A Deontic Formalism for Co-ordinating Software Development in Virtual Software Corporations

Zsolt Haag, Richard Foley, Julian Newman
Glasgow Caledonian University
Department of Computer Studies
70 Cowcaddens Road
Glasgow G4 0BA
United Kingdom
email: {z.haag, r.foley, j.newman}@gcal.ac.uk

Abstract

The concept of the Virtual Software Corporation (VSC) has recently become a practical reality as a result of advances in communication and distributed technologies. However, there are significant difficulties with the management of the software development process within a VSC. The main problem is the significantly increased communicational complexity of the process model for such developments. The more classic managerial hierarchy is generally replaced by a "flatter" network of commitments. Therefore new solution approaches are required to provide the necessary process support. The purpose of this paper is to present a solution approach which models the process based on deontic logic. The approach has been validated against a case study where it was used to model commitments and inter-human communications within the software development process of a VSC. The use of the formalism is exemplified through a prototype system using a layered multi-agent architecture.

1. Introduction

In just over four decades of software engineering a multitude of approaches have been defined to improve the controllability of developing software and to improve the quality of the final product [Christie 1995]. Current challenges for software engineering include economies of scales, specialisation of activities and the emergence of the global network [Boldyreff et al 1996]. A natural response to these challenges is the emergence of a new concept - The Virtual Software Corporation (VSC) which represents a novel form of co-operation between firms. VSCs are alliances of firms with distinct expertise which are co-operating to achieve a common goal. A characteristic of VSCs is the intensive use of IT to provide support for stronger interactions between its members. The VSC is not only a modern approach to economies of scale but also brings together scarce competencies and resources to meet development requirements [Boldyreff et al 1996].

The specific feature of a VSC is the replacement of the classical hierarchical managerial structure with a dynamic network of commitments [Zimmermann 1996]. These changes induce an increase in the environmental complexity of the software development process. The issues identified as crucial in reducing the environmental complexity and providing efficient support in maintaining consistency within VSCs are co-ordination of actions and inter-human communication [Haag et al 1997].

Process Support Environments(PSEs), as the approach to software process automation, have to address the issues of software development in VSCs. Their main goal is to support developers by providing process management, tool integration and capabilities for communication between actors. Actions carried out by the individuals working within a PSE will cause changes to those working contexts, therefore PSEs must be seen as providing support for human beings as they carry out their activities within the overall development process [Snowdown & Warboys 1994]. The underlying process model of current PSEs (which have been based on Petri Nets, logical rules or object abstractions) is an apriori process model [Christie 1995] which cannot readily adapt to "on the fly" modifications. On the fly modifications cannot be foreseen in 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. Therefore VSCs have a more dynamic development process and require a more flexible process model.

Such factors are causing exceptions from the apriori model and have to be dealt with. The solution approach is to reconsider the apriori model and re-enact it [Sommerville 1992], resulting in a modified set of constraints and requirements to be fulfilled by the human actors. As the process unfolds, occurring exceptions will require changes in the process model as the result of a series of feedbacks, leading to an iterative process, as illustrated in Fig. 1. adapted from [Lehman 1996].

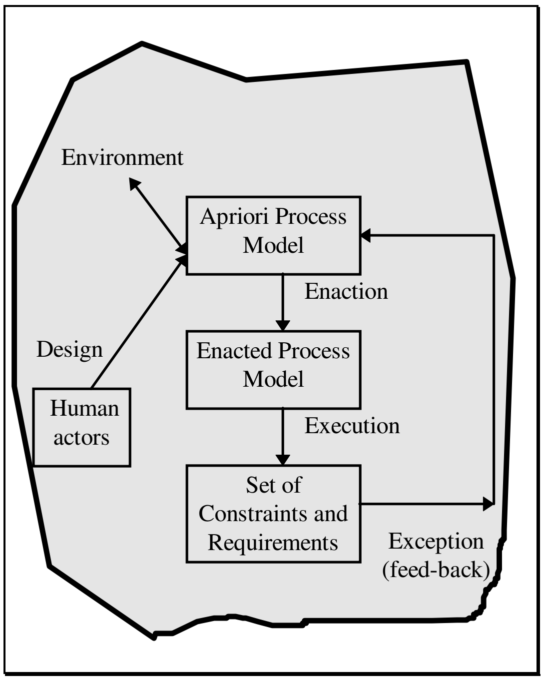

Fig. 1. The iterative nature of the process model definition

The purpose of this paper is three fold. First, it presents the specific issues to be addressed by process support environments for VSCs. These issues are emerging from the case study of an actual (and typical) VSC. Second, it introduces and discusses a formalism for process modelling based on a reduction of deontic logic to action logic which provides support for VSCs. Third, details a practical usage of the formalism through the implementation of a prototype co-ordination mechanism. The prototype is adapted to the flatter network of commitments characterising VSCs [Zimmermann 1996], the set of relations occurring between process actors and the temporary nature of emerging alliances between VSC members. This provides support for co-ordination issues and addresses the iterative nature of the process model.

2. The case study of a VSC

The issues to be addressed in supporting co-ordination of the software development in VSCs have been identified from the information flow of a case study VSC. The analysis of the actual (and typical) VSC included interviewing key organisational roles and accessing corporate documents which formally described the development process.

2.1. The information flow in the case-study VSC

The organisational pattern of the case-study VSC included two distinct firms having a common parent organisation, located in London and Edinburgh and a third partner located in Singapore. The firms in London and Edinburgh formed what was called "The Group", and there main task was the development of the "Core Product" which was a system for the international financial investment market. The role of the team in Singapore was to develop the user interface for the core product being developed by The Group. Communication between sites was co-ordinated by the Product Manager located in Edinburgh.

The information flow of the VSC is summarised in Fig. 2 on the next page. The arrows in Fig. 2. Indicate informational pathways. The meaning of the numbers is summarised in Table 1. Below. The informational pathways presented with a dotted line represent undocumented informational flows for which the structure and/or informational content was not specified in any corporate documents and have been identified through interviews.

Table 1. - Keys for the informational pathways

Nr	Meaning
1	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 must know and follow the procedures defined in the PDCD, therefore they are permitted to read the PDCD.
3	PT are permitted to access the Core Product Repository (Cruk) in order to implement the required functionality of the product but are forbidden to modify the structure.
4	the SC must 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

Nr	Meaning
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.

2.2. Identified problems

Building on the information flow, problems specific to co-ordination within VSCs have been identified. The analysis indicated that the problems varied depending on the nature of the co-operation between teams, tightly or loosely coupled. For example, due to geographical, temporal and corporate proximity, organisational borders were blurred between London and Edinburgh. The volume of information exchanged was high with several "to do" lists generated; which often led to important actions on the list being delayed. In contrast, the co-operation between UK and Singapore was characterised by a low volume of information being exchanged and in many instances implicit knowledge was assumed, which led in cases to misunderstandings of procedures and subsequent process rollbacks.

Further study into the causes of the undocumented information flows (5 to 7 in Table 1) identified that the classical approach to process co-ordination leads to informational overload and managerial bottle necks [Haag et. Al 1997a]. The Product Manager, was flooded with change reports from the developer teams. This led to failure in informing all relevant partners about changes and to the generation of these undocumented information flows.

The findings of the case study support the views emerging from research on Virtual Corporations [Zimmermann 1996] 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 bottle-necks, assumptions about the practices of co-operating organisations.

3. Deontic logic for process modelling in VSCs

Research has indicated that deontic logic, a modal logic concerning norms, can be successfully applied in modelling and maintaining the consistency of processes with a high degree of complexity [Meyer and Wieringa 1993]. Such examples include legal expert systems [McCarty 1989], normative systems [Minsky & Lockman 1985] and organisational bureaucracies [Lee 1988].

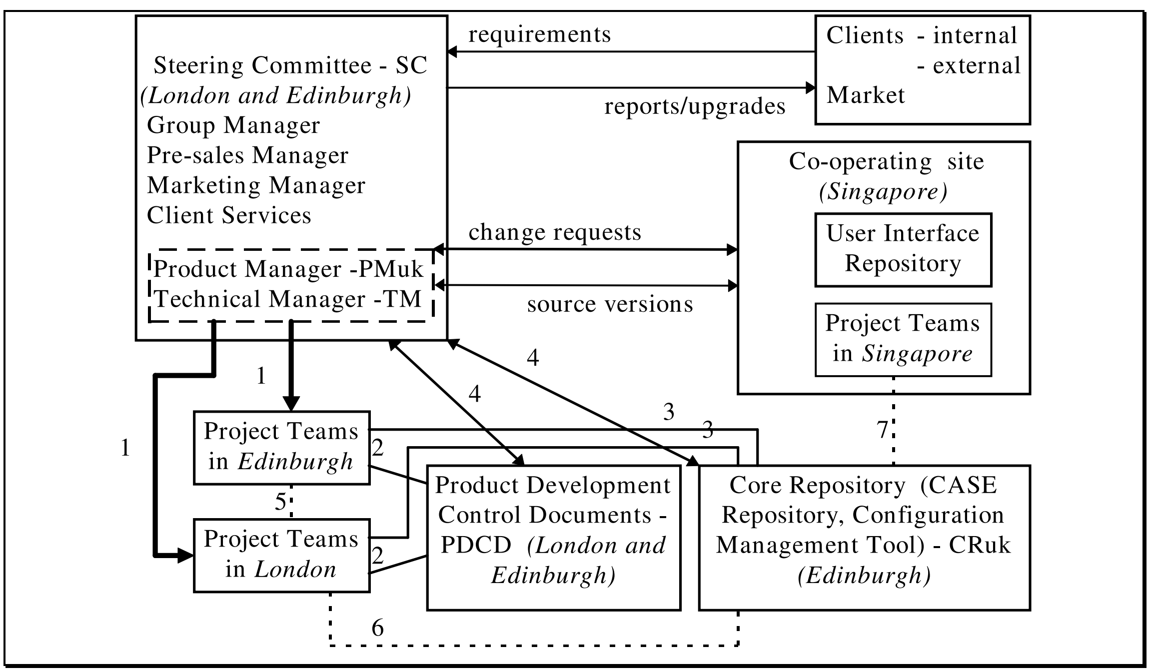

Fig. 2. - Parties involved in the development process and the links between them.

3.1. A case for using deontic logic in modelling VSCs

The network of commitments within VSCs requires a freer flow of information to which traditional managerial hierarchies 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 Fig. 3. 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.

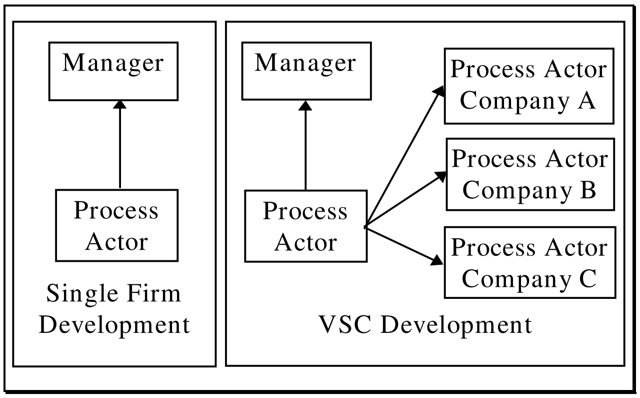

Fig. 3. Reporting the performance of an action

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 section 2.2 of this paper. However it is possible to define, using deontic rules, a formalism which could form a suitable basis for supporting such commitment networks. The feature that makes this logic successful is the ability to formalise both obligatory behaviour (duties) resulting from a formal definition and discretionary actions which result from individual initiatives of process actors.

3.2. Introducing the formalism used for process modelling

The proposed formalism for modelling the information flow of VSCs includes a non standard logic and an abstraction of commitments. The non standard logic is derived from deontic logic which has been successfully applied in consistency maintenance in several fields of computer science. Building on these results, a reduction of deontic logic to action logic proposed by [Meyer 1988] is considered. 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 concepts of deontic logic are summarised in Fig. 4.

The notations in Fig. 4. are: α represents a generic action, $-\alpha$ is the non-performance of α, $[\alpha]$ is the execution of α and $\langle\alpha\rangle$ a possible execution of α. The deontic operators provide a formalism to represent the restrictions, constraints and possible actions which define the development process. To ensure the consistency of the model a set of deontic axioms and theorems of the standard system KD presented in [Wieringa et al 1991] are used.

$F\alpha \equiv [\alpha]\,V$:action α is forbidden if the performance of α yields a state where V holds

$P\alpha \equiv \neg F\alpha$ ($\equiv \langle\alpha\rangle\neg V$) :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)

$O\alpha \equiv F(-\alpha)$ ($\equiv [-\alpha]\,V$) :action α is obligatory if not-doing α is forbidden

Fig. 4. The Deontic operators

The actions, part of the deontic rules, are carried out by human actors within a context. An abstraction of commitments introduced by [Castelfranchi 1995] provides an integration of an action and its context. The abstraction considers that an organisational role is committed to perform an action on a target object or transfer authority for an action; therefore a triplet and quadruplet structures is considered 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. These abstractions are summarised in Fig. 5.

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

Fig. 5. Abstraction for commitments

Considering the abstractions from Fig. 5. and the defined deontic operators it becomes possible to formalise 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 core repository (CRuk). Similarly, TMuk is required (O) to inform the Project Manager (PMuk) about modifications to the CRuk. These statements are formalised by the rules in Fig. 6. which is a snapshot from a specific example of a tightly coupled problem identified in section 2.2.

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

Fig.6. Formalised process rules

Using the presented formalism the content of company documents of the case study VSC has been formalised [Haag et al, 1997a]. The resulting deontic rules have been parsed applying the consistency rules of the standard KD system. The resulting logical equivalencies have demonstrated the ability of the formalism in modelling the process of VSCs. Any deviation from the formally defined process would raise the violation atom identifying any inconsistencies. Commitments which would be an implication of actor's actions constitute the context of the given action. The commitments cover different levels of generality and have to be available in the process of computing the implications without regard to geographical and organisational boundaries.

3.3 The prototype co-ordination mechanism

The prototype is built on a process model defined using the deontic formalism. The multi-agent system of the prototype includes agents representing actors and artefacts of the development process. The different levels of interactions identified in the case study are addressed through a layered implementation as presented in Fig. 7.

Layer 1 formed by Generic Model Agents (GMA) contains generic rules of the process. At this level actors are not instantiated and reference is made to actor categories rather than individual actors. The agents at this stage are an abstraction for artefacts (company documents). Each agent has a database containing formalised rules and a deontic consistency checker. When a new partner joins the VSC, the GMAs exchange the content of their database using KQML performatives (such as ask_all and reply), the rules are parsed and human actors are informed about contradictory rules (rules for which the violation atom holds). The violation atom holds if action permitted in one organisation are forbidden in any other organisation. 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 result 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. Similarly to GMAs, when interaction occurs between different roles, RLAs exchange the content of their database and draw attention to any inconsistencies that could affect the development process.

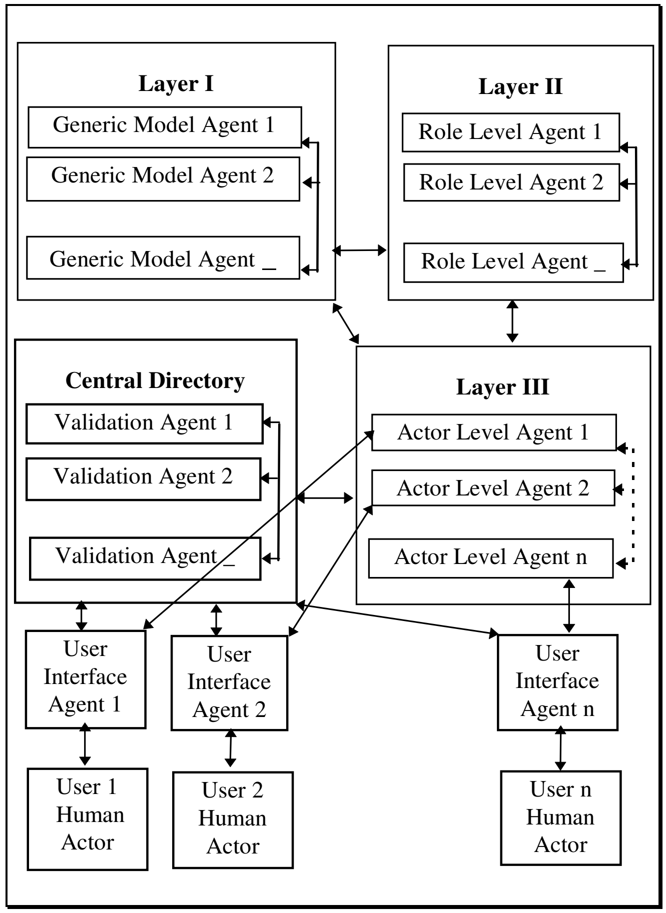

Fig. 7. - The layered multi-agent architecture of the co-ordination mechanism.

The commitments captured by Layer 1 and 2 come from the formal company documents or formal meetings. Layer 3, in contrast, captures the commitments of individual human actors as they emerge from daily interactions and contains references to relevant commitments from previous layers. Actor Level Agents (ALAs) are the abstraction for human actors within the development process.

The interface between the co-ordination mechanism and human actors is provided by User Interface Agents (UIAs). Human actors initiate a work session by starting a UIA on their local machine. The UIA seeks a connection with the closest Validation Agent (VA). Once the human actor has been identified, the VAs which form the Central

Directory will provide the address of the corresponding ALA. The UIA retrieves current actions and their contexts from the ALA. The human actor is provided with a list of action being carried out or to be carried out.

The presented layered architecture allows for changes to be localised. The co-ordination support consists of identifying and highlighting possible reasons for conflict created by different practices and restrictions of the individual development process.

4. Conclusion

The concept of VSCs is a practical reality, firms are already taking advantage of the competitive edge offered by this novel managerial approach. This paper presented a formalism to model the commitments within VSCs as a means to support co-ordination.

From the analysis of a VSC, specific problems requiring support were investigated and identified. The analysis included interviewing key organisational roles and accessing corporate documents which formally describe the development process. Building on the results of the case study a formalism has been introduced based on deontic logic and an abstraction of commitments. The ability of the formalism to represent commitments within the software development was discussed.

The defined formalism constitutes the basis of a prototype co-ordination mechanism. The mechanism has a layered architecture to adapt for the specific problems within VSCs. The trials conducted with the prototype on the case study examples demonstrate that the use of deontic logic and commitment management based approach can support the key areas of inconsistency and co-ordination within the software process of a VSC.

Further work on the formalism and the prototype mechanism consists of testing and validating them against other VSCs different from the case study. This will provide an increased level of generality for the presented approach.

References

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

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

[Christie 1995] - A. M. Christie - Software Process Automation - The Technology and Its Adoption - Springer-Verlag 1995

[Finkelstein et al. 1994] - Software Process Modelling and Technology - edited by: A. Finkelstein, J. Kramer and B. Nuseibeh - Research Studies Press Ltd. 1994

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

[Haag et al 1997a] - Zs. Haag, R. Foley, J. Newman - "Using a Deontic Logic Based Model for maintaining Consistency within the Software Process of Virtual Software Corporations", Research Report COS/CSCW/02/1997 -Glasgow Caledonian University

[Lee 1988] - R.M. Lee - Bureaucracies as Deontic Systems - ACM Transactions on Office Information Systems, vol 6 no 2 April 1988

[Lehman 1996] - M. Lehman - Feedback, Evolution And Software Technology - European Workshop on Software Maintenance, Durham 1996

[McCarty 1989] - L.T. McCarty - A language for legal discourse I: Basic features - Proceedings of The Second International Conference on Artificial Intelligence and Law, June 1989, printed by ACM

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

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

[Minsky & Lockman 1985] - M. Minsky, A. Lockman - Ensuring integrity by adding obligations to privileges. In 8th IEEE International Conference on Software Engineering, p92-102, 1985

[Snowdown & Warboys 1994] - R.A. Snowdown and B.C. Warboys - An Introduction to Process-Centred Environments in [Finkelstein et al 1994] p1-8

[Sommerville 1992] - I. Sommerville - Software Engineering - 4th Edition, Adison-Wesley 1992

[Wieringa et al 1991] -R.J. Wieringa, H. Weigand, J.-J.Ch. Meyer & F.P.M. Dignum - The Inheritance of Dynamic and Deontic Integrity Constraints - Annals of Mathematics and Artificial Intelligence 3, 1991, pp393-428.

[Zimmermann 1996] - F.-O. Zimmermann - Structural and Managerial Aspects of Virtual Enterprises - electronically available at http://www.teco.uni-karlsruhe.de/IT-VISION/vu-e-teco.htm