A Distributed Agent Based System For Supporting Virtual Software Corporations

ZSOLT HAAG, RICHARD FOLEY and JULIAN NEWMAN
Department of Computing, Glasgow Caledonian University, 70 Cowcaddens Road, Glasgow G4 0BA, Scotland, UK

Abstract. Virtual Software Corporations (VSCs) are a novel organisational form that use the competitive advantage provided by access to scarce competencies and economies of scale in software development. The main feature of a VSC is the distributed and temporary nature of the teams involved and the use of communication and information technology to support its activities. Large-scale software developments have always involved significant co-operative work and have been supported by automated process support environments, which help manage and support the associated workflow activities. However, VSCs are characterised by significantly more dynamic processes which cannot be adequately supported by current process modelling representation and enactment systems. Here we present a distribute multi-agent system using a deontic logic based formalism to model the commitments of process actors in VSCs. The actions performed in a VSC are captured using this formalism and support for specific co-ordination issues identified by the case study of a real-life VSC is provided. We demonstrate how this distributed multi-agent system can support the dynamic nature of the executable process model of VSCs.

Key words: Distributed Agents, Dynamic Process Modelling, Deontic Logic, Virtual Software Corporations

1. Introduction

The software development process has significantly evolved since the software process has been coined (Humphrey 1989), however it remains a labour intensive occupation, requiring the involvement of an increasing number of developers with varied fields of expertise. The challenges of managing such complex processes lead to a number of organisational forms, which address some of the problems. Virtual Software Corporations evolved into becoming the most important organisational solution (Davidow and Malone 1992) with the potential of 80% of companies adopting it by 2003 (Phifer 1998).

A Virtual Software Corporation is a temporary network of independent institutions, enterprises or specialist individuals that through the use of information technology dynamically unite to utilise an apparent competitive advantage (Boldyreff et al. 1996). They integrate vertically bringing their core competencies and act in all appearances as a single organisational unit, which adapts itself to requirement changes by switching resources as needed. The co-operation between several independent units and the need for high volume unrestricted information exchange has caused a change in the managerial structure whereby the classical hierarchy has been replaced by a flatter network of commitments. Thus the very nature of a VSC generates highly dynamic processes due to resource reallocation, process redefinition and changes in the managerial structures. These particularities of VSCs have a considerable impact on systems designed to support software development; the majority of such tools are not suited for supporting development processes within VSCs (Christie 1995).

The shortcomings of present tools are mostly due to failure in modelling the dynamic aspects of the development process. Existing modelling approaches require an apriori process model (Bandinelli et al. 1994) which cannot readily adapt to “on the fly” modifications. On the fly modifications cannot be foreseen in their entirety during the process model definition phase and are due to events such as process roll backs after performed actions have been invalidated, changes in requirements, and changes in personnel when firms are joining or leaving the VSC. Thus VSCs are a classic example of environments requiring the support of adaptive workflow systems.

Dynamic processes require a dedicated modelling approach, capable of integrating on the fly changes and exception handling by capturing the different levels of interaction occurring in distributed environments. It has been suggested that deontic logic is able to formalise different levels of interaction; obligatory behaviour (duties) resulting from a formal definition, and discretionary actions which result from individual initiatives of process actors (Meyer and Wieringa 1993). The logic has been successfully applied to modelling processes with a high degree of complexity and dynamism, such as organisational bureaucracies (Lee 1988).

This paper presents a distributed agent prototype system based on a modelling approach using deontic logic to address the identified needs of dynamic processes in VSCs. We build on the results of (Meyer 1988) in reducing deontic logic to action logic and applying it to the work of (Castelfranchi 1995) in formalising commitments. The resulting modelling approach is suited to commitment

management, which we consider to be the most important software development issue requiring support in VSCs (Haag et al. 1997).

The next section will present a case study of a real life VSC from which we draw the requirements for future support tools and conclude that VSCs are a classical example for dynamic process modelling and support. We continue by introducing the distributed agent prototype system and show how it addresses the identified VSC requirements. The paper discusses the results of this work as an approach for implementing adaptive workflow systems.

2. The case study of a typical real life VSC

Research addressing support for VSCs is only now starting to investigate specific issues, mostly concentrating on technical aspects of configuration management (VISCOUNT) or addressing solutions to distributed process support environments (Bandinelli et al. 1996). However, VSCs are more than distributed systems, their heterogeneous nature implies that modelling requirements are different. Previous work indicates that the most important aspect in VSC process modelling is to address action co-ordination and commitment management (Haag et al. 1997). Based on these initial findings a case study has been conducted to identify the specific details to be captured by a modelling approach.

2.1. The information flow in the case study VSC

Recent research has indicated the importance of inter human communication within software development and the need to identify the requirements for supporting communication (Saeki 1995). Empirical research on process-oriented groupware (Prinz & Kolvenbach 1996) and efforts on designing process centred environments (Conradi et al. 1994) offer relevant insight into local aspects. However, there is no sufficient work addressing issues of interactions within large, distributed organisations or between legally independent organisations (Riemp 1998).

The case study was designed to identify human communication requirements within VSC developments. It has concentrated on analysing the interaction between organisational roles and individual process actors and identifying the information flows between them. The analysis consisted of interviewing key organisational roles and accessing corporate documents, which formally described the development process.

The organisational pattern of the case study VSC, used for this purpose, included two distinct firms having a common parent organisation, located in London and Edinburgh. The two firms constituted what in the company documents was called “The Group”, and its main task was the development of the “Core Product”. The Core Product was a system for the international financial investment market. The product being developed by The Group required the involvement of a third firm located in Singapore which had its own managerial structure and it was integrated in The Group by the Product Manager who was acting as a communicational gateway. The role of the team in Singapore was to develop the user interface for the core product being developed by the UK firms.

The Group Manager and the Engineering Project Manager were located in London, while the Product Manager and most of the development teams were located in Edinburgh. The site in Edinburgh was designated as the main site, and owner of the core product. The source code of the core product was physically situated on a machine at this location. The repositories for the core product and user interface were being replicated between UK and Singapore, and between London and Edinburgh a remote access link was set-up, to allow developers to co-operate in developing the product.

The Steering Committee (SC), as an organisational body had the role of controlling the development process. This was achieved by developing and maintaining the Product Development Control Documents (PDCD) which included The Product Management Policy, Terms of Reference for Organisational Roles, Product Management, Group Management Plan and Development Configuration Control Procedures.

The analysis of the PDCD has indicated that on the UK site, activities were formally described, foreseeable exceptions were documented and metrics were collected to monitor progress. The processes of the company in Singapore were well less documented and the project manager, in many instances, was required to redefine processes when exceptions occurred. The differences in process maturity meant that each site used different support tools and therefore a model for co-operation was difficult to define and enact.

Interviews with organisational roles have indicated that the documented processes failed in capturing exceptions occurring from co-operation between sites, as these were highly dynamic and

impossible to foresee in the definition stage. From the interviews it emerged that there were a considerable number of undocumented, informal information flows, which affected the development process. Tracking the informal flows and their consequences would have required a model able to capture highly dynamic and flexible processes. It can be argued that by a stricter co-operation process definition and enforcement such difficulties would not arise. The penalty of such an approach, as presented in (Josephson 1997) is a high increase of the cost of co-ordinating activities, which reduces the advantages of a VSC development.

The resulting information flow of the VSC is presented in Figure 1 and the flows critical to further analysis and identification of requirements are explained in Table I. The key aspect, requiring support, is the identified informal (undocumented) information flow represented in Figure 1 with a broken line. The next subsection details these aspects in tightly coupled and loosely coupled co-operation.

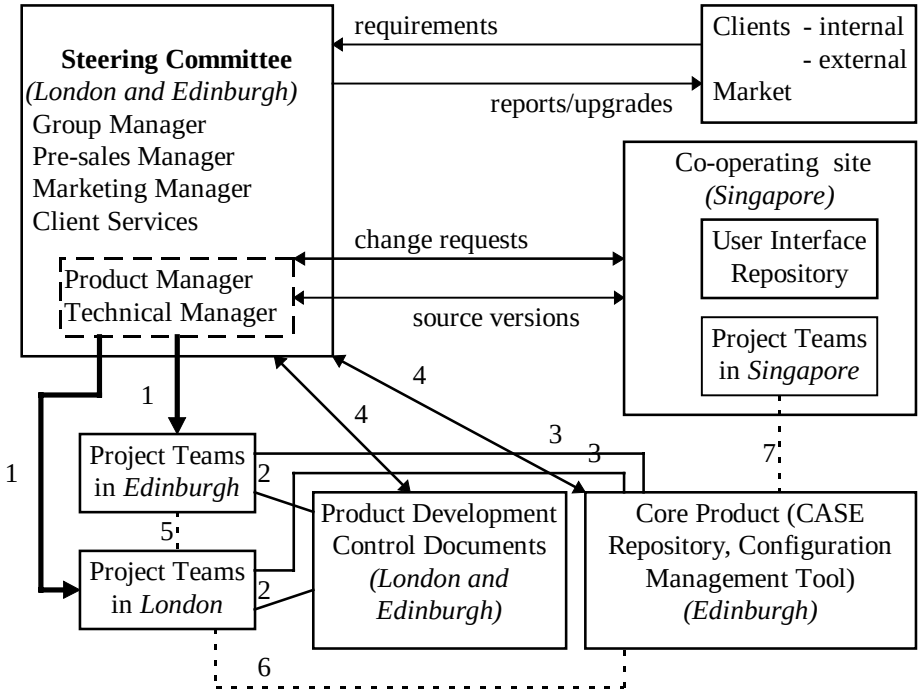


Figure 1. The information flow in the case study VSC

Number	Meaning
1	Information flow containing managerial and technical directives to the project teams. The Product Manager and the Technical Manager are considered to provide the necessary feed-back from Project Teams(PT) to the SC.
2	PT accessing the PDCD to retrieve information regarding formally defined procedures.
3	PT accessing the Core Product Repository (CRuk) in the process of implementing the required functionality of the product. PT are only allowed to change the content and not the structure of the CRuk.
4	SC accessing the PDCD and CRuk. SC's responsibility is to maintain and develop the PDCD and the CRuk while following the procedures defined in the PDCD.
5	Undocumented information flow; informal communication between the UK PT involving actions undocumented in the PDCD
6	Undocumented information flow; change to the CRuk by PT due to technical constraints.
7	Undocumented information flow; change to the CRuk by a co-operating site due to differences in development practices.

Table I. Details for critical information flows in the case study VSC.

2.2. Tightly coupled and loosely coupled co-operation

The analysis indicated that the problems varied depending on the nature of the co-operation between teams: tightly coupled or loosely coupled. For example, due to geographical, temporal and corporate

proximity, organisational borders were blurred between London and Edinburgh. The co-operation in tightly coupled mode was characterised by a large volume of information being exchanged with several "to do" lists generated; which often led to important actions on lists being delayed. It was noted that most of the information exchanged was regarding individual commitments, such as identifying who is doing what, who is assigned to a given organisational role, whose responsibility it is to perform a given action.

In contrast, the loosely coupled co-operation between UK and Singapore was characterised by a low volume of information being exchanged and in many instances implicit knowledge was assumed across cultural and corporate boundaries, which led in cases to misunderstandings of procedures and subsequent process rollbacks. The loosely coupled co-operation also indicated that the actions carried out in the development process are identical across organisational boundaries, however the execution order for actions and the total set of actions are different. Therefore actions are the only constant aspect, and as such represent the building element for a modelling approach suitable for integrating the process of VSC member firms.

Further study into the causes of the undocumented information flows (5 to 7 in Table I) identified that the classical approach to action co-ordination leads to informational overload and managerial bottlenecks (Haag et al. 1997). The Product Manager, was flooded with change reports from the developer teams that he was supposed to make available to collaborating groups. This led to failure in informing all relevant partners about changes and to the generation of the undocumented information flows. The process of new developers joining and leaving the VSC implied that contact lists had to be updated on a regular basis, and this resulted at times in not knowing whose responsibility a given action was.

The findings of the case study support the views emerging from research on Virtual Corporations (Zimmermann 1997), which has identified a major process of change in the organisational structure of corporations, involved in virtual organisations. The classical hierarchy of the managerial structures is being replaced by a network of commitments, often with more than one actor assigned to the same organisational role. The change leads to commitments being blurred across organisational borders (including invisibility of key roles and artefacts), communication bottlenecks, and assumptions about the practices of co-operating organisations.

2.3. Requirements for process modelling

From the formal definition of tasks within the development process it was observed that in many instances the organisational roles authorising or performing certain co-operative activities were not instantiated in the firm documents. The unavailability of the specific data required was one of the possible causes of decisions being made without enough background information. However, two reasons have been identified for the generic nature of the formal definitions. Firstly, the development process is dynamic, commitments of organisational roles are changing, especially since alliances within VSCs are temporary. Secondly, in some instances more than one organisational role (or even an organisational body consisting of several roles) would have the authority to approve a request or to perform an action.

The network of commitments within VSCs requires a freer flow of information to which traditional managerial hierarchies and support tools find it difficult to adapt. Using a hierarchical approach within VSCs leads to information overload and managerial bodies becoming a bottle-neck in the process. This is exemplified in Figure. 2. which presents the obligation of a process actor to report the performance of an action to the manager in two scenarios: single firm and VSC development.

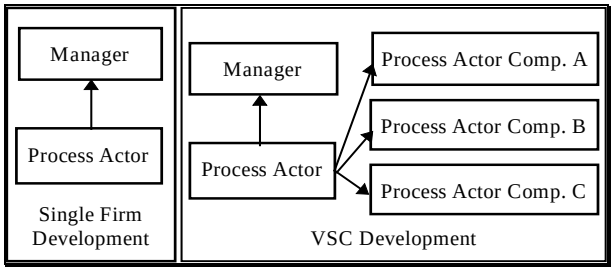


Figure 2. Example of reporting the performance of an action

Actions within the development process could be discretionary or obligatory, however reporting the performance of an action is an obligatory consequence. For a single site development this is achieved by informing the line manager. This generally works since reporting channels are well established and clear procedures are in place. Within VSCs, however, the network of commitments and the interactions of the process actors mean that often one action at one site requires as a consequence several related actions to take place at other sites. Therefore using the hierarchical reporting structure leads to problems, examples of which have been identified in the case study.

The case study indicates that there is a need to support human actors in VSC software development processes. A support tool addressing specific issues in VSCs should provide two functions: to capture formally documented and informal actions and; to provide a mechanism for managing commitments. These functions have to be achieved in a highly dynamic and distributed environment and therefore require a dynamic modelling approach able to adapt to "on the fly" changes.

Workflow systems are often viewed as an essential approach to integrate heterogeneous and often distributed information systems (Georgakopoulos et al. 1995). The basis of any workflow system is a modelling approach, which allows the representation and enactment of organisational processes. Since organisational processes are often not static and evolve over time, there is an increasing attention on modelling approaches for adaptive workflows. In (Casati et al. 1996) the requirements for adaptive workflow systems are presented, and these include dynamic changes and supporting informal co-operation. These represent similar challenges to those identified by the VSC case study. Therefore we conclude that VSCs are a classical example for adaptive workflow systems.

3. The distributed multi-agent based prototype system

The case study has identified the requirements for a support tool that can address specific VSC issues. In the following paragraphs we make a case for a distributed multi-agent system using deontic logic modelling which is able to adequately model the identified VSC issues.

3.1. The modelling formalism

The requirements emerging from the case-study and the research on deontic logic leads to the definition of a formalism with two components. While deontic logic is able to capture both levels of formally defined actions and informal actions it does not support commitment management. Therefore the formalism is based on the work of (Meyer 1988) in reducing deontic logic to action logic and builds on the research of (Castelfranchi 1995) in formalising individual and social commitments in organisations.

The operators defined by deontic logic formalise aspects regarding the nature of rules as being obligatory, permitted or forbidden. The case study has identified that actions are the primary elements to be used by the modelling paradigm. In (Meyer 1988) deontic logic is reduced to action logic, by this the operators are applied to actions rather than abstract rules. The reduction defines V as the violation atom, meaning a liability to some sanction or punishment as the result of an action. With the V atom the operators of deontic logic are summarised in Figure 3.

[1]: $F\alpha \equiv [\alpha] V$: action α is forbidden if the performance of α yields a state where V holds.
[2]: $P\alpha \equiv \neg F\alpha$ ($\equiv <\alpha>\neg V$) : action α is permitted if action α is not forbidden (if there is some way to perform α that leads to a state where V does not hold).
[3]: $O\alpha \equiv F(-\alpha)$ ($\equiv [-\alpha] V$) : action α is obligatory if not-doing α is forbidden.

Figure 3. Deontic operators and their reduction to action logic

The notations in [1], [2] and [3] are: α represents a generic action, $-\alpha$ is the non-performance of α, $[\alpha]$ is the execution of α and $<\alpha>$ a possible execution of α. Classical deontic logic presents a number of paradoxes related to contrary-to-duty imperatives (Chisholm 1963, Forester 1984) which are removed in the reduction to action logic. This makes it possible to use the action logic version for consistency checking based on the deontic axioms and theorems of the standard system KD (Wieringa et al 1991), the axioms of which are presented in Figure 4.

KD0: All (or enough) tautologies of Propositional Calculus
KD1: $O(\alpha \Rightarrow \beta) \Rightarrow (O\alpha \Rightarrow O\beta)$
KD2: $O\alpha \Rightarrow P\alpha$
KD3: $P\alpha \equiv \neg O(-\alpha)$
KD4: $F\alpha \equiv \neg P\alpha$
KD5: Modus Ponens

Figure 4. Axioms of the standard system used in deontic consistency checking.

KD1 is the K-axiom formalising that the obligatory implication of two actions implies that performing one action makes obligatory the execution of the implied action. KD2 is the D axiom and formalises "obligatory implies permitted". Similarly in KD3, if an action is permitted then this is equivalent to not doing the action being not obligatory. A forbidden action, in KD4, is equivalent to the action being not permitted. Two theorems of the KD system, additional to the axioms, are important for consistency maintenance. These theorems are presented in Figure 5.

T1: $O\alpha \equiv \neg P(-\alpha)$ it is impossible of being permitted not to perform an obligatory action
T2: $\neg(O\alpha \wedge O(-\alpha))$ one cannot be obliged to do conflicting actions

Figure 5. Theorems used in consistency checking

Human actors carry out the actions of a deontic rule. The actors are not free of context and therefore an abstraction of commitments is required. In (Castelfranchi 1995) an integration of an action and its context is provided. The abstraction considers that an organisational role is committed to perform an action on a target object or transfer authority for an action; therefore triplet and quadruplet structures are used to represent commitments. The triplet contains: the committed actor; the action the actor is committed to perform (an elementary process such as inform, change structure, change content); and the target of the action (an organisational role, artefact or commitment). The quadruplet extends the previous structure with an additional element indicating a commitment, a construct allowing recursive definitions. These abstractions are summarised in Figure. 6.

(actor, action, target)
(actor, action, target{, comm})

Figure 6. Representation of commitments

The two parts of the formalism (deontic operators defined in Figure 3 and abstractions for commitments in Figure 6) provide the means for capturing actions within the software development process. For example, in company documents it is specified that the Technical manager (TMuk) is permitted (P) to modify the content (m_c) of the UK core repository (CRuk). Similarly, TMuk is required (O) to inform (i) the Project Manager (PMuk) about modifications to the CRuk. The deontic operators used in this example are P and O ($P\alpha$ and $O\alpha$ in the reduction to action logic), which are applied to actions formalised using commitments. The first process rule contains the commitment (TMuk, m_c, CRuk) and by substituting α the first process rule is obtained: P(TMuk, m_c, CRuk).

The statements formalised by the rules in Figure 7 represent a part of a tightly coupled problem identified in the case study. The tightly coupled co-operation between Edinburgh and London was built around a remote access link between London and Edinburgh, which didn't meet the requirements and initial expectations, therefore a request was formulated by the Project Team (PT) in London to copy the repository from Edinburgh. This step was considered to increase the efficiency of the London team by performing coding on a local version. It was proposed to upload regular updates from London to the main repository in Edinburgh in order to maintain the consistency of the project repository. The request to change the structure of the repository had been sent to the TMuk. The TMuk instructed by the PMuk who was the owner of the repository, performed an impact analysis as prescribed in the control documents, and then the PMuk was informed of the result. Based on the impact analysis PMuk has approved the request. The information unavailable to the managerial roles at the moment of taking the decision was the impossibility of the repository management software in Edinburgh to automatically include versions from distinct repositories for the same product. The PT in Edinburgh knew this fact, however they were not consulted in the impact analysis process.

P(TMuk, m_c, CRuk)
O(TMuk, i, PMuk, (TMuk, m_c, CRuk))

Figure 7. Formalised process rules

3.2. The architecture of the distributed multi-agent system

The prototype has been designed around a layered multi-agent architecture. Each layer contains agents with different functionality based on the level of abstraction they represent. There is a direct mapping between the layers of the prototype and the sets of rules identified in defining the formalism. Figure. 8. details the different layers and the interaction between them. Individual agents are using a restricted set of KQML (DARPA 1993) performatives for exchanging knowledge.

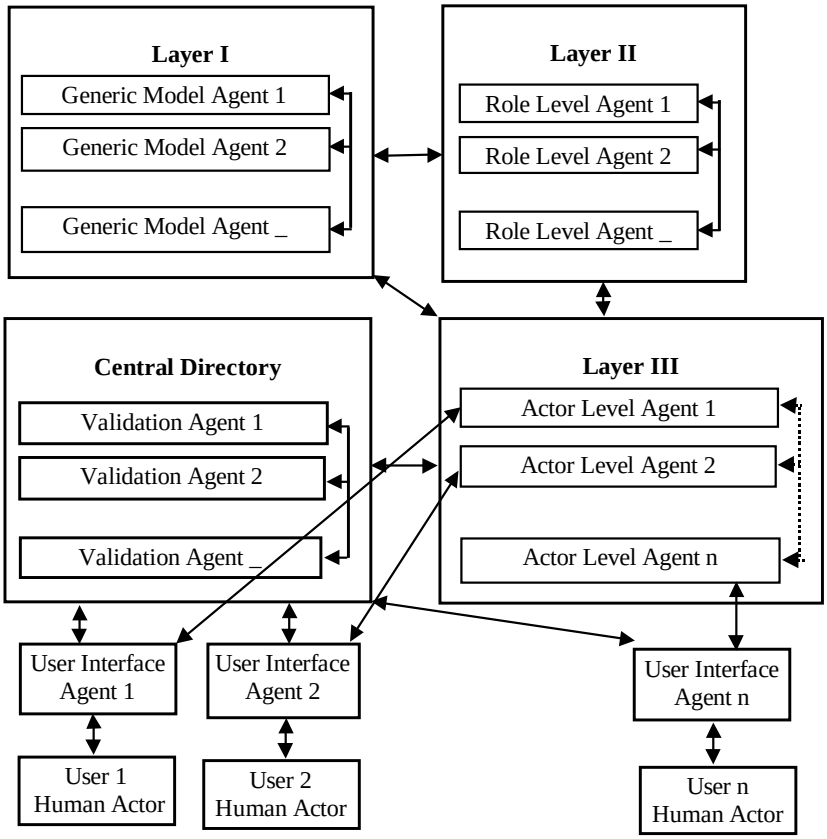


Figure 8. Architecture of prototype support tool.

Layer 1, formed by Generic Model Agents (GMA), formalises generic rules of the process. Here actors are not instantiated and reference is made to actor categories, not individual actors. The agents in this layer are an abstraction for artefacts. Each agent has a knowledge base containing formalised rules and a deontic consistency checker. When a new partner joins the VSC, the GMAs exchange the content of their knowledge base, the rules are parsed and human actors are informed about contradictory rules (rules for which the violation atom holds). The resolution of such inconsistencies is left to the human actors who will have to define additional generic rules or modify existing ones.

Layer 2, formed by Role Level Agents (RLA), captures the commitments of organisational roles providing a model of the development process within groups. The commitments at this level identify the committed roles and artefacts. The commitments of this layer are instantiated rules from the first layer and additional rules contained in artefacts local for the group. The organisational roles can be assigned to one human actor or a group of human actors.

The commitments captured by Layer 1 and 2 come from formal company documents or formal meetings. Layer 3 (Actor Level Agents - ALAs), in contrast, captures the commitments of individual human actors as they might later emerge from daily interactions. This, together with the ability of the

formalism to capture informal actions, is the key to enabling subsequent “on the fly” changes to the process model during enactment.

User Interface Agents (UIAs) provide the interface between the co-ordination mechanism and human actors. Human actors initiate a work session by starting a UIA on their local machine. The UIA retrieves current actions and their contexts from the ALA and provides a context for actions to be carried out. The Deontic Consistency Maintenance Unit, which is integral part of all other agents, has been replaced by a graphical user interface. This interface provides the user with a list of current commitments stored by the corresponding Actor Agent. At present the human actor can perform only a limited number of operation with the displayed rules (confirm performance of a commitment, deny execution, check the feasibility of performing an action). The User Interface agent sends the operations performed by the user to the corresponding Actor Agent where the actions are processed and the results returned to the user. The approach of a lightweight User Interface agent has been considered useful in the prototype stage. Since user needs are not yet clearly defined and further development of these agents will be necessary, only the interaction with the actor agents has been specified allowing for future functionality to be added.

The implementation of the prototype uses generic functional blocks, with well-defined interfaces. This implementation approach allows changes to be made to blocks without affecting the overall operability of a given agent and provides support for incremental evolution of the architecture. For instance, if there were a need to change the transport mechanism, this would require only changes to the network interface modules. Since the blocks of an agent are highly generic (customisation is achieved through configuration files), this means that once a block has been modified, all agents in the system can use the new version.

The message exchange between agents is structured in discussions, similar to the approach taken in (Finkelstein and Fuks, 1989). Each message has a unique discussion code, which is generated by the agent initiating a discussion and is used in any subsequent replies to it. Once a discussion has reached a conclusion (a request has been fulfilled or denied) the discussion is marked finished and no further processing will be done on it. This approach enables a categorisation of messages and acts as a log for performed actions and their cause. In the eventuality of a process rollback, each discussion can be rolled back to a specified point in time. This is possible since messages are time-stamped and assertions can be removed from the knowledge base.

4. Evaluating the prototype distributed multi-agent system

The previous section presented the underlying formalism and the architecture of the distributed multi-agent system. This section discusses the suitability of the prototype distributed multi-agent system by applying it to sample software process fragments. The first sample is derived from the initial case study; the second sample is derived from an independent study of a different VSC.

4.1. Evaluation against the case study example

This section presents the evaluation of the prototype, and implicitly of the underlying formalism, against the initial case study. Firstly, the process example is defined, followed by a discussion on the enactment of the example.

The enactment of the case study process models

Figure 9.details the structure of the prototype support system for the case study process example. The aim of this representation, highlighting the agents creating the layers of the multi-agent system, is to facilitate the process of including rules in the knowledge base of the individual agents. This process begins by identifying the individual agents that will include a given rule.

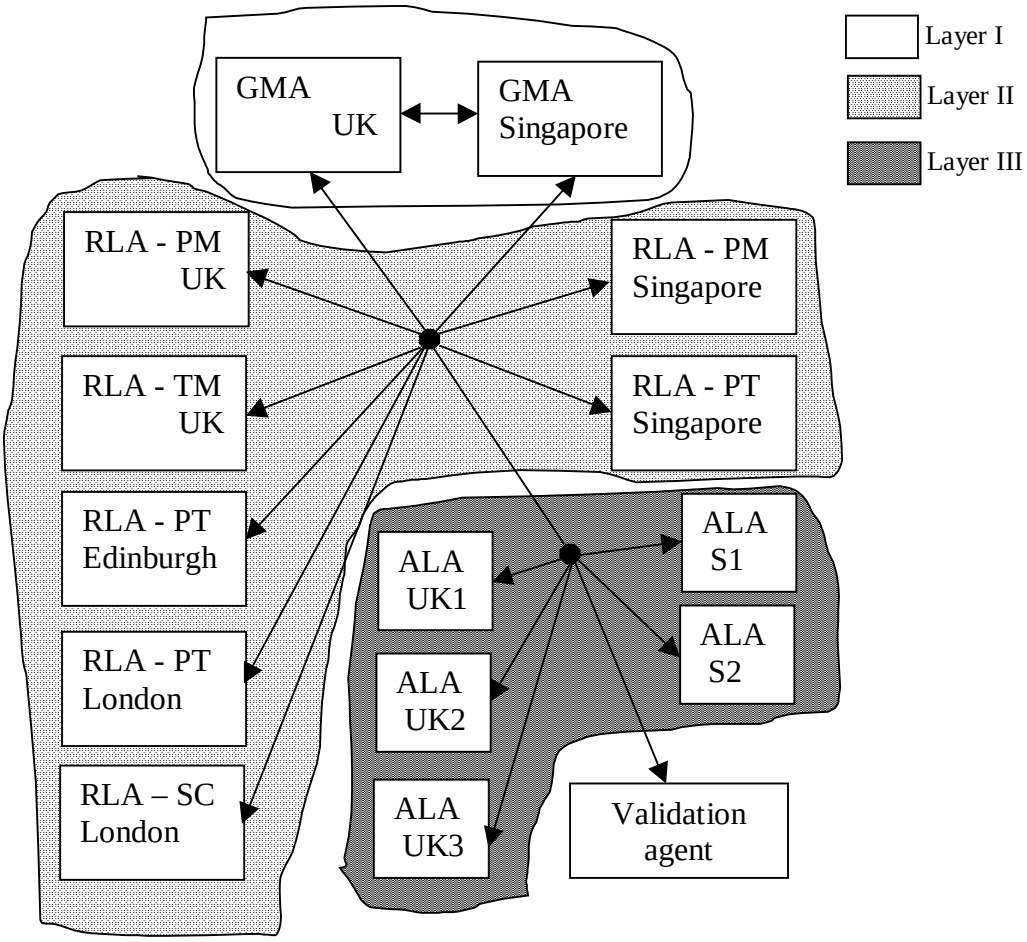


Figure 9. The multi-agent system for the case study

Through the enactment process it was considered that the ALAs will be less in number than the RLAs. This implies that one human actor will be assigned more than one organisational role. At the same time, PM-UK and TM-UK constitute the SC-UK, and therefore, one role has more than one actor associated with it. The relationship between actors and roles is detailed in Table II. This setting caters for nearly all-possible associations between actors and roles, which is important for evaluating the visibility of roles, the extent to which support is provided for identifying the actor associated to a role.

Role	**Performed by (actor)**
PM – UK (PMuk)	ALA-UK1
TM – UK (TMuk)	ALA-UK2
SC – UK (SCuk)	ALA-UK1, ALA-UK2
PTL	ALA-UK3
PTE	ALA-UK3
PMs	ALA-S1
PTS	ALA-S2

Table II. Roles and associated actors

Comments on the execution of the case study process model

The execution of the enacted process demonstrated how the prototype and the defined formalism are able to identify and provide support in solving VSC specific issues, identified in section 2.3. These issues are present in the enacted process example and are captured by rules of the defined formalism.

One of the issues relates to identifying discrepancies between organisational practices, which is exemplified in the case study by the co-operation of teams in UK and Singapore. The rule capturing the discrepancy is presented in Figure 10.

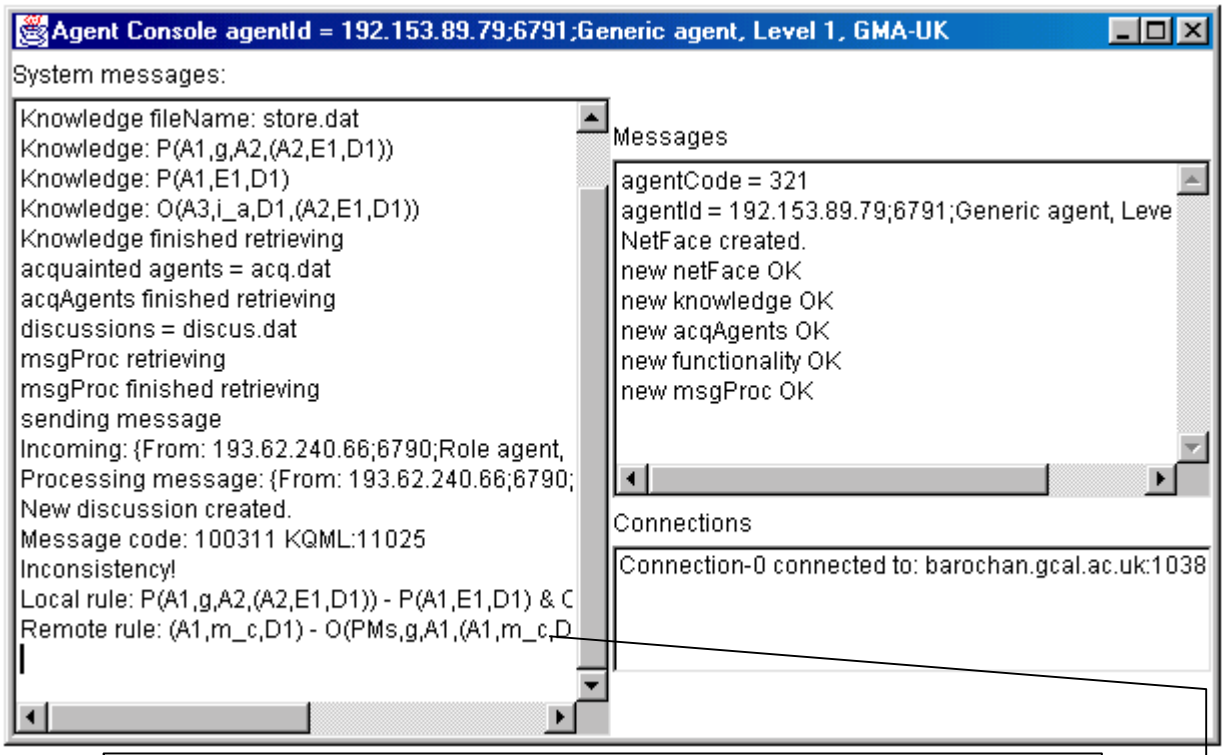
$$(PMuk, g, (PTS, m\_c, CRuk)) \wedge P(PMuk, m\_c, CRuk) \wedge$$
$$O(A1, i\_a, (PTS, m\_c, CRuk)) \wedge$$
$$O(A1, i, *Auk \wedge (Ai, *E, CRuk), (PTS, m\_c, CRuk)) \leftrightarrow (PMs, g, (PTS, m\_c, CRuk))$$

Figure 10. Rule capturing the discrepancy between UK and Singapore

The rule is a double identity, represented by the ↔ sign. For the actions included in the rule to be executed without raising the violation atom, each commitment has to be fulfilled. The left side contains the commitments on the UK side. These are: for the PMuk to grant (g) permission, it must be permitted (P) to modify the CRuk; and that some actor (A1) is obliged (O) to perform an impact analysis (i_a); and that actor is obliged to inform all UK actors (*Auk) who can perform any action (*E) on CRuk.

The right side formalises the action at the Singapore site, where the Project Manager (PMs) grants the permission to the project team (PTS) and this can only be valid if the left side commitments hold. In reality, due to corporate cultural differences and communication difficulties, the action was carried out prematurely and it produced a process roll back and significant rework.

The prototype captures the discrepancy through the two GMAs storing the rules for the UK and Singapore site (as presented in Figure 9). The GMAs exchange their knowledge base and in the System messages panel of the GMA – UK Agent console the message presented in Figure 11 is displayed. The message indicates that an "Inconsistency!" was detected and details the rules which caused the inconsistency. The "Local rule" is the one stored by the current agent, while the "Remote rule" is stored by the other agent taking part in the discussion.



Inconsistency!
Local rule: P(A1, g, A2, (A2, E1, D1)) ↔ P(A1, E1, D1) ∧ O(A3, i_a, D1, (A2, E1, D1)) ∧ O(A3, i, *, (A2, E1, D1)).
Remote rule: (A1, m_c, D1) ↔ O(PMs, g, A1, (A1, m_c, D1))

Figure 11. Rules capturing discrepancy between organisational practices.

Another issue identified to be specific for VSCs, and addressed by the prototype, is the better visibility of organisational roles, artefacts and commitments. The execution of example process models has shown that the prototype is providing an increase visibility for these categories. This is observed when, in the case study VSC example, several actors are associated with one role and one actor is performing the duties of several roles. Under these circumstances the prototype is able to locate roles having a specific commitment or artefacts required for a given action.

4.2. Evaluation against an independent VSC example

This section presents the evaluation of the prototype, and implicitly of the underlying formalism, against an independent VSC case study. Firstly, the process example is defined, followed by a discussion on the enactment of the example.

The independent VSC example

The case study process example, presented in the previous section, was used in developing the formalism and the prototype support system. As a result there is a need to evaluate the prototype against an independent VSC example too, to provide a level of generality to its applicability. The independent VSC was studied as part of work undertaken in the Esprit funded VISCOUNT project (VISCOUNT 1997). The information flow and the issues arising from the software development within this VSC is detailed in (Haag et al. 1998).

The enactment of the independent VSC process model

The multi-agent system for this VSC is presented in Figure 12, which details the structure of the prototype system when enacting example process. In this setting, each role has one and only one associated actor, as this possibility was not considered in the case study example. Similarly to the case study, the rules specifying the processes of the VSC are included in the knowledge base of agents.

The execution of the enacted processes consists of initiating a message exchange between the agents of the system. For example, to evaluate the ability of the prototype to identify discrepancies between the practices of co-operating companies, a message exchange will be initiated between the GMAs in the case study example. The results of the process execution and their analysis are presented in the following sections.

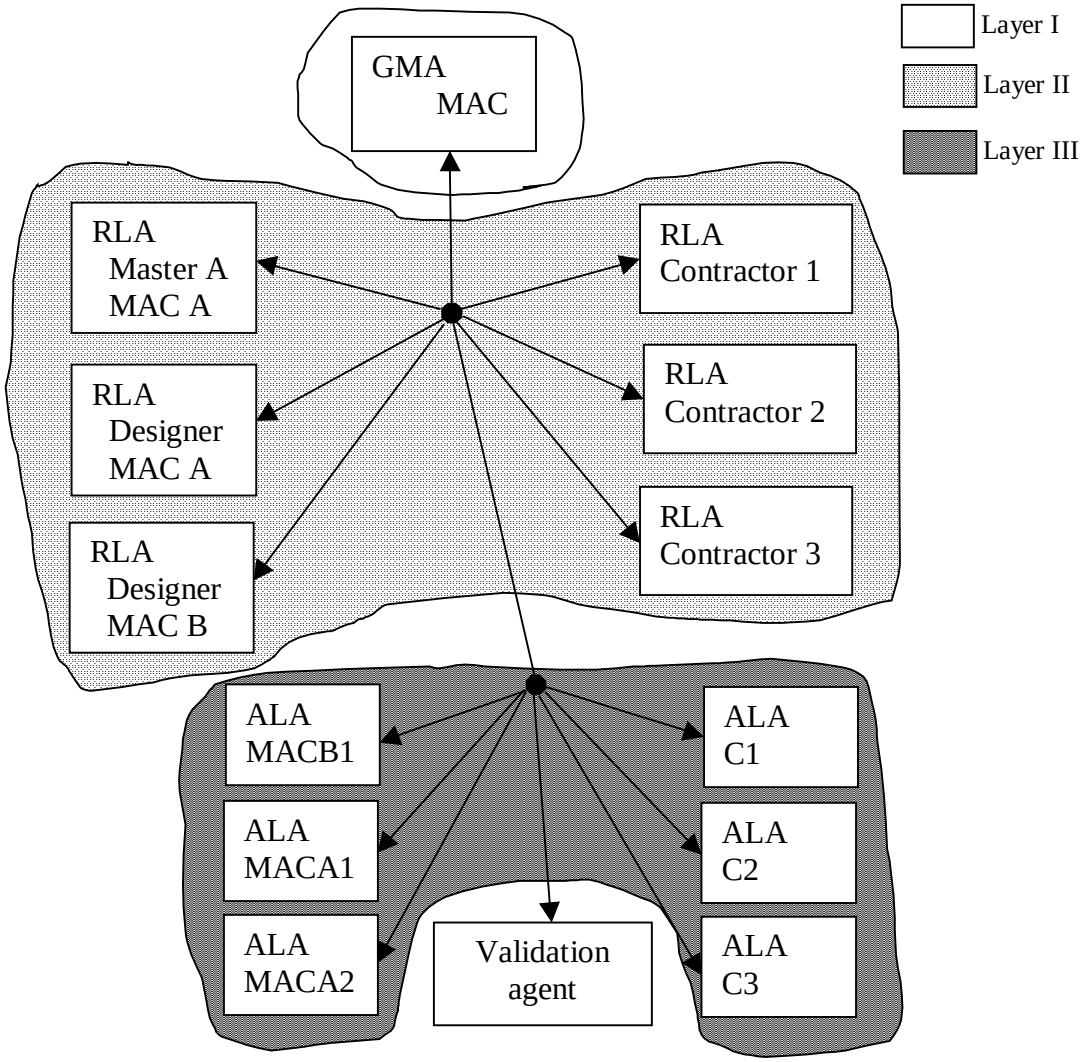


Figure 12. The multi-agent system for the independent VSC

Comments on the execution of the independent VSC process model

In the case of the independent VSC, the prototype provides the context for modifying the contents of a Module 1 for an actor. A particularity of performing this action is the interdiction on parallel locking, which proved to be difficult to achieve in the distributed environment of the VSC. Figure 13 presents the rule generated by the prototype to capture the commitments of a process actor when intending to perform a change in Module 1.

(A, m_c, Module 1) → O(A, r, Master 1, (A, m_c, Module 1)) ∧ O(Master 1, g, A, (A, m_c, Module 1)) ∧ F(Master 1, g, *A-A, (*A-A, m_c, Module 1)) ∧ O (Master A, i, (*A, *E, Module 1), (Master 1, m_c, Module 1))

Figure 13. The rule controlling changes to Module 1

The commitments emerging from this action are three obligations and one interdiction. The first two obligations relate to the process of checking out a document. In the VSC, loosely coupled co-operation was characterised by a failure in enforcing these commitments, as actors were unaware of them. The prohibition caters for the scenario of parallel locking and models the practice of the VSC by which such actions are forbidden. The prohibition was impossible to achieve in the studied process with the existing support, in either loosely coupled (i.e. co-operation between the two MACs) or tightly coupled case (i.e. between MAC A and LCs). This was due to commitments being obscured and thus impossible to identify if somebody has already checked out a copy from the replicated or original site.

The final obligation captures the commitment of the Master actor to inform all involved roles about the change being performed. From all commitments, this proved to be the most difficult to achieve and the failure in enforcing it was the cause of inconsistency between repositories. A typical example was the omission of informing a development team from a change made to Module 1 which lead to parallel development and difficulties in negotiating the resulting versions at a later time.

The previous points presented in the evaluation indicate that the formalism addresses the issues emerging within VSCs. The deontic consistency axioms and theorems, and the violation atom provide a mechanism through which enactment and process monitoring can be achieved in a physical implementation. By this the prototype, and therefore the underlying formalism, provides support for co-operative processes in VSCs.

5. Conclusions

This paper has presented the case study of a VSC, as a novel approach to software development. The main characteristics of VSC developments have been identified as being the dynamic nature of their processes, the replacement of classical hierarchies by a flatter network of commitments and frequent exceptions from defined process models. The informal interactions between human actors, leading to exceptions, had a significant contribution to the overall development, however no existing support for them could be identified. Therefore, there is a significant need for supporting inter human communication within VSCs.

The result of the case study was to indicate that future support environments will have to provide two functions: to capture formally documented and informal actions, and to provide a mechanism for managing commitments. Support systems require the use of a modelling approach in order to represent, reason and support processes. To address this need a formalism has been defined, building on work in formalising commitments and using a variant of deontic logic. The formalism formed the basis of a distributed multi-agent system which was detailed next. Based on the examples of the case study and experiences from an independent VSC, the distributed multi-agent system has been applied to concrete situations. This evaluation concluded that formal and informal actions are handled by the formalism and that support for commitments management is able to make visible crucial organisational roles. These results constitute the basis of using the formalism in modelling and enacting processes within VSCs.

The requirements for adaptive workflow systems identified by (Casati et al. 1996) have a significant overlap with the issues being addressed by the current formalism. Based on these considerations we conclude that the modelling approach could be used as the underlying paradigm of future workflow systems, addressing distributed and heterogeneous developments.

References

Bandinelli, S., Fuggetta, A., Ghezzi, C. and Lavazza, L. (1994): SPADE: An Environment for Software Process Analysis, Design, and Enactment. In (Finkelstein et al. 1994) p223-248

Bandinelli, S., Di Nitto, E., Fuggetta, A. (1996): Supporting co-operation in the SPADE-1 Environment. In *IEEE Transactions On Software Engineering, vol. 22 no. 12 December 1996*

Boldyreff, C., Newman, J., Taramaa, J. (1996): Managing Process Improvement in Virtual Software Corporations. In *Proceedings, IEEE 5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '96)*, June 19-21, 1996, Stanford University Californian, USA

Casati, F., Ceri, S., Pernici, B., Pozzi, G. (1996): Workflow Evolution. In *Proc. of the 15th ER'86 International Conference*, October, Cottbus, Germany, Springer-Verlag Lectures Notes in Computer Science, 1996

Castelfranchi, C. (1995): Commitments: From Individual Intentions to Groups and Organizations. In *Proceedings ICMAS '95 The First International Conference on Multiagent Systems.* June 12-14, 1995 San Francisco, California

Chisholm, R.M. (1963): Contrary-to-duty imperatives and deontic logic. *Analysis*, vol 24 p33-36

Christie, A. M. (1995): *Software Process Automation - The Technology and Its Adoption*. Springer-Verlag 1995

Conradi, R., Fernström, C., and Fuggetta, A. (1994): Concepts for Evolving Software Processes. In *Software Process Modelling and Technology* - edited by: A. Finkelstein, J. Kramer and B. Nuseibeh - Research Studies Press Ltd. 1994 p9-31

DARPA Knowledge Sharing Initiative (1993): Specification of the KQML Agent-Communication Language. Electronically available at: http://www.cs.umbc.edu/kqml/papers/

Davidow, W.H., Malone, M.S. (1992): *The Virtual Corporation: Structuring and Revitalizing the Corporation for the 21st Century*. Harper Business, New York

Finkelstein, A. and Fuks, H. (1989): Multi Party Specification. In *Proceedings of the 5th International Workshop on Software Specification and Design*. IEEE CS Press

Forester, J.W. (1984): *Gentle murder, or the adverbial Samaritan*. Journal of Philosophy, vol 81 p193-197

Georgakopoulos, D., Hornick, M., Sheth, A. (1995): An Overview of workflow management: from process modeling to workflow automation. In A. Elmagarmid, editor, *Distributed and Parallel Databases*, Volume 3. Kluwer Academic Pub., Boston 1995

Haag, Zs., Foley R., Newman, J. (1997): Software Process Improvement in Geographically Distributed Software Engineering: An Initial Evaluation. In *Proceedings of The 23rd Euromicro Conference*, Budapest September 1997, Hungary, IEEE-CS Press

Haag, Zs., Rahikalla, T., Taramaa, J., and Välimäki. A. (1998): Case Study of a VISCOUNT VSC Development. *Technical Report - COS/CSCW/01/1998*, Glasgow Caledonian University

Humphrey, W.S. (1989): *Managing the Software Process.* Adisson-Wesley Publishing Company

Josephson, F. (1997): Distributed Software Design. Presented at the *EuroBest Seminar on Distributed Software Development*, SISU, Stockholm, February 1997

Lee, R.M. (1988): Bureaucracies as Deontic Systems. In *ACM Transactions on Office Information Systems.* vol 6 no 2 p87-108

Meyer, J.-J. Ch. (1988): A Different Approach to Deontic Logic: Deontic Logic Viewed as a Variant of Dynamic Logic. In *Notre Dame Journal of Formal Logic* 29 (1) p109-136

Meyer, J.-J.Ch. and Wieringa, R.J. (eds) (1993): *Deontic Logic in Computer Science: Normative System Specification.* Wiley 1993

Phifer, G (1998): The Future of the Virtual Organization. GartnerGroup, Research Note, Strategic Planning, 11 September 1998, electronically available at www.gartnerweb.com/gg/purchase/0/00/725/*39/doc/ 00072539/00072539.html*

Prinz, W., Kolvenbach, S. (1996): Support for workflows in a minsterial environment. In *Proceedings of CSCW 96 ACM Boston, MA*

Riemp, G. (1998): *Wide Area Workflow Management – Creating Partnership for the 21st Century*. Springer-Verlag 1998

Saeki, M. (1995): Communication, Collaboration and Cooperation in Software Development – How Should We Support Group Work in Software Development. In *Proceedings of the 1995 Asia Pacific Software Engineering Conference*

VISCOUNT (1997): VIrtual Software COrporation UNiversal Testbed (VISCOUNT), Esprit Project No. 25754

Wieringa, R.J., Weigand, H., Meyer, J.-J.Ch., and Dignum, F.P.M. (1991) "The Inheritance of Dynamic and Deontic Integrity Constraints" in *Annals of Mathematics and Artificial Intelligence* 3, pp393-428.

Zimmermann, F.-O. (1997): Structural and Managerial Aspects of Virtual Enterprises. *electronically available at http://www.teco.uni-karlsruhe.de/IT-VISION/vu-e-teco.htm*