가상기업의 형성을 위한 컨텍스트 기반 프레임워크

이 경 휘* · 오 상 봉**

Context-Driven Framework for High Level Configuration of Virtual Businesses

KyungHuy Lee* · Sangbong Oh**

Abstract

In this paper we suggest a context-driven configuration model of virtual businesses to form a business network model consisting of role-based, interaction-centered business partners. The model makes use of the subcontext concept which explicitly represents actors and interactions in virtual business (VB) context. We separate actors who have capacities on tasks in a specific kind of role and actor subcontext which models requirements in specific interaction subcontext. Three kinds of actors are defined in virtual service chains, service user, service provider, and external service supporter. Interaction subcontext models a service exchange process between two actor subcontexts with consideration of context dependencies like task and quality dependencies. Each subcontext may be modeled in the form of a situation network which consists of a finite set of situation nodes and transitions. A specific situation is given in a corresponding context network of actors and interactions. It is illustrated with a simple example.

Keywords : Virtual Enterprise, Configuration Framework, Context-Driven

논문접수일 : 2006년 10월 02일　　　논문게재확정일 : 2006년 12월 06일

* 교신저자 대전대학교, IT경영공학과 부교수, (300-716) 대전광역시 동구 용운동 96-3, Tel : 042-280-2563, e-mail : khlee@dju.ac.kr
** 대전대학교, 정보통신공학과 교수

1. INTRODUCTION

Virtual enterprises are generally described as *conglomeration of outsourced services that collaborate temporarily to achieve a shared business goal* [9, 15]. Lots of service providers and external supporters are needed in interactively doing virtual businesses as well. One of the issues in enabling virtual enterprises is how to configure and integrate internal and external services so that it could cope effectively with a massive range of user's demanding requirements on market. Virtual businesses need the nature of highly dynamic and configurable which includes several users and external service providers. Context means any information that can be used to characterize the situation of entity [3]. Because of its ubiquity and heterogeneity over the Web, one of the promising approaches is *context-driven* in developing information systems. Many efforts of recent research on context have been focusing on user's preference and context issues in a ubiquitous computing environment, in which context and context awareness provide the necessary concept of relevant services and information to users based on situational conditions. The notion of context is important in that *it can capture many of the interesting aspects such as relativity, partiality, locality, and independence* [7, 10, 14]. Creating better context models which represent situations based on specific context information has always been a major goal in context aware computing. In the domain of virtual businesses, context contains information on virtual teams as well as individuals. Although a few research efforts on this issue have been made so far [2, 4, 5, 8, 12, 13], we recognize that it would be still in its infancy.

In this research we propose a context-driven approach of modeling virtual businesses, which aims specially to model both 1) virtual business configuration and 2) actor and interaction subcontexts. In the model, actor subcontext gives context information about requirements on role-task-quality in a given situation and interaction subcontext models context dependencies like task and quality dependencies. The rest of this paper is organized in the following : section 2 reviews problem statements and requirements for modeling virtual business configuration. Section 3 presents the concept of subcontext and then models subcontext of actors and interactions and in section 4 the procedure of modeling context-driven configuration is given. In section 5, we give concluding remarks.

2. VIRTUAL BUSINESS CONFIGURATION

In this section, virtual business configuration is first explained and then modeling requirements are followed in context of demand and supply chain.

2.1 Terminology

Before going further, we provide informal meanings of terms and terminologies that will be used here, as listed in the following.

VB	: *Virtual Business* (used to represent business being temporarily and virtually formed with participation of two or more business partners)
VBC	: *Virtual Business Configuration* (used as a static or dynamic organizational structure of *VB* being composed of actors that is, business partners)
VBS	: *Virtual Business Service* (used to represent a set of services in *VB*)
VSR	: *Virtual Business Service User* (used to represent user context of *VBS*)
VSRm	: *VSR member* (used to represent possible members of *VSR*)
VSP	: *Virtual Service Provider* (used to represent provider context of *VBS*)
VSPm	: *VSP member* (used to represent possible members of *VSP*)
ESP	: *External Outsourcing Service Provider* (used to represent supporter context of external outsourcing services)
ESPm	: *ESP member* (used to represent possible members of *ESP*)
ToS	: *Task of Service* (used as task of services required for service exchange)
QoS	: *Quality of Service* (used as quality of services required for service exchange)

2.2 Problem Definition

As described in the previous section, *VB* starts with recognizing *VSR* demanding business services on market. It requires formation of several business partners who want to participate in the opportunistic businesses. It requires also specifying contracts about products or services to be delivered, deadlines, quality of products, and cost of services. *VBC*, for example in form of coalition or consortium, can be roughly described as an aggregated, organized group of agentified actors, whose cooperation is regulated by a mutual contract, interacting in order to generate aggregated *ToS*s and *QoS*s that, in some cases, are more complex than the simple addition of their individual capabilities [9]. Actors here mean aggregated concepts consisting of *VSP* and/or *ESP* which are not necessarily software components. From the description, the core actor of *VBC* is *VSP* which plays agentified roles of regulating, aggregating, and providing *VBS*, respectively. In other words, *VBC* can be viewed as a business collaborative network consisting of *VSP* and/or *ESP* with functional (*ToS*) and non-functional (*QoS*) capabilities in a given specific situation. *VBC* continues to adaptively configure or reconfigure *VSP* and/or *ESP* in response to contextual changes in operational environments. Sometimes it necessitates new service or actor creation by merging multiple *VSPs or ESPs*, replacing *VSPm or ESPm*, or regulating *QoS* totally or partially. <Figure 1> shows the influence diagram of *VBC*, which consists of actors (*VSR, VSP, ESP*), interactions (*VSR-VSP, VSP-ESP, VSP-operational environment, etc.*), and an operational environment that they collaborate to operate on. As can be seen in the <Figure 1>, the basic structure of *VBC* in terms of actors and interactions can be understood in a straightforward manner. Those factors in the

<Figure 1> vary a lot in *VB* context over every situation of *VBS*, which means single business service commitment. In other words, every situation of *VBS* has its own context and context information such as roles-tasks-qualities etc. Moreover, *VB* should be operated in heterogeneous computing environment in terms of both business computing and information system. Therefore, we recognize *VBC* as a high level context model which necessitates a very complicated abstraction process from low level context information models about actors or environments.

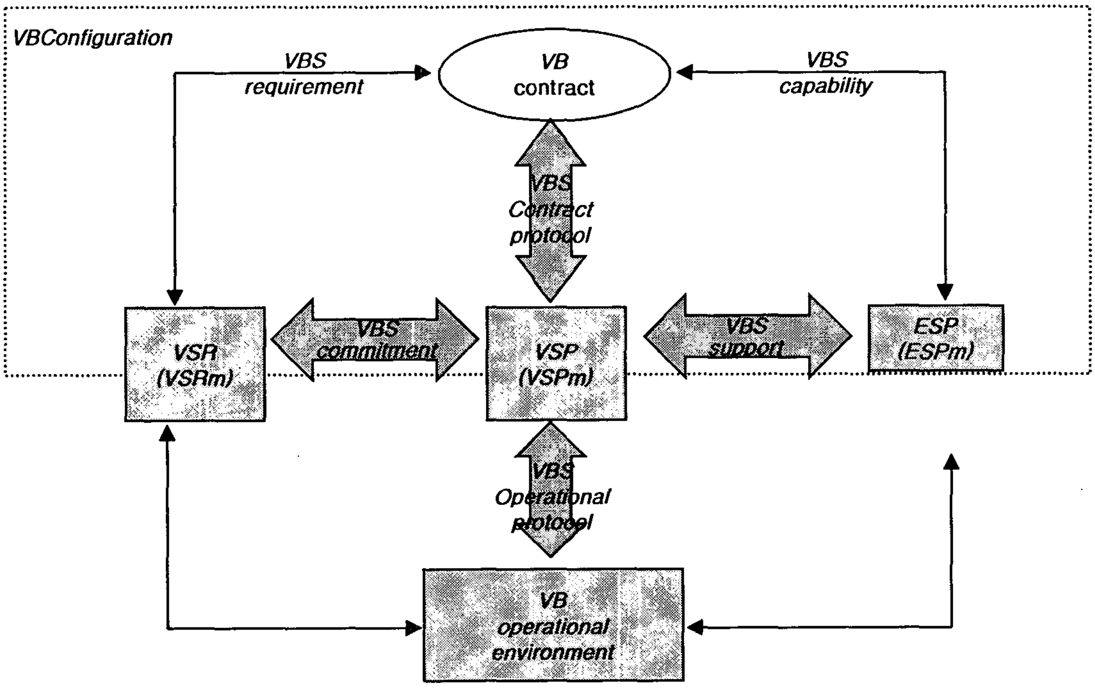
In order for *VBC* to be practically managed in each situational *VB* context, the following requirements must be satisfied.

R1 : Separation of actor and actor context.

Generically, actor means individuals or organizations out of *VB* context. Once *VB* has been created, actor creates its (actor) context for *VB*, enters into *VB* context, and then plays its role within interaction context. Therefore, actor context means capability that the actor can provide in a specific *VB* situation. It may be differently figured out in every situation.

R2 : Context linking and composition with ToS and QoS.

Once *VB* has been created a lot of interactions need to be configured in order to achieve a goal, for example, *VSR-VSP*or *VSP-ESP*, etc. Interaction context requires a coupling of service provider (*VSP*) context and service receiver (*VSR*) context under a specific interaction intent, rules and protocols.



<Figure 1> Influence Diagram of VBC

That is, potential actor contexts interact each other in order to perform exchange of information or service according to business intents such as contracting, fulfillment, communicating acts. Therefore basically *VBC* requires context collaborative network which links and composes relevant contexts according to situational requirements and capabilities of *ToS* and *QoS*.

3. SUBCONTEXT MODEL

In this section, we first review the concept of virtual business context and then describe actor and interaction subcontexts which are primary constituents of *VB* context.

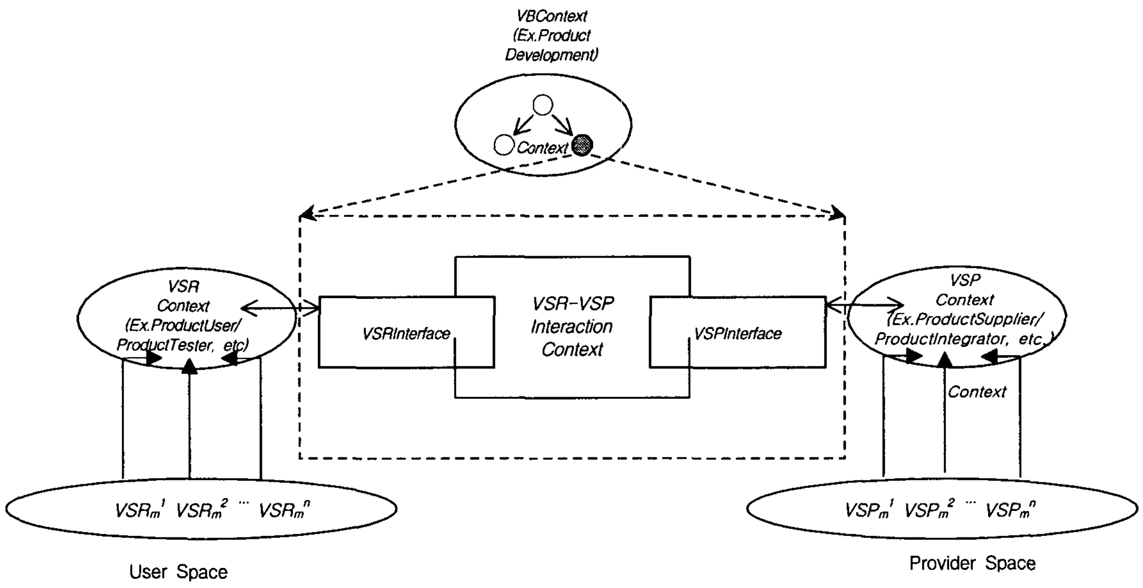
3.1 Virtual Business Context

In this study, *VB* may be described as a temporary business project conducted by two or more collaborating business partners, that is, virtual enterprises. In modeling *VBC*, *VB* context can be described as integrated concepts and properties about local entities such as actors and their relations in the global context. It should be instantiated when a virtual enterprise composed of several actors initiates *VB*. Basically subcontexts are localized, but they need to interact or collaborate in order to achieve *VB* context. Subcontexts are groups of feature values that change together and have the following characteristics[1]:

- Subcontexts can be nested and conforms to a schema that defines (1) available features and (2) nesting relationships. For example, one *VB* actor context has another nested actor subcontext or one interaction context has also another nested interaction subcontext. Those contexts may be represented with conceptual schema objects.
- For each subcontext schema, there is at most one subcontext active at a certain instant of time. The others are present as inactive, "parallel" subcontexts yielding the same aspects about the user. For example, one actor context has its unique role in a single *VB* context, that is, a different role the actor provide in the different *VB* context is recognized as different actor context.

<Figure 2> shows the concepts of *VBC*, in which we assume that two types of subcontexts exist in *VB* context, that is, one is *actor* context and another is *interaction* context, as briefly explained in the following.

- *Actor context*: In *VB*, an actor means an individual or an organization which is a collective actor which represents organized groups of interacting individuals with specific purposes such as teams or departments [12]. A collective actor may be further refined by nested relations. Actor context plays a specific role within a given interaction context.
- *Interaction context*: In *VB*, an interaction is defined as *a unique association between specific service type, particular trigger event, and application roles that send and receive the service*[6]. In our research, interactions

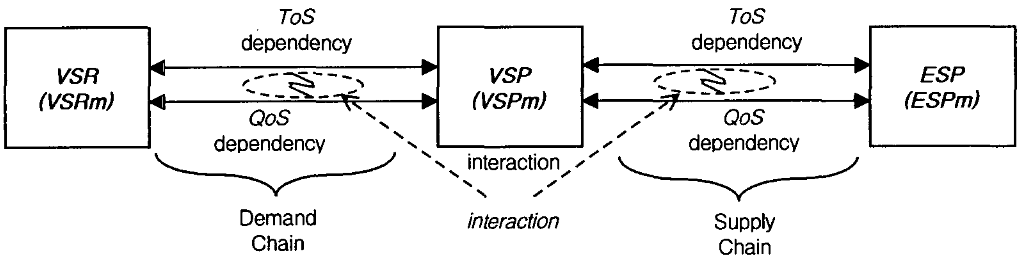


〈Figure 2〉 VBC contexts

may come out in several forms such as commitment, business transaction, service (information) exchange, action pair, or communicative act, etc. It plays connecting role between requester and provider actors in value or service exchange process or in conversation or communication process. Interaction context is virtually created only when *VB* is configured in the context. Examples in *VB* interactions are a customer demand chain and a supply chain.

In other words, *VB* is viewed as a collection of actor contexts and interaction contexts in configuration and collaboration. There exists a couple of relevant dependencies between sub-contexts. <Figure 3> shows context dependencies which are commonly up in *VBC*, that is, *ToS* dependency and *QoS* dependency. The former denotes functional dependency and the latter non-functional dependency, respectively. The couple of dependencies is especially important in configuring an interaction context model.

In <Figure 3>, there are two different types



〈Figure 3〉 Context dependency

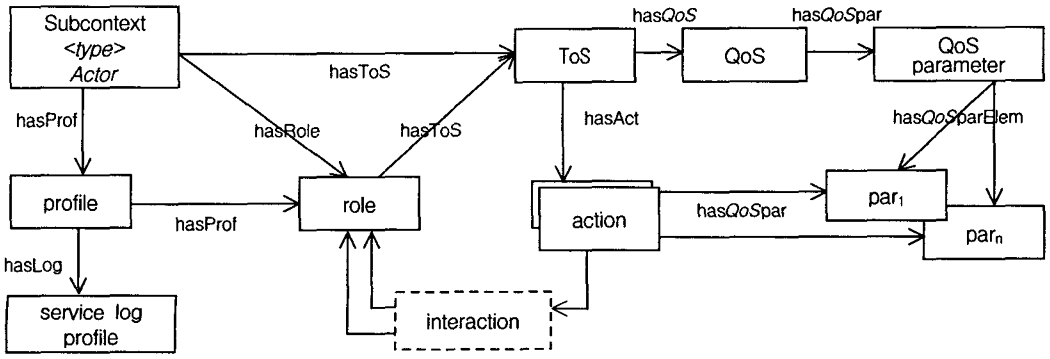
of interaction sub-context, each of which belongs to a demain chain and a supply chain. Other dependency may exist between/among contexts such as trust dependency.

3.2 Subcontext Actor

In general actors have several roles, each of which fulfill a set of tasks in different business context. For example, an organization may play a couple of product or service providers on-line and off-line. Customers may make an order as one of the customers and give his or her assessment about *QoS* to be taken as a kind of evaluators. It is therefore possible for an organization to play simultaneously two or more business roles without specifying business context. Furthermore *VB* context contains many actor and interaction subcontexts and a specific actor would be able to join two or more of them with different roles. For example, a specific organization may be members of a design chain and a demand chain at the same time in *VB* context. However, subcontext actor should have one and only one role within one type of subcontext (it may have any other role within the different type of subcontext) and come up with possible types in *VB* context. So each actor has a finite set of multiple tuples, <*type of subcontext, role,* ......> and instantiate actor subcontext in couple with interaction subcontext which specifies its type in provided or required service specification. Each role is associated with tasks, each of which has *QoS* properties. Service usage information creates its record in service (log) profile in order to reuse it for next usage. Subcontext *Actor* is briefly described as follows (see <Figure 4>).

Definition 1. Subcontext $Actor_{type}$: = $< R, P, T, Q, f_1 >$, where

- R is one and only one role which is corresponded to the type given,
- P is a usage profile which includes service usage log information in a specific context,
- T is *ToS* which has a finite set of actions and a finite set of *QoSs*
- Q is *QoS* which consists of a finite set of property tuples <parameter, value>
- f_1 is a function which maps from *ToS* to *QoS*.



<Figure 4> Subcontext Actor model

<Table 1> A partial list of concepts in Actor context (VSR1)

Actor <Type>	Role	ToS	Action	Context Info

VSR_1 <Collector>	UserGroupAgent ($VSR_m^1, \cdots, VSR_m^n$)	ManageDemand(*t*)
		ManageOrder(*t, VSP*)
		ManageContextGroup(*t*)
		ManageContextIndividual(*t*)
		………
			CollectDemand(*t*)
			CollectContextIndividual(*t*)
			ComputeContextGroup(*t*)
			OrderDemand(*t, VSP*)
			………
				Time
				Cost
				Location
				Profile
				Etc.

Example : The following table Table1 shows a partial list of a collector agent : *VSR* Actor Context of customers, $\{VSR_m^1, VSR_m^2, \cdots, VSR_m^n\}$, in a specified time *t*. *VSP* and *ESP* may be described as Actor contexts in the similar way of VSR_1.

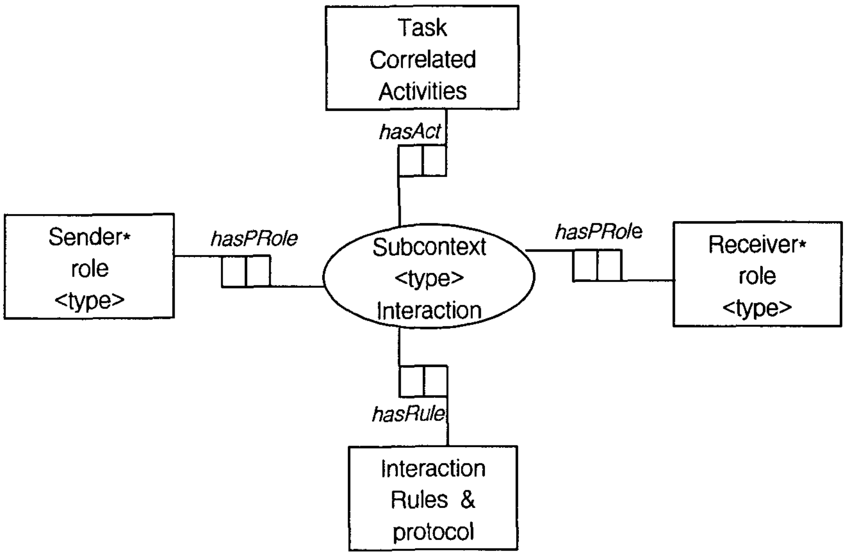
3.3 Subcontext Interaction

In *VB* context, we explained interaction as a unique association between a single sender role and a single receiver role in service exchange activity. In another words interaction connects one actor context with a consumer role and another actor context with a provider role. There is some degree of duplication in role semantics between actor and interaction subcontexts [5]. We solve the problem with the separation of capacities and requirements. Actor subcontext models available capacities, but interaction subcontext models requirements necessary for performing activities. Therefore, interaction subcontext is a sort of requirement specification in subcontext. When a specific *VB* is created, it is immediately followed by interaction subcontext which connects *VSR* and *VSP* in *VB* context (we call this *front interaction*). Interaction ends its life cycle in synchronization with *VB* life cycle. During fulfilling *VB*, lots of interactions should be generated in accordance with necessity of *EBS*s, which depends on *ToS* and *QoS* of *VSP* (*VSPm*) (we call this *back interaction*). Whether front or back, each interaction subcontext has its own life cycle and is represented with interaction rules and protocols. Interaction protocols describe valid sets of action and interaction sequences among actions that meet specific purposes related to tasks or activities. Interaction rules define the set of conditions required for successful interactions among action supported by actor contexts. Subcontext *Interaction* may be briefly described as listed below:

Definition 2. Subcontext $Interaction_{type} := < R, T, M, f_2 >$, where

- *R* is a role specification which is composed of {*receiverRole, senderRole*},
- *T* is *ToS* specification which consists of a series of correlated activities $\{Activity_1, Activity_2, \cdots, Activity_n\}$,
- *M* is a set of interaction rules and protocols, $\{Rule_1, \cdots, Rule_m ; M_1, \cdots, M_n\}$,
- f_2 is a function which maps from *R* to *M*.

<Figure 5> shows a partial subcontext in


Task Correlated Activities
hasAct
Sender* role <type>
hasPRole
Subcontext <type> Interaction
hasPRole
Receiver* role <type>
hasRule
Interaction Rules & protocol

<Figure 5> Subcontext Interaction model

<Table 2> A partial list of concepts in Interaction context (IR1)

Interaction <Type>	SenderRole	ReceiverRole	Activities	Rules Interactions
VSR-VSP <DemandChain>	Requirements (*ToS, QoS, OtherContextInfo*) of *VSR* in IR_1 ($VSR_m^1, \cdots, VSR_m^n$)	Requirements of (*ToS, QoS, OtherContextInfo*) *VSP* in IR_1 ($VSP_m^1, \cdots, VSP_m^n$)	DoAcceptanceTest(*VSR, t*) DoAcceptanceTest(*VSP, t*) SendTransition(*t*) ReceiveTransition(*t*)	TransactionGroup BusinessTransaction ServiceExchange ActionPair BusinessAct

teraction model which is represented using Object-Role Modeling method (*ORM*) [16]. In the model, each rectangle represents a corresponding concept and each link a corresponding dependency relation with properties. Therefore, *VBC* is a *VB* network model of a finite set of actor subcontexts and a finite set of interaction subcontexts, being in valid configuration of *ToS*(*QoS*) based on context information.

Example: The following <Table 2> shows a partial list of an Interaction context: *VSR-VSP* interaction (demand chain), in a specified time t. *VSP-ESP* interactions may be further described as interaction contexts in the similar way of VSR_1.

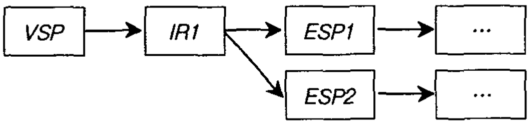
4. CONTEXT-DRIVEN MODELING FOR VIRTUAL BUSINESS CONFIGURATION

In this section, we present a context-driven *VBC* model using the subcontext concept of actors and interactions. In the modeling procedure, a couple of *ToS* (functional) and *QoS* (non-functional) dependencies play a critical role in linking actor and interaction subcontexts. With the dependencies, they decide how to compose *VSP* or *ESP* and how to couple the two or more in *VBC*. We call these procedures subcontext composition, as will be briefly reviewed in next.

4.1 Subcontext Linking and Composition

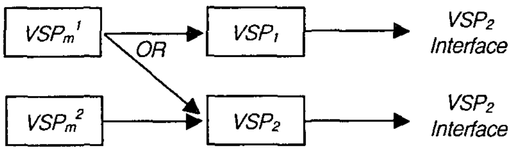
In order to collaboratively provide *VBS* through a chain of subcontext interaction processes, several business partners (*VSPm* or *ESPm*) need to virtually compose *VBP* or *ESP* satisfying *ToS* and/or *QoS* requirements. We make use of two kinds of compositions, that is, *horizontal* and *vertical*.

Horizontal composition(*H-composition*) is to horizontally connect interaction subcontext and actor subcontext based on service requirement specification of *ToS* and *QoS* (We call this *subcontext linking*). For example, these are *VSR-VSP* and/or *VSP-ESP* linking. It is easy to imagine the procedure like assembling a customer demand chain or a supply chain (see <Figure 6>). Basically, this is done by how to assign the whole required tasks to internal and external ones with taking into *ToS* functionalities and *QoS* metrics consideration each. In subcontext interaction, it is modeled as a series of alternating couples of actor subcontext-interaction subcontext until there needs no external service. *H*-composition requires the whole *QoS* computation of *VBS* from each *QoS* of *VSP*s and *ESP*s or vice versa.



〈Figure 6〉 H-composition

Vertical composition(*V-composition*) is to vertically compose actor subcontext such as *VSP (or ESP)* from possible members *VSPm* (*or ESPm*) with considering *ToS* and *QoS* like composing a virtual team with a single service interface (see <Figure 7>). For example, these are the case of *VSP* which hierarchically includes another *VSP* or members *VSPm*. In view of modeling subcontext interaction, it differs from *H*-composition in that the procedure is done through single node in a *VB* value chain (We call this *context composition*). The *V*-composition procedure is to allocate *VSP* tasks (or *ESP* tasks) to one or more *VSPm* (or *ESPm*). It needs predicting *VSP QoS* (or *ESP QoS*) during the time period of service fulfillment from *VSPm QoS* (or *ESPm QoS*) or vice versa.



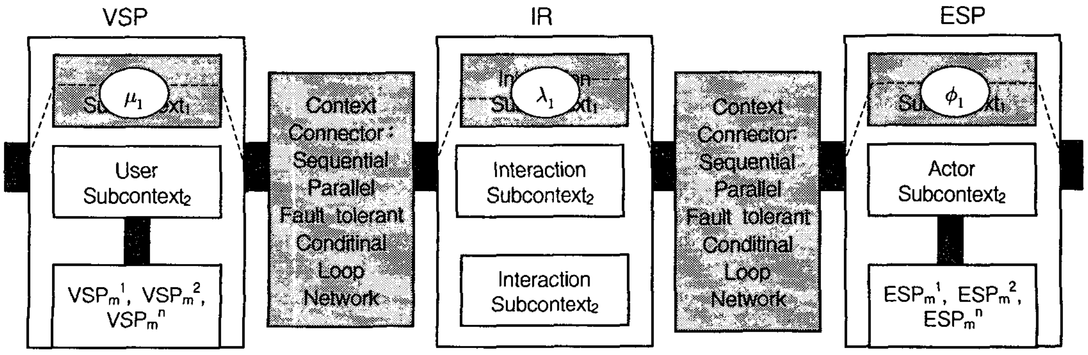
〈Figure 7〉 V-composition

4.2 QoS Computation

We mentioned in the previous that *QoS* computation models are necessary for linking and/or composing subcontexts. *QoS* in *VB* context depends generically on various influence factors of task structure, task sequencing, and service transition between two subcontexts as well as each task *QoS* itself. *QoS* computation may be described as computation (or prediction) of *QoS* for an entire or partial network of *VBS* context based on subcontext

QoS. Basically in order to solve the problem there must be resolved a couple of computational cases of a) computing an internal *QoS* level of single subcontext (what is called *intra-subcontext QoS*) as well as b) computing a combined *QoS* level of multiple subcontexts (what is called *inter-subcontext QoS*). For the former, Cardoso et al. [6, 8] presented the mathematical reduction algorithm which repeatedly applies a set of reduction rules to a specific task structure until only one atomic task remains. We apply this algorithm to an internal task structure of actor subcontext and then can represent a single node of subcontext with *QoS level* (*intra-subcontext QoS*). A task structure which is possibly reduced to a single node is the structure which consists of the following six building blocks of tasks, that is, sequential, parallel, conditional, loop, fault-tolerant, and network. For the latter, Cardoso et al.[6, 8] presented the predictive *QoS* model that makes it possible to compute *QoS* for workflows automatically based on atomic *QoS* attributes. This model computes inter-subcontext *QoS* in *VB* context based on service usage log profile. In the model a total *QoS* value depends on each *QoS* value of actor and interaction subcontexts in *VBC*, which depends successively on each *QoS* in task structure within internal subcontext. <Figure 8> shows the schematic diagram of the *QoS* computation model in a simple *VB* chain. In the figure, it is given a simple example of the customer demand chain which is composed of *VSR* and *VSP*, each *QoS* of which depends on the subcontext and its internal task structure to be chosen. First of all, for each subcontext intra-subcontext *QoS* must be computed and inter-sub-context *QoS* is then predicted in entire *VB* chain. We can represent *QoS* computation diagram, as can be seen in the <Figure 8>.

Example: Consider the following simple *VB* network of *VSP* (VSP_m^1, VSP_m^2) and *ESP* (ESP_m^1, ESP_m^2), and the *QoS* computation procedure is then given in <Table 3>. It may be also computed from the *QoS* computation model for other kinds of *VB* business network configurations or other kinds of *QoS* parameters like cost, reliability, or trust.


VSP
IR
ESP
μ_1
λ_1
ϕ_1
User Subcontext2
VSPm1, VSPm2, VSPmn
Context Connector: Sequential Parallel Fault tolerant Conditinal Loop Network
Interaction Subcontext2
Interaction Subcontext2
Context Connector: Sequential Parallel Fault tolerant Conditinal Loop Network
Actor Subcontext2
ESPm1, ESPm2, ESPmn

<Figure 8> QoS computation model(supply chain)

4.3 Context-Driven VBC Modeling Procedure

In this subsection, we give the overall modeling procedure of context-driven *VBC* based on actor and interaction subcontexts. *VB* starts with *VSP* who perceives business opportunity in a market or *VSR* who requests products or services to *VSP*s. *VSP* have first to analyze *VSR* subcontext as well as *VSP* subcontext (capacities of *VO* services) and/or *ESP* subcontext (capacities of external services) in terms of business or technological point of view. Once *VB* has been assessed *feasible*, the *VBC* procedure should be started as follows.

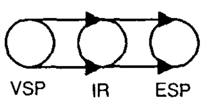
1) Situation Analysis and Modeling VSR Subcontext

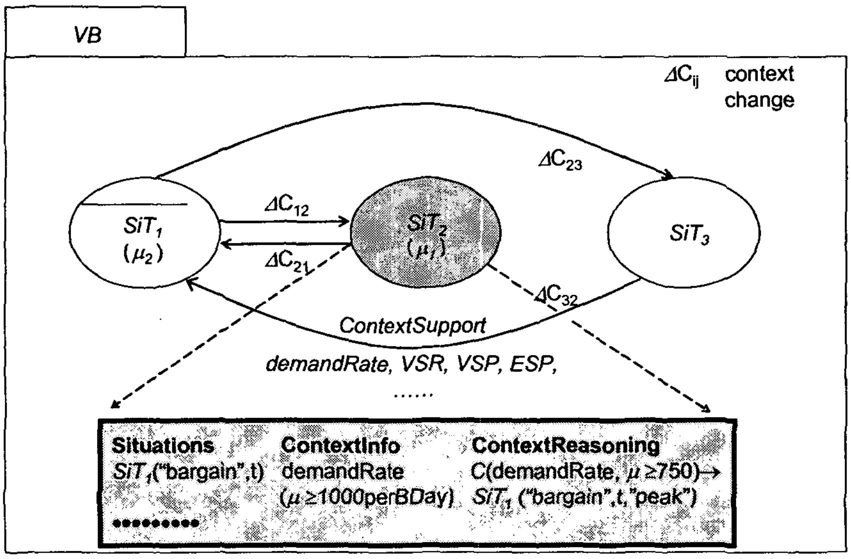
First thing that should be done about *VBC* is to analyze *VB* global context which means possible business scenarios and cases in it. <Figure 9> shows the situation network model of *VB* global context. In the figure, there are three different situations, each of which gives different demanding rate to *VSP*, as listed in the following.

① *SiT₁*: regular demanding rate ($\mu \geq 500$ and < 1000 per business day),

② *SiT₂*: peak demanding rate ($\mu \geq 1000$ per business day),

〈Table 3〉 QoS computation example (QoS:time)

VB network	VSP	IR	ESP
 QoS(t) = T(VSP)+2*T(IR)+T(ESP)	OR(AND(VSP_m^1,VSP_m^2), VSP_m^1) T(VSP) = Choice{ t(VSP_m^1), (t(VSP_m^1)+t(VSP_m^2))	AND{Transition1 : t(IR), Transition2 : t(IR)} /* timedTransition */ T(IR) = 2*t(IR)	OR(ESP_m^1, ESP_m^2) T(ESP) = Choice{ t(ESP_m^1), t(ESP_m^2))



Situations	ContextInfo	ContextReasoning
SiT_1("bargain",t)	demandRate ($\mu \geq$1000perBDay)	C(demandRate, $\mu \geq$750)→ SiT_1 ("bargain",t,"peak")
●●●●●●●●●●		

〈Figure 9〉 Situation network of VB context

③ *SiT_3*: low demanding rate ($\mu < 500$ per business day).

Once the initial *VB* context *SiT_i* has been determined as "peak" by context reasoning, it figures out initial *VSR* subcontext *and VSR_m^i*, $i = \{1, 2, \cdots, n\}$. A transition from *SiT_i* to *SiT_j* can be occurred from external policy change such as bargaining or internal context change such as increasing or decreasing demand rate in *VB*. Whatever the change has been occurred, it causes *VSR* context change and then *VSP* and *ESP* subcontext changes successively. The sort of context changes are detected by monitoring relevant context information and proactively dealt with switching and reasoning subcontexts. In the example given, an initial *VSR* subcontext assumes to be *SiT_2* which is the "peak" demanding rate.

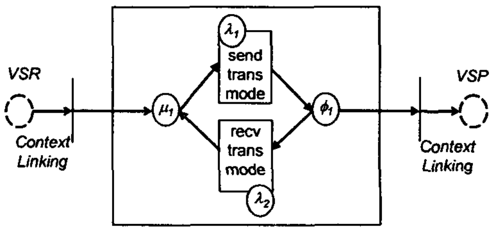
2) Modeling Interaction Subcontext

Once initial *SiT_i* and *VSR* subcontext have been figured out, it should make an initial customer demand chain which includes interaction subcontext. As shown in <Figure 5>, interaction subcontext performs a couple of things as listed in the following.

① *determining transition rules and protocols.* Interaction subcontext creates, supports, and regulates value or service transition between two binding actor subcontexts with intent and protocol (See <Table 2>). Therefore, business transition rules and protocols should be determined in interaction subcontext, for example, how to order, communicate, or serve, what kind of information to have to be given for the order, and the like. A transition can be divided to a couple of one-way transitions, {sendingTransition, receivingTransition}. In the example given service transition modes are designed as asymmetric timed transition, each of which is characterized in different *QoS* level (*QoS* transition, for example timed transition).

② *specifying provided service requirements (with QOS computation models).* Another important thing in interaction subcontext is to specify service requirements derived from *VSR* subcontext and then *VSP* requirements. This can be done with the *QoS* computation models based on service requirements and internal transition tasks.

<Figure 10> shows *VSR-VSP* interaction subcontext, which comprises a pair of transitions with transitions (λ_1 and λ_2) and specifications of both sender service role requirement (μ_1) being linked with *VSR* and provider service role requirement (ϕ_1) being linked with *VSP* subcontext.



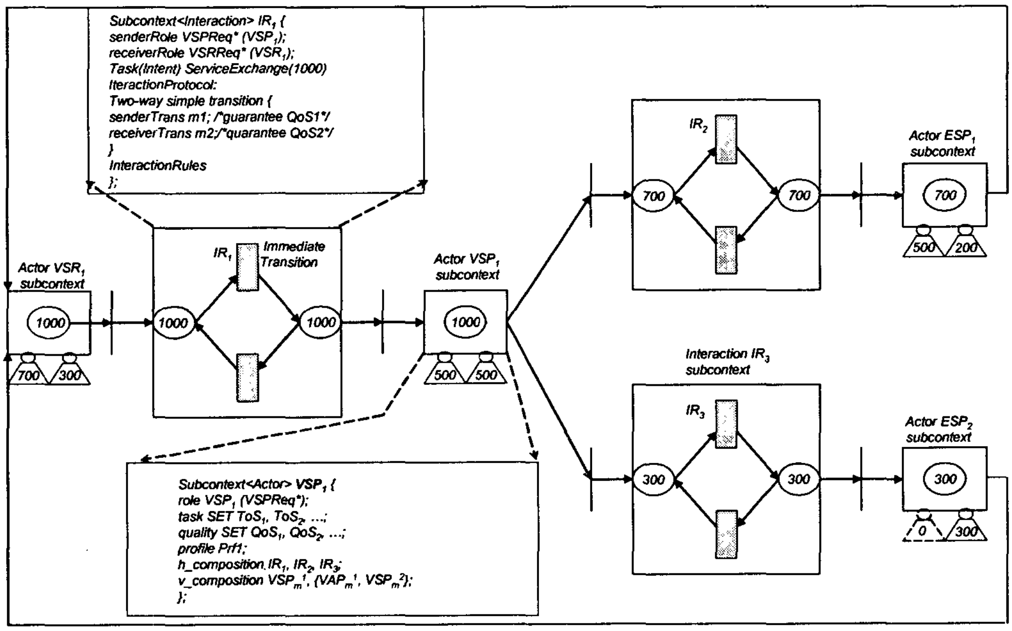
<Figure 10> An example of interaction subcontext

3) Modeling *VSP*(or *ESP*) Actor Subcontext and configuring *VSPm* (or *ESPm*)

In this stage, *VSP* subcontext (or *ESP* just in case) and *VSPm* are configured from provider service requirements specified in the interaction subcontext (In case that *VSP* successively needs other external services, repeat the modeling procedure of interaction and actor subcontexts of *ESP* and *ESPm*).

① *specifying service capacity of VSP (or ESP) subcontext.* *VSP* subcontext (or *ESP*) should be comparatively determined from provider service requirement of both interaction subcontext (ϕ_1) and possible valid configuration of *VSPm* (or *ESPm*). It may be done by prescribe-subscribe mechanism between *VSP* and each of all *VSPm*'s.

② *Allocating VSP (or ESP) service capacity to VSPm (or ESPm).* For valid configuration of *VSP (or ESP)* which comprises two or more *VSPm's (or ESPm's)*, *VSP QoS* must be allocated to each *VSPm (or ESPm).* <Figure 11> shows the overall *VBC* network of Actors *VSR* (VSR_m^1, VSR_m^2), $VSP(VSP_m^1$, $VSP_m^2)$, $ESP1(ESP_m^1$, $ESP_m^2)$, and $ESP2$ (ESP_m^1, ESP_m^2), respectively (For example, *VSR* is a service customer agent, *VSP* service provider agent, *ESPs* regional delivery agents). *VB* gets started of its *VB* process with identifying *VSR* context of "peak situation" with demand rate 1000 per business day. It creates, links, and composes relevant subcontexts alternately. *QoS* computation has been used to link and compose the relevant subcontexts.



```
Subcontext<Interaction> IR1 {
senderRole VSPReq* (VSP1);
receiverRole VSRReq* (VSR1);
Task(Intent) ServiceExchange(1000)
IteractionProtocol:
Two-way simple transition {
senderTrans m1; /*guarantee QoS1*/
receiverTrans m2;/*quarantee QoS2*/
}
InteractionRules
};
```

```
Subcontext<Actor> VSP1 {
role VSP1 (VSPReq*);
task SET ToS1, ToS2, ...;
quality SET QoS1, QoS2, ...;
profile Prf1;
h_composition.IR1, IR2, IR3;
v_composition VSPm1, (VAPm1, VSPm2);
};
```

〈Figure 11〉 Virtual business configuration for the example

5. CONCLUSION

In this paper we presented a context-driven modeling concept for virtual business configuration. Actor and interaction subcontexts have been defined as basic building blocks in configuring virtual businesses. That is, virtual business configuration was represented in the network model of inter-related subcontexts, actors and interactions. *ToS* and *QoS* dependencies have been used as major context information to switch one subcontext to another subcontext and link and compose relevant subcontexts of actors and interactions.

The proposed concepts and modeling approach are preliminary results of our ongoing research.

Additional observation and implementation work are required in order to test, verify, and refine the proposed model. More detailed descriptions of modeling elements should be refined for the proposed subcontext model of actors and interactions. Moreover context information should be additionally considered in more practical manner.

REFERENCES

[1] Andreas Schmidt, "A Layered Model for User Context Management with Controlled Aging and Imperfection Handling", *FZI Research Center for Information Technologies*, Germany, 2005.

[2] AndrZej Bialecki, "Business Context Equivalence : Using REA and UMM for Interoperability", *ECIMF Project Group Meeting*, Brussels, 2001.

[3] Anind K. Dey and Gregory D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", *Georgia Institute of Technology*, 2001.

[4] Birgit Hofreiter and Christian Huemer, "Modeling Business Collaborations in Context", *Proceedings of On the Move to Meaningful Internet Systems*, 2003 : OTM 2003 Workshops, Springer, 2003.

[5] Boriana Rukanova, "Business Transaction and Standard", PhD *Dissertation, University of Twente*, 2005.

[6] Cardoso, J., "Stochastic Workflow Reduction Algorithm", *LSDIS Lab, Department of Computer Science*, Athens, GA, University of Georgia, http://lsdis.cs.uga.edu/proj/meteor/QoS/SWR_Algorithm.html, 2002.

[7] Indulska, J., Henricksen K., McFadden, T., and Mascaro, P., "Towards a Common Context Model for Virtual Community Applications", 2005.

[8] Jorge Cardoso, Amit Sheth, John Miller, Jonathan Arnold, and Krys Kochut, "Quality of Service for Workflows and Web Service Processes", *Journal of Web Semantics*, 2004.

[9] Jose Barata and Luis M. Camarinha-Matos, "Coalitions of Manufacturing Components for Shop Floor Agility - the CoBASA architecture", *International Journal of Networking and Virtual Organisations*, Vol. 2, No. 1, 2003.

[10] Anupama Kalyan, Srividya Gopalan, and Sridhar V., "A Multicycle Context Model

Using Oracles and Micrositations for Contact Centers", *Proceedings of the Ninth LASTED* International Conference INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS, Hawaii, 2005.

[11] Karen E. Fisher, "Sanda Erdelez, and Lynne E. F. McKechnie (edited by)", *Theories of Information Behavior*, American Society for Information Science and Technology, 2005.

[12] Marielba Zararias, et al., "Modeling Contexts for Business Process Oriented Knowledge Support", *Springer-Verlag* LNAI 3782, pp. 431-442, 2005.

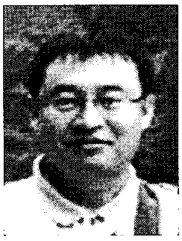
[13] Maus H., "Workflow Context as a Means for Intelligent Information Support", *Third International and Interdisciplinary* Conference on Modeling and Using Context, Springer-Verlag LNAI 2116, pp. 261-274, 2001.

[14] Oliver Brdiczka, Patrick Reignier and James L. Crowley, "Supervised Learning of an Abstract Context Model for an Intelligent Environment", *Joint SoC-EUSAI conference*, France, 2005.

[15] Tae-young Kim, "Meta-model Driven Framework for Modeling Virtual Enterprises", *PhD Dissertation*, POSTECH, 2005.

[16] Halpin, T. A., "Information Modeling and Relational Databases : From Conceptual Analysis to Logical Design", *Morgan Kaufman, San Francisco*, 2001.

▨ 저자소개



이 경 휘

서울대학교 산업공학과(B.A.), 한국과학기술원 산업공학과(M.S.), 포항공과대학교 산업공학과(Ph.D.)를 졸업하였고 국방과학연구소 연구원과 한국생산성본부 경영전문위원을 거쳐 현재는 대전대학교 IT경영공학과 부교수로 재직중이다. International Journal of Production Research, Computers In Industry, Computers and IE, Internal Journal of Engineering Management 등에 논문을 발표하였으며 Collaborative Network, Social Network, Virtual Organization, e-Transformation 등에 관한 연구를 수행하고 있다.

오 상 봉

서울대학교 경제학과(B.A.), 한국과학기술원 경영과학과(M.S., Ph.D.)를 졸업하였고 한국통신 본사 경영전산부장, 연구개발단 요금전략연구실장을 거쳐 현재는 대전대학교 정보통신공학과 교수로 재직중이다. International Social Work, Fuzzy Sets and Systems, Expert Systems with Applications, Information Processing Letters, Decision Support Systems, Pattern Recognition Letters 등에 논문을 발표하였으며 인공지능기법을 이용한 예측방법론, 인공지능분야의 알고리즘, 지능정보시스템 및 전자상거래에 관한 연구에 관심을 갖고 있다.

◆ 이 논문은 2006년 10월 02일 접수하여 1차 수정을 거쳐 2006년 12월 06일 게재확정되었습니다.