Journal of Universal Computer Science, vol. 18, no. 17 (2012), 2493-2513
submitted: 16/1/12, accepted: 29/8/12, appeared: 1/9/12 © *J.UCS*

A Conceptual Ontology-based Resource Meta-Model towards Business-driven Information System Implementation

Hongming Cai
(School of Software, Shanghai Jiao Tong University, Shanghai, China
hmcai@sjtu.edu.cn)

Boyi Xu
(Shanghai Jiao Tong University, Shanghai, China
byxu@sjtu.edu.cn)

Fenglin Bu
(Shanghai Jiao Tong University, Shanghai, China
bu-fl@cs.sjtu.edu.cn)

Abstract: Enterprises need a flexible and configurable IT architecture to meet complex business requirements agilely. For the purpose of bridging business modelling in build-time and application configuration in runtime seamlessly, a Conceptual Ontology-based Resource meta-Model (CORM) is proposed. Firstly, a resource meta-model is built as a referred model to describe business elements and relationships. Then, by means of ontology, these business elements are transformed into IT service-oriented components such as SOAP services, RESTful services and BPEL files. Referring to Model-View-Controller pattern, these service components are then configured in a runtime supported environment. Next, a state space defined by a resource array is built as the control mechanism to realize a completed IT system. Finally, a CORM-based supported Platform is built to support business modelling, service transformation and system configuration. The research indicates a new approach to develop and implement enterprise information systems in a more flexible and configurable way.

Keywords: Conceptual model, Software Engineering, Information Systems, Ontology, Resource Oriented Architecture, Business Process, Service Composition
Categories: D.2.11, H.1.1, H.4.0

1 Introduction

Business process management (BPM) is applied to facilitate the development and implementation of Enterprise Information System (EIS) so as to recognize and execute business requirements agilely. Nowadays, the architecture of enterprise information system is changed from a programming, data-driven, carefully planned and designed system to a process-aware [Howard Smith, 03], service-oriented, and rapid assembling application configuration.

A major shortcoming of existing approaches is the lack of mechanisms to bridge build-time business models and runtime system components, which occurs inevitably with the purpose to develop and implement EIS according to business requirements.

The business models are quite different from executed system components, and logical structures of business processes are separated from detailed realization of business functions. Business process model generated from build-time stage doesn't always contain all the information to construct an executable information system in runtime time. Some semantic information such as associations of processes, tasks, roles and functions are lost when switching from form design stage to realization stage.

Therefore, a Conceptual Ontology-based Resource meta-Model (CORM) is proposed to design business process and build a configurable service-oriented system in a consolidated manner by means of bridging a business model definition and executable services. Correspondingly, based on CORM, a supported platform for business process modelling, and services configuration, is built to implement enterprise information systems.

The paper is structured as follows. Section 2 reviews previous works on process modelling and system configuration. Section 3 introduces the idea of business-driven system configuration based on CORM, which allows one to recognize business elements and transform them into service components. Section 4 presents a formal definition of CORM for describing business elements and their relationships, as well as service transformation. Based on CORM, Section 5 presents a referenced architecture, allowing one to develop and configure a complete service-oriented system. Section 6 describes the system implementation of a CORM-based supported platform, and a case study and related discussion is also given. Finally, Section 7 concludes the paper and discusses future works.

2 Related works

In order to develop and implement enterprise information systems according to business requirements, related researches could be divided into three aspects: business modelling, transforming methods and IT support environment.

(1) Conceptual model for business process modelling

Given a huge number of business modelling approaches, we focus on the conceptual models that emphasize both process definition and execution, leaving aside some process describing methods only for business process design purpose like IDEF [Cheol-Han Kim, 03].

Alessandro [Alessandro, 09] proposed a conceptual modelling approach for business service development to support top-down, model-driven design of professional service integration automation. But the approach lacked implemented architecture, thus difficult for enterprises to use in practice. Jung [J.J. Jung, 09] proposed a novel framework based on aligning business ontology for integrating heterogeneous business processes. It gave a feasible method to realize integration between business processes. But the scalability of alignment-based distributed BPMs was hard to evaluate.

(2) Transforming methods

As web service is a popular method adopted for business process execution recently, we discuss transforming methods mainly focused on service-level transformation.

Ziemann [J. Ziemann, 05] introduced a pragmatic transformation approach to map Event-driven Process Chains (EPC) process to BPEL. This approach used EPC for modelling business process with web services associated, to provide syntax transformation technology. Kopp [Oliver Kopp, 06] defined a kind of EPC to simplify the logic description of complex business process, and then transformed it to BPEL process, but the limitation lies in the inadequacy of web service semantic information itself. Zhao [Chenting Zhao, 09] studied how to generate language-specific business process models and process interaction semantics from UML 2.0 process models. He proposed a set of transformation patterns to map BPEL. However, he lacks some consideration for some other elements such as data, roles or department, which are crucial to construct a complete system for business use.

(3) Supported software architecture for business process execution

Supported software architecture is mainly concerning system configuration in order to implement information system rapidly.

Bianchini [D. Bianchini, 06] proposed a model, a methodology and a tool environment based on ontology for service discovery. The proposed service ontology was in view of functional aspects and was organized on three layers. It aimed to support traditional search based on UDDI. Zaupa [Zaupa, 08] combined traditional product line approach and SOA to present the WIDE-PL environment which supported the generation of Web applications. Because there is an explicit different concern between the business logic of the application and the services logic, Web applications are allowed to be developed and evolved in a simpler manner. K. Votis [K. Votis, 08] proposed an ontologically principled service-oriented architecture for the administration and integration of distributed nodes. Two-level ontology was introduced to enable efficient integration by mapping concepts to actual information in a distributed environment. This method gave a good direction for the searching services and resources from distributed environment. Wei Ye [Wei Ye, 09] introduced a web-based integration platform which used a component model to encapsulate integrated objects and a connector model to specify communication. Based on the platform, end user could integrate enterprise applications and services in a more light-weight manner. The core of the platform focused on the abstractions of attribute, operation, and event.

From the above literatures, we could find that researchers have made lots of improvements for business modelling or application integration, and some transformed methods are also provided to connect these two aspects. But a conceptual model which could seamlessly bridge build-time design stages and runtime execution stage is still left to be proposed.

3 Idea of business-driven system configuration based on CROM

The lifecycle of business process management [W.M.P. van der Aalst, 03] is divided into four phases: process diagnosis, process design, system configuration and process enactment. Figure 1 shows the idea of business-driven system configuration based on CORM. CORM consists of resource models, ontology, and semantic information. It acts as a conceptual model so as to connect two phases of process design and system configuration. By means of service transformation which realizes the transformation from business models to service components in execution stages, CORM, which

contains both structural relationships and operational relationships, could be used to construct both a resource-centric business modelling and a system configuration platform.

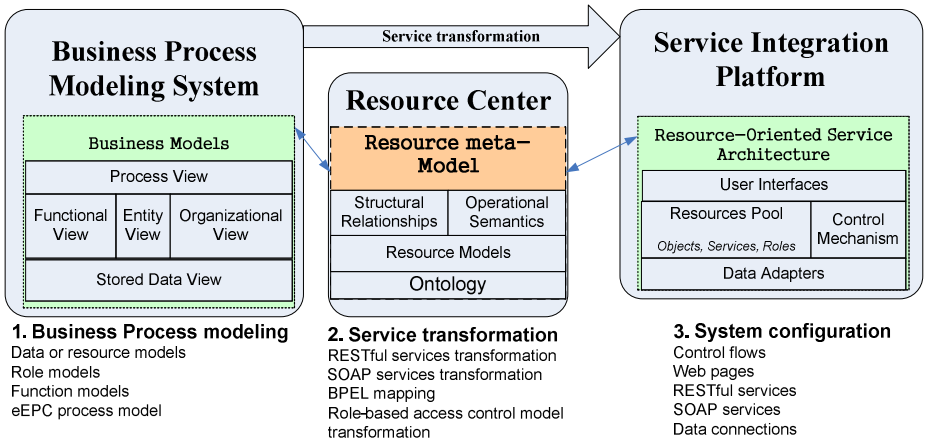

Figure 1: Idea of business-driven system configuration based on CORM

In the design phase, CORM is used to support business modelling and maintain business elements consistence of different views. To connect business elements with service components, service transformation is carried out based on CORM. In the system configuration phase, business functions and information entities are transformed into SOAP and REST services [M. Muehlen, 05] based on structural relationships and operational relationships of CORM.

Three actors are related the process of Business-driven System Configuration. These three actors are described as follows: (1) Business process modellers design business process model according to business requirements. The models obtained from business process modelling provide reference for service transformation and composition. Although business process involves lots of business factors, at least some basic factors such as data, users, and tasks should be included in the process models. (2) IT engineers implement the service transformation and system configuration according to business models. By integrating services to construct a complete system, business requirements and IT infrastructures are aligned. (3) End-users execute the services which are supported by enterprise application. These services are invoked in a convenient and simple way.

4 Conceptual model for business modelling and service transformation

Section 4 presents a formal definition of Resource meta-Model in order to describe individual business elements such as user, information entity and function in an integrated manner. Based on these conceptual descriptions, business modelling and

service transformation could be carried out so as to construct business models and static service descriptions in the design stage.

4.1 Definition of Resource meta-Model

To achieve the goal of system configuration at application-level, all resources in an existing distributed and heterogeneous system have to be firstly identified, analysed and encapsulated. Then all encapsulated resources will be registered into the resource centre under the unified management, so as to hide the information heterogeneity from different systems. Therefore, it is very essential to define a unified architecture for information modelling. Based on business process analysis, a resource meta-model is shown in Figure 2.

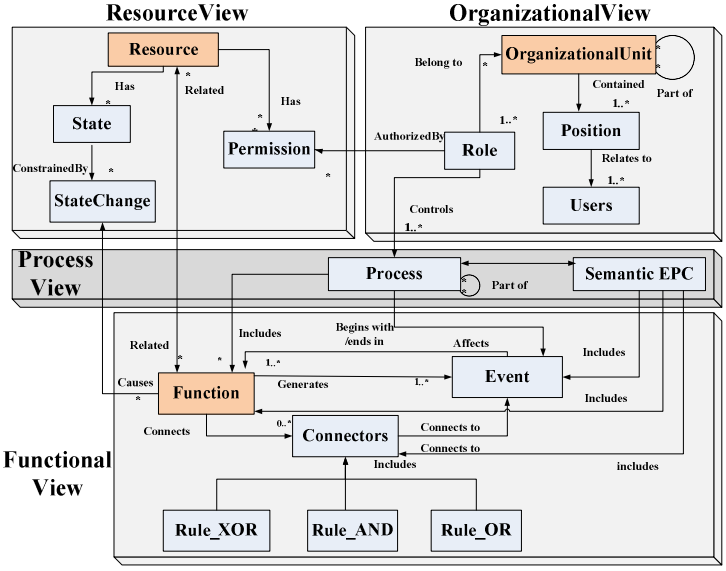

Figure 2: Resource meta-Model

Definition 1: Resource meta-Model (ReM)

The Resource meta-Model is a set of Resource View Model (RVM), Organizational View Model (OVM), Functional View Model (FVM), Semantic Relationship (SR), and Process View Model (PVM). Thus, Resource meta-Model (ReM) is defined as the following tuple:

$$ReM = < RVM, OVM, FVM, SR, PVM >$$

Definition 2: Resource View Model (RVM)

The Resource View Model (RVM) is defined as the following tuple:

$$RVM = <ResID, ResType, SProps, AS, OP, OIP >$$

Hereby the ResID, which is the unique identifier of a resource view model, could be represented as a Uniform Resource Identifiers (URI) in the web environment.

ResType is the underlying category, which describes the type of a resource with a specific static property and can be automatically organized by using a hierarchical tree method which is based on ontology classification.

SProps consists of the static properties of a resource. These properties, e.g. the manufacturing data of a product, will remain unchanged after its establishment.

AS (Activity State) is defined to describe the states of a resource during the execution process. AS represents a set of status of a resource, e.g. a set of state like (created, locked, outgoing, and archived) of resource Order in a sales process.

AS = <Sstart, Send, Sn>

Where Sn = {s1, s2, s3, … }, |Sn|> 0, Sstart∈Sn, Send∈Sn, si = <StateName>. Sstart indicates the initial state of a resource in the state diagram, and Send means the end state. Every si in the set of Sn represents an intermediate state of the resource.

OP (Operation Set) is defined as a set of internal functions related to properties so as to change resource state. Every operation in OP could be used as the format like opi.

OIP (Operation Interface with Permissions) is used to maintain resource security by granting a role the permissions of some operations or some resources.

Definition 3: Organizational View Model (OVM)

The OVM is defined as a set of Positions (position), Roles (role), Group (group), and Relationship between these factors. Thus the Organization Unit Model (OUM) is defined as the following tuple:

OVM = <positions, roles, group, relationship>

OVM is the basic blocks for business actor. It adds relations to tasks.

Definition 4: Functional View Model (FVM)

Functional Point Model (FPM) is defined as the following tuple:

FVM = <InputRes, OutputRes, CtrlRes, PreEvent, PostEvent, ChangedState>

The FVM is defined as a set of Input Resources (InputRes), Output Resources (OutputRes), Control Resources (CtrlRes), Initial Event (PreEvent), Generated Event(PostEvent), and Changed State (ChangedState).

FVM is the basic element to represent atomic logical task in the enterprise.

Events: Events describe the situation before and/or after the execution of a function. Functions are connected through events. An event corresponds to a post-condition of one function and acts as a precondition of another function.

Definition 5: Semantic Relationship (SR)

The Semantic Relationship is defined as a set of Relationships between RVMs, FVMs, and OVMs. Thus, Semantic Relationship (SR) is defined as the follows:

SR=<Relation_RO, Relation_RF, Relation_OF>

In it, Relation_RO represents the relationships between RVMs and OVMs, Relation_RF represents the relationships between RVMs and FVMs, and Relation_OF represents the relationships between FVMs and OVMs. SR describes the association of business elements, which could be used in access control models or mapping processes to construct control-flow.

To improve the integration of the business elements of different views, SEPC, which extends structure of the traditional EPC, is designed to describe business process. The definition of Semantic EPC is given as follows.

Definition 6: A SEPC is defined as:

SEPC= {Events，FVMs，Connectors，Arc，Label, EP}

In the definition, *Events* is a finite nonempty set of events; *FVMs* is a set of function metamodels, including semantic information of the basic web service description, interaction partnership, and related operation data; *Connectors* is a multi-

set of connectors, $Connector \subset \{AND, OR, XOR\}^{+}$; *Arc* is a nonempty set of control flow arcs, $Arc \subset \{Connector \rightarrow (Event \cup FVM) \cup (Event \cup FVM) \rightarrow Connector\}$; *Label* is a set of labels, containing conditional judgment expressions to control the business process logic; *EP* is a set of start points and end points of the process, which could be used as an connected point of two process.

The definition above shows the basic elements of SEPCs. Besides, it is meaningful to restrain the activities and interactions among these elements to keep the semantic precision of SEPCs process.

Definition 7: a SEPC is valid SEPC if it satisfies the following conditions

1) SEPC has exactly one start point and at least one end point.
2) Start/End point must be events, that is, a SEPC start by an Event, and end by one Event or several Events.
3) Each FVM has exactly one predecessor such as Event.
4) Connectors always connect FVMs with Events, and there has to be more than one predecessor or successor, FVM decides which Event will be active based on Connector.
5) Structured Cycles are allowed in SEPC, but Arbitrary Cycles are excluded.

The constraint of start point ensures the uniqueness of process entry, which allows nesting between different processes. Connectors are classified according to the number of predecessors and successors. We define join connector with more than one predecessor (e.g. AND join), and split connector with more than one successor (e.g. XOR split). On the purpose of avoiding concept confusion, connector with multi predecessors and successors is forbidden.

In order to transform business process, we define function meta-model to describe activities for the purpose of service execution. It is useful and clear enough to design activity sequences with straightforward logic, while for the web-service-based business processes, each of which is performed by different partners, the collaboration and data interaction is hard to model. Our function model is based on the extension of attributions in BPEL basic activity units, because BPEL contains enough semantic information for web service composition and execution.

Definition 8: Process Meta-Model is an expression satisfying

< PMM > =<PN, FT, IOs, RF, PT, FH, CF, RealtedSEPC>

- In it, PN is the process name, using string type.
- AT is the action type of process, includes string types {Invoke，Receive，Reply，Assign，Throw，Wait，Empty，Exit} by referring to the categories of BPEL basic activity.
- IOs is a string list which contains input and output parameters of one function. The formula is IOs = (InputParas, OutputParas). Both Input and Output could be a list of string or a single string. IOs will be used to select appropriate resources from resource pool.
- RF describes the partner relationship between functions. RF = (partnerLink, (“myRole”|“partnerRole”), roleName). The partnerLink is the name of the partner relationship. The roleName identifies the role of the function in specified partnerLink. (“myRole”|“partnerRole”) means the function which takes one of the two role types.
- PT expresses the Port Type for BPEL description.

- FH expresses the function including fault handler information, which could contain other PMM for fault handling;
- CF is an optional element, which describes the compensation operations. CF performs differently with FH. CF will start only when the function is finished and will compensate operations to cancel the effect of the function.
- RealtedSEPC is a link to connect a SEPC.

Process Meta-Model includes two parts. One part named Semantic EPC is used to describe the control-flow of functions in the business process, and another part named PMM is used to describe business process for execution. And these two parts are connected through the relationship as one to one.

Semantic information contained in the function meta-model covers these following aspects: profile of process activities; messages and data involved in activities interaction; partnership of functions; port types that classify functions by different function providers; exception handling and compensation handling. Partnership is the key element for understanding the business process with several participants (web service providers). Function meta-models contain partnership information to distinguish different interaction logics. Also it is beneficial for mapping BPEL partner links. Exception handling and compensation handling are necessary when a modelling process is to be deployed and executed.

Therefore, based on the Resource meta-Model, business process of enterprise could be modelling. Using business modeller, RVM, FVM, and OVM could be built. In addition, the advantage of the standardization allows user to export the process or import the process created by other modelling tools like ARIS Toolset.

4.2 Service transformation based on Resource Meta-Model

In order to realize business process by means of service, it is necessary to implement the automated service generation based on resources instances. The process, which is called service transformation, is carried out to generate services related to information resource after business modelling. There are four kinds of service transformation:

- Based on Resource View Model (RVM), RESTful services could be generated. A RESTful service is corresponding to an information resource which could be accessed as by a URI.
- Based on Functional View Model (FVM), Standard SOAP services could be generated. A SOAP service is corresponding to a functional resource case which could define input and output parameters.
- Based on Organizational View Model (OVM), roles and its related functions and information entities could be exacted. This information could be used to constructed access control model in runtime environment.
- Based on Process Meta-Model (PVM), process-centred semantic relationships including data-function relationships, role-function relationships and function-function relationships are used to construct SEPCs. Then, SEPCs are transformed into BPEL or BPMN according to execution requirement in the runtime stage. This information could be used to construct control mechanism for application integration in the runtime environment.

The first three service transformations are single-view transformation, the process could be carried out based on some pre-defined templates. Thus we just take RESTful service as an example to explain the transforming method.

In RESTful services, every identified resource is assigned with a unique URI. And these identified resources could be operated by standard interfaces, i.e. HTTP protocols PUT, GET, POST and DELETE. In order to invoke RESTful services, developers have to assemble URI manually and call appropriate HTTP method to send a request to the resource. Furthermore, there can be inconsistency between the documentation and the actual implementation of RESTful services. It is therefore necessary to describe services formally, thus we choose WADL to describe RESTful services in our framework.

The root of WADL is the application element, which consists of grammars and resources elements. The grammars element contains definition of the exchanged data format by means of XML schema. The resources element is a container for all resources and defines the base URI for all resources. Each resource is divided into an independent resource, which is then represented as the resource element, which is identified with a path to the base URI. Under a resource element, there exists a list of method elements, which represents the allowed HTTP method to operate on the resource. Each method element contains a request element and a response element. The request element is composed by a few parameters and some complex representation. The response element contains the representation of the resource and a fault element if something goes wrong during the execution.

When the structure of WADL is parsed, we developed an algorithm to generate WADL document by means of RVM and the state chart of resources.

Process transformation is complex compare with RESTful service transformation. Process generally could be used as a centre view to integrate some business elements such as tasks, roles and entities. It's an essential to describe the business process following some standardized process execution language such as BPEL or BPMN. Therfore, the process could be easily executed on the workflow engine, such as JBPM.

Basic activities in BPEL include information that describe activities interaction, referred messages, partnership of service providers and related operations. Table 1 shows the details about process meta-models mapping.

To generate executable basic activities, the semantic information in functional models is extracted. We first distinguish the types of function. Then, according to the different types, semantic information is transformed to form properties of BPEL basic activity. Partner relationships in basic activities are directly from the graph of PVMs. Ahead of the mapping process, the partnership logic defined in PVMs should be verified to satisfy peer-to-peer conversational partner relationship patterns.

Mapping process should obey the following rules:

- The FVM has to define partner relationships and has only one partner role.
- The number of function meta-model with same partnerLink name should be no more than two.
- The function meta-model with same partnerLink name should have different kinds of partner roles.

BPEL Basic Activity	Meta model Semantics Extraction	BPEL Mapping
Invoke	<AT>< PN > <IOs><PT> <RF><FH> <CF>	<invoke partnerLink= "partnerLinkName" portType="portType" operation="Function Name" inputVariable="input" outputVariable="output"> <catch faultName="faultName"> **Fault Handler: activity** </catch> <compensationHandler> **Compensation Function: activity** </compensationHandler> </invoke>
Receive	<AT>< PN > <IOs><PT> <RF>	<receive partnerLink ="partnerLinkName" portType="portType" operation="Function Name" variable="input"> </receive>
Reply	<AT>< PN > <IOs><PT> <RF><FH>	<reply partnerLink ="partnerLinkName" portType="portType" operation="Function Name" variable="input" faultName="faultName"> </reply>
Assign	<AT><IOs>	<assign> <copy> <from variable="input"><to variable="output"> </copy> </assign>
Throw	<AT><FH>	<throw faultName="faultName"> </throw>
Wait	<AT><IOs>	<wait> <for>input</for> <until>output</until> </wait>
Empty	<AT>	<empty></empty>
Exit	<AT>	<exit></exit>

Table 1: Process Meta model mapping

5 System configuration based on MVC pattern

By means of the business models and service components archived in design stage, Section 5 presents a method of configure system to construct a complete information system in the runtime stage. The method focuses on a referred system architecture, which contains resource-based services and related control mechanism.

5.1 Resource-based service architecture for system configuration

The software architecture for system configuration is mainly focused on two types, SOA and ROA. [X. Xu, 08]. SOA provides methods for developing and integrating systems which package functionality as interoperable services. Compared with resource-based service [L. Richardson, 07], it could be regarded as a kind of process-based service. Since process-based services need to send messages as input and output information directly or indirectly, there are still more logical connections between services. RESTful web services, which are based on resources, could run more

flexibly and efficiently than process-based web services for end-users, especially in a large-scale and distributed environment.

Based on standard MVC pattern [S.S. Hasan, 11], A reference software architecture for system configuration is constructed. Figure 3 shows the architecture included three layers. In Data Layer, data connections are used to attain data from different data sources based on adapters. In Logical Layer, the external and internal resources are discovery and invoked. The controller in Logical Layer refers to access control and process schedule. In the User Layer, users could apply services to the personalization of their work environments so as to fulfil business requirements.

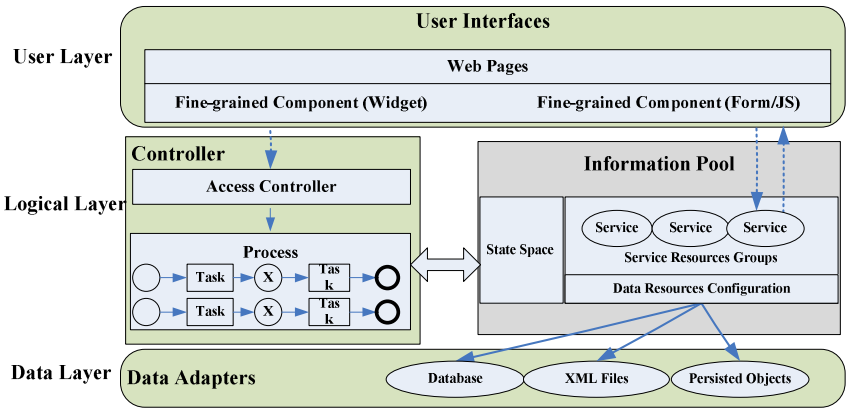

Figure 3: A reference software architecture for system configuration

User Layer. The integration in this layer refers to the construction of applications by integrating components at the graphical user interface level. There are mainly two kinds of components here: the finer-grained component and the coarse-grained component. Finer-grained components could realize relatively simpler function with fewer interfaces; they are typically independent, and always shown as simple applications, such as the Widget. Coarse-grained components mostly present as web pages like Javascript or Forms. Both of them could be used to construct the application.

Logical Layer. There are two modules: Controller and Information Pool for service integration in Logical Layer. Controller realizes access control and task schedule according to the business process. Every business process consists of a start event, an end event and several tasks. Information Pool could be regarded as a container for service execution, where a state space based on resource is constructed to monitor and manage resource-based services. According to the requirements, BPM engine could be also included in the information system for complex application.

Data Layer. We could use data adapter to gain more effective usage of existing data in Data Layer. This layer contains the distributed data information from heterogeneous database with different data formats. Data services enable the sharing of structured data in database and semi-structured XML documents. The purpose of data layer is to simplify the integration of distributed data sources.

The architecture provides a software configuration to run services. Components of user interface, such as web pages, could be generated based on services description. Based on the service parsed, some parameters could be obtained from service description files such as WADL, WSDL. User interfaces could be generated automatically by means of JavaScript. The information resource pool consists of SOAP services, RESTful services, Data Connection and Access Control Model. Access Control Model contains roles, and their relationship with functions and entities, which act as authorization for services invocation. Data Connection acts as persisted objects to connect data-sources in distributed environment.

5.2 Construction of information pool based on ontology

In resource-based service architecture, information pool acts as a runtime environment for resource accessing, disposing, and displaying during service execution period. For the purpose of constructing a flexible and configurable environment, ontology is built to organize and utilize resources dynamically.

Definition 9: Domain Ontology for Resource (DORe)

Domain Ontology for Resource is a logical structure for resources management and disposing. This facilitates the construction of semantic relations between resources as well as the use of these resources in a dynamic manner. In fact, DORe could be regards as a general representation to express the relations between resources. It is represented as:

$$DORe = (ReM, VirReM, Relations)$$

In the above definition, ReM is the basic node related to a concrete resource such as services, business entities, and functional components. VirReM is the node named virtual resource, representing a set of composed resources. And Relations denotes the relationship used to connect different nodes.

Based on OWL-S [Matthias Klusch, 09], we could depict it in ontology description language. A logical grammar structure of DORe is shown in Table 2.

<DORe>::=<ReM> | <ReM> ∪<VirReM>
<VirReM>::=<Node><Relation><Node>
<Node>::=<VirRemClass> | < ReM> | <ReMObject>
<ReM>::=CLASS:<RemIdentifier><Property><Operation> END
<Relation>::=< Generalization > | <Specialization> | < Association >
<VirRemClass>::=CLASS:<VirRemIdentifier><Property><ReMi><Relationij><ReMj>END
<ReMObject>::=OBJECT:<ObjectIdentifier>< Property><Operation>END
<Property>::=PROPERTY:<PropertyIdentifier>:<InitialValue>END

Table 2: Logical structure of Domain Ontology for Resource

Referred to Table 2, the relationships of resources are divided into three types:

- Generalization is used for constructing VirReM based on two or multiply ReMs;
- Specialization is used for generating a ReM from a VirReM , and the new ReM is always a part of a VirReM;
- Association represents a weak link between two independent resources.

Figure 4 illustrates an ontology fragment related to an example ERP system. This fragment includes several resources and correlative relationships, the resources include several independent resources such as PurchaseOrder, Product, GoodsReceipt, as well as a virtual resource PurchaseDocument which consists of some independent resources like Invoice and PurchaseOrder. Resources of domain ontology may have complex and dynamic associations to fulfil different requirements. Some operations are inside the resources so as to connect operational semantic with structural semantic, which could be used to connect RESTful services and SOAP services in the run time.

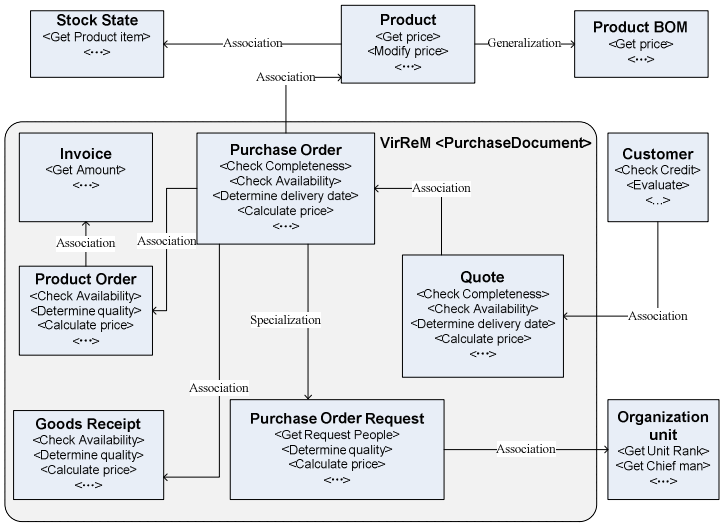

Figure 4: Ontology fragment of an example ERP system

For the purpose of maintaining the semantic relationships based on ontology, according to Stojanovic [Stojanovic, 02], high-level changes are considered as composite changes which are constituted by a certain number of elementary changes. Therefore, it's crucial to identify all the essential elementary changes and express them in a formal way. As Table 3 shown, to maintaining the information pool dynamically, the primary elementary changes of operations are defined.

In the information pool, all information resources related to this process are collected together as a resource array of several dependent or independent resource objects. The static information has been described with a set of resources and activities in business process model. When this business process is executed, these resource objects could be changed step by step. During the runtime of business process, every resource in the resource array is disposed and its state is converted. What should be emphasized is that one activity should have the state of at least one resource object, and the conversion is allowed to occur once a time only. After assigning resource objects with different states according to information disposing process, we can deduce the functions of the whole business process by the state transitions of all related resource objects.

Elementary Change	Syntax & Semantic
Create new ReM	CreateReM (ReM rem) ReM={CLASS | OBJECT} Create a new ReM into ontology.
Deep delete ReM	Delete(ReM rem) Recursively delete all rem refer in ontology then delete rem assertion.
Generalize ReMs	ExtractVirReM(ReMs rems) ReMs ={ReM1, ReM2,…, ReMn} Create a generic ReM for a set of unrelated ReMs and transfer common properties to it.
Specialize ReMs	ExtractSubReM(ReM) Split a ReM into several ReMs and distribute properties among them.
Create new relation	CreateRelation(Relation rel) Create a new Relation assertion into ontology with no connections.
Deep delete relation	Delete(Relation rel) Recursively delete all rel refer in ontology and delete rel assertion.
Add relation	AddRelation(Node domain, Relation rel, Node range) Node={VirClass | ReM} Add a relation rel between domain and range
Remove relation	RemoveRelation(Node domain, Relation rel, Node range) Remove the relation rel from domain and range
Add/Remove property	AddProperty(Node node, Property prop) /RemoveProperty(Node node, Property prop) Add a property prop into node. /remove the property prop from node

Table 3: Elementary operations of Information Resource Pool

Based on the constraints from initial state of resources in the resource array and state diagram inner every resource, a finite state machine [Huang Shuai, 10] could be built to control connections between different RESTful services related to a process instance. The services would depend on the resources only without considering the business activities. Based on the state space defined with the resource array, the goal and constraint could be used to control the whole process, no matter whether the granularity of services is coarse or not.

6 System implementation and case study

In this section, a CORM-based supported platform is presented to implement and verify the idea. A case study with all the steps is also shown to explain the application. At last, related discussion is given by comparing with one typical similar platform.

6.1 A software platform for system implementation

Based on the idea of business-driven system configuration, a CORM-based supported platform could be built, which included three parts: Business Modeller, Service Integrated Environment, and Resource Centre.

(1) Business Modeller for business elements description

Business Modeller supports multi-views of business modelling so as to retrieve and encapsulate information resources from distributing and heterogonous application.

Business Modeller is used to build functional model, organizational model, business resource model and process model, as well as a united data view interactive with database for storage purpose to support the business models.

(2) Resource Centre for service transformation and management

Resource Centre is designed to connect Business Modeller and Service Integrated Environment, thus focus in on service transformation and management of these resources and services to achieve reusability in system configuration stage. To invoke application and data in a convenient way, there is a need to execute business process by means of services. Therefore, realizing the automated generation of services based on resource models becomes very significant. The process of service transformation is carried out to generate services relevant to information resource. SOAP services are related to functional model, and RESTful services are related to information entities. After service transformation, resource models from different sources are mapped into a resource centre according to these relationships. Based on ontology, these resources are organized and managed to construct a semantic and powerful resource centre.

(3) Service Integrated Environment for system configuration

Referring to MVC pattern, system configuration is then fulfilled by Runtime Service Environment. When services are generated from Resource Centre, resource state space could be constructed as control mechanism during the service execution. Then the tasks of service parse, process mapping, and service integration are carried out. Therefore, a complete system could be constructed by means of system configuration.

The CORM-based supported platform is already used to construct a configurable integrated platform for a ship-building manufacturing company. Based on re-useable resource models, the platform supports rapid modification of information systems when the process changes. By employing different types of resource as driven element, we also proposed different development patterns, such as service-driven, data-driven, and user-interface-driven pattern. We do not make further discussions on these patterns here due to the page limitation. Referred to the idea of business-driven system configuration in Section 3, the process is illustrated in Figure 5.

There are three documents generated in CORM-based supported platform during the disposed process. (1) According to the business models created in Business Modeller, information resources in XML format are recognized and encapsulated from heterogeneous system environments through resource adapters. Based on the CORM, these resources can be registered in the Resource Centre, which acts as a reusable enterprise resource library for the purpose of attaining business models and extracting application cantonments for system configuration. (2) Resource service description in WADL format is generated to describe resource services, where all resources in resource model are assigned with well-designed URIs, possible HTTP methods and proper representation, so as to be understood and accessed by the application. (3) Process orchestration document in BPEL format describes the control-flow of business process. All involved resources of a certain process are put into a resource array to construct a resources state space, thus carrying out the process state transitions to control the process execution.

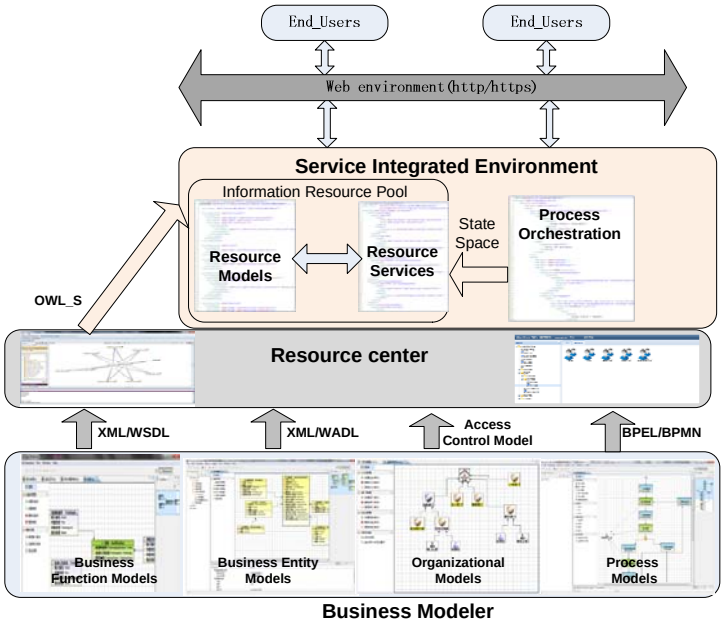

Figure 5: System implementation of CORM-based supported platform

6.2 A case study

In this session, we will explain and validate the busness-driven system configuration through the CORM-based supported platform step by step based on the following procurement scenario:

Company A needs to build a enquipement purchase system according to the existing business process. A typical process could be described as follows: Purchase department create POR(Puechase Order Request) accroding to requremnets from different sources, then combines these requremnets and submited POR to office for pre-check. Management department will carry our POR check accroding to company execution plan, them submit it to Financial Depaetment for Financial Approval. If POR approved, Purchase department create RFQ(Request for Quotation), which contained product types and numbers. Then, Purchase department will interact with company B to get ReturnedGoods Info. Purchase department then decides vendor and create PO. Accordingly, Office and Financial department will carry out some activities according to PO Submission.

Step 1: Busibess modeling by means of Business Modeller

First of all, based on the business requirements, related business entities and activities are being modeled in the business modeling system. We describe functional models, entity models, organazational unit models, and business process models in Business Modeller, which is a eclipse plugin supporting multi-view modeling with a connection to the web-based Resource Centre. Business Entity resources, such as PurchaseOrder, PurchaseOrderRequest and RFQ are created in this step , some fragment could be refered to Figure 4.

Step 2: Service transformation and Service registering

Based on business modeling, corresponding services are generated by means of service transformation: inforamtion-centric services are generated from entity models, and interface-centric services are generated from functional models.

Table 4 displays several services related to the purchase process, which are transformed and registered in the Resource Center.

ActivityDescription	**Service Name**	**Related Organazation**
Create PO Request	PODraft	Purchase Dept.
Submit POR	POSubmission	Purchase Dept.
Pre-Check	OfficeApproval	Office
POR Check	DepartmentApproval1st	Management Dept.
Financial Approval	FinancialApproval	Financial Dept.
Create RFQ	RFQCreation	Purchase Dept.
Return Goods Info	POInfoReturned	Purchase Dept.(Interact with Company B)
Decide Vendor	SupplierChoose	Purchase Dept.
Create PO	POCreation	Purchase Dept.
Office Submission	OfficeSubmission	Office
Financial Record	FinancialSubmission	Financial Dept.

Table 4: Services list of the example purchase scenario

In this step, we need to map operations and parameters of the specified service to the component ID to support the implementation of the service functions. The mapping allows for two types of service invocations, SOAP and REST,adopting two different invocation paths. The SOAP service invocates resource with WSDL format, and the invocation path is in the form of Operation/Message/MessagePart/Parameter, while REST invocates resources with WADL format and the path is Operation/ElementPath. In contrast, REST architecture is easier to invocate.

Step 3: Resources registered and management in Resource Center

When we carry out service intergation, we should make sure that the necessary resources are available. There are three possible cases at the beginning of the service discovery. In the first case, all necessary services are registered in the Resource Center. The Service Parser analyzes the service information and extracts the operations as well as input and output parameters in the WSDL. In the second case, users want to find services that match the functions they specified. They could query their target services via key words, since services have been named following specified principles and services adopting the semantics could be intelligently discovered. System allows for services management and automatic expression of services as a service tree based on semantics, making it easier to discover certain services. The query results are registered in the Resource Certre as well. In third case, the ideal services are not available. Therefore, users have to add new services with the help of IT staff. The occurence frequency of the third case depends on the capacity of the existing instances.

Step 4: Control mechanism construction based on business processes

Accroding to the business process model, developers will configure the process by assigning resource instances to a task of business process. Each task in this process is

corresponding to an activity in the procurement process. Users prefer to configure the business process in a visualized interface through drag-and-drop operation. Hereby, we define the unit in a "process" as a "task", which is a draggable component. In accordance with previous scenario, PO submission is a task of the procurement process. It support the creation of relations among tasks, such as "and", "or", "not" even more complicated logic relationship. A web-based user interface of the resource integrated module is given in Figure 6 to show how the process mapping tasks.

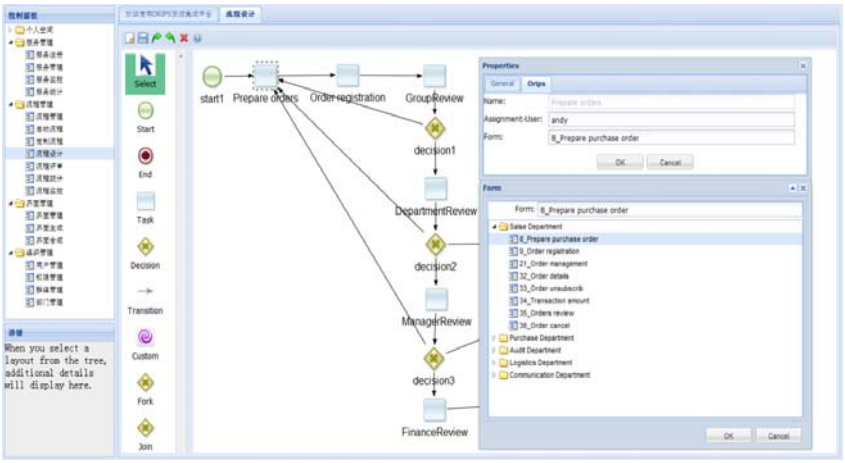

Figure 6: A program interface of process mapping resource

Step 5: System configuration based on services description.

After services are assigned to the tasks of process, we need to attach the executable function to the UI component so as to construct a complete system. Each component, such as textbox, label and table, is assigned a unique id during the configuration of it. In this step, users could design their own UI corresponding to specified task by drag-and-drop. For PO submission task, you can add an uneditable textbox on a Form to present the PO number and another textbox to allow the purchaser to input the price and quantity. Each task corresponds to one Form. Moreover, on account of improving user experience with better UI design process, Some other web page developing tools that support standard XHTML can be used.

After these five steps, the process called services-driven system configuration is accomplished. From the perspective of cost, CORM-based supported platform greatly shortens the system development cycle, and saves more labor efforts than other traditional developing methods. Also, end users could develop a complete system without the participation of the IT staff, with more attention paid on the business logic to reduce the mistaken understanding of business requirements.

6.3 Discussion

Web2Exchange is a model-based service transformation and integration environment [Srinivasmurthy, 09]. On the purpose of explain our approach, some features of CORM-based supported platform are compared with Web2Exchange, see Table 5.

Stage	Feature	CORM-based supported platform	Web2Exchange
Design stage	Model element	Entity, Functions, OrganazationUnit (Role, User), Process	Entity(Class, method, property)
	Meta-model	Multi-views business model extended ARIS	Common Information Model in a DMTF standard
	Involved models	Both business and system model	system model
Transformation	Transformation level	Service level by REST service generation, SOAP service generation, and process transformation	Code level by the mean of system model, source code notation, packaging services
	Service component	REST/SOAP services Process services(BPEL/BPMN) Accessed Components(RBAC Model) Data adapter	Foundation Services Model-based services Communication
	Access mode	Both REST or SOAP	Both REST or SOAP
Execution stage	Integration pattern	Business-driven System configuration	Lightweight Mashup
	Configuration	Referred to MVC pattern, Services connected to Data adapter and UI components, Process (BPEL/BPMN) act as controller to configure a complete a web-based system.	Based on Model-based Service Integration Environment (SIE), Foundation Services, Communication, and Model Management are configured.
	Control mechanism	State Space based on resources	Not mentioned
	Platform	Both a modelling and service integration platform, could be used as a EIS development and execution platform.	A service integration platform similar to Enterprise Service Bus (ESB), focus on services communication.

Table 5: Features of CORM-based supported platform

See Table 5, features of CORM-based supported platform with Web2Exchange are divided into three stages of design, transformation, and execution. We could find that CORM-based supported platform provides a more flexible approach to construct a complete information system by connecting build-time business models with run-time execution. Additionally, application integration is based on resources with semantics, thus carrying out complicated and rapid integration. It owns more professional advantages than some lightweight application integration such as Mashup.

7 Conclusion

For the purpose of bridging the gap between businesses models with running service-oriented system, a conceptual model, CORM is proposed to describe, encapsulate, manage and configure information resources. These business models and IT elements could be recognized, managed, transformed and configured to develop and implement enterprise information systems. In some information-centric software such as ERP system, E-Business system, the IT elements attained from existed systems and databases could be re-used to configure a new system according to the business requirements. Therefore, CORM provides a more effective approach to support information system development and business adjustment.

Our further tasks will focus on the functions features towards cloud computing environment, for example, realizing multi-rents mechanism and group-based service scheduling. And one other direction is to maintain consistence of different types of resources by means of ontology evolution autonomously.

Acknowledgements

This research is supported by the National Natural Science Foundation of China under No. 71171132, and the National High Technology Research and Development Program of China (“863” Program) under No.2008AA04Z126.

References

[Alessandro, 09] Alessandro Bozzon, Marco Brambilla，Federico Michele Facca，Giovanni Toffetti Carughi, A Conceptual Modeling Approach to Business Service Mashup Development, Proceedings of IEEE International Conference on Web Services 2009, pp.751-758

[Chenting Zhao, 09] Chenting Zhao, Zhenhua Duan, Man Zhang: A Model-Driven Approach for Generating Business Processes and Process Interaction Semantics. Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science.

[Cheol-Han Kim, 03]Cheol-Han Kim, R.H. Weston, A. Hodgson, Kyung-Huy Lee，The complementary use of IDEF and UML modelling approaches，Computers in Industry, Volume 50, Issue 1, January 2003, Pages 35-56

[D. Bianchini, 06] D. Bianchini, V. D. Antonellis, B. Pernici and P. Plebani, Ontology-based methodology for e-service discovery, Information Systems 31 (2006), pp.361–380

[Howard Smith, 03]Howard Smith, Business process management the third wave: business process modelling language (bpml) and its pi-calculus foundations, Information and Software Technology, V45(15), 2003, pp1065-1069

[Huang Shuai, 10] Huang Shuai, Cai Hongming, Xu Boyi,A Resource State-based Business Process Control Mechanism for BPM, Proceeding of IEEE PIC-2010,IEEE Computer Society, 2010.12,p1157-1162

[J. Ziemann, 05] J. Ziemann, J. Mendling. EPC-Based Modelling of BPEL Processes: a Pragmatic Transformation Approach. In Proceedings of MITIP 2005, 2005.

[J.J. Jung, 09] J.J. Jung, Semantic business process integration based on ontology alignment. Expert Systems with Applications, Volume 36, Issue 8, October 2009, Pp 11013-11020

[K. Votis, 08] K. Votis, C. Alexakos, B. Vassiliadis and S. Likothanassis, An ontologically principled service-oriented architecture for managing distributed e-government nodes, Journal of Network and Computer Applications 31 (2008), pp 131 – 148

[L. Richardson, 07] L. Richardson, S. Ruby, RESTful Web Services, Cambridge: O'Reilly, 2007

[M. Muehlen,05] M. Muehlen, J. V. Nickerson, and K.D. Swenson, "Developing web services choreography standards: the case of REST vs. SOAP," Decision Support Systems, vol. 40(1), 2005, pp. 9-29

[Matthias Klusch,09] Matthias Klusch, Benedikt Fries, Katia Sycara, OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 7, Issue 2, April 2009, pp121-133

[Oliver Kopp, 06] Oliver Kopp, Tobias Unger, Frank Leymann: Nautilus event-driven process chains: Syntax, semantics, and their mapping to bpel. In 5th GI workshop on Event-Driven Process Chains(EPK 2006) vol. 224, pp. 85–104.

[S.S. Hasan, 11]S.S. Hasan, R.K. Isaac,An integrated approach of MAS-CommonKADS, Model–View–Controller and web application optimization strategies for web-based expert system development",Expert Systems with Applications, V38(1)1, 2011, pp 417-428

[Stojanovic, 02] Stojanovic, L., et al., User-Driven Ontology Evolution Management, Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web, A. Gómez-Pérez and V. Benjamins, Editors. 2002, Springer Berlin / Heidelberg. pp133-140

[Srinivasmurthy, 09] Venugopal Srinivasmurthy, Sanjeeva Manvi, Ravi Gullapalli, et al. Web2Exchange: A Model-Based Service Transformation and Integration Environment, Proceedings of IEEE International Conference on Services Computing 2009, pp 324-331

[Wei Ye, 09] Wei Ye, Wenhui Hu, Xin Gao, Wen Zhao, Shikun Zhang, Lifu Wang, A Mashup Platform for Lightweight Application Integration, Proceedings of International Conference on New Trends in Information and Service Science 2009, pp27-32

[W.M.P. van der Aalst, 03] W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, Proceedings of the 2003 international conference on Business process management (BPM'03), Springer Publish, pp 1-12

[X. Xu, 08] X. Xu, L. Zhu, Y. Liu and M. Staples, Resource-Oriented Architecture for Business Processes, 15th Asia-Pacific Software Engineering Conference 2008, pp. 395-402

[Zaupa, 08] Fábio Zaupa, Itana M. S. Gimenes,Don Cowan. A Service-oriented Process to Develop Web Applications, Journal of Universal Computer Science, vol. 14, no. 8 (2008), pp 1368-1387