GCUCE: Grid Computing for Ubiquitous Computing Environment

Dong-Bum Seo, Tae-Dong Lee and Chang-Sung Jeong

Abstract— **In this paper, we describe GCUCE (Grid Computing for Ubiquitous Computing Environment) core service modules which are composed of context, environment, event, task, space management for application mobility. They are interacting with each other based on context-awareness infrastructure which provides intelligent functions to users. Moreover, we present a unified ubiquitous service interacting with Grid core service modules on Computation Grid and Access Grid computing, which provides scalability and collaboration. Also, we describe an enterprise service model, and base on that model, shall show the performance of GCUCE by performing several experiments for DOWS (Distributed Object-oriented Wargame Simulation).**

Index Terms—**Ubiquitous Computing, Context-Aware, Context Reasoning, Access Grid, Computation Grid.**

I. INTRODUCTION

Ubiquitous computing presents both an opportunity for users and a challenge for system engineers [4]-[7]. For users, a wealth of computing, informational, and communication resources available everywhere should allow them to work more effectively. For system engineers, however there is a challenge of using these resources without overburdening users with management of the underlying technology and infrastructure. One particularly important aspect of this problem is to support continuity in the face of time varying resources.

While ubiquitous computing promises to make many more resources available in any given location, a set of resources that can be used effectively is subject to frequent change because the resource pool itself can change dynamically, and a user may move to a new environment, making some resources available and others not accessible. As we detail later, traditional solutions normally associated with mobile computing [2],[15] are inadequate to solve this problem either because they are unable to exploit resources as they become available in a user's environment, or because users must pay too high price to manage those resources.

For the solution to this problem, we insert the concept of Grid computing [1]-[2] into the existing concept of ubiquitous computing [3]. Many ubiquitous computing systems provide application developers with a powerful framework [18]; however, its design is not intended to support applications either requiring many resources that are needed to integrate instruments, displays, computational and information managed by diverse organizations, or requiring collaborative environment by audio/video conferencing.

Moreover, the existing ubiquitous systems do not consider coordinating and managing the resources to complete the task efficiently and effectively. The structure of the paper is as follows: In section Ⅱ we describes GCUCE(Grid Computing for Ubiquitous Computing Environment) architecture. In section Ⅲ, we discuss context reasoning, and in section Ⅳ mobility in GCUCE. In section Ⅴ, we describe two experiments on Grid computing, and in section Ⅵ, conclusions are given.

D. B. Seo is with Ph. D program at the Department of Information and Communication Engineering in University of Korea, Student Member, IEEE (e-mail: treeline@korea.ac.kr).

T. D. Lee, was with Ph. D program at School of Electronics Engineering in Korea University, Korea, Now he is working at Digital Media, Samsung Electronics (e-mail: leetd@korea.ac.kr).

C. S. Jeong is the corresponding author, a Professor at the Department Electronics Engineering in Korea University, Korea (e-mail: csjeong@korea.ac.kr).

II. ARCHITECTURE

GCUCE is a context-aware ubiquitous computing [20]-[21] environment supported by grid computing [1]. It is composed of three layers as shown in Fig.1: Grid Layer, Context-aware Layer, and Ubiquitous Main Layer.

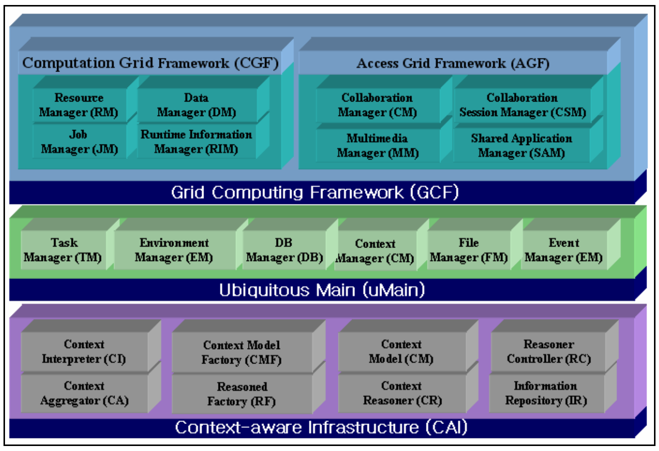

Fig. 1. GCUCE Architecture

A. *Grid Layer*

The Grid Layer [9]-[12] is divided into two elements: Computation Grid Framework (CGF), Access Grid Framework (AGF). The CGF provides the functions of grid computing using a Java cog kit which supports the fault tolerance and high performance computation through resource sharing, monitoring, and allocation. The CGF is composed of four managers: Resource Manager(RM), Data Manager(DM) , Job Manager(JM), and Runtime Information Manager(RIM). RM uses the resource management services offered by Grid, DM provides speed and reliability for files being transferred, JM acts as an agent for the tasks in a job, providing a single entity from which the tasks will request resources, and RIM provides the information to applications or middleware. AGF supplies collaboration with audio/video streaming and view of sharing through shared applications on collaborative environments [2]. It consists of four managers: Collaboration Manager(CM), Collaboration Session Manager(CSM), Shared Application Manager(SAM), and Multimedia Manager(MM). CM gathers the information about the shared stubs from the venue, CSM provides venue addresses that the user can access, SAM integrates the whole features of the shared applications used in the AccessGrid, and MM controls the base service of AccessGrid like Audio/Video streaming service.

B. *Context-aware Layer*

The Context-aware Layer [20]-[21] provides a Context-Aware Infrastructure(CAI) which has a responsibility for functions which support the gathering of context information from different sensors and the delivery of appropriate context information to applications. Also, it supports the context model by ontology methods. The development of formal context models satisfies the need to facilitate context representation, context sharing and semantic interoperability of heterogeneous systems. It provides an abstract context ontology that captures general concepts about basic context, and also provides extensibility for adding domain specific ontology in a hierarchical manner. In CAI, there are eight elements: Context Interpreter(CI), Context Aggregator(CA), Context Model Factory(CMF), Context Model(CM), Reasoner Factory(RF), Context Reasoner(CR), Reasoner Controller(RC), and Information Repository(IR).

CI gathers contextual information from sensors, manipulates the contextual information, and makes it uniformly available to the platform. CA processes and aggregates the data through sensor network after context extraction, which provides high-level contexts by interpreting low-level contexts. CMF defines a context based on concept of specific domain ontology through internal/external providers, and associates a data set with some reasoners to create a CM. RF creates the specified reasoners, and CR has the functionality of providing the deduced contexts based on direct contexts, resolving context conflicts and maintaining the consistency of IR, and RC starts and stops the specific CR.

C. *Ubiquitous Main Layer*

The Ubiquitous Main Layer [5]-[8] (called uMain) is responsible for shielding the user from the underlying complexity and variability through self-tuning environment by mobility and adaptation which are weak points in CGF and AGF. Whenever the user moves from one place to another, the tasks and devices such as grid authentication, environment variables, video/audio device or large display are automatically set up or executed, keeping their environment. The uMain has six managers: Task Manager(TM), Environment Manager(EM), DB Manager, Context Manager, File Manager, and Event Manager(EVM). TM have something concerned with user's task processes. EM supports services concerned with making same user's environment in everywhere. The recording about user information is managed by Database Manager DBM. CM has a role of detection about user's activities such as entering or leaving to/from the environment. FM has to take a charge for both remote and local file operations. EVM is to send and receive messages among them locally.

III. CONTEXT REASONING

The basic concept of our context model is based on ontology[16],[18] which provides a vocabulary for representing and sharing context knowledge in a pervasive computing domain, including machine-interpretable definitions of basic concepts in the domain and relations among them. An ontology-based model for context information allows us to describe contexts semantically in a way which is independent of programming language, underlying operating system or middleware.

A. *Context Ontology*

There are several reasons for developing context models based on ontology. The use of context ontology enables computational entities such as agents and services in ubiquitous computing environments to have a common set of concepts about context while interacting with one another.

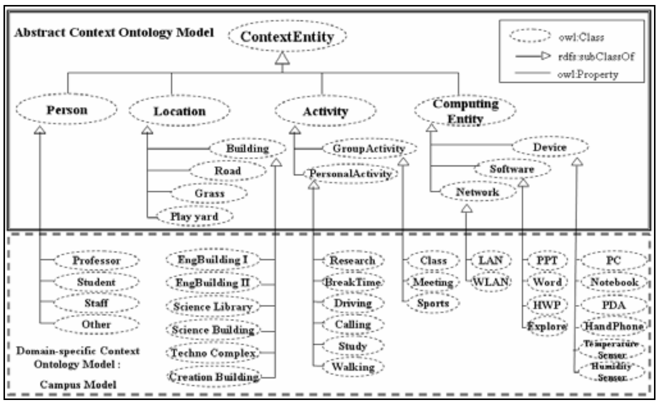

Fig.2. Context Ontology Campus Model

This is knowledge sharing. Based on ontology, context-aware computing can exploit various existing logic reasoning mechanisms to deduce high-level, conceptual context from low-level, raw context, and to check and solve inconsistent context knowledge due to imperfect sensing. By reuse of well-defined ontologies of different domains, we can compose large-scale context ontology without starting from scratch. Fig.2 shows the context model, which is divided into two layers: abstract context ontology and campus domain context ontology. The abstract context model is structured around a set of abstract entities, each describing a physical or conceptual object including Person, Location, Activity, and Computational Entity, as well as a set of abstract sub-classes. Each entity is associated with its attributes (represented in owl:DatatypeProperty) and relations with other entities (represented in owl:ObjectProperty). The built-in OWL property owl:subClassOf allows for hierarchically structuring sub-class entities, thus providing extensions to add new concepts that are required in a specific domain. Besides general classes defined in abstract ontology, a number of concrete subclasses are defined to model specific context in a given environment. The campus domain ontology for specific domain model is depicted with inheritance from abstract context model (e.g., the abstract class GroupActivity of campus domain is classified into three sub-classes Class, Meeting, and Sports).

B. *Ontology Relationship*

Fig.3 shows a simple definition of specific ontology for a campus application domain. The Student inherited from Person is engaged in Class with Notebook in EngBuilding I. Where each concrete values are set such as Student name, Class lesson, Notebook owner, and EngBuildingI roomnumber.

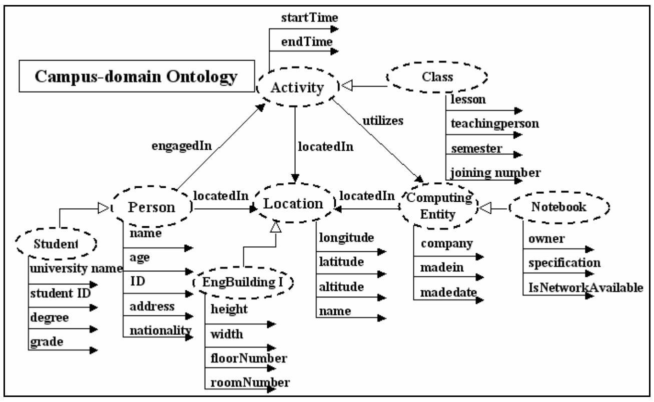

Fig.3. Simple Context Ontology Relationship

IV. MOBILITY IN GCUCE

The system architecture in uMain has the components: ubiCore and ubiContainer as shown in Fig.4. The ubiCore is responsible for the application mobility [20]. When a user moves from one place to another; ubiCore provides the automatic movement of computing environment through ubiContainer which supplies the user information such like IP, user preference, etc.

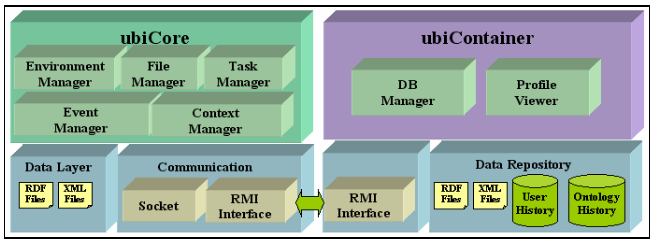

Fig.4. Architecture of uMain

The communication component in GCUCE uses socket, Java RMI (Remote Method Invocation). The socket is used by File Manager for transferring the files. RMI is Java mechanism for supporting distributed object based computing, and it allows client/server based distributed applications to be developed easily because a client application running in a Java virtual machine at one node can invoke objects implemented by a remote Java virtual machine (e.g., a remote service) in the same way as local objects.

The RMI mechanism in Java allows distributed application components to communicate via remote object invocations, and enables applications to exploit distributed object technology rather than low level message passing (e.g., sockets) to meet their communication needs.

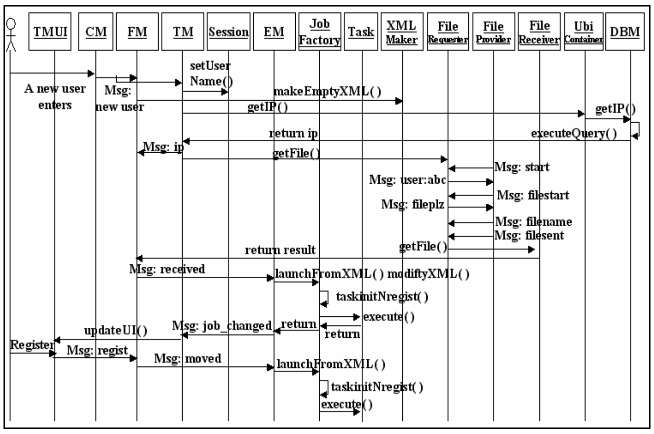

Fig.5. Sequence Diagram of uMain

In Fig.5 shows the sequence diagram of uMain. When a new user enters into a new place or device, CM detects the entrance of the user, and TM brings the information related to the user, and FM makes the directory. TM copies the files associated with the user by File Requester. File Requester sends and receives messages to File Provider on remote host, and File Provider sends the files to File Receiver. After the end of file copies, EM executes the registered tasks through JobFactory which commands the start of tasks to Task. If a user registers the task through the TMUI(TM user interface), the related files are moved to a user directory and the XML file is updated.

A. *Enterprise Model*

Let us think about mobility the enterprise model of competitive ubiquitous grid service [8]-[9], [22]. Assume a set G of N grid service providers. That is, $G = \{1, 2, \ldots n, \ldots N\}$. Each provider can be distinguished by three service parameters such as $\{r_n, l_n, m_n\}$, where r_n is the response time of nth service provider for unit resource demand from its subscribers and l_n is the loss probability experienced by that service provider, and m_n is the mobility probability accumulated by nth service provider. The selection of these parameters has a significant impact on completion of the job of ubiquitous users within a limited time frame. Due to the competitiveness in between the service providers, the ubiquitous user gets an option to shift from one service provider to another for job completion at the earliest possible time. So, the nth grid service provider experiences a demand d_n, which depends not only on his own response time, loss probability, mobility probability but also on the response time, loss probabilities, and mobility probabilities offered by its competitors. So, d_n depends upon entire response time vector $r = [r_1, r_2, \ldots, r_N]$, loss probabilities vector $l = [l_n, l_2, \ldots, l_N]$, and mobility probabilities vector $m = [m_1, m_2, \ldots, m_N]$. The strategy of each grid service provider will be always to provide a response time, loss probability, mobility probability which are in between the maximum and minimum values offered by all of its competitors. Then, the strategy space, S_n of *n*th grid service provider, as in

$$S_n = \left\{ \begin{matrix} (r_n, l_n, m_n) : 0 \le r_{\min} \le r_n \le r_{\max}; \\ 0 \le l_{\min} \le l_n \le l_{\max}; 0 \le m_{\min} \le m_n \le m_{\max} \end{matrix} \right\}.$$

We assume $r_{\min}$ depends on l_n in the sense that if the value of the loss probability increases, then the service provider has to decrease its response time r_n. The upper bound on the response time and loss probability express that after a certain value, the demand will be zero.

B. Demand Model

For a particular grid service provider, the demand d_n for its services decreases as its response time r_n increases; on the other hand, it increases with the increase of its competitor response time r_m, for $m \ne n$. The analogous relationship holds for loss probabilities, but then, d_n increases with decrease of l_n and increase of l_m, for $m \ne n$. The reverse relationship holds for mobility probabilities, but then, d_n increases with increase of m_n and decrease of m_m, for $m \ne n$. We now consider the case where the demand function (d_n) is linear in all QoS parameters [9]. That is,

$$d_n(r,l,m) = \left\{ \begin{matrix} \sum_{m \subseteq G, m \ne n} \alpha_{nm} r_m - \theta_n r_n + \\ \sum_{m \subseteq G, m \ne n} \beta_{nm} l_m - \delta_n l_n - \\ \sum_{m \subseteq G, m \ne n} \chi_{nm} m_m + \eta_n m_n \end{matrix} \right\} \qquad (1)$$

where $\alpha_{nm}, \theta_n, \beta_{nm}, \delta_n, \chi_{nm}, \eta_n$ are constants, and m_n is mobility value at *n*th grid service provider. Here we make some minimal assumptions regarding the demand function.

$$\left[\begin{matrix} \frac{\partial d_n(r,l,m)}{\partial \gamma_n} \le 0, \frac{\partial d_n(r,l,m)}{\partial l_n} \le 0, \frac{\partial d_n(r,l,m)}{\partial m_n} \ge 0; \\ \frac{\partial d_n(r,l,m)}{\partial \gamma_m} \ge 0, \frac{\partial d_n(r,l,m)}{\partial l_m} \ge 0, \frac{\partial d_n(r,l,m)}{\partial m_m} \le 0; \\ \forall m \ne n \end{matrix} \right]$$

This results in a decrease of own demand while increasing those of its competitors, if a service provider increases response time (loss probability), assumption the demand d_n is non-negative over the strategy space (i.e., response time and loss probability are decreasing) and negative over the non-strategy space (i.e. response time and loss probability are increasing). Now each service provider n will charge a cost, n_c, per unit of the demand provided to the ubiquitous user. Then the *gain* earned by the *n*th service provider is

$$G_n = c_n \times d_n(r,l,m) = c_n \times d_n \qquad (2)$$

We define that ε is the ratio of proportionate change in quantity demanded to proportionate change in cost

$$\varepsilon = -\frac{(\partial d_n)/d_n}{(\partial c_n)/c_n} = -\left(\frac{\partial d_n}{\partial c_n}\right)\left(\frac{c_n}{d_n}\right) \qquad (3)$$

Since the revenue of *n*th service provider is given by $G_n = c_n \times d_n$, then taking the partial derivative of both sides we get,

$$\left[\begin{matrix} \frac{\partial G_n}{\partial c_n} = d_n + c_n \times \frac{\partial d_n}{\partial c_n} \\ \frac{\partial G_n / \partial c_n}{G_n} = \frac{(d_n + c_n \times \partial d_n / \partial c_n)}{G_n} \\ \frac{\partial G_n / \partial c_n}{G_n} = \frac{1-\varepsilon}{c_n} = -\left(\frac{\varepsilon - 1}{c_n}\right) \end{matrix} \right] \qquad (4)$$

Now integrating Equation (2) and considering the initial value to demand as k, and then we get,

$$d_n = \frac{k}{c_n^{\varepsilon}} \Rightarrow c_n = \left(\frac{k}{d_n}\right)^{1/\varepsilon} \quad (k \text{ is initial value}) \qquad (5)$$

Equation (5) represents a more generalized demand function by incorporating constant price elasticity model which is more sensible and appropriate for our scenarios.

C. Gain maximization

The revenue maximization problem is

$$\max_{(c_n, d_n)} G_n = c_n \times d_n(r,l,m) \qquad (6)$$

Subject to the following constraints:

$$c_n \ge 0, \forall n \in N \qquad (7)$$

$$\left\{ \begin{matrix} \sum_{m \subseteq G, m \ne n} \alpha_{nm} r_m - \theta_n r_n + \sum_{m \subseteq G, m \ne n} \beta_{nm} l_m - \delta_n l_n - \\ \sum_{m \subseteq G, m \ne n} \chi_{nm} m_m + \eta_n m_n \le d_n(r,l,m), \forall n \in N \end{matrix} \right\} \qquad (8)$$

In the above formulation, the cost and demand variable $d_n(r,l,m)$ are both present. We will find it convenient to replace the price variable by Equation 6 and retain only the demand variables. The optimum price may be recovered from the demands in the solution. Transforming the objective function gives: max

$$\max_{(c_n, d_n)} G_n = \left(\frac{k}{d_n(r,l,m)} \right)^{1/\varepsilon} \times d_n(r,l,m)$$
$$= k^{1/\varepsilon} \times d_n(r,l,m)^{\frac{\varepsilon-1}{\varepsilon}} \quad (9)$$

subject to the constraints in (7), (8). Here we use a Lagrange multiplier λ, associated with the constraint implied by demand satisfaction in (9) to form the Lagrange expression

$$L(d_n(r,l,m),\lambda) = k^{1/\varepsilon} \times d_n(r,l,m)^{\frac{\varepsilon-1}{\varepsilon}} + \lambda \left(d_n(r,l,m) - \left(\begin{array}{l} \sum_{m \subseteq G, m \neq n} \alpha_{nm} r_m - \theta_n r_n + \\ \sum_{m \subseteq G, m \neq n} \beta_{nm} l_m - \delta_n l_n - \\ \sum_{m \subseteq G, m \neq n} \chi_{nm} m_m + \eta_n m_n \end{array} \right) \right) \quad (10)$$

Now the first-order condition for maximization of $L(d_n(r,l),\lambda)$ is found by equating the partial derivative of L to zero. Thus,

$$\frac{\partial L}{\partial d_n} = 0 \Rightarrow \left[\frac{\varepsilon - 1}{\varepsilon} \left(\frac{k}{d_n(r,l,m)} \right)^{1/\varepsilon} + \lambda = 0 \right]$$
$$\Rightarrow c_n = \left(\frac{\varepsilon - 1}{\varepsilon} \right) \lambda \quad (11)$$

So, the strategy of each grid service provider are will be always to provide a response time, loss probability, mobility probability which are in between the maximum and minimum values offered by all of its competitors.

V. Experiment

We use DOWS (Distributed Object-oriented Wargame Simulation) on RTI (RunTime Infrastructure) on Grid [3, 19]. DOWS is an object-oriented simulation system based on a director-actor model which can be mapped efficiently on object-oriented and distributed simulation. The existing RTIs for HLA (High Level Architecture) do not consider coordinating and managing the resource for distributed simulation to complete the simulation efficiently and effectively. The RTI on Grid is a grid-enabled implementation of RTI for solving the problems.

In the scenario as shown in Fig.6, John wants to know the result when DOWS simulation ends while he moves around the building, and continues to do other tasks. Currently, John is at lab B, and registers DOWS and RTI on computation Grid, and executes DOWS on RTI. After execution, he leaves the lab, and will go into the meeting room (Room C in Building C) which is 30 minutes away. The simulation will end 20 minutes after execution, and is transferred to GCUCE as text result. When John arrives at Room C in Building C, he will receive and analyze the result file. For test, the implementation is accomplished on 4 PCs as clients, and 10 clusters (5: Pentium IV 1.7GHz, 5: Pentium III 1.0GHz Dual) and one 486 computer as servers on a VO.

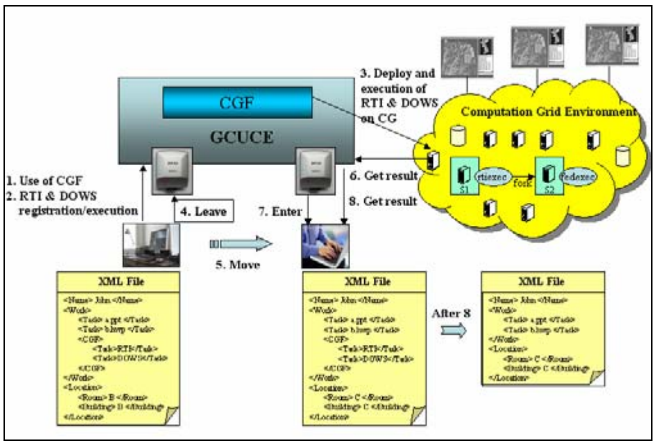

Fig.6. Testbed of DOWS on RTI using Grid for GCUCE

Our experiments are accomplished to confirm key services of CGF. The first experiment is for the automatic distribution service. We organize the system which has five servers (we assume that 486 computer is included as server, because of the limitation in local condition), and the GCUCE which has a VO (Virtual Organization) of 11 servers (10 clusters and 486 PC). Then, we estimated the complete time of simulation as the number of forces increases. As we expected, the resource selection of the GCUCE did not choose the 486 computers.

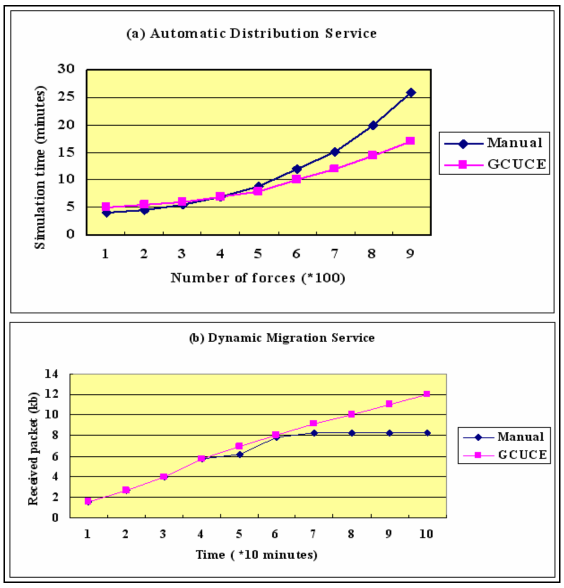

Fig.7. Evaluation and result of GCUCE

As shown in the result of experiment, the demand in GCUCE is increasing because the response time in automatic distribution test is shorter, loss rate in dynamic migration test is smaller, and mobility rate in dynamic migration test is higher. In Fig.7 (a), GCUCE is superior as the scale of simulation is increasing, although the time consumption of initialization have an effect on the state of small forces. GCUCE can utilize abundant computing power and adapt for various environments, as well as provide convenient user interface. This brings a fast response time, and the demand becomes larger. To verify the dynamic migration service, we execute a second experiment. In this test, we measured accumulated received packets updated by 600 forces per 30 second. One packet has the size of 100 bytes. In 70 minutes, we intentionally made a failure on one server. The information related to execution like object information is stored periodically, and then the application resigns from the execution. The application sends the stop message to applications before resignation. RM gathers the information, and DM sends the application to a selected host. The application sends the restart message to all applications and receives the data stored before failure. As shown Fig.7 (b), GCUCE can fulfill its mission after the failure, while the original DOWS is halted. The enterprise model shows that both responses time and loss probability are decreased, mobility is increased, and the demand of this service is increased.

VI. Conclusion

In the paper, we have described the development of GCUCE that is the unified ubiquitous computing environment using grid computing. The GCUCE is composed of the modules which support the unified efficient ubiquitous computing interacting with the core service modules among high performance computing, collaborative computing and context-aware infrastructure.

For the traditional distributed computing model, we have developed two frameworks based on grid computing: Computation Grid Framework (CGF) and Access Grid Framework (AGF). For a large computation application, CGF provides the high performance computing framework which processes the real time data with high speed computation through distributed resources using computation grid. For collaborative computing, AGF supplies the collaborative computing framework which co-works the analysis, agreement and discussion among people with a lot of data using Access Grid. The framework gives the merits of collaboration with multimedia functions.

As a primitive element for ubiquitous computing, the context-aware infrastructure (CAI) provides the intelligent context after process of data entering through sensor network. The context ontology model was suggested, which provides an abstract context ontology that captures general concepts about basic context, and also provides extensibility for adding domain-specific ontology in a hierarchical manner. The development of context models satisfies the need to facilitate context representation, context sharing and semantic interoperability of heterogeneous systems. Also, CAI brings the merits to the system such as adaptation, self organization, and self reconfiguration.

The heterogeneity and mobility among a number of devices for ubiquitous computing make the system vastly more complex. The development for solution of the requirements as ubiquitous infrastructure is needed, and we have developed the uMain (Ubiquitous Main) as the core modules which provides the automatic computing environment, and provides applications or services used everywhere.

For GCUCE, we did several experiments for both DOWS (Distributed Object-oriented Wargame Simulation) on CG and DOWS on AG, including completion time, packet bytes, and frame rate after we explained the enterprise service model, an interactive simulation, and RTI-G (RunTime Infrastructure on Grid). The experiments showed the superior result of the performance of enterprise service model for GCUCE.

References

[1] I. Foster, C. Kesselman and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” International J. Supercomputer Applications, 15(3), 2001.

[2] R. Stevens, M. E. Papka, and T. Disz, “The Access GRID: Prototyping the workspaces of the future, Internet Computing, IEEE, Volume: 7 Issue: 4, pp 51-58, July-Aug. 2003.

[3] IEEE Standard for Modeling and Simulation, “High Level Architecture (HLA) Federate Interface Specification,” IEEE Std 1516.1-2000.

[4] M. Weiser, “The Computer for the Twenty-First Century,” Scientific Am., 1991, pp. 94–101.

[5] G. D. Abowed, “Software engineering issues for ubiquitous computing**,”** Software Engineering, 1999. Proceedings of the 1999 International Conference on, 16-22 May 1999 Page(s): 75 – 84.

[6] M. Glesner, T. Hollstein and T. Murgan, “System design challenges in ubiquitous computing environments,” Microelectronics, 2004. ICM 2004 Proceedings. The 16th International Conference on 6-8 Dec. 2004 Page(s): 11 – 14.

[7] Tim Kindberg, A. Fox, “System software for ubiquitous computing,” Pervasive Computing, IEEE, Volume 1, Issue 1, Jan.-March 2002 Page(s): 70 – 81.

[8] N. Davies, A. Friday and O. Storz, “Exploring the grid's potential for ubiquitous computing,” Pervasive Computing, IEEE Volume 3, Issue 2, April-June 2004 Page(s): 74 – 75.

[9] N. Roy, S.K Das, K. Basu, and M. Kumar, “Enhancing Availability of Grid Computational Services to Ubiquitous Computing Applications,” Parallel and Distributed Processing Symposium, 2005 Proceedings. 19th IEEE International 04-08 April 2005 Page(s): 92a - 92a.

[10] I. Foster, C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit,” Intl J. Supercomputer Applications, 11(2): 115-128, 1997.

[11] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, “Condor-G: A Computation Management Agent for Multi-Institutional Grids,” Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.

[12] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, “A Resource Management Architecture for Metacomputing Systems,” Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998.

[13] R. El Azouzi, E. Altman and L.Wynter “Telecommunications Network Equilibrium with Price and Quality-of-Service Characteristics”, Proceedings of the ITC, Berlin, Sept 2003.

[14] Towards Open Grid Services Architecture, http://www-fp.globus.org/ogsa/.

[15] Access Grid, http://www.accessgrid.org/

[16] J. Tsujii, “Domain ontology and top-level ontology: how can we co-ordinate the two?” Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on 26-29 Oct. 2003 Page(s): 814.

[17] Hsin-Chuan Ho and Chao-Tung Yang Chi-Chung Chang," Building an Elearning Platform by Access Grid and Data Grid Technologies", Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE’04), IEEE Press, 2004.

[18] R. Harrison, D. Obst, C.W. Chan, “Design of an ontology management framework,” Cognitive Informatics, 2005. (ICCI 2005). Fourth IEEE Conference on 8-10 Aug. 2005 Page(s): 260 – 266.

[19] P. Ghosh, N. Roy, S. K. Das and K. Basu, “A Game Theory based Pricing Strategy for Job Allocation in Mobile Grids”, Proceedings of 18th International Parallel and Distributed Processing Symposium, Snata Fe, New Mexico, Apr 2004.

[20] T. Gu, H.K. Pung, and D.Q. Zhang, “A middleware for building context-aware mobile services,” Vehicular Technology Conference, 2004. VTC 2004 Spring. 2004 IEEE 59th Volume 5, 17-19 May 2004 Page(s): 2656 - 2660 Vol.5.

[21] W.G Griswold, R. Boyer, S.W Brown, T.M. Truong, “A component architecture for an extensible, highly integrated context-aware computing infrastructure,” Software Engineering, 2003. Proceedings. 25th International Conference, 3-10 May 2003 Page(s): 363 – 372.

[22] N. Roy,“Providing Better QoS Assurance to Next Generation ubiquitous Grid Users”, MS Thesis, University of Teaxs at Arlington, USA, Apr 2004.