On Self Managing Autonomous Ubiquitous Networks

Junaid Ahsenali Chaudhry , Sajjad Hussain Chaudhry , and Park Seungkyu
Department of Information and Communication,
Ajou University, South Korea.

Abstract: *The interfaces of computing systems are expanding. This expansion has revealed many new problems that were never considered for conventional software systems. Ubiquitous networks belong to the generation of computing systems that deals with high levels of mobility and 'interface everywhere' theme. So they require more resilience. Conventional methods for managing these networks are inappropriate and very costly. Autonomic self management is a promising new concept that aims to (i) increase reliability by designing systems to be self-protecting and self-healing; and (ii) increase autonomy and performance by enabling systems to adapt to changing circumstances, using self-configuring and self-optimizing mechanisms. In this paper we propose that the cost attached with network management can be cut short with increased quality of service and performance. We give the architecture of an Autonomic Management System (AMS) in a smart home scenario, which we plan to implement in our project.*

Keywords: Algorithms, artificial intelligence, information systems

1. INTRODUCTION

To date, the natural growth path for systems has been in supporting technologies such as data storage density, processing capability, and per-user network bandwidth, with growth increasing annually for 20 years by roughly a factor of 2 (disk capacity), 1.6 (Moore's Law), and 1.3 (personal networking; modem to DSL [Digital Subscriber Line]), respectively. The usefulness of Internet and intranet networks has fueled the growth of computing applications and in turn the complexity of their administration [1]. The cost attached to manage the complex system of today is a lot more then the actual cost of the system [3]. In such a complex environment, the importance of system reliability and dependability gains more importance. Dependability is defined as the property of a computer-based system that enables reliance to be placed on the service it delivers. The customized service delivered is the result of its behavior as perceived by other systems or its human users [4]. Dependability covers many relevant system properties such as reliability, availability, safety, security, survivability and maintainability [5] [6] [14]. IBM being the leader in taking the initiative in Autonomic Computing [7] has worked for making their products more dependable [2]. Tivoli [8], Lotus [9], IBM WebSphere [10] and IBM DB2 [11] being the prime examples in software. In hardware IBM eServer [12] is among prime examples. Regarding the use of Autonomic computing in ubiquitous systems, it all started off when the systems became very complex mainly because of many types of devices, the ubiquitous systems cater [1]. The over all flux on a ubiquitous becomes so much that it requires a management system separately. The slogan of seamlessness and 'desktop everywhere' increased the importance of some entity that manage and look after the performance of ubiquitous systems. The main purpose of ubiquitous management server is to provide the system much needed device independence. But in ubiquitous systems it has to go through many management functions like self healing, fault diagnosis, fault handling, security, performance management, context awareness and prediction algorithms [13]. These requirements were the reasons for autonomic computing to become a self management tool for huge management systems. Various features of autonomic systems are presented in [15, 16, 17, 18, and 19]. In this paper we present the architecture of an autonomic management system designed for residential gateways. We discuss various service modules included in the autonomic management engine and their working for service management and better quality of service.

2. BACKGROUND

Making Human Nervous System a bench mark, the term autonomic systems was started by IBM in 2001. Since then they have invested billions of dollars to get the real picture. Now the pioneer of Autonomic Computing defines it in [2]. Autonomic computing is making its space in ubiquitous systems because if silent work. It resides in the system and keep tracks of all the dynamics of it. Whenever and wherever needed, it takes preemptive measures accordingly. SELFCON is the architecture for self configuration in ubiquitous networks [15]. SELFCON associates the configuration intelligence with the components of network rather then limit it to a centralized management station. AUTONOMIA is an autonomic computing environment [16]. They propose Autonomic Middleware Service (AMS) the service that takes care of dynamic needs of autonomic applications. The idea of Personal Home Servers is presented in [17]. They believe that personalization is the key to reduce the complexity. To each user, they have provided a server that deals with the service customization and each user communicates

with others via his personal server. The mobility of the user and the service portability is also managed by the same. Embedded micro servers are provided to every physical device in [18] called Pervasive Servers. With emphasis is on user customizability and policy enforcement. They say that the personal server carried by each person coordinates with pervasive servers. This framework is very dynamic and may be a good candidate for dynamic service composition. They have used Universal Plug and Play (UPnP) for service discovery, traditional web server for location management, XML messaging protocol for message passing among the individual modules. The context management is difficult in this environment but they define it as their future goal. In the future ubiquitous P2P communication environment, they emphasize on autonomic service composition. They believe that in place of defining one service for many users then user investing time and effort, or the middleware via behavior learning algorithms, to customize it according to his likings the services consider the user context and dynamically gets published in the network. They define every object as service element and individual service as a service element also. The user Service Composition Core (SCC) technology to enable service elements and make groups, State Acquisition (SA) technology for acquiring the state of service elements and Network Reflective (NR) technology for monitoring the state of the network and keeping track of all the activities going on in the network. In [20] the authors discuss about the architecture of service gateway for smart home. Building on OSGi Platform, they propose that the residential gateways are connected with a management gateway that does the management functions for the residential gateways. And in [21] the authors enumerate challenges in building service-oriented application for OSGi. Since last many years scientists of the whole world are looking for some technology that can assist humans, shoulder to shoulder.

3. AUTONOMIC SELF MANAGEMENT

Software-based systems today increasingly operate in changing environments with variable user needs, resulting in a continued rise of administrative overhead for managing these systems. Thus, systems are increasingly expected to dynamically self-adapt to accommodate resource variability, changing user needs, and system faults. A common approach to adding self-management capabilities to a system is to provide one or more external control modules, whose responsibility is to monitor system behavior, and adapt the system at run time to achieve various goals. Autonomic Self Management is itself a bunch of management function that the system developer aims to add in the system [13]. For many application domains, managing a system requires managing multiple dimensions. For example, consider a video conferencing system with different user applications in a heterogeneous network environment, where the aim is to provide the best service at the lowest cost. At once, several potential dimensions of control exist, including composition, change, performance, and cost of service, each corresponding to different domain expertise, and thus, modules of control. Many self-management modules (SMs) are available today, each typically capable of addressing a distinct aspect of self-management. The challenge is to allow developers to coordinate multiple distinct management modules together in a coherent and consistent fashion to manage a system. When those functions perform the allocated tasks only then we say that the management is enforced hence the self management is a collective function of many sub functions. We discuss those sub functions one by one and how we embed the autonomic functions for bringing self management into action. At the moment we are considering the F(ault), C(onfiguration),A(ccounting),P(erformance),S(ecurity) functions in autonomic self management paradigm.

3.1 Self Fault Management

Network operations are complex and network operators are frequently flooded by a number of event messages when a few network failures occur. These event messages can't be monitored by the operator especially when it comes to ubiquitous networks with large volumes and countless geographic scope. The successful management of a network is the reduction of the alarm volume while improving the information content [31]. The fault identification involves the following steps.

1- Event registry: There is some monitoring 'all time up' service. At some specific points the registry logs manage the event logs.
2- Event Correlation: Some event correlation models are provided to the system. Those models help the system identify the exact nature of the fault and then the faults are directed to related sub system.
3- Fault Contextualization: After identification of a fault alert, the next task is the determination of the nature of the fault. Pattern matching is done to identify the nature of the fault.

Fault can be of two types. 1) Avoidable faults 2) Unavoidable faults [22]. We in our project have categorized the faults into the same two categories. The purpose is to keep the humans away from control loop. Case Based Reasoning is a technique for solving problems based on experience. The technique's intuitive approach is finding increasing use in complex system diagnostics. Each module has some exceptions mentioned and the exception pool is maintained and updated by the user checkup or system update. The cases are defined against

exceptions and the policies are managed to handle the new fault exceptions. In [30] it is said that the CBS is efficient because

a- Reduction in diagnostic time
b- Capture and reuse of a technician's experience
c- Increased acceptance of diagnostic equipment
d- Feedback of field experience into the design process

In case of second type of faults we try to keep the user as away from evaluation as possible and have put him in the last loop. At first when the unidentifiable faults come, the system asks the autonomic recovery engine to generate some service that recovers the system from the chaotic state.

3.2. Self Configuration Management

The main purpose of self-management technologies is the reduction of the cost of network deployment and operations. Providing self-management technologies should increase the number of network elements managed per person or the time needed to operate a network. Additionally, they increase the usability and enable inexperienced users to run network with little networking skills and knowledge. In ubiquitous systems where computers are embedded into the environment and the user is supposed to have no idea about them, self configuration becomes more important. In general, self management technologies increase the autonomy of individual network elements or of small, tightly connected groups of network elements. Network elements take over the significant amount of management functions, reducing the required interaction with the centralized manager to a minimum. Self-Configuration is a part of self-managing systems. It follows the paradigm of Plug-and play for the systems. It mainly covers the initial setup and some maintenance during the life-time of a system. Additionally, it covers the ability to compose system from parts. So when they get plugged together, they should just work as they are supposed to work. The self-configuration itself increases the number of additional removals of systems per working hour. And it is the only way to let inexperienced users to setup a network at all.
In heterogeneous networks, autonomic configuration is very important. It gains more importance in a ubiquitous smart space scenario where seamlessness is a very important measure of Quality of Service (QoS). Not only the Quality of service is a big issue but the device discovery and service discovery are the issues that come into consideration very strongly. The device interface identification is the responsibility of device discovery module residing at service gateway level. Once the context is transferred to the management gateway, it is looked up in the context database by lookup manager. The context is converted into query by lookup manager and the context database is queried. If the software support is present, its handler is passed to residential gateway and it installs that piece of software else the service provider is paged and software support is requested. If even after consulting the service provider, the support is unavailable, a default profile is attached and a generic driver is attached to that device until that device leaves the Area of Influence (AoI). Another scenario we have considered is the identification of a Ubiquitous ID. If that device was attached with some ubiquitous online gateway, the context of that device is requested and downloaded from that gateway. The following is the resource model for interface identification

- The device is detected by Home Gateway.
- Home Gateway makes a Context File
- That Context File is transferred to the Active Directory
- When some entry is found in Active Directory, the u-Configuration module detects it and marks it for processing.
- Intra system support detection i.e. Database lookup
- Inter System support detection i.e. Service provider
- If support found, it is returned to Home Gateway else some default provide is assigned to that device.

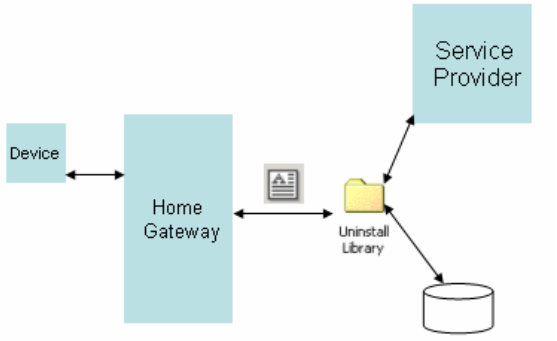

Fig: Resource Model of Interface Identification

Context File: An XML file containing context information of the device
Active Directory: A Directory monitored constantly.

3.3. Self Accounting

Being accountable means being able to provide an explanation for 'why things are this way' that is adequate for the purposes at hand. Accountability becomes even more relevant for systems whose role calls for users to make sense of their behavior precisely and accurately [23]. In autonomic systems the user is pulled out of the execution loop completely so there are very less chances for the system to 'explain' use about the tasks he is performing. The user can either look at the policies or the

event logs that the system has been generating. But it is cumbersome to do. Most often this feature of the autonomic systems is considered as similar to the expert systems. The following figure shows the architecture of expert systems.

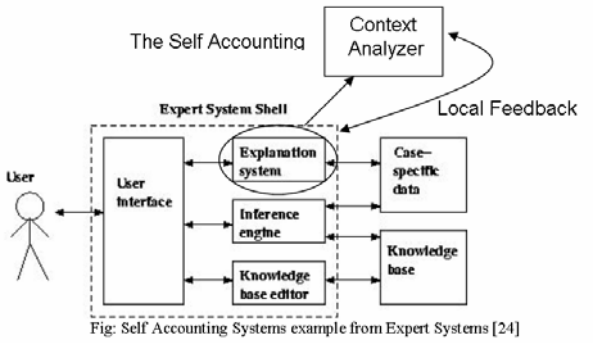

Fig: Self Accounting Systems example from Expert Systems [24]

Fig: Self Accounting Systems example from Expert Systems [24]

In the architecture of the expert system, the explanation system is the facility provided to the user so that he can have a look at the logic that expert system has used to infer the results. It can be in the form of log files of inference engine or some rule it has used from rule base to infer to the results. Putting user out of execution unit can rise some serious issues like system transparency is breached and the user is not trained according to the system. Plus there is no hands-on experience for the user so he remains unaware of the system and hence can not be able to develop the trust on the system. This is a big issue on the part of autonomic systems that they are not accountable to the users.

3.4. Self Optimization

One of the major challenges in the runtime management of computing environments in any enterprise is both the initial optimization of these environments, and keeping these environments optimized when changes occur. In [28] the authors have shown as much as 21.72% improvement in execution time targeting optimization in performance of processors in multi processor systems. We have divided the areas optimization it the following matrix.

- Remote User Optimization
- Service Operation Optimization
- Platform Optimization
- Mobile Device Optimization
- Gateway Optimization
- Database optimization
- Network Optimization

Since optimization of all the network is a super set of optimized components so its hard to cover all of the area mentioned above but in the following passage we discuss them some of them.

3.4.1. Remote User Optimization

The remote user is connected to the system via management server through internet. There are some policies for each connected user. A fault model enforcement technique [27] is used to bring classification among the users and for policy implementation. A good candidate for user session management is proposed in [26]. When the user session starts the policy enforcement entity, and some monitoring entity constitutes to form a user session management layer and that layer is useful for maintaining the session record for future predictions, profile updating and providing better quality of service to remote user.

We in our system have used Session State Management (SSM) technology to deal with the remote user optimization. Following this approach is feasible because of lot of diversity in heterogeneous devices and the complex operations attached to them like user mobility management, prediction algorithms and recourse allocation to each session. In the scenario given, the user requests for connation establishment with the remote server. The server invokes monitoring, policy enforcement, resource allocator and scope allocator service to that session and at the end when user requests for closing the session, the profiles are updated and logs are dealt accordingly.

3.4.2. Service and Service Elements Optimization

Arguably the performance of every system depends upon the performance of services in a Service Oriented Network. The dynamic environments like the one presented in [28]. The authors propose three technologies in Ubiquitous Service Oriented Networks 1- Service Composition Core Technology 2- State Acquisition Technology and 3- Network Reflective Technology. The performance of a service depends upon the performance of these three technologies. The architecture we propose carries OSGi as its platform. We have certain performance templates or profiles. The monitoring service is monitoring the activities and whenever there is some unusual change identified, the profile manager having system rights will change the profile of the system. We haven't selected more complex approaches because implementing those will bring lots of problems for other self management modules. Since in every profile, the parameters for all services is defined. So there will be no clash in the performance of the system.

3.4.3. Database Optimization

Besides all the database level optimization schemes for storage or check of data for cardinality and duplication, we think that in most cases the database level delay is due to

inaccurate querying. The parts of AMS system dealing with database contain intelligent query generators. We are working on implementation of the query generators and they are expected to improve the performance of our system.

3.5. Self Security Management

Security is the weakest link in the ubiquitous systems. The systems that present interface everywhere and are all time any where present are more vulnerable and prone to some external attack. Besides internal malfunctioning or some QoS problems, the external attacks, intrusion detection, and then cure is a very important issue in ubiquitous systems. Arguably the most exposed systems to the external attacks, ubiquitous systems, need such a security mechanism that is always awake and present everywhere alongside the interface.

Here in our system we propose a Natural Immune System (NIS). The human immune system consists of a large number of specialized cells, which operate autonomously and interact with each other to create complex chains of events leading to the destruction of pathogens. Broad categories of cells can however be defined and form two groups: detectors and effectors. The role of detectors is to identify pathogens and the role of effectors is to neutralize pathogens. We follow the same approach.

The NIS launches many detection agents in the network for checking the security level of each part. These agents have some goal set to check the functioning of the related modules and then report back the health report. The other type of agents is effectors. They are reaction agents of the system. Whenever the health report is diagnosed as worth treatment, the agent engine launches the effectors as the cure for that problem and they move autonomously and fix the problems.

One particular aspect of multi-agent systems that we have focused on is their ability to form dynamic social groups. The interest in this behavior stems from the concept that by linking together the sensory and intelligence capabilities of a large number of agents distributed across a network, we can amplify the ability of the network to resist attacks or intrusion. Specifically, through social co-operation agents can benefit from the combined defensive capabilities of their particular group. Keeping all the scenarios in mind, it's a tough job to device a security system for ubiquitous systems. But we have decided to research more in this direction too.

4. CONTEXT AWARENESS

Context awareness is one of the most debated issues around among ubiquitous systems researchers. What and when to gather context are the issues. Since we are benefited by ever increasing performance speed and memory space, we can gather the context of everything happening around but to get the sense of that heap and then collecting the information that can be best suitable in a problem is the biggest issue of all times. Had that problem solved, computers would have been betters computers by now.

We have followed the Gather and Pluck approach (GnP). In this we gather all the events and the logs are forwarded to the Context Control Center (CCC). The CCC gets the logs and identifies the contexts and forwards them into their respective profiles. These profiles are then observed by data mining agents. The data mining agents are included in our future work. After the data mining agents finish their tasks the profiles become light weight and easy to transport over network. This is called Profile Transformation. The profile transportation gives the facility to the systems to share, donate and update profiles easily.

5. THE UBIQUITOUS AUTONOMIC MANAGEMENT SYSTEM

5.1 Description

The Ubiquitous Autonomic Self Management System is an autonomic self management module in our project. We have a scenario of a ubiquitous Smart Home on hands. Every house is connected with a Residential Gateway (RG). That gateway is responsible for providing services to users (devices, human user). The Service Oriented Architecture of the gateway serves as an execution point for the services to interact with the user. Each Residential Gateway is connected with a management gateway. The task of management gateway is to manage the gateways and services running on them.

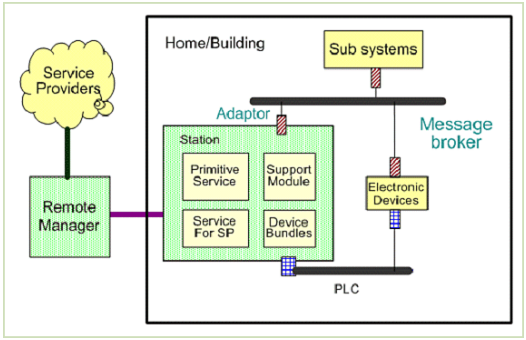

Fig: A Connected Smart Home

Since the scope of our project is lot broader then one management gateway we can face the following problems if we manually manage the over all system.

Ubiquitous Computing has eliminated almost all manual tasks so manual management will not be suitable for such systems. Or in other words, human intervention will not serve the purpose of seamlessness. The system is

so complex that we need to have some system generated self monitoring and managing service. The administrating cost goes so high if we count the over all system that it's not possible to run this business system.
So we decided to use an autonomic self management module in our project. The following figure shows the over all system overview. There are many remote managers, serving various gateways. Every house has its own gateway so a house is represented by a gateway. Connected remote managers will broad the scope when every single house is accessible from various remote locations.

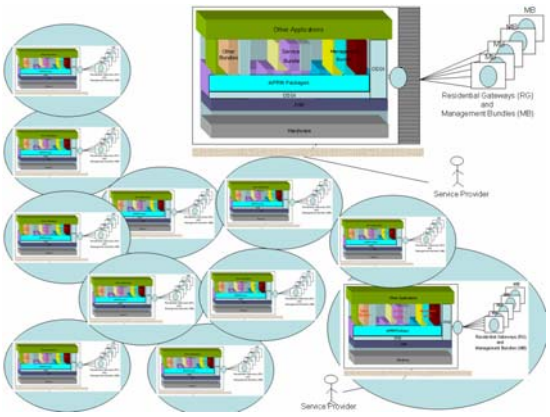

Fig: Remote Manager Scenario

5.2Architecture

In the following context we discuss the architecture of u-AMS. There are some core services that directly interact with the user. These are platform independent services that are either provided by service provider or generated by the system itself. These services are initiated when they are loaded from the service pool at the initialization of management gateway. Later on the service pool is updated as the new services are added or old services are updated. In u-AMS the services are called service bundles and service pool as u-function bundles pool. In that service pool all services individually and service pool collectively is monitored by the monitoring service. It is the logging system utility that gives the facility to monitor every instance taking place within the system. That monitoring services leads to Context Control Center (CCC). This part of system manages the profiles and reports to the context analyzer. It works out on the log files generated as a result of all the monitoring activities and hence it updates the profiles of all the entities. This part of u-AMS is even driven it goes through all the activities and if some activity other then normal pattern is found, it reports to context analyzer. The Context Control Center is a kind of exchange that diverts the related to the relevant profile or things out of context (exceptions) to context analyzer. The profile, if left unexamined, will become too bulky. So to reduce the size and for more efficient processing some data mining functions are performed on profiles. These data mining functions are performed through data mining agents. There is some normal profile of each entity and that is defined when it first gets attached to the system or the system software defines it default operation. When there is something happening other then the default operation that is separated and recorded. That makes the information size low (data mining agents are included in our future implementation plans).

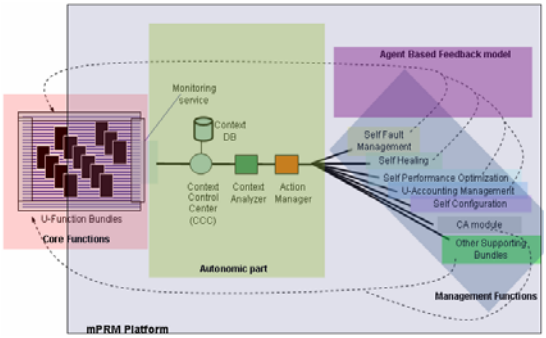

Fig2: System Architecture of u-ASM system

When some exception or malfunctioning is reported, the context handler is passed to context Analyzer module. The function of this part is mainly the pattern matching. Using Pattern matching, it identifies the problem and handles the context to the Action Manager.
Action manager is mainly the policy checker. It checks the policies and takes the appropriate action as directed in policy. The type of violation is important here. If the breach is of the sever type, or it is of a type that system can handle with the help of policies. This part of the system predicts the level of reaction to the breach or malfunctioning.
After the analysis of the problem and its identification, the context is handed over to the related management module. A reaction policy is devised by that management module and Agent Based Feedback Model is actuated and that policies are implemented on the services. We have chosen Agent based technology over here by considering the scenario in mind that the services may reside at remote place so agents can be a good candidate for execution of remote action.

6. IMPLEMENTATION

We plan to implement our work using toolkit provided by IBM for autonomic systems. We first make resource model and then those resource models are implemented using that toolkit. Behind any autonomic computing

system is some form of autonomic management engine. This engine acts as the hub of the wheel, keeping all of the spokes together and pointed in their proper directions. An autonomic management engine such as IBM's Autonomic Management Engine (AME) coordinates the control loop for an autonomic computing system. In a control loop, a system continually monitors the environment, analyzes the information received, plans a response, and executes that response. If you had to manage a new autonomic application for every issue you had, you wouldn't be saving much in terms of labor. You'd also start to run into problems in terms of standardization between systems, increased software complexity, and so on. Instead, autonomic computing technology works on the idea of a single management engine running multiple resource models. A resource model is a set of criteria and instructions that get plugged in to AME. The resource model defines what resource to monitor (that is, disk space, processor time, and so on), what events and conditions to look for, and what to do in the event that those conditions do in fact occur [32].

7. FUTURE WORK

In this paper we propose the architecture of autonomic self management module that we plan to implement in our project. For a big, all time up system it is very important to have some management facility and since the complexity of the system is so much that the management cost of adopting human monitored management is very high. So we propose an architecture that gives the liberty to the system to make her own decisions and when it feels convenient page the user about her problems. We plan to implement this architecture in our project. The tasks that we aim to do in future are the data mining agent development for better context awareness, fault policy model development, Inference engine for reasoning explanation and security plan for our ubiquitous business management system.

8. ACKNOWLEDGE

This research is supported by the ubiquitous Autonomic Computing and Network Project, Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea." The authors acknowledge all the colleagues in Center of Ubiquitous System, Ajou University South-Korea.

REFERENCES

[1] R.Want, T. Pering, D. Tennenhouse, Comparing Autonomic and Proactive Computing, IBM systems Journal, Vol 42, No1, 2003.

[2] www.research.ibm.com/autonomic

[3] F. P. Brooks, Jr., The Mythical Man-Month: Essays on Software Engineering, Twentieth Anniversary Edition, Addison-Wesley Publishing Co., Reading, MA(1995), p. 226. See also, F. P. Brooks, Jr., “No Silver Bullet: Essence and Accidents of Software Engineering,” Computer 20, No. 4, 10–19 (1987).

[4] B. Randell, “Turing Memorial lecture- Facing up to Faults” , Comp.J. 43(2), pp 95-06, 200.

[5] A. Avizienis, J.-C. Laprie, B. Randell, “Fundamental Concepts of Dependability”, UCLA CSD Report #010028, 2000.

[6] R Sterritt, DWBustard, "Autonomic Computing-a Means of Achieving Dependability?", Proceedings of IEEE International Conference on the Engineering of Computer Based Systems (ECBS'03), Huntsville, Alabama, USA, April 7-11 2003, pp 247-251.

[7] Paulson, L.D., IBM begins autonomic-computing project Computer , Volume:35 , Issue: 2 , Feb. 2002 Pages:25 - 25

[8] http://www-306.ibm.com/autonomic/tivoli.shtml

[9] http://www-306.ibm.com/autonomic/lotus.shtml

[10] http://www-306.ibm.com/autonomic/websphere.shtml

[11] http://www-306.ibm.com/autonomic/db2.shtml

[12] http://www-1.ibm.com/servers/uk/eserver/zseries/ feature040704/

[13] A.G. Ganek, T.A. Corbi, “The Downing of the Autonomic Computing Era”, IBM systems Journal Vol 42 No1, 2003.

[14] H. Morikawa, M. Jeong, and T. Aoyama: "Bringing Flexibility into Ubiquitous Personal Networks", In Proceedings of First International Workshop on Active Network Technologies and Applications, pp. 75-79, Tokyo, Japan, March 2002.

[15] R. Boutaba, S. Omari and A. Virk. SELFCON: An Architecture for Self-Configuration of Networks. International Journal of Communications and Networks (special issue on Management of New Networking Infrastructure and Services), Vol.3. No.4, pp.317-323, 2001.

[16] Autonomia: an autonomic computing environment Xiangdong Dong; Hariri, S.; Lizhi Xue; Huoping Chen; Ming Zhang; Pavuluri, S.; Rao, S.;Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International , 9-11 April 2003

[17] Tatsuo Nakajima, Ichiro Satoh: Personal Home Server: Enabling Personalized and Seamless Ubiquitous Computing Environments. PerCom 2004: 341-345

[18] Tatsuo Nakajima, Servers: A framework for creating a society of appliances, Springer-Verlag London Ltd, Volume 7, Numbers 3-4, March 2003

[19] The ubiquitous service-oriented network (USON)-an approach for a ubiquitous world based on P2P technology Takemoto, M.; Sunaga, H.; Tanaka, K.; Matsumura, H.; Shinohara, E.; Peer-to-Peer Computing, 2002. (P2P 2002). Proceedings. Second International Conference on , 5-7 Sept. 2002

[20] Valtchev, D.; Frankov, I.;Service gateway architecture for a smart home, Communications Magazine, IEEE , Volume: 40 , Issue: 4 , April 2002

[21] Hall, R.S.; Cervantes, H.;Challenges in building service-oriented applications for OSGi Communications Magazine, IEEE , Volume: 42 , Issue: 5 , May 2004

[22]Sterritt,R.;Bustard, D.;McCrea, A.; Autonomic computing correlation for fault management system evolution; Industrial Informatics, 2003. INDIN 2003. Proceedings. IEEE International Conference on , Aug. 21-24,2003 Pages:240 – 247

[23] Anderson, S.; Hartswood, M.; Procter, R.; Rouncefield, M.; Slack, R.; Soutter, J.; Voss, A.;Making autonomic computing systems accountable: the problem of human computer interaction;Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on , 1-5 Sept. 2003

[24]http://www.cee.hw.ac.uk/~alison/ai3notes/subsection2_5_2_1.html

[25] Aiber, S.; Gilat, D.; Landau, A.; Razinkov, N.; Sela, A.; Wasserkrug S.; Autonomic self-optimization according to business objectivesAutonomic Computing, 2004. Proceedings. International Conference on , 17-18 May z`04 Pages:206 – 213

[26] Ling, B.C.; Fox, A.; A self-tuning, self-protecting, self-healing session state management layer; Autonomic Computing Workshop, 2003 , 25 June 2003

[27] Kiran Nagaraja, Ricardo Bianchini, Richard P. Martin and Thu D. Nguyen; Using Fault Model Enforcement to Improve Availability; In Proceedings of 2nd Workshop on Evaluating and Architecting System Dependability (EASY 2002). San Jose, CA, October 202

[28] Takemoto, M.; Sunaga, H.; Tanaka, K.; Matsumura, H.; Shinohara, E.;The ubiquitous service-oriented network (USON)-an approach for a ubiquitous world based on P2P technology Peer-to-Peer Computing, 2002. (P2P 2002). Proceedings. Second International Conference on , 5-7 Sept. 2002

[29] Zhu, H.; Parashar, M.; Yang, J.; Zhang, Y.; Rao, S.; Hariri, S.; Self-adapting, self-optimizing runtime management of Grid applications using PRAGMA; Parallel and Distributed Processing Symposium, 2003. Proceedings. International , 22-26 April 2003

[30] Derere, L.; Case-based reasoning: diagnosis of faults in complex systems through reuse of experience; Test Conference, 2000. Proceedings. International , 3-5 Oct. 2000 Pages:27 – 34

[31] Akbas, E.; System independent and distributed fault management system; Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on , 30 June-3 July 2003 Pages:1359 - 1363 vol.2

[32] Nicholas Chase; Understand the Autonomic Management Engine; https://www6.software.ibm.com/developerworks/education/ac-ame/ac-ame-a4.pdf