A TAO-based adaptive middleware for pervasive computing

Yanhui Guo, Zhenmei Yu*, Fuli Qu and Hui Liu

School of Data and Computer Science,
Shandong Women's University,
Ji'nan, Shandong, China
Email: guoyanhui03@163.com
Email: zhenmei_yu@sdwu.edu.cn
Email: qufuli23@163.com
Email: 972648894@qq.com
*Corresponding author

Abstract: Over the past years, a considerable amount of effort has been devoted, both in industry and academia, towards the development of innovative applications for the internet of things (IoT). An important challenge of IoT application is to adapt to dynamic environments, which can be modified at runtime considering the emergence of other requirements. To address this issue, pervasive computing can provide us with a good solution. Aiming at the environments which are open, dynamic and heterogeneous, we propose an adaptive middleware named PAmiddleware. PAmiddleware is service-oriented, context-aware, and supported by QoS. The architecture of PAmiddleware is based on TAO which is a standard-based, CORBA middleware framework. Meanwhile, we present a model for context awareness to allow the adaptation and give a modelling method of context description for context resource. We also propose an adaptive strategy which uses a genetic algorithm for optimisation. The proposed model can well meet the needs of pervasive computing, and provide more convenient service.

Keywords: pervasive computing; middleware; components; context awareness; adaptive.

Reference to this paper should be made as follows: Guo, Y., Yu, Z., Qu, F. and Liu, H. (2021) 'A TAO-based adaptive middleware for pervasive computing', *Int. J. Embedded Systems*, Vol. 14, No. 2, pp.108–116.

Biographical notes: Yanhui Guo received his BS in Information Management and Information System from the Xi'an University of Finance and Economics, in 2006, and MS in Computer Software and Theory from the Shaanxi Normal University, in 2009. Since 2009, he has been with the School of Information Technology, Shandong Women's University where he is currently a Senior Lecturer. His research interests include middleware and machine learning.

Zhenmei Yu received her MS from the School of Management Science and Engineering, Shandong Normal University. Currently, she is a Professor at School of Data and Computer Science, Shandong Women's University. Her main research interests include machine learning and artificial intelligence.

Fuli Qu received her Master's in Computational Mathematics from Shandong University. She is an Assistant Professor at the Institute of Data Science and Computing. Her research focuses on numerical methods of differential equations.

Hui Liu graduated from the Shandong Normal University, his major is Computer Science and Technology. His research interests include machine learning and intelligent optimisation.

1 Introduction

Owing to the growing popularity of mobile applications and IoT, the era of pervasive computing has come. Pervasive computing environment is an integration of various kinds of technologies, wherein heterogeneous objects are different with capabilities of sensing, communication, computation, storage. Pervasive devices collect contextual information and run local computation. Therefore, applications in pervasive computing environment need to provide different service according different devices (Kakousis et al., 2010). Al-Khawaldeh et al. (2019) introduce knowledge-based auto-configuration ubiquitous robotics for smart home environments, which utilises the Sobot to achieve auto-configuration of the system. Because devices also want to perceive the dynamic environment

at runtime, to satisfy their requirements. However, these context-aware applications are very difficult to develop and reuse (Knappmeyer et al., 2013). To support a dynamic pervasive computing environment and simplify software development, adaptive middleware can be applied, which provide ubiquitous access to context-aware information. Adaptive middleware technology has given a great impulse to the development of mobile applications and IoT. For pervasive computing, a variety of resources are heterogeneous, dynamic and open. Hence, the middleware needs to be elastic to dynamically adjust their behaviour according to the changing context, to adapt to the different resources and requirements of service.

Generally, pervasive computing platforms are based on service-oriented computing, a compositional approach where applications are built through composition of independent software elements (Chollet et al., 2015). A number of services-oriented platforms related to pervasive computing applications has been developed over the years, such as PCOM (Becker et al., 2004), iCasa (Escoffier et al., 2014). While the diversity of these platforms prevents services and applications developed on different middleware platforms. So, a middleware is needed to deal with various forms of communication, context modelling, interoperability. Several researchers have attempted to introduce adaptations into pervasive computing system. And there have been many attempts to introduce the notion of adaptation into pervasive computing system (Weyns et al., 2012). Previous studies focused on discovery and connection strategies. Discovery-based approaches have been explored to discover services and resources. Jaeger et al. (2007) introduced the self-adaptation to an object request broker. The broker can adapt dynamically to a better efficiency on the application and the network layer. Ben Mokhtar et al. (2006) proposed a web services discover method which can identify service semantic and match it. Connection-based adaptation approach allows communication between software components with connectors or unified interfaces. Garlan et al. (2004) proposed a framework called Rainbow which provided a language for specifying self-adaptation. Its method of specifying adaptation is worth learning. Software-based approach enables software to be dynamically modified in accordance with changes. Later, computational reflection aspect-oriented programming (AOP) enables software to be open to dynamically defining itself without compromising portability or exposing parts unnecessarily. Then, AOP was used for adaptation of applications. Then, context-awareness is also the main aspect of adaptive middleware research, such as Forkan et al. (2014), Vahdat-Nejad et al. (2013) and Freitas et al. (2015). Nocera et al. (2019), focusing on the heterogeneity of different objects, proposed a context-aware middleware based on fuzzy rule. They implemented and validated the proposed model on a real IoT middleware in a smart home scenario. Djeddar et al. (2017) proposed a context-driven composition for mobile applications.

Belcastro et al. (2019) designed a service-oriented middleware, called Geocon. Geocon provided a geocon-service for storing, searching and selecting metadata obout users, resources and place of interest. Finally, Geocon was evaluated on a real world application. Eibel et al. (2018) proposed an energy-aware middleware platform, which adjusts the system configuration to achieve the best energy. Also, some papers mainly study adaptive strategy (Portocarrero et al., 2017; Sun and Satoh, 2016). For complex engineering in automotive or aerospace, Beni et al. (2019) proposed a policy-driven adaptive middleware, which supports smart cloud-based deployment and execution of engineering workflows.

Nevertheless, most pervasive computing middlewares lack quality of service (QoS). QoS has become the main measurement of software performance. For complex pervasive computing environments, it is necessary to clarify the QoS indicators in different devices. Usually, QoS includes user and application expectations relative to network, devices, and data. Agirre et al. (2016) proposed a QoS aware middleware as a support for dynamically reconfigurable component-based applications. This middleware is component-based and driven by a QoS aware self-configuration algorithm. Finally, this platform was deployed over an automated warehouse supervision system. Ouedraogo et al. (2018) presented a run-time pluggable QoS management mechanism for middleware platform. Their approach consists in the dynamic, autonomous, and seamless deployment of QoS management mechanisms, which can improve QoS of middleware services. Lately, a QoS aware middleware was proposed by Mukherjee et al. (2020) for mobile edge computing in opportunistic internet of drone things. They analysed parameters like message transfer latency, overhead ratio, message delivery changes under several QoS values, which enhances the reliability of middleware.

In this paper, we attempt to design a middleware platform for pervasive computing. By analysing above projects, component-based and service-oriented middleware technologies in pervasive computing are commonly used and effective. This paper introduces the QoS mechanisms as part of the context awareness, which better meets the adaptability of the middleware, facilitates application development. The main contributions are listed as follows:

- We designed an adaptive middleware, called PAmiddleware, which is based on real-time platform TAO. We supplemented three aspects, including service mapping, adaptive mechanism and context manager, which can make up for the shortcoming of TAO. We also added QoS mechanism into TAO.
- For context-awareness, we gave an approach to describe context information. A three tuple was applied for context resources, and a XML description was used for deployment in applications.
- We also proposed an adaptive mechanism, introduced the process of adaptation of middleware. An adaptation strategy based on particle swarm optimisation (PSO) was implemented in our model.

The rest of this paper is organised as follows. In Section 2, we introduce the design principles of our middleware platform. Section 3 describes the details of framework structure. Section 4 presents context description in our framework. And the process of adaptation is shown in Section 5. Finally, we conclude the whole paper in Section 6.

2 Principle of design

In pervasive computing, resources are inadequate and changing. If we want to let pervasive computing offer reliable performance required by the current workload, it is necessary for the application to have the ability of self-adaptation. QoS is widely used strategy for meeting the requirement in application. Traditionally, QoS mechanism has been added to application as a logic function, which is easy to confuse QoS with function attribute of applications. This will increase the complexity of the application, increase the difficulty of management, and weaken the reuse of the application. Hence, we set QoS into middleware layer, making the QoS separated from the application, to ensure the reconfiguration at runtime.

Component technology can improve the reusability of software, reduce the cost of software development, and enhance its reliability and the maintenance of the application. Component technology is widely used in software development. Service-oriented architectures has become an incontrovertible paradigm for the development of applications in pervasive computing environments, as they enable publishing and consuming heterogeneous networked software and hardware resources (Ben Mokhtar et al., 2006). In this paper, the component technology and service combination are adopted in our middleware model.

Our middleware model is an open, adaptive, component-based, service-oriented architecture. It has QoS management component, and is able to support different levels of QoS requirements. There are four requirements in the middleware architecture in order to adapt to the development and application of pervasive computing. Therefore, a good middleware platform needs to have the following characteristics.

- *Component-based services:* to simplify the development and maintenance of middleware. This is also the main way to solve the reusability, readability, and cost of software development.
- *Independence:* includes the various steps of the development process (design, deployment, running). The independence also means that the functional attributes and QoS are independent of each other.
- *Openness:* the components of middleware services and applications can be deployed or replaced according to the requirement. This is good for adding new functions or removing obsolete functions easily.
- *Context awareness:* the middleware should be aware of the changes in the context and make the appropriate processing. Context awareness is an essential element of pervasive computing middleware, especially for adaptive middleware.

In this paper, we introduce the concept of component service. Component service has the following characteristics:

- Component service has two forms: atomic component service, or composite component service.
- Atomic component service is independent, and it is implemented by atomic component. It is the smallest unit to be invoked by an application.
- Composite component service is composed of one or more atomic components, or composite component services else. There are not only functional dependencies among these atomic component service, but also QoS non-functional dependences.

In addition, we design the context model to adapt to the environment of pervasive computing. The model uses extensible markup language (XML) to describe. QoS informations are used as context profile.

3 Architecture of PAmiddleware

This section describes the structure of our proposed adaptive middleware. Firstly, we introduce the idea of design. Then, we improve the TAO middleware platform and propose an adaptive middleware. Our middleware platform support context-awareness and QoS, which can deal with the changing environments. Finally, we described the extension mechanism we proposed in detail.

3.1 Idea of design

This adaptive middleware we designed is called PAmiddleware (pervasive adaptive middleware), which provides services by the creation and combination of components. The logic type is used, whose configuration parameters and QoS requirements are specified by the users. As an abstract type, it has no specific implementation. PAmiddleware is responsible for positioning and instantiating the logical type, and also responsible for passing parameters between them. Application developers can use middleware only if they know the logical type of component and its configuration parameters in the middleware.

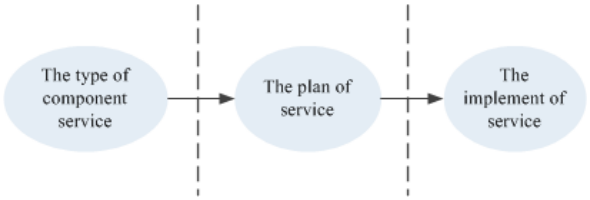
In the PAmiddleware, component service is divided into three steps (design, deploy, runtime phases), as shown in Figure 1. In the design, the type of component service is used. In the deployment, component service is described as a plan, which way is used to achieve the service type. In implementation of services, it implements the service, in accordance with the planning of the deployment phase. Application developers only need to specify the type of component service. The other is completed by

the middleware. This makes declarations, deployments, and operations separatedly, which is in line with the software development specification.

PAmiddleware is an open architecture, providing user interfaces and kernel interfaces. Its core services are composed of lifecycle management, adaptive management of service composition, and hooks for QoS mechanisms. When users want to get an application instance from the PAmiddleware, you should tell the QoS request to the PAmiddleware. The PAmiddleware makes a plan for your application. Then it constructs service according the plan. Finally, the PAmiddleware dynamically adjust the plan for the changing context.

Service, as a function unit, is composed by a component or more components. In this paper, we call it component service, and call component as service component. Service works as shown in Figure 1.

Figure 1 The deployment of component service (see online version for colours)



3.2 Architecture of TAO-based PAmiddleware

The following section describes the architecture of our proposed middleware. It also is an improved version of a middleware model that we have done before.

TAO is a freely available, open-source, and standards-compliant real-time C++ implementation of CORBA based upon the ACE. It attempts to provide efficient, predictable, and scalable quality of service (QoS). However, it does not support the context management and adaptive mechanisms of management. This cannot meet the requirements of pervasive computing. This paper extends the TAO for three aspects (as shown in Figure 2), including service mapping, adaptation mechanism, and context manager. These extensions make up for the lack of TAO, and improve its performance.

- *Service mapping mechanism:* mainly responsible for combination between the application and TAO, package, and classification. It maps from the service request to the service implementation. The service mapping mechanism is also a container to update, delete and add components.
- *Adaptive mechanisms:* responsible for the dynamic mediation of the application during operation. It includes the configuration of the service request, as well as the reconfiguration in the runtime. It also can order various service contracts, according to the context information.
- *Context manager:* responsible for management on context information, including acquisition, the integration, as well as classification and storage. We can create a different latch of context management so that we can flexibly manage context.

Figure 2 The architecture of TAO-based adaptive middleware

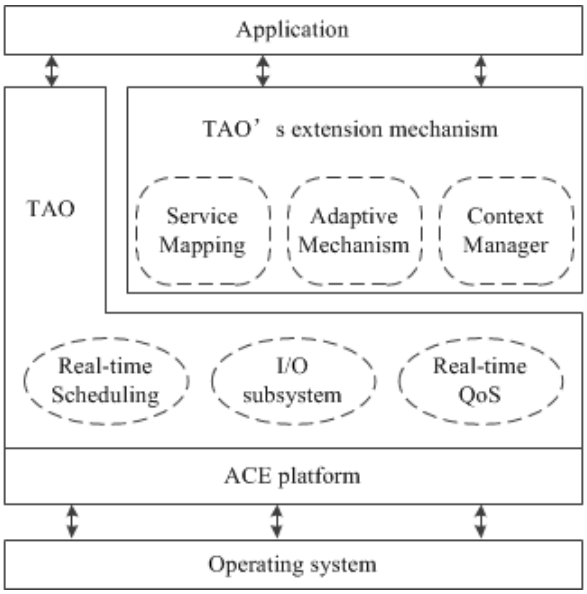
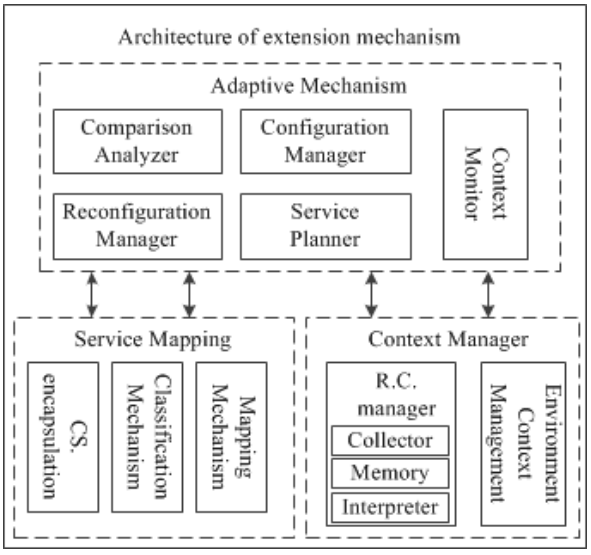


Figure 3 The architecture of extension mechanism



3.3 Architecture of extension mechanism

In this part, we mainly introduce the extension of TAO. The architecture of the extension is shown as Figure 3. The extension is composed of service mapping mechanism, adaptive mechanism and context manager. Context manager provides the same interfaces to the adaptive mechanism. The adaptive mechanism monitors the changing context, make a plan, and then take the plan to the service mapping mechanism. The service mapping mechanism is as a container, managing components for encapsulation, classification and linking.

3.3.1 Context manager

Context awareness is one of the key technologies in pervasive computing. It generally has three steps for context to gather, to storage and to model.

In this paper, resources and context manager are introduced, to accomplish the integration between physical space and information space, which shield the complexity. They provide unified interface of resources and environment context. According to different types of resource context information, it is divided into different types. Each type has its own context and resources bolt, to use in your applications. Collection, integration, and storage of the context in classifications are achieved by gatherers, translator, and memory.

a *Resource context manager:* mainly for management of the resource context (reference on OMG classifications and definitions of resources), including the storage and extraction of resource information, implementation of resource interface, and resources latch and management on creation and deletion.

b *Environment context management:* management of the context information, including collection, extraction, and integration for information, such as user's location, temperature, humidity, etc. At the same time, context manager should manage on link among different sensors, as well as on context of the implementation of the interface. Modelling and storage of context information also are realised by this part.

In order to manage and understand the context information of pervasive computing, the context can be divided into three levels, the context node layer, the logical abstraction layer, and the user layer.

a *The context node layer:* this is the bottom of pervasive resource management, involving various types of pervasive resource node physical, such as a single sensor node equipment, a variety of software services, this layer mainly refers to the management and control. Each resource node has attributes, such as the access capability of the access terminal, the display capability or the service function provided by the software service entity. Its main function is to collect the original data, and to quantify the original data.

b *The logical abstraction layer:* virtual resource layer. This is the middle level of pervasive resource management, the layer of the layer is the main context of the semantic representation of context information is analysed, abstracted can be provided to the user layer using the context information.

c *User level:* this is the highest level of pervasive resource management, involving the establishment of user service quality model, user preferences and user context and other functions. In this paper, the context information is classified according to the different application fields.

3.3.2 Adaptive mechanism

Shown as Figure 3, adaptive mechanism is composed of five parts. Adaptive mechanism as a core on pervasive computing, should regulate by itself according to the context, to ensure the user's QoS request. Firstly, when available resources increase, the service should improve the level. Secondly, when available resources decrease, on reduce the level of service of system. It includes the following sections.

a *Context monitor:* get and monitor the changes on context information, completion of the context acquisition in the service configuration, responsible for monitoring.

b *Service planner:* plan the service and get a reasonable service contract, according to the user s requirement.

c *Configuration manager:* accomplish service configuration, according to service contract which is generated by service planner.

d *Reconfiguration manager:* adaptation reconfiguration on mediation process.

e *Comparison analyser:* in reconfiguration stage, get a new contract, analyse similarities and differences between old service contract and the new one, get a improved component service.

3.3.3 Service mapping mechanism

Service mapping mechanism is the link between extension mechanism and TAO platform. It is responsible for the classification and registration of component service. And it is also responsible for encapsulating TAO components. It mainly includes the following parts:

a *Component service encapsulation mechanism:* encapsulate a variety of component service as a unified format to use.

b *Classification mechanism:* reclassify the TAO's component depending on application behaviour, in order to fit for adaptive mechanism.

c *Mapping mechanism:* responsible for conversion from component type to component implementation. The component type is mapped into the corresponding concrete component, so the function is realised.

Here, we need to understand the concept of component-based service. A service is a functional unit provided by the service provider to its customers, which is composed of one or more components. Component-based services have the following features:

a There are two types of component services: either an atomic component service or a composite component service.

b The function of atomic component service is independent, and its function is used by accessing the interface of atomic service component.

c The composite component service is provided by the combination of atomic service components. There are not only functional dependencies among these atomic component services, but also QoS non-functional dependencies.

In order to be dynamically allocated and assembled, the components of services should have the following features to provide good service.

- Using the method of aspect oriented programming (AOP) to provide functional and non-functional interfaces for outside users.
- Having a good message transmission mechanism. A good message sending and receiving mechanism ensures real-time communication between components.
- Having a self-description interface. By this interface, we can know the function, running state and resource utilisation of the component.
- Supporting heterogeneity. In distributed environments, applications are heterogeneous at different levels, which must be supported by adaptive components.

4 Context modelling and description

In this section, we mainly describe an approach of context modelling and a method of context description.

4.1 Context modelling

Context-awareness contains gathering, storage and modelling of context. Among the above, context modelling is the key of applying context resource. In order to fit pervasive computing, we divide context information into resource context and environment context. Resource context consists of delay, bandwidth of network, processor, memory and so on. Environment context consists of user information, location, time, temperature and other factors related to physical environment. QoS is as a profile of context information, including type of information, maximum, minimum, dimension, and so on.

To integrate with the implementation platform, context information should be modelled by uniform pattern. We adopted general resource model (GRM) defined by Object Management Group (OMG). GRM divides resource management into resource service and resource service instance. And resource service instance is an implement of resource service at runtime. Entity is a basic object in context description. Entity can be a physical entity or a logical entity. It is a physical device, component, service or others related to environment context and resource context.

4.2 Context description

In terms of pervasive computing, context refers to any information that can be used to characterise entity runtime state. The entity here can be an individual, a location, an object in physical or information space. Meanwhile, it can also be a virtual entity, such as software, network connection, social relations and so on. The variety of context information makes the context difficult to use. The key to solve the problem of context aware service is how to use a unified description method to describe the context and how to interact the context information (Da et al., 2011). So, a unified context description method is necessary, which is also beneficial to the classification and storage of context information. Context description must follow the principle of simplicity and flexibility.

In this paper, the context information is described as a three tuple $CP = (SN, PR, QoS)$, where SN is the only identification of context; $PR = \{P_1, P_2, \ldots, P_n\}$ is composed of a series of $P_i = (A, T, V)$, where A is the attribute name, T is the attribute type, V is the attribute value, $QoS = \{Q_1, Q_2, \ldots, Q_m\}$ is a series of $Q_i = \{T, Max, Min\}$, where T represents the aspect of QoS concern, Max represents the maximum resource provided, Min represents the minimum value provided by the resource.

Extensible markup language (XML) is a source language that allows users to customise data tags. It has become the factual standard of data exchange over internet for its platform and network independence. Besides, XML has good readability and maintainability. XML is also an effective tool for dealing with distributed structure information. So, we adopt XML to describe our context information.

The following is an example of context information described by XML language. It is a video stream system which contains network bandwidth, CPU-occupy, QoS and so on.

$<entity\ name = \text{“}videostream\text{-}one\text{”}>$
$<contexts\ name = \text{“}context1\text{”}>$
$<resource\text{-}contexts>$
$<resource\text{-}context\ name = \text{“}Network\_Bandwidth\text{”}$
$value = \text{“}100\text{”}\ unit = \text{“}MB\text{”}/>$
$<resource\text{-}context\ name = \text{“}CPU\text{-}occupy\text{”}$
$value = \text{“}0.5\text{”}\ unit = \text{“}s\text{”}/>$
$<resource\text{-}context\ name = \text{“}MemoryFree\text{”}$
$value = \text{“}560\text{”}\ unit = \text{“}MB\text{”}/>$
$<resource\text{-}context\ name = \text{“}MediaType\text{”}$
$value = \text{“}MPG\text{”}/>$
...
$</resource\text{-}contexts>$

<environment-contexts>
<environment-context name = "location">
<location positionX = "100" positionY = "200"
unit = "km"/>
</environment-context>
</environment-contexts>
<resource-QoSs>
<resource-QoS name = "Network"
max = "20" min = "5" unit = "MB">
...
</resource-QoSs>
...
</contexts>
...
</entity>

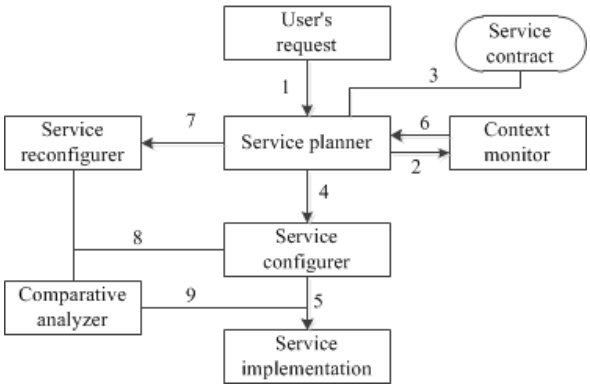
5 Adaptive mechanism of PAmiddleware

In this section, we propose an adaptive strategy for pervasive computing middleware. We put forward the concept of service contract. We use PSO algorithm to optimise the service composition.

5.1 Process of the adaptation

Adaptive mechanism is the core of adaptive middleware, which indicates the manner in the configuration. In the implementation of adaptation, service planner plays an important role. It receives the user's request, and checks the context information. Then, it gives a plan of service, and generates service contract. The planner also has a link with service configurator and service reconfigurer. Together, they complete the adaptation.

Figure 4 The process of the adaptation



In this section, service contract is an important concept, which is a platform-independent specification, including configuration information, environment dependencies, and QoS features. A service contract includes four information elements, service name, service implementation, parameter configuration, context-dependency.

Another concept we need to know is service type, which is the interface of adaptive middleware to be used by application developer. When using services, applications only need to specify the service type and parameter information.

When an application uses the middleware service, the service type and user's QoS request as the request are sent to the adaptive middleware layer (PAmiddleware), and finally achieve the service planner. Service planner according to the environmental context-dependency and the current context of the resources generates the configuration information which meets the QoS requirements of the user. The entire workflow is as shown in Figure 4.

1 User or application sends a request to the service planner, and requests the appropriate service.

2 According to the type of request, service planner gets context through the monitor. It also makes context monitor to oversee the change.

3 Service planner, according to the context information, generates service contract.

4 Service planner notifies service configurator to configure the service in accordance with the service contract.

5 In the operating mechanism, the component is instantiated.

6 If the context monitor oversees the context change, it notify service planner, which should compare the configuration information in the conflict in the current context and service contracts.

7 Service planner notifies service reconfigurer to complete the service reconfiguration.

8 Compare the old service contract and the new one.

9 Go to step 5, component service is instantiated.

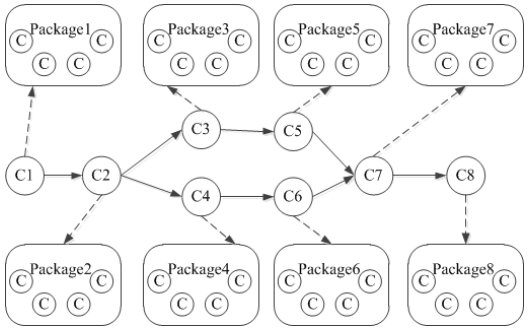
5.2 Component service planning method based on genetic algorithm

It can be seen from Section 3.1 that service planning is required from component service type to component service implementation. Service planning is to select suitable components from many component packages and combine them into an integrated service under various restrained conditions. We need to transform composite component services into atomic component services. In order to obtain the best solution, we adopted genetic algorithm to service planning.

In component library, each atomic component implements an independent function. And atomic components with the same function but different parameters are assigned to the same package. The role of genetic algorithm is to determine the appropriate components in each package according to the user's request. Taking

Figure 5 as an example, a composite component service consists of eight atomic component services, each of which comes from a different package. We will implement service planning according to different QoS requests. Firstly, we need to encode the problem, which transforms the feasible solution space of the problem into the search space that genetic algorithm can handle. In this paper, we adopt binary code to encoding genes, which is more suitable for coping with discrete problems. Suppose $N = max n_1, n_2, ..., n_m$, where n_i denotes the number of components in package i, and N is the maximum value in all packages. The gene code should be $q = [\log_2(N-1)] + 1$. if $N = 16$, then $q = 4$. The coding of component is from 0000 to 1111. For Figure 5, a chromosome is composed of eight genes.

Figure 5 The case of composite component service



Another key point of genetic algorithm is to choose an appropriate fitness function, which can measure the quality of the solution. In this paper, we mainly choose the service with the best overall performance according to the non-functional attribute of each component service. Suppose component service S is composed of n components $\{c_1, c_2, ..., c_n\}$, each component contains m resource context attributes, and their QoS characteristics are represented by $\{Q_1, Q_2, ..., Q_m\}$. We can get the objective function f of each resource as equation (1).

$$f = \sum_{i=1}^{m} (w_i \times Q_i) \tag{1}$$

where w_i denotes the value of QoS feature, $0 \leq w_i \leq 1$, $\sum_{i=1}^{m} w_i = 1$. And Q_i denotes the value of the i^{th} QoS feature. The fitness function should be equation (2).

$$F = F_{reqirement} - \sum_{i=1}^{N} (f_i) \tag{2}$$

where $F_{reqirement}$ denotes what user requires, f_i denotes value of the i^{th} component. There are different orders of magnitude and units due to different features, such as processors, network conditions, memory, etc. They must be standardised to ensure their fairness in service. Hence, we adopted max-min normalisation method to transform each feature.

6 Conclusions

In this paper, we take the process of soft development into account. We proposed an improved TAO middleware platform for pervasive computing named PAmiddleware, which contains type of component service, plan of service, and implementation of service, corresponding to three stages, designing, deployment, and running, respectively. We described the extension mechanism of TAO in detail. And we gave a context description method by XML language. Also the process of the adaptive mechanism was presented. A component service planning method was proposed using genetic algorithm. We take a video streaming service as an example to build a prototype system. The prototype system can adapt to the changes of scene and network speed, and get better video transmission.

The next important job is to improve the genetic algorithms in the configuration and reconfiguration mechanisms or adopt other intelligent optimisation algorithms. Meanwhile, it is necessary to adopt ontology technology for modelling context information, which is suitable for supporting the reasoning services in the scope of pervasive computing applications, increasing the expressiveness of context information and providing support for reasoning.

Acknowledgements

The project is funded in part by the National Institutes of Health, under Grant No. 5R01CA136535. This research is funded in part by the Science and Technology Project for the Universities of Shandong Province, under Grant No. J18KB171, Open Research Fund of Shandong Provincial Key Laboratory Of Infectious Disease Control and Prevention, Shandong Center for Disease Control and Prevention, under Grant No. 2017KEYLAB01, Discipline Talent Team Cultivation Program of Shandong Women's University, under Grant No. 1904, Shandong Women's University High level scientific research project Cultivation Fund, under Grant No. 2019GSPGJ07, Key projects of Education Department of Shandong Province, under Grant No. C2016M058.

References

Agirre, A., Parra, J., Armentia, A., Estévez, E. and Marcos, M. (2016) 'QoS aware middleware support for dynamically reconfigurable component based IoT applications', *International Journal of Distributed Sensor Networks*, Vol. 12, No. 4, p.2702789.

Al-Khawaldeh, M., Chen, X., Moore, P. and Al-Naimi, I. (2019) 'Knowledge-based auto-configuration system using ubiquitous robotics for services delivery in smart home', *International Journal of Embedded Systems*, Vol. 11, No. 2, pp.182–199.

Becker, C., Handte, M., Schiele, G. and Rothermel, K. (2004) 'PCOM – a component system for pervasive computing', *Proceedings of theSecond IEEE Annual Conference on Pervasive Computing and Communications*, pp.67–76, IEEE.

Belcastro, L., Marozzo, F. and Trunfio, P. (2019) 'A scalable middleware for context-aware mobile applications', *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 31, No. 2, pp.112–122.

Ben Mokhtar, S., Kaul, A., Georgantas, N. and Issarny, V. (2006) 'Efficient semantic service discovery in pervasive computing environments', *Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, pp.240–259, Springer-Verlag New York, Inc.

Beni, E.H., Lagaisse, B. and Joosen, W. (2019) 'Infracomposer: policy-driven adaptive and reflective middleware for the cloudification of simulation & optimization workflows', *Journal of Systems Architecture*, Vol. 95, pp.36–46.

Chollet, S., Lalanda, P. and Escoffier, C. (2015) 'Extension of service-oriented component models for dynamic environment', *2015 IEEE International Conference on Services Computing*, pp.648–655, IEEE.

Da, K., Dalmau, M. and Roose, P. (2011) 'A survey of adaptation systems', *International Journal on Internet and Distributed Computing Systems*, Vol. 2, No. 1, pp.1–18.

Djeddar, A., Bendjenna, H., Amirat, A., Roose, P. and Chung, L. (2017) 'Context-driven composition for mobile applications: a metamodelling approach', *International Journal of Embedded Systems*, Vol. 9, No. 6, pp.505–522.

Eibel, C., Do, T-N., Meißner, R. and Distler, T. (2018) *EMPYA: An Energy-Aware Middleware Platform for Dynamic Applications*, Friedrich-Alexander-Universitat Erlangen-Nurnberg, Dept. of Computer Science, Technical Reports, CS-2018-01, January.

Escoffier, C., Chollet, S. and Lalanda, P. (2014) 'Lessons learned in building pervasive platforms', *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pp.7–12, IEEE.

Forkan, A., Khalil, I. and Tari, Z. (2014) 'CoCaMAAL: a cloud-oriented context-aware middleware in ambient assisted living', *Grid Computing*, pp.114–127.

Freitas, C.F., Meireles, A., Figueiredo, L., Barroso, J., Silva, A. and Ramos, C. (2015) 'Context aware middleware in ambient intelligent environments', *International Journal of Computational Science and Engineering*, Vol. 10, No. 4, pp.347–358.

Garlan, D., Cheng, S-W., Huang, A-C., Schmerl, B. and Steenkiste, P. (2004) 'Rainbow: architecture-based self-adaptation with reusable infrastructure', *Computer*, Vol. 37, No. 10, pp.46–54.

Jaeger, M.A., Parzyjegla, H., Mühl, G. and Herrmann, K. (2007) 'Self-organizing broker topologies for publish/subscribe systems', *Proceedings of the 2007 ACM Symposium on Applied computing*, pp.543–550, ACM.

Kakousis, K., Paspallis, N. and Papadopoulos, G.A. (2010) 'A survey of software adaptation in mobile and ubiquitous computing', *Enterprise Information Systems*, Vol. 4, No. 4, pp.355–389.

Knappmeyer, M., Kiani, S.L., Reetz, E.S., Baker, N. and Tonjes, R. (2013) 'Survey of context provisioning middleware', *IEEE Communications Surveys & Tutorials*, Vol. 15, No. 3, pp.1492–1519.

Mukherjee, A., Dey, N. and De, D. (2020) 'Edgedrone: QoS aware mqtt middleware for mobile edge computing in opportunistic internet of drone things', *Computer Communications*, Vol. 152, pp.93–108.

Nocera, F., Mongiello, M., Parchitelli, A., Di Sciascio, E. and Patrono, L. (2019) 'A model for reflective middleware based on fuzzy rule for context-awareness injection in ubiquitous computing environments', *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp.1–7, IEEE.

Ouedraogo, C.A., Medjiah, S., Chassot, C. and Drira, K. (2018) 'Enhancing middleware-based IoT applications through run-time pluggable qos management mechanisms. application to a oneM2M compliant IoT middleware', *Procedia Computer Science*, Vol. 130, pp.619–627.

Portocarrero, J.M., Delicato, F.C., Pires, P.F., Costa, B., Li, W., Si, W. and Zomaya, A.Y. (2017) 'RAMSES: a new reference architecture for self-adaptive middleware in wireless sensor networks', *Ad Hoc Networks*, Vol. 55, pp.3–27.

Sun, J. and Satoh, I. (2016) 'Theory and implementation of an adaptive middleware for ubiquitous computing systems', *Journal of Information Processing*, Vol. 24, No. 6, pp.878–886.

Vahdat-Nejad, H., Zamanifar, K. and Nematbakhsh, N. (2013) 'Context-aware middleware architecture for smart home environment', *International journal of smart home*, Vol. 7, No. 1, pp.77–86.

Weyns, D., Iftikhar, M.U., De La Iglesia, D.G. and Ahmad, T. (2012) 'A survey of formal methods in self-adaptive systems', *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, pp.67–79, ACM.