A Middleware Framework for Leveraging Local and Global Adaptation in IT Ecosystems

Soojin Park
Graduate School of MOT
Sogang University
Seoul, South Korea
psjdream@sogang.ac.kr

Young B. Park
Dept. of Computer Science & Engineering
Dankook University
Seoul, South Korea
ybpark@dankook.ac.kr

Abstract — **Impressive advancements in recent smart devices suggests that the direction of software engineering's future is in development of System of Systems (SoS). Among many concepts that emerged from SoS, we focus on IT Ecosystem – a type of SoS that evolves itself in response to unplanned environment changes. Maintaining autonomy of its participant systems while preserving controllability over entire ecosystem involves various challenges that we need to solve. In this paper, we propose a middleware framework for supporting global adaptation of IT Ecosystem which guides how to determine the optimal adaptation strategy for configuring available systems to satisfy local constraints while achieving global goals. To support the selection of optimal set of participant systems, we have applied genetic algorithm. The effectiveness of our approach is evaluated through analyzing the results of a simulated unmanned forest management IT Ecosystem running the proposed framework while undergoing various environmental changes.**

Keywords-self-adaptive systems; dynamic reconfiguration; IT ecosystems.

I. Introduction

We live among numerous and constant interactions with smart devices running software. Recent technologies such as cloud computing and Internet of Things (IoT) herald a paradigm shift in the operation of software systems, where its focus is transiting from operation of a single system to operation of System of Systems (SoS). A number of recent researches built on the idea of SoS introduced the new concept of IT Ecosystem [2][3]. An IT Ecosystem is a complex system compound composed of interactive and autonomous individual systems, adaptive as a whole based on local adaptivity [1]. Individual component systems within an IT Ecosystem must constantly monitor their environmental contexts in their working territories. If an identified environment change demands reactive change to the system configuration in a participant, that participant dynamically changes its configuration using predefined strategy or knowledge accumulated from previous learnings. The local adaptation loop can be identified as a MAPE-K [3] loop and can be realized through application of adaptation frameworks such as Rainbow [5], MUSIC [6], or DiVA [7].

To create a sustainable IT Ecosystem, we need more than a local adaptation mechanism: we need a means for global adaptation as to enable IT Ecosystem-wide dynamic reconfiguration in reaction to environmental changes. Unfortunately, existing adaptation frameworks mainly offer benefits limited to local adaption of a single system, restrictive in their applicability to ensuring sustainability across entire IT Ecosystem. Therefore, in our research, we propose a new adaptation middleware framework designed to support both local and global adaptation mechanisms.

While the proposed framework is to be included in all participant systems, not all components are always run. Among the constituent components, those which implement local adaptation mechanism in response to changes in individual environments are always run. On the other hand, components that execute global adaptation mechanism via deploying new participant systems or dynamically reconfiguring existing systems in response to drastic environmental changes or significant performance drop across the entire IT Ecosystem are only run on participant system with <<*Team Leader*>> role. The role of *Team Leader* is assigned dynamically, based on the environmental situations of participant systems. The global adaptation mechanism applies a genetic algorithm to make decisions regarding where to place the most appropriate participant system within a working environment, because genetic algorithms can solve computational overhead problems when IT Ecosystems grow in scale.

We have implemented our proposed adaptation framework in a case study of IT Ecosystem for unmanned forest management system, which is among the target domains of our on-going research project. The case study will help understand the benefits, along with the drawbacks, of the proposed adaptation framework. The rest of the paper is as follows: Section 2 introduces the proposed middleware framework for IT Ecosystem adaptation. Section 3 illustrates our proposed mechanisms in use for local and global adaptations within the IT Ecosystem for unmanned forest management. Section 4 discusses the effectiveness of the global adaptation mechanism of the proposed framework through analysis of experiment results. Section 5 reviews related works addressing self-adaptation problems. Finally, in Section 6, we present the conclusions for this study, along with plans for our on-going work.

(DOI reference number: 10.18293/SEKE2015-131)

II. Adaptable Middleware Framework for IT Ecosystem

In this section, we provide an overview of the proposed adaptation middleware framework for IT Ecosystem. The architecture of the framework is composed of five packages: *Felix, MAPE Core Bundle, Local Adaptation, ITE Global Adaptation,* and *ITE Bridge* (as shown in Fig. 1). *Felix* package acts as the bridge between Android platform and OSGi [4] which provides dynamic life cycle management services for components. As depicted in Fig. 1, the proposed framework targets Android platform because of the platform's support for mobility and for its flexibility in its applicability to various application domains where it can control a wide range of passive devices in individual domain's IT Ecosystems.

Felix package includes two major components: *Adaptation Bundle Activator* which invokes bundle service components required to run in higher level packages according to system roles assigned at the participant system's initiation time, and *Configuration Manager* which manages configuration changes when the need for internal component reconfiguration arises due to external environment change. *Effectors* implemented on Android follows instructions given by *Configuration Manager* to carry out the actual configuration change.

MAPE Core Bundle package includes MAPE-K cycle managing components which enables applications to perceive its environments and determine the next system action to take. Components included in either *ITE Global Adaptation* package and *Local Adaptation* package are architecturally above *MAPE Core Bundle* package and provided as OSGi bundles as to leverage basic component lifecycle management services. All components included in *Felix* package and *MAPE Core Bundle* package are domain independent components. *Local adaptation* package, on the other hand, includes different components depending on what missions individual systems must carry out in order to achieve the global goal of the ITE Ecosystem they participate in. While basic skeleton components for maintaining MAPE cycle operation threads are within *MAPE Core Bundle* package, the domain-specific logic determining what to monitor in order to analyze situational changes and what action to execute are implemented by components included in *Local Adaptation* package. In short, actual adaptation is executed by binding *Local Adaptation* package components to *MAPE Core Bundle* package components.

Information regarding the binding between *MAPE Core Bundle* components and *Local Adaptation* components are stored in *Bundle Registry* within *Felix* package. As participant systems are activated, *Adaptation Bundle Activator* is invoked to look up for information in *Bundle Registry* to check which executable bundle should be bound to *MAPE Core Bundle*, and thereafter activate its corresponding bundles.

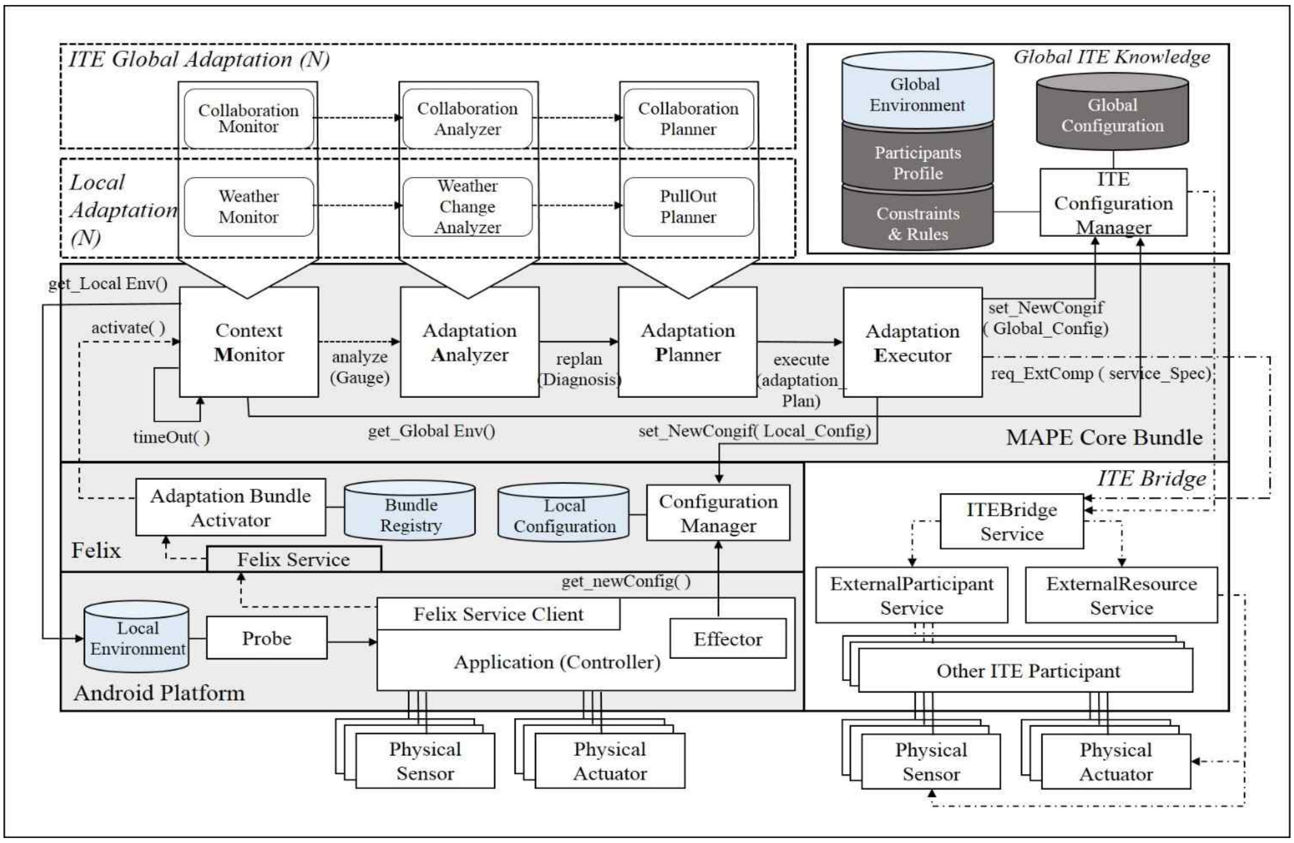


Figure 1. Overview of the proposed adaptable middleware framework for IT Ecosystem

While *ITE Global Adaptation* package is included in all participant systems, it is only run on the participant system assigned with <<*Team Leader*>> role, tasked to monitor the collaboration performance of the entire IT Ecosystem. Where *Local Adaptation* package components provide functional services required to achieve domain goals, *ITE Global Adaptation* package components measure the collaboration efficiency among participant systems executing individual local adaptations, triggering reconfiguration of participant systems when the efficiency is below target threshold.

The core capability of maintaining overall-balanced and sustained service across IT Ecosystem's domain is provided by the *ITE Global Adaptation* mechanism. Details of global adaptation mechanism are introduced alongside a specific case example in the next section. Components in *Local Adaptation* package and *ITE Global Adaptation* package are designed as plug-ins for the four components defined in *MAPE Core Bundle* package. The components within the two packages in Fig.1 illustrate components for unmanned forest management IT Ecosystem, which will be introduced in Section 3. However, the components are replaceable with other components as the target domain changes.

Global ITE Knowledge package stores knowledge to be shared among participant systems and includes *ITE Configuration Manager* which provides the APIs for accessing the knowledge. Knowledge stored at *Global ITE Knowledge* includes environment model which reflects the environment in which IT Ecosystem operates, profiles of participant systems, and constraints or rules that must be considered when mapping participant systems in their working regions. In addition, the model for currently running global configuration is also managed by this package. Such global knowledge can be stored on a separate server or on a cloud storage.

Finally, *ITE Bridge* package handles requests for remote systems when required service components in demand are not within the local system and provides protocols for activating passive devices which are not directly controlled by local systems.

III. A Case of Adaptation Mechamism Application: Unmanned Forest Management IT Ecosystem

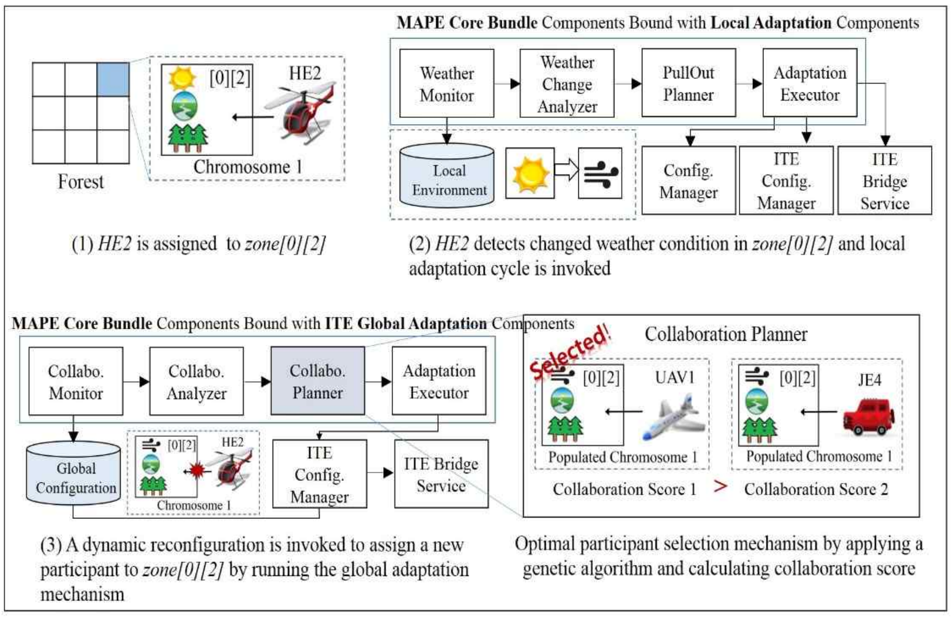
To demonstrate the proposed framework, in this Section we describe a simulated IT Ecosystem for unmanned forest management (hereafter UFM IT Ecosystem). UFM IT Ecosystem is a management system equipped with twelve unmanned aircrafts, helicopters, and ground vehicles for the purpose of managing nine forest zones each 100 km^2 in size. Unmanned vehicles assigned to each forest zone utilize sensors and actuators to achieve their goals. In our simulated case, we highlight the process of dynamic reconfiguration within UFM IT Ecosystem in achieving its *Monitor Drought* goal. Depending on the situation, twelve unmanned vehicles are assigned to appropriate roles as to achieve the goal. An unmanned vehicle with <<*Chief Gardner*>> role activates UFM IT Ecosystem's global adaptation cycle to maintain the Ecosystem's sustainability. On the other hand, nine <<*Surveillant*>> vehicles adjust their driving routes in response to the layout of obstacles in their assigned zones to achieve their goal *Monitor Drought*. In this context, Fig. 2 illustrates an instance of dynamic sequence of adaptation mechanisms in operation where a weather state change triggers a constraint rule violation in a <<*Surveillant*>> role vehicle, causing it to withdraw from its positioned forest zone, leading to local adaptation mechanisms within the vehicle along with global adaptation executed by a <<*Chief Gardner*>> as to select the optimal candidate vehicle for providing sustainable service in the zone.

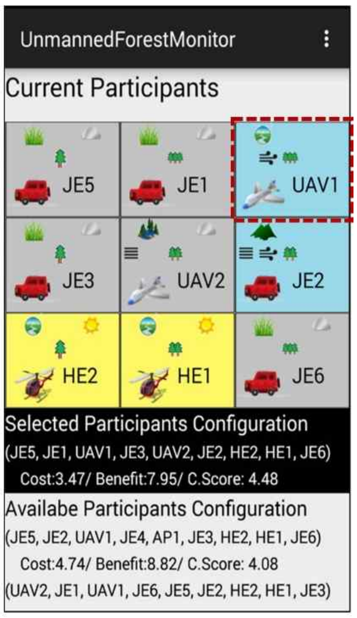
A. Local Adaptation Mechanism for Dynamic Reconfiguration of Individual Participant Systems

Initially, the forest zone[0][2] has fair weather, has a lake in the zone, and has high forest density. An unmanned helicopter *HE2* has been selected as the appropriate <<*Surveillant*>> for the environment in this zone and is performing *Monitor Draught* goal. We define the paired information of a participant and a zone in the form of (HE2, zone[0][2]) as a chromosome. Such definition becomes useful when genetic algorithm is later applied to select the optimal configuration as part of the global adaptation mechanism.

While HE2 is carrying out its goal, suddenly a turbulent gale of 25 m/s blows in zone[0][2]. The change in weather is detected by the sensors in the zone and the sensor installed on HE2, and is updated to the local environment storage. As HE2 is assigned a <<*Surveillant*>> role which is not <<*Team Leader*>>, its *MAPE Core Bundle* components are bound with *Local Adaptation* components.

As in Fig. 2 (2), local adaptation is carried out in the following order: *WeatherMonitor* periodically reads sensor data from *LocalEnvironment* storage, calculates gauge values to reflect the current environment zone[0][2], and sends the data to *WeatherAnalyzer* to diagnose if current environment in zone[0][2] violates HE2 assignment. The diagnose results is passed to *PullOutPlanner* as parameters. *PullOutPlanner* creates a component reconfiguration plan for HE2 to land safely in a safe region in zone[0][2] and sends the plan to *AdaptationExecutor*. If any component specified in the reconfiguration plan does not exist within the particular system, *AdaptationExecutor* requests it from *ITEBridgeService* by passing the required service features as parameters. *ITEBridgeService* is an OSGi bundle which provides access to external resources. The REST[8] style services provided by *ITEBridgeService* enables the proposed framework to share components, services or other resources among all participants. When all components necessary to land HE2 have been secured, the results of component reconfiguration is delivered to *ConfigurationManager* within HE2 as to update its *LocalConfiguration* storage. *Effector* in HE2's controller then reads the updated new configuration and implements the actual component reconfiguration. Lastly, the changed environmental information in zone[0][2] and the service-incapable status of HE2 are updated to *ITE Global Knowledge* through *ITEConfigurationManager*.


MAPE Core Bundle Components Bound with Local Adaptation Components
Forest
[0][2]
HE2
Chromosome 1
Weather Monitor
Weather Change Analyzer
PullOut Planner
Adaptation Executor
Local Environment
Config. Manager
ITE Config. Manager
ITE Bridge Service
(1) HE2 is assigned to zone[0][2]
(2) HE2 detects changed weather condition in zone[0][2] and local adaptation cycle is invoked
MAPE Core Bundle Components Bound with ITE Global Adaptation Components
Collabo. Monitor
Collabo. Analyzer
Collabo. Planner
Adaptation Executor
Global Configuration
[0][2]
HE2
Chromosome 1
ITE Config. Manager
ITE Bridge Service
Collaboration Planner
Selected!
[0][2]
UAV1
Populated Chromosome 1
[0][2]
JE4
Populated Chromosome 1
Collaboration Score 1 > Collaboration Score 2
(3) A dynamic reconfiguration is invoked to assign a new participant to zone[0][2] by running the global adaptation mechanism
Optimal participant selection mechanism by applying a genetic algorithm and calculating collaboration score



Selected Participants Configuration
(JE5, JE1, UAV1, JE3, UAV2, JE2, HE2, HE1, JE6)
Cost:3.47/ Benefit:7.95/ C.Score: 4.48

Availabe Participants Configuration
(JE5, JE2, UAV1, JE4, AP1, JE3, HE2, HE1, JE6)
Cost:4.74/ Benefit:8.82/ C.Score: 4.08
(UAV2, JE1, UAV1, JE6, JE5, JE2, HE2, HE1, JE3)

Figure 2. Local and global mechanism by applying the adaptable middleware framework and GUI form of simulated Unmanned Forest Monitor

B. Global Adaptation Mechanism for Dynamic Reonfiguration of the Entire IT Ecosystem

In the given case, AP2(unmanned airplane) was assigned the role of <<*Chief Gardner*>>. As HE2 implements its local adaptation to land during turbulence, AP2 starts to find a new optimal configuration against the changed situation as depicted in Fig.2 (3). The following process is for the global adaptation for finding a new optimal configuration: *CollaborationMonitor* periodically reads the global configuration and calculates the current configuration's global collaboration score. Global collaboration score is obtained as the sum of all collaboration scores of gene chromosomes (participant and environment zone pair) comprising a configuration. The global collaboration score represents how effective the particular assignment of the selected participant to the zone was. Details of calculating each collaboration score are not presented in this paper. The result of global collaboration score calculation is passed on to *CollaborationAnalyzer* as a gauge value. If the global collaboration score is below a predefined threshold, *Collaboration Analyzer* identifies gene chromosomes that violate any constraints and passes them to *Collaboration Planner* as its diagnosis result. Fig. 2 illustrates a case where the gene chromosome (HE2, zone[0][2]) is passed on to *CollaborationPlanner*.

First, *CollaborationPlanner* takes invalid gene chromosomes passed on as a diagnosis result and generates second generation population by mutating the unmanned vehicle information with other possible candidates applicable to the particular zone. Then collaboration scores for the newly generated gene chromosomes are individually calculated, and the chromosome with the highest score is selected and included to the next configuration. Fig. 2 (3) shows a case where the second generation population (UAV1, zone[0][2]) and (JE2, zone[0][2]) have been generated to replace the invalid gene chromosome (HE2, zone[0][2]). After comparing their collaboration scores, in the next configuration (UAV1, zone[0][2]) will replace (HE2, zone[0][2]) because it has a higher collaboration score. If multiple invalid gene chromosomes have been detected, the above process is repeatedly applied to each invalid gene chromosome as to obtain the optimal configuration with the highest global collaboration score.

Reconfiguring or moving participant systems using obtained optimal configuration requires a plan. In the current example, the change of configuration from chromosome (HE2, zone[0][2]) to chromosome(UAV1, zone[0][2]) implies that HE2 assigned at zone[0][2] must withdraw and UAV1 must move to the zone[0][2]. AP2, now assuming <<*Team Leader*>> role, issues orders to other participants using *ITEBridgeService* as to move them sequentially according to the adaptation plan. Then, *ITEBridgeService* activates external *ParticipantService* to implement the operation ordered by *AdaptationExecutor*. Lastly, the new configuration obtained as the result of dynamic reconfiguration is updated to *ITE Global Knowledge*.

The right-most part of Fig. 2 shows the captured GUI form representing the status of nine forest zones in the simulated IT Ecosystem for unmanned forest management, along with the configuration of unmanned vehicles stationed. As the result of the aforementioned mechanisms in operation, we can visually confirm that UAV1 is newly stationed in the forest zone[0][2] which is highlighted by a dashed box.

IV. Evaluation

Here we look at the performance evaluation results. The main objective of the proposed framework is to provide efficient global adaptation mechanism to guarantee sustainability in entire IT Ecosystem without requiring human intervention. To evaluate the performance, we have created a UFM IT Ecosystem that simulates continuous weather change (wind velocity, weather type, etc.) to trigger series of global system reconfigurations. Weather changes are divided into two types: slow and rapid. Another element of change introduced is fuel consumption: an internal status of participant systems simulated in correspondence to the distance covered by the participant. Fuel consumption represents fuel efficiency determined for each vehicle.

Graphs in Fig. 3 trace the trends in cost, benefit, and global collaboration scores (*c.score* in the graphs) of optimal configurations selected at every monitoring interval. Cost and benefit factors of each IT ecosystem naturally depend on its corresponding domain. In this case, the required amount of money for operating each unmanned vehicle is calculated as a cost value, and the coverage of drought monitoring work by an individual unmanned vehicle per unit time is calculated as a benefit value. *c.score* is derived from the cost and benefit values and represents the degree of configuration efficiency of the 9 participants in the forest zone; a higher *c.score* indicates a more efficient configuration. Graphs (a) and (c) at the left of Fig. 3 depict the changes in cost, benefit, and *c.score* values in such cases where the proposed dynamic reconfiguration framework is not provided to participant systems that become incapable of continuing *Monitor Draught* goal due to weather changes or fuel shortages. In case of graph (a) where weather changes were mild, *c.score* decrease is found from the 6th monitoring interval. This decrease indicates an event where one or more participants in the forest zone, among 9 total, have become unavailable. As time passes, the number of disabled participants increases dramatically around the 8th monitoring interval, and by the 9th interval all participants have become disabled. Since all participants are disabled at the 14th interval, further monitoring renders no additional information. Therefore, in graph (a), and in all other graphs in Fig. 3, the scope of trend tracing is limited from the first to the 15th monitoring interval.

Like graph (a), graph (c) depicts the trends in cost, benefit, and *c.score* values when initially positioned participant systems operated statically. However, in contrast with graph (a), graph (c) represents an environment in which weather conditions change more rapidly. As the result, where graph (a) shows gradual trends change, graph (c) shows acute decrease in *c.score* starting from the second monitoring interval where the participants begin to fall into service unavailable status. The point of time when all participant become unavailable remains the same at the 14th interval, but the average *c.score* during 15 monitoring intervals was significantly lower in the weather turbulence in the environment of graph (c), measured at -3.3 which is much lower than -1.09 of graph (a). Rapid weather changes accelerated the occurrence of constraint violations in participant unmanned vehicles, drastically reducing collaboration efficiency among participants.

Unlike graph (a) and (c), the two graphs (b) and (d) on the right side of Fig. 3 show collaboration scores of UFM IT Ecosystem where the proposed global adaptation mechanism is applied. In contrast with (a) and (c) where dynamic reconfiguration is not in effect, it can be seen that measured cost, benefit, and *c.score* values are stable at all monitoring intervals regardless of weather conditions. Initial configuration of participants in each forest zone was identical as graph (a) and (c). Likewise, the first participant to become unavailable occurs on the second interval, as the result of local internal adaptations in each participant that leads to a constraint violation. However, the values captured in graph (b) and (d) indicate that the global adaptation executed by a *Team Leader* participant ultimately ensured sustained service where unavailable participants were replaced with the most appropriate replacement participant.

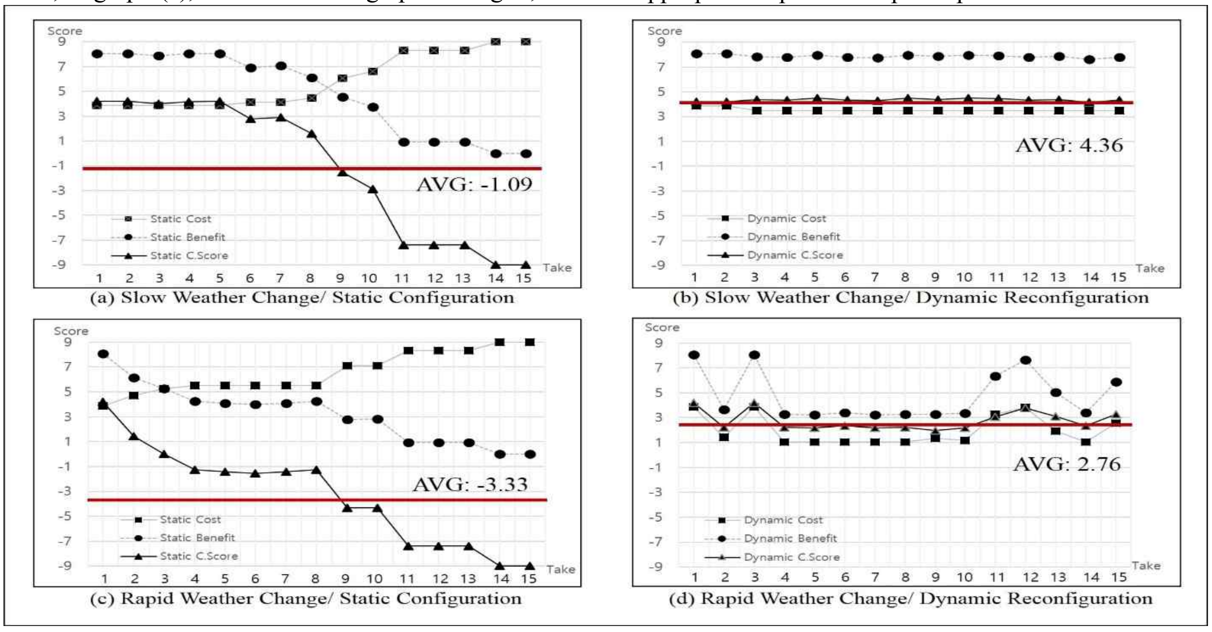


Figure 3. Calculated results of cost-benefit value and collaboration score on selected optimal configuration extracted from each take: (a)(c) without global adaptation mechanism vs. (b)(d) with global adaptation mechanism

In case of graph (b) where weather changes were relatively mild, the value trends are stable without any major fluctuation with average *c.score* at 4.36. This value is significantly higher than the average of -1.09 in graph (a) where no global adaptation cycles were applied. In case of graph (d) where rapid weather changes took place, value changes in cost, benefit, and *c.score* can be observed. However these changes are minor in comparison with graph (c) where global adaptation was not in use: the average in graph (d) is 2.76, considerably higher than -3.3 in graph (c).

Even acknowledging the limited nature of simulated environment test results, it can be safely assumed that the proposed framework's local adaptation and global adaptation mechanism played a positive role in ensuring sustainability of services that are vital in completing the goal of the entire IT Ecosystem.

V. RELATED WORK

There are several frameworks for single self-adaptive systems, such as Rainbow [5], MUSIC [6], and DiVA [7]. Such frameworks are invented to support MAPE-K adaptation control loops. Rainbow [5] framework introduced a reusable infrastructure as to separate concerns between adaptation and application logic, thereby providing architecture-based self-adaptability. While the reusable infrastructure enables self-adaptation with relatively small cost and effort, the Rainbow framework is limited in that its scope supports self-adaptation only in certain situations when situation-specific action rules are applicable. MUSIC [6] combines previous component-based development methods with Service Oriented Architecture (SOA) in that it breaks down all necessary components of self-adaptation into business logic, context awareness, and adaptation concerns as to respond to the distributed and dynamic requirements in mobile environments. However, MUSIC is limited in its need for manual adaptation plan update or replacement because the framework does not include goal management features in its MAPE-K self-adaptation layers. DiVA [7] mainly provides methodologies and framework for developing self-adaptive systems and for managing variability of self-adaptive systems. Its architecture is based on the characteristics of aspect-oriented programming and supports self-adaptation through dynamic addition of appropriate aspects in the form of plug-ins.

While existing works have differentiated benefits, they share the common limitation that self-adaptation is limited to single systems with focus on local adaptation. As their architecture is proposed as conceptual models, developers implementing self-adaptive applications in real-life must rely on their own experiences to find working solutions in their actual environments.

VI. CONCLUSIONS AND ON GOING WORK

In this paper, we have proposed an adaptation framework supporting local adaptation for individual participant system as well as global adaptation across the entire IT Ecosystem. Where existing framework architectures mainly focus on the concept of adaptation, our research details components at a more concrete level. An IT Ecosystem literally creates an ecosystem composed of individual systems assigned to achieve a common goal even without human intervention. In this context, reconfiguration is a core capability in maintaining overall balance and sustainability in service operation across domains covered by IT Ecosystem. Our work provides optimal configuration in response to environmental changes. Quantitative evaluation shows that the proposed dynamic reconfiguration framework helps IT Ecosystem provide sustainable services even in frequent environmental changes and through successive failures in its participant systems.

In our future research, we plan to continue our designed experiments to quantitatively verify how genetic algorithm reduces the overhead from adaptation cycles to determine optimal configurations. Further, we will continue to self-evaluate as we extend our framework to other domains of IT Ecosystems.

ACKNOWLEDGMENT

This work was supported by the Industrial Convergence Foundation Technology Development Program of MSIP/KEIT [10044457, Development of Autonomous Intelligent Collaboration Framework for Knowledge Bases and Smart Devices] and Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning [No. 2012M3C4A7033 348].

REFERENCES

[1] A. Rausch, J. Muller, D. Niebuhr, S. Herold, and U. Goltz, "IT Ecosystems: A new paradigm for engineering complex adaptive software systems," In Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on, pp. 1-6, 18-20 June 2012.

[2] K. Manikas and K. M. Hansen, "Software ecosystems - a systematic literature review," Journal of Systems and Software, vol. 86(5), pp. 1294-1306, 2013.

[3] IBM Autonomic Computing Architecture Team, "An Architectural Blueprint for Autonomic Computing, Tech.Rep," IBM Hawthorne, NY, USA, June 2006.

[4] http://www.osgi.org/Specifications/HomePage

[5] D. Garlan, S. Cheng, A. Huang, B. Schmerl, and P. Steenkiste. "Rainbow: architecture-based self-adaptation with reusable infrastructure," , IEEE Computer, vol. 37(10), pp. 46-54, 2004.

[6] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. A. Papadopoulos. "A development framework and methodology for self-adapting applications in ubiquitous computing environments," Journal of Systems and Software, vol. 85(12), pp. 2840-2859, December 2012.

[7] A.Z, M. Araujo, F. Kuiper, D. Valente, J. Wenkstern, R.Z. "DIVAs 4.0: A Multi-Agent Based Simulation Framework," Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on, pp.105-114, Oct. 30 2013-Nov. 1 2013.

[8] http://en.wikipedia.org/wiki/Representational_state_transfer