A Role-based SOA Architecture for Community Support Systems

Bin Xu¹, Xiaohu Yang¹, Yuanhong Shen¹, Shanping Li¹, Albert Ma²

¹College of Computer Science & Technology, Zhejiang University, Hangzhou 310027, China
binxu@ieee.org, yangxh@zju.edu.cn, yhshen@zju.edu.cn, shan@cs.zju.edu.cn

²State Street Corporation, Boston MA 02111, USA
amma@statestreet.com

ABSTRACT

Software architecture is the backbone of a software-intensive system, many architecture models and styles are struggling to make the artifacts more understandable and reusable. Service-oriented programming is proposed to support reusing and enhancing distributed system development and a service-oriented architecture is essentially a collection of services which communicate with each other. However, the practitioners face the trouble in defining the services and identifying the communication request between all the services. Here in this paper, we adopted a role based architecture model named E-CARGO to facilitate the service definition and communication request identifying. An experience was conducted in prototyping a community support system and E-CARGO model was extended with data and authority access in the case study. The case study indicated that the suggested model could facilitate communication request identifying and service definition and could be helpful in identifying the authority control request during role shifting.

KEYWORDS: Service oriented programming, service oriented architecture, role based architecture, role based service oriented architecture, prototype experience.

1. INTRODUCTION

Software architecture is very important as it forms the backbone for successful software-intensive systems. It is the primary carrier of a software system's quality attributes such as performance and reliability. The architecture which correctly designed to meet its quality attribute requirements, clearly documented, and conscientiously evaluated - is the linchpin for software project success. Those incorrectly designed architectures always resulted in disaster [10-11].

When we learn from the evolution of software architecture model (from structure programming to object oriented programming, to component base development, and to service oriented development), we can find out that the different architecture model and styles are struggling in their way to make the software artifacts understandable and reusable.

While software architecture design is still a big problem to enterprises, people realized that software architecture design should be the collaboration work between the stakeholders. Furthermore, not a few people recognized that the design itself should deal with the inner collaboration between different objects, different scenarios, and different procedures. As so, service oriented method and role based method have been suggested to facilitate the software architecture design work.

A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services. A service-oriented architecture is essentially a collection of services and these services communicate with each other.

Figure 1. Basic Service Oriented Architecture [1]

Figure 1 illustrates the dual-way communication in a basic service-oriented architecture. A service consumer at the right sends a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. A service provider can also be a service consumer [1].
Service-oriented programming (SOP) [Figure 2] is proposed to support reusing and enhancing distributed system development [3, 4]. SOP originates from the concepts in object oriented and component-based development. It also expands these with distributed computing mechanisms [5].

Role-based collaboration model E-CARGO [Figure 3] is introduced by Zhu in [6, 7] to establish the development/business environment as a role net. Each role provides a certain services and applies a certain services in the proposed role net. Role based collaboration and its kernel mechanism were introduced in [8]. E-CARGO is helpful to build a more collaborative system, e.g. roles can be regarded as agent dynamics in multi-agent systems [12]. The role transferability in emergency management systems has also been introduced in [13]. The role model, the cooperative process in the system is described with roles. A user who wants to take part in the cooperation in the system must take on some responsibilities and fulfill some requirements specified by a role, so a role is formally defined as an entity consisting of a set of required permissions, a set of granted permissions, a directed graph of service invocations, and a state visible to the runtime environment but not to other users [12]. Obviously, different roles in the system differ a lot in their service permissions, rights of access, service invocations and so on.

In order to encourage the expertise contribution in State Street Zhejiang University Technology Center, global collaborative software development team in the R&D department is trying to develop a community support system. With the support from the senior management, the development team has begun to architecting for the prototype. Before adopting Service-Oriented Architecture model for the prototyping, we tried the E-CARGO model to define the services and identify the communication requests between the services. The E-CARGO model was extended with data and authority access features. The experience showed that the extend model benefit the design work as it clarified the collaboration of the roles, the relationship of the scenarios and the coupling of the classes. With the extended model, the classes were cataloged into unauthorized-role, authorized-role and objects. As a result, suitable test planes were made according to the difference. In this paper, the comparison is also made between the suggested model and object oriented model.

Our main contribution in this paper is the extension of role based architecture model and adoption of role based architecture model in the initial phase of service oriented architecture. The practice introduced in this paper can also bring some insight to the people are dealing with large complicated system designing.

The rest of this paper is organized as following. Section 2 presents the extension of E-CARGO and Section 3 introduces the experience in prototyping a community support system. The comparison between our suggested model and object oriented model will also be stated. The conclusion and discussion are made in Section 4.

2. EXTENSION OF E-CARGO WITH DATA AND AUTHORITY ACCESS

On the basis of E-CARGO model proposed by Zhu in his papers, we extend the role net to facilitate the design work in SOA [Figure 4].

The detail extension includes:
- Authorized role and non-authorized role,
- Data reference from service,
- Service to service reference,
- Outside service to other system, and Outside service reference to other system.
2.1. Authorized Role and Non-authorized Role

When users are taking part in a system for collaboration, their roles are changeable and flexible, so role transferability and collaboration is a basic requirement for a collaborative system. One of the major problems in current collaborative system is how to specify and define role while maintaining flexibility when roles are collaborating and transferring for different requirements [13]. The other is how to guarantee system security and information confidentiality when roles are transferring in response to complex events, for we know role transfer is a complex event, e.g. one role’s change might initiate a series of role transfers.

Service can be accessed independent of the platform, but the difference of the platforms and requirements should be considered during the definition and implementation of a service. In the prototype, the visitors can view some public information such as the vision of the collaboration or the top stars within the contributors. Such services don’t need critical security protecting. There are some other functionalities enabling someone publish projects or tasks. Since such information contains business secret, the security protecting should be established. Such difference brings different cost in preparing the environment including hardware and software equipments. Besides, the security protecting will definitely impact the system operating as high security requirement results in complicated operating.

In the case study, we extend the E-CARGO model with authority access. We catalog the roles into authorized roles and non-authorized roles. The non-authorized role, such as Role 0 in Figure 4 can be low security protecting while authorized role such as Role 1 or Role 2 in Figure 4 should be high security protecting. All the non-authorized roles can be gathered together into non-authorized area. While at the other hand, all the authorized roles can be gathered together into different authorized areas with different security requirements. The role shift from Role 0 to Role 1 or Role 2 should be authorized by high security components. Furthermore, when the authority information of a role has been changed, there should be some mechanism to re-validate the role existence in the high security area.

2.2. Data Reference from Service

In the E-CARGO model, specifying role is to collect information to specify outgoing messages (requests) and incoming messages (services) [14]. Inner data reference and outside data reference from services are in the process of service specification and definition from users’ requirements. Many issues also need to be considered for the data reference from service including data modeling, data sharing, data integration and data management which are key elements of optimizing service implementation for the software architecture. Besides, the technology of data analysis and data mining is also to be considered which can strongly support the process of service definition.

Before adopting SOA in prototyping the community support system, the team tried to define the services with minimized coupling. Therefore, the data reference is identified in the extended E-CARGO model. With such data reference, the data coupling can be clarified and the coupling of services can be reduced easily.

2.3. Service to Service Reference

E-CARGO model focuses on the role relationship by the request and service interface. In reality, one requested service may invoke a serial of other services automatically; one service may acquire other services responded by other role for its implementation; two services may depend on each other for their implementation. In the case study, we use service reference to specify such relationship. For example, in the human resource system, one user who requests the service of registration must invoke security control service provided by the system and acquire service of acceptance by the administrator. That’s two kinds of service reference.

Service to service reference can be helpful to identify the possible reuse in implementing the services. Besides, service to service relationship modeling is very important to maintain system flexible, collaborative and with high performance. For example, if one service is too frequently requested and called, it should be carefully designed so as
to reduce the possible influence to the system performance.

2.4. Outside Service to Other System and Outside Service Reference to Other System

A system built with a set of services can also be regarded as a super-service. While at the same time, a system can be regarded as a role when it interacts with outside environment. A system may provide some services to other systems and access some service provided by the other systems. In order to clarify the interface requirements, the outside service to other system and outside service reference to other system is identified in our case study.

Such identification clarifies not only the service operations but also the security and performance requirements. The service operating protocol, security model, and network requirements are identified.

When the inside service needs service reference from outside, the inside role can send messages to tell the outside role to follow when producing or customizing the outside service to support the application. For the implementation of the system, parts of the system may transfer to the other environment with new roles to implement the service providing and requesting, i.e., database (part of the system for role’s implementation) transfer and remote access, the database’s rights of access, service invocations in the new environment may also change(role change).

However, the service relationship in different environment is more complex, more issues need to be considered in our future research:
1. How to define and specify service interface and service agreement for different environment;
2. How to define and specify new roles when parts of the system transfer;
3. How to realize service security and information confidentiality, i.e., for a remote provider service, it may use an explicit defined boundary to separate the service interface to be access and confidential code.
4. How to send and receive message in service reference across different system environment

3. EXPERIENCE IN THE COMMUNITY SUPPORT SYSTEM PROTOTYPING

In our case study, the community support system was designed using the Service-Oriented Architecture model. However, we believed that Role based Architecture model will be very helpful to identifying the collaboration requirement within the application. Therefore, we extended E-CARGO model and adopted it before the architecting.

3.1. Background of the Case Study

Community Collaboration Contribution (3C) system is part of Global Collaborative Software Development Framework aimed to improve the efficiency of resource allocation, reduce the risk of resource lacking and optimize resource during the project life cycle in nowadays flourishing software engineering communities.

There are three kinds of actors in the system, administrators who maintain the system and control security of users, managers who publish tasks and employees who request and finish tasks. After registration by administrators, managers publish tasks with certain kinds of attributes including main skills requirement, effort, budget, deadline, requesting expire time and so on owning to a firstly published project. Employees can contribute their free time, intelligence and capabilities through requesting tasks, finishing them and getting the payment. Each employee in the system is assigned a changeable role in form of their working experience, skills, capabilities, working group and so on when registration. Usually, there are several employees who request the same task in a project, thus we define a match algorithm to assign task to a most suitable employee focus on the task attributes and employee role to realize the resource optimizing. When the employee finishes the matched task, he delivers it to the manager for acceptance. If the manager’s published task cannot be matched by any employee or he is unsatisfied with the delivered task, he can republish the same task (can change some of the task’s attributes) or reopen the task (send it to the matched employee for the second time). Managers after login can view their status and detail of their published task; employee after login can view details of published, republished and requested task. The state transfer diagram is shown as Figure 5.

Figure 5. State Transferring Diagram
3.2. Adoption of the Extended E-CARGO Model

The adoption of extended E-CARGO aims to easily specify each object, scenario and procedure based on different users’ operation environment, make service driven loosely-coupled software architecture so as to reuse the logical models independent of their platform and development environment, accelerate developing speed and benefit the software extension. The whole chart of the prototype is shown in Figure 6.

The adoption was made with several steps:

- Specifying the roles
- Identifying the role relationship
- Identifying the service interface
- Specifying the data reference

There was no critical requirement for the sequence of the steps. However, step 1 is suggested to be conducted in the first. After that, the following steps can be performed in a few loops until all the relationships and references have been clarified.

Table 1. Role Specification

<table>
<thead>
<tr>
<th>Role names</th>
<th>Requesting service name</th>
<th>Providing service name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest</td>
<td>1. Register</td>
<td>New user control</td>
</tr>
<tr>
<td></td>
<td>2. Login</td>
<td></td>
</tr>
<tr>
<td>Administrator</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. Accept task</td>
<td>1. Login</td>
</tr>
<tr>
<td></td>
<td>2. Reopen task</td>
<td>2. Logout</td>
</tr>
<tr>
<td></td>
<td>3. Publish task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Republish task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Reject task</td>
<td></td>
</tr>
<tr>
<td>Manager</td>
<td>1. Deliver task</td>
<td>1. Accept task</td>
</tr>
<tr>
<td></td>
<td>2. Request task</td>
<td>2. Reopen task</td>
</tr>
<tr>
<td></td>
<td>3. Publish task</td>
<td>3. Publish task</td>
</tr>
<tr>
<td></td>
<td>4. Republish task</td>
<td>4. Republish task</td>
</tr>
<tr>
<td></td>
<td>5. Reject task</td>
<td>5. Reject task</td>
</tr>
<tr>
<td></td>
<td>6. Deliver task</td>
<td>6. Deliver task</td>
</tr>
<tr>
<td></td>
<td>7. Request task</td>
<td>7. Request task</td>
</tr>
<tr>
<td></td>
<td>8. Match task</td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td></td>
<td>1. Accept task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Reopen task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Publish task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Republish task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. Reject task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Deliver task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. Request task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8. Match task</td>
</tr>
</tbody>
</table>

3.2.1. Specifying the Roles

Some new roles are defined by the outgoing message specified by identifying requesting service for the role’s rights and the incoming message specified by identifying providing service for the role’s responsibilities. Service is not the same as incoming message for the role, but defined as a new entity or resource for the software process. We can then easily specify the role based on the requesting service name and providing service name. The role specification in 3C system is listed in Table 1.

3.2.2. Identifying the Role Relationship

Based on the specification of the roles, we need to describe the relationships among the roles, such as classification, promotion, request/provide, transferability and confliction [15], the collaborative environment should also be considered. In our system, we consider three kinds of role, role staying in non-authorized area, role in authorized area and the role staying in boundary of the two areas [Figure 7].

For the role relationship design, besides role coordination in the form of Requesting and providing services, we import role transfer concepts in our role-based SOA model to facility the human users’ performance in the system. If one role wants to coordinate with the other role in different area, it must first transfer its own role; we also know that only the role performed by the human users (guest role, manager role, employee role) may transfer in...
our system. The basic procedure for the role transfer in 3C system’s architecture design is as follows:

1. Separate the collaborative area into unauthorized area and authorized area, specify the two areas;
2. Set some of the specified role into the two area called authorized role and unauthorized role;
3. Identify role transfer services between the two areas for the role transfer performed by the human users considering the different areas;
4. Re-specify the left roles in the area boundary: identify their providing service name for the role transfer between the unauthorized and authorized area;
5. Re-specify roles which are to transfer in the unauthorized and authorized area.

3.2.3. Identifying the Services Interface
Before we can specify the services, we should identify the service’s interface. On the basis of extended E-CARGO chart and the role specification, the service interface can be identified [Table 2].

Service interface specification table contains not only the interface between service and role, but also the service to service. As shown in Table 2, service “register” will access the service “New user control” and service “Request task” will access the service “Match task”.

Table 2. Service Interface Specification

<table>
<thead>
<tr>
<th>Service name</th>
<th>Request by</th>
<th>Provides by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>Guest</td>
<td>Security control</td>
</tr>
<tr>
<td>New user control</td>
<td>Register service</td>
<td>Administrator</td>
</tr>
<tr>
<td>Login</td>
<td>Guest role</td>
<td>Security control</td>
</tr>
<tr>
<td>Logout</td>
<td>1.Manager 2.Employee</td>
<td>Security control</td>
</tr>
<tr>
<td>Accept task</td>
<td>Manager</td>
<td>Task</td>
</tr>
<tr>
<td>Reopen task</td>
<td>Manager</td>
<td>Task</td>
</tr>
<tr>
<td>Publish task</td>
<td>Manager</td>
<td>Task</td>
</tr>
<tr>
<td>Republish task</td>
<td>Manager</td>
<td>Task</td>
</tr>
<tr>
<td>Reject task</td>
<td>Manager</td>
<td>Employee</td>
</tr>
<tr>
<td>Deliver task</td>
<td>Employee</td>
<td>Task</td>
</tr>
<tr>
<td>Request task</td>
<td>Employee</td>
<td>Task Match task service</td>
</tr>
<tr>
<td>Match task</td>
<td>Request task service</td>
<td>Task</td>
</tr>
</tbody>
</table>

3.2.4. Specifying the Data Reference
To simplify the data reference, there are 2 kinds of legends for data reference. One is similar as that of service provide and another is a new legend only for service to data reference. As shown in Figure 6, the EmpData object will be access by service “register” and service “new user control”. The service “new user control” will also access the AEmpData object. Role “Task” owns TaskData object which will be accessed by all the services provided by Role “Task”.

3.3. Discussion of the Extended E-CARGO Model Adoption
On the basis of extended E-CARGO model adoption, the communication request can be clarified during the specifying of roles and services, service-service reference and service-data reference. The service definition can be facilitated after the service interface specification, service-service reference and service-data reference.

The coupling and possible reuse can be indicated with the extended E-CARGO model. The case study indicated that the suggested model could facilitate communication request identifying and service definition and could be helpful in identifying the authority control request during role shifting.

4. CONCLUSION

Though Service-Oriented Architecture is prevalent these days, the practitioners face the problems in defining and organizing the services for the systems. In order to facilitate the SOA activities, the authors extended a role based architecture model E-CARGO suggested by Prof. Zhu and extended it with data and authority access to clarify the communication and coordination request in the final system.

In this paper, the extension of E-CARGO model has been presented in detail. The main contributions of the authors are the extension of E-CARGO and adopt it in the initial phase of SOA. A case study was conducted in prototyping a community support system. The adoption of extended E-CARGO before SOA design turned out to be fruitful. The communication of the role could be clarified easily and early and the project manager could identify the possible coupling of the software development work easily so that the tasks were well scheduled and performed.

Currently, extended E-CARGO in our research was performed manually. We will try to facilitate it by some utilities in further adoption.

5. ACKNOWLEDGEMENT

This work is part of “Global Collaborative Software Development” research project in State Street Zhejiang
University Technology Center, which is an attempt to improve the dual-shore software development with integrated best practice, software engineering technology and project management methodology. The research project is funded by State Street Corporation, USA. The project is collaboration between Zhejiang University, China and State Street Corporation, USA. All the company and product names are trademarks or registered trademarks of their respective owners.

The contents of this paper are the opinions and conclusions of the author only and do not necessarily represent the position of State Street Corporation or its subsidiaries, affiliates, officers, directors or employees.

The authors would like to express their special thanks to Professor Haibin Zhu for his wonderful comments and presentation improvement to this paper. Thanks to the team members including Shirong Lin and Xiejun Liang, their work made the experiments performed with more pleasure and fruitful. Special thanks to the anonymous reviewers, their precise and relevant comments and suggestion are very helpful for us when revising this paper.

REFERENCES


