**

Fig. A.8: Comparison of Concept[14]

Concepts & Properties	MaSE	Prometheus	Tropos
Autonomy	H/M/DK	H/NA/H	H/M/M
Mental attitudes	L/M/H	H/M/H	H
Proactive	H/M/H	H/M/DK	H
Reactive	M	H/M/DK	H/L/DK
Concurrency	H/M/H	M/L/DK	H/M/H
Teamwork	H/M/H	N/L/NA	H/H/M
Protocols	H	M/H/M	NA/M/M
Situated	M/L/H	H	H
Clear concepts	SA/A/A	A/A/DA	SA/A/N
Concepts overloaded	A/N/SA	N	SDA/N/DA
Agent-oriented	SA/A/A	SA	SA/A/SA

Fig. A.9: Comparing a methodology ’s properties, attributes, process and pragmatics. Notation: L for Low, M for medium, H for High, DK for Don ’t Know, SDA for Strongly Disagree, DA for Disagree, NA for Not Applicable, N for Neutral, A for Agree, SA for Strongly Agree, for no response. The first two entries in each column are the developers of the methodology, the third is the student. A single entry in the column indicates that all three answers agreed.[7]

Concepts & Properties	MaSE	Prometheus	Tropos
Modelling & Notation			
Static+Dynamic	SA/A/A	SA/A/A	N/A/A
Syntax defined	A/A/SA	SA/A/A	SA/N/A
Semantics defined	A/SA/SA	A	SA/A/A
Clear notation	A	SA/A/A	SA/A/N
Easy to use	SA/A/A	A/N/A	SA/A/N
Easy to learn	N/N/A	SA/NA/SA	SA/N/A
Different views	N/N/A	A/A/SA	SA/A/N
Language adequate & expressive	SA/N/N	A	SA/A/N
Traceability	A/SA/SA	A	A/N/A
Consistency check	SA/A/SA	SA/A/A	_/A/DA
Refinement	SA/A/A	SA	SA/A/DA
Modularity	SA/A/A	SA/SA/A	SA/A/N
Reuse	N/SA/A	N/A/N	_/A/DA
Hierarchical modelling	N/A/A	SA/A/A	SA/A/DA
Process			
Requirements	SPEH	SPEH	SPE
Architectural design	SPEH	SPEH	SPE
Detailed design	SPEH	SPEH	SPE
Implementation	SEH/SPE/S	SPEH/S/n	SE/SPE/SPEH
Testing & Debugging	SPE/n/n	SPEH/S/n	n
Deployment	SE/SPE/SPEH	n	n
Maintenance	n/SPE/n	n	n
Pragmatics			
Quality	N/DA/A	A/N/N	DA/A/_
Cost estimation	_/DA/SA	DA/DA/N	DA/N/_
Management decision	_/DA/SA	SDA/N/_	SA/A/_
apps	21+	6-20	1-5
Real apps	no	no	no
Used by non-creators	yes	yes	yes/no/no
Domain specific	no	no	yes/no/no
Scalable	_/N/N	N/A/N	N/N/_
Distributed	_/SA/SA	SA/A/N	N/A/_

Fig. A.10: Comparing methodology ’s properties, attributes, process and pragmatics. Notation: L for Low, M for medium, H for High, DK for Don ’t Know, SDA for Strongly Disagree, DA for Disagree, NA for Not Applicable, N for Neutral, A for Agree, SA for Strongly Agree, for no response. S for Stage mentioned, P for Process given, E for Examples given, H for Heuristics given, n for none. The first two entries in each column are the developers of the methodology, the third is the student. A single entry in the column indicates that all three answers agreed.[7]

Appendix B

Example of Generated DASH Rule Set Code

Content of the generated file Contract_Net_Protocol_Participant.rset:

```
(rule-set Contract_Net_Protocol_Participant


  (property
  )


  (initial_facts
    (Contract_Net_Protocol_Participant :thread 0 :state init :max_thread 0)
  )


  (rule Contract_Net_Protocol_Participant_receive_CNP_cfp_on_Init_to_ComposingProposal
    (Msg :performative CNP_cfp :content ?task :protocol_from ?pfrom) = ?msg
    (Contract_Net_Protocol_Participant :state init :max_thread ?max_thread) = ?init
    -->
    (bind ?new_thread (+ ?max_thread 1))
    (modify ?init:max_thread ?new_thread)
    (bind ?new_state ?init)
    (Modify ?new_state:thread ?new_thread)
    (bind ?state ?new_state)
    (make ?state)
    (Modify ?state:state ComposingProposal)
    (Modify ?state:partner ?msg:from)
```

```
  (Modify ?state:partner_thread ?pfrom)
  (make (CNP_compose_proposal :task ?task :state ?state))
  (remove ?msg)
)

(rule Contract_Net_Protocol_Participant_receive_CNP_cfp_on_Init_to_ComposingProposal_from_user
  ( Msg :performative CNP_cfp :content ?task ) = ?msg
  (Contract_Net_Protocol_Participant :state init :max_thread ?max_thread) = ?init
  -->
  (bind ?new_thread (+ ?max_thread 1))
  (modify ?init:max_thread ?new_thread)
  (bind ?new_state ?init)
  (Modify ?new_state:thread ?new_thread)
  (bind ?state ?new_state)
  (make ?state)
  (Modify ?state:state ComposingProposal)
  (Modify ?state:partner ?msg:from)
  (make (CNP_compose_proposal :task ?task :state ?state))
  (remove ?msg)
)

(rule Contract_Net_Protocol_Participant_receive_CNP_send_proposal_to_WaitingForAccept
  (CNP_send_proposal :proposal ?proposal :timeout_after ?timeout_after :return_to ?interface) = ?command
  (bind ?rs ?interface:state)
  (Contract_Net_Protocol_Participant :state == ?rs:state :thread == ?rs:thread ) = ?state
  -->
  (send :performative CNP_proposal :content ?proposal :to ?state:partner :protocol_to ?state:partner_thread
   :protocol_from (Contract_Net_Protocol_Participant :thread ?state:thread))
  (Modify ?state:state WaitingForAccept)
  (alarm :after ?timeout_after :content (Contract_Net_Protocol_Participant :state WaitingForAccept
   :thread ?state:thread))
```

```
  (remove ?command)
)

(rule Contract_Net_Protocol_Participant_receive_CNP_refuse_to_Terminate
  (CNP_refuse :return_to ?interface) = ?command
  (bind ?rs ?interface:state)
  (Contract_Net_Protocol_Participant :state == ?rs:state :thread == ?rs:thread ) = ?state
  -->
  (Modify ?state:state Terminate)
  (remove ?command)
)

(rule Contract_Net_Protocol_Participant_receive___Alarm_on_WaitingForAccept_to_Terminate
  (Msg :performative __Alarm :content (Contract_Net_Protocol_Participant :thread ?athread
   :state WaitingForAccept)) = ?msg
  (Contract_Net_Protocol_Participant :state WaitingForAccept :thread == ?athread) = ?state
  -->
  (Modify ?state:state Terminate)
  (remove ?msg)
)

(rule Contract_Net_Protocol_Participant_receive_CNP_accept_on_WaitingForAccept_to_PerforimgTask
  (Msg :performative CNP_accept :content ?task :protocol_to (Contract_Net_Protocol_Participant
   :thread ?tthread) :protocol_from ?pfrom) = ?msg
  (Contract_Net_Protocol_Participant :state WaitingForAccept :thread == ?tthread) = ?state
  -->
  (Modify ?state:state PerforimgTask)
  (Modify ?state:partner ?msg:from)
  (Modify ?state:partner_thread ?pfrom)
  (make (CNP_perform_task :task ?task :state ?state))
  (remove ?msg)
```

```
  )

  (rule Contract_Net_Protocol_Participant_receive_CNP_task_succed_to_Terminate
    (CNP_task_succed :result ?result :return_to ?interface) = ?command
    (bind ?rs ?interface:state)
    (Contract_Net_Protocol_Participant :state == ?rs:state :thread == ?rs:thread ) = ?state
    -->
    (send :performative CNP_inform :content ?result :to ?state:partner :protocol_to ?state:partner_thread
     :protocol_from (Contract_Net_Protocol_Participant :thread ?state:thread))
    (Modify ?state:state Terminate)
    (remove ?command)
  )

  (rule Contract_Net_Protocol_Participant_receive_CNP_task_failed_to_Terminate
    (CNP_task_failed :return_to ?interface) = ?command
    (bind ?rs ?interface:state)
    (Contract_Net_Protocol_Participant :state == ?rs:state :thread == ?rs:thread ) = ?state
    -->
    (Modify ?state:state Terminate)
    (remove ?command)
  )

  (rule Contract_Net_Protocol_Participant_kill_terminated_thread
    (Contract_Net_Protocol_Participant :state Terminate) = ?state
    -->
    (remove ?state)
  )
)
```