VM-FLOW:
Using Web Services Orchestration and Choreography to Implement a Policy-based Virtual Marketplace

Ivo J. G. dos Santos and Edmundo R. M. Madeira
Institute of Computing - UNICAMP – University of Campinas
13083-970 - Campinas, SP, Brazil
{ivo, edmundo}@ic.unicamp.br

Abstract: Companies are daily trying to find new ways to cope with the increasing competitive pressures imposed by the global economy. Static and huge enterprises are being replaced by dynamic business networks where each participant offers to the others specialized services. *Service-Oriented Computing* (SOC) is being considered by many as a very interesting technological solution to the new B2B interactions introduced by this economical scenario. This paper presents a Virtual Marketplace infrastructure, called VM-Flow, which supports Dynamic Virtual Enterprises, is workflow-based and introduces a series of interaction policies that treat aspects like autonomy and privacy. Also, Service Composition is shown as a suitable solution to implement these policy-based interorganizational interactions. Some issues on the developed prototype are discussed and an application built over it is described.

Key words: Service-Oriented Computing; Orchestration and Choreography; Virtual Enterprises; Marketplaces; e-Business; Workflow; Interaction Policies.

1. INTRODUCTION

The digital and global economy represents a daily challenge to companies. They need to find new ways to cope with increasing competitive pressures imposed by the market. One of the main goals is to reduce costs, and, therefore, increase sales, always maintaining (or even improving) the quality of the products and services [OT01].

Static and huge enterprises are being replaced by dynamic business networks where each participant offers to the others specialized services. Traditional technological infrastructures previously managed and owned by a single enterprise are giving way to networks of applications, whose control is distributed among many business partners [CKMT03]. On this context, *Service-Oriented Computing* is being applied by many as one good technological solution, mainly because of the way SOC treats the heterogeneity introduced by these new business networks. Services become then the basic building blocks for the construction of applications through the use of *Service Composition.*

On the e-Business/e-Commerce field, concepts like Virtual Enterprises (VE) and Virtual Marketplaces (VM) have already been applied for some years as a way to improve the quality and efficiency of the Business-to-Business (B2B) interactions. The VEs in particular allow the distribution of the business processes among different partners, trying to reduce the time-to-market and operational costs. They also permit companies that in the past could only reach local markets to operate, sometimes, on global scale [OT01].

We put together these two approaches, Service-Oriented Computing and Virtual Enterprises/Marketplaces, and propose a model for a *Dynamic Virtual Marketplace* (called *VM-Flow*) that offers supporting mechanisms for all phases of an e-Business process (including both inter- and intra-organizational aspects). The VM-Flow is workflow-based and its control is decentralized – the process instance (a *case*) carries the execution plan and moves with it from host to host [SWME00] (respecting some privacy issues that will be later discussed), what brings scalability to the infrastructure. Also, all basic services needed for the creation and maintenance of the Dynamic Virtual Enterprises (DVEs) are offered by the VM-Flow.

The main contribution of this work is the proposal of a set of interaction policies between the marketplace and its business partners (the DVE members – "real world" companies). Also, differently from other works in the area [BBS98, OT01], our infrastructure is implemented through the use of *Orchestration* and *Choreography* of Web Services.

This article is organized as follows: Section 2 presents some concepts related to our work; Section 3 introduces and discusses the VM-Flow model, the Interaction Policies and also shows how the Orchestration and Choreography are performed; in Section 4 some infrastructure issues are presented; an application built over the platform is shown in Section 5; Section 6 presents the final considerations and suggests some extensions to this work.

2. BASIC CONCEPTS

In this section we present the Virtual Enterprises, the Marketplaces and also how Service Composition is achieved using Orchestration and Choreography.

2.1 Virtual Enterprises and Marketplaces

The Virtual Enterprises (VEs) represent a set of entities geographically distributed, probably functionally and culturally different, and linked through Information Technology (IT) mechanisms. They share competencies, infrastructure and business processes and have as main goal to fulfill some specific market necessity. The VEs may be classified in two groups [Ouz01]:

- *Static Virtual Enterprises* (SVEs): in this category the relationships between partners are static, pre-configured and can't be easily changed.
- *Dynamic Virtual Enterprises* (DVEs): this category is an evolution of the SVEs. The DVEs take advantage of the opportunities offered by the Internet and by the global economy. They may have short lifecycles, dynamic business relationships between partners and a flexible and autonomous behavior.

A more recent approach to automate the creation and management process of a VE is the use of a Virtual Marketplace (VM). The potential members register their resources and business processes on the VM. The marketplaces are very important to keep the competitiveness of the VEs [OT01]. These centers offer, besides the basic infrastructure, different types of services to the VEs, increasing their flexibility and scalability; examples are search and mediation functions for the customers and the support for business partner selection. These services can be complemented by more advanced ones, like automated negotiation and electronic contracts.

2.2 Services and Composition

Service-Oriented Computing (SOC) is a computational paradigm that considers services as fundamental elements to the development of applications. In this context, services can be defined as open, self-described components that support a fast and low cost development of distributed applications [PG03]. The application of SOC on the web is manifested through the *Web Services* technology.

A *Web Service* can be defined as an interface (or a port) to some functionality performed by an application behind it (note that the way the application implements this functionality is not important at all to a service client). This interface is described and accessed through some Internet standards and protocols, like XML, HTTP, SOAP and WSDL.

A description of a service is used to publish what the service offers, its interface, behavior and quality. Service clients (organizations that act as final users) and service aggregators (organizations that compose multiple services into new ones) use these descriptions to achieve their objectives [PG03]. On the *Web Services* world, descriptions are based on WSDL *(Web Services Description Language)* [W3C04a], an XML-based language proposed by the W3C *(World Wide Web Consortium).*

To build a composition, services are combined following a certain pattern, in order to achieve a business goal, solve a scientific problem, or provide new service functions in general. These compositions may themselves become new services, what makes composition a recursive operation [CKMT03].

When composing *Web Services,* two approaches are usually considered: *orchestration* and *choreography.* The difference between these concepts is, sometimes, not so clear [Pel03], but there are some characteristics that may help this differentiation:

- In the *orchestration* approach, all interactions that are part of a business process (including the sequence of activities, conditional events etc) must be described, like on a traditional workflow system. This description is then executed by an orchestration engine, which has control of the overall composition;
- The *choreography* approach is more collaborative and less centralized in nature. Only the public message exchanges are considered relevant – and more, each service only knows about its own interactions and behavior. Differently from Orchestration, there is not an entity that has a global view/control of the composition.

While at the first moment the choreography approach seems to be more interesting because of its collaborative advantages, there are some scenarios where a global view of the process being performed by the composition is necessary (or even mandatory) - that's where orchestration comes as an affordable solution.

Regardless of the nature of the composition (orchestration or choreography), some questions should be considered when building a composite service:

- Can the interactions happen in any order?
- If no, which rules govern the sequence of interactions?
- Is there any relationship between messages sent and/or received?
- Are there a "beginning" and an "end" on a given sequence of interactions?
- Can a given sequence of interactions be undone?

- Is it possible/necessary to draw a global view of all message exchanges?

Two important specifications in the area of *Web Services* composition are BPEL4WS [BIM+03] and WSCI [BIS+02].

BPEL4WS *(Business Process Execution Language for Web Services),* or simply BPEL, is a specification published by IBM, Microsoft and BEA that models the behavior of Web Services inside a business process applying workflow concepts. It defines a language based on XML that describes the control logic required to coordinate the participant Web Services on a process flow. This description can then be interpreted and executed by an orchestration engine, controlled by one of the participants. This engine coordinates the different process activities and takes care of compensation mechanisms when errors happen. BPEL is, essentially, a layer over WSDL.

WSCI *(Web Services Choreography Interface)* is a specification from SUN, SAP, BEA and INTALIO that defines an XML-based language to describe Web Services choreography. An important aspect of WSCI is that only the visible behavior is described – WSCI does not treat the definition of executable private processes like BPEL. A WSCI-based choreography includes a set of WSCI documents, one for each partner. There is no process controlling the composition globally.

There is also an on-going effort being held by the W3C to establish a choreography standard language: WS-CDL *(Web Services Choreography Description Language)* [W3C04b]. Its first working draft has been published by the end of April 2004. WS-CDL, like BPEL and WSCI, is an XML-based language that describes peer-to-peer collaborations of *Web Services* by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal. WS-CDL does not treat executable processes (like BPEL), but only the choreography aspects of a composition.

3. INFRASTRUCTURE MODEL

The VM-Flow model, its facilities and the set of Interaction Policies we propose are presented next on this section.

3.1 The Virtual Marketplace

The VM-Flow is composed of a set of facilities, each one responsible for specific tasks that are necessary to support the DVEs and their interactions [SM03]. The infrastructure scheme is shown in Figure 1. The facilities that are part of the infrastructure are:

- *MPCI* (Marketplace Customer Interface): it is the interface between the VM-Flow and the customers that wish to acquire some product or service;
- *MPRS* (Marketplace Repository Set): consists of a set of repositories and services, responsible for the storage of different data sets (product and service catalogs, contracts, infrastructure information, history and backup data, auditing information, among others);
- *VBM* (Virtual Business Manager): the VBMs are the coordinators of a determined business process. They are responsible for tasks such as building a proposal, writing an execution plan and also the selection (or creation) of a DVE to a give process. The VBMs are grouped into agencies. There can be different kinds of VBMs, derived accordingly to the necessities of a specific business sector. A VBM can manage various business process instances, but given an instance, there is only one VBM associated to it;
- *DVEC* (Dynamic Virtual Enterprise Coordinator): each DVE has one (and only one) DVEC associated to it during its whole lifecycle. The DVEC is responsible for: 1. Selecting members ("real" enterprises) to a DVE; 2. Managing the contracts among those members; 3. Coordinating the interactions between the members and the VM-Flow; 4. Applying the execution plan prepared by the VBM; 5. Managing the entries and exits of DVE members; 6. Renegotiating dynamic plan changes when necessary.

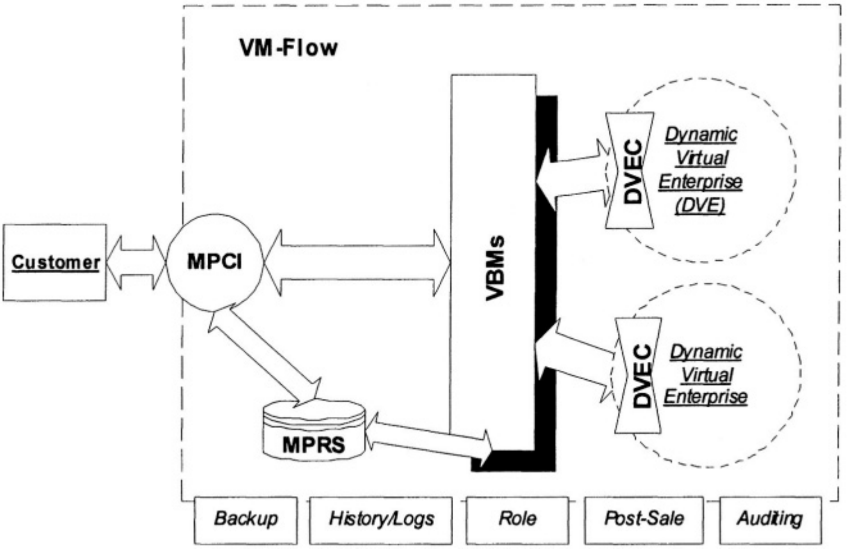


Figure 1. VM-Flow general infrastructure scheme

3.1.1 Supporting Services

The VM-Flow also offers some additional services:

- *Backup Service:* responsible for keeping security copies of the operations held on the hosts of the platform, in order to guarantee a safe recovery in case of faults;
- *History/Log Service:* together with the backup service, it is part of the fault recovery system;
- *Auditing Service (External and Internal):* used to evaluate the efficiency and integrity of the business processes executed by the VM-Flow and by the DVE partners;
- *Role Coordinator:* responsible for the resource allocation (services or people) in order to execute a determined task that is part of an execution plan;
- *Post-Sale Coordinator:* responsible to contact the customers to discover their opinion about the products/services acquired and also to manage warranty issues imposed by regional laws.

3.2 Interaction Policies

In order to guarantee a greater level of flexibility, autonomy, privacy and support different kinds of collaboration between the DVE members and the VM-Flow, our model defines two orthogonal Interaction Policy Categories: *Partner Autonomy Policies* (VM-Flow x DVE) and *Partner Cooperation Policies* (DVE member x DVE member).

3.2.1 Partner Autonomy Policies

This category defines the interaction level between the DVEC and each member of the DVE. At the moment a (real) company candidates to participate on a DVE, a negotiation takes place to define what kind of interaction it wishes to have with the VM-Flow. The DVEC then acts in one of the following manners:

- *Supervisor:* interaction through a well-defined interface with the company's private workflow – the VM-Flow does not act on the partner's inside domain. The following kinds of interaction are supported for a Supervisor DVEC:
  - Consulting-only: the DVE can only ask for status information of a process instance;
  - Selective: the DVEC and the partner negotiate in which points of the execution plan interactions will be allowed;

- Participative: the DVEC can interact with all activities of the execution plan (start, pause, resume, cancel, send/receive data, check status).
- *Coordinator:* the DVEC (through a Proxy shown later) has total control over the tasks being executed by the partner's internal workflow, which becomes an extension of the VM-Flow.

The DVEC is the responsible for determining the different policy combinations that could exist inside a DVE, based on the necessities imposed by the business process, by the execution plan and by each one of the partners.

3.2.2 Partner Cooperation Policies

The main question regarding the interaction between partners (real companies that are part of a DVE) is, usually, how to treat the privacy and integrity of data that are part of a business process instance. Through this perspective, we identify three classes of partner-partner cooperation:

- *Total Cooperation:* the two partners fully trust each other. When a process instance leaves one partner and moves to another one, it is not necessary to hide neither the plan nor the data from the previous stage;
- *Controlled Cooperation:* there is a pre-established set of information that should be passed to the next partner and another set that should be hidden by the DVEC (actually by its Proxy);
- *Total Privacy:* there is no interaction between the partners. All information is returned to the DVEC, which has access to the plan and then decides what to do next, hiding from the following partner the activities and data from the previous stage.

3.2.3 Policies and the "Real" World

As mentioned before, the two policy categories previously presented are fully orthogonal. Their selection and combination inside a DVE depend both on business questions (confidentiality of data, for example) as well as on technological limitations (compatibility level among the different partner's workflow systems, for example).

We could, for instance, associate a *Supervisor DVEC + Controlled Cooperation Policy* with an e-Business Service Provider scenario – this provider would be a third entity, independent, that offers its infrastructure and services to other companies that wish to participate on a virtual marketplace.

In another example, an automobile industry supply chain [MZ02, SRKT00] could apply a policy with a *Coordinator DVEC + Total Privacy.* In this scenario, the car manufacturer could be, for instance, the owner of the marketplace, controlling the production processes of its suppliers.

3.3 DVEC x DVE Interaction

In order to present how the composition of the activities that are part of an execution plan is achieved it is necessary to take a deeper look at the DVEC x DVE interaction.

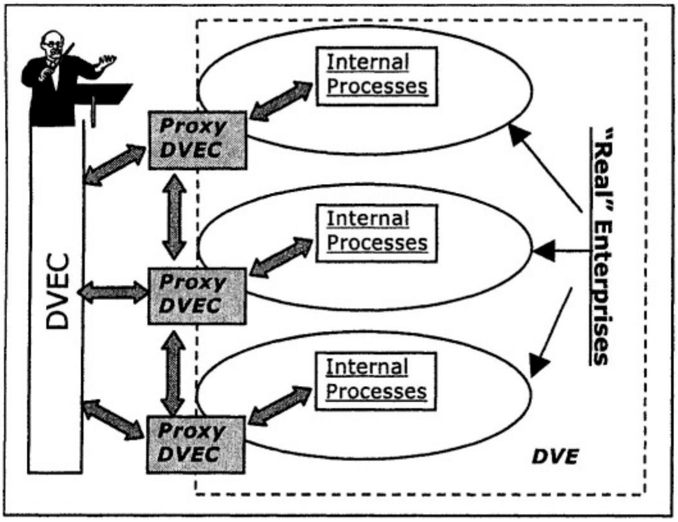


Figure 2. DVEC, DVE and the Proxies

A new element is introduced in Figure 2: the *DVEC's Proxy*. It is the responsible for implementing the interaction between the VM-Flow and a partner, executing the local portion of the plan, always according to the *Autonomy Policy* selected. This proxy must also "talk", when applicable, to the next partner (respecting the execution plan and the selected Cooperation Policy). The DVEC becomes then responsible to orchestrate the process globally through the various proxies (each one of them coordinating the orchestration/choreography of the local plan).

The DVEC and its proxies participate on two different levels of service composition:

1. The DVEC orchestrates its Proxies. The global execution plan consists of the composition of all partner proxies' Web Services;
2. Each Proxy is responsible to orchestrate (or simply participate on a choreography) of the local execution plan – according to each partner's Autonomy Policy.

3.4 Related Work

The area surrounding *e-Business, Marketplaces* and *Interorganizational Business-to-Business Interactions* is, although relatively new (pushed by the advent of the Internet), already broad and with much research being done. Due to this broadness, and in order to situate our work in the field, on this section we decided to present only the works that are closely related to our platform.

3.4.1 Marketplaces and Virtual Enterprises

DIVE. The DIVE project *(Agent-based Life Cycle Management for Dynamic Virtual Enterprises)* [OT01] proposes an infrastructure for DVEs based on mobile agents and introduces a life cycle model for the DVEs. On Table 1, a comparison between the VM-Flow and DIVE is presented.

Table 1. DIVE x VM-Flow

----	DIVE	VM-Flow
Marketplace Management	Mobile Agents	Mobile Cases + Workflow
DVEs Management	Mobile Agents / Life Cycle	Service Compos. / Life Cycle
Scalability	Yes	Yes
Interaction Policies	No	Yes
Interorganizational Aspects	Partially	Yes
Implementation	FIPA / XML / Java	Web Services / XML / Java

Even though DIVE and VM-Flow propose different approaches to implement a marketplace, the life cycle model proposed by DIVE guided the definition of VM-Flow's DVE behavior.

Other works in the VE area not directly related to VM-Flow but that present interesting solutions are [APC02, BBS98, TBV02].

3.4.2 Interorganizational System Integration

BPFA. The interorganizational system integration is discussed by [SO01], including privacy issues. It presents a classification of the business processes into *private* and *shared.* The private processes expose interaction points where the shared processes connect to, in such a way that a business process can be part of two or more organizations. A framework to support these two categories of processes, BPFA *(Business Process Framework Architecture)* is also introduced by [SO01]. The BPFA consists of a set of components that execute instances of an interorganizational process model, extending a company's workflow infrastructure and allowing processoriented communication among partners and customers. Table 2 presents a comparison between the VM-Flow platform and the BPFA framework.

Table 2. BPFA x VM-Flow

----	BPFA	VM-Flow
VE Support	Partial	Yes
DVE Support	No	Yes
Core	Workflow-based	Workflow-based
Interorganizational Interact.	WfMS[23] Responsibility	Orchestration / Choreography
Privacy and Autonomy	Private and Shared Processes	Interaction Policies

The Interaction Policies proposed in the VM-Flow platform complement the solution given by [SO01], offering, to the context of Virtual Marketplaces, more flexibility on the definition of privacy and collaboration levels between partners.

Public-To-Private. The approach proposed by van der Aalst on [AW01], called Public-To-Private (P2P), is based on the notion of inheritance. It consists of three main steps:

1. The specification of a shared public workflow;
2. The partition of this public workflow over the participant organizations;
3. The creation, for each organization, of a private workflow. This private workflow is a subclass of its respective part on the public workflow.

The P2P solution is very elegant and surrounded by a formalism called *WF-Nets* [Aal98] (derived from *Petri nets*). We believe that as P2P is an approach to model interorganizational workflows (regardless of the technology used to implement it), we could apply a Service-Oriented implementation to a P2P based workflow model (an interesting solution would be to apply *orchestration* on the private workflows and *choreography* on the public workflows).

3.4.3 Workflow Management

Table 3. WONDER x VM-Flow

----	WONDER	VM-Flow
Main Goal	Large Scale WfMS	Virtual Marketplace
Core	Workflow-based	Workflow-based
Distributed Objects Techn.	CORBA	Java/RMI
Interorganizational Interact.	No	Yes
Decentralized Control	Yes (Mobile Cases)	Yes (Mobile Cases)

---

[23] WfMS: *Workflow Management System*

WONDER. Traditional *Workflow Management Systems* have an intrinsic scalability limitation, the central server. It represents a performance bottleneck and a point of failure on systems where a great number of processes is executed simultaneously [SWME00]. To solve this limitation, the WONDER [SWME00] architecture proposes a *Large Scale WfMS* that introduces the use of *mobile cases* (business processes instances) that migrate over the system nodes, following an execution plan that they carry with themselves. VM-Flow's decentralized management model was inspired by the ideas presented by the WONDER platform. Although having different goals, a comparison between WONDER and VM-Flow is shown on Table 3.

3.4.4 Service Composition

In the area of Service Composition, [Pel03] presents a survey about Orchestration, Choreography and their main specifications and tools. On [VDDTR03] the FUSION system, a framework for dynamic composition and automatic execution of *Web Services* is analyzed.

4. IMPLEMENTATION ISSUES

We built a prototype of the VM-Flow platform that implements the main functionalities described on the model (Section 3). Our choice was to implement it on an object-oriented language – more specifically Java because of its platform independence characteristics. The access to the MPCI and to the partners' internal systems is achieved through Web Services. The orchestration is implemented based on BPEL4WS - the engine used was BPWS4J [IBM03]. Next we discuss in more details the implementation.

4.1 Platform Core

The Figure 3 presents a snapshot of the main interfaces that are part of the platform core. Besides *MPCI_I, VBM_I, VBMAgency_I, DVE_I, DVEC_I and DVECProxy_I,* related to the facilities previously presented in the model, there some other relevant interfaces:

- *BusinessProposal_I:* a business proposal, when approved, becomes the base for the construction of the execution plan;
- *ExecutionPlan_I:* the execution plan;
- *VMData_I, BusinessProcess_I, Case_I:* all data exchanged among VM-Flow objects inherit from *VMData_I. BusinessProcess_I* is used to build the business process templates, while *Case_I* represents these processes' instances;

- *RealEnterprise_I:* a "real" enterprise or a partner is a member (or member candidate) of a Dynamic Virtual Enterprise.

The *case* (business process instance), represented by the class *Case* (Figure 4), migrates from one facility to the other carrying the execution plan (note that *ExecutionPlan* is one of *Case's* attributes), what makes the control inherently decentralized. This migration is achieved through the method *ReceiveCase(),* present in many of the classes shown in Figure 4. Another important characteristic illustrated on this figure is the *DVECProxy* family class hierarchy: *DVECProxy_SP, DVECProxy_SC, DVECProxy_SS* and *DVECProxy_CC* implement the different *Autonomy Policies.* These classes are responsible for coordinating the execution of the local portion of plan, according to the selected policies, and also to coordinate the orchestration/choreography of the DVE members' internal services (note that there is one DVECProxy object associated with each DVE member).

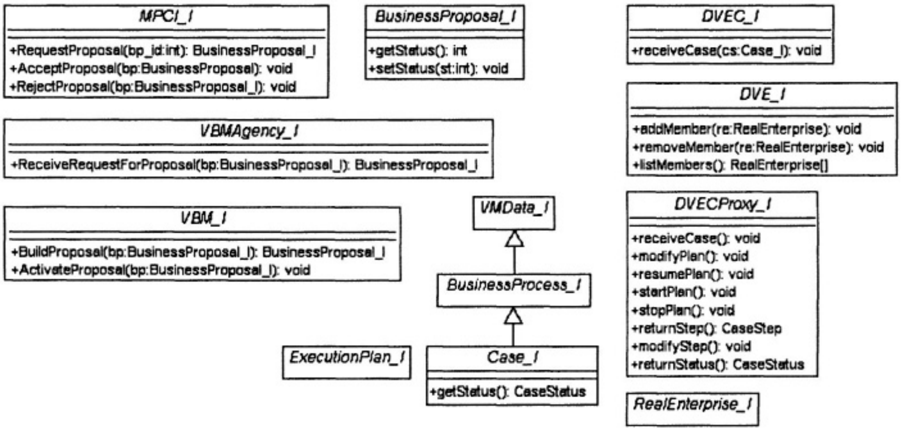

MPCI_I
+RequestProposal(bp_id:int): BusinessProposal_I
+AcceptProposal(bp:BusinessProposal): void
+RejectProposal(bp:BusinessProposal_I): void
BusinessProposal_I
+getStatus(): int
+setStatus(st:int): void
DVEC_I
+receiveCase(cs:Case_I): void
DVE_I
+addMember(re:RealEnterprise): void
+removeMember(re:RealEnterprise): void
+listMembers(): RealEnterprise[]
VBMAgency_I
+ReceiveRequestForProposal(bp:BusinessProposal_I): BusinessProposal_I
VBM_I
+BuildProposal(bp:BusinessProposal_I): BusinessProposal_I
+ActivateProposal(bp:BusinessProposal_I): void
VMData_I
BusinessProcess_I
ExecutionPlan_I
Case_I
+getStatus(): CaseStatus
DVECProxy_I
+receiveCase(): void
+modifyPlan(): void
+resumePlan(): void
+startPlan(): void
+stopPlan(): void
+returnStep(): CaseStep
+modifyStep(): void
+returnStatus(): CaseStatus
RealEnterprise_I

Figure 3. VM-Flow Core Interfaces

4.2 Orchestration and Choreography

Our model for orchestration and choreography is built over BPEL (Subsection 2.2). BPEL defines two business process models:

- *Executable Business Processes:* models the behavior of partners in a specific interaction, essentially like a private workflow. They are executed by an orchestration engine;
- *Abstract Business Processes*: business protocols that specify the public message exchanges between partners. The business protocols are not executable and do not treat internal process details.

Essentially, the executable processes offer support for orchestration while the abstract processes focus on the choreography of the services. As already mentioned on Subsection 3.3, the services composition is held on two levels:

1. A DVEC orchestrate its Proxies; each Proxy is implemented as a JAVA class that exports its interface as a Web Service. The global execution plan consists on the composition of those Web Services through a BPEL executable business process.
2. According to its partner's autonomy policy, each Proxy is responsible to orchestrate (via executable business processes) or simply participate on a choreography (using abstract business processes) of the local execution plan.

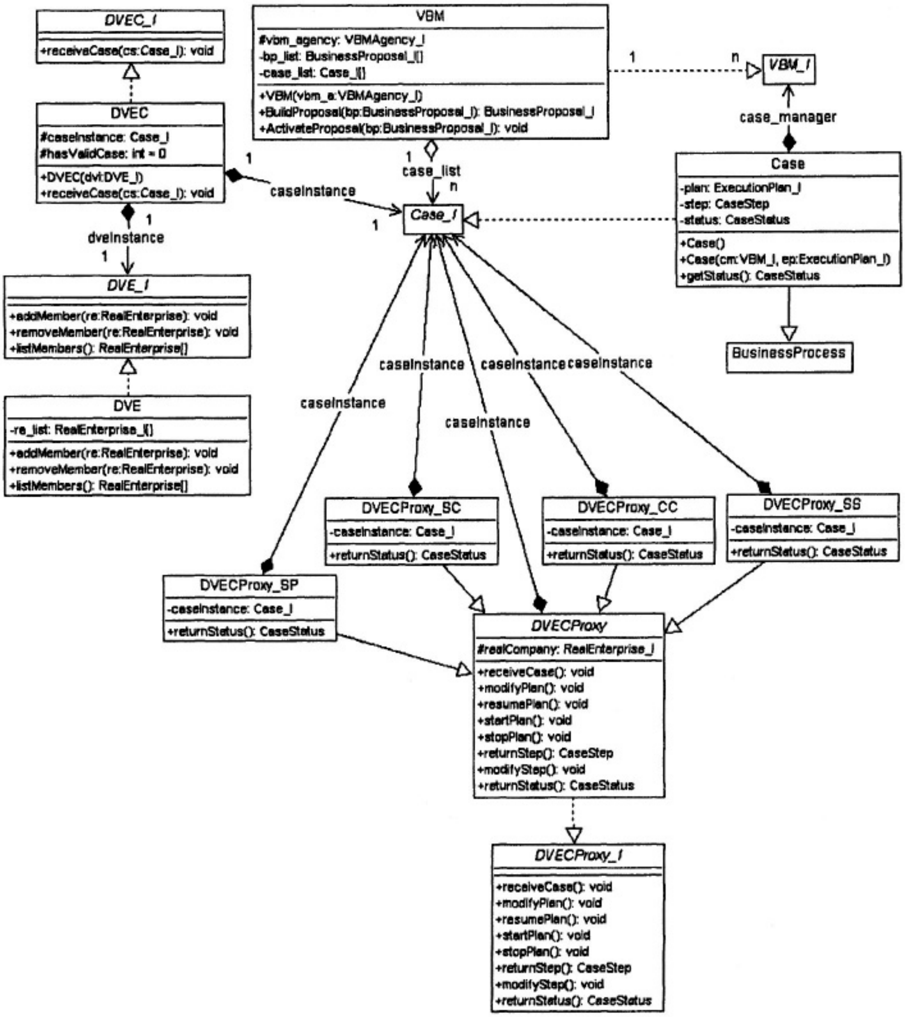


Figure 4. Platform Core Class Diagram

4.2.1 VM-Flow x Business Partners interaction

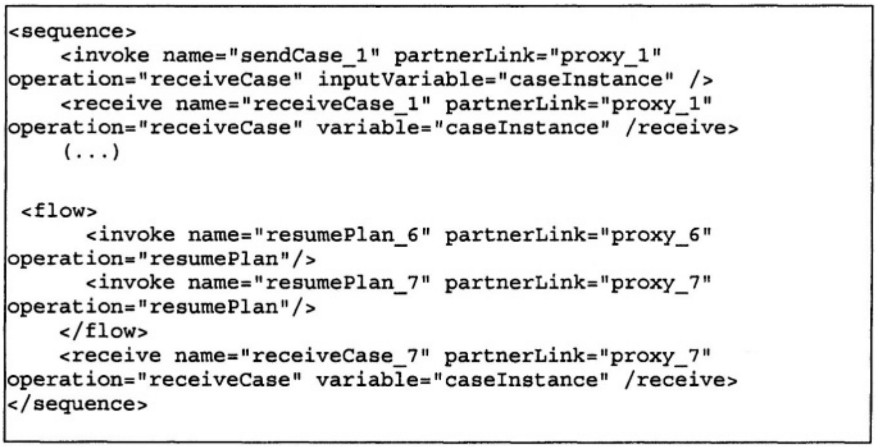
When the DVEC acts as *Coordinator,* the VM-Flow, through the DVEC's proxy, uses the BPEL's executable business process definitions to *orchestrate* the plan inside a DVE member. On the other hand, when the DVEC acts as *Supervisor,* the abstract definitions are used in a *choreography* context.

4.2.2 Partner x Partner interaction

When *Total* or *Controlled Cooperation* is the chosen policy for a partner-partner interaction, abstract business processes are useful to define the message exchanges between the proxies (*choreography* context).

4.2.3 BPEL Orchestration Example

The Figure 5 presents an example of a BPEL executable business process used by the DVEC to orchestrate its proxies. The `<sequence>` tag indicates a series of activities that should be executed one after the other, while the tag `<flow>` determines activities that should be executed in parallel; `<invoke>` presents a call to some external Web Service operation and `<receive>` prepares the BPEL process to receive a call from another Web Service. This example illustrates some migrations of case between the DVEC and its proxies (on the DVE members).



```
<sequence>
    <invoke name="sendCase_1" partnerLink="proxy_1"
operation="receiveCase" inputVariable="caseInstance" />
    <receive name="receiveCase_1" partnerLink="proxy_1"
operation="receiveCase" variable="caseInstance" /receive>
    (...)

 <flow>
      <invoke name="resumePlan_6" partnerLink="proxy_6"
operation="resumePlan"/>
      <invoke name="resumePlan_7" partnerLink="proxy_7"
operation="resumePlan"/>
    </flow>
    <receive name="receiveCase_7" partnerLink="proxy_7"
operation="receiveCase" variable="caseInstance" /receive>
</sequence>
```

Figure 5. Executable Business Process Extract

The structure of a document defining an abstract business process is similar to an executable process definition – the difference is that it is used only as a protocol to validate a given sequence of messages exchanged by other running processes instead of being executed by the engine.

5. APPLICATION EXAMPLES

In this section we present a detailed example of an application built over the VM-Flow platform. We also show other scenarios where the VM-Flow could be used.

5.1 Computer Industry

The application implemented to validate our infrastructure models one hypothetic computer industry called *LEED*. In the business model adopted, this industry is responsible only for integrating the components, not being responsible for their manufacturing. Thus, *LEED* uses the VM-Flow platform to find component suppliers that will attend its customers' needs.

In Figure 6, a general scheme of the VM-Flow use by the *LEED* industry is presented. It has the following peculiarities: the Customer can be an equipment reseller or a big corporate customer; three kinds of VBMs are defined to manage the different product categories offered by *LEED* (desktops, notebooks and servers); the Partner Candidates represent the potential component suppliers.

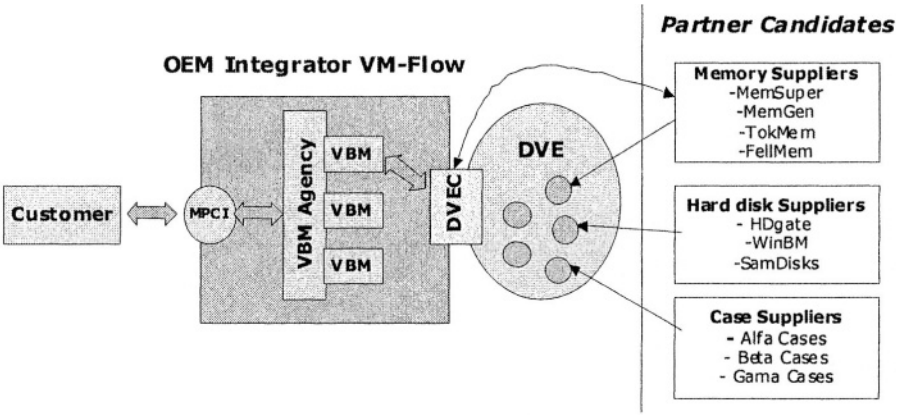


Figure 6. Application Example – *LEED* PC Industry VM-Flow

A typical *LEED* business process consists of the following steps (Figures 7 and 8 present a Sequence Diagram showing these steps):

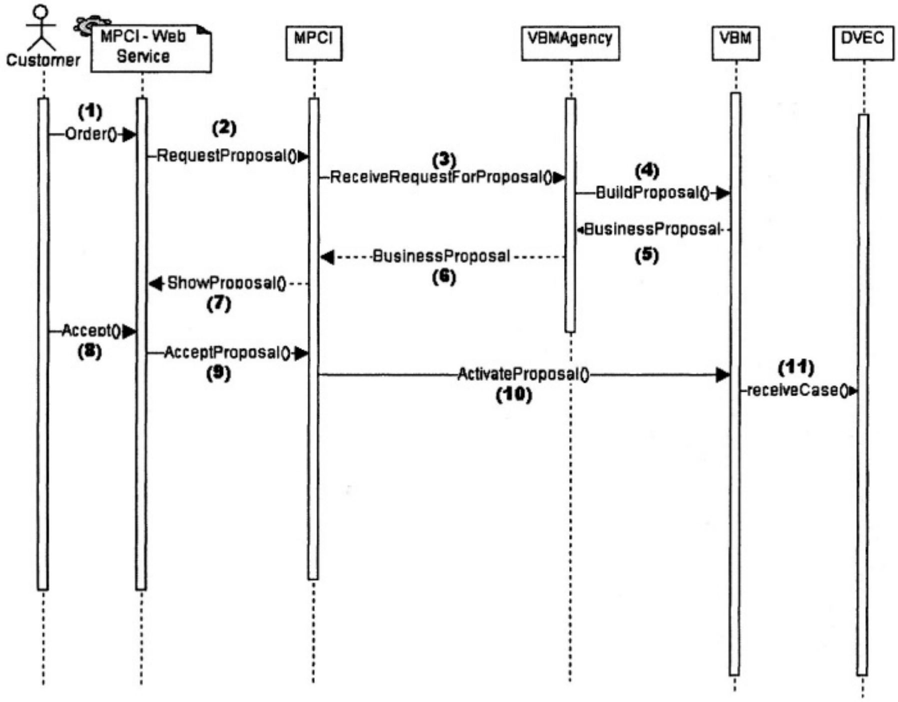


Figure 7. LEED Business Process Example – Part 1

- The *customer* interacts with the *MPCI,* consulting product information and asking for a business proposal (1,2);
- The *MPCI* contacts the *VBMAgency* (3). According to the product(s), an adequate *VBM is* allocated to handle the business proposal (4);
- The *VBM,* based on the customer needs and on the information given by the potential suppliers, builds a business proposal and a draft version of the execution plan (5,6). The proposal is then presented to the customer through the *MPCI* (7);
- In case of approval (8,9,10), the *VBM* immediately creates a *DVEC,* sending to it the draft version of the execution plan (11). This DVEC then selects the *DVE* members and finishes together with the *VBM* the execution plan definition;
- From this moment on, the *DVEC* becomes the responsible for the execution of the plan, interacting with the Proxies placed on each one of the *DVE* members (12). Note that the member associated with *DVECProxy_1* has a *Cooperative* relationship with the member associated with *DVECProxy_2,* because it sends the case directly to its partner (13). The same does not happen in the partner 2 x partner 3 relation – *DVECProxy_2* is forced to send the case back to the *DVEC,* which for-

wards it to *DVECProxy_3* (with the applicable privacy restrictions imposed by the policy between 2 and 3) (14,15,16).

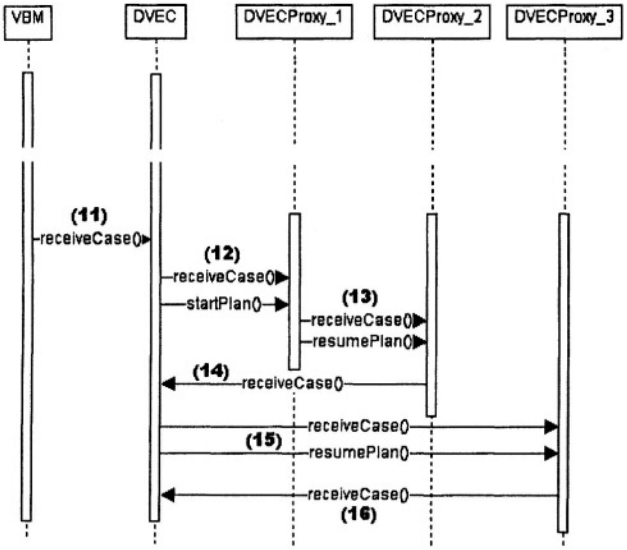


Figure 8. LEED Business Process Example – Part 2

5.2 Other Scenarios

Next we present other scenario examples in which the VM-Flow platform can be applied (besides the ones already mentioned on Subsection 3.2.3 – e-Business Service Provider and Automobilist Industry).

Tourism. The customer (or a Travel Agent on behalf of a customer) uses the VM-Flow to find hotels, air companies and car rental stores. In another example, an Agency/Operator, with the help of the VM-Flow, can build a tour that will be offered to various tourist groups. In this context, a *Consulting Supervisor DVEC* could be combined with a *Total Cooperation Policy* (on the *Hotel x Air Company x Car rental store* relationship) and *Total Privacy* (on the relationship of potentially concurrent companies – two air companies that fly to the same destinations, for example).

Civil Construction. Real-state agencies, material suppliers, entrepreneurs, engineering offices and architects can associate with a specialized VM-Flow and offer projects, building and decoration services. This scenario seems adequate for the appliance of a *Total Cooperation* policy inside a DVE (except for potentially concurrent partners – two material suppliers, for example).

Government Applications. The platform could be adapted (through VBM specialization) to offer support for public bids. Being the Government the owner of the platform, *Participative Supervisor DVECs* or even *Coordinating DVECs* could be used on this scenario.

6. CONCLUSION

This paper presents and discusses the *VM-Flow* platform, a Virtual Marketplace infrastructure that supports Dynamic Virtual Enterprises and is workflow-based. The platform offers as its main contribution different privacy and autonomy levels to its partners through various *Interaction Policies,* implemented applying resources like Orchestration and Choreography of *Web Services.* The use of BPEL4WS as a *Web Services* composition technology was successful and showed us its potentiality, even though some issues remain open and an effort to create a standard for composition and coordination of services is still in progress.

The model presented is flexible and extensible, with the goal to support different market rules and needs. Implementation aspects are discussed and also an application built over the infrastructure is presented.

Extensions to this work may include the enhancement of the infrastructure (facilities specialization and creation of new ones), adaptation of the *Interaction Policies* to specific market sectors and the proposal of new applications. Some problems introduced by dynamic discovery of partners (expressiveness and semantics, for instance) were not in the scope of this work and could be part of extension studies.

ACKNOWLEDGEMENTS

This work was funded by CAPES and CNPq.

REFERENCES

[Aal98] W.M.P. van der Aalst. The application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[APC02] P. Ávila, G.D. Putnik and M. M. Cunha. Brokerage function in Agile/Virtual Enterprise integration – A literature review. 3rd IFIP Working Conference on Infrastructures for Virtual Enterprises (PRO-VE´02), pp. 65-72, Portugal, 2002.

[AW01] W.M.P. van der Aalst and M. Weske. The P2P Approach to Interorganizational Workflows. In *Proceedings of the 13th International Conference on Advanced Informa-*

tion Systems Engineering (CAiSE'01), volume 2068 of Lecture Notes in Computer Science, pages 140-156. Springer-Verlag, Berlin, 2001.

[BBS98] M. Bichler, C. Beam and A. Segev. An electronic broker for business-to-business electronic commerce on the Internet. Int. Journal of Cooperative Information Systems 7, pp 315-331, 1998.

[BIM+03] BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems. Business Process Execution Language for Web Services (BPEL4WS) – Version 1.1, 2003; http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[BIS+02] BEA Systems, Intalio, SAP and Sun Microsystems. Web Services Choreography Interface 1.0, 2002; http://www.sun.com/software/xml/developers/wsci/sci-spec-10.pdf.

[CKMT03] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in Web Services. Communications of the ACM, 46(10):29-34, 2003.

[IBM03] IBM. Business Process Execution Language for Web Services Java Run Time, 2003; http://alphaworks.ibm.com/tech/bpws4j/

[MZ02] H. Min and G. Zhou. Supply chain modeling: past, present and future. Computers & Industrial Engineering 43, pp 231-249, 2002.

[Ouz01] V. Ouzounis. An Agent-Based Platform for the Management of Dynamic Virtual Enterprises. Doctor Thesis, TU Berlin, 2001.

[OT01] V. Ouzounis and V. Tschammer. Towards Dynamic Virtual Enterprises. Proceedings of The First IFIP Conference on e-Commerce, e-Business, e-Government (I3E 2001), pp 177-192, Zurich, Switzerland, 2001.

[Pel03] C. Peltz. Web services orchestration - a review of emerging technologies, tools, and standards. White Paper, Hewlett Packard, 2003.

[PG03] M.P. Papazoglou and D. Georgakopoulos. Service-oriented computing. Communications of the ACM, 46(10):25-28, 2003.

[SM03] I.J.G. Santos and E.R.M. Madeira. Policy-based Orchestration and Choreography of Services on Dynamic Virtual Enterprises. 3rd IFIP I3E Conference on e-Commerce, e-Business and e-Government, Research Colloquium, Brazil, 2003.

[SO01] K. Schulz and M. Orlowska. Architectural Issues for Cross-Organisational B2B Interactions. International Workshop on Distributed Dynamic Multiservice Architectures (DDMA), IEEE Computer Society Press, USA, 2001.

[SRKT00] C. Strieker, S. Riboni, M. Kradolfer and J. Taylor. Market-based Workflow Management for Supply Chains of Services, 33rd Annual Hawaii International Conference on System Sciences (HICSS-33), 2000.

[SWME00] R.S. Silva Filho, J. Wainer, E.R.M. Madeira, and C. Ellis. CORBA Based Architecture for Large Scale Workflow. IEICE Transactions on Communications, Vol. E83-B, No. 5., pp. 988-998, 2000.

[TBV02] M. Tolle, P. Bernus and J. Vesterager. Reference Models for Virtual Enterprises. **3rd** IFIP Working Conference on Infrastructures for Virtual Enterprises (PRO-VE´02), pp. 3-10, Portugal, 2002.

[VDDTR03] D. VanderMeer, A. Datta, K. Dutta, H. Thomas, K. Ramamritham and S. B. Navathe. FUSION: A System Allowing Dynamic Web Service Composition and Automatic Execution. IEEE International Conference on E-Commerce, p. 399, California, USA, 2003.

[W3C04a] World Wide Web Consortium (W3C). Web Services Description Language (WSDL) Version 2.0 – Part 1: Core Language. W3C Working Draft, http://www.w3.org/TR/wsdl20, March 2004.

[W3C04b] World Wide Web Consortium (W3C). Web Services Choreography Description Language Version 1.0 (WS-CDL). W3C Working Draft, http://www.w3.org/TR/ws-cdl-10/, April 2004.