Process Model Integration in Internet-based Virtual Software Corporations

Quentin Mair and Zsolt Haag
Glasgow Caledonian University
Department of Computing
70 Cowcaddens Road
Glasgow G4 0BA
United Kingdom
{z.haag, qma}@gcal.ac.uk

Abstract

This paper presents initial results for defining a method to integrate process models and supporting tools for the support of software development in Internet-based Virtual Software Corporations (VSCs). VSCs are emerging as the organisational form enabling the development of complex, large scale software. However, due to the heterogeneous and temporary nature of VSC interactions, no dedicated software engineering environments exist that would support VSC processes. In this paper we present an approach for specifying, reusing and integrating the processes of VSC member companies. The practical applicability of the framework is validated for an experimental VSC. The resulting process model is enacted using a commercial Internet-based e-mail workflow management package.

NOTE: This work is part of the VISCOUNT1 Esprit Project.

1. Introduction

The environment in which software is being developed has changed dramatically over the past ten years. The changes are a consequence of developments in Internet technologies and the increasing size of developer teams required for large-scale software projects. Several organisational forms, such as outsourcing and multinational corporations, are trying to address these environmental changes in order to provide the necessary competitive edge on the market. However, an emerging form, that of Virtual Software Corporations (VSCs), is enabling the development of large scale, complex software while taking full advantage of economies of scale, access to scarce resources and adaptability to market requirements [Boldyreff et al 1996]. A VSC is characterised by an often loose and flexible alliance of geographically distributed teams and organisations, with particular specialisms, who come together for a particular, often one-off, software development [Byrne 1993]. Today TCP/IP provides a natural conduit for inter-organisation communication but higher-level tool support for VSCs has still to be investigated. Such environments, characterised by a collection of heterogeneous companies, require specific tool support for process management [Hardwick and Bolton 1997] and distributed Software Configuration Management (SCM). Support (which must be built on IP) is thus required to address two fundamental areas:-

1. Integration and interoperability of existing and new process models brought by organisations forming the VSC i.e. an emphasis on process
2. Exchange of SCM Configuration Items (CIs) between organisations i.e. an emphasis on product

The emphasis of this paper is on the former.

The deployment of process management support tools in a VSC context will have to accommodate the different approaches to specify, model and enact processes in the member companies. The legally independent nature of the member companies and the necessity of applying a less centralised approach in managing processes generate

---

1 VISCOUNT (VIrtual Software COorporation UNiversal Testbed) is funded by the European Commission’s Esprit Programme (Project Number 25754) from September 1997 until February 2000. The project partners are: BAE, Glasgow Caledonian University, VTT Electronics, SIA, debis and Valmet Automation.
further issues [Riempp 1998]. These aspects define a set of requirements for process models within VSCs and will be discussed in the section 2. The requirements have identified the need for the formal definition of an atomic process element, labelled a VSC Component Process (VCP).

The practical use of the formalism has to be assessed, and thus the processes for a real VSC had to be captured. This was achieved using use case diagrams describing the desired interaction between process actors and a support environment in a software development scenario. The selection of use cases is presented in section 3.

The requirements identified in section 2 are the basis for defining a formal notation to specify processes. The purpose of this notation is to address the modelling requirements and to provide a common representation for exchanging processes, process data, CIs and CI meta-data between the different support tools. The formal notation is presented in section 4 and it is built on the process specification language (PSL) [PSL 1998a] and the resource definition framework (RDF) [W3C 1999a]. This section also presents a sample representation in PSL/RDF for use cases.

The formalised use cases are enacted using a commercial Internet-based email workflow management package. The approach taken and the results of this experiment are presented in section 5.

The identified requirements, the definition of VCPs, the formal notation for processes and the experiences of the enacted example represent the approach for specifying, reusing and integrating processes in a VSC environment. The final section of the paper presents the conclusions of this work and indicates directions on how to further this research.

2. Supporting processes within VSCs

VSCs, as defined in [Byrne 1993] are a temporary alliance of independent corporations which use the benefits of economies of scale and access to scarce resources to accommodate the changing needs of a particular market, such as large scale, mission critical software. The temporary and heterogeneous nature of VSCs implies that interaction between partners will have to be supported by tools to a greater extent than in traditional software development [Haag et al. 1997]. The role of these tools is to provide the context users require to carry out their tasks and to co-ordinate their activities [Davidow and Malone 1992].

It has been suggested that Wide Area Workflow Management can support processes in virtual corporations [Riempp 1998]. However, when considering VSCs, in particular VSCs involved in the development of mission critical systems, there is the added issue of supporting companies at different levels of maturity, as defined in [Paulk 1995], while complying with relevant quality standards.

Therefore, to allow companies to use, to a certain extent, their existing practices and to ensure the quality of the overall development process, support is required both for the top-down and bottom-up definition of processes. The top-down definition caters for critical aspects requiring an overall control, such as compliance with specific quality standards for projects. The bottom-up approach is used when integrating processes between companies, for example a configuration audit process can be included in a release manufacturing process, with the processes being performed by different companies.

Supporting both approaches to process definition lends itself to componentisation and we introduce the term of VSC Component Process (VCP) to represent an atomic process that is performed within one organisation. The use of VCPs, by encapsulating elementary process definitions, enables the integration of processes between VSC members and provides support for sharing and reusing processes. This aspect is important in the context of companies using different process support tools.

However the fact that the VSC contains VCP instances which do not communicate with other VCP instances during internal execution means that synchronisation will take place at defined points i.e. at the start and finish of the VCP instances. Therefore, once VCPs have been defined within a VSC, the interfaces, the interaction between processes has to be defined and controlled. In [Riempp 1998] several forms of control are presented, from these equal partnership is best suited for peer partners in a VSC. In essence this means that interfaces between VCPs are defined and agreed at VSC inception; any changes have to be renegotiated. This joint modelling exercise is similar to the process undertaken in generating the use cases and use case diagrams in defining the requirements for the VISCOUNT project, and will be discussed in a later section.

Another requirement for supporting processes within VSCs is a result of having to accommodate distinct tools used by member companies. VSC partners will bring with them their own software process tools and the need for these applications to inter-operate has become increasingly important. It is now well established that canonical neutral representations are a better way to enable communication than bespoke translators between pairs of tools. Technologies that implement transparent communication at the syntax level e.g. HTTP, CORBA are now mature. However communication at the semantics level remains a challenge with various initiatives attempting solutions in some areas with various techniques, but none with universal uptake e.g. STEP, CDIF, PIF, Ontilingua.
The Process Specification Language (PSL) project [PSL 1998a] at the National Institute of Standards and Technology (NIST) is creating a neutral, standard language for process specification to serve as a semantic glue to integrate multiple process-related applications throughout the manufacturing life cycle. There is nothing to preclude PSL’s application to software development processes, but currently this has not been attempted. Section 4 will suggest a possible application of PSL to modelling VCPs.

The following section presents use cases defined by partner companies of the VISCOUNT project. The use cases are the basis of defining VCPs and implementing a prototype process support environment.

3. Use cases of a VSC

Use cases defined in this section serve the purpose of identifying VCPs for an existing VSC and formalising the VCPs using PSL.

The VISCOUNT project is defining the interaction between VSC member companies and a VSC support system based on use cases. Use cases are interconnected in UML use case diagrams, each diagram representing a process model functional area. This section builds a scenario from two inter-linked use case diagrams that would be carried out by two distinct companies.

The use cases and use case diagrams have been defined by partner companies in the VISCOUNT project and represent desired software configuration management (SCM) practices [VISCOUNT D.4.1.1]. The use cases were defined for the following functional areas:

- SCM teamwork, use cases related to culture, environment, switching, communication, security and management
- SCM planning, use cases related to current state analysis, activities, roles, applications, control and standards
- SCM core activities, use cases related to configuration identification, configuration status accounting, release manufacturing and management, change control and configuration audit.

While processes falling in the SCM teamwork and SCM planning category are processes that can be identified within any virtual corporation [Hardwick and Bolton 1997], the SCM core activities are specific for VSCs. This is the reason for selecting use cases and use case diagrams from this category to be modelled and provide support for their enactment.

The first use case diagram defined by a VISCOUNT partner describes the interaction between organisational roles and the support system when a release is prepared and delivered to a customer (Release Manufacturing and Management Diagram). Appendix 1 is the UML representation for this use case diagram. The roles involved in this use case diagram could be located within distinct partners of a VSC and could be enacted by one or more persons.

The element that makes this use case diagram important for testing the applicability of our modelling approach is the inclusion of configuration audits, the process for which is defined in the second use case presented in Appendix 2 (Request QA Approval Use Case Diagram). Configuration audits are defined based on practices different from the first use case diagram. This has as result a different set of pre- and post-conditions for the use case diagram, and the need for the VSC members to negotiate how best to interface the processes.

The two use cases diagrams considered in this section represent a process fragment with typical issues for a VSC: roles being distributed, interaction between VCPs have to be negotiated, VCPs may be reused by partners other than the partner who defined the VCP. Modelling the resulting VCPs is discussed in the next section.

4. Modelling VSC processes in PSL/RDF

This section presents our formalism based on PSL and RDF. This formalism is used for modelling VCPs and to integrate the VCPs into a VSC process model.

4.1 PSL

The foundation of PSL is an extendable ontology, which provides rigorous and unambiguous definitions of the concepts necessary for specifying manufacturing processes to enable the exchange of process information. The PSL ontology is represented using the Knowledge Interchange Format (KIF) specification. KIF provides the level of formal rigour necessary to unambiguously define concepts in the ontology, a necessary characteristic to exchange manufacturing process information using the PSL ontology. Although defined in KIF, PSL is decoupled from any syntactical or structural representation, and so equivalent representations for process models can be defined e.g. for XML/RDF discussed below.

The PSL ontology is based upon a small set of primitive concepts: activity, object, time point, and relationship which can be extended.

Currently NIST's pilot studies apply PSL to the translation of complete manufacturing process models from one representation to another [Polyack & Aitken 1998]. We use PSL in a slightly different manner to define VCPs in a form which is independent of any enactment tool. It will also be used to define VSC process models by specifying a network of VCPs and the communication
between them. We consider that this work is complementary to the PSL.

4.2 PSL/RDF

Resource Description Framework (RDF) is a standard from W3C [W3C 1999a]. Its main aim is to provide interoperability between tools which exchange information over the Web. It does this by defining the general structure by which meta-data can be described, and by providing a mechanism (RDFS [W3C 1999b]) for meta-data schema definition for particular domains.

We perceive four advantages in using RDF to define PSL:-
1. RDF’s support for structure in defining meta-data vocabularies whilst retaining understandability
2. RDF uses XML syntax and therefore the existing tool base for processing XML and RDF, including Java based parsers, can be leveraged
3. RDF and XML are text based and easy to transmit and receive in particular for transmitting data and meta-data content between executing VCPs
4. RDF’s ability to combine vocabularies in our case in particular SCM meta-data

An initial experiment of representing PSL as RDF/XML has been carried out by NIST [PSL 1999b]. Although this shows the basic ideas it is not based on any defined schema for PSL and is ad hoc. We have defined, which we consider to be more rigorous, a subset of PSL as an RDFS schema. This allows PSL semantics to be properly captured in RDF.

The RDFS schema is based on the PSL Informal Documentation [PSL 1999a]. The documentation defines the semantics of PSL Kinds (similar to the idea of a class) and Relations. In essence we map the Kinds to RDFS Classes and Relations to RDF Properties. We do not include any reasoning about the semantics of terms and the relationship between them, just encapsulate the same information. Part of the schema is shown in Figure 1 (previous page).

SCM meta-data can also be modelled in RDF. A RDF schema allows for example Unapproved and Approved states of a CI to be described and the changes modelled as the process model is enacted (see Figure 2).

We perceive four advantages in using RDF to define PSL:-
1. RDF’s support for structure in defining meta-data vocabularies whilst retaining understandability
2. RDF uses XML syntax and therefore the existing tool base for processing XML and RDF, including Java based parsers, can be leveraged
3. RDF and XML are text based and easy to transmit and receive in particular for transmitting data and meta-data content between executing VCPs
4. RDF’s ability to combine vocabularies in our case in particular SCM meta-data

An initial experiment of representing PSL as RDF/XML has been carried out by NIST [PSL 1999b]. Although this shows the basic ideas it is not based on any defined schema for PSL and is ad hoc. We have defined, which we consider to be more rigorous, a subset of PSL as an RDFS schema. This allows PSL semantics to be properly captured in RDF.

The RDFS schema is based on the PSL Informal Documentation [PSL 1999a]. The documentation defines the semantics of PSL Kinds (similar to the idea of a class) and Relations. In essence we map the Kinds to RDFS Classes and Relations to RDF Properties. We do not include any reasoning about the semantics of terms and the relationship between them, just encapsulate the same information. Part of the schema is shown in Figure 1 (previous page).

SCM meta-data can also be modelled in RDF. A RDF schema allows for example Unapproved and Approved states of a CI to be described and the changes modelled as the process model is enacted (see Figure 2).

We perceive four advantages in using RDF to define PSL:-
1. RDF’s support for structure in defining meta-data vocabularies whilst retaining understandability
2. RDF uses XML syntax and therefore the existing tool base for processing XML and RDF, including Java based parsers, can be leveraged
3. RDF and XML are text based and easy to transmit and receive in particular for transmitting data and meta-data content between executing VCPs
4. RDF’s ability to combine vocabularies in our case in particular SCM meta-data

An initial experiment of representing PSL as RDF/XML has been carried out by NIST [PSL 1999b]. Although this shows the basic ideas it is not based on any defined schema for PSL and is ad hoc. We have defined, which we consider to be more rigorous, a subset of PSL as an RDFS schema. This allows PSL semantics to be properly captured in RDF.

The RDFS schema is based on the PSL Informal Documentation [PSL 1999a]. The documentation defines the semantics of PSL Kinds (similar to the idea of a class) and Relations. In essence we map the Kinds to RDFS Classes and Relations to RDF Properties. We do not include any reasoning about the semantics of terms and the relationship between them, just encapsulate the same information. Part of the schema is shown in Figure 1 (previous page).

SCM meta-data can also be modelled in RDF. A RDF schema allows for example Unapproved and Approved states of a CI to be described and the changes modelled as the process model is enacted (see Figure 2).
4.3 Defining a VSC with PSL/RDF

As an example of defining a VSC, we take a subset of the use case requirements defined by the SCM process modelling requirements phase of the VISCOUNT project. This is a bottom-up VSC as the requirements have been defined from scratch. As an example we use two use case diagrams and map each to a VCP: the Request QA Approval Use Case Diagram which is initiated in the Release Manufacturing and Management Diagram (see Appendix 1 and 2) although the emphasis within our explanation below will be on the former. (This is an interesting example as it embeds one VCP within another, but there is nothing to preclude sequentially connecting VCPs with the appropriate PSL.)

There are five parts to the PSL declaration (full code for the examples can be found at [Mair 1999]):

1. **PSL VCP Declaration**: specifies layout of the process by defining a network of activities. From the Request QA Approval Use Case Diagram each use case becomes a PSL Activity as illustrated in Figure 2 (previous page). The control flow sequence is embedded in the layout of the representation (XorSplit and xorSplitSubactivity are non-elegant but consistent with PSL). Activity etc. are defined as belonging to a XML namespace psl, in the PSL schema (not shown in diagram). In the xorSplitSubactivity relation only one subactivity will be executed. Which one is defined by different state transitions specified in the ActivityOccurence (Figure 3).

2. **VSC Process model**: A VSC will normally be a network of VCPs connected together. This is defined in exactly the same formalism as the VCPs above e.g. plugging VCPs together sequentially is done by simply arranging VCP activities sequentially (see below for discussion on pre- and post-conditions). Note that in our examples in this paper we do not show a VSC Process model; rather we have one VCP definition which initiates another VCP; this illustrates the same points and allows us to illustrate synchronous subcontractor-like behaviour.

3. **VSC Instance data**: we need also to model information that is in the use case tables e.g. the actors and roles (locations, e-mail accounts etc.), information for an instantiated VCP and VSC process model (including CI and actor instances) and mapping information.

4. **SCM Configuration Items**: a schema is defined to indicate the state of the CI as it progresses through the process. This is defined in RDF with the states: Approved, NotApproved and BeingApproved. SCM CIs form the bulk of the data that is transferred both within and between VCP instances. It is therefore important that they are defined in a form which can be transferred between heterogeneous tools. Part of the CI schema is shown in Figure 4 (next page) (Fluent is defined in the RDFS PSL schema – a Fluent is PSL’s notion of state). Two views are thus incorporated: the process model showing activity sequences, and state transitions of the CI item. The extendibility of RDF allows extra vocabularies such as these to be easily combined.

![Partial CI Schema](image)

5. **Instance data**: as illustrated in Figure 3, this binds the VCP to the VSC Instance data and CI items. ActivityOccurrences are used in PSL to denote instances of activities i.e. each ActivityOccurrence is in turn bound to a particular actor as defined in the VSC instance data. We have added the ability to reason about pre- and post-conditions (currently on single CIs) by adding the notion of requires and provides. This enables the facility of interconnecting opaque VCPs by negotiating their interfaces. In addition we can model state (Fluent) changes e.g. using stateAfter and determine alternative process execution paths, denoted by xorSplitSubactivity subactivities, based on state values. An abridged version of this instance data is shown in Figure 3 where qa is a XML namespace for the PSL VCP declaration shown in Figure 2. Lifespan1 is a pseudo PM actor for the SCM tool we currently use (Lifespan [BAe 1999]) (pseudo as the SCM tool and the process modelling functionality are here together treated as one functional unit; but are in fact in the experiment described below two software entities).

4.4 Mapping PSL/RDF to Enactable Process Models

A VSC process model consists of a network of VCPs (or equivalently as in our example), definitions of the VCPs themselves and various aspects of VSC instance data. These descriptions are not in a form which is directly enactable. As illustrated in Figure 5, two transformations
are necessary to this representation to obtain an enactable model.

The first transformation maps the VCPs to a set of VCP instances. The VSC process model is also instantiated using the same VSC instance data to describe communication between the VCP instances (although this paper does not discuss this directly). Information about the communication mechanism is also required i.e. send or share model. The first transformation therefore results in a set of partitioned VCP instances bound to actor instances.

The actor field in this case represents the actor which enacts the VCP instance; in this case it is Lifespan1. The specification determines next node in the sequence, message specifications and conditional control. The conditional control can be implemented by sending e-mail questions to the other actors. This form is suitable as a specification for designer use and is used as a template for workflow design CAD tools. We anticipate that the existing tool support for XML and RDF can be applied to creating automated translation tools for this transformation. Currently it is undertaken by hand by applying a set of rules. These rules are yet to be documented.

The second transformation involves taking the first output generated and mapping it to specific process model or workflow representations. This is suitable for tools requiring language-type workflow descriptions.

Figure 5: Mapping definitions for enactment

In terms of our results presented below, which use a send model based on an Internet email based workflow tool, the output generated is in terms of a generic node and arc based representation. As well as specifying the control flow it also specifies the information flows, described as messages between the actor instances in terms of an e-mail interface e.g. attachments and subject fields. Conditional flows are also modelled. An edited example of a VCP instance for the Request QA Approval VCP and VCP instance data previously discussed is shown in Figure 6.

Figure 6: Partial Result of First Mapping

5. A prototype implementation

As validation of our work we have implemented a partial VSC to support instantiation of the case study supported by the two use case diagrams. The implementation has the following components and topology:

1. We use two instances of GFI's Emaiflow [GFI 1999], each implementing a VCP instance. Each Emailflow instance simulates the execution of the
SCM process internal to a separate VSC company. We define this architectural viewpoint as a **VSC domain**. Note that this means personnel from either company can participate in the process of either VCP instance; but that the enacted VSC process is partitioned. This naturally fits the working practices of temporary VSC alliances. The primary reason for choosing Emailflow was its provision for interfacing with other tools. The definition of workflows follows a node-arc model albeit not based on any formalism. Currently meta-data information is communicated in e-mail messages as name-value pairs (which Emailflow provides a simple mechanism for parsing). We anticipate that this will be restructured as equivalent XML in a second prototype. Workflows can be started by sending e-mails which facilitates VCP instances starting others. An example of the workflow development tool showing the design of the workflow for the Request QA Approval VCP is shown in Figure 7.

Figure 7: Emailflow Implementation of Request QA Approval VCP

2. Each Emailflow instance is "back-ended" into an instance of BAe's Lifespan [BAe 1999]. Lifespan is an industrial strength SCM tool with support for ISO 9001 and other quality standards. Although Lifespan comes with non-distributed process model support, it does not address issues such as process interoperability, and we use Lifespan only as a CI and CI meta-data repository. The meta-data includes roles, modules and packages, a design control mechanism, and reporting mechanism. It is the intention that, within the context of the VISCOUNT project, this work is viewed as a feasibility study for future versions of Lifespan with improved process support. Lifespan supports Microsoft Windows and Motif user interfaces, a C++ API and client UNIX and NT command line interfaces.

3. The interface between Emailflow and Lifespan is bespoke for particular VCPs. This means that we have specific interfaces for system use cases and not a generalised API. Currently our platform uses Emailflow on two NT boxes with Lifespan also on one of the NT boxes and also on HP-UX. Emailflow can maintain run-time data in a number of different repositories - we use Microsoft Access. The interface between Emailflow and Lifespan involves scripts written VB-Script, which allows manipulation of the run-time data. This synchronously calls a batch file on NT to manipulate the Lifespan client command line interface, passing data both ways as necessary.

6. **Results, Conclusions and Further Work**

So far we have successfully demonstrated our approach of defining a specification of communicating Internet-based VCP instances from user requirements. However we have only a partial solution; it is deficient in the following general areas:-

1. **homogeneity**: we require a second type of process modelling tool to validate heterogeneous communication.

2. **lack of an industrial case-study**: a more robust field trial in industry is required.

Our intention is that our next efforts will be aimed at proving a more general solution covering the above points. Specific areas we would like to address in future are:-

1. **multiple CIs**: so far we pass only the contents of single CIs; we wish to extend this to a more general case especially for pre- and post-conditions.

2. **structured meta-data**: replace name-value pairs with XML/RDF.
3. a comparative mapping and experiment to support interaction based on a *share* model.

More generally and in the longer term, we anticipate work investigating:-

1. heterogeneous SCM tool integration: translation of SCM CI and meta-data perhaps incorporating an ontology for SCM.
2. development of a methodology for negotiating the set-up of VSCs. We anticipate that this would focus on top-down agreement of pre- and post-conditions, perhaps with the addition of agent-type interfaces.
3. iterative process improvement whereby partner VCPs can be measured and improved whilst still participating in the VSC.
4. automatic assessment to determine if specified VCPs meet designated international standards e.g. ISO 9001. Potentially also a library of VCP specifications which meet these standards and can be reused.
5. VSC topology models i.e. to assess the impact of subcontractor or peer-to-peer VSC domains on process models.

**Bibliography**

[Baee 1999] Lifespan Technical Overview, BAeSEMA Ltd., +44 181 942 9661


[Mair 1999] – Mair, Q: [http://bankfoot.gcal.ac.uk/psl/](http://bankfoot.gcal.ac.uk/psl/)


[VISCOUNT D.4.1.1] – Second Revision of User Requirements. Esprit project number: 25754


Appendix 1: Release Manufacturing and Management Use Case Diagram
Appendix 2: Request QA Approval Use Case Diagram

Use Case: Request of QA Approval

1. Automatic notification
   - Release build passes inspections?
     - Yes
       - Release build passes formal tests?
         - Yes
           - QA approves the request
         - No
           - Use Case: QA inspect code and tests procedures
             - No
           - Use Case: QA performs formal tests
             - No
           - Use Case: QA Grant Approval
             - No
           - Use Case: QA Deny Approval
             - No
     - No
       - Use Case: Request QA Approval

Note: A_FAU_UC1, A_FAU_UC2, A_FAU_UC3, A_FAU_UC4, A_FAU_SUC1, A_FAU_SUC2, A_FAU_SUC3, A_FAU_SUC4