Journal of Computer Science 2 (1): 86-91, 2006
ISSN 1549-3636
© 2006 Science Publications

On Recipe Based Service Composition in Ubiquitous Smart Spaces

Junaid Ahsenali Chowdary, [1]Seung-Kyu Park and [2]Suk-Kyo Hong
[1]Graduate School of Information and Communication
[2]Department of Electrical and Computer Engineering, Ajou University, South Korea

Abstract: We present a recipe based service composition scheme in smart spaces. We discuss real time problems of highly synchronized and user customizable processes with the help of an example. On-the-fly Quality of Service (QoS) shows up as a major issue in such environment. Furthermore the QoS of a dynamically composed service depends upon the aggregate QoS of its developing components. The dynamic composition of a service is done through the recipe file which is composed by service provider, it is like the control extension that checks the latest contest of the system from Context Servers and then generates the parameters for required services. Service Composition Manager is the entity that reserves the services from service pool and generates self management services. We setup context servers for context exchange and synchronicity of various processes. The generated service further takes the context from individual devices present in the area of influence and the effective use of context information helps us divide the complex jobs functionality into many simpler jobs on the bases of free time slots available per device. Each device performs its role and reports back to the context aware server and waits for new instruction in the next time slot. The principle objective is the presentation of system requirements of high level of QoS, required in delegating human centric ubiquitous applications. In the end we demonstrate our scheme with the help of an example and discuss its feasibility with the help of our model.

Key words: Recipe based service composition, component based development, ubiquitous systems, autonomic systems

INTRODUCTION

The vision of omnipresent ubiquitous technology is to provide “off-the-desktop” computing that emphasizes on embedding the interface in whole physical world in such a way that the user feels seamless service delivery at anytime and anywhere fashion[1-3]. Building on top of the ubiquitous infrastructure i.e. OSGi ®[4], the Service Oriented Platform (SOP) is considered to be strong candidate for future ubiquitous networks[5]. All such services that reside on a platform can combine to make applications that directly interact with the users[6]. Ubiquitous networks provide services to various devices such as cellular phones, PDAs, laptop computers and many others. That is why they have to manage heterogeneity at high scale. Due to these contemplations, the quick application development is very tough. Atomized home automation solutions were not serving the vision of future community level networks. So it was important to expand the control and increase the usability in ubiquitous society slogan[7].

We simulate parts of the system to check the validity of our system to test the functional feasibility. Using Session Initiation Protocol (SIP) leverage us from many mobility management and device authentication issues[8] that makes it an “on the fly” solution. The target area of the application is ubiquitous smart homes.

Overview and architecture: Here, we present an overview and architecture of the system. The components are defined considering the requirements of the system and maintaining the original layout of present kitchen environment. Figure 1 shows the service architecture of ubiquitous application generated through service integration. The domestic gateway manager is the managing entity present on the gateway and its job is to regulate the services that are being used through a gateway. The framework provides the operational platform to the services to negotiate with the devices, protocols and among the services themselves. These two constitute the platform. The active profile manager runs throughout the application process time and it monitors the whole system and reports to the manager about the changes happening in the dynamic system. It is responsible for maintaining the log files that are used later for reactive actions taken by the management system of the application. The use of resource model based approach is very useful for autonomic reactive actions. The recipe manager, interpreter, actuator unit, pot manager, grocery manager and application manager shown in Fig. 1 are the core components of the system. We call them the service stack. In the scheme we propose, the service stack includes those services that are directly involved in the application development process. There is a service Composition Manager that manages the composition of the management service. It takes the services from the service pool and the latest context from the context servers and composes the Management Services. Service Composition Manager produces the ability of Self Management as management services are developed but Service SCM serves as the bridge between the two paradigms. The details about them are given later.

Corresponding Author: Seung-Kyu Park, Graduate school of information and communication, Ajou University, South Korea

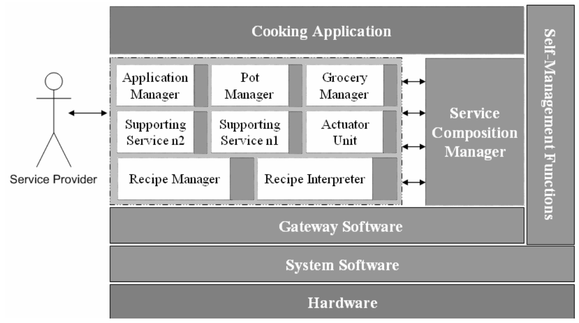


Fig. 1: Service architecture

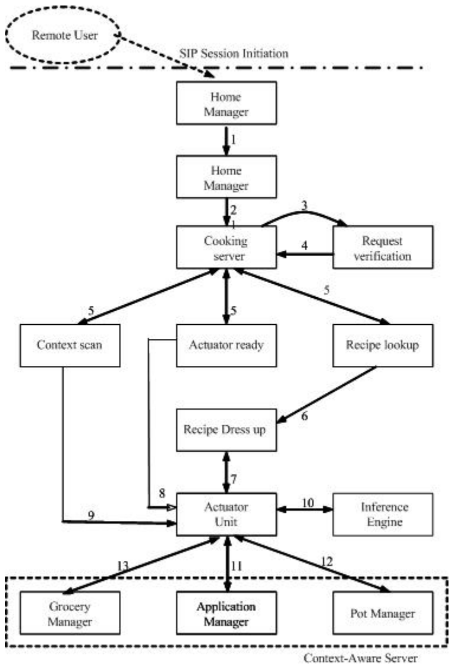


Fig. 2: Control flow architecture

The cooking application runs on top of the entire infrastructure laid and it development scheme is developed by service provider, we call this as a Recipe -file in this study, which is responsible for its advertisement too. There are some interfaces exposed by every service. The Recipe-file contains the interfaces needed for the target application development. Whenever this Recipe-file is downloaded and invoked at some service gateway level it checks all the services present and matches the interfaces required, gets registered for the use of the target application and after gathering all the services needed, the new application is advertised to the other components of the network e.g. remote managing gateway, neighboring residential gateways etc.

We present a ubiquitous cooking application that is developed through the propose scheme. The control flow architecture of the application in various time slots is shown in Fig. 2. It is shown how the control is passed to other components and at what pattern. The remote user connects to the home server and the control manipulation takes him to cooking server. The cooking server is the point of entrance for our scenario. At this moment the request verification is done for the determination of the exact specifications. The Actuator unit, Recipe-look up agents and Context scanning start working thereafter. The conversion of the recipe into the instruction format with reference to the existing Context is done at the Recipe-dress up manager. Thereafter, the visible preparation starts. The Context Server, Actuator Unit and Active Profile Manager becomes the active parts of the system. Jobs are assigned to different devices according to the assigned slots available per device per unit time.

We try to restrict the interaction according to the scenario of the application under consideration. It can be broadened (for dealing with the exceptions) but our purpose in this study is to keep the scenario simple.

Recipe manager: The recipe manager is a part of the service that contains the whole record of the recipe pool in XML format. The service provider is responsible for recipe formation. An XML parser parses the whole Recipe-file and inference engine allots the jobs to individual devices by keeping the available slots of that device in context and the nature of job. The benefit of keeping the recipe in the XML format is to obtain ease in transportation of the data when and where it is needed. It can be directly fed to the machine instruction translator unit or recipe interpreter. It saves time and provides efficiency to the system the remote user requests the dish the interface is shown in Fig. 6. The dish name can be in the form of a pre-filled items list that the cooking server provides to the remote user. In Fig. 6 the recipe lookup button checks the availability of the dish and its contents. If the user wants to specify the recipe or the contents destination through the insurance of some third party, he can do so on this interface. The jobs are divided into time slots and during those time slots the system activates many threads and monitors the performance of each device. The manager can download updates and new recipes which can also be located on user request recipes. After all the inputs are prepared, the system gets busy for the specified time to make the dish as requested. The remote user is connected to the server using Session Initiation Protocol (SIP)[8]. For connectivity, each device is SIP enabled. According to the hierarchy, the remote user gets connected to the home server that has SIP agent and also the Context Server. All the remote hosts get connected to the Context Server via the SIP service. For session initiation one device becomes a client and the other one becomes a SIP server.

Figure 3 shows the overall scenario of a remote user and his connectivity to the home server. The remote user pages the proxy server by requesting for a session establishment. When established, it requests the status service to search for availability of the cooking server. If the cooking server is busy, the user gets (a) ‘System Busy’ response otherwise he gets connected and interacts with the context aware server as shown in Fig. 3. The remote user pages the proxy server that is present at the home gateway and gets connected after Trying and Ringing. Before connecting the user, the proxy server INVITES the SIP service in the service bundle of the Context Aware Server that is allocated inside the kitchen server. After this initial handshaking, the remote user gets connected to the

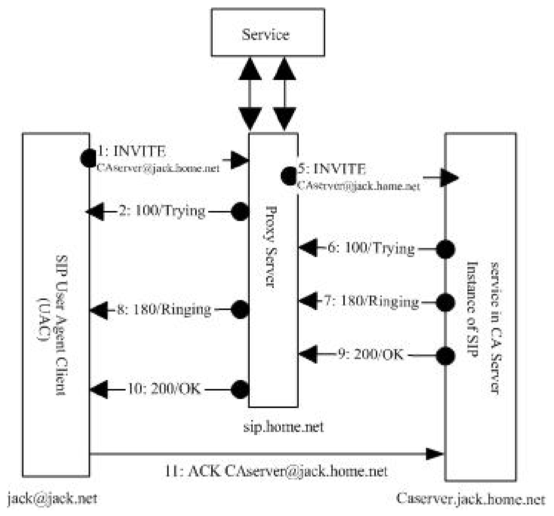


Fig. 3: SIP for remote user connectivity

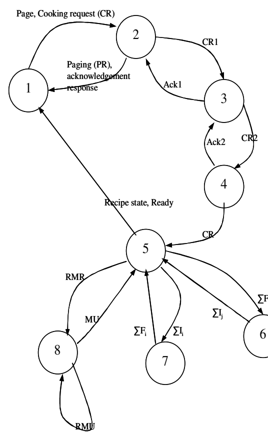
kitchen server and interacts with the application. The prototype for the user connectivity is a service that has been designed to run on an iPAQ® using the Jeode JVM™, although it should work with any other PersonalJava™ compliant JVM. This service waits for an iPAQ-alert message to be posted to the event heap and then displays the text attribute of the message in a popup window. If further events are posted before the popup window has been closed, these will be displayed in sequence after the window has been closed.

Actuator unit: The piece wise actuator unit is considered feasible for the kind of a scenario we have. The support of hardware automation is needed in various time slots and various jobs can be assigned to one piece of hardware in two adjacent time slots. We propose to divide the actuator unit into two. One is the software decision making and automatic low level code generation and the second one includes the communication with the hardware infrastructure.

Figure 5 is model architecture of a ubiquitous kitchen. The shelves on the right side contain the grocery chambers. On the left side the pots chambers are shown and in the middle of that the stove, a smoke collector and a place for robotic arm are shown. A conveyer belt is shown at the left bottom side where the pot is placed and then is moved to the table along with the cooked food in it. The collector tray is placed between the chambers and the stove. That collector tray collects the grocery items and pots from their respective chambers and puts them into either pot on stove or for some further process i.e. chopping, mixing, blending etc. then injects them into the main process. We have a robotic arm present over the collector tray. The purpose of the arm is to place the pots in the right spots i.e. on the stove, moving it away from stove and placing a new one on it etc. The entire infrastructure includes the actuator unit.

It may not be a good idea to have the devices attached to the actuator unit, working in a liner feedback control model. A quasi linear system that can make those devices run in various time slots, under varying conditions is needed. We propose the piece-wise control feedback system. It can be said that we can achieve the desired amount of precision required from the Actuator unit to carry out the mechanical tasks[9]. A high level of precision is required so that higher level of Quality of Service is maintained. Lucibello[9] discussed piecewise control systems with learning capability. Rantzer and Johnsen[10] proposed the piecewise linear quadratic optimal control. Their simulation results have[9,10] shown that the error factors reduces considerably and consequently as more control over the system is attained and that is our objective using them in our system. In the following passage we consider a control system following piece- wise feedback control model.

Using the scheme Lucibello proposed[9], the quasi-linear systems through an appropriate state extension and the use of an iterative learning algorithm make it is possible to track a piece-wise continuous output trajectory minimizing a suitable norm of the error. So we conclude that the desired amount of accuracy can be obtained provided that we gather the context of the environment and provide it to the context server. The Context server makes sure that the robotic arm and other parts of the actuator unit are precise enough to carry out the instructions. The instructions are distributed piecewise and feedback is gathered for the future improvements.



1. User	RMR	1- $2\sum_{i=1}^{i=\infty}\sum_{j=1}^{j=\infty} F_{(i,j)}$
2. Remote Host	- Recipe Messaging	(Feedback)
3. Home Manager	Request	2- $2\sum_{i=1}^{i=\infty}\sum_{j=1}^{j=\infty} I(i,j)$
4. Home server	MR	(Instructions)
5. Cooking server	- Managed Recipe	3- $2\sum_{i=1}^{i=\infty}\sum_{j=1}^{j=\infty} U_{(i,j)}$
6. Actuator	RMu	(Uncertainty)
7. CA Manager	- Recipe Makeup	
8. Recipe Manager	F- Feedback, Instruction	

Fig. 4: Automata model for integrated application

Recipe dress-up manager: After the recipe is stored in the recipe knowledge base, there is a need to translate it into executable instructions so that the actuator unit may start working the preparation process of the food. The context of the environment is very important at this stage. Knowledge of the resources is related to how many, what kinds of devices and pans we have and the current status of the groceries. The recipe dress up manager is the entity that translates the recipe in XML format into machine instructions. It is done when we have the recipe template generated by recipe dress up manager and then the planning unit of the dress up manager allocates the sub tasks against the time slots available with each device. First it should be determined whether (or not) all the hardware is ready to start the job in which we rely in the context manager.

The context manager keeps track of every item in the kitchen (functionality shown in Fig. 4). The (1) shows the time taken for feedback cycles, (2) shows the instructions allocations and (3) shows the amount of uncertainty involved in the whole process. Whenever there is some need to start a new job, the context manager is queried about the ingredients and their resources required. The planning module formulates the roadmap table and reserves the resources for that job.

Table 1: Job allocation to devices in time slots

Device ID	Time Slot	Instruction	
C001455D	001589GHJ	DGHJ003	*C001455D*; Consumer
C001456D	001589GHJ	DGHJ013	001455 Device
C001457D	001589GHJ	DGHJ005	*001589GHJ*; 001589th
C001458D	001589GHJ	DGHJ001	Second in General
C001459D	001589GHJ	DGHJ007	Hour for Jack (Jack
C001455D	001590GHJ	DGHJ004	is the name of the job)
C001456D	001590GHJ	DGHJ014	*DGHJ003*; Device
C001457D	001590GHJ	DGHJ006	General Hardware
			Number Jack 003

All the communication to the devices is done via context manager. After requesting adequate resources from context manager, the dress-up manager prepares the job allocation tables as shown in Table 1. In those tables the jobs of all the system are assigned with reference to each time slot. Every time slot contains information of each and every device in contact. There can be two states of each device, e.g. 1- device in context and 2- device out of context. Every attached device is sent the instructions in each time slot. The sample instructions are shown in Table 1.

The tags shown in the Table 1 reflects a scheme that we use to allot the ID's to each activity that is carried out as a result of the instruction generated as a result of recipe dress up process. Many micro instructions of this type are generated. Many of those instructions are assumed to be related with a device containing a specific context e.g. a device that can check the temperature of the room. Now for that device there is a queue of instructions in different time slots.

The allocation of these tasks to that particular temperature checking device depends upon the free time available with that device and of course keeping the mobility factor into consideration also. During the job in process time there is a provisioning agent, the provisioning agent is the one that is responsible for determining the future availability of the time slot in the context of that device. A monitoring service is provided so that the instructions are in a queue and no micro instruction is missed, In case of execution failure or device leaving the area, the same responsibility is assigned to some other device for the same task. Here we assume that there is all type of devices present in the kitchen, at least those who are mandatory in the normal kitchen environment.

Context aware server: Network resource availability varies with time not only due to dynamic demand from the users but also dynamic channel conditions. A residential gateway needs to dynamically assess the availability of system resources and allocate them accordingly to meet QoS requirements of supported applications[11]. The context aware server is a mini gateway to access the devices in the kitchen environment. It works the other way around too by devices reporting their latest context. Another responsibility of the Context server is that they participate in the recipe dress-up process.

When the recipe is prepared, keeping the context of all the devices ready to participate in a new process and user preferences, the context server presents a copy of the latest context of devices so that the whole process can be accomplished with maximum efficiency. The message passing nature of various modules of the application and tight time slots marking are mandatory for high level of Quality of Service (QoS). The presence of an entity handling the context is important in this scenario. Since the probability is very high that the components installed in the kitchen come from different vendors. The autonomic self configuration plays a major role in such environments. After the device discovery process its resources are added in the resource pool that is in the form of knowledge base. For this the UPnP™ support is important and self configuration demands the ubiquitous driver access service that may enable a system to be detected and attached and that can make device work seamlessly. A device attached to the network is desirable to contain the following characteristics.

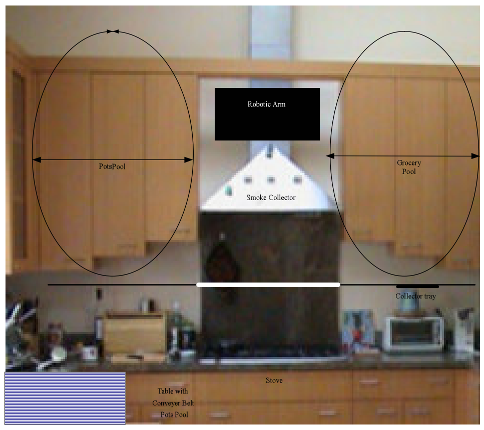


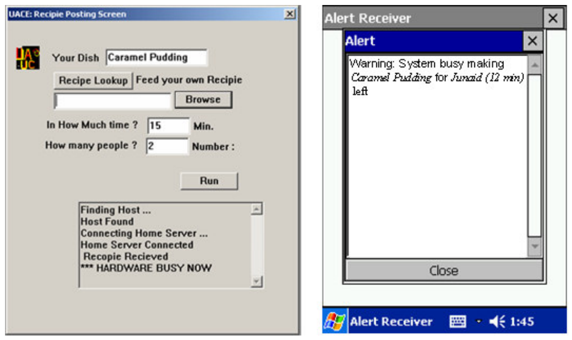
Fig. 5: A model ubiquitous kitchen environment

Many of these depend upon the vendor but the more the probability of presence of these features into the devices is, the more efficient the network will be.

* Discovery of other devices on the network. The number of hand-over and take-over happening
* Discovery of the new interfaces and search of their support present for the network on the vendor site or at some other location.
* Auto-assignment of Ubiquitous ID to new devices.
* Profile management of each device coming in and handing over the profile to the new gateway if device leaving the network.
* Self-provisioning and providing of data to other devices to set their state based on addition of new devices to the network.

Based on these requirements we explored various alternatives for discovery and initialization. While JINI™ is an attractive proposition, the UPnP™ suite of protocols appears to be more intuitive to use.

It is obvious that suitable "Device Control Protocols" (DCP's) would have to be defined for the various components of the application. This would allow various vendors to communicate with each other while still allowing for vendor differentiation and value addition. These features (addressability and discovery) fit to our scenario to assign Ubiquitous ID to the devices and self configuration features in the network. Once discovery has been performed, UPnP™ allows for the exchange of device description in an XML based format which includes URL's for device control. This mechanism can be used by the various devices in ubiquitous kitchen to describe them and obtain a description of other devices on the network. This would allow the newly added device to initialize itself with network specific parameters. In addition to a suitable discovery and initialization protocol the controller needs to access interfaces on the devices to accomplish a task.


UACE: Recipie Posting Screen

Your Dish: Caramel Pudding

Recipe Lookup | Feed your own Recipie | Browse

In How Much time ? 15 Min.

How many people ? 2 Number :

Run

```
Finding Host ...
Host Found
Connecting Home Server ...
Home Server Connected
 Recopie Recieved
*** HARDWARE BUSY NOW
```

Alert Receiver

Alert

Warning: System busy making *Caramel Pudding* for *Junaid (12 min)* left

Close

Alert Receiver 1:45

Fig. 6: Recipe request and lookup interface

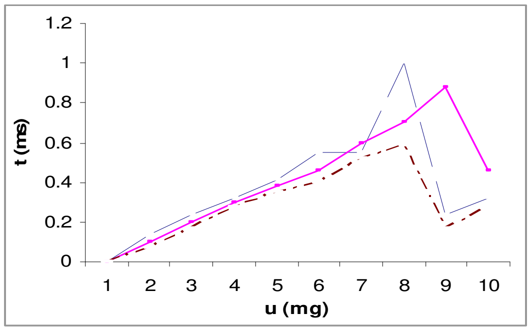


Fig. 7: The simulation results of grocery item (μ) verses time slots (t). (The performance curve shows linearity showing uniform flow out from chamber after continuous refills). The variation in the simulation shows the different value of ψ=1/2q and θ

This can be completed by passing an appropriate message to the device. A suitable abstraction for this is the device control interfaces as described by UPnP™. This allows for the device interfaces being invoked by sending SOAP messages over the network. In the absence of a message passing and event notification mechanism the device management and synchronization of the whole process will be very difficult or nearly impossible. That is why we need some event notification message passing architecture so that all the devices that are involved in the active operation could be synchronized

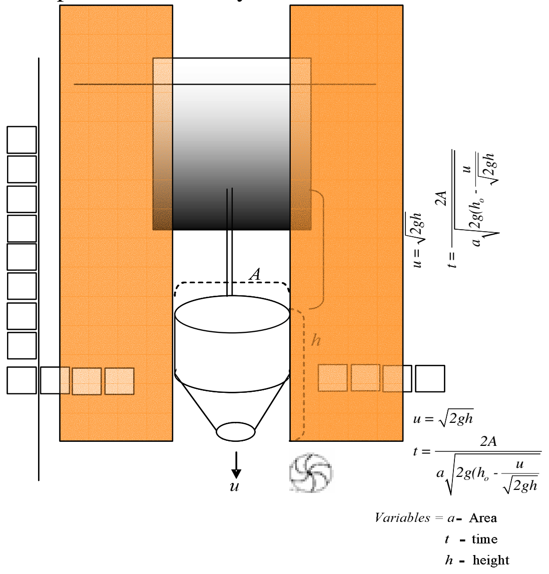

A
h
u
$u = \sqrt{2gh}$
$t = \frac{2A}{a\sqrt{2g(h_o - \frac{u}{\sqrt{2gh}}}}$
$u = \sqrt{2gh}$
$t = \frac{2A}{a\sqrt{2g(h_o - \frac{u}{\sqrt{2gh}}}}$
Variables = a - Area
t - time
h - height

Fig. 8: Internal architecture of grocery chamber

and in 'ready to perform' state. This lowers the delay in the process and increases the efficiency.

Grocery manager: A lot of research has been done and going on concerning this issue of grocery management[12,13]. For managing the grocery items we have a grocery managing mechanism. The grocery is stored in the chambers shown on the right side of Fig. 5. A chamber has some area *A* and height *h*. The area of the aperture is *a* and the amount of item coming out is *u*. The structure of each chamber is shown in Fig. 8. Now according to Torricelli model we can predict the amount of grocery item coming out from the chamber. Torricelli model is based on two physical concepts.

First is Bernoulli relationship between pressure *p*, density and speed for the matter along a stream line

$$\Delta p = \left(\frac{1}{2}\sigma_u^2\right). \qquad (1)$$

The second is the relationship between changes in pressure over the height column and gravity (acceleration *g*), density and the height *h* of the column

$$\Delta p = (pgh). \qquad (2)$$

Combining (1) and (2) gives a relationship between height of the column above the hole and the speed at which it spews forth, *u* is equal to square root of $2gh$. The volume of grocery item lost from the bucket must equal the flux through the bucket's hole (with area *a*) and using the fact that (for a chamber with regular sides and constant cross section in height) the relationship becomes

$$ua = \left(\frac{dV}{dt}\right) = \left(\frac{d[Ah + V_o]}{dt}\right) = A\frac{dh}{dt}. \qquad (3)$$

From the relationships mentioned, the following formulae help in estimating and thus controlling the amount of grocery item flowing in and out of the grocery chamber.

When some quantity of grocery item is drown out from the chamber, Δu is the change in quantity and u is the amount coming out. When some amount of grocery item is needed, instructions are sent to the aperture. The time *t* in which the aperture was open has a direct relationship with the quantity drown out from the chamber, we can say that *t* is the function of change in amount of the grocery coming out. So

$$h = \sqrt{h_o} - \frac{a\sqrt{2g}}{2A}t^{2}, \tag{4}$$

and the aggregate height *h* of *n* number of chambers is,

$$\sum_{i=1}^{i=n} h_i = \sqrt{h_{oi}} - \frac{a_i\sqrt{2g}}{2A_i}t_i^{2}. \tag{5}$$

It gives the relation between the height of the grocery item present in the chamber and the rate of change in the height of the column h_o and

$$t_{empty} = \left(\frac{2A\sqrt{h_o}}{a\sqrt{2g}}\right) = \left(\frac{A\sqrt{2h_o}}{a\sqrt{g}}\right), \tag{6}$$

$$\sum_{j=1}^{j=m} t_{empty_m} = \left(\frac{2A_m\sqrt{h_{om}}}{a_m\sqrt{2g}}\right) = \left(\frac{A_m\sqrt{2h_{om}}}{a_m\sqrt{g}}\right), \tag{7}$$

Where, t_{empty} is the amount of time required for a grocery chamber with area *A*, rate of change of grocery column h_o, *a* as the area of the aperture and *g* as gravitational acceleration. In time t' *and* $\sum_{j=1}^{j=m} t_{empty_m}$ is the time taken by *m* number of chambers to get empty.

$$u = \left[\sqrt{2g}\left(\sqrt{h_o} - \frac{a\sqrt{2g}}{2A}t'\right)\right], \tag{8}$$

$$\sum_{k=1}^{k=\infty} u_k = [\sqrt{2g}(\sqrt{h_o} - \frac{a_k\sqrt{2g}}{2A_k}t_o')] \tag{9}$$

Where, *u* is the amount of grocery item coming out from the chamber and $\sum_{k=1}^{k=\infty} u_k$ is the aggregate amount of grocery
items coming out from *o* number of chambers in time *t* from chambers of height *h,* area *A and* gravitational acceleration *g.* The following is the expression for two parallel quasi-linear control systems. We show the inter process flow error control and mechanical error expressions along with the two indigenous process expressions (derived from eq. 9).

$$\sum_{j=1}^{j=\infty}\sum_{k=1}^{k=\infty} u_{kj} = [\sqrt{2g}(\sqrt{h_o} - \frac{a_k\sqrt{2g}}{2A_k}t_k')] +$$
$$+\psi_k \frac{1}{2}q_j + \theta + [2g(h_o - \frac{a_k\sqrt{2g}}{2A_j}t_j')]$$

CONCLUSION

In this study we present a scheme for application development in ubiquitous environments. We discuss the scenario with an example. The ubiquitous services integrate to form new applications and those applications interact directly with users. Previously it was considered that the services in the service oriented environments will interact with the user directly[6,11]. However we propose that the applications developed through service integration serves the preferences of the users in a better way by keeping the preferences in mind and at the same time keep on monitoring the resources available.

ACKNOWLEDGEMENT

This research is supported by the Ubiquitous Computing and Network Project, the Ministry of Information and Communication (MIC) 21st Century Frontier R&D Program in Korea.

REFERENCES

1. Junaid Ahsen Ali Chowdary, Won-Sik Yoon, Jai-Hoon Kim and We-Duke Cho, 2004. U- Kitchen: application scenario. Proc. WSTFEUS, pp: 169-171.
2. McDonald, D.W., 2003. Ubiquitous recommendation systems. Computer, 36: 111-112.
3. Hedberg, S.R., 2000. After desktop computing: A progress report on smart environments research. Intelligent Systems, IEEE. 15: 7-9.
4. Open Service Gateway Initiative, www.osgi.org <http://www.osgi.org>.
5. Takemoto, M., H. Sunaga, K. Tanaka, H. Matsumura and E. Shinohara, 2002. The ubiquitous service-oriented network (USON) - An approach for a ubiquitous world based on P2P technology. Proc. P2P 2002, pp: 17-21.
6. Yu, M., A. Taleb-Bendiab, D. Reilly and Omar, 2003. Ubiquitous service interoperation through polyarchical middleware. Proc. IEEE/WIC' 03, 2003, pp: 662-665.
7. Ricci, A. and A. Omicini, 2003. Supporting coordination in open computational systems with TuCSon. Proc. WET ICE 2003 pp: 365-370.
8. Stan Moyer, Dave Marples and Simon Tsang, 2001. A protocol for wide-area secure network appliance communication. Commun. Mag., IEEE., 39: 52-59.
9. Lucibello, P., 1998. A learning algorithm for improved hybrid force control of robot arms. Systems, Man and Cybernetics, Part A, IEEE Trans., 28: 241-244.
10. Rantzer, A. and M. Johansson, 2000. Piecewise linear quadratic optimal control. Automatic Control, IEEE Trans., 45: 629-637.
11. Bansal, D., J.Q. Bao and W.C. Lee, 2003. QoS-enabled residential gateway architecture. Commun. Mag., IEEE., 41: 83-89.
12. Kourouthanassis, P., L. Koukara, C. Lazaris and K. Thiveos, 2001. Last-mile supply chain management: MyGROCER Innovative Business and Technology Framework. Proc. 17th Intl. Logistics Cong. Logistics' 01, pp: 264-270.
13. Asthana, A., M. Cravatts and P. Krzyzanowski, 1994. An indoor wireless system for personalized shopping assistance. Proc. Mobile Computing Systems and Applications' 94, pp: 69-79.