A Framework for Enterprise Information Systems

Xi-Min Yang[1,2] and Chang-Sheng Xie[1]

[1] Department of Computer Science, Huazhong University of Science & Technology, Wuhan, China
[2] College of Computer Science, South-central University for Nationalities, Wuhan, China
yangximin@scuec.edu.cn, cs_xie@mail.hust.edu.cn

Abstract. In this paper, we present that Enterprise information systems (EIS) can be abstracted to a scenario in which users could schedule the task suites of controlled entities under enterprise security mechanism, and propose a novel Entity Driven Task Software Framework (EDTSF) for EIS. Based on existing hierarchical information system architectures, the EDTSF could clearly implements enterprise business partition, reduces coupling between data objects and tasks; it also increases flexibility and expansibility of the EIS by expanding service function of data object and reusing the scheduling mechanism at system level. We have noticed that little research has been done on the partition rules and methods of enterprise business. The EDTSF is a new software framework and its application results show that the EDTSF is an effective approach to analyzing, designing and implementing EIS.

Keywords: enterprise information system, framework, controlled entity, enterprise businesses partition.

1 Introduction

Enterprise information systems (EIS) is a computer application which contains a set of interrelated components that collect, storage, manipulate, and disseminate data and information, and provide feedback [1]. Today, enterprise requires that EIS must have higher sensitivity or dynamic adaptability in order to respond their demand changes over time. These demands include the adjustment of workflow, the alteration of management model, the evolution of user needs, or even if the redesign of process. Therefore, EIS should be updated quickly and reconfigured dynamically. Especially, EIS is increasingly apt to networking and integrating trends, what has gained more and more attention is how to protect enterprise's investments effectively and take full advantage of enterprise information resources.

Over the past decades, Research on EIS has made a number of important advancement with the evolvement of software engineering methodology and database techniques, such as CBSD (Component-Based Software Development)[2], SOA (Service Oriented Architecture)[3,4,5], AOIS (Agent-Oriented Information Systems)[6,7], and AUP (Agile Unified Process)[8,9]; But sometimes, developers would need to face the limited factors at every phase of the EIS lifecycle, which will be discussed in section 2. Currently, the development efficiency and software reuse

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part I, HCII 2007, LNCS 4557, pp. 227–236, 2007.

are still two comparatively important aspects of the EIS research. According to the best of our knowledge, most of existing EIS development technologies focus on the research on the component description methods, the component composition technologies, and the connectors; but all of them are lacking in research on the partition rules and methods of enterprise business.

In this paper, we describe our approach to partitioning enterprise business. We pay much attention to how to apply the partition rules and methods of enterprise business to the hierarchical information system architecture; we also study approaches that can reduce the coupling of data and its processing procedures. As a matter of fact, all of (actual and logically defined) enterprise resources could be viewed as logic entities that are managed or controlled by operators. Therefore, we abstract EIS to a scenario in which users could schedule the task suites of controlled entities under enterprise security mechanism, and propose a novel Entity-Driven-Task Software Framework (EDTSF) for EIS of the existing hierarchical information system architectures. In EDTSF, it does not assign users to tasks for each application directly, instead, assigns users to entities and tasks to entities for each application. At the same time, data are only viewed as organizational form of enterprise information resource, and are degraded to the task's processing content. Any change in the user's duties and other requirements can be simply solved by assigning the user to another entity, reconfiguring the entity's tasks, and/or updating data processing procedures.

The remainder of this paper is organized as follows: Section 2 briefly analyzes some problems in the EIS development; Section 3 presents EDTSF, its architecture and framework development process; Section 4 compares EDTSF with other EIS development technologies; Section 5 gives a conclusion of this paper.

2 The Problem

Historically, the existing enterprise applications, especially those department applications that have been built at the different periods, usually are likely to be rather heterogeneous between them and even among one's components. The heterogeneities may exhibit themselves in the use of different techniques (i.e. programming languages, directive ideologies, and etc.), the availability on different hardware and operating system platforms and the use of different representations for the exchange of data[10~12]. So, the first task is the transplant of existing applications and their seamless integrating with each other. In the mean time, what must also be considered is how to furthest share and take full advantage of available enterprise resources, as well as how to protect enterprise's investments. This is also one of the reasons why the enterprise application integration (EAI) is a research hotspot at all times. Some of software development techniques (i.e. WebServices, SOA, ESB (Enterprise Service Bus) and Event Oriented Architectures) mainly focus on the approaches for exchanging data among applications, which could lead to increasing developer's effort in maintenance of data consistency and data integrality.

The methodology of CBSD has become one of major technologies used in today's EIS development and maintenance although it is yet to be mature. Most developers, especially primary ones, cannot gain practical advice in practice from CBSD except theoretic guidance. They must fill the gap with assumptions by themselves on the

architecture when the Framework or Architecture was developing. While components can speed up the EIS project, the interface and communication mechanism in different components must be considered and translated. However, the more details are hidden in component or numbers of components were embedded in the projects, the more difficulties could be faced in the application maintaining and updating period. Furthermore, one that remains to be settled is how to implement the mapping from the problems of application domain to the realization of components, frameworks, and/or architectures [2,13~16].

Today, most of the EISs are generally built based on RDBMS and OO (Object-Oriented) technique. Application developers have been faced with a task of persistent object, including its storage approaches and ability to provide a transparent and robust applying method for developers. Aiming at the complexity of storage systems and the changeability of information organization structure, there are multiple ways that an object can be persisted, such as object/relational mapping and object-oriented databases. But relational databases lack many of the constructs necessary for true object persistence, and the difficulty in storing object-oriented data into a relational database, known as the object-relational impedance mismatch problem, still has not been completely solved. Object-oriented databases have advantages over relational databases because they are actually composed of objects instead of tables. Unfortunately, database schema migrations due to changes in class definitions can be costly, and support for ad-hoc queries is typically not as good as it is with relational databases[17]. Other options(i.e. approaches to embed SQL commands into objects, specialized data objects or data middleware, and etc.) probably result in tight coupling between data and object in which data are processed, or restrict their fitness, or affect the application system's performance with their complexities increasing.

When OO technique is used in the EIS development, enterprise business is often expressed as methods of objects. It means that the partition of enterprise business is achieved by aggregating methods into related objects. However, this processing method not only could affects the rationality of enterprise business representation, but also meets in turn the influence on object design, methods used in data expressing and data manipulating, and so on. For instance, it is not a good choice in digital hospital software to put the "register" method into patient-object, and is not conducive to the patient-object's persisting representation. To date, some approaches, including dynamic menu and role-based enterprise business partition [18,19], are expansion towards traditional methods in applications.

3 Entity Driven Task Information System Framework (EDTSF)

3.1 The Method and Definition

Enterprise employee can be regarded as relatively stable management units, and every management task can be regarded as a series of management and decision-making activities. These activities, in essence, are the rational usage or scheduling of enterprise resources under the direction and control of specific management ideas and models. All of the management objects, such as personnel, commodities, capital, information and so on, are generally called as the enterprise resources [19]. For

example, in a hospital, the registrar's duty is management of registered evidence, and their work is to fill data items into registered evidence based on information what is patient provided. Therefore, all of actual and logically defined enterprise resources could be regarded as logic entities that are managed or controlled by operators. Thus, EIS could be abstracted to a scenario in which users could schedule the task suites of controlled entities under enterprise security mechanism. We then propose a novel framework for EIS named EDTSF based on this abstraction.

In EDTSF, enterprise information processing is performed through interactions among user, controlled-entity, task, and data-object. The first function of data-object is to describe the organization structure of enterprise information resources, which includes data items (name, type, length, etc.) and source (DBMS, tables or views, field list, agent-user info, etc.). The second function of data-object is to provide a consistent data accessing approach for other objects. The task is an all-encompassing procedure that collects, stores, manipulates, and disseminates information, and can result in properties change of controlled entity. To the goal, the task defines a set of data-object and the sequence of processing. One step is a single data processing function with format descriptions of input and output data. The controlled-entity is a real reflection to enterprise management object that could be an object container or an actual object. User's work is accomplished by gaining the management right of controlled-entity and scheduling tasks of controlled-entity. Finally, we define the EIS as following components:

- U, E, T, D, P, and R (users, controlled-entities, tasks, data-objects, processes, and rights, respectively);
- $UA \subseteq U \times E$, a many-to-many user to controlled-entity assignment relationship;
- $TA \subseteq T \times E$, a many-to-many task to controlled-entity assignment relationship;
- $PA \subseteq P \times T$, a many-to-many process to task assignment relationship;
- $DA \subseteq D \times E$, a many-to-many data-object to controlled-entity assignment relationship;
- $RA \subseteq U \times D \times R$, both user to data-object and data-object to right are many-to-many assignment relationships;
- $\forall p_i, \exists t_i e_i (t_i \in T \wedge e_i \in E \wedge (t_i, e_i) \in TA \wedge (p_i, t_i) \in PA), i = 1,2,3....,;$
- $user: P \to U$, a function mapping each process p_i to the single user $user(p_i) \in \{u \mid (u, e_i) \in UA\}$; and
- $data\_objec\ t: P \to 2^D$, a function mapping each process p_i to a set of data objects $data\_objects(p_i) \subseteq \{d \mid (d, e_i) \in DA\}$ and process p_i has rights $\bigcup_{d \in data\_objects(p_i)} \{r \mid (user(p_i), d, r) \in RA\}$.

3.2 Architecture

The architecture of EDTSF, shown in figure 1, consists of application layer and presentation layer and data layer from the top to the bottom, and each layer can be subdivided relying on the actual applications.

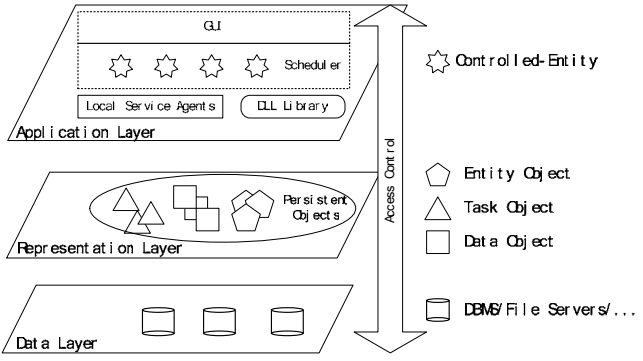

Fig. 1. The Architecture of EDTSF

Application layer: This layer consists of several function sub-layers: GUI, Controlled-entities centralized scheduler, local service agents and DLL library, represents all functions of the EIS. GUI is an interactive platform between user and application system. After a user logins, all controlled-entities managed by the user are shown in GUI's main window. Then the user can perform tasks by selecting one controlled-entity, scheduling its tasks, and browsing results. The Scheduler is a key component of application layer. After accepting user's instruction to execute one task of the selected controlled-entity, scheduler creates an instance of the controlled-entity and then sends a "start" message with one task's id to the controlled-entity. The local service agent is used for providing a uniform interface of presentation layer. The function of local service agent is transferring data access instructions of entities and tasks to presentation layer, and completing the inter-conversion of data format between presentation layer and objects in certain time. DLL library stores information of all data processing procedures, each procedure is relative to one processing step respectively. When receiving a request from the task object, DLL library automatically accomplishes finding and loading and running of a suitable procedure for the request.

Presentation layer: The key service of this layer is providing a presenting approach of persistence object for application layer. A group of associated or unassociated service objects, and coordinator objects, are expressed in this layer. For instance, data-object's duty is receiving requests from the application layer, creating connections to data sources, accessing data from these data sources, and converting data format to the objects required.

Data layer: This layer is a composition of various storage systems for enterprise, including local storage systems, remote storage systems, and others shared by existing applications or by cooperative organizations.

3.3 Development

The definition of controlled-entity and its task generally is stated as stability in EIS. Task's processing, workflow, and structure and format of data in task are easy to be

changed. It is most important in EDTSF development to design and implement pervasive mechanism for tasks scheduling and approaches for presenting data-objects flexibly, namely the designs and implements of application layer and presentation layer. Figure 2 shows a simple development model of EDTSF we used in hospital information system project. A new object named operator is introduced for reducing quantity of controlled-entity. Here, the operator is a virtual object that can get the relative controlled-entities together according to partition rules of enterprise business duty. One operator can be assigned to different users and a user can be an owner of multi-operators. Relative to changeability of user duty, operator could keep stability between itself and controlled-entity. Therefore, administrator's efforts can be significantly alleviated. System configure is a tool in which administrator can define multifarious objects and set relationships between them and grant or revoke user's rights.

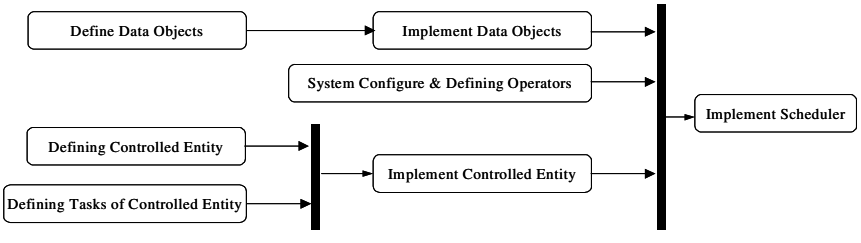

Fig. 2. Development Model of EDTSF

1) Definition of controlled entity

A clear characteristic of enterprise resources is that they would maintain stably in the certain time after they are vested in one department in enterprise, such as recipes and system users. At the other hand, one controlled-entity could also consist of other controlled-entities. To the goal, the concept of group is introduced, and one with the type of entity indicates itself being a controlled-entity, one with the type of group indicates itself being an entity group. So, we define controlled entity as follows:

```
Type = [Entity | Group];
CE = {ID, Name, Type, FGID, Icon, …, Description}.
```

Where, FGID is used to create hierarchical structure of controlled-entities, which is in response to organization form of enterprise resources. The value 0 indicates that the controlled-entity is a common entity, otherwise, it is one vested in a controlled-entity group.

2) Definition of data-object

The data-object's implement includes two missions. The first is the description of its data sources and each data's constituent parts. The question what should be considered in this part is the case of multi-data items combination. For the data what comprises multi-constituent parts, source of its each part and how to combine them all must be described. The second is to specialize data-object's applying properties, such as the form of output data (individual values or a dataset), the info of data item (i.e. name, display name, format, etc) and so on. Example 1 illustrates a definition of recipe data object based on XML.

Ex. 1. Partial definition of recipe data object based on XML

```
<DataObject ID=DO1011 name=Inpatient_recipe>
<Title>Inpatient Recipe</Title>
<Section name=Patient_info>
  <Item source=db_Patient Table=Inpatient_info>
   <Field=Patient_id>Inpatient ID</Field>
   <Field=Data_in format="yyyy-nn-dd hh:mm:ss">
    Enter Date
   </Field>
   ……
  </Item>
  <Item source=db_Patient table=Patient_info>
   <Field=Name>Name</Field>
   ……
  </Item>
</Section>
<Section name=Medicaments>
  ……
</Section>
<Section name=Physician_Info>
  ……
</Section>
<Section name=Finance_Info>
  ……
</Section>
</DataObject >
```

3) Implementation of controlled-entity

Known from the architecture of EDTSF, the key of implementing controlled-entity is task scheduling mechanism. A typical workflow of controlled-entity consists of three steps: data gaining, data processing and data storing, as shown in figure 3. The workflow's nonlinear execution would happen at time when data gaining and data storing need user's interactions to accomplish. Because EDTSF disperses these steps in different objects, the task scheduling mechanism is simplified to define the DLL exported function. For example, a exported function can be defined as follows:

```
Function Create(Task: TTask; OpID: Integer; UserID:
string): IStep;
```

Where Task is a task objects used by step process, OpID is a sequence number executed currently, UserID is user's identifier, and the function's result IStep is an interface definition. Methods the IStep interface included are:

```
IStep = interface(Iinterface)
   function RunStep: Boolean;
   function Abort: Boolean;
   ......
end;
```

Controlled-entity objects achieves its task scheduling by call methods of IStep interface implemented by task in relative event procedures.

4 Contributions

A kernel function in existing software architectures, such as SOA, CBSD and so on, is design enterprise business function module for reuse. Data representation middleware and Agent-based technologies are effective methods used to integrate generally heterogenic data source in enterprise. EDTSF, a hierarchical information system architecture employing entity driven task technology, combines the benefits of CBSD, SOA, AOIS and middleware technique in a manner that retain traceability and preserve correctness with respect to the original. EDTSF not only provides a method for implementing legible partition of enterprise businesses, but also decreases the coupling between data and data processing.

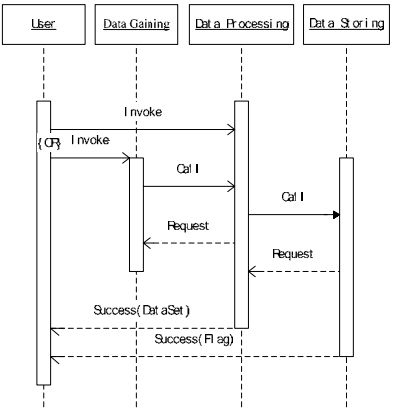

Fig. 3. Steps of a Single Task

Compared with other system developing technologies which we observed, the discussions on advantages of EDTSF as follows:

Using system configuration tool, EDTSF-based EIS can expediently plug-in/remove controlled-entities (include its tasks and the task's processes) and data-objects according to enterprise management demand change, such as business expansion and adjustment and so on. It means that EDTSF-based EIS can support enterprise business's plug and play. At the same time, application system's dynamic reconfiguration would be achieved too.

EDTSF has the universal flexibility because it implements reuse of the business partition approach and the task scheduling mechanism at system level, which is independent on the application environment. At the other hand, another one with distinction from component and service concepts in other technologies is the task object, which doesn't contain actual data it processed, only describes its required data and parses result back from data objects. The separation of task object and data-object decreases the coupling between them, and increases the responsibility of EDTSF in change on enterprise information structures and enterprise management demands. First, the recombination and the alteration of assignment for task object and data-object is easy-to-achieve. Second, if data description format doesn't change, the task object's modification cannot influence data-object, vice versa. If data description format changes, the only required to be modified is the parsing function that is global consistent. So, the application system would be continuously evolving tends to be prefect.

After introducing the concept of "controlled-entity", EDTSF could increase trustworthiness in enterprise business definitions, and make legible partition of enterprise business. In EDTSF, main form of the application system could be unified as a scenario in which controlled-entities are represented as a hierarchical structure based on the supply from enterprise resources management and the task scheduling approaches. Accordingly, EDTSF can reduce the complexity in traditional GUI such

as MDI and SDI, and can support interface control technologies (i.e. dynamic menu, dynamically form loading, and so on) as well.

EDTSF can support the many kinds of development models and methods. The design and implement of task scheduling mechanism are simpler than the design and implement of connectors, which reduces the difficulty in developing EDTSF framework.

From the development processing view, other technologies referred in this paper realize the customization of existed components and services module at various levels, EDTSF realizes the customization of system configuration and its emphases is definition of controlled-entity, which include assignments from tasks to them, and data-objects. EDTSF is only an information system framework, and doesn't produce new system modules, but it may be used as an effective approach to exporting business and data components or service units. This is not the same as CBSD and SOA, which could bring new components or services unit in applications.

Besides the above benefits, the most important characteristic is that EDTSF would be used repetitiously after its development process had been completed, which would enable information system developers to pay much attention to analysis and design of enterprise businesses, and the Rapid Application Development (RAD) would also be archived.

5 Conclusions

In this paper, we abstract information systems to a scenario in which operator could schedule task suites of controlled-entities under enterprise security mechanism, and propose a novel entity driven task information development framework, named EDTSF, based on this abstraction. In EDTSF, the concept of "controlled-entity" is introduced to reflect the organization structure of enterprise resource and management duty in truth. EDTSF makes enterprise business partition more clearly, which is lacking in other information systems development technologies. EDTSF can speed up the development processing of application by archiving the reuse of task scheduling mechanism at system level. As an effective approached to information systems analyzing, designing, and implementing, EDTSF has been successfully used in some projects such as digital hospital software, student information system, and so on. In future, we will do research on the following aspects of EDTSF: 1) approaches to simplify express the data object; 2) optimization on business scheduling mechanism; 3) formalized definition of layer structure and; 4) access control technique.

References

1. Stair, R.M., Reynolds, G.W.: Principles of Information Systems (2005)
2. Feng, C., Qianxiang, W., et al.: An architecture-based approach for component-oriented development. In: Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International, August 26-29, 2002, pp. 450–455 (2002)
3. Gold, N., Mohan, A., Knight, C., et al.: Understanding service-oriented software. Software(IEEE) 21(2), 71–77 (2004)

4. Chu, S.C.: From component-based to service oriented software architecture for healthcare. In: Enterprise networking and Computing in Healthcare Industry, HEALTHCOM 2005. Proceedings of 7th International Workshop on June 23-25, 2005, pp. 96–100 (2005)
5. Karhunen, H., Jantti, M., Eerola, A.: Service-oriented software engineering (SOSE) framework. In: Services Systems and Services Management, 2005. Proceedings of ICSSSM '05. 2005 International Conference, June 13-15, 2005, vol. 2, pp. 1199–1204 (2005)
6. Wagner, G.: Toward Agent-Oriented Information Systems. Technical report, Institute for Information, University of Leipzig (March 1999)
7. McDonald, J.T., Talbert, M.L., Deloach, S.A.: Heterogeneous DataBase Integration Using Agent-Oriented Information Systems
8. Stark, J.A., Crocker, R.: Trends in Software Process: The PSP and Agile Methods. IEEE Software, vol. 20(3), pp. 89–91
9. Alhir, S.S.: The Agile Unified Process (AUP), 2005, http://home.comcast.net/˜salhir/TheAgileUnifiedProcess.PDF
10. Emmerich, W., Ellmer, E., Fieglein, H.: TIGRA - an architectural style for enterprise application modelling. In: 23rd international conference on Software engineering, Toronto, Ontario, Canada, ACM Press, New York (2001)
11. Majumdar, B., Dias, T., Mysore, U.: ESB: A bandwagon worth jumping on (May 2006), http://www.infosys.com/Technology/bandwagon-worth-jumping-on.pdf
12. Mosawi, A.A., Zhao, L., Macaulay, L.: A Model Driven Architecture for Enterprise Application Integration. In: Proceedings of the 39th Hawaii International Conference on System Sciences – 2006, IEEE, New York (2006)
13. Succi, G., Pedrycz, W., et al.: Package-oriented software engineering: a generic architecture. IT Professional 3(2), 29–36 (2001)
14. Xu, W., Yin, B.L., Li, Z.Y.: Research on the business component design of enterprise information system. Journal of Software 14(7), 1213–1220 (2003)
15. Kirk, D., Roper, M., Wood, M.: Defining the problems of framework reuse. In: Computer Software and Applications Conference, 2002. COMPSAC2002. Proceedings. 26th Annual International, August 26-29, 2002, pp. 623–626 (2002)
16. Chaudet, C., Greenwood, R.M, et al.: Architecture-driven software engineering: specifying, generating, and evolving component-based software systems. Software(IEEE) 147(6), 203–214 (2000)
17. Fontaine, A., Truong, A.-T., Manley, T.: A Survey of Strategies for Object Persistence. CSci 5802 - Spring (2006)
18. Ming-Hui, Y., Qi, F., Xue-Guang, C.: A Role Interactive Model Based on Method of information System Function Partition. Science & Technology Progress and Policy 9, 105–107 (2004)
19. Yue-ting, C., Xiao-dong, Z., Jin, D.: Research on the reconstructivity of agile supply chain management system. Journal of Tsinghua University 40(3), 68–71 (2000)