International Journal of Computational Intelligence Systems, Vol. 6, No. 4 (July, 2013), 778-795

Model Reference Adaptive Control in Fuzzy-Based Context-Aware Middleware

Ronnie Cheung[*]
University of South Australia
Australia

Hassan B. Kazemian
London Metropolitan University
United Kingdom

Jiannong Cao
Hong Kong Polytechnic University
Hong Kong

Received 14 March 2011

Accepted 19 February 2013

Abstract

Owing to the dynamic characteristics of mobile environments, a mobile application needs to adapt to changing contexts to improve performance and resource utilization. We have developed an adaptive middleware infrastructure that simultaneously satisfies the individual needs of applications while maintaining overall system performance. A Model Reference Adaptive Control mechanism has been implemented in the middleware using control theory and fuzzy-based techniques. With reference to the model reference adaptive control theory, we also present a Self-Adaptive Fuzzy-based Service Adaptation Model (SA-FSAM) by taking historical adaptation information into account, and utilizing a closed-loop control mechanism to fine-tune adaptation decisions. The SA-FSAM and a conventional threshold-based linear-control model have been evaluated using a campus assistant mobile application. With the introduction of self-adaptive elements in the control model, the SA-FSAM shows significant improvements in service adaptation decisions.

Keywords: Fuzzy logic; control theory; service adaptation

1. Introduction

For mobile applications to operate efficiently in mobile environments, they should be able to sense and evaluate the current operating context and then adapt their services to the context in order to optimize system performance. A generic context-aware middleware architecture—adaptive middleware infrastructure (AMI)—has been developed, which functions between the mobile applications and the operating system. The middleware approach simultaneously satisfies the individual needs of applications, while maintaining the overall operating system performance. In order to handle the vast amount of contextual information and numerous combinations of adaptive system services, fuzzy logic is utilized to cope with the uncertainties in adaptation control. The use of fuzzy logic serves two purposes: in addition to adaptation control, fuzzy logic is also applied on context reasoning and representing various low-level detector-based contexts and

[*] Corresponding Author: ccheung@acm.org

composite abstract contexts, which simplifies input factors for adaptation decisions. By incorporating control engineering techniques, the middleware also employs a model reference adaptive control mechanism to handle the dynamic aspects of adaptive control.

A fuzzy-based service adaptation model (FSAM) has been designed as the core inference engine in the middleware infrastructure. Our focus is on the service adaptation model, which compares context situations with the desired response from a model reference depository. Fuzzy linguistic variables were used to define context situations and the policies for adopting suitable services in the context-aware middleware. Fitness functions were designed to calculate the fitness degree for each service policy based on the distance between the service policy and the current context situation. The decision for service adaptation was made by selecting the policy with the best fitness degree with reference to a model reference depository. The experimental results have proved the effectiveness of the FSAM in our previous publications[12]. However, in-depth experimental results also indicate certain limitations of the FSAM. When it comes to large-granular fluctuations, the FSAM cannot handle the adaptation decisions effectively with drastic changes in context situations. Control-theory and fuzzy-based approaches are combined to develop the Self-Adaptive Fuzzy-based Service Adaptation Model (SA-FSAM) in the middleware. By employing model reference adaptive control techniques in control engineering, the SA-FSAM monitors service adaptation in a real-time manner. The self-adaptive fuzzy-based approach is described in the following sections.

2. Background

2.1. *Context-Aware Middleware*

According to Dey[14,18], context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including location, time, activities, and the preferences of each entity. A system is context-aware if it uses the context information to provide relevant information and/or services to the user, where relevancy depends on the user's task. Inferred from the analogy between context-awareness and human consciousness, context-aware applications can be characterized by a variety of detectors that monitor multiple contexts, a service adaptation engine, and actuators for executing adaptation decisions. There are two critical issues in the development of adaptive context-aware applications. First, it would be inefficient for individual applications to maintain the required contexts independently, and it would be infeasible for application developers to provide a detailed description of every possible potential context. This extra layer of contextual description can be implemented by a middleware approach. Second, it is challenging for application developers to provide each application with its own adaptation mechanism down to the system level. Again, a middleware approach provides programmable system services to the developers. With the context-aware middleware infrastructure AMI[10,11], the balance between the mobile applications and the overall operating system performance is taken into account.

2.2. *Fuzzy-Based Approach*

Fuzzy theory[36] has been used in industrial control and automation systems for a long time. Fuzzy logic controllers are very effective for complicated and imprecise processes, for which either no mathematical model exists or the mathematical model is severely nonlinear. Fuzzy logic controllers can attain performance close to that of human experts under such poorly defined environments[32]. Bettini et al.[2] identified two important aspects that should be addressed in context representation and reasoning: high-level abstraction of contexts, and uncertainty in context information. Fuzzy logic provides rich semantic features to provide high-level abstraction of context information. In order to handle the vagueness in contextual information, a fuzzy-based approach to model service adaptation[10,11] is developed in this research. Fuzzy logic is used to represent contextual information and formalize service adaptation mechanisms.

2.3. *Control Theories and Fuzzy-Based Approaches*

Due to the highly dynamic characteristics of mobile environments, the variations in the ever-changing contexts are complicated. These variations are

categorized into small-granular oscillations and large-granular fluctuations. Both categories of variations introduce undesirable effects in service adaptation. They also cause negative user experience and inefficiency in resource utilization. The use of fuzzy-based approaches can significantly alleviate these effects in the small-granular oscillations in contexts. The fuzzy-based inference engine can generate smooth service adaptation by reducing unnecessary adaptation behavior between neighboring quality of service (QoS) levels. However, for large-granular fluctuations in contexts, it is possible to experience drastic service adaptations across different QoS levels. To control drastic adaptations from large-granular fluctuations, the stability of the system and compliance with application preference need to be improved. It is necessary to provide mechanisms to define, detect, and reason with changes in context information resulting from the large-granular fluctuations. Control-theoretic techniques provide a solution to deal with adaptations resulting from drastic changes in context situations.

Control-theoretic approaches have been widely applied to computing systems in recent years. The advantage of these approaches is the improvement of system performance with optimal resource exploitation. With control-theoretic mechanisms, the performance of large-scale and loose-coupled computing systems can be enhanced using a controlled-based approach. Self-adaptive systems operate with the guidance of a central controller to assess their own behaviors in the surroundings to provide service adaptations. The fundamental principle behind self-adaptivity in control-theoretic approaches is to monitor the system states continuously to adjust service adaptation. The operations are based on closed-loop control and open-loop control mechanisms. As shown in Fig. 1, with a closed-loop control mechanism, the output signal is used to provide feedback to the system. The controller implements an algorithm to decide upon a suitable correction u to drive y_p closer to u_p using process specific actuators. The feedback is used to reduce the effects of uncertainty that appear in the form of noise in the contextual information used for designing the controller. The closed-loop control mechanism outperforms open-loop control operations, with a tradeoff between complexity and stability.

The major challenge in building context-aware adaptive systems is to develop the capability of the system to adjust its behavior in response to the environment in the form of self-adaptation. In the implementation, self-adaptivity refers the ability of the SA-FSAM middleware to adapt autonomously (i.e. with minimal interference) in response to contextual variations to improve QoS. A typical application might involve the need to improve response time when the available bandwidth is low and a dynamic relocation of the system sources is required in the adaptation process (for example, to allocate CPU resources to encrypt the data before transmission or reduce the level of details in a Web page). However, frequently oscillations in context variations appeared as noise or disturbances to the fuzzy controller, which might cause adaptation decisions to fluctuate too frequently. Feedback loops provide the generic mechanism for self-adaptation in the SA-FSAM middleware. The historical records of adaptation decisions and context variations provide feedback information for achieving self-adaptivity in the SA-FSAM middleware. By considering the adjustable parameters in the model (including the size of large-granular fluctuation, the time window of past adaptation information, the damping effect of service adaptation, etc.), the SA-FSAM can be fine tuned on the basis of domain knowledge.

The remainder of this paper is organized as follows. Section 3 provides a discussion on the related work. In Section 4, the details of the middleware architecture are presented. In Section 5, the SA-FSAM framework is presented, including definitions, formulas, algorithms, and a sample application. Section 6 illustrates the experimental results. Conclusions and future work are discussed in Section 7.

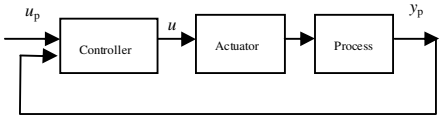

Fig. 1. A Feedback Loop

3. Related Work

Various researches on context-aware computing have been carried out in the past few years. In the Gaia

project[27], an infrastructure was built to provide context awareness and semantic interoperability. Physical spaces like rooms, homes, buildings, and airports are converted into a programmable computing system called Smart Spaces. Regarding research on autonomous and multidimensional contexts, significant progress has been made on fusion services[6] that extract and infer sensor-based contextual information using Bayesian networks. To deal with heterogeneous contexts, an adaptive online mechanism was presented by Kim and Varshney[21] for QoS-sensitive applications.

Winograd[34] provided different architectures for handling contexts. Johanson and Fox[20] proposed the use of a centralized Event Heap for handling contexts. Shafer, Brumitt and Cadiz[30] performed experiments on multimodal interactions in context-aware environments. A reconfigurable context-sensitive middleware was proposed by Yau *et al.*[35] to provide adaptive object containers for run-time context data acquisition, monitoring, and detection. The WebPADS[13] project was developed by Chuang, Chan, Cao and Cheung to provide actively deployable and dynamically reconfigurable service chaining for adapting the dynamic changing contexts. A context-aware middleware architecture with the FSAM was introduced by Cheung, Yao, Cao and Chan[12].

Control theory, originally developed for handling industrial control and automation, has been applied to model physical systems. Control-theoretic approaches have been applied to computing systems with prevailing large-scale software systems and wide-area communication networks. The research community in software engineering domains and computing environments has established feedback loops as the core design elements in large-scale software development of adaptive systems[26]. Tu *et al.*[33] provided an implementation of load shedding by applying feedback control in a real-time data stream database system to balance the database workload. Robertson[29] proposed a closed-loop control mechanism for admission control in web server systems, and the experiments proved the stableness of the model. The major reason for using feedback was to reduce the effects of uncertainty in Internet load, which was difficult to model without feedback mechanisms.

Fuzzy logic has been widely used since fuzzy set theory was first introduced by Zadeh[36]. A fuzzy logic controller[1,17,19] comprises of a fuzzification interface, a knowledge base that consists of a rule base and a data base, a decision-making logic unit, and a defuzzification interface. In mobile computing, researchers have been focusing on the use of fuzzy logic for mobile location management[7] and power control[8,31] in wireless networks. However, these systems were developed for adaptations relating to a single aspect. Recently, researchers have shown increasing interest on fuzzy controlled QoS adaptation[16,25,28] to fulfill application requirements in terms of more generic QoS specifications. In the CARISMA system, the context-aware middleware compared the application profile and the current context to evaluate which policy should be adopted[4,5]. In the context-aware system implemented by Kwan *et al.*[23], the functionality of a service code module was adapted on the basis of the estimated resource usage.

There has been a significant amount of research work on fuzzy-based adaptation. Ghinea *et al.*[16] developed a fuzzy logic-based descriptions of QoS specifications, as well as applying fuzzy logic to specifications based on user perceptions, and fuzzy programming was used to obtain a user-oriented ordering of the QoS parameters. Koliver *et al.*[22] proposed fuzzy rule-based techniques for assisting QoS adaptations in distributed multimedia systems. To deal with the uncertainty in variations in context environments, researchers have been building self-adaptive systems that are capable of dealing with continuously changing environments and emerging requirements by introducing feedback loop mechanisms in the design[9]. Li and Nahrstedt[24] introduced a fuzzy control model and a task control model to enhance the effectiveness of QoS adaptation decisions for a distributed visual tracking application. Applications were modeled as a serious of tasks assisted by a feedback loop. A middleware was used to implement the components to control adaptations so that various adaptive transient properties relating to stability and agility were addressed formally. These approaches were usually developed for specific problem domains (e.g., multimedia applications) while our SA-FSAM middleware emphasizes on a generic adaptation model that can be applied to different applications.

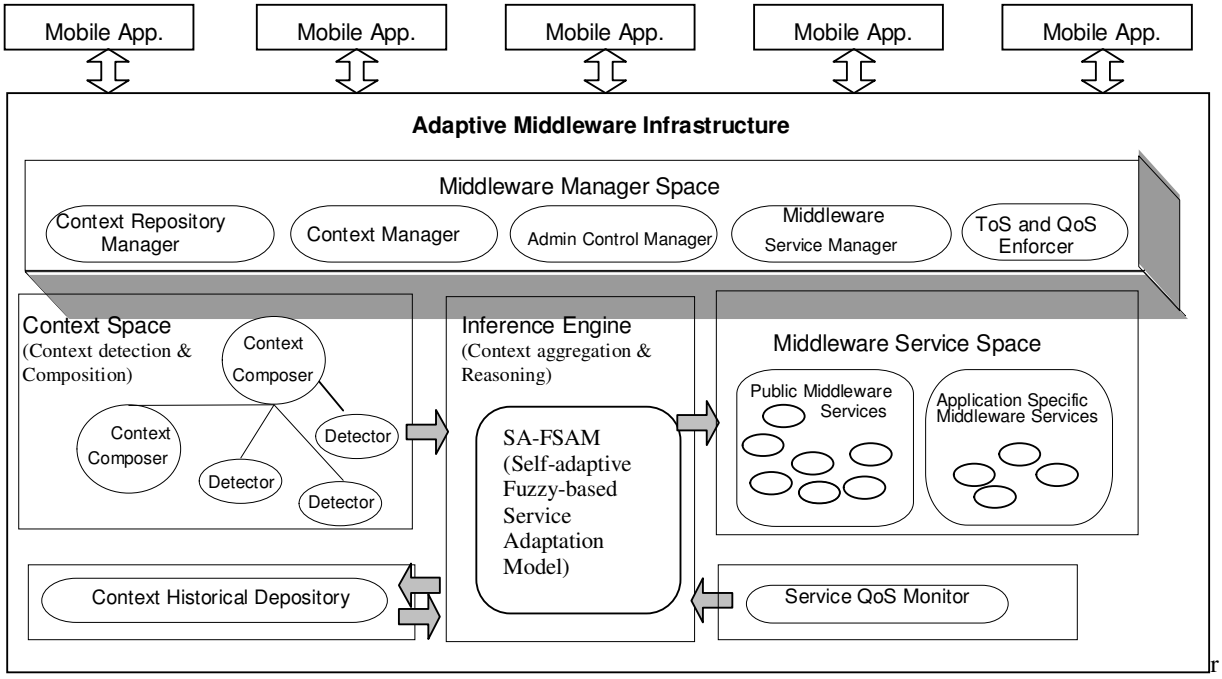

Fig. 2. Adaptive Middleware Infrastructure with Self-Adaptive Fuzzy-based Service Adaptation Model

4. Adaptive Middleware Infrastructure

An adaptive middleware infrastructure (AMI) is developed to facilitate generic applications to exercise context-awareness[10,11]. It aims at integrating all the relevant features by utilizing a unified framework to facilitate the development of context-aware computing. The advantage of introducing adaptation mechanisms into the middleware layer is that the middleware has both the knowledge of individual mobile applications and the operating environment. All the relevant features that AMI integrates include context detection, context composition, context reasoning, middleware service delivery, Type of Service (ToS) and Quality of Service (QoS) enforcement. As shown in Fig. 2, the features are embodied in the four major modules in the proposed system.

The first module is the *Context Space*, which contains Context Detectors objects and Fuzzy Context Composers that reason and compose low level contexts into higher level representations. Examples of context detectors include wrappers for OS events, which could also be directly communicating with the device drivers. At a higher level, context detectors could also be detecting an application's communication and computation activity. Fuzzy context composer gather low level information, marshalling the dynamic and uncertainty of the mobile environment, and describe the current context in a more generic and coarse form. For example, a fuzzy context composer could be monitoring all the network related detectors, and determine the quality of network connectivity – good, average, poor and no connectivity. High level contextual information is also useful for adaptation with mobile applications.

The second module is the *Middleware Service Space*. This is the execution environment for both Public Middleware Services (one set of services that are shared among all mobile application) and Application Specific Middleware Services (each mobile application has its own set of services). These adaptive middleware services are categorized by their service nature into either public or application specific aspects. For example, a Web caching and prefetching middleware service is categorized as public since the cache pool benefits from the "economy of scale", and the operations involved are generic among different mobile applications; a media transcoding service is categorized

as application specific, since different applications have different data semantics that require different settings for the media transcoding service to work properly.

The third module is the *Fuzzy Adaptation Engine* (SA-FSAM), which adjusts the middleware services according to the current context and the mobile application's specific ToS and QoS requirements. In order to control the middleware services, the adaptation engine makes use of the programmable properties of the middleware services. To adjust the controllable parameters for the middleware services, the adaptation engine calls the corresponding adjustment functions. The middleware services are responsible for exporting the required interface for adaptation control, and to adjust the internal logics to follow the adjustments specified by the adaptation engine.

The fourth module is the *Middleware Manager Space*. It contains five system components, which coordinates all operations within the mobile middleware. The *Administration Control Manager* manages the admission of mobile applications that subscribe the services of the middleware. The duties include: authentication, type of service (ToS) and quality of service (QoS) negotiations, service subscription and un-subscription. The *Context Manager* controls the runtime environment for the context objects, including the low level context detectors and high level context composers. The *Context Repository Manager* maintains the records of all contextual information, based on the predicted future trend of the contexts. The *Middleware Service Manager* controls the execution environment for both public middleware services and application-specific ones. It coordinates with admission control, and controls the resources for newly subscribed services. The *ToS and QoS Enforcers* monitor the ToS and QoS levels for each mobile application with inputs from the Context Space.

5. Self-Adaptive Fuzzy-Based Service Adaptation Model

5.1. *From FSAM to SA-FSAM*

The inference engine is the core module in our middleware architecture. Our major focus is on the mechanisms provided by the inference engine to provide adaptation service for upper layer applications. The FSAM is developed as the inference engine in the adaptive middleware infrastructure[3,10]. The fuzzy-based approach demonstrates its effectiveness in service adaptation. Our approach handles the vagueness in contextual information as well as adaptations in multidimensional contexts. Linguistic variables and membership functions are employed to represent contexts. The vagueness of contexts becomes measurable, computable, and quantified. A fitness function is then developed to establish the relationship between multidimensional contexts and the policies for service adaptation. Each policy is associated with a kind of service for mobile applications. For example, a mobile meeting application may switch to text chat when the network bandwidth does not provide reasonable QoS for voice communications. This is implemented by using a fitness function to compute the overall fitness for the current context to fit the corresponding communication policies. By measuring and comparing the fitness degree between the current contexts and a predefined optimal contextual situation, the most suitable policy is adopted.

It is observed that although the adaptive middleware reacts to changes in the environment, it is not desirable to adapt too frequently to drastic changes in context situations. For example, for multimedia playback, being too sensitive to any change in contextual situations could deteriorate application performance. The frequent adjustments could be irritating to the user and waste precious computing resources. For this study, two different kinds of contextual variations are observed that cause undesirable service adaptation. By considering the size of granularity, the contextual variations are classified into two categories, which are presented in Fig. 3.

The first category is called small-granular oscillations and the other is described as large-granular fluctuations. Both kinds of contextual variations occur as jitters in a time series. The small-granular oscillations are microscopic but frequent variations of contexts, which could result from individual or composite contextual values changing back and forth around certain threshold values in a marginal area. For example, a spatially diversified wireless environment could lead to undesirable jitters in service adaptation switching between neighboring QoS levels. The large-granular fluctuations are drastic changes in contextual values

occurring in a fracture of time, and result in drastic service adaptation jumping across different QoS levels.

Both categories of variations can introduce undesirable effects on service adaptation, which also cause negative user experience and inefficiency in resource utilization. Therefore, service adaptations need to be insensitive at a certain level to contextual variations. Concerning the small-granular oscillations, fuzzy theory has been applied for solving this kind of problem. The FSAM was implemented to handle situations with small-granular oscillations context situations. However, concerning the large-granular fluctuations, the variations in contexts are so severe that an adaptation could jump from one QoS level to another QoS across multiple levels. Our current research combines control-theoretic approaches with our FSAM to solve the problem.

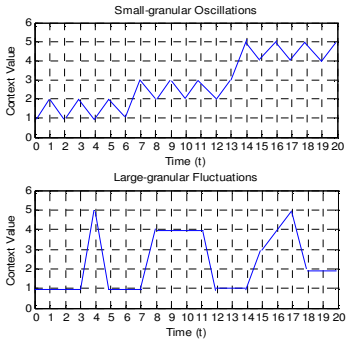

Fig. 3. Small-granular Oscillations and Large-Granular Fluctuations

5.2. *Model Reference Adaptive Control in Context-Aware Middleware*

Building self-adaptive software systems with predictable performance is a major engineering challenge. Adaptive control theory provides a viable solution by modifying the control law in the controller to cope with changes in the controlled process. Using control theory, a second control loop is included in the control model, which is installed on top of the main controller. By employing Model Reference Adaption Control (MRAC) mechanisms in control engineering, the second loop adjusts the controller's model by operating slowly to provide gradual adjustments to the controlling model, and operates with slower adjustments than the major feedback control loop (as shown in Fig. 4). For example, a major feedback loop in the Web server farm reacts rapidly to bursts of Internet load to manage the QoS. A second slow-reacting feedback loop may adjust the control law in the controller to accommodate the anomalies emerging over time.

The SA-FSAM controller implements the concepts in Model Reference Adaptive Control to provide service adaptation that matches requirements of the application with reference to the current context situation. The Model Reference Adaptive Control (MRAC) model was originally proposed for flight-control problems[15]. In the MRAC model, the adaptive algorithm compares the output of the process y_p with results from the control value u of the controller to the desired responses y_m from a standard reference model depository (as shown in Fig. 4), and then adjusts the controller model by setting controller parameters to improve the fit for the best policy in the future. In the SA-FSAM implementation, the standard model reference depository contains the standard quality requirements for different application policies. For example, the requirements for text chat and voice chat are different from video communications. The service adaptation controller switches between different service policies (such as text chat, voice chat, and video communications) to meet QoS requirements for mobile applications, depending on the match between the current context and the parameters stored in the model reference depository. In the above example, the model reference depository may contain the quality requirements for network bandwidth, CPU clock rate, network delay, and free RAM space corresponding to different service quality levels for text chat, voice chat, and video communications respectively.

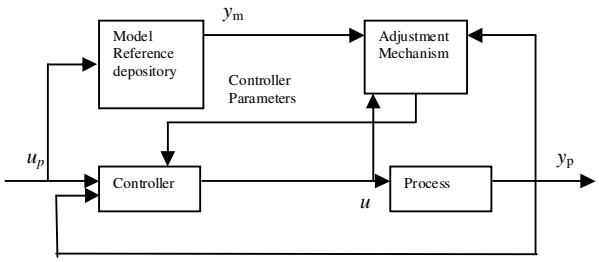

Fig. 4. Model Reference Adaptive Control (MRAC)

In the SA-FSAM framework, historical contextual information and service adaptation records are implemented as a main feedback loop to the main controller. This implementation provides a mechanism for handling frequent adaptations that appear as disturbances or noises resulting from frequent or drastic changes in contextual parameters. However, when such fluctuations jump across different QoS levels frequently, they produce undesirable experiences for user. Therefore, the fuzzy controller in the SA-FSAM middleware includes a second feedback loop in the model to provide adjustment mechanisms for the control rules in the controller. The feedback information (e.g. the number of service adaptations across two QoS levels in a time period) can also be used to improve the adjustment rules gradually to avoid undesirable adaptations due to large-granular context fluctuations.

5.3. *The SA-FSAM Framework*

The major contribution of the Self-Adaptive Fuzzy-based Service Adaptation Model (SA-FSAM) is that it introduces model reference adaptive control mechanisms with a fuzzy-based approach to fine tune adaptations decisions. It offers better application performance as well as optimized resource usage.

In the design of the SA-FSAM, self-adaptivity is achieved by taking current contexts as well as historical contextual information into consideration. The principle behind the self-adaptivity mechanism is assisted by historical information. The middleware acquires the capability of making a judgment to decide on whether or not the current variation of contexts is drastic, and implements appropriate adjustment mechanisms to determine the appropriate adaptation policy accordingly. For example, the greater is the gap between the current contexts and the past contexts, the stronger is the applied damping effect. By using this approach, self-adaptivity in the FSAM is implemented effectively. From another perspective, the inference engine becomes less susceptible to the variations in contexts by implementing the adjustment mechanisms in the MRAC model.

It is noticed that for most adaptive context-aware middleware implementations, the feedback mechanisms are either hidden or ignored[9]. The explicit design of feedback control mechanisms has an important impact on the middleware's design, architecture, and adaptive capabilities. By incorporating MRAC mechanisms and explicit control feedback loops in the design, the fuzzy controller can be implemented with the capability to maintain self-adaptivity and stableness in service adaptation. As shown in Fig. 5, when the output of the service adaptation engine meets certain conditions (e.g. number of service adaptations in a time period greater than a threshold value), the inference engine repeats the contextual reasoning by taking historical contextual information into account. A damping technique is applied in the adjustment mechanisms in the adaptation control engine to determine the degree of large-granular fluctuation alleviation.

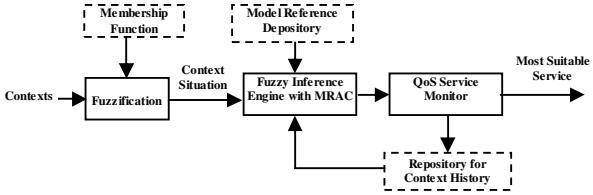

Fig. 5. The SA-FSAM Architecture

To describe the details of the Self-Adaptive Fuzzy-based Service Adaptation Model (SA-FSAM), examples of mobile services available on smart phones are used to demonstrate the effectiveness of service adaption. The mobile applications may provide different services s_1 and s_2, corresponding to email service and chat service for mobile users. Based on the FSAM model, the context-aware services running on mobile platforms are able to provide adaptations to meet different quality of service requirements. Taking the chat service as an example, it enables users to communicate through the mobile platform, by providing different quality of service levels relating to the chat service. In particular, due to the spatial and temporal variations of wireless communication and computing resources, the inference engine is able to react to the changing contexts and deliver the most suitable service policy. In order to maintain an acceptable user experience when the resource constraints become tight or even severe, the adaptation mechanisms are predefined by certain rules (or policies). For the chat application in the mobile platform, there are three adaptation policies

corresponding to three QoS levels: textChat, voiceChat, and videoChat.

Service adaptation has to take into consideration different context parameters that are reported by the middleware. The context parameters c_1, c_2, c_3, c_4 may include network bandwidth, CPU clock rate, network delay and memory usage. A fuzzy-based approach is being used to model different context situations at different time intervals. In order to model context situations, linguistic variables are used to model the values associated with the context parameters. Instead of using linear values to represent the context situation at time t for network bandwidth, CPU clock rate, network delay and memory usage, a fuzzy-based context representation is used. For example, at time t, when c_1= 10M, c_2=1000MHz, c_3=0.2ms, c_4=200KB, the context situation is represented by the degree of membership associated with four membership functions. Using the linguistic variables {lv_1="*high*", lv_2="*low*"}, four membership functions are defined to represent the degree of membership for context situations. A context situation at time t could be represented by a set of tuples that describes the degree of membership for the four membership functions corresponding to c_1, c_2, c_3 and c_4. Through the fuzzification process, the context situation at time t (when c_1= 10M, c_2=1000MHz, c_3=0.2ms, c_4=200KB) corresponds to the degree of membership for the membership functions: $\mu_{\text{Network\_maxRate high}}(10\text{M})$, $\mu_{\text{CPU\_clockRate high}}(1000)$, $\mu_{\text{Network\_delay low}}(0.2)$, $\mu_{\text{RAM\_freeSpace high}}(200)$. These membership functions are described in detail in section 6 of this paper. Using the FSAM model, $\mu_{\text{Network\_maxRate high}}(10\text{M})=0.5$ means that the values of Network_maxRate is high with a degree of membership of 0.5 when the Network_maxRate is 10Mbps.

Service adaptation is implemented by comparing the context situation with a standard reference for each policy. In the chat service example, the standard reference values refer to the most suitable context situations for the policies textChat, voiceChat, and videoChat respectively. For example, a high CPU clock rate is appropriate for the videoChat service. The standard reference value for the clock rate that is suitable for the videoChat service is determined according to user experience. In the implementation of the fuzzy-based service adaptation model, standard reference values for each policy are also fuzzified using membership functions. For example, with the chat service, consider the policy that is associated with videoChat for chat service, the membership function $\mu_{\text{CPU\_clockRate high}}(best\_value\_of(c_1))$ corresponds to the most suitable value for the context parameter c_1 associated with the policy videoChat for the chat service. It is used for calculating the standard reference value associated with a certain policy for a particular service. For the chat service, the standard reference must include the fuzzified parameters for all the context values c_1, c_2, c_3 and c_4. Model reference adaptive control is implemented by developing a model reference depository for each service in the mobile platform. The model reference depository for the mobile application platform should include all the standard reference values for all policies. The current mobile application example provides two services: s_1=chat service, s_2=email service, and the model reference depository includes the standard reference values for all the policies for chat and email services (as shown in table 2). In the following paragraphs, the details of the SA-FSAM are presented:

Definition 1 *(Service):* A service is a functionality provided by the middleware in the application. The services are delivered at multiple QoS level with different resource constraints based on different policies. Let S= {$s_1, s_2, s_3, \ldots, s_q$} ($1 \le q$), represents the service set, where s_i ($1 \le i \le q$) represents the i-th service, and q represents the total number of services available.

Definition 2 *(Policy):* A policy represents a method used to deliver a service with a certain resource requirement and quality-of-service condition. Let P_i= {$p_i^1, p_i^2, \ldots, p_i^{mi} \mid i \in [1, q]$} be a set of policies, where p_i^j (with $1 \le j \le m_i$) represents the j-th policy corresponding to the i-th service s_i, and m_i represents the total number of policies available for the i-th service s_i, and q represents the total number of services available.

Definition 3 *(Context):* Let C={$c_1, c_2, \ldots, c_n$} be a set of context parameters, where c_a ($1 \le a \le n$) represents the a-th context parameter, and n is the total number of context parameters detected by the middleware.

Definition 4 *(Context Situation):* Context Situation is defined as the composite parameters to represent a context at any given time t. The Context Situation at time t is denoted by a set, which includes n pairs of 3-element tuples:

$SI(t) = \{(c_a, lv_b, \mu_{c_a\ lv_b}(\text{value\_of}(c_a, t)) \mid c_a \in C, a \in [1,n]\ b \in [1,k]\}$

where, c_a ($1 \leq a \leq n$) is the a-th context (e.g. c_1=Network_maxRate), n is the total number of context parameters, lv_b *with* ($1 \leq b \leq k$) is a linguistic value (e.g. lv_2=high), $\mu_{c_a\ lv_b}(x) \in [0,1]$ is the predefined membership function for "c_a is lv_b", *value_of*(c_a,t) represents the value of context c_a at time t.

For example, when a=1, c_1= Network_maxRate, lv_2=high, t=current, and the value associated with *value_of*(c_a, t)= *value_of*(Network_maxRate, *current*) = 10 M bps, then a possible value of $\mu_{\text{Network\_maxRate high}}$(10Mbps) is 0.5. The value 0.5 means that Network_maxRate is high with a degree of 0.5 when the Network_maxRate is 10Mbps at the current time interval.

Definition 5 *(Standard Reference):* Given a service s_i, with respect to each policy p_i^j where ($1 \leq j \leq m_i$), it is assumed that there exists a specific Context Situation associated with p_i^j. On the basis of such a Context Situation, the policy p_i^j is the most suitable policy for service s_i, and should be adopted. Intrinsically, the most suitable policy means a tradeoff between resource constraints and the QoS level to be delivered. The most suitable Context Situation for policy p_i^j associated with service i is referred as a Standard Reference $SR(p_i^j)$.

Given a set of linguistic values $LV = \{lv_1, lv_2, \ldots, lv_k\}$, $SR(p_i^j)$ can be represented by a set of 3-element tuples:

$SR(p_i^j) = \{(c_a, lv_b, \mu_{c_a\ lv_b}(best\_value\_of(c_a)) \mid c_a \in C, a \in [1,n], lv_b \in LV, b \in [1,k]\}$.

where n is the total number of context parameters, and k is the total number of linguistic variables, and *best_value_of*(c_a) refers to the most suitable value for context c_a corresponding to policy p_i^j for service i.

Definition 6 (Model Reference Depository): With m_i corresponding to the number of the policies for service s_i, the Model Reference Depository for P_i is defined to be the set of Standard Reference $\{SR(p_i^1), SR(p_i^2), \ldots, SR(p_i^{m_i})\}$, and is denoted by $SRD(P_i)$. The values for the Standard Reference Depository represent the most suitable contextual values for each policy. They can be obtained from empirical experiments. During the adaptation process, the Model Reference Depository values are organized into a two-dimensional table (as shown in Table 2).

Formula 1 (Fitness Function): Given a service s_i (for example s_1=chat service), the Context Situation at time t is at certain distance away from any $SR(p_i^j)$, the Fitness Function is defined to evaluate the fitness degree of the Context Situation at time t against the Standard Reference $SR(p_i^j)$. The fitness function $FF(SI(t), SR(p_i^j))$ for the corresponding context situation and policy is defined as:

$$FF(SI(t), SR(p_i^j)) = \frac{1}{\sum_{y=1}^{size\_of(SR(p_i^j))} \left| \mu(\text{best\_value\_of}(c_y)) - \mu(\text{value\_of}(c_y, t)) \right|^{l_y}} \quad (1)$$

where *size_of*($SR(p_i^j)$) represents the number of tuples in $SR(p_i^j)$, $\mu(x)$ is the membership function corresponding to the context parameter c_y, l_y is a positive integer corresponding to the weight attached to context parameter c_y, and $\mu(\text{value\_of}(c_y, t))$ refers to the degree of membership for context parameter c_y at time t.

Formula 2 (Self-Adaptive Fitness Function): The Fitness Function is extended to the Self-Adaptive Fitness Function in order to enable self-adaptivity. Given a service s_i, the Self-Adaptive Fitness Function is the mapping from the jointed distances to the fitness degree of policy p_i^j, which aggregates the distance of Context Situation at time t to the Standard Reference Context Situation $SR(p_i^j)$, and the mean distance of the Context Situation at previous time intervals t_m, t_{m-1},…, t_{m-k}.

$$SA\text{-}FF(SI(t), SR(p_i^j)) = \frac{1}{\sum_{y=1}^{size\_of(SR(p_i^j))} \left(\left| \mu(\text{best\_value\_of}(c_y)) - \mu(\text{value\_of}(c_y, t)) \right|^{l_y} + \frac{\sum_{z=m-k}^{m} \left| \mu(\text{best\_value\_of}(c_z)) - \mu(\text{value\_of}(c_z, t)) \right|^{l_z}}{k+1} \right)} \quad (2)$$

The Self-adaptive Fitness Function takes the current Context Situation and the Contextual Situation in previous k+1 time intervals into calculation. The parameter k is used to calculate the mean of the previous Context Situation, and is regarded as the damping factor to adjust the damping effect. The Self-adaptive Fitness Function applies when large-granular fluctuations occur.

In formula (1) and formula (2), the denominators are used for the calculation of the distance between $SI(t)$ and $SP(p_i^j)$. After obtaining the fuzzy distance, the

reciprocal value is calculated to obtain the fitness degree. When l_i=*1*, the function uses Hamming Distance; when l_i=*2*, the function uses Euclidean Distance, both are classical methods for calculating fuzzy distance between two states. When l_i=3, l_i is regarded as a weight value for contexts, which can be adjusted by specific applications to have an effect on policy selection[1].

Definition 7 (FSAM): The FSAM is a mapping from the current Context Situation SI(current) to a set of suitable policies $P_{suitable}$, using the fitness function FF in formula (1), where each element of $P_{suitable}$ is the most suitable policy associated with a service $s_i \in S_{need}$. The number of elements in $P_{suitable}$ is equal to the number of elements in S_{need}.

Definition 8 *(SA-FSAM):* The SA-FSAM is a mapping from the current Context Situation *SI*(*current*) to a set of suitable policies $P_{suitable}$, using the self-adaptive fitness function SA-FF in formula (2), where each element of $P_{suitable}$ is the most suitable policy associated with a certain service $s_i \in S_{need}$. The number of elements in $P_{suitable}$ is equal to the number of elements in S_{need}. The SA-FSAM and the FSAMs uses the same algorithm. By using the self-adaptive fitness function SA-FF in the SA-FSAM, self-adaptivity is achieved.

For context-aware mobile middleware, application profile, user preference and system feedback are often considered as rules for service adaptation. In this paper, such rules are regarded as *additional interventions*. Three types of additional interventions are defined here:

- Exclusive Intervention: For a given service s_i, certain p_i^j in P_i cannot be used.

- *Preference Intervention*: For a given service s_i, certain p_i^j in P_i should be used.

- *Conditional Intervention*: For a given service s_i, certain p_i^j in P_i should be used under certain context situation.

Before calculating the fitness degrees of policies, additional interventions are used to select the set of policies that can be possibly applied, i.e. additional intervention rules are used to filter P_i, and then get a subset of P_i (denoted as P_i') whose elements satisfy all the intervention rules, and then we start to calculate SRD(P_i') and use fitness function to obtain the final choice. As described in Fig. 6, the SA-FSAM algorithm includes four major steps:

```
Upon adaptation
{ //Convert the current context value into the fuzzy-based context situation
    for each c_a ∈C do
  {
     for each c_a-related predefined membership function μ_{c_a lv_j}(x) do
      {
        SI(current) = SI(current) ⋃ { (c_a, lv_j, μ_{c_a lv_j}(value_of(c_a,current))) };
      }
  }
  //Select the most suitable policies for each needed service
  for each s_i ∈S_need do
  { //Establish SRD for P_i'
      for each p_i^j ∈P_i do
      {//Calculate SR(p_i^j) using related predefined membership function;
         SRD(P_i') = SRD(P_i') ⋃ {SR(p_i^j)};
      }
     //Calculate the Fitness Degree with Fitness Function for each policy
      and select the most suitable one
     for each p_i^j ∈ P_i' do
     {
        FD(p_i^j) = FF (SI(current), SR(p_i^j));
        if (FD(p_i^j) is larger) then best_policy_for_s_i = p_i^j;
      }
     //get the set P_suitable for S_need
     P_suitable = P_suitable ⋃ { best_policy_for_s_i };
  }
  if (|P_suitable (t) - P_suitable (t-1)| is greater than E_xpected,) then
  {
  for each p_i^j ∈ P_i' do
     {//Calculate the Fitness Degree with Self-Adaptive Fitness
      Function again if large-granular fluctuation occurs.
        FD(p_i^j) = SA-FF (SI(current), SR(p_i^j));
        if (FD(p_i^j) is larger) then best_policy_for_s_i = p_i^j;
      }
     //get the set S_uitable for S_need
     P_suitable = P_suitable ⋃ { best_policy_for_s_i };
  }
} //end of adaptation
```

Fig. 6. The SA-FSAM algorithm

Step 1, each context is fuzzified into a linguistic variable with associated linguistic values. All the values are composed as Context Situation.

Step 2, the Context Situation and a membership degree determined by predefined membership at time interval *t*, i.e. SI(t), is substituted into the Fitness Function to calculate the fitness degrees corresponding to each policy.

Step 3, after all fitness degrees are compared, the policy with the largest fitness degree is specified as the most suitable policy $P_{suitable}$, i.e. If FD(p_i^j) is the maximum fitness degree, p_i^j is selected as the most suitable policy $P_{suitable}$.

Step 4, if the distance between the current $P_{suitable}$ and the $P_{suitable}$ in the previous time interval is less than an expectation $E_{xpected}$, the corresponding QoS level is delivered. However, if the distance between the current $P_{suitable}$ and the previous $P_{suitable}$ is greater than $E_{xpected}$, which implies that large-granular fluctuations occur, the fitness degrees will be calculated once again with the Self-Adaptive Fitness Function. Until each available policy has been assigned a new fitness degree, the policy with the largest fitness degree is considered as the most suitable policy $P_{suitable}$, so that the most appropriate QoS level is determined again.

In the SA-FSAM algorithm, the parameter $E_{xpected}$ is set as the threshold value for determining whether or not a large-granular fluctuation occurs. It denotes the condition in the closed-loop control and should be attained according to application preference and user experience. As discussed before, the parameter *K in the Self-Adaptive Fitness Function* acts as the damping factor to adjust damping effect. The provision of $E_{xpected}$ and the K in the SA-FF (Self-Adaptive Fitness Function) enables the context-aware middleware to precisely adjust the extent of self-adaptivity and the damping effect to provide adjustment mechanism for the service-adaptation engine, so that the performance of the application can be fine-tuned according to the needs.

6. Implementation and Evaluation

The SA-FSAM is implemented as the inference engine in the middleware architecture. A Campus Assistant application is used to demonstrate and evaluate the effectiveness of adaptation decisions. The major functionalities of the Campus Assistant are to provide chat or email services at different QoS level according to the changing contexts, e.g. the network bandwidth and the memory. The variations in the context situations are simulated, and changes in the context values may trigger a service adaptation and corresponding adjustment mechanisms in the fuzzy controller. With the experimental results, the performance of the SA-FSAM is evaluated by comparing SA-FSAM with a conventional threshold-based linear control model.

6.1. *A Campus Assistant Application*

The Campus Assistant application is a mobile context-aware application running on mobile platforms. Through the wireless access, the Campus Assistant enables users to communicate with each other in a real-time interactive manner, or to receive and send emails, by providing Chat and Email services. In particular, due to the spatial and temporal variations of wireless communication and computing resources, the Campus Assistant application tries to detect the changing contexts and reflectively deliver the most suitable QoS level, while maintaining an acceptable user experience when the resource constraints become tight or even severe. Such adaptations are predefined by certain rules (or policies), in the application. The Chat service has three policies corresponding to three QoS levels for Chat service: textChat, voiceChat and videoChat. The Email service has five policies corresponding to five QoS levels for Email service: headMail, fullMail, encryptedMail, bigMail and encryptedBigMail. The contexts cover many aspects because of the multi-dimensional characteristics including communication, computing, geographical, organizational, etc. To simplify context modeling without loss of generality, four kinds of contexts: Network_maxRate, CPU_clockRate, Network_delay, RAM_freeSpace are taken into account in the simulation.

6.2. *Setting in the Campus Assistant Application*

The Campus Assistant application, it is assumed that before carrying out each round of service adaptation, the middleware has obtained the following information:

Service: S = {Chat, Email}, which represents two categories of services Chat and Email provided by middleware.

Policy for S_1 and S_2:

P_1 = {textChat, voiceChat, videoChat}

P_2 = {headMail, fullMail, encryptedMail, bigMail, encryptedBigMail}, which denotes the policies relating to the services provided at different QoS levels.

Context: C = {Network_maxRate, CPU_clockRate, Network_delay, RAM_freeSpace}, where C denotes four kinds of context include network bandwidth, CPU usage, network delay and memory usage.

Linguistic Values: LV = {low, high}

Context Situation: SI(t), SI(t) is the 3-tuple vector to represent the fuzzified context:

$$SI(t) = \{(\text{Network\_maxRate, high, } \mu_{\text{Network\_maxRate high}}(\text{value\_of(Network\_maxRate, t}))),$$
$$(\text{CPU\_clockRate, high, } \mu_{\text{CPU\_clockRate high}}(\text{value\_of (CPU\_clockRate, t}))),$$
$$(\text{Network\_delay, low, } \mu_{\text{Network\_delay low}}(\text{value\_of (Network\_delay, t}))),$$
$$(\text{RAM\_freeSpace, high, } \mu_{\text{RAM\_freeSpace high}}(\text{value\_of (RAM\_freeSpace, t})))\}$$

The graphs of the membership functions for network bandwidth, CPU clock rate, network delay and RAM free space are described in Fig. 7. The corresponding formulas for the membership functions C_1, C_2, C_3 and C_4 are defined by (3), (4), (5) and (6). Fig. 8 shows the variations of context values for the four context parameters.

Standard Reference Context Situation: The most suitable values of contexts associated to each policy are application-specific and determined by relevant domains. They are predefined in the simulation and should be heuristically adjusted to the best values in practice. Table 1 shows the most suitable values (Standard Reference Context Situations) for the resource required by each policy.

The membership function for Network_maxRate high:

$$C_1=\begin{cases} 0 & B<1\text{Kbps} \\ \dfrac{\log_{10} B/1K}{5} & 1\text{Kbps}\leqslant B\leqslant 100\text{Mbps} \\ 1 & B>100\text{Mbps} \end{cases} \quad (3)$$

The membership function for CPU_clockRate high:

$$C_2=\begin{cases} 0 & F<2\text{MHz} \\ \dfrac{\log_{10} F/2M}{3} & 2\text{MHz}\leqslant F\leqslant 2\text{GHz} \\ 1 & F>2\text{GHz} \end{cases} \quad (4)$$

The membership function for Network_delay low:

$$C_3=\begin{cases} 0 & T>1000\text{ms} \\ 1-\dfrac{\log_{10} T/0.1}{4} & 0.1\text{ms}\leqslant T\leqslant 1000\text{ms} \\ 1 & T<0.1\text{ms} \end{cases} \quad (5)$$

The membership function for RAM_freeSpace high:

$$C_4=\begin{cases} 0 & R<50\text{KB} \\ \dfrac{\log_{10} R/50K}{4} & 50\text{KB}\leqslant R\leqslant 500\text{MB} \\ 1 & R>500\text{MB} \end{cases} \quad (6)$$

Table 1 Most Suitable Context Values for the Policies

	Network_maxRate(kbps)	CPU_clockRate(MHz)	Network_delay(ms)	RAM_freeSpace(KB)
textChat (p_1^1)	4	20	500	0.2
voiceChat (p_1^2)	200	300	10	4
videoChat (p_1^3)	10000	1000	0.2	200
headMail (p_2^1)	2	4	n/a	0.2
fullMail (p_2^2)	10	10	n/a	0.4
encryptedMail (p_2^3)	10	100	n/a	10
bigMail (p_2^4)	500	50	n/a	2
encryptedBigMail (p_2^5)	500	1000	n/a	100

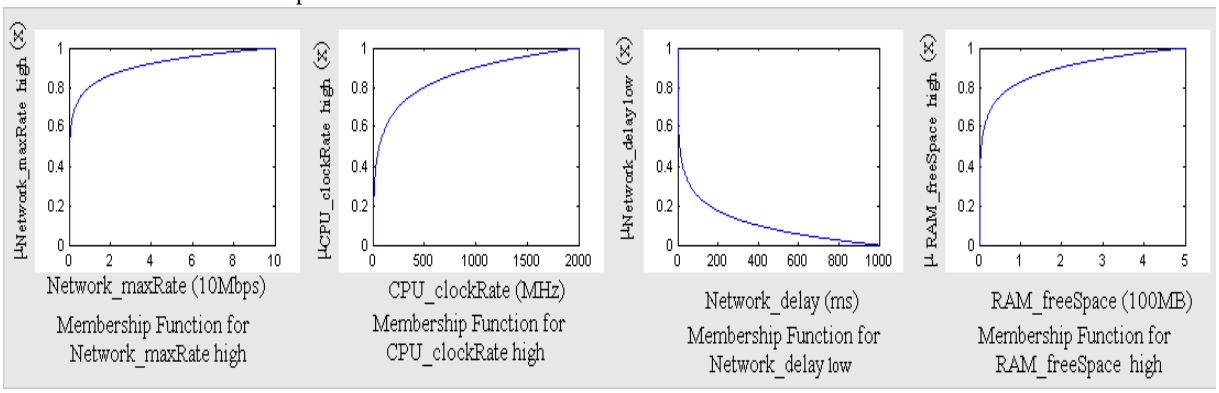

Fig. 7. The Membership Functions

Model Reference Depository SRD(P_1') and SRD(P_2'): Based on the most suitable values in Table 1 and the membership functions (3), (4), (5) and (6), the values for SRD(P_1') and SRD(P_2') are calculated and listed in Table 2. It is assumed that the Chat service requires interaction between users so that the Network_delay is taken into account. For Email service, which does not require interaction rather than passive reading, Network_delay is neglected.

Table 2 Model Reference Depository, SRD(P1') and SRD(P2')

	Network_ maxRate High	**CPU_clock Rate High**	**Network_ delay Low**	**RAM_ freeSpace High**
SR(p_1^1)	0.12	0.33	0.08	0.15
SR(p_1^2)	0.46	0.72	0.50	0.48
SR(p_1^3)	0.80	0.90	0.92	0.90
SR(p_2^1)	0.06	0.10	n/a	0.15
SR(p_2^2)	0.20	0.23	n/a	0.23
SR(p_2^3)	0.20	0.57	n/a	0.58
SR(p_2^4)	0.54	0.47	n/a	0.40
SR(p_2^5)	0.54	0.90	n/a	0.83

With the above information, the steps for developing the SA-FSAM are illustrated. First, the predefined membership functions are used to map the current context into the Context Situation as follows:

$SI(t) = \{$ (Network_maxRate, high, $\mu_{\text{Network\_maxRate high}}$ (value_of (Network_maxRate, t))),
(CPU_clockRate, high, $\mu_{\text{CPU\_clockRate high}}$ (value_of (CPU_clockRate, t))),
(Network_delay, low, (v $\mu_{\text{Network\_delay low}}$ (value_of (Network_delay, t))),
(RAM_freeSpace, high, $\mu_{\text{RAM\_freeSpace high}}$ (value_of (RAM_freeSpace, t)))}

Secondly, the Fitness Function and Self-Adaptive Fitness Function are utilized to calculate the fitness degrees of the current context for each policy. The SA-FSAM algorithm is applied and the values of the current context SRD(P_1) and SRD(P_2) are substituted into the formulas FF and SA-FF, respectively.

Regarding the Self-Adaptive Fitness Function, the threshold value of expectation E_{xpected} is set to 2, which implies that any service adaptation jumping across two QoS levels will be detected as a large-granular fluctuation. The damping factor K in SA-FF is set to two, which means that the historical information at the previous two time intervals is considered. Finally, the policy $P_{suitable}$ which has the maximum fitness degree will be the most suitable one to be adopted.

For Chat service:

$$P_{suitable} = \text{Max } \{FF(SI(t), SR(P_1^1)), FF(SI(t), SR(P_1^2)), FF(SI(t), SR(P_1^3))\}$$

For Email service:

$$P_{suitable} = \text{Max } \{FF(SI(t), SR(P_2^1)), FF(SI(t),SR(P_2^2)), FF(SI(t),SR(P_2^3)), FF(SI(t),SR(P_2^4)) FF(SI(t),SR(P_2^5))$$

6.3. *Experimental Results*

We simulate the variations in contexts values by generating 160 sets of 4-element tuples such as Network_maxRate, CPU_clockRate, Network_delay and RAM_freeSpace. It is assumed that the context parameters for network bandwidth, network delay, CPU usage and memory usage represent the major influence on the performance of mobile applications. The 160 sets of 4-element tuples are fed into the SA-FSAM inference engine round by round in a time series manner. For evaluation, a conventional threshold-based linear-control approach is also implemented in the Campus Assistant application.

By referring to the formula for QoS parameter aggregation in the video distribution system by Koliver *et al.*[22], we follow their approach to produce the conventional context aggregation values. The QoS parameters are considered as a subset of the contexts and the formula is generalized to be used in service adaptation. A service adaptation mode M is defined by the formula: $M = (f_1 w_1 + f_2 w_2 + \ldots + f_i w_i)$, where f_i denotes the adaptation factor of each independent context, w_i represents the weight value for each context, $i=\{1,2,3,4\}$, and w_i is used to adjust the individual effect on adaptation of each context and alleviate the compensation among them. According to the predefined association between the mode M and each policy, the most suitable policy is adopted to deliver the most suitable service, according to the context at a given time t. In the mapping from each context c_i to its adaptation

factor: $f_i = s_i c_i$, where $c_i \in C_i, i=\{1,2,3,4\}$, s_i is a scaling factor to normalize f_i , C_i = [*min*, *max*] determines a valid numerical range for each context, and *min* is the lower bound while *max* is the upper bound. When the context is out of the range of C_i, it is rejected and neglected.

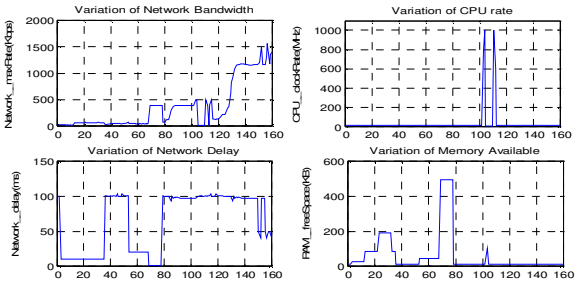

Fig. 8. Variations of Contexts

In the implementation, the setting of the threshold-based linear control is outlined as follows:

Context: C = {Network_maxRate, CPU_clockRate, Network_delay, RAM_freeSpace}
Weight: $w_i = 1$
Scaling Factor: $s_i = 1$
Adaptation Factor: $f_{i,} = c_{i,}$ where $i=\{1,2,3,4\}$
For Chat service: $M = c_1 / c_3 + c_2 + c_4$
For Email service: $M = c_1 + c_2 + c_4$

The threshold values are defined in Table 3 and Table 4 respectively.

Table 3 Thresholds of Adaptation Mode for Chat

Chat Policy	***M***
textChat(p_1^1)	[*min*, 53.2)
voiceChat(p_1^2)	[53.2, 926)
videoChat (p_1^3)	[926, *max*)

Table 4 Thresholds of Adaptation Mode for Email

Email Policy	***M***
headMail (p_2^1)	[*min*, 15.2)
fullMail (p_2^2)	[15.2, 38)
encryptedMail (p_2^3)	[38, 98)
bigMail (p_2^4)	[98, 496)
encryptedBigMail(p_2^5)	[496, *max*)

With the 160 sets of tuples as the input (there are 8 groups while each group contains 20 sets of tuples), the performance of the SA-FASM and FASM models and the threshold-based linear control was evaluated. The three models produce the results to determine the respective service adaptation. The adaptation decisions are recorded in the log files and compared as in shown Fig. 9 and Fig. 10. In general, it is observed that the variations caused by the small-granular oscillations of context are filtered by the fuzzy inference engines associated with both the SA-FASM and FASM models. With the FSAM and the SA-FSAM, there is no service adaptation from time interval t = 3 to time interval t = 38 for Email service (as shown in Fig. 9). For Chat Service there are no service adaptations for the FSAM and SA-FSAM from the time interval t = 52 to time interval t = 78 (as shown in Fig. 10). In contrast, the threshold-based inference engine is sensitive and produces zigzag fluctuations in service adaptation. In a situation of severely fluctuations in context values, the service adaptations become overactive and keep changing accordingly. These small-granular oscillations in service adaptation may lead to unbearable deterioration of QoS and consumption of large amounts of precious communication and computational resources.

From a microscopic view, it is not difficult to observe that the adaptation curves generated by the SA-FASM and FSAMs are relatively smoother than those by the threshold-based linear control. It indicates that the fuzzy-based approaches have better tolerance to these marginal small-granular oscillations of context, which helps to improve the effectiveness of the service adaptation and resource utilization.

However, in the event of large-granular fluctuation in context values, which is the Email service from the time interval t = 104 to time interval t = 115 shown in Fig. 9, the large-granular fluctuations in service adaptation are not handled using the FSAM. The conventional fuzzy-based technique cannot process such fluctuations, since the variations of contexts are neither marginal nor minor. Using the SA-FSAM approach, the closed-loop control enables the inference engine to detect the change adaptations that are greater than one QoS level, and adjust the radical change with the Self-adaptive Fitness Function. The past adaptations in the previous two time intervals have a joint effect on the current adaptation decision to decide the damping effect. By including self-adaptivity in a time series, the adaptations of the application system to those large-granular fluctuations become adjustable.

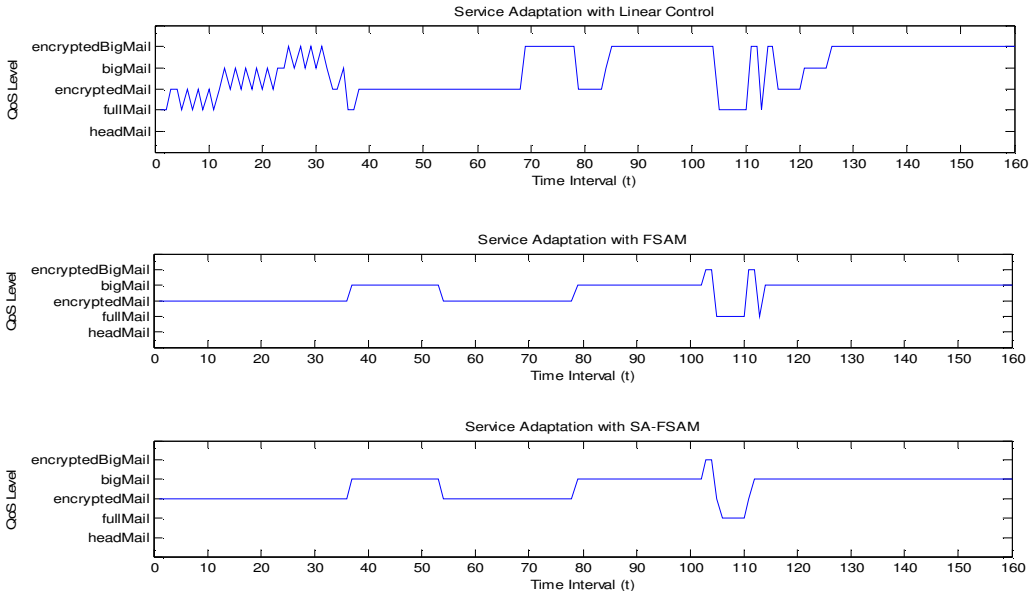

Fig. 9. Email Service Adaptation

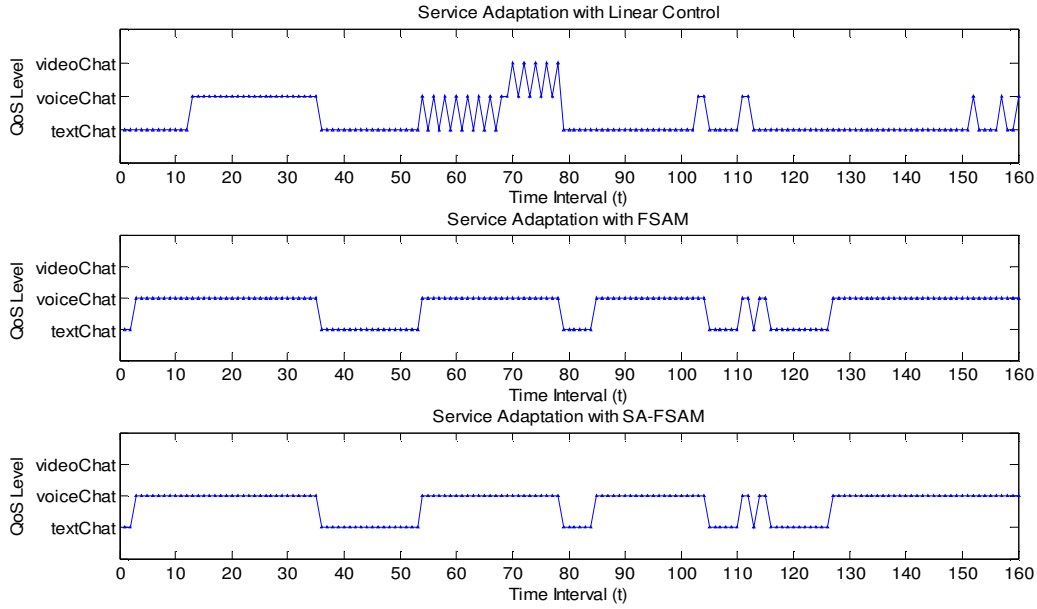

Fig.10. Chat Service Adaptation

It can be seen from Fig. 9 that for Email service, the SA-FSAM eliminates the zigzag adaptations in the FSAM in the time internal t = 113. The SA-FSAM migrates smoothly to the next QoS level compared with the FSAM, avoiding drastic changes in QoS level resulting from adaptation decisions.

7. Conclusions and Future Work

In this research, a generic adaptive middleware infrastructure is introduced. The middleware infrastructure accommodates a self-adaptive model for service adaptation. A Model Reference Adaptive Control mechanism is implemented in the SA-FSAM middleware. Control-theory and fuzzy-based approaches are combined to develop the SA-FSAM in the middleware. The core work lies on the Fitness Function and the Self-Adaptive Fitness Function. A campus assistant application is used to demonstrate the adaptation process. Based on the experimental results, it is observed that the SA-FSAM inference engine effectively alleviates the small-granular oscillations in context, and also smoothes the large-granular fluctuations that occur due to drastic changes in context information. The mechanisms behind the SA-FSAM are utilization of historical adaptation information and closed-loop control. With introduction of model reference adaptive control mechanisms, the static FSAM is enhanced to a dynamic control process. The performance of the context-aware middleware is enhanced in terms of resource utilization by avoiding drastic changes in adaptation decisions.

Certain limitations lie in the proposed framework. In practical scenarios, the research work provides a generic framework for mobile application development. However, the effectiveness also relies on the fuzzy membership functions used for defining context situations. It requires domain knowledge, user experience, and application preference to construct suitable membership functions for different context parameters. One direction of the future work would aim at developing suitable membership functions for different context situation parameters. Another direction of future research would be to enhance the adjustment mechanisms in the model reference adaptive control process by making use of the historical records of context values and services adaptations to make context predictions.

Acknowledgments

The authors would like to thank Gang Yao for providing the simulation results for the evaluations.

References

1. C. V. Altrock, Fuzzy Logic and Neuro-Fuzzy, *Applications Explained* (Prentice Hall, 1995).
2. C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, D. Riboni. A Survey of Context Modelling and Reasoning Techniques, *Pervasive and Mobile Computing*, **6**(2) (2010) 161-180.
3. J. Cao, N. Xing, A. T. S. Chan, Y. Feng, B. Jin, Service Adaptation Using Fuzzy Theory in Context-aware Mobile Computing Middleware. *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, (Hong Kong, China, 2005), pp. 496–501.
4. L. Capra L., Mobile Computing Middleware for Context-Aware Applications. *Proceedings of the 24th International Conference on Software Engineering*, (Limerick, Ireland, 2000), pp. 723–724.
5. L. Capra, W. Emmerich, C. Mascolo, CARISMA: Context-aware Reflective Middleware System for Mobile Applications. *IEEE Transactions on Software Engineering*, **29**(10) (2003) 929–945.
6. P. Castro, M. Mani, S. Mathur, R. Muntz, Managing Context for Internet Video Conferences: The Multimedia Internet Recorder and Archive. *Proceedings of Multimedia and Computer Networks* , (San Jose, CA., 2000), pp. 1-12
7. P. M. L. Chan, R. E. Sheriff, Y. F. Hu, P. Conforto, C. Tocci, Mobility Management Incorporating Fuzzy Logic for Heterogeneous an IP environment., *IEEE Communications Magazine*, **39**(12) (2001) 42–51.
8. P. R. Chang, B. C. Wang, Adaptive Fuzzy Power control for CDMA Mobile Radio Systems. *IEEE Transactions on Vehicular Technology*, **45**(2) (1996) 225–236.
9. B. H. Cheng, R. de Lemos, H. Giese et al., Software engineering for self-adaptive systems: A research roadmap. In: Cheng, B.H., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) *Software Engineering for Self-Adaptive Systems*, LNCS, vol. 5525 (Springer, Heidelberg, 2009).
10. R. Cheung, An Adaptive Middleware Infrastructure Incorporating Fuzzy Logic for Mobile Computing, *Proceedings of the International Conference on Next Generation Web Services Practices*. IEEE Computer Society Press, (South Korea, 2005), pp. 449–451.
11. R. Cheung, J. Cao, G. Yao, A. T. S. Chan, A Fuzzy-based Service Adaptation Middleware for Context-aware Computing. *Proceedings of the IFIP International*

Conference on Embedded and Ubiquitous Computing, (Korea, 2006), pp. 580–590.
12. R. Cheung, G. Yao, J. Cao, A. T. S Chan, A Fuzzy Service Adaptation Engine for Context-aware Mobile Computing Middleware. *International Journal of Pervasive Computing and Communications*, **4**(2) (2008) 147–165.
13. S. N. Chuang, A. T. S. Chan, J. Cao, R. Cheung, Dynamic Service Reconfiguration for Wireless Web Access. *Proceedings of the 12th International World Wide Web Conference*, ACM Press, (Budapest, Hungary, 2003), pp. 58–67.
14. A. K. Dey, Understanding and using context.. *Personal and Ubiquitous Computing*, **5**(1) (2001) 4-7.
15. G. Dumont, M. Huzmezan, Concepts, Methods and Techniques in Adaptive Control. Proceedings of the *IEEE American Control Conference*, (Anchorage, USA, 2002), Vol 2 1137–1150.
16. G. Ghinea, G. A. Magoulas, C. Siamitros, Perceptual Considerations in a QoS Framework: a Fuzzy Logic Formulation. *IEEE Fourth Workshop on Multimedia Signal Processing*, (Cannes , France, 2001), pp. 353–358.
17. K. Hirota, W. Pedrycz, Analysis and Synthesis of Fuzzy Systems by the use of Fuzzy Sets. *Fuzzy Sets and Systems*, **10**(1) (1983) 1–13.
18. J. Hong, E. Suh, S. Kim S., Context-Aware Systems: A literature Review and Classifications, *Expert Systems with Applications*, **36**(4) (2009) 8509-8522.
19. J. R. Jang, Self-learning Fuzzy Controllers Based on Temporal Back Propagation. *IEEE Transactions on Neural Networks*, **3**(5) (1992) 714–713.
20. B. Johanson, A. Fox, The Event Heap: An Enabling Infrastructure for Interactive Workspaces. *Proceedings of the 4th IEEE Workshop on Mobile Computer Systems and Applications*. IEEE CS Press, (Callicoon (NY), USA, 2002), pp. 1-19
21. S. Kim, P. K. Varshney, Adaptive Online Bandwidth Allocation and Reservation for QoS Sensitive Multimedia Networks, *Computer Communications* **28**(17) (2005) 1959–1969.
22. C. Koliver, J. M. Farines, K. Nahrstedt, QoS Adaptation Based on Fuzzy Theory. *Soft Computing for Communications* (Springer-Verlag, 2004).
23. V. W. M. Kwan, F. C. M. Lau, C. L. Wang, Functionality Adaptation: A Context-aware Service Code Adaptation for Pervasive Computing Environments. *Proceedings of IEEE/WIC International Conference on Web Intelligence*, (Halifax, Canada, 2003) , pp. 358–364.
24. B. Li, K. Nahrstedt, A Control-based Middleware Framework for Quality of Service Adaptations. *IEEE Journal on Selected Areas in Communications*, **17**(9) (1999) 1632–1650.
25. K. R. Lo, C. J. Chang, C. Chang, C. B. Shung, A QoS-guaranteed fuzzy channel allocation controller for hierarchical cellular systems. *IEEE Transactions on Vehicular Technology*, **49**(5) (2000) 1588–1598.
26. H. A. Müller, M. Pezzè, M. Shaw, Visibility of control in adaptive systems. Proceedings of the Second International Workshop on Ultra-Large-Scale Software-Intensive Systems, Workshop at 30th IEEE/ACM International Conference on Software Engineering, (Leipzig, Germany, 2008), pp. 23-26.
27. M. Roman, C. K. Hess, R. Cerqueira, R. H. Campbell, K. Narhstedt, Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, **1** (2002) 74–83.
28. B. Qiu, The application of fuzzy prediction for the improvement of QoS performance. Proceedings of *IEEE International Conference on Communications*, Vol 3 (Atlanta, GA , USA, 1998), pp. 1769–1773.
29. A., Robertson, B. Wittenmark, M. Kihl, Analysis and design of admission control in Web-server systems. *Proceedings of American Control Conference*, Vol 1 (2003, Denver, CO), pp. 254–259.
30. S. A. N. Shafer, B. Brumitt, J. Cadiz, Interaction Issues in Context-Aware Interactive Environments. Special issue on Context-Aware Computing, *Human-Computer Interaction*, **16**(2) (2001) 363–378.
31. Y. M. Siu, K. K. Soo, CDMA Mobile Systems with Tailor-made Power Control to Each Mobile Station. *Proceeding of the First International Conference on 3G Mobile Communication Technologies*, (London, UK, 2000), pp. 46–50.
32. H. Takagi, Cooperative System of Neural Network, Fuzzy Logic and its Application to Consumer Products, *Industrial Applications of Fuzzy Control and Intelligent Systems*, (Von Nostrand Reinhold, New York, 1993).
33. Y. Tu, S. Liu, S. Prabhakar, B. Yao, Load Shedding in Stream Databases: A Control-based Approach, *Proceedings of the 32nd International Conference on Very Large Data Bases*, Vol 32 (Seoul, Korea, 2006) pp. 787–798.
34. T. Winograd, Architectures for Context. Human-Computer Interaction, *Human Computer Interaction* **16**(2) (2001) 401-419.
35. S. Yau, F. Karim, Y. Wang, B. Wang, S. Gupta, Reconfigurable Context-Sensitive Middleware for Pervasive Computing. *IEEE Pervasive Computing*, July-Sept (2002) 33–40.
36. L. A. Zadeh, Fuzzy sets, *Information and Control* **8**(3) (1965) 338–353.