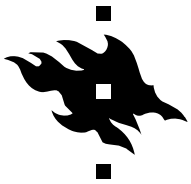
DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS C
REPORT C-2005-16

Virtual organizations

Lea Kutvonen, Sampo Karjalainen, Anna-Kristiina Ritola,

Sini Ruohomaa, Mikko Hämäläinen, Tuomas Nurmela,

Toni Ruokolainen, Teemu Virtanen



UNIVERSITY OF HELSINKI
FINLAND

Contact information

Postal address:
Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki
Finland

Email address: postmaster@cs.Helsinki.FI (Internet)

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 51120

Computing Reviews (1998) classification: H.4, J.1, J.4, K.4.3, K.4.4, K.4.6
Helsinki 2005
Helsingin yliopisto, tietojenkäsittelytieteen laitos

DEPARTMENT OF COMPUTER SCIENCE
SERIES OF PUBLICATIONS C
REPORT C-2005-16

Virtual organizations

Lea Kutvonen, Sampo Karjalainen, Anna-Kristiina Ritola,
Sini Ruohomaa, Mikko Hämäläinen, Tuomas Nurmela,
Toni Ruokolainen, Teemu Virtanen

UNIVERSITY OF HELSINKI
FINLAND

Virtual organizations

Lea Kutvonen, Sampo Karjalainen, Anna-Kristiina Ritola,
Sini Ruohomaa, Mikko Hämäläinen, Tuomas Nurmela,
Toni Ruokolainen, Teemu Virtanen

Department of Computer Science
P.O. Box 26, FIN-00014 University of Helsinki, Finland
firstname.lastname@cs.helsinki.fi

Technical report, Series of Publications C, Report C-2005-16
Helsinki, September 2005, v + 96 pages

Abstract

The Virtual organizations seminar is embedded in the research and educational agenda of the Cinco research group. This report is based on the seminar presentations, and studies various relevant research topics and projects contributing to the field of virtual organization research.

Computing Reviews (1998) categories and subject descriptors:
H.4 Information System Applications
J.1 Administrative Data Processing
J.4 Social and Behaviorals Sciences
K.4.3 Computers and Society: Organizational Impacts
K.4.4 Computers and Society: Electronic Commerce
K.4.6 Management of Computing and Information Systems

General terms:
Design, Management, Security

Additional key words and phrases:
Virtual Organizations, Modelling, Semantic Interoperability, Monitoring, Collaborative Workflows, Contract Management, Breeding Environment, Trust Management

Contents

Chapter 1

Introduction

The Virtual organizations seminar is embedded in the research and educational agenda of the Cinco research group[1] at the Department of Computer Science, University of Helsinki, focusing on B2B interoperability middleware. The goal of this seminar has been to introduce to new students some topical issues in virtual organization management.

The focus of the Cinco (Collaborative and INteroperable COmputing) research group is on the development of collaborative and interoperable computing facilities[2]. Collaborative and interoperable computing is a domain of research where the impact of business or social needs meet the hard computing technology solutions. Collaborative computing refers to interworking of autonomously provided services (e.g., Web Services, Business Applications) and computer support for controlling collaborations between those (e.g., virtual enterprises, electronic business networks). Interoperable computing refers to the variety of challenges placed on the computing and communication platforms, and global computing infrastructure, that support the application-level services. The capability to collaborate means effective capability of mutual communication of information, proposals and commitments, requests and results. Interoperability technology supports this goal on multiple levels: technical aspects (transport of messages), semantic aspects (mapping between information representation and messaging sequences), and pragmatic aspects (the willingness of partners to collaborate, and awareness of the joint process model).

The research challenges on this field include

- service discovery and service type management
- creation of business network models
- trust management
- eContracting
- collaboration or contract breach detection and management, and
- development of a service-oriented software engineering methodology that utilizes these common infrastructure facilities.

International research on virtual organizations reveal one approach to this multifaceted problem area. In the following chapters, based on the presentations prepared for the seminar, various relevant research topics and research projects contributing to the field are studied. The topics cover

[1] The group website: http://cinco.cs.helsinki.fi.

[2] See also: Lea Kutvonen: Challenges of Collaborative and Interoperable Computing—A working version, http://cinco.cs.helsinki.fi/data/files/5452ee57b1/cinco-strategia.pdf

- a holistic, business-level view to virtual organizations
- methods and reference model issues on modeling virtual organizations
- use of metadata for semantic interoperability and managing the virtual organization lifecycle
- collaborative workflows
- virtual organization monitoring
- contract management, and
- trust management issues.

The appendices cover research project case studies that have given background for the various presentations.

In a number of cases, the presentations either relate or have lead to thesis work around the topic.

Chapter 2

Virtual organisation - a perspective on the holistic approach

Tuomas Nurmela

The increase of globalization, turbulence of markets and increased need for global coordination enterprises and public institutions requires investigation of new approaches in computer and management sciences. Virtual organizations (VOs) are one prominent area that is common to both areas of study. For computer sciences, VOs also attack the interoperability crises, requiring design of common support infrastructures to ensure faster integration of applications in future platforms [9].

However, the current state of research suggests that there is a multitude of problems. These include a growing gap between researchers on these fields, partial understanding of the research area as a multidisciplinary science and lack of reference models for either ICT-based solutions or for enterprise models. In computer sciences this has led to an array of focused solutions that may be extremely complicated to integrate with each other and with other, non-VO -based solutions [7]. In management sciences this has resulted in frameworks for management understanding of VOs that lack evaluation of the ICT support infrastructure influence to VOs [22].

The structure of the chapter is as follows: First we focus on the background to virtual enterprises in Section 2.1 to provide a context for the chapter. This is followed by a discussion on the definition of the virtual organizations in Section 2.2. The focus is firstly on providing an overview of different attributes discussed in relation to virtual organizations and secondly on overviewing related terminology to virtual organizations. This discussion attempts to bridge the gap between definitions that concentrate on social network -aspects of virtual organizations with those that concentrate on the IT-systems interaction. Likewise we note the related terms, which provide treatment of similar attributes as virtual organizations. After this, we discuss dynamism in virtual organizations. The focus here is to separate different levels of dynamism, each with their own particular goals. Through this we note that especially medium and high levels of dynamism have non-technical issues that are of central concern. This is followed by a summarization of some of the non-technical perspectives to VOs in section 2.3. Finally we discuss current approaches in ICT support infrastructures in section 2.4 to enable VOs. We first discuss the need to a limit of scope through reference models. However, due to the lack of commonly accepted reference models, the focus shifts to typologies, categorizations of architectures and different attributes of interoperability. Interoperability is especially important since there is no foreseeable ICT support infrastructure that could address all issues relating to VOs. This is especially true for research projects that concentrate to a specific problem area, leaving other areas open. Section 2.3 summarizes the most

significant issues raised, concluding the chapter.

2.1 Background to virtual organizations

This section discusses the background to VOs from two perspectives, that of management sciences and computer sciences. It should be noted that the management sciences approach to virtual organizations is partially based on developments in social sciences. However, the social sciences background of VOs is outside the scope of this chapter.

2.1.1 Management sciences background

The management sciences background to virtual organizations stems from at least three related developments: of the view to enterprises, process management and models on transactions between firms.

View to enterprises is a rather subjective area, but according to one account [33], it has gone through six phases (or trends). First there was focus on functions and organizational structure. However, efficiency brought the need for a concept of process in the second phase. This was followed by a third phase where the interaction between the organization structure (i.e. functions) and processes was investigated. In the fourth phase the processes became the center of attention. Some could argue that at this phase, the organization culture became relevant in order to attempt to portray holistically the way work was carried out in the enterprise. Cultural studies were a major shift to softer management studies applied to the whole enterprise instead of just the group level. In the fifth phase focus shifted to how competencies mapped onto the processes. This coincided with radical process redesign methodologies that required a way to analyse that which is fundamental to the survival of the enterprise. The sixth phase that the mainstream is experiencing currently focuses on understanding these core-competencies of the enterprise and using this knowledge to outsource non-core competency areas. The core competencies -view of the enterprise is in many scenarios also key to understanding the rationalization behind more dynamic VOs.

Process Management developments relate most significantly to the third and fourth phases of the enterprise view. Process management techniques on a high level can be categorized to those that focus on continuous improvement (whether approaching this issue from the quality, logistics or accounting perspective), radical redesign of the way the enterprise does work (as suggested by Business Process Redesign) or combining the previous philosophies to a holistic approach as done by various consultant agencies. For an overview of these see e.g. [17]. It was during these efforts that the concept of *flexibility* (i.e. ability to respond to expected changes, implying manufacturing enterprises [41]) became popularized. While manufacturing improvements through flexibility were key to manufacturing industries in the 80s, today many see that flexibility is not enough for other industries or increased turbulence in the environment. This gave rise to the concept of *agility* (i.e. responding to rapid and unexpected changes [41]). Process management also developed business modelling and provided impetus for development tools to support this. In regards to VOs, supply-chain -oriented VOs are especially close to process management.

Models on transactions between firms relate closely to both process management and core competencies. *Transaction Cost Economics* (TCE) [40] is one of the better known "hard" economic theories. It concentrates on effects of frequency, uncertainty and economies-of-scale to transaction costs. TCE suggests the use of rational decision-making between whether costs will be lower doing it oneself or buying from an external source. The transaction cost is built from internal and external cost. *Internal costs* consist of related non-direct costs including estimates on costs incurred due opportunitism, contradicting fit, and bureaucracy as well as support functions.

External costs relate to direct-costs of specific phases of transaction, including partner search, contract negotiation, operations, monitoring and administraiting. In regards to VOs, based on research on for this chapter, it seemed that computer sciences VO projects explicitly noted only TCE and more specifically the reduction of direct-costs, if any management sciences theory was referred to at all.

Although extensions to TCE discuss relationships that last longer than single business transactions (e.g. Hennart discusses market failure as impetus to joint-ventures [20]), due to their nature, such relationships with loose dependencies are problematic for TCE. These have been addressed by other "softer" theories, such as network models and related theories [25]. *Network models* in general discuss the interconnections of firms and their relative positions of power. Typically network models are far more comprehensive than supply chains that focus on material flows. Network models also often discuss types of dependencies (e.g. social, legal, market transaction-related, governance-related). The Network models typically can be extended through multiple specific viewpoints. *Resource Dependence Theory* [28] as an example is interested in the autonomy and survival of the members of the network. It focuses on dependencies existing between members, suggesting the enterprise should steer its own dependencies to unspecific areas (i.e. areas that can be provided by as many suppliers as possible to maintain maximal autonomy) while trying to ensure others built specific dependeciens (i.e. dependencies of others on the given enterprise) to the enterprise itself. Another view relating to networks is *Social Exchange Theory*. It views interorganizational transactions in the network as social exchanges with interest in development of interdependence and trust. Social Exchange Theory also examines the reciprocality and voluntarity of exchanges. Relating also to networks, *Economic ecology* suggests that natural selection can be extended to market environments. In order to cope with environmental states, an alliance supports specialists that are able to provide critical competencies for specific market conditions. However, in order to adapt, members that are not critical for a given market environment may be required as part of the alliance to ensure adaptation in the future. While the theories are possibly better at modelling different levels of dependencies, they each have their own problematic areas.

2.1.2 Computer sciences background

The computer sciences background to virtual organizations stems from at least four related developments: those of electronic information exchange, workflow systems, middleware and software agent technology. A wide range of both loosely related and centrally related emerging technologies has been discussed by Camarinha-Matos et al. [10], but these are outside the scope of this chapter.

Electronic information exchange has always been standards-driven, with standards on both exchange protocols and message content. Early developments on it center on EDI, EDIFACT (for commercial information) and STEP (for product information). Various vertical standards (e.g. healthcare (HL7, DICOM), finance (FIX, SWIFT) have been developed for specific needs with no overarching standards. Recent work on electronic information exchange has taken advantage of popularisation of structured markup languages (e.g. with XML/EDI) and moved towards process-oriented integration (e.g. RosettaNet) as well as service federation -based approaches (e.g. ebXL). In regards to VOs, electronic information exchange is especially critical to supply-chain -oriented VOs.

Workflow system developments can be categorized into workflow design and specification, inter-task dependency specification and management as well as workflow systems design [31]. These research agendas coexist with standardization efforts (e.g. WfMC WAPI, OMG WAF/jFlow, IETF SWAP, Wf-XML) and research on *advanced transaction models* (ATMs, for an overview

see e.g. [31]). ATMs are mainly intended for enabling flexible data-driven integration by reducing locking requirements for long-lasting transactions (aka. extended transactions, business transactions). However, these may not always be suitable for workflow requirements, requiring other approaches [31]. In regards to VOs, workflow systems are typical to transaction-oriented infrastructures.

Middleware has grown from abstracting separate areas such as distributed computing (through e.g. RPC and later through DCOM and RMI), messaging and transaction management to providing common services through generic platform architectures. With developments to component-based object-oriented middleware in the turn of the millennia, both common and domain-specific services are constantly being upgraded and extended in all major platforms (OMG CORBA, MS .NET, Sun J2EE). Middleware industry leaders have also acquired knowledge from the workflow systems area to further develop application integration frameworks as a part of the platforms. Through this the middleware vendors are addressing issues such as tool-based software development and models of process modelling, which is to be expected given the rise of popularity in the enterprise application integration area. Meanwhile standardization work (e.g. in Sun Java Community Process) is proceeding to assure a certain level of openness and portability between competing solutions. In regards to VOs, for now middleware is especially important to VO systems geared towards providing lower levels of dynamism.

Software agent technology was an especially active area of research in the 1990s. Typical attributes of agent systems contain sociality, autonomy, reactivity and proactivity. Two approaches became prominent: *mobile agents* in which case the agent migrates to the location where task is performed and *multi-agent systems* (MAS), in which case multiple agents, each with incomplete information, coordinate tasks between themselves [19]. Standardization efforts are still ongoing in e.g. OMG (relating to mobile agents) and in FIPA (with FIPA agent architecture). Likewise software agent technologies are closely aligned to developments in ontology and agent programming languages. In regards to VOs, MAS architectures are used especially in systems that are geared towards automated negotiation of differing interests of parties.

2.2 Defining virtual organizations

This section first reviews definitions of virtual organizations, discussing central attributes that vary in definitions. This is followed by a discussion on the related terminology and the issues involved. The focus is to separate the social networks from the computer-mediated ones as well as the business-oriented virtual structures from non-profit organizations. Likewise the previously discussed central attributes are extended with attributes that are relevant to management sciences. Lastly we discuss dynamic VOs, a term that is used to separate the short-timeframe, temporary structures from more permanent forms of virtual organizations.

2.2.1 Definitions of virtual organizations

Literature contains a multitude of definitions for virtual organizations. Compilations of these definitions are available [7, 38]. NIIP project defined Virtual Enterprise as “a termporary consortium or alliance of companies formed to share costs and skills and exploit fast-changing market opportunities”, whereas Byrne et al. define Virtual Corporation as “a temporary network of independent companies - suppliers, customers, even rivals - linked by information technology to share skills, costs and access to one another’s markets. It will have neither central office nor organizational chart. It will have no hierarchy, no vertical integration”, a very contrasting definition to Walton and Whisker who define Virtual Enterprise as “a series of operating ’nodes’ of core competence

which form into a supply chain in order to address a specific opportunity in the market place". Defintions commonly discuss a network and the collaboration between the members of the network [7]. It could be added that clearly adaptation and SME specialization are often part of the assumptions behind the concept.

Given that the term is used as a broad umbrella definition, it appears each use of the term is related to particular problem scope of each author or research project. While this is understandable given the focused nature of research projects, the gap between research programs in managerial and computer sciences studies definition would indicate there is no multidisciplinary concensus. Typically three issues seem to change from definition to another. First is the issue of whether a social dependency or computer-mediated arrangement is central to the definition. Secondly, the timeframe is an issue: some definitions hold on to the temporary nature of VOs while others see virtual organizations also as more permanent arrangements. Thirdly, depending on whether virtual enterprise/corporation (VE) or virtual organization (VO) is used, the definition is aimed more at either profit-making and business or inclusion of non-profit institutions respectively.

Of course the main issue issue is not what is the exact definition but first on what do the different definitions of VO contribute in regards to attributes of VOs and second on what are the related definitions and their contribution. These help understanding VOs from multiple perspectives in order to create VO systems that either are comprehensive or are designed to be integrated to the varying needs.

2.2.2 Hybrid Arrangements and other related terminology

The focus on categorization provided here is on providing grounds for both computer and management sciences to understand the relationships between terms. The categorization is provided in Figure 2.1.

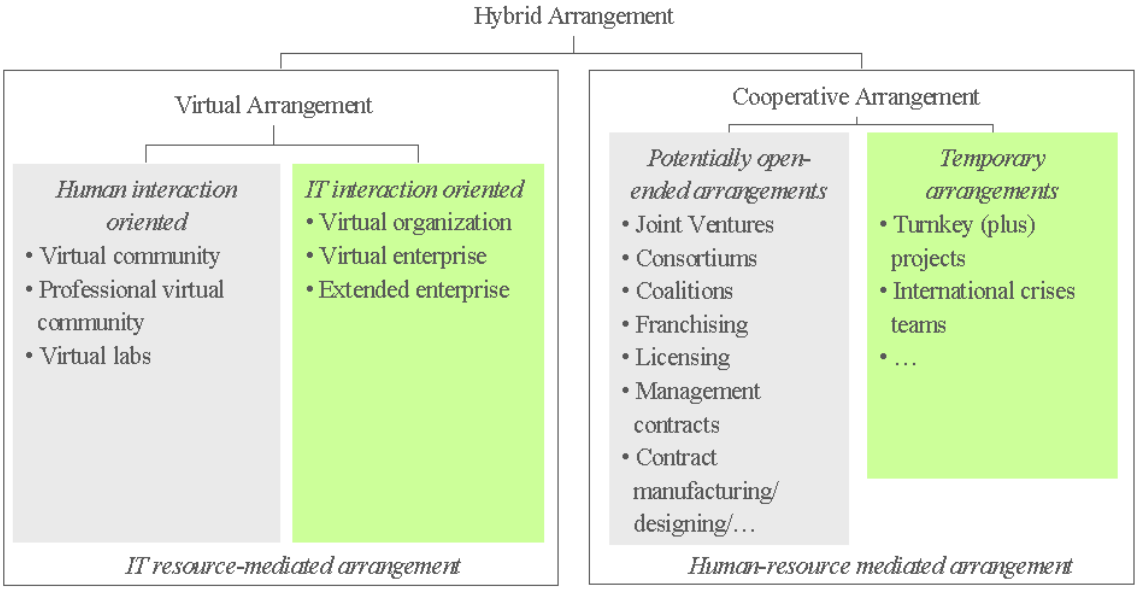


Figure 2.1: Categorization of Virtual Arrangements and Related Terminology.

Hybrid Arrangements are "organizational arrangements that use resources and/or governance structures from more than one existing organization" [2]. The authors of the term originally conceived it to group various organizational arrangements in order to find central commonalities.

However, their discussion implies only arrangements that are here called "cooperative arrangements" (i.e. human-resources mediated arrangements). It should be noted that the possibility of a shared governance structure is something that some VO-programs are somewhat against (e.g. VOSTER [38]) while others remain neutral to it.

Implicitly hybrid arrangements move the viewpoint from reductionism to holism, i.e. hybrid arrangements do not view the issue from a point of view of one member, the dependency between the members or the environmental constraints, rather claiming these form an inseparable entity. Therefore it is advocating emergence (i.e. new attributes that become relavant from the combination of organizational structures). The hybrid also evolves over time. This in practice means changes in the four major issues that from a management perspective deserve focus [2]:

- Breadth of Purpose: The initial purpose, agreement on it among members and the similarity of reality to the agreed is central to minimizing potential opportunistic behaviour of members of the arrangement.
- Boundary defintion: Boundary definition includes definition of boundary between the environment and the members of the hybrid as well as between each member within the hybrid arrangement. Resources of each member put into the hybrid, the power of each member through the governance structure over the hybrid and earlier/latter external commitments of each member all influence boundary definition. *Boundary permeability* (i.e. the level of freedom for individual members or the common agreement allows elements such as resources or governance arrangement related issues such as obligations and authority to cross the boundary) affects how value is created by the hybrid.
- Realization of value creation: Value creation is based on managing the interdependence through coordination between members. Interdependence can be pooled (i.e. resources are treated as a resource pool that can be utilized by members), sequential (i.e. relating to handoff points where the function of a handoff point is critical) or reciprocal (i.e. swap of resources between members requiring a broader fit over time). Value creation is important especially in regards to models of transactions between firms, because e.g. reciprocal interdependence is complicated to justify through Transaction Cost Economics or Resource Dependence Theory.
- Stability mechanisms: Stability mechanisms are required to reap the benefits of flexibility and efficiency of hybrid arrangements compared to completely permanent structures (e.g. acquisition). However, the first stability mechanisms develop slowly. E.g. even the understanding of cultural differences and the processes relating to the arrangement on a person level in the operative work takes time, for communication is never instantaneous. Secondly, naturally the autonomy of members is threatened by use of stability mechanisms, for they intrude to the organization. External institutions, exchange of personnel and structured workshops are just some of the methods used to avoid this.

In addition there are at least three issues that are of concern in hybrid arrangements: opportunistic behaviour, evolution and emergence. *Opportunitistic behaviour*, whether for profit or learning maximization, has been widely discussed in areas such as risk management, game theory and strategic alliances research. Risk of opportunistic behaviour relates to the amount of stability mechanisms used and *trust* between members. Trust is often seen to be an issue of contractual arrangements, but recent strategic alliances research has discussed evolution of relationship from contractual trust to relationship-based trust on longer relationships [26] following group dynamics-based team development [4]. Such approaches suggest that also for more temporary arrangements

other group dynamics-based theories related to the temporality of teams, such as presented by Gersnick [15], could be investigated.

However, opportunitistic behaviour cannot be seen as *merely* an issue of trust. Both *exogenous changes* (i.e. external, environmental) to resource allocation and *endogenous changes* (i.e. internal to alliance or intra-firm) can incite opportunistic behaviour [16, 24]. Endogenous changes suggest that e.g. the mere perception of opportunitistic behaviour by personnel working in a decision-making role in one of the members of the network can start counter-acvitities that result as a whole in opportunitistic behavior emerging within the hybrid.

Evolution (relating to the life cycle of the hybrid arrangement) also contains more issues than mere development of trust. To understand hybrid arrangements from the behavioural perspective the following attributes (in addition to those discussed earlier) are important, affecting creation and the potential evolution [25]:

- Extension: Describes if the arrangement is horizontal (i.e. overlapping functionality between parties dominates arrangement) or vertical (i.e. location of members on different areas of the value chain or outsourcing a function to other arrangement members). This is related to aim and nature of the arrangement.
- Scope: Describes if the arrangement is bilateral (i.e. between two parties) or multilateral (i.e. between more than two parties).
- Aim: Describes if the arrangement is mainly for economies of scale (i.e. reducing costs of operations through size), economies of scope (i.e. benefits gained from improved product/service selection), knowledge transfer or whether it is imposed (e.g. forced by government as a requirement for getting a contract). Also describes how these are interrelated if due to more than one reason.
- Equity: Describes if the arrangements concern no equity, investment into one partner, a reciprocal equity swap (e.g. to signify willingness of the parties), establishment of a separate organization (in case of a Joint Venture) or investment into a governance structure (sometimes called "a superstructure").
- Nature: Describes if the arrangement is focused on concentration (i.e. overlapping functionality is partially reduced) or complementary efforts.
- Timeframe: Describes from a contractual point-of-view if the arrangement is open-ended or a temporary arrangement. In regards to VOs, the VOSTER project is one of the strong advocates of VOs being typically termporary arrangements with fixed lifespan.

Evolution and opportunitistic behaviour (especially through endogenous changes) are examples of *emergence* in networks. Emergence is a concept studied mostly under chaos theory and complexity. Typically emergence is strongly related to (or even synonymous to) holism that contradicts reduncti onism (e.g. assumption that all properties of a system remain even if it is divided in order to help investigation) [35]. Emergence is typically used to indicate the notion that "the whole is more than the sum of its parts". To compromise between reductionistic and holistic views, *weak emergence* has been expressed as "a systematic way of science to introduce theoretical terms that are not observed but are defined by the relations to the observables" [34] to provide some means of avoiding either of the extreme views.

Jeffrey Goldstein's work helps provide more practicality to the evalution of emergence. Goldstein provides five features that can test whether a property classifies as emergent. The features

are radical novelty (i.e. sudden apprearence of a non-deducible property), coherence/correlation (i.e. integration to the whole and maintaining of identity over time), global or macro-level (i.e. shows in the whole system rather than something that can be pinpointed as a property of a particular part), dynamicity (i.e. emerges over time, non-observable at the beginning) and ostensivity (i.e. recognizing emergence through perceiving its influence. Therefore an emergent property typically has is different yet similar appearence) [18]. These features are discussed in context of moral and ethical issues relating to VOs, yet they can generally help to distinguish social emergent phenomena.

Hybrid arrangements are here used as a superset for both virtual arrangements (i.e. IT-resource mediated arrangements) and cooperative arrangements (i.e. human-resource mediated arrangements). The attributes of hybrid arrangements reflect in either form of arrangement. Still, they are not defining features. *Virtual arrangements* stress the role of IT, containing virtual organizations and virtual enterprises that are particular forms of it focusing most on IT-systems interaction. On the other hand virtual communities concentrate to human-interaction *with* the help of the IT-systems. As can be seen, purely social networks are not within virtual arrangements. Also, virtual arrangements are not required to be temporary arrangements (i.e. they can be open-ended). It should be noted that other terms such as "Networked organizations" [5] and "Collaborative Networked Organizations" [7] have been offered as a similar collective term as virtual arrangements, but have emphasized different issues. Here different forms of arrangements are used more for categorization. *Cooperative arrangements* on the other hand are used collectively for forms of cooperation from the management perspective. These are by nature contractual.

It should be noted that virtual and cooperative arrangements are not mutually excluding, although some authors do make this leap, suggesting that e.g. VOs are a more advanced form of organizing (e.g. [39]). However, it would seem VOs are rather a form of organizing suitable for certain contexts. Therefore here the classification to arrangements merely reflects *the most significant feature* of a specific arrangement. The sub-categorization below virtual arrangements is based on [12] and [11] with minor modifitations to definitions provided in these.

Virtual enterprises are non-public, profit-oriented virtual organizations. Attributes of virtual enterprises (VEs) can be used to understand the commonalities and possible differences to hybrid arrangements and the management viewpoint it represents. It has been suggested that ICT support infrastructures could be described from the perspetive of five attributes of VEs. These are [5]:

- Duration: Duration can range from a single business opportunity (requiring support for rapid creation and dissolution) to a long-term alliance (differing from the contractual view provided in Timeframe attribute of hybrids). Authors note that automotive industries and food industries typically have long-term alliances.

- Topology: Topology is closely related to VE evolution and dynamism of VO. In a fixed structure it is assumed that during the life cycle there will be a relatively small amount of changes to the members of the VE. Dynamic (aka. variable) topology allows dynamic joining and leaving to the topology. This also includes temporary arrangements.

- Participation: Participation relates to whether a member is allowed to join multiple alliances or restricted to a single alliance (from the contractual point of view, e.g. airline multilateral alliances are of this type, limiting alliances within the industry members).

- Coordination: Coordination relates to the topology of the VE. A star-like coordination structure is typical to VEs with a dominant member that uses a network of suppliers, with the dominant company defining the business process model, interoperability standards etc. In

case no dominant member exists the coordination is provided in a democratic alliance. In this case members are somewhat equal.

- Visibility: In the case of single-level visibility a node sees its next-hop neighbours. This is typical to supply-chains. In multilevel visibility a node can receive rights to access to other nodes as well. Visibility is an important part of the ICT support infrastructure security especially for transaction-oriented layered approaches, as portrayed by the PRODNET architecture (see Appendix G).

VOSTER defines seven additional characteristics that exemplify their viewpoint on VOs [38]. Most markedly these stress the lack of head quarters ("non-institutionalization"), support of mass customisation and personalization ("individualization"), reduction of synchronous human interaction ("asyncronization"), potentially making e.g. products or people immaterial through representation ("dematerialization"), focus on defining and developing core competence by each participant and the integration of these ("integrative atomisation"), clear creation and dissolution phases ("temporalization") and potential lack of geographic proximity ("delocalisation", although it must be noted that if the VO is in the product manufacturing business, there is no getting away from basic logistics). VOSTER uses these characteristics to analyse and evaluate a multitude of VO projects.

Extended enterprise relates to forms of networks where one firm is the dominant player that uses suppliers to extend its own boundardies. Therefore extended enterprises are a subset of VEs that directly relate to virtual integration [13]. It can be said to either emphasise a particular form of VE or define it as a competitive form to VE, in case the author's definition of VE stresses equal distribution of power between the participants to VE (e.g. [38], although from the management perspective, "equal power" is mostly just a theoretical view rather than something that occurs in actual reality.

Virtual Communities (VCs) are virtual arrangements that are related to enhancing the social networks and human interaction functionality. VCs can contain forms that are geared to describe enhanced social interaction related thorugh use of ICT to coordinate their movements, like smartmobs [32]. They can be established for hobby or professional purposes.

Professional Virtual Communities (PVCs) are defined as "a social system of networks of professionals, who use computer technologies to mediate their relationships" [7]. Therefore they are not profit-oriented necessarily, although may well be as is in the case of e.g. consultancy networks mediated by computer networks. PVCs do utilize different types of technologies. E.g. portal web chats dedicated to specific specialists and shared knowledge databases are forms of PVCs. In managerial sciences the closest representation could be communities-of-practice [3] to link PVCs to aim of knowledge transfer that is typical to them.

Virtual Laboratories (VLs) include both collaborative environments for a specific area of study (instead of community) as well as remote laboratory facilities for sharing costs of expensive equipment. VLs also include virtual product development spaces. Potential areas of use include education, research, medicine and other knowledge intensive industries. Typically VLs enable remote operations, simulation of actions in the remote lab, storage of data (e.g. from tests) and collaboration tools.

The significance of the categorization comes when we try to understand both the computer and management sciences viewpoint, as it helps us identify the common misconceptions between the two parties. Management sciences often discusses virtual organizations as it would cover all virtual arrangements. However, this is not the case for computer sciences, which are rather concentrated on the IT-systems' interaction related forms of virtual arrangements, i.e. virtual organizations,

virtual enterprises and extended exterprises as they were used here. Secondly, it reflects the typical misconception of computer sciences that virtual organizations are an organizational form in themselves. However, if we take e.g. a market opportunity like bidding for a construction-project abroad, it would involve turnkey project arrangements (i.e. a particular form of that has a project that has a lead supplier and subcontractors and is responsible for delivery of the whole projected package. Likewise, it may involve arranging temporary operational responsibility for e.g. the plant that is to be built and transfer of knowledge or other additional arrangements, forming to be a turnkey plus project). However, because of evolution the cooperative arrangement may well change during the project, affecting contractual issues as well as resourcing. If this were to be managed through virtual organization IT-systems negotiations between supplier parties, it would seem to benefit from understanding what type of cooperative arrangements exists and the basics of their interrelation possibilities, as each of the forms of cooperative arrangements have their specific challenges in addition to those discussed by hybrid organizations (and likewise, mid-range theories provide additional context-dependent issues). Likewise, it could be argued that the use of IT-systems-based virtual organizations can diminish the boundary definition problem of cooperative arrangements, but if a cooperative arrangement is indeed in effect, it cannot be completely removed.

2.2.3 Dynamic virtual organizations

Dynamic VO/VEs are more of a concept than an actual applicable organization type, although they have been discussed as such. *Ad-hocracies* are seen as temporary project-organizations or task-forces that combine a set of specialists to complete a specific assignment forfeiting hierarchies and power differences. Discusssion on Ad-hocracies also notes that new management techniques combined with new technology provide "'... productive capability alive with intelligence, alive with information, so that at its maximum its completely flexible; one could re-organize the plant from hour to hour if one wished to do so'. And what is true of the plant is increasingly true for the entire organization" [36]. Dynamic VOs envision the same benefits for the entire network.

Dynamicity as discussed here relates to the timeframe it takes to create and dissolve the VO/VE or adapt to a change to evolve it. Dynamicity is always relative to the given context of the project. The context can range from "'one-click' shopping... to many years in aerospace industry" [21], which markedly affects the definition of VO itself. Therefore it seems reasonable to classify requirements of dynamism of VO/VEs in research projects based on context requirements, for the whole end-solution may turn out to be unsuitable for different contexts.

Current VO/VE scenarios [23] suggest at least three levels of dynamism would be required to include both dynamic leaving and joining and foreseen temporary structures. *Minimum dynamism* relates to fixed structures where the VE needs to enable changes to interconnected systems (e.g. updates of ERP software, changing ERP software from one vendor solution to another, interface modifications to software, change of hardware platform etc). *Moderate dynamism* requires enabling evolution of the network during its operation-phase of life cycle to accommodate new members as well as leaving of members in addition to minimum dynamism. Therefore the VO system management needs to support activities relating to evolution (e.g. managing roles and responsibilities of parties through modelling support etc). *High dynamism* requires rapid creation and dissolution of the VE in addition to dynamic evolution to enable rapid reaction to market opportunities. This category also includes VOs that are e.g. geared towards crisis management on a global scale to e.g. deploy humanitarian aid workers efficiently. Therefore the VO system needs to support activities relating to these life cycle phases (such as proactive search, formation of framework agreement to enable faster creation of final contract etc). While minimum dynamism

is mostly an ICT support infrastructure issue, the other two require extensive work also on the management of VEs. While the focus is here on VEs, it should be stressed high dynamicity is extremely beneficial for certain non-profit VOs as well.

VEs can be built from an "open universe" or by forming an framework agreement with the potential members to provide a stability mechanism to build trust and counter opportunitism in a "controlled-borders universe". This difference can be visible in the ICT support infrastructure as the open universe -model fits better to VO-projects using a directory for searching services (e.g. Web Services -based systems). It should be noted there are multiple namings for the "controlled-borders universe" including VO Breeding Environment (VBE), Source Networks, Network and Support Network [38]. Figure 2.2 describes the two approaches, their phasing and challenges.

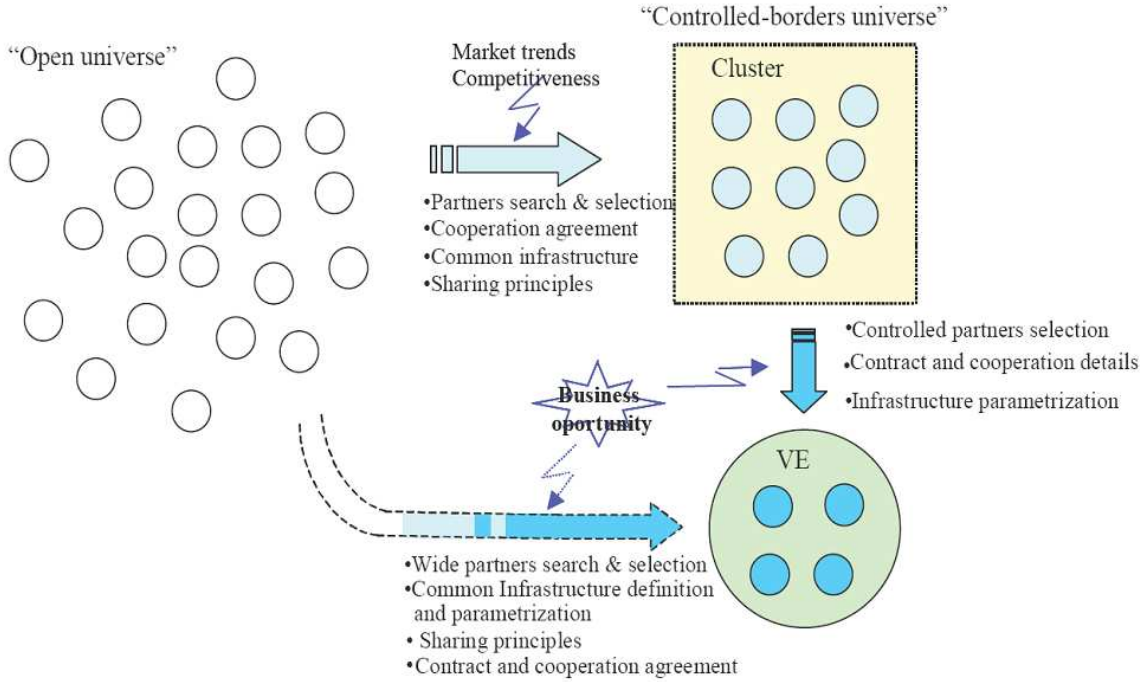


Figure 2.2: Two approaches to the formation of virtual organizations [8].

Both views are interesting in their own way. In regards to a *controlled-border universe environment*, it remains open how much the stabilizing mechanism can support VO creation and management. Some work has suggested that industry clusters (i.e. "a geographic concentration of interconnected companies and institutions in a particular field" [30]) could prove useful for this purpose [7]. The problem with these approaches from the management sciences perspective seems to be that firstly the creation of industrial clusters is related to chance events, government actions and/or macrolevel factors. While the firms can work as an impetus, they can have very limited say to these factors. Secondly, cluster formation for world-class clusters takes a long time, and a world-class environment is likely to be required to provide the true specialization to maintain a competitive advantage. Thirdly, industry clusters themselves can deteriorate based on changes in macrolevel factors [29] which are outside the control of a particular group of firms. Therefore firstly, this applies only if a cluster exists and secondly, some approach to managing the framework agreement between parties would be expected to react to changes in the cluster. Thirdly, some means of reacting to a member's state (e.g. poor financial condition etc.) changes would be required but this would have to avoid opportunitism.

In regards to an *open universe environment*, it seems that their dynamicity is severely limited by the requirement to negotiate what the share of each participant will be on every occasion. Therefore reacting to market opportunities may be difficult. On the other hand, open universe environments seem to better support arguments that geographic location is somewhat irrelevant to VOs, as this implies that clusters are not required, running counter to the view of using a controlled-border universe that would be based on clusters. Of course location can only be irrelevant to some types of VOs, as if there are material flows between the VO members, this automatically leads to logistics issues and potentially (in international VOs) to questions whether export/import of goods is preferred over foreign direct investment.

2.3 Perspectives on VOs

While VOs can be addressed as a technological challenge, many other concerns are raised by the legal, ethical, social and cultural perspectives. Likewise, as an area of a multidisciplinary research field, other sciences have provided contributions that can be of use when investigating VOs as a holistic approach. In terms of the reviewed projects (see Appendix), the issues seemed to play little role in the projects, although the subject has re-surfaced in research [18].

2.3.1 Legal-ethical perspective

Immaterial Property Rights (IPR) Management relates to firstly agreeing on how trademarks (brands, logos), copyrighted material, patents etc. will be used by the VE if one of the members is able to contribute them. Secondly, if the VO itself creates any IPR related innovations, the ownership of these becomes an issue.

Liability of the products and services provided by the VE is of central concern especially for manufacturing industries. The participants to VEs need to agree how product faults will be handled and how to manage potential liabilities claims after the dissolution of the VE.

Work especially in medium- and high-level dynamic virtual organizations can stress the workforce in a very different manner than in traditional organizations. Workforce job security, both in terms of market changes and layoffs due to moving work to cheaper countries needs to be addressed. Likewise the work roles may change from working within one organization to another, requiring corporate social responsibility over maintaining education level. Likewise, allegience to the virtual organization compared to one's "own organization" may become a complex issue. This is furthermore complicated by corporations that move towards using more temporary workforce. Whether one could even expect loyalty towards the organization in such arrangements is an open issue.

2.3.2 Social-cultural perspective

The socio-cultural perspective is especially important for medium- or high-level dynamic VOs as well as VOs looking to provide computer-mediated human interaction. Many VO scenarios stress the changing relationship between personnel and the organization with increase towards individuality and personal responsibility. VOs need to address at least the following factors:

- will the VO have its own management or operative personnel and if it will, what will their relationship to the individual companies be
- how personnel management will be done and what is the role of group leaders

- if the VO is mainly created to enable co-operation of disperate groups, how will the social interaction of the workers be satisfied
- how will the workers be trained for the VO's multicultural environment
- what sort of working habits in general will best suit this potentially different style of working
- will personnel develop identification with the VO and how can changes for this development be supported

2.3.3 Other perspectives

Authors prompting for the multidisciplinary nature of VO research have been keen to draw connections between research subjects and various other fields. In discussing multi-agent systems [8], the authors discuss how semiotics, complexity, graph theory, game theory and multiple other research fields contain potential areas of applications, although not without limitations. Information systems analysis and design problems are approached through semiotics in attempting to demonstrate how semiotics can be used for organizational models and agent-based systems [14].

In addition to these, given that groupware for collaborative environments and virtual laboratories supports social networks, cognitive sciences developments (through research on distributed cognition) are obviously related to VOs.

2.4 ICT support infrastructures

While VOs can be approached in a multidisciplinary fashion, an assumption that a "grand unified theory" will be reached seems unpractical from the computer sciences perspective. More likely some of the perspectives will evolve to separate areas of study while others wither away. However, to ensure the interoperability of solutions, a reference model would be crucial to ensure shared view of the field. This would additionally help research projects, which do not typically have enough resources to investigate the whole issue and as such steer into limited or artificial research problems.

Given the lack of a commonly accepted reference model for either enterprise models [37] or IT architectures, this section discusses first typologies of VO architectures found in projects, reducing the scope of the VO from taking into account all related attributes. This if followed by one categorization of VO architectures. Finally, given that interoperability cannot be discussed through a reference model, we conclude by discussing different attributes of interoperability that need to be taken into account when designing or evaluating a VO system.

2.4.1 Typologies for IT-based VO-projects

Instead of approaching all attributes for VEs as discussed in section 2.2, the current view suggests that there will be few dominant forms of VEs to which the ICT support infrastructure will need to respond to. These relate to levels of dynamism that were discussed in subsection 2.2.3.

One approach is to categorize VO forms based on the assumed scenarios [1]: *Type A* is based on a dominant partner that builds long-term relationship to multiple SMEs. *Type B* is a VO that lacks a dominant partner and is based on dynamic project organizations. *Type C* is a termporary arrangement created to explore short-term market opportunities. This typology is similar to the categorization discussed in dynamic VO's, but explores the area only from VO-perspective, disregarding the requirements for IT systems based on the type of VO.

Reviewing the current field of research [22], it has been noted that approximately two thirds of European VO-projects use one of three topologies (determining the structure of material and information flows as well as power and governance structure). The topology of VEs is either an implicit assumption or explicit in the models of the projects. A *supply chain* is assumed to exist by process-oriented projects. Projects using this topology are geared to managing long-lasting dependencies, although most are capable of supporting adaptation to rapid logistical changes. *Star topologies* are typical to projects geared to support one central (dominant) member (e.g. the case of original equipment manufacturers), tiered supplier networks (e.g. auto industries) or establishment of consortia (e.g. construction company turnkey projects). *Peer-to-peer topologies* exist when there is no central coordination. All (or most) members are deemed equal in terms of power over the virtual organization functionality. Topology mostly exists in knowledge-intensive industries (e.g. biotech).

2.4.2 ICT support infrastructures

ICT support infrastructures categorize disparate virtual arrangements approaches, as was discussed in 2.2. The categorization mainly relies on separating computer-mediated approaches aimed to support IT system collaboration from solutions enhancing social networks. The social networks aspect also includes ways of sharing research tools to share costs.

However, the categorization also has other elements namely classification of basic horizontal infrastructures and discussion on attributes found in VO projects each of the types of basic horizontal infrastructures, on which they concentrate. This is to be expected given their background on manufacturing industries. Likewise their discussion reflects the typical concentration to "how"-questions of computer sciences approach to VEs (whereas management sciences are also inclined to discuss the "why"-questions). The categorization is presented in Figure 2.3.

Basic horizontal infrastructures can be divided to transaction-oriented and layered infrastructures; agent-based infrastructures and service federation-based infrastructures.

Transaction-oriented and layered infrastructures were used on the earliest systems, being reminiscent of workflow systems that were extended with an access layer and typically using a distributed or a federated (i.e. transparent vendor interoperability providing) database management system. Due to the historical background of the systems, some standards exists for data exchange, workflow concepts are well known, various approaches for distributed transaction management exist and in addition middleware technologies are intergratable. Current limitations include lack of reference model for platform architecture, complexity solution both in terms of operation and interoperability between solutions (limiting both low and medium dynamism), complete lack of support for high dynamism (i.e. fast creation and dissolution of cluster) and lack of support for monitoring and managing the environment for virtual organization creation.

Agent-based infrastructures refer to encapsulated computational systems that infere to provide autonomous behaviour based on MAS. MAS are not necessarily about systems integration as they about representation of the enterprise for some system relating to a virtual organization. Due to historical background research projects typically use development platforms that follow the FIPA agent architecture. Likewise from the VE life cycle perspective the creation and operation of VE contract management is easier and fits well to the infrastructure, and the combination of autonomy, agent communication and federated information systems provides support for dynamic role change of members, scalability as well as incomplete information -based decision making. Current limitations include lack of suitability for over the Internet operations, lack of production implementations and limited integration of security, persistence to the system and modelling to

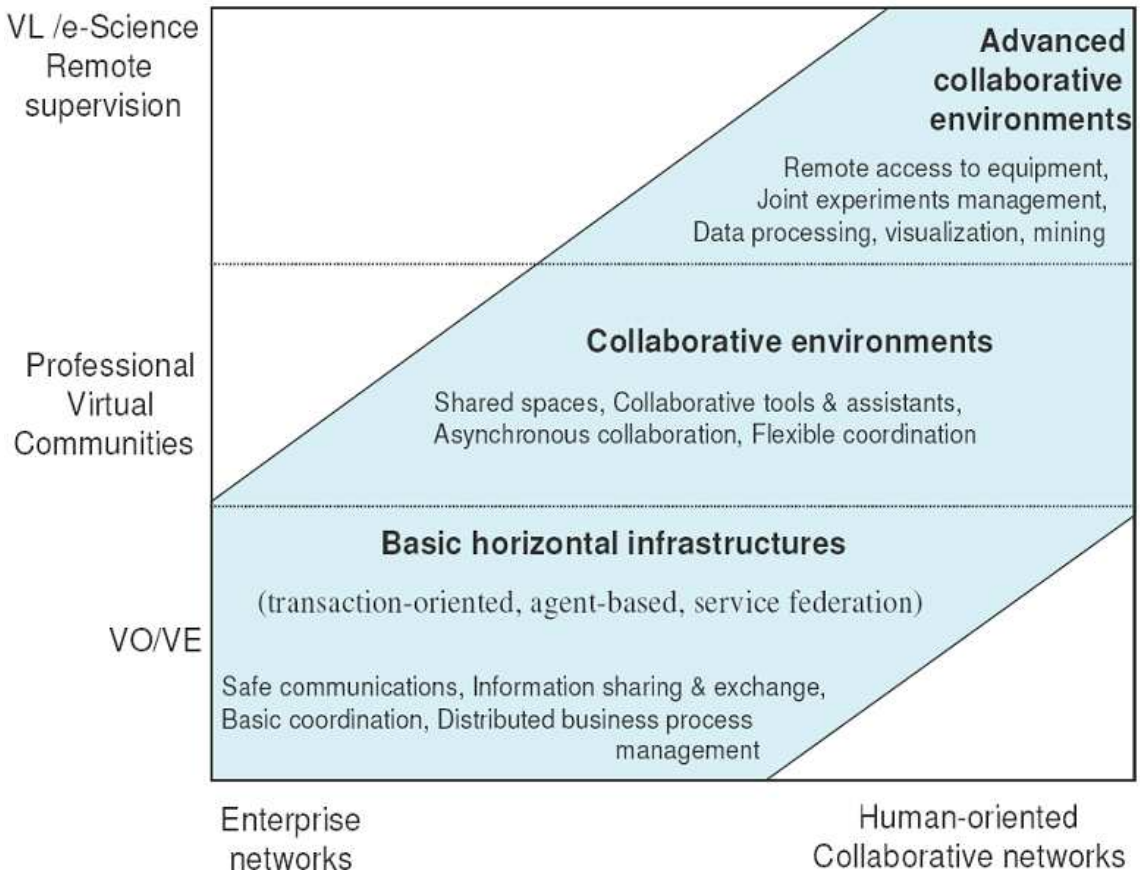


Figure 2.3: Categorization of ICT Support Infrastructures [11].

support the available dynamism of system.

Service federation -based infrastructures have as a dominant feature support for the composition of a business service from several separate services. Typically services are advertised through a distributed directory infrastructure. Portals are considered a special case of service federation by the authors. However, it should be noted that the human-interaction aspect often outweights the application integration aspect of portals. Therefore portals can significantly contribute to building a collaborative environment and in some cases are more of the latter (e.g. sourceforge.org). With ongoing work in Web Services, many of the projects are basing the approach on UDDI, WSDL, SOAP and additional standards as Web Services work progresses. Current limitations are related to the ongoing work on Web Services where security and privacy mechanisms are developed and semantic search through shared ontologies is still at its infancy in terms of adoption to projects.

Collaborative environments can be categorized to mainly simple tools and packaged solutions supporting PVCs (although the authors of the categorization do not provide such). Simple tools include web chat, IRC, instant messaging, email, web forums, net collaboration tools etc. Packaged solutions (used in this sense) are a collection of tools in a comprehensive package to facilitate human interaction. Naturally since human collaboration is at center stage, typically resulting to asynchronous communication and local processes based execution, the processes are not rigid and therefore workflow based systems do not typically adapt to requirements. Still, authors note some flexible workflow systems have been proposed. Current limitations include a wide variety of tools often lacking integration, poor support for IPR management and lack of coordination support for

group work.

Virtual Laboratories (VLs) support tools are centered on a specific area contrary to the more generic collaborative environments. Current limitations include lack of security mechanisms, lack of implementation experience, lack of integration of cooperation management to tools and poor error recovery mechanisms. No real categorization for the tools available in this group was found during the preparation of the chapter.

It should be noted that while the different levels of ICT support infrastructures are not intended to be handled by a single system, the interoperability of different levels of the system is seen as one of the key trends in the enterprise resource planning market. This relates to enabling front-end oriented portal systems (e.g. Microsoft Sharepoint, SAP Enterprise Portal) central to handling unstructured data to also access the structured data of backend systems (e.g. SAP CRM). It is not unforeseeable that similar interoperability requirements would be part of different levels of ICT Support Infrastructures (e.g. Collaborative environment solution accessing basic horizontal infrastructure data). With this in mind, we turn ways of categorizing interoperability capabilities from an ICT support infrastructure viewpoint.

2.4.3 Interoperability

Interoperability contains at least four specific issues: level, depth, target and layer. *Level of interoperability* defines whether interoperability is intended to include systems outside the VO (*external-to-VO interoperability*, i.e. the system has integration support either for other VO-systems or supports interfaces to non-VO systems such as operative systems) or designed to function within the VO (*internal-to-VO interoperability*, i.e. requires use of the same VO system by all members of the VO). These issues are especially relevant in understanding issues regarding the level of dynamism taken into account in the design and through that the flexibility of configuration in the VO. Another view to this issue are industry-type approaches. E.g. from the manufacturing perspective, the cell-level integration (i.e. integration required for interoperability of e.g. robots in one function such as assembly) and shop-floor level (i.e. integration of different functions within manufacturing, e.g. from assembly to painting) are levels of interoperability fitting for manufacturing industries [6].

Depth of interoperability can be approached from two viewpoints: firstly from a high-level technical perspective [27], VO can provide *technical interoperability* (i.e. application-layer protocols are shared by members), *semantic interoperability* (i.e. members can infere from expressions that are not expressed in the syntactically same manner through e.g. translation services) or *pragmatic interoperability* (i.e. members can communicate higher-layer issues such as fulfilment constraints, capabilities, trust, goals etc. through e.g. separate ontologies), Secondly the issue can be approached from the organizational impact perspective, as in [6], in which case a VO can be based on *basic communication* (i.e. business transaction-level syntactic interoperability), *application integration* (i.e. support of developing interconnectivity by abstracting the actual technical interface), *business integration* (i.e. support of distributed business process modelling through tool and coordination through e.g. a framework whereby the managers of the process become involved) or *team integration* (i.e. providing support for the collaboration of members across organizations). Of course as with the categorization, this need not be looked as a strict layereing given that it contains both elements of systems integration and groupware. It should be noted that application integration has especially been one of the latest hot topics in the computer industry, also giving rise to classifications based on purely technical solution perspective on high-level.

Layer of interoperability refers to layer on which interoperability is intended to take place while at the same time indicating constraints to interoperability. Contraints refer to the require-

ments for specific systems in terms of operating system, middleware technology, domain frameworks and related infrastructure (e.g. DBMS, directories) in a platform solution (i.e. middleware-based approach). *Application-layer* interoperability basically leaves interoperability as the problem of application developer. This includes non-configuration -based domain frameworks. *Meta-model-driven* interoperability assumes that the modelling-phase can be used to design the interoperability. Code-generation tools are used to translate the meta-model to support a specific middleware or programming language used by a specific member of a VO. *Middleware service* interoperability provides partial means of integration by providing some level of configurable interoperability to the application as well as potentially providing e.g. a connector-architecture to extend integration to enterprise systems. Middleware service can be based on existing middleware standards or be provided by an integrated platform in which case it may not provide the basic services similar to standards-based middleware (while providing the integration related services). *Federation-based* interoperability acknowledges that not everything needs to be integrated and use a level of indirection, such as a directory-service, to provide loose-coupling for integration of application services.

2.4.4 An example categorization of research projects

Some of the previous categorizations are applied in Table 2.1 to categorize projects described in the Appendix. The categorization could also include other categorizations such as the type of virtual organization, level of dynamism and topology of the VO.

2.5 Conclusions

There are currently no generally accepted virtual organization reference models for either management scinces or computer sciences. Likewise the definition of virtual organizations varies between different fields of science. Although definitions within computer sciences research programs are moving towards a collectively agreed form, this is a long way from a combined view of virtual organizations that would cover both management and computer sciences. This is to be expected, given the early state of research.

Central to the variances in definition is whether human- or IT-interaction is stressed by the definition. Some definitions also enforce the profit-making requirement or the temporary nature of collaboration, others do not. Extending this, the dynamism is another element that clearly separates different types of virtual organization definitions.

Besides definition differences, there are worries that non-technical dimensions of virtual organizations, such as those portrayed by the social-cultural and legal-etchical perspectives, will be neglected in research of virtual organizations. This is to be expected, because the soft, social issues are not within the traditional research agenda of computer sciences.

This article used hybrid organizations in order to provide an overview of the related terminology and discuss the similarities and differences in issues whether discussing VOs from a social network perspective or IT-system interaction perspective. We also noted that any type of virtual arrangement may well be related to a cooperative arrangement(s) concurrently, leading to a need to understand the relationships between these elements. Also, a management sciences discussion relate to issues surrounding the evolution of VOs. These issues need to be at some level tackled, whether in the VO system or in the VO organization, if VOs need to be able to function for a longer period of time.

While there are no reference models, classifications of typologies and architectures do exist for IT-architectures. Combining these with a classification of interoperability does enable a certain

	Architecture (2.4.2)	**Classification of interoperability (2.4.3)**			
Project	**Basic horizontal infrastructure**	**Level**	**Depth (technical)**	**Depth (organizational)**	**Layer**
AO4BPEL	(infra)	(level)	(tech depth)	(org depth)	(layer)
CrossFlow	Service-federation	Internal-to-VO	Semantic	Application	Federation-based
E-ADOME	(infra)	(level)	(tech depth)	(org depth)	(layer)
FETISH-ETF	Service-federation	External-to-VO	Technical	Application	Federation-based
Massyve	Agent-based	Internal-to-VO	Technical	Application	Integrated platform
Meteor-S	Service-federation	External-to-VO	Semantic	Application	Integrated platform
Prodnet	Transaction-oriented	External-to-VO	Technical	Business	Integrated platform
Projects at U. of Tilburg	Service-federation	External-to-VO	Technical	Business	Meta-model-driven
WebPilarcos	(infra)	(level)	(tech depth)	(org depth)	(layer)
WISE	Service-federation	External-to-VO	Pragmatic	Business	Meta-model-driven

Table 2.1: The architectures and classifications of interoperability of the Appendix research projects.

level of comparison between solutions, which may help the reuse of research results. While this paper provided one classicification scheme, VO research projects such as VOSTER have also provided classifications.

With no standard intefaces between solutions, it seems that most of the solutions are more or less suitable for integration between themselves or through a lenghty configuration. It would seem that the interoperability of different solutions seems to be no better than that provided by current developments in integration frameworks of many middleware products. Therefore it may be that configurable middleware built on top of today's middleware standards could be extended to provide support for VO requirements. This would provide multiple benefits: first, the approach would support transactional and layered architectures as well as service federations. Second, the approach would enable extending the basic horizontal infrastructure with collaborative environments provided by off-the-shelf portal frameworks. Thirdly, integration to operational systems would be handled mostly by different types of existing connector solutions.

Bibliography

[1] AFSARMANESH, H., CAMARINHA-MATOS, L., AND MARIK, V. *Challenges of Collaborative Networks in Europe*. Kluwer Academic Publishing, 2004, pp. 77–90.

[2] BORYS, B., AND JEMISON, D. B. Hybrid arrangements as strategic alliances: Theoretical issues in organizational combinations. *Academy of Management Review 14*, 2 (Apr. 1989).

[3] BROWN, J. S., AND DUGUID, P. Organizational learning and communities-of-practice: Towards a unified view of working, learning and innovation. *Organization Science 2*, 1 (1991).

[4] B.W.TUCKMAN, M. Stages of small group development revisited. *Group and Organization Studies*, 2 (1977), 419–427.

[5] CAMARINHA-MATOS, L., AND AFSARMANESH, H. *The virtual enterprise concept*. Kluwer Academic Publishing, Sept. 1999.

[6] CAMARINHA-MATOS, L., AND AFSARMANESH, H. Elements of a base VE infrastructure. *Journal of Computers in Industry 51*, 2 (2003), 139–163.

[7] CAMARINHA-MATOS, L., AND AFSARMANESH, H., Eds. *Collaborative Networked Orgazations*. Kluwer Academic Publishing, 2004, ch. 1, pp. 3–10.

[8] CAMARINHA-MATOS, L., AND AFSARMANESH, H. *Formal modeling methods for collaborative networks*. Kluwer Academic Publishing, 2004, pp. 15–26.

[9] CAMARINHA-MATOS, L., AFSARMANESH, H., AND ABREU, A. *Targeting major new trends*. Kluwer Academic Publishing, 2004, pp. 69–76.

[10] CAMARINHA-MATOS, L., TSCHAMMER, V., AND AFSARMANESH, H. *On emerging technologies for VO*. Kluwer Academic Publishing, 2004, pp. 207–224.

[11] CAMARINHA-MATOS, L. M. Infrastructures for virtual organizations - where we are. In *Proceedings of ETFA'03 - 9th Int. Conf. On Emerging Technologies and Factory Automation, Lisbon, Portugal, 16-19 Sept 2003* (2003). URL `http://www.uninova.pt/~thinkcreative/ETFAvoster.PDF`.

[12] CAMARINHA-MATOS, L. M., MENZEL, K., AND CARDOSO, T. Deliverable D4.4: ICT support infrastructures and interoperability for VOs. Tech. rep., Uninova, 2003. URL `http://www.uninova.pt/~cam/ev/VosterD44.pdf` [21.5.2004].

[13] DELL, M. *Direct from Dell*. Harper Business, 1995.

[14] FILIPE, J. *The organizational semiotics normative paradigm*. Kluwer Academic Publishing, 2004, pp. 261–272.

[15] GERSNICK, C. Time and transition in work teams: Towards a new model of group development. *Academy of Management Journal 3*, 1 (1988), 9–41.

[16] GULATI, R., KHANNA, T., AND NOHRIA, N. Unilateral commitments and the importance of a process in alliances. *Sloan Management Review 35*, 3 (1994), 61–70.

[17] HANNUS, J. *Prosessijohtaminen*. HMV Research Oy, 1994.

[18] HAWKINS, S. *Ethical and Moral issues facing the virtual organization.* Kluwer Academic Publishing, 2004, pp. 153–160.

[19] HELIN, H. *Supporting Nomadic Agent-based Applications in the FIPA agent architecture*. PhD thesis, University of Helsinki, Department of Computer Science, 2003. URL http://ethesis.helsinki.fi/julkaisut/mat/tieto/vk/helin/supporti.pdf.

[20] HENNART, J.-F. A transaction costs theory of equity joint ventures. *Strategic Management Journal 9* (1988), 361–374.

[21] KATZY, B., AND HORODYSKIV, V. Virtual organization types: Towards a state of the art compendium on European developments, 2002. URL http://www.ve-forum.org/Projects/285/katzy20-%20e2002%20-%20virtual%20organization%20types.pdf [21.5.2004].

[22] KATZY, B., AND LOEH, H. Virtual enterprises research state of the art and ways forward. In *Proceedings of the ICE 2003, Espoo 16-18.6.2003* (2003). URL http://portal.cetim.org/file/1/63/104_Katzy_Loehprint.pdf [21.5.2004].

[23] KATZY, B., LOEH, H., AND ZHANG, C. *Virtual Organizing Scenarios.* Kluwer Academic Publishing, 2004, pp. 27–40.

[24] KHANNA, T., GULATI, R., AND NOHRIA, N. The dynamics of learning alliances: Competition, cooperation and relative scope. *Strategic Management Journal 19*, 3 (1998), 193–210.

[25] KLEYMANN, B. Strategic alliances, joint ventures and mergers in international business. In *Helsinki School of Economics lecture notes* (2004).

[26] KLEYMANN, B., AND SERISTÖ, H. Levels of airline alliance memberships: Balancing risks and benefits. *Journal of Air Transport Management 7*, 5 (2001), 303–310.

[27] KUTVONEN, L. B2B middleware for managing process-aware eCommunities. *Helsinki University Department of Computer Sciences C-Series (to be published)* (2004).

[28] PFEFFER, J., AND SALANCIK, G. *The External Control of Organizations: A resource dependence perspective.* Harper and Row, New York, 1978.

[29] PORTER, M. E. *Competitive Advantage of Nations.* Free Press, 1990.

[30] PORTER, M. E. Clusters and the new economics of competition. *Harvard Business Review* (1998).

[31] PUUSTJÄRVI, J. *Transactional Workflows.* PhD thesis, University of Helsinki, Department of Computer Science, 1999. URL http://ethesis.helsinki.fi/julkaisut/mat/tieto/vk/puustjarvi/transact.pdf.

[32] RHEINGOLD, H. *Smartmobs: The Next Social Revolution.* Perseus Books, 2002.

[33] SHAPIRO, S. *24/7 Innovation: Blueprint for surviving and thriving in the age of change.* McGraw-Hill, 2002.

[34] SIMON, H. A. The axiomatization of physical theories. *Philosophy of the Science 37* (1970), 16–26.

[35] SIMON, H. A. *The Science of the Artificial (3rd edition)*. MIT Press, 1996.

[36] TOFFLER, A. *Future Shock*. Pan, 1971.

[37] TØLLE, M. Reference models supporting enterprise networks and virtual enterprises. *International Journal of Networking and Virtual Organizations 2*, 1 (2003).

[38] VOSTER. VO concepts report. *VOSTER report D14.1* (2003).

[39] VOSTER. VO guidelines. *VOSTER report D54.1* (2004).

[40] WILLIAMSON, O. E. Transaction-cost economics: The governance of contractual relations. *Journal of Law and Economics 22*, 2 (1979), 233–261.

[41] WONG, S.-W., AND WHITMAN, L. Attaining agility at the enterprise level. In *Proceedings of the 4th Annual International Conference on Industrial Engineering Theory, Applications and Practice, Texas, USA, 17-20 1999* (2003).

Chapter 3

Virtual organisation modelling

Sampo Karjalainen

3.1 Background for VO modelling

VO modelling is the key point for creating an IT architecture, which answers to the needs of rapidly changing virtual organisations. A reference model should describe the context where a VO is established, managed and finished. The model should also point out its breeding environment. There are at least two different types of breeding environments: open and closed environments. The open environments rely on partner publishing and discovery, whereas the closed environments are based on pre-contracts between partners. These contracts give the basis for the interoperability.

The idea in VO modelling is to achieve business opportunities with the help of a reference model or models which make it possible to build up an IT system to support dynamic change in the organisation. The dynamic change means a variety of things: introduction of new partners and agreements between existing partners, the possibility to make changes to a partner's internal processes without interfering with the VO's processes, that partners can stop providing a given service, closing one partner out if violation of rules occurs, and a relatively easy infrastructure to close down the VO.

A reference model can also be described as an instruction for a partner to follow when creating, operating and managing a VO or VE. Thease instructions should include models of business processes, semantics of the information in the processes, structures of the data, model of the hardware and software system infrastructure and model for the human resources [10].

The search for a good reference model is still on and for that reason there are no widely accepted standards available yet. To be able to implement a well-founded software system infrastructure for VOs a reference model should be chosen and before that of course developed. There are three major approaches for the foundation of the infrastructure: transaction-oriented, agent-based and service-federation. Because of this background almost every project defines their own reference model, which suits their needs and objectives [5].

This section is focusing on the common concepts of different VO reference models and approaches. First the three different approaches for infrastructure are introduced and some reference projects are described. The second part describes things that a general reference model should cover. The last part compares different approaches and tabulates them against the general reference model.

3.2 Approaches for infrastructures

3.2.1 Transaction-oriented layer-based

The main idea in this approach is to create a new layer on top of an existing IT infrastructure. This enables cooperation between partners and makes it possible for almost anyone to join the VO. The issue in this cooperation layer is that adapting it to almost any modern IT system takes time and money. In other words, it is too expensive in many cases. This approach is more suitable to the closed-environment concept, because the VO's processes are usually executed on one platform and service providers (partners) join this platform.

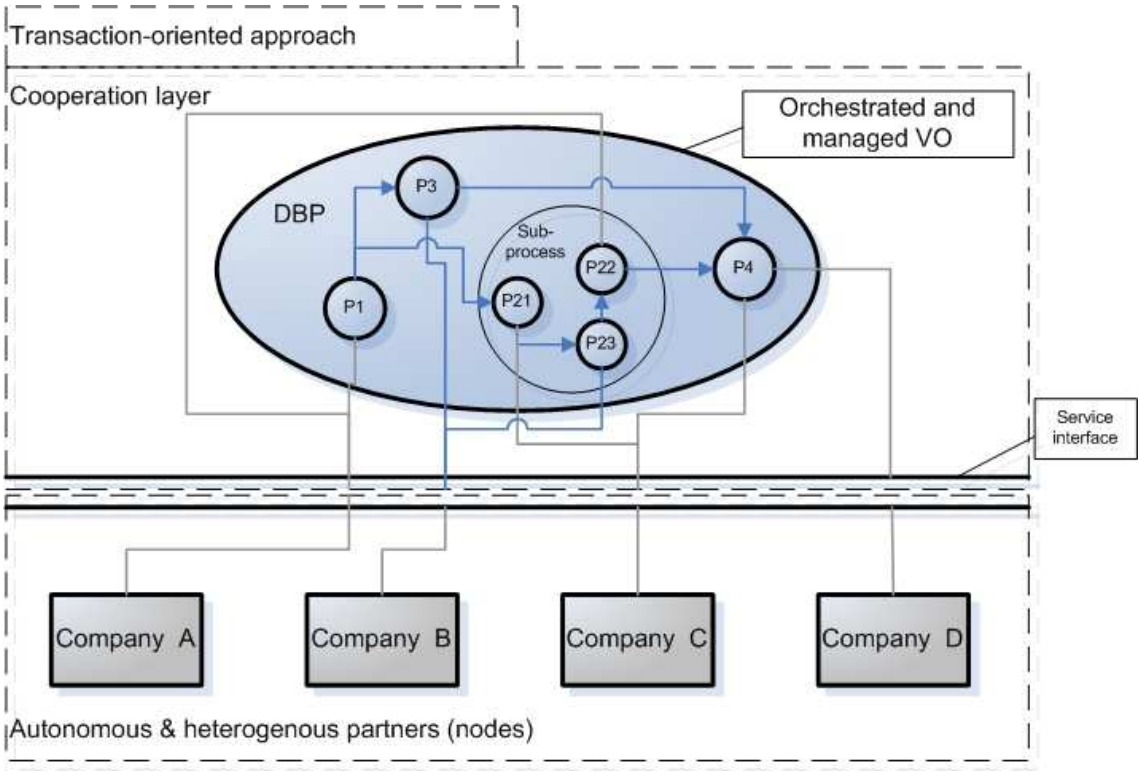


Figure 3.1: Transaction-oriented approach.

In Figure 3.1 is shown the basic model of VO in transaction-oriented approach. There are two ways to analyse this model—top-down and bottom-up. Bottom-up might be more obvious, because then it is easy to understand that the partners are actually just service providers. They have a unified service description method and framework. Partners model their services and build the interfaces (service interface) to their ERP, legacy or other systems. The service interface framework consists of business process modelling tools and general interfaces, which can be tied to the underlying system. In the top-down method the VO's process is modelled first and split into smaller entities like process and enterprise resource. After that the jobs of these entities are carried out by some partners' processes [5].

In both aspects a distributed business process (DBP) is usually built using workflow technic. The DBP is then orchestrated, managed and observed. Orchestration means that workflow's transactions and choices are evaluated between partners. This means that in this approach there exists a centralized process virtual machine or as in PRODNET [4] a coordination kernel. In the PIEC

project networked enterprise applications are build from components modelled with service description language (SDL) [13]. This is quite similar to PRODNET's approach and they both can be seen as examples of a transaction-oriented approach.

Another solution has been introduced by the BPMI.org. They have been designing a Business Process Management System (BPMS). It is based on Web Services and WSDL (*Web Service Description Language*) [6]. When adapting BPMS to the VO case, the partners' WSDL documents are the service descriptions that can be used to build the DBP. The BPML (*Business Process Modelling Language*) [1] is used to describe the VO's business process and it is deployed on the process virtual machine. BPMS is a centralized solution where the DBP is executed on a virtual machine and partners provide their processes available for orchestration. Based on these facts this solution is a example of the transaction-oriented approach [3].

3.2.2 Agent-based

Developing an information system which would be artificially intelligent is of course one of the top targets in the field of computer science. Network agents are often discussed also on the field of AI, because agents can be used for decision making. Looking from these facts it is easy to see them used to build a VO. Agents are networked and are able to execute complicated processes. Because of the intelligent decision making, agents also give support for the VO creation process.

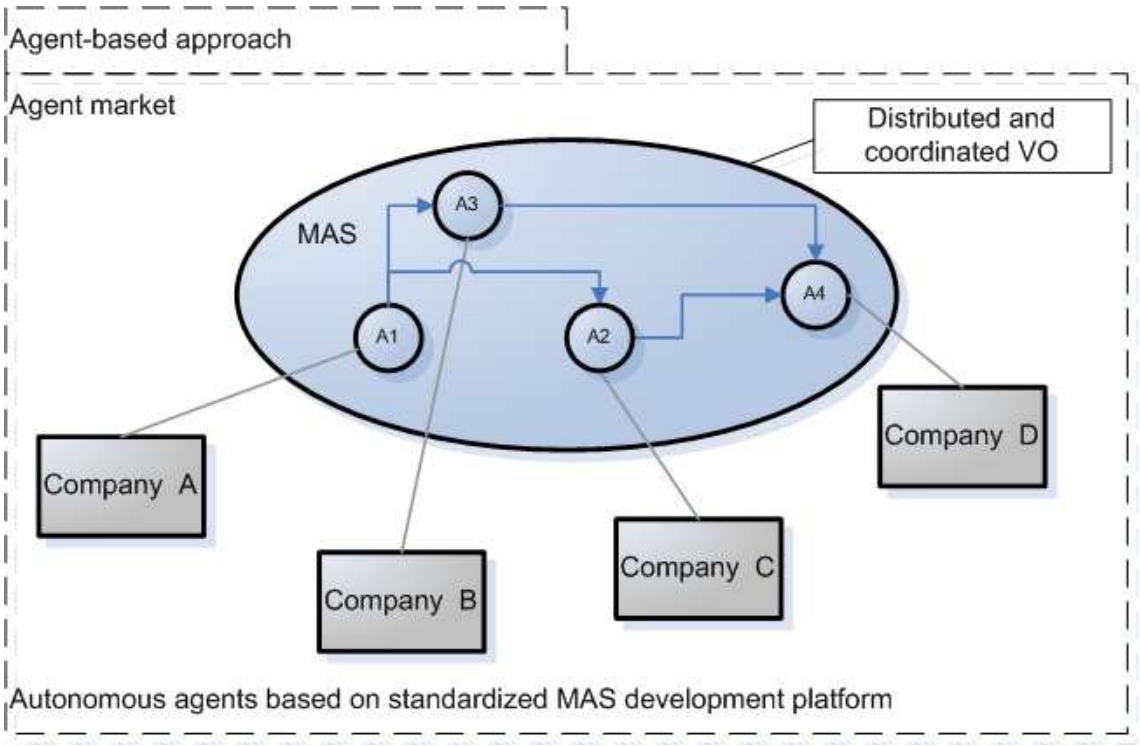


Figure 3.2: Agent-based approach.

The basic architechture looks quite the same as in transaction-oriented approach, but the base is far from similar. As seen in Figure 3.2, the DBP is built out of agents. Each agent supports one subprocess or transaction in the workflow. The interface to the partner's systems is actually developed into the agents and therefore it is not visible in this picture. As long as the agents

are developed according to the same standard, the partners have the possibility to choose their integration tools.

There are lots of aspects in agents supporting VOs. The components of VOs and multi-agent systems (MAS) are distributed, heterogeneous and autonomous. Agents can solve problems, interact with human resources and they can easily be replaced by another agent. To support VOs fully the MAS technology needs to expanded with a better framework for contract management, distributed process management, service quality and good communication channels [5].

One example of this approach is the MASSYVE project. It introduces a federated information management system (FIMS) and multi-agent system integration. FIMS provides a interoperation and exchange channel for the agents. Because the agents are autonomous, the MASSYVE system has strong control on information visibility. MASSYVE provides quite good support for the VO creation process and some support for VO operation. VO evolution is also supported, because agents are changeable [7].

3.2.3 Service-federation

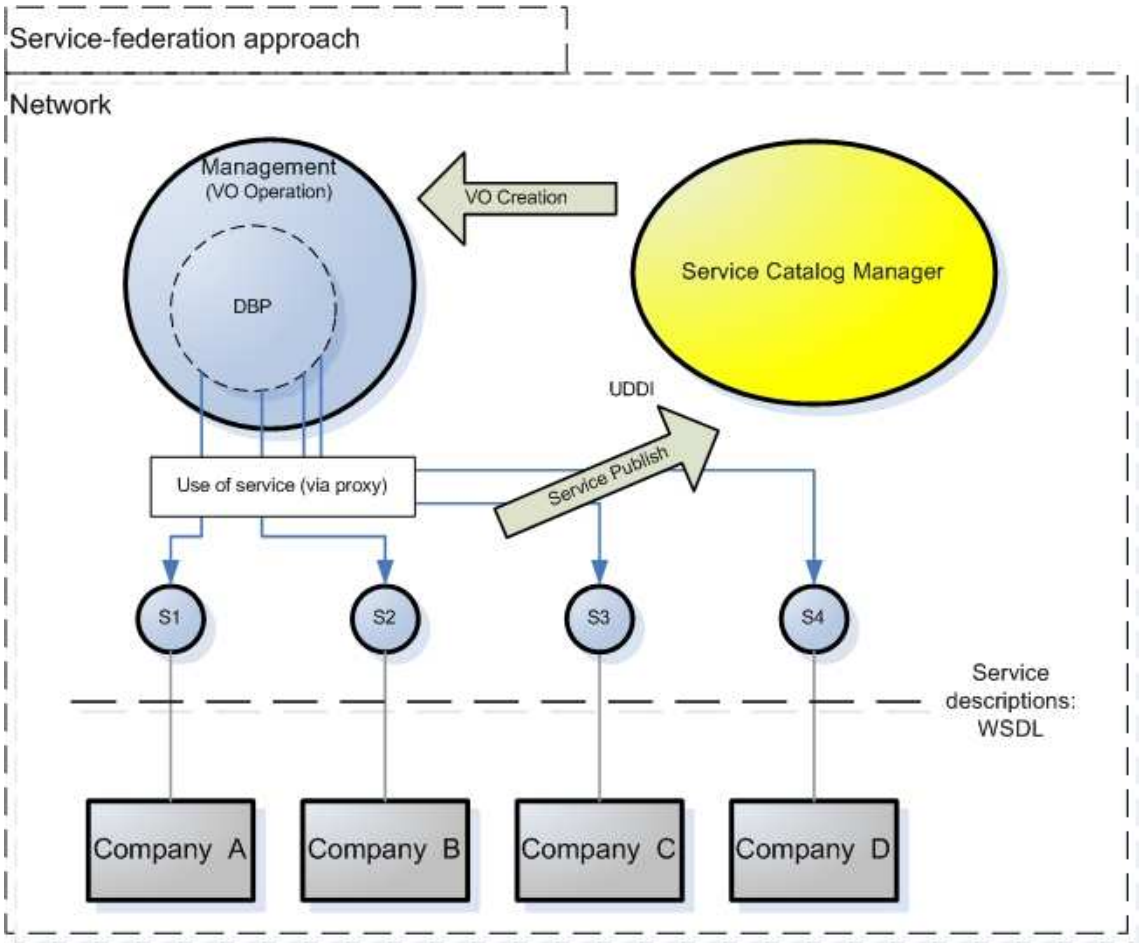


Figure 3.3: Service-federation approach.

This approach has probably the best support for distributed information management. In service-federation the main components are: catalog, services and applications. Services are the

partners' public business processes, which are published to the catalog of the given domain. In the big picture the catalog is in the Internet, but of course this architechture can be used also in restricted networks. Finally are the applications using these services provided by the catalog manager. In the context of VO the application is the environment where the DBP should be created and operated. If the breeding environment wants to control the visibility and usability of services between partners, then the application should be implemented together with the catalog manager.

In the bottom part of Figure 3.3 are the partners of a VO. They have described and implemented their public processes with some language supporting this functionality. At the moment Web Services provide the WSDL as a good solution for this. These services are registrated on the catalog manager with UDDI (Universal Description, Discovery and Integration) [12] or a similar technique. The catalog is distributed with the same kind of way as the Internet name services. The VO application is in the left hand corner of the picture. The application implements the management, distributed business process and other functionalities needed in the operation of a VO.

One example project of this approach is FETISH-ETF. It was developed to support tourism and enable value-added composed packages for the tourists. The catalog manager includes information about different tourism services and tools for accepting new partners, creating and managing business processes and controlling the access rights. The manager can be distributed to different network nodes, which reduces the fragmentation of information and gives a better service quality. The VO is established when a customer looks for a value-added service (VAS). The VAS could be for example a holiday package including hotel, car and theatre tickets [5].

A top-down view to service-federation is presented by ebXML (*Electronic Business XML*). The building parts in ebXML are: repository, messaging service and agreements. Business process specifications are used to describe the services and products of different partners in VO. These specifications are published to the repository, in other words the business library. The DBP are presented as choreographies which define the order and transitions between different business processes. The contract management is also part of ebXML. It provides a CPA (*Collaboration Protocol Agreement*), which describes the conditions of business between two partners [8].

The Web Service Choreography Interface (WSCI) provides also some of the service- federation components. It is not a business process modelling tool, but it enables dynamic composition of multiple services built with WSDL. The main idea is that all the partners involved in a shared business activity have a copy of the same public interface description document and therefore they are aware of each others' activities in the DBP. This technique has no support for VO creation, but it could handle pretty well the VO operation, if it had a contract and communication layer. The strongest part of this technic is that it has a good support for distributed processes [2].

Camarinha-Matos writes that one instance of service-federation is a portal where services are brought available in a centralized form. The VO is formed between these different resources that the web site provides. Portals can also represent the information of two or more resources as it came from one virtual enterprise. Complex interorganizational activities can be implemented behind the scenes of the portal [5].

3.3 Reference model concepts

3.3.1 Reference model

For VOs a reference model is needed to make the creation process re-usable and configurable and therefore quick, inexpensive and secure. The general idea of reference models is to collect the common concepts and characteristics of several entities on the same domain. In this case the

domain is virtual organizations. With the help of rerence models it is possible to save the previous knowledge into model libraries. Once these exist, the modelling of a new VO doesn't need to begin from the scratch, instead it can re-use the existing information and become a more efficient process.

The ARchitecture for Information Systems (ARIS) framework (reference model) has been a critical success factor for many SAP system designers because of the availability of a vast set of reference models for common business functions in many industries [10].

3.3.2 Life cycle of a VO

In the domain of VOs the life cycle means four different phases: creation, operation, evolution and dissolution. The connections between these steps are presented in Figure 3.4. In short, the life cycle demonstrates the environment, where VOs are bred and managed.

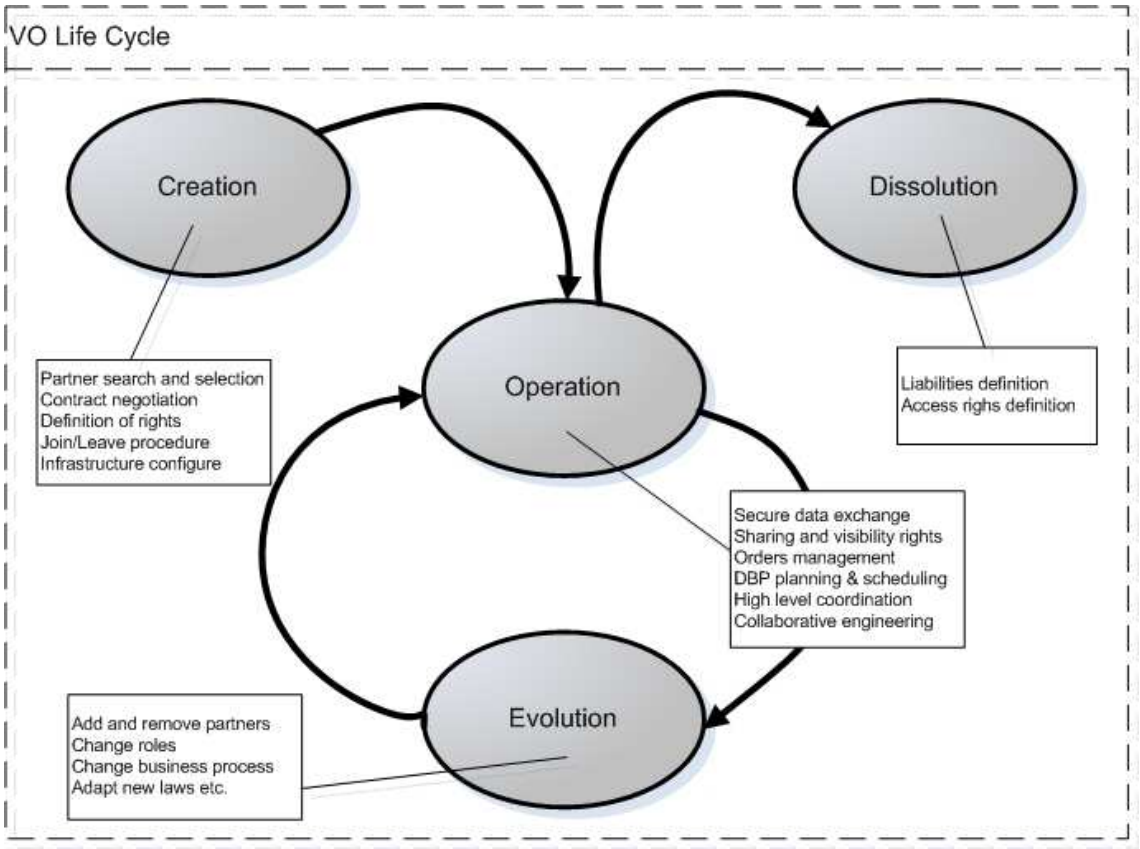


Figure 3.4: VO life cycle support [5].

At the moment the VO creation process has been seen as important phase and support towards dynamic establishment is increasing. The dissolution phase is still quite unpopular. The dynamic environments are also giving more and more support to evolution and operation [5]. The different phases are described in more detail below.

3.3.3 VO creation

To be able to rapidly create a VO to support changing business scenarions the partners in the breeding environment have to have a high degree of prepardness. Those companies or other organizations who are prepared for VO creation have knowledge about enterprise engineering including a set of reference models that have been done earlier. The existence of these models makes it possible to create or re-create VO/VE fast. One important aspect which has to be in condition is the ICT environment. Interfaces used in external business processes should be tested and verified. When this is true, it is possible to achieve operational IT systems based on models created for the VO.

VO creation should be a configuration process which implements an already known business process and is based on a partner's existing reference models. The creation process should not include any new software design and implementation. This approach supports dynamic, efficient and fast VO creation and re-creation.

This kind of preparedness involves a set of reference models about the business network, VO creation and operation. In a closed environment it also includes the pre-agreement-phase for setting up the foundational rules of the network enterprise. In the more open and dynamic scenario partners prepare themselves first and then publish their open services to a catalog system. In this case the catalog system is the network enterprise. The power between partners in a networked enterprise are often unequally divided. This is a key point affecting to the VO creation process. It is also clear that companies do not wish to share their core competencies in a network, but only those that are strategically good to share [10].

Several projects are concentrating on VO creation functions, but it doesn't mean that they were designing real automated solutions on this area. In some cases VO creation means human interaction. The functions that are addressed are: VO planning, partners search and selection, enterprise catalogues, contract negotiation, etc. Selection is probably the most popular area, but the agent-based solutions are almost the only ones who have really provided some computer-assisted means for search functionalities. The lack of directories is an obstacle on VO creation research. These directories should provide information about companies and organizations, and their profiles including skills, resources, services and processes. To increase the amount of usable directories, standards for representing the profiles need to be developed [5].

3.3.4 VO operation

Once the VO has been created begins the operational phase. This phase includes of course many important tasks concerning basic operations: executing the decided distributed business processes and managing the system. Another as important aspect is to operate so that the knowledge and experience gained during the operation is saved for future use. It is the network which should gain the information from the operation of the VO, because it might be needed again when a similar business case occurs and a virtual organization is needed to support it [10].

The coordination and management of DBPs in a VO is one of the main functions in operation phase. The WfMC (*The Workflow Management Coalition*) [9] reference architechture has been the base for many VO operation models. The original version has been extended to support cross-organizational workflow supervision. This has included some preliminary work on role-based management, multi-level coordination and exception handling.

The work on graphical expression and a design language for distributed business processes has just been suggested and therefore not too many examples exists. This is one of the key points for effective business process sharing and re-use. There is an initiative called UEML (*Unified Enterprise Modelling Language*) [11], which is trying to harmonize the modelling area [5].

3.3.5 VO evolution and dissolution

The evolution in VO means that the DBP and the infrastructure executing it has to support dynamic changes in the relationships between partners and services. New partners may join the organization and others leave. It also has to be possible for existing partners to change their public processes. Most importantly it has to be possible to improve the business processes of the VO.

The dissolution of a VO has not been noted in many projects or at least it has not gained any real solutions yet. Though this area is still quite new in the VO research area it is important, because the operation of a VO has to be legal. In many countries there are strict laws which precisely describe how long product and service history has to be maintained. This is the reason why the reference models should also concentrate on the steps taken, when the VO or VE is dissolved [5].

3.4 Conclusions

In this report I tried to find out the concepts which should be addressed when a reference model for virtual organization is designed. These concepts are mainly based on the referenced authors' articles and their descriptions about the life cycle of a VO. In Figure 3.5 I have presented the different phases and aspects involved in VO's life cycle and marked with the 'x' those approaches that support a particular property.

	Approach for reference infrastructure		
	transaction-oriented	agent-based	service-federation
VO's life cycle			
Creation			
preparedness			
IT System (infrastructure, interfaces)	x	x	x
Busines Processes (internal / external)	x	x	x
Possesion of reference models	x	x	x
Management	x	x	x
automatic creation		x	x
Operation			
executing DBP	x	x	x
managing and controlling the operation	x	x	x
saving the operational knowledge	x		x
graphical modelling	x		x
Evolution			
process improvement	x	x	x
dynamic partner management	x		x
dynamic service connectors	x		x
Dissolution			
operational data history			x
other legal aspect			
technical aspects	x	x	x

Figure 3.5: VO life cycle support in different approaches.

The IT system means that the partner has the possibility to implement its ICT infrastructure before the actual VO creation phase. Business process preparedness stands for the possibility to define a partner's internal and external processes separately. If the partners can make reference models for different business scenarions in advance, then the possesion of reference models is supported. The last aspect tells that the partner is prepared for DBP management tasks. This involves

human resources with enough skills, IT systems and communication channels. Preparedness is well addressed in all three approaches, but there are differencies in the level of technical readiness. The automatic creation describes how well the approach supports dynamic and computer-assisted partner search and negotiation. The agent-based and service-federation have addressed this functionality.

In the operation phase I have found four main aspects: executing DBP, managing and controlling the operation, saving operational knowledge and graphical modelling. The two first ones are basicly the workflow management and dynamic adjustability properties. These are quite well supported by all approaches. If saving the operational knowledge is possible in the VO, it means that the network around the VO can re-use old processes and collected information. None of the three approaches had total support for this aspect, but the agent-based doesn't support it at the moment. The agent-based approach was still lacking the software for graphical modelling.

The evolution's aspects were pretty well handled by all three architechtures. The use of proxies in service-federation and a possibility to build good service-wrappers in the transaction-oriented approach give a better chance to be succesful in connecting new service (partners) to VO.

In service-federation approach we have the catalogs (repositories) to collect the data and preserve historical information about distributed business processes and partners after the VO has been dissolved. The legal aspects aren't well supported in any of the architechtures. The technical problems are usually addressed in the different VO approaches.

To be able to build a good reference model, I think, we need more example projects. From these projects we can learn, what is important and what's not. This is hard to see otherwise. Because the VO modelling area has still many even technically demanding parts, it will still take some time before good reference models can be developed. In the meanwhile reference models will emerge, but I think it will still take some time for them to be accurate enough.

Bibliography

[1] ARKIN, A. *Business Process Modeling Language*. BPMI.org, Jan. 2003. URL: http://www.bpmi.org/specifications.htm [27.1.2005].

[2] ARKIN, A., ET AL. *Web Services Choreography Interface (WSCI) 1.0*, Aug. 2002. URL: http://www.w3.org/TR/wsci [27.1.2005].

[3] *Business Process Management Initiative*, Dec. 2004. URL: http://www.bpmi.org/aboutus.htm [27.1.2005].

[4] CAMARINHA-MATOS, L. Execution system for distributed business processes in a virtual enterprise. In *Future Generation Computer Systems* (2001), Elsevier Science B.V., pp. 1009–1021.

[5] CAMARINHA-MATOS, L. Infrastructures for virtual organizations - where we are. In *Proceedings of ETFA 03 - 9th international conference on Emerging Technologies and Factory Automation* (2003), pp. 405–414.

[6] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. *Web Services Description Language (WSDL) 1.1*, Mar. 1999. URL: http://www.w3.org/TR/wsdl.html [27.1.2005].

[7] The MASSYVE project homepage, 2004. URL: http://www.gsigma-grucon.ufsc.br/massyve/ [27.1.2005].

[8] MEDJAHED, B., BENATALLAH, B., BOUGUETTAYA, A., NGU, A. H. H., AND ELMAGARMID, A. K. Business-to-business interactions: issues and enabling technologies. *The VLDB Journal 12* (2003), 59–85.

[9] *The Workflow Management Coalition*, Dec. 2004. URL: http://www.wfmc.org/ [27.1.2005].

[10] TOLLE, M., AND BERNUS, P. Reference models supporting enterprise networks and virtual enterprises. *International Journal of Networking and Virtual Organizations 2* (2003), 2–15.

[11] *Unified Enterprise Modelling Language - initiative*, May 2003. URL: http://www.ueml.org/ [27.1.2005].

[12] *Universal Description, Discovery and Integration*, July 2002. URL: http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm [27.1.2005].

[13] YANG, J., VAN DEN HEUVEL, W., AND PAPAZOGLOU, M. Service deployment for virtual enterprises. *Australian Computer Science Communications 23* (Jan. 2001), 107–115.

Chapter 4

Semantic interoperability

Teemu Virtanen

4.1 Semantic interoperability

This chapter is about different roles of semantics in the virtual organization breeding environment (VBE). The emphasis is on the significance of semantics in the virtual organization interoperability point of view.

Semantic interoperability is the modelling of conceptual integration requirements to enable automated interpretation of those in the information system level. Semantics are a key instrument in automation and the need for automation is central in VO breeding environments.

Semantic interoperability also means that participating information systems in VBE share the same conceptualization of used services and transported data [11].

Semantic interoperability in VBE related research is still in initial state. However, it has been recognized to be a central issue in virtual organization/enterprise (VO/VE) domain [5]. The major goal of research on use of semantics in VBEs is to make possible the automation of many tasks in VO processes.

There are three approaches to defining required IT infrastructures for virtual enterprises [6] that classify also the related semantic approach:

1. Transaction-oriented
2. Agent-based
3. Service-federation approach

The first one focuses on static VOs where a VO contract is usually present. However, the focus in this chapter is on the third one: service-federation approach, which includes the assumption of an "open universe" or "open service market" concept, where breeding environment is populated when some enterprise decides to begin a VO formulation. As obvious, the partner discovery phase is central in this approach and the major focus of semantics is also related to that stage.

The concepts of metadata, semantic metadata and ontology are described as a basis for this chapter. After that the central technologies present in VBE semantics research are given an overview.

The approach to semantics discussed in the end of this chapter is on defining the semantics related to different stages of a Semantic Web process life cycle. The Semantic Web process is a close concept to VO process that was introduced in the preceding chapter. The words Semantic

and Web refer to the taken approach on the issue. The concept of VO is observed on top of that approach in this chapter. The stages of the life cycle are: development, annotation, discovery, composition and execution of service [2]. Firstly are semantics described in development and annotation of a service. The other phases of the life cycle follow in chronological order from the point of view of the service requester.

The semantic interoperability related reference models have not yet reached large scale practical application level. That is why the concepts of the domain are discussed here mainly through theoretical study.

4.2 Metadata, semantics and ontologies

Metadata is a basis for defining semantics. By different levels of metadata, a more specific meaning of the contents of the target is achieved. A concept of ontology is introduced for narrowing the scale that is described by means of semantics. All these three concepts have first been developed for Semantic Web needs but they are applicable to Semantic Web Processes as well [14, 16].

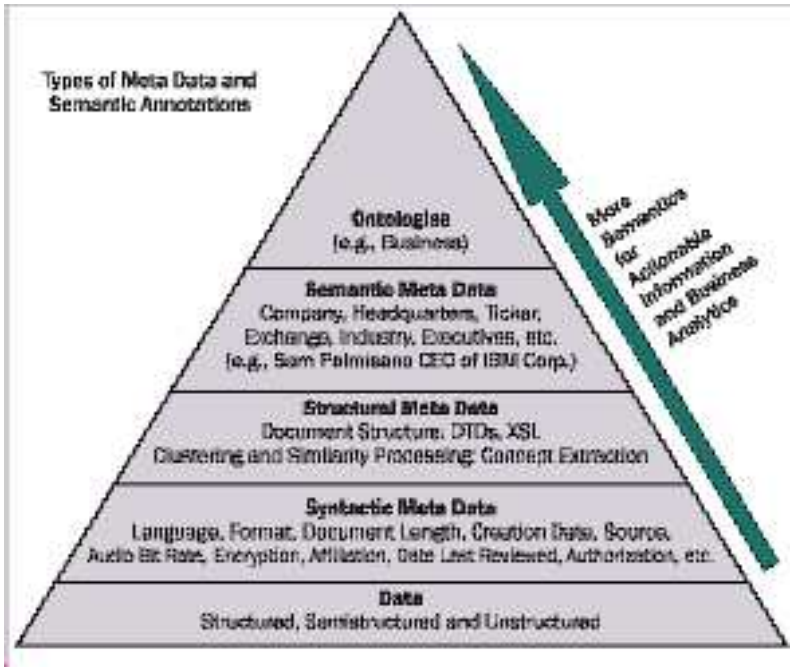


Figure 4.1: Metadata, semantics and ontologies [15].

In Figure 4.1 the different levels of metadata are presented. At the bottom there is the data itself. Syntactic metadata describes details about the data source. This is e.g. the title, language format etc. of the target. Syntactic metadata can be used mainly for cataloguing and categorizing purposes.

Structural metadata above syntactic metadata gives information about the structure of the target. It is meant for facilitating e.g. processing and presentation of the data, as well as information retrieval from the data. In VO systems structural metadata is central in defining the functionality of services [16].

Semantic metadata is the level of metadata that meets automated machine processing demands.

It focuses on describing contextually significant, domain specific information. Semantics have a high-level capability of providing meaning to the underlying syntax and structure. This is the basis for describing the meaning of VO services. Making semantics domain specific happens by means of ontologies [16].

An ontology is a formal, explicit specification of a shared conceptualisation [9]. It defines and presents the concepts and their relations of a certain domain. Ontologies can be considered as substitutes to real world entities. More practically an ontology can also be described as a shared dictionary for actors of some industry. For example travel industry can specify a set of shared service descriptions to assist the flexible use of distributed services on that domain.

In Figure 4.2 there is an example where the ontological relations of a flight booking service are presented. In part (a) there is a shared ontology that is common in a flight booking domain. In part (b) is presented how the shared ontology can be used for implementing a more detailed local ontology by some service provider. Those local ontology entities can then be mapped into the shared ontology entities.

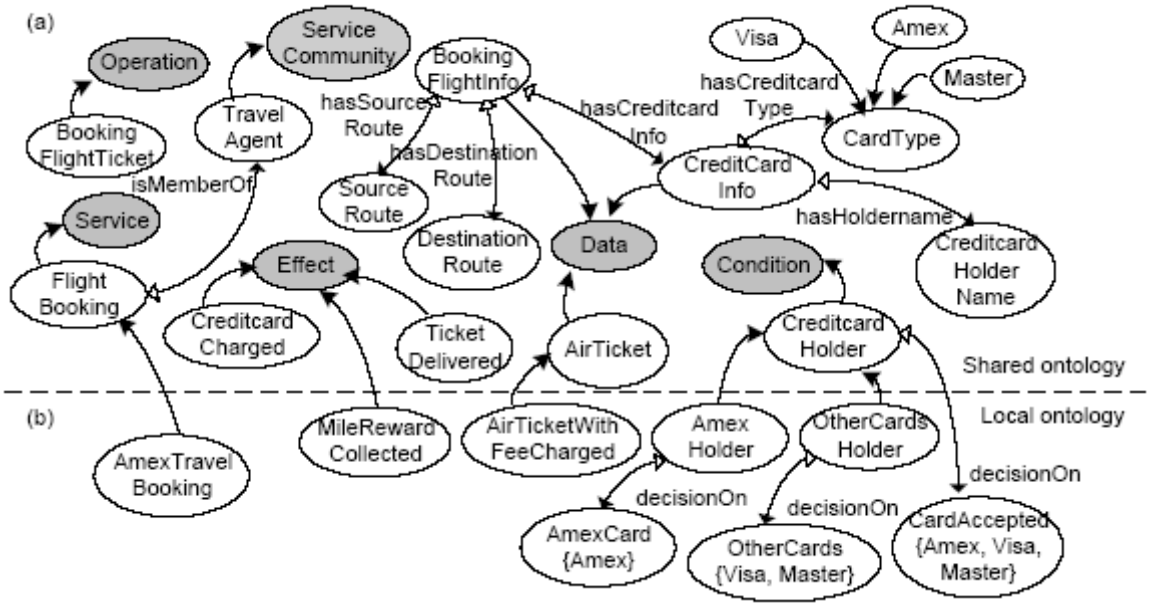


Figure 4.2: Service ontology [18].

Ontologies are central in all phases of Web Process life cycle but they have an special role in automated service discovery [2]. How ontologies can be mapped with actual information systems is described in the following section.

4.3 Central technologies

The central technologies in the VBE semantic interoperability domain are mainly related to defining semantics for partner discovery. Some of the technologies are specialized only on that, while others also cover the rest of the Web process life cycle. Many central technologies are meant for adding semantics to service description languages that are otherwise limited to functional description level. Most of the developed technologies are studied via a reference implementation model or a framework that has its main focus on some of the Web Process life cycle phases.

Service oriented architecture (SOA) is central in service-federation infrastructure, which is the main focus of this chapter. The most of the current study approaches the individual service integration challenge by using Web services technology [4] for system integration. Web services is seen as a solution to emerging requirements of flexible system integration in independent decentralized (also geographically) virtual organization environments. This is especially the case in "open service market" domain, which was described in the introduction section.

To enable the easy use of Web-based services, their classification and discovery have to be supported by means of semantic descriptions. This chapter follows the common trend and has its main focus on Web services -approach and the technologies that support adding semantic dimensions to otherwise syntactic service descriptions. WSDL [8] is used for describing the structure of a Web service. However, it is limited in defining only the structure. That is why the semantic extension of WSDL is studied in many reference implementations.

One proposed solution uses standard extensions to WSDL to map the service with related ontologies [18]. This is presented as an example in Figure 4.3. WSDL elements, like: wsdl:message, wsdl:operation, wsdl:input and wsdl:output which are the behaviour of the service are mapped to shared and local ontologies in Figure 4.2. The preconditions and effects as well as the conditional outputs are specified by means of ontological map. For example, in Figure 4.3 the operation BookingTicket gets flight information as input and it has a precondition that the consumer has to have a credit card. As a conditional output there is a flight ticket and possibly a service charge.

The approach used in Figure 4.3 uses DAML+OIL [3] language based ontology. DAML+OIL ontology called DAML-S [3] is a common service-ontology mapping technology in the reference models. It includes three modules: a Service Profile that describes the Web service and its capabilities, a Process Model that describes what the Web service does and how to interact with it, and a Service Grounding that maps the information exchanges described in the Process Model into actual messages between Web services [3].

DAML-S descriptions as well as other mapping techniques can be used for advertising Web services as well as for describing wanted Web service capabilities in service requests. The example uses extended WSDL as an approach but also WSDL documents with separated ontological descriptions are studied [17].

Technologies that connect WSDL and ontologies have been introduced as tools for defining service behaviour. To take advantage of the enhanced service descriptions, the discovery process of services has to be improved.

UDDI is an Internet-wide registry for advertising and searching Web services. It is also expected to become a de facto standard of the domain because of its strong industrial support [12]. However, standard UDDI only supports service discovery by non-semantic search criteria. Some research projects have studied the expansion of UDDI registry with an interface that makes possible the search of services with semantic search criteria based on ontologies described i.e. with the introduced DAML-S language [13].

As the whole field of semantic interoperability research, also the most of the technologies introduced here are still in developing state.

4.4 Semantic interoperability in VO life cycle

A VO Process is defined here as a set of actions performed in VBE. This process includes the creation of VO and its operation phase. The focus in this section is on the formulation phase but it also partly covers the operational phase of a dynamic, already functional VBE. The operational phase issues are described more specifically in the Monitoring chapter of this writing.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
<!ENTITY sh "http://www.wssemantic.com/flightbooking.daml#">
<!ENTITY lo "http://www.AmexTravel.com/amextravel.daml#"> ]>
<definitions
   name="AmexTravel"
   targetNamespace="http://www.AmexTravel.com/amextravel.wsdl"
   xmlns:tns="http://www.AmexTravel.com/amextravel.wsdl"
   xmlns:WSDLext="http://www.wssemantic.com/WSDLext.dtd"
   xmlns:sh="&sh;"
   xmlns:lo="&lo;"  ...>
...
 <portType name="AmexTravelPortType">
  <operation name="BookingTicket"
    WSDLext:semantic-operation="&sh;BookingFlightTicket">
   <documentation>Provide service for booking air ticket
       </documentation>
   <input message="tns:BookingTicketRequest"
      WSDLext:semantic-data="&sh;BookingFlightInfo">
   </input>
   <WSDLext:precondition
       WSDLext:semantic-condition="&sh;CreditcardHolder"/>
   <output message="tns:BookingTicketResponse" />
   <WSDLext:conditionalOutput
       WSDLext:semantic-data="&sh;AirTicket">
    <WSDLext:condition
       WSDLext:semantic-condition="&lo;AmexHolder">
      <WSDLext:conditionalEffect
        WSDLext:semantic-effect="&lo;MileRewardCollected"/>
      </WSDLext:condition>
   </WSDLext:conditionalOutput>
   <WSDLext:conditionalOutput
       WSDLext:data="&lo;AirTicketWithFeeCharged">
    <WSDLext:condition
       WSDLext:semantic-condition="&lo;OtherCardsHolder"/>
   </WSDLext:conditionalOutput>
   <WSDLext:effect WSDLext:semantic-effect="&sh;TicketDelivered"/>
  <WSDLext:effect WSDLext:semantic-effect="&sh;CreditcardCharged"/>
   </operation>
  </portType>
   <binding name="AmexTravelBinding" type="tns:AmexTravelPortType">
    ...
  <service name="AmexTravel"
        WSDLext:semantic-service="&lo;AmexTravelBooking "/>
      <WSDLext:servicecommunity
        WSDLext:semantic-community="&sh;TravelAgent"/>
      <port binding="tns:AmexTravelBinding" name="AmexTravelPort">
       <soap:address
         location="http://www.AmexHolder.com/BookingAmexTicket"/>
      </port>
  </service>
</definitions>
```

Figure 4.3: Extending WSDL with ontological information [18].

The concept of Web process is used here as a reference to VO process, to explain the actions that are characteristic to both of them. As a technical approach Web services is chosen here due to its widespread popularity and flexible suitability for VBE IT system communication.

A Web Process life cycle can be used for defining clear borders between different types of semantics present in each stage of the life cycle [16, 7, 1]. This classification is used by e.g. METEOR-S project [2] among others. The life cycle is approached from both ends of the service delivery chain: from the view of the provider of the service, and from the view of the requester of the service. The phases of a Web process life cycle are as follows:

Providers' point of view:

1. Service creation
2. Annotation and publication of service

Requesters' point of view:

3. Discovery of service
4. Composition of process
5. Execution of process

4.4.1 Service creation

When creating a service, one can implement it from the basis of a VO contract where the service to be created is meant for satisfying some specific business need whose content is well known beforehand. In an "open universe" approach the use purpose of the service to be created is usually not that specific. In the "open service market" approach, the service is created and advertised, rather than implemented on-demand as in the other approach.

Practically any object oriented programming language can be used for implementation of a service because they can communicate with other entities using SOAP [10]. This means that that kind of programs can be easily translated into Web services, which is one way to satisfy the needs related to technical semantics.

Three different approaches to defining semantics in Web services [4] are compared below in the sense of how they meet the semantic interoperability needs of VBE.

1. Semantics can be offered by the provider.

This is usually not the way how things are done in VBEs. When things are done this way, as a result there are a lot of heterogeneous services that vary by their behaviour. That is why they are costly to use in dynamic environments.

2. The requester can ask the provider to offer certain kind of functionality.

The functional interoperability might be achieved this way but this is not enough to cover the other semantic interoperability needs.

3. A service can be specified by an industry standard body, when both the requester and the provider implement the shared conception of semantics.

This is a common approach in VBEs. The industry standard body is equivalent to ontology. This is how both the provider and the user of a service can interpret the service in a shared domain.

4.4.2 Annotation and publication of service

The process of adding semantic descriptions to Web service is called annotation. Publication means that the created service is published into some public or private registry so that others (VO partners) can find it and use it.

In this stage one has to decide the way of adding semantic descriptions to a created service. The semantics involved in this stage are called data semantics [1]. They are meant for Web services to communicate with each other. This means that the services need to understand each others' data, which are inputs, outputs and exceptions [2]. That kind of behaviour is described by means of ontologies that have been introduced earlier in this chapter.

Annotation can already be semi-automated by e.g. means of writing standard comments into application's source code [16]. Those comments function as input to the description generator.

When a provider service has been created, it has to be published to make possible to others discovery and access of that service. Publication is making the created service accessible to a restricted group of players (VO participants) as in closed VBE. In the "open service market" approach the service is annotated and published to some public search engine, such as UDDI with enhanced semantic capabilities.

4.4.3 Discovery of service

Discovery of service is a central and relatively much researched phase in the "open service market" approach. The semantics that are present in that stage are called functional semantics. Those are a combination of the service's data semantics and classification of its operations' functionality [16]. Data semantics also cover the preconditions and postconditions of the service and they can be described by means of ontologies.

Services are searched by given data-, functional- and QoS parameters, that is its data semantics and QoS semantics. QoS semantics are described more accurately in the following subsection.

The formal presentation of functional semantics is described in Table 4.1. S is a service and o is one of its operations.

Fc(s,o)	Functional classification of operation 'o' in terms of ontological concepts
$I(s, o)$	Inputs of operation 'o' in terms of ontology concepts
$O(s, o)$	Outputs of operation 'o' in terms of ontology concepts
$E(s, o)$	Exceptions throwable during execution of operation 'o' in terms of ontology concepts
$P(s, o)$	Pre-conditions of operation 'o' in terms of ontological concepts
Po(s,o)	Post-conditions of operation 'o' in terms of ontological concepts

Table 4.1: Functional semantics [2].

In Figure 4.4 the Web service discovery phase and semantics that are related to it are presented. Data semantics are presented in form of Web Service Description (WSD) that describes the inputs, outputs and data types of the service. Functional description (FD) presents a more specific explanation of the meaning of the service and is considered the related functional ontology that is mapped to the WSD.

The process begins when the provider publishes his service (1.a.). In the second phase the requester uses a discovery service to find a provider service that applies to his search criteria, which is a combination of Data - and possibly some QoS semantics described by means of QoS ontology (1.b.). The requester receives a WSD of a service candidate (1.c.) that matches his search criteria. The second phase of semantic interoperability inspection belongs to Web Process composition phase (2.). After making sure that both parties share the same conception of semantics, the system interaction may begin (3. and 4.).

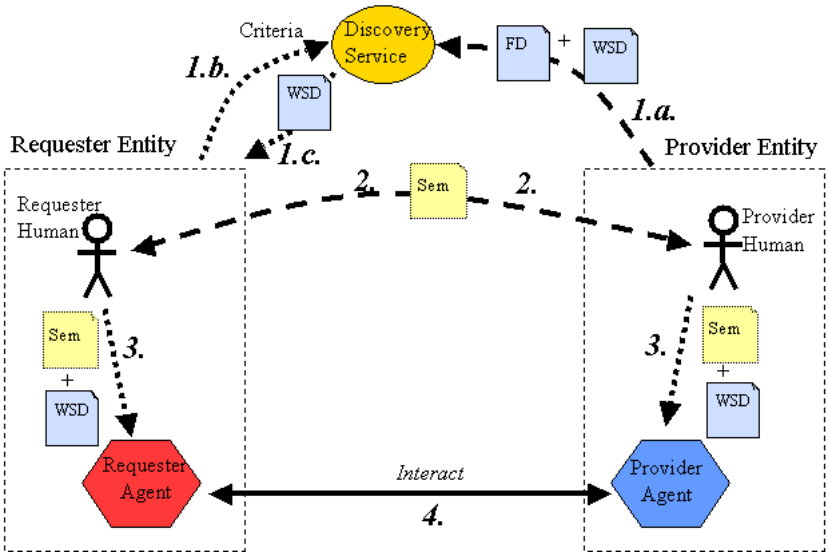


Figure 4.4: Web Service discovery process and related semantics [4].

Automated discovery of Web services meets the concept of automated partner discovery in "open universe" VBEs. Automated discovery is based on the automated interpretation of semantics. This is how the relevant services can be found to be offered to the direction that makes the final selection of the service to be used to satisfy the business need.

4.4.4 Composition of process

Composition of Web Process means combining a set of Web services to fulfill some certain business task. In the "open service market" approach the composition phase follows the discovery phase where a set of service candidates have been listed by means of using semantic properties as search criteria. However composition is also naturally present in the private virtual breeding environments with its own characteristics.

In composition phase the candidates' properties are examined by using Quality of Service (QoS) semantics. Those are already present in the discovery process but their significance becomes more obvious in the composition phase.

QoS semantics characterize performance as well as other qualitative and quantitative aspects of services. Once again ontologies are used for explicating QoS semantics. QoS semantics can be domain-independent, which means that they describe some universal value, like the duration of the execution of a service. They can also be domain-specific, like e.g. a star rating of a hotel in travelling domain. The formalization of QoS semantics is presented in Table 4.2.

$T(s,o)$	Execution time of Web service 's' when operation 'o' is invoked
$C(s,o)$	Cost of Web service 's' when operation 'o' is invoked
$R(s,o)$	Reliability of Web service 's' when oper 'o' is invoked
$A(s,o)$	Availability of Web service 's' when oper 'o' is invoked
DSi(s,o)	Service/operation level domain specific QoS metrics

Table 4.2: QoS semantics [2].

It is natural to expect some degree of mismatch between desired and found service properties [12]. Matching functions for search engines which allow a tolerable amount of mismatch need to be developed. The mismatch should be in line with the overall QoS limits of the process that is being composed.

QoS semantics are central in the process composition phase but other described semantics are also examined to confirm semantic interoperability.

4.4.5 Execution of process

Execution semantics together with QoS semantics make possible the simulation and verification of functionality and properties of a Web process that is to be executed. This happens by using an execution pattern that describes the overall structure of the process [2].

In execution phase a service can be replaced with another if it does not provide the expected QoS. Replacement may also occur if a VO participant that offers the service breaches the VO contract. These concepts are related to VO monitoring, which is explained in the Monitoring chapter of this writing.

4.5 Conclusions

Virtual organizations are used for making cooperation of business partners more flexible and semantic interoperability research aims to reducing the effort needed to construct information system infrastructures for their use. This is what semantic interoperability research tries to make more efficient, the integration process of heterogeneous and autonomous information systems. This happens by means of increasing the amount of automation in different phases of VO life cycle.

Web Services is a common technology in VO environments. Standard Web Services satisfy many of the issues that are related to semantic interoperability in virtual organization breeding en-

vironments. Web Services technology can also be extended to satisfy the semantic interoperability needs that cannot be satisfied with standard Web Services.

The concept of ontology is central in semantic interoperability field. The role of ontologies as well as the different types of semantics were observed in this chapter in the different phases of the VO life cycle. Also the VO life cycle concept itself seems to be central in semantic interoperability research.

The semantic interoperability field is still relatively little studied. That is why it is likely, that significant level of human participation will still be needed for some time in VO information system integration. In addition to that, there exists a variety of technologies that are meant for guaranteeing the semantical interoperation. Application of those technologies is diverse as well. It can be concluded that standardization has to progress before results of the semantic interoperability research can be efficiently used in practice.

Bibliography

[1] ABHIJIT, P., OUNDHAKAR, S., SHETH, A., AND VERMA, K. Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web* (2004).

[2] AGGARWAL, R., VERMA, K., MILLER, J., AND MILNOR, W. Constraint driven web service composition in METEOR-S. In *Proceedings of the Services Computing, 2004 IEEE International Conference on (SCC'04) - Volume 00* (2004).

[3] ANKOLEKAR, A., ET AL. DAML-S: Web Service description for the Semantic Web. In *Proc. 1st Int'l Semantic Web Conf. (ISWC 02)* (2002). URL `http://citeseer.ist.psu.edu/ankolekar02damls.html`.

[4] BOOTH, D., HAAS, H., MCCABE, F., NEWCOMER, E., M., C., FERRIS, C., AND ORCHARD, D. *Web Services Architecture*, Feb. 2004. URL `http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/` [27.1.2005].

[5] CAMARINHA-MATOS, L. Infrastructures for virtual organizations - where we are. In *Proceedings of ETFA'03 - 9th Int. Conf. on Emerging Technologies and Factory Automation* (2003).

[6] CAMARINHA-MATOS, L., MENZEL, K., AND T., C. Deliverable D4.4: ICT support infrastructures and interoperability for VOs. Tech. rep., Uninova, Mar. 2003.

[7] CARDOSO, J., AND SHETH, A. Introduction to Semantic Web services and web process composition. In *Semantic Web Process: powering next generation of processes with Semantics and Web Services, Lecture Notes in Computer Science*. Springer, 2005. To be published in print 2005.

[8] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. *Web Services Description Language (WSDL) 1.1*, Mar. 1999. URL `http://www.w3.org/TR/wsdl.html` [27.1.2005].

[9] GRUBER, T. Towards principles for the design of ontologies used for knowledge sharing. In *Formal Ontology in Conceptual Analysis and Knowledge Representation*, G. N. and R. Poli, Eds. Kluwer Academic Publishers, Deventer, The Netherlands, 1993. URL `http://citeseer.ist.psu.edu/gruber93toward.html`.

[10] GUDGIN, M., HADLEY, M., MENDELSOHN, N., MOREAU, J., AND NIELSEN, H. *SOAP Version 1.2*, June 2003. URL `http://www.w3.org/TR/soap12-part1/` [27.1.2005].

[11] KUTVONEN, L. B2B middleware for managing process-aware eCommunities. *Helsinki University Department of Computer Sciences C-Series (to be published)* (2004).

[12] PAOLUCCI, M., SYCARA, K., NISHIMURA, T., AND SRINIVASAN, N. Toward a Semantic Web e-commerce. In *Proceedings of BIS* (2003).

[13] PAOLUCCI, M., T., K., PAYNE, T., AND K., S. Semantic matching of Web Services capabilities. In *First Int. Semantic Web Conf.* (2002). URL `http://citeseer.ist.psu.edu/paolucci02semantic.html`.

[14] SHETH, A. Changing focus on interoperability in information systems: From system, syntax, structure to semantics. In *Interoperating Geographic Information Systems*. Kluwer Academic Publisher, 1999, pp. 5–29.

[15] SHETH, A. Semantic meta data for enterprise information integration. *DM Review Magazine* (jul 2003).

[16] SIVASHANMUGAM, K., SHETH, A., MILLER, J., VERMA, K., AGGARWAL, R., AND RAJASEKARAN, P. Metadata and semantics for web services and processes. In *Datenbanken und Informationssysteme*. Publication Hagen, 2003. URL `http://lsdis.cs.uga.edu/library/download/Schlageter-book-chapter-final.pdf`.

[17] SIVASHANMUGAM, K., VERMA, K., SHETH, A., AND MILLER, J. Adding semantics to web services standards. In *In Proceedings of the 1st International Conference on Web Services (ICWS'03)* (2003).

[18] SRIHAREE, N., AND SENIVONGSE, T. Discovering Web Services using behavioural constraints and ontology. In *4th Int. Conference on Distributed Applications and Interoperable Systems* (nov 2003), J.-B. Stefani, I. Demeure, and D. Hagimont, Eds., Lecture Notes in Computer Science, IFIP TC6, Springer-Verlab.

Chapter 5

Collaborative workflows

Mikko Hämäläinen

Virtual organizations allow the creation of enterprises that bring together organizations or individuals from all around the world. On the other hand, they support the leveraging of local competence by bringing local service providers together to compete against global businesses.

The thing that is common to most virtual organizations, global or local, is the need for dynamic and collaborative workflow management. Traditional organizations have relied on well defined and structured processes and workflows for decades so the concept is not a new one. What makes virtual organizations different from traditional ones in these aspects is the stronger reliance on flexibility, adaptability and dynamic support for emergent processes. Emergent processes are at the core of virtual organizations. They represent the opposite view of predefined and static workflows or processes that are found in the traditional organizations. Emergent processes are *"opportunistic in nature and are often characterized by parallel and disconnected activities that must be coordinated to meet global goals"* [6]. Emergent processes are one of the main reasons that organizations come together as virtual organizations and must be acknowledged when managing workflows. They can be highly dynamic in nature, begin and end unexpectedly as goals change and their execution can vary from situation to situation.

5.1 Processes and workflows

Organizations come together as a virtual organization to achieve a business goal. The functions that the virtual organization performs to achieve the goal are grouped as processes that define the steps that need to be taken. The process defines rules, constraints and conditions of the tasks that form the work that is going to be performed [2]. Processes are rarely seen as monolithic entities but rather as a hierarchy of subprocesses and tasks. The organization might assign a single subprocess to each organizational section and they would coordinate their work in order to achieve the goal. As the processes and subprocesses grew in complexity, it became apparent that coordination and managing of processes should mature too. Well defined workflows were created for this reason. Workflow Management Coalition defines workflow as *"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"* [1].

To support workflow management, automatic workflow management tools were created. They manage how the processes should be modelled, work assigned, process executed and its progress monitored [11]. They help the organization to concentrate on their core competence instead of spending most of their time coordinating the work processes. The work is also well defined and it

is easy to make sure that processes are executed as they were planned. In addition, an automatic workflow management system has other benefits as well. It can be used to pick the right person or a sub-organization to perform the activity from number of similar candidates. The system can perform processes in parallel manner, accounting possible dependencies with other processes and other conditions that may affect the work order and it can structure activities inside the process flexibly depending on the situation.

Business processes and workflow management systems are a well understood and utilised part of modern organizational management. Virtual organizations, however, have many unique aspects that need to be understood and taken into consideration when managing processes and workflows. What is true to conventional workflow systems may not apply to virtual organizations since they are designed to work inside a single organization. Virtual organizations have to handle processes that go beyond organizational boundaries in a collaborative fashion. Virtual organizations operate on distributed business processes [2]. Distributed business processes are composite processes that are executed by multiple members of the virtual organization. This means that the workflow system must be able to manage inter-enterprise processes to be useful for virtual organizations.

5.2 Common themes

Distributed business processes are modelled and executed in a distributed environment that usually consists of autonomous and heterogeneous members. Such environment places varying requirements on the workflow management system. It must be able to model workflows in a way that every member can understand them and assign work to members no matter where they are located. In addition it must provide security, interoperability between virtual organization members and act as glue between wide varieties of legacy systems.

The Workflow Management Coalition defines the three functional areas of workflow management systems as build-time functions that are used to define and model workflow processes, run-time control functions that are concerned with managing the operational workflows and run-time interaction with humans and machines [7]. This means that workflow systems require efficient governance and interaction. To support emergent processes the workflow system must be able to accept new processes and start executing them flexibly. This requires interaction and good communication channels between the organization members. Efficient interaction requires interoperability between autonomous members.

Goal management can be seen as one of the most important purposes of efficient inter-organizational interaction. The basic views of goal management are making sure that everybody is aware of the current workflow's status and that there is adequate support for finding out how similar goals were treated in the past in this organization or in some other member organization [6]. These can be addressed by setting up different kinds of milestone systems and by efficiently sharing what has been learned from the past.

When it comes to virtual organizations there are also other points of goal management that are easily overlooked. Nowadays, it is commonly well understood that it is beneficial for the entire organization to be aware of the goal that they are working to fulfil. An organization should not limit itself to just giving the bare minimum of information to sections and subsections so they can organize their work, but aim to give every member of the organization the possibility of coming to know the big picture. This way everyone can see how their work improves the organization as a whole and how improvements in their area can lead to significant improvements in other areas as well. Failure to provide members of the organization the larger view can lead to decreased motivation and sub par services or products as members have to work with just the information

that the organization chooses to give them instead of being able to gather additional information and requirements themselves.

In virtual organizations, extra care should be placed on keeping all the member organizations of the workflow aware of the overall goal. This is not only an issue with workflow management thought. The contracts that govern the virtual organizations may not allow this kind of information sharing and interaction. The organization members may not be seen as peers but as subcontractors that are given enough information to provide their part of the service or product without compromising the overall goal. From the point of view of the collaborative workflow thought, everyone should have a clear idea of the goal for the same reason as in traditional organizations and process models.

The style of inter-organizational workflow management relies heavily on how the workflow managers may interoperate. Many categories have been raised but for electronic commerce the following have been suggested [13].

- Capacity sharing. Using a single workflow manager. This is usually not desirable in virtual organizations as it implies that all the data is shared while in virtual organizations the members prefer to remain autonomous without revealing their inner workings.
- Chained execution. Activities are distributed by task and different organizations are always responsible for certain tasks. This is not very flexible since members may change rapidly in the organization.
- Subcontracting. An organization subcontracts one of its tasks to some other member and receives the outcome. This is a common way of handling work in virtual organizations.
- Case transfer. A task may be executed by many organizations and it may be moved from any of the capable organizations. This is a flexible way of doing things in a modern organization. It can be used to assign the process to the most capable organization or to manage load balance. It requires extensive coordination and information sharing among the members.

The level of interoperability may limit the choice of a workflow system. Some workflow systems, such as many multi-agent systems, expect an organization where tasks are handled as case transfers in a peer-to-peer fashion. Some other systems may be more suitable for subcontracting. The level of human involvement is not strictly defined either and may differ between systems. Some workflow management systems may perform well without any human interaction and some may require human input.

5.3 Workflow models

Coordination is what stands out as the most important aspect of a workflow management system. Distributed business processes have to be coordinated among heterogeneous and autonomous nodes in virtual organization as well as taking care of process dependencies. Achievement of a business goal depends on different members of the virtual organization handling their processes in a timely manner.

The Workflow Management Coalition has attempted to address a need for workflow management system model by designing an abstract reference model that defines functional components and interfaces of a workflow system [7]. The model defines software components that are used to provide necessary functions, system and control data that are used by the software and application databases that can be invoked by the workflow.

The reference model is not meant to be used as such but its job is to identify the generic workflow application structure. There can be variance on how the structure is implemented between different workflow systems and for that reason the reference model defines interfaces for communication between these systems.

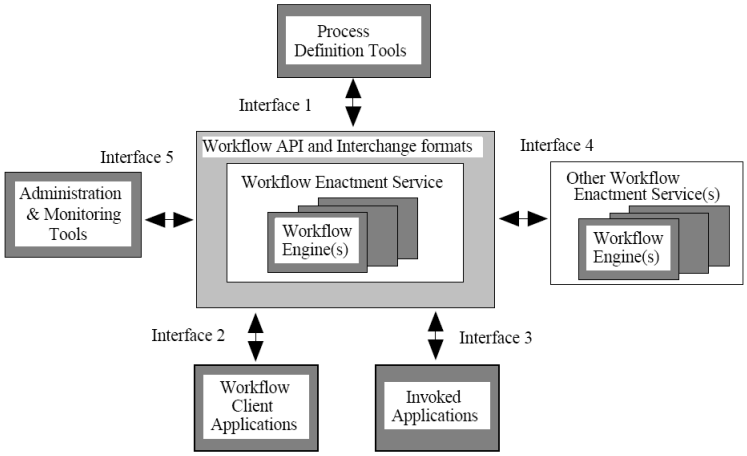


Figure 5.1: Workflow reference model [7].

Figure 5.1 shows the reference model. Workflow enactment service is the core of the model. It may contain one or more workflow engines that are responsible for managing and executing workflow instances. The enactment service interprets the process definition, assigns the activities to workflow managers and invokes application tools as necessary. It also defines interfaces to process definition tools, collaborative workflow engines, administrative tools and applications that are invoked during the workflow execution.

To support homogeneous services in heterogeneus environment, that is very common in virtual organizations, Workflow Management Coalition defines the following conformance levels to support increasing levels of common functionality [7].

- A common naming scheme across the heterogeneous domain
- Support for common process definition objects and attributes across the domain
- Support for workflow relevant data transfer across the domain
- Support for process, subprocess or activity transfer between heterogeneous workflow engines
- Support for common administration and monitoring functions within the domain

There have also been efforts to standardize the languages that are used to model the workflows. The reason being that workflow should be easily configured and should not require in-depth technical knowledge such as programming ability. From the system's point of view, every member in the organization has to have a clear idea of the process they are handling. Process must be defined in a way that it supports automated manipulation and execution by workflow systems [10]. The process definition will define what activities are to be taken, what the relationships between the actions are, how the process is started and stopped and what resources are accessed (Figure 5.2).

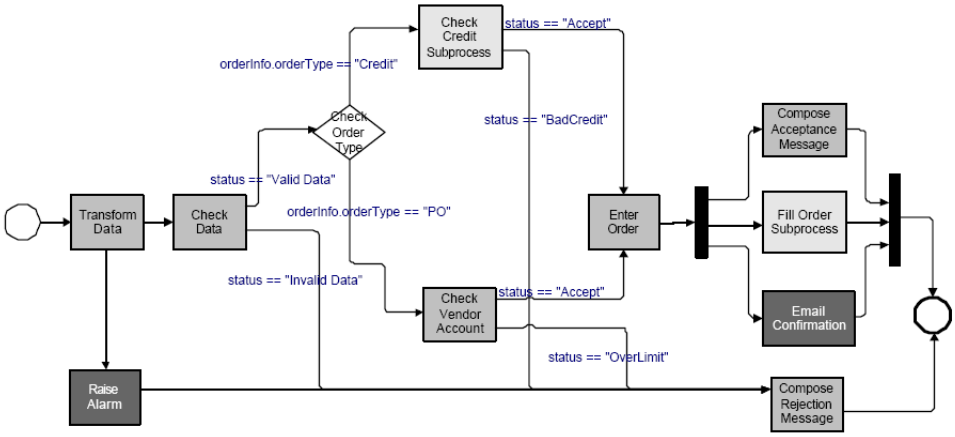


Figure 5.2: An example of a workflow model [10].

The Workflow Management Coalition's answer for workflow modelling is XPDL, a XML based language developed for process definition. Many different workflow systems use this language to model their workflows and provide interoperability between systems by using a common language. Other languages used to model workflows are Unified Modelling Language, Process Interchange Framework, Process Specification Language and OASIS WS-BPEL.

As introduced in the previous chapters, there are common themes and standards to workflow requirements, modelling and management systems. However, there have been multiple different views on how to design and implement a working workflow management system that supports emergent distributed business processes in virtual organizations. There is no clear cut right approach for this problem and in the following sections four different viewpoints are introduced. These range from centralized and collaborative management system based views to agent and contract based views of a higher abstraction level.

5.3.1 Centralized

A basic way of handling such a collaborative workflow is to assign one member of the virtual organization as a manager and coordinator. The coordinator supervises the dependencies among the business processes and assigns processes to organization members. These members may in turn decompose their processes into smaller pieces and assign them to other members. The virtual organization's execution system is in charge of executing this workflow [3].

The workflow is usually modelled in a standard language such as XPDL (see 5.3). The execution system has to do with process execution and coordination as well as information management and communication. The PRODNET project [12] has designed an execution system that is based around a coordination kernel that coordinates the process execution. It has three layers. The core cooperation layer is responsible for interactions between virtual organization members, enterprise management functionalities handle enterprise level coordination of business processes with virtual organization partners, and virtual enterprise management functionalities include the functions that the virtual organization coordination member will use to manage the workflow.

Each node in virtual organizations includes this coordination kernel as well as infrastructure for safe communication and support services functions. The Local Coordination Manager handles the local workflow execution and interaction between nodes is handled through PRODNET Communications Infrastructure.

The PRODNET approach is a fairly traditional, centralized and transaction based system. Its roots are based on intra-organization workflow management and Workflow Management Coalitions reference model when it comes to workflow handling.

5.3.2 Collaborative

Transaction based execution systems such as the one designed in PRODNET have received some criticism for not being flexible enough and not particularly well suited for a modern virtual organization that has to cope with emergent processes. The proposed idea is to move from centralized coordination of one member to a more collaborative view.

HP Labs introduces a Collaborative Business Manager to support decentralized and peer-to-peer process management [4]. The organization executes a distributed business process that is instantiated multiple times into peer process instances that are in turn run by the Collaborative Process Managers in the organization members. There is no centralized manager such as the one in PRODNET's execution system. The workflow is synchronized between the members using a peer-to-peer messaging protocol.

Collaborative Business Managers (CPM) handle distributed business processes that have been extended to include a list of process roles that describe the participants of the process. These Collaborative Processes may have multiple participants that are named by their role in the business process and for each role, one peer instance of the Collaborative Process is run. The instances have the same process definition but have other properties that differentiate them. Process role property indicates the process instance's task and goal. For example in a purchase process there may be a *buyer* and *seller* peer instances that are executed in their respective Collaborative Business Managers. Each task has also a role called task role that must be the same than one of the process roles. This task can only be executed by a peer process instance that has the same role name. Collaborative Process also supports member autonomy by adding sharing scope to process data. Sharing scope is defined by using data templates that have a precise scope. The data can be made public or limit the scope to just some process-roles.

Collaborative Processes are executed in Collaborative Business Managers. One instance of the process is made for each process role and the instance is run by the virtual organization member's Collaborative Business Manager that is responsible for the process role. The Collaborative Business Managers have a *Player* property that specifies its responsibility in the process execution by four attributes. The *role* attribute defines what peer process instance the CPM will bind to and run. The *domain name* defines a domain that the CPM belongs to, *local-name* its unique name inside the domain, *coop-key* for identifying process instance and *messaging service* that defines what messaging infrastructure the CPM will use to communicate with other CPMs outside of its domain.

All the peer process instances share a process definition that is defined in a suitable modelling language but will execute different parts of it based on their process role (Figure 5.3). For example a CPM that is responsible of the role buyer will execute buyer side instance of the process, identified by role name and coop-key, that executes buyer actions and then informs seller that it's finished. Both virtual organization members running their CPMs remain autonomous but follow the same protocol defined in the Collaborative Process. A peer-to-peer message passing protocol is used between CPMs to manage process execution and dependency handling.

This approach supports a more collaborative workflow management. It does not rely on a single member of a organization to act as a coordinator and so it strengthens the member autonomy.

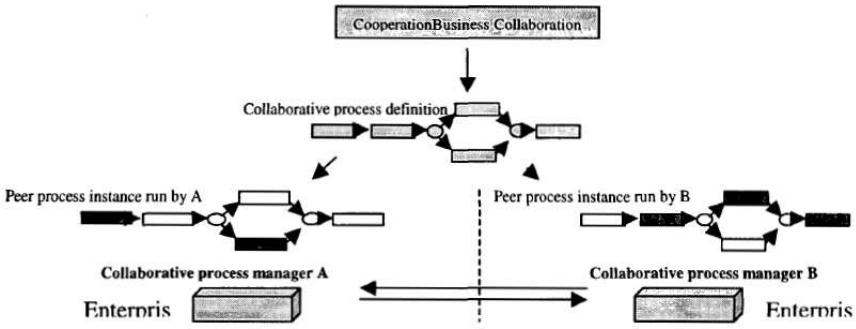


Figure 5.3: Collaborative workflow execution [4].

5.3.3 Agent-based

Modelling virtual organizations as a society of agents is a natural way of raising the abstraction level. Each member of the organization is seen as an agent that remains autonomous while providing services. Such multi-agent system have usually their own way of handling the workflow that leverages the strengths of the agent-based modelling.

There have been many suggestions on how to handle the virtual organization as a multi-agent system. Research projects such as MASSYVE [9] have designed systems to implement them. The basic idea behind them is modelling the virtual organization as a multi-agent system and loading the workflow model into it.

A paper by a project in the University of Tsinghua [5] describes attempts to handle virtual organization management and workflow collaboration using distributed business process model that is loaded on multi-agent system (Figure 5.4). This is achieved by starting from the comparison of distributed business processes and agent-based virtual organizations. They both form a distributed model where nodes in the network provide services while hiding their local architecture and subprocesses. The members of a distributed business process model can be mapped to agents in the agent-based model. In this model, there are activity agents that handle execution and resource agents that correspond to human or machine resources. Once the distributed business process has been loaded to the multi-agent system, its workflow management can be seen as agent interaction and problem solving instead of pre-planned process execution. Coordinating the agents' actions appropriately, the distributed business process can be executed according to the workflow.

Two types of agents come to play when coordinating the workflow. Collaboration between activity agents manages the workflow dependencies. The activity agents have information about what interactions and collaborations should be handled before the current activity and how the activity depends on them. They wait for notifications from agents that have to complete their work before the current work can begin. Collaboration between activity agents and resource agents invoke necessary resource assignments such as the execution of the actual work by a machine or a human. Once the work is executed, the activity agent can inform other activity agents of it.

Agent-based workflow modelling raises the workflow management abstraction level. It is no longer necessary to consider the distributed business processes as steps in the workflow model and instructions to the workflow manager but as agent interactions. Research on rational agents has been going on for decades and there are a wide range of techniques on managing agent collaborations. Emergent processes can be handled by adding new activity and resource agents to the model and defining their interactions.

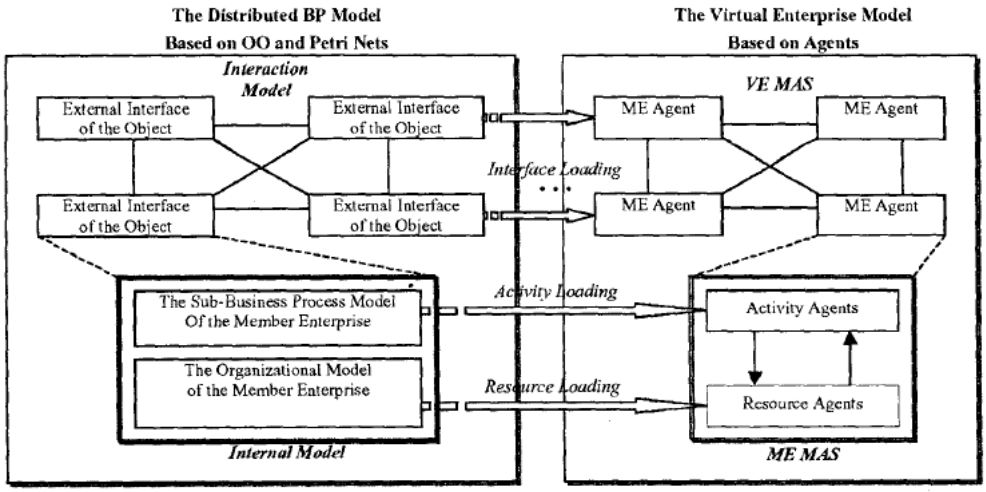


Figure 5.4: A mlti-agent system [5].

5.3.4 Contract-based

Contract-based workflow management takes a different view of the collaboration. Whereas agent-based workflow systems emphasise the modelling of the whole virtual organization as a society of agents, the contract-based workflow management concentrates on the relationships between the members. Contracts are especially useful when the relationships between virtual organization members are not seen as peer-to-peer but rather as contractor-to-subcontractor.

The CrossFlow project [8] has developed a contract-based workflow system. A contract defines what sort of a product or service this process handles. It also makes clear the rules that are used in the transaction. Contracts can be formed from template contracts by adding the appropriate data. The requirements for the contracts are that they are well structured and easily searchable by automatic systems, are reusable and not dependant on particular workflow manager, encapsulate the information about the business process in an abstract way with fine-grained control and offer legality between partners (Figure 5.5).

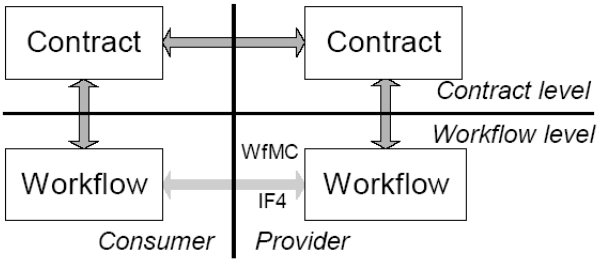


Figure 5.5: Contracts layer [8].

Contracts can be used to trade products or services but in this case they are used as a backbone for a workflow management system and the contracts are used to trade business processes. The contract handling is built on top of the workflow manager. When a service consumer wants to outsource a process to a service provider, it uses a matchmaker to search for it and receives a

contract in response. The contract managers between consumer and provider are not tightly linked. The contracts specify the rules and interactions that process requests and these are passed to the workflow engine below that will communicate them to the provider workflow manager. This raises the abstraction level of the actual workflow rules handling and thus adds more flexibility to the system.

A single contract can be divided into five basic concepts that together define the workflow. These concepts interleave and use each other and are not independent. The concepts are usage model, natural language description, concept model, process model and the enactment model [8] (Figure 5.6).

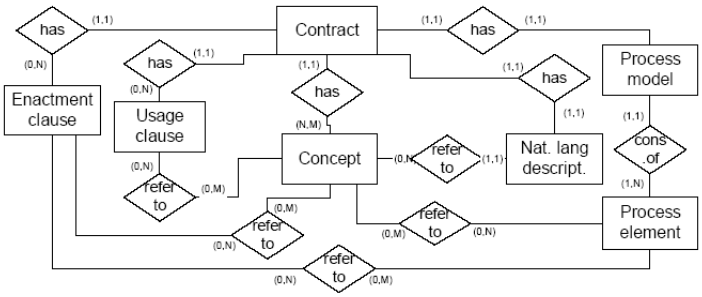


Figure 5.6: Contract structure [8].

Usage model and natural language description are more minor details. The usage model defines usage scenarios. It describes how the contract will be used from a simple service execution to multi conditional resource reservations and process executions with dependencies. The natural language description is a description meant for human reading.

The concept model defines the terminology for the contract as a list of parameters that have a name, type and description. General parameters are part of every contract and are meant to make sure that every contract has the same basic structure with parameters such as consumer and provider. Service specific parameters, such as address or cost, are relevant only to the current service and are used as parameters for the service provider. Process variables are important for the business process and workflow. They define the workflow data and are meant to control the workflow execution. Process variables may be for example intermediate results in the workflow chain that are returned to the service consumer that can use the values to abort the workflow or otherwise control it.

Whereas the concept model defines the terminology, the process model specifies the workflow. The process model defines an abstraction of the workflow whose exact implementation is hidden by the provider. The idea is to make sure that both parties have a similar view of the workflow but neither has to reveal the actual implementation. To make sure that the process is defined in a way that both parties can understand it, XPDL (see 5.3) or a similar language is used to define the workflow.

The enactment model describes the collaboration functions and information that can be used during the workflow execution. These include control operations to manage the workflow, monitoring functions to query the process variables, operations to change the workflow and other operations such as transaction management.

To execute the contract, it is passed to the contract manager that will forward it underlying workflow managers based on concept, process and enactment models. Contract-based workflow management is based on the idea of higher abstraction and hiding the implementation from virtual

organization partners. The contract encapsulates all the information that is required to execute the workflow and it can be used on top of a workflow manager. The contract-based workflow model is designed to allow a flexible and dynamic way to add workflow definitions and execute them.

5.4 Conclusions

Virtual organizations operate on distributed business processes. Business processes specify activities, conditions and rules that are required to achieve a specific business goal. Distributed business processes are handled between autonomous and heterogeneous members of the virtual organization. Workflow management systems are needed to handle complex decencies, interaction and otherwise manage the process workflow.

Workflow management is a well researched area. Workflow Management Coalition has defined standard language for modelling workflows and a workflow reference model that can be used as an abstract framework for building workflow management systems. Virtual organizations, however, require more from a workflow management system than a traditional organization. The workflow management system must be able to interact between varying member organizations and preferably to be able to handle dynamic and emergent processes.

Different workflow management models have been suggested and four of them were introduced in this chapter. The centralized model relies on a single coordinator to handle the workflow. PRODNET is an example of this kind of system. More collaborative versions use peer-to-peer kind of messaging protocols to execute the workflow without centralized command. The workflow management abstraction layer can be raised by modelling the workflows as multi-agent systems or as contracts. Multi-agent systems require that the virtual organization is modelled as an agent system and workflow handled as collaboration of agents. Contract-based workflow management encapsulates workflow information into contracts that can be run on top of a chosen workflow management system. This approach relies on higher abstraction level to provide flexibility.

There are already a host of fairly mature workflow management systems and the Workflow Management Coalition aims to support the spreading of standardised workflow systems and modelling languages. From the point of view of virtual organizations, though, many of the solutions seem lacking. Many of the projects are still at infant stages or rely on centralized managers that do not support a truly collaborative view of workflow management. In addition, many of the common themes that are suggested in various papers, such as goal management, are not supported at all in any of the workflow management systems. It will still take some time for the workflow systems to mature for the use of virtual organizations.

Bibliography

[1] ALLEN, R. Workflow: An introduction. In *Workflow Handbook 2001* (2001). URL http://www.wfmc.org/standards/docs/Workflow_An_Introduction.pdf.

[2] CAMARINHA-MATOS, L. M., AND AFSARMANESH, H. Virtual enterprise modelling and support infrastructures: Applying multi-agent system approaches. In *Multi-Agent Systems and Applications : 9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School* (2001), Springer-Verlag, p. 335.

[3] CAMARINHA-MATOS, L. M., AND PANTOJA-LIMA, C. Towards an execution system for distributed business processes in a virtual enterprise. In *Proceedings of the 8th International Conference on High-Performance Computing and Networking* (2000), Springer-Verlag, pp. 149–162.

[4] CHEN, Q., AND HSU, M. Inter-enterprise collaborative business process management. In *Proceedings of the 17th International Conference on Data Engineering* (2001), IEEE Computer Society, pp. 253–260.

[5] GOU, H., HUANG, B., LIU, W., LI, Y., AND REN, S. Operation management of virtual enterprises. In *2001 IEEE International Conference on Systems, Man, and Cybernetics* (2001), vol. 3, pp. 2046–2051.

[6] HAWRYSZKIEWYCZ, I. Supporting teams in virtual organizations. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications* (1999), Springer-Verlag, pp. 550–559.

[7] HOLLINGSWORTH, D. The workflow reference model. Tech. rep., The Workflow Management Coalition, 1995. URL http://www.wfmc.org/standards/docs/tc003v11.pdf.

[8] KOETSIER, M., GREFEN, P. W. P. J., AND VONK, J. Contracts for cross-organizational workflow management. In *Proceedings of the First International Conference on Electronic Commerce and Web Technologies* (2000), Springer-Verlag, pp. 110–121.

[9] The MASSYVE project homepage, 2004. URL http://www.gsigma-grucon.ufsc.br/massyve/.

[10] NORIN, R., ET AL. Workflow process definition interface – XML process definition language. Tech. rep., The Workflow Management Coalition, 2002. URL http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf.

[11] PLESUMS, C. An introduction to workflow. In *Workflow Handbook 2002* (2002). URL http://www.wfmc.org/information/introduction_to_workflow02.pdf.

[12] The PRODNET project homepage, 1999. URL http://www.uninova.pt/~prodnet/.

[13] TAGG, R. Workflow in different styles of virtual enterprise. *Aust. Comput. Sci. Commun. 23*, 6 (2001), 21–28.

Chapter 6

Virtual organisation monitoring

Toni Ruokolainen

Inherent properties of virtual organizations such as heterogeneity of technological platforms, autonomic participants and dynamism of both service and collaboration network evolution makes it necessary to monitor the operation of a community.

Monitoring is a procedure where the execution or communication of a program is intercepted such that an execution trace can be created. An execution trace represents transitions between program state and is usually an abstracted model of its execution. From an execution trace some interesting information can be extracted, most importantly its conformance to a given specification.

In this chapter we will introduce the basic concepts of monitoring or runtime verification. We will start in Section 6.1 with some background from software testing and requirements engineering areas, where monitoring of monitoring of program execution has been actively used since early nineties [47, 18].

Concepts of monitoring platforms are identified in Section 6.2 and runtime verification in general is discussed in Section 6.3. The notion of formal specifications which are used for expressing the "monitoring rules" are discussed in Section 6.4. Methods for instrumenting programs with necessary code and generation of behavioral observers are introduced in Sections 6.5 and 6.6 respectively. In Section 6.7 a few tools and platforms that can be used for implementing virtual organization monitoring platforms are introduced.

6.1 Background

Virtual organization monitoring techniques and concepts are based on the research initiated in the early nineties in areas of software testing [47, 14] and requirements engineering [24, 18]. Research in these areas concentrated on the problems involved in different phases of software engineering processes. Concepts such as specification based observers and code instrumentation, which are used also for virtual organization monitoring, were addressed already in these research areas. In this section we will introduce approaches taken in software testing and requirements engineering to validate the runtime behavior of software systems.

The software testing process involves several tasks, such as the description of test cases, implementation of test cases (e.g. test drivers, suites or frameworks), test execution monitoring and test result verification [47]. Usually these tasks are quite informal and ad hoc in a sense that the description of test cases is done using for example native language and test cases are implemented manually. Such an informal testing process usually does not provide sufficient coverage and it

lacks effective means for determining if the system has behaved correctly under test executions [47].

To overcome deficiencies of manual testing processes, procedures to derive *test oracles* from formal specifications have been developed [47, 14, 13]. A test oracle is a mechanism for determining behavioral correspondence between implementation and its specification under a test run [47]. Test oracle provides a partial correctness validation criterion for a software system, i.e. correct behavior under the assumptions of a restricted test case. A test oracle has two components, namely *oracle information* and *oracle procedure* [47].

Oracle information specifies the correct behavior or property expected from the system in a specific testing scenario [47]. The oracle procedure verifies the test execution results with respect to the corresponding oracle information. A simple form of oracle information is input/output value pairs which are compared by the oracle procedure with the corresponding runtime values. The oracle procedure for input/output values is simply a equivalence operator, "=" [47]. Another example of simple test oracles are embedded assertions available in many programming languages such as C/C++ and Java.

Describing the multi-paradigmatic nature of complex, reactive systems may require the use of different specification formalisms. For example the specification of a distributed system may require use of different formalisms to describe all the behavioral aspects. For example computational or functional aspects of a component or the whole system might be specified using a process description language whereas another formalism could be more suitable for defining synchronization or causality aspects between distinct components. Every specification produces a test oracle and the consistency of a system against its specifications is tested by verifying all oracles against the test data [13].

In [13] procedures applicable for generating test oracles from multi-lingual specifications are given. Given a specification formula (in temporal logic) and a set of rules representing the semantics of the specification language, the generic algorithm described in [13] generates a nondeterministic finite automaton which can be used as a test-oracle.

A test oracle usage in testing process includes three phases: 1) deriving test oracles from specifications, 2) monitoring test execution, and 3) applying oracle procedures to an execution profile which includes all the information gathered by the monitor [47].

The derivation of test oracles (and monitors) from specification can be done using a tableau method [13], a general decision method first implemented for use in context of Propositional Dynamic Logic [45]. The tableau generated from a temporal logic formula describes a finite state automaton which accepts precisely those execution traces generated in testing satisfying the specification [13]. Generation of observers from specifications is discussed in more detail in Section 6.6.

The monitoring of test execution is accomplished by collecting information of relevant execution states and mapping the runtime control and data related concepts into concepts of specification language [47]. An information collection mechanism can be provided by e.g. logging or event functionality. The actual program code must of course include necessary functionality to emit the information needed for monitoring purposes. This is usually done manually (as in traditional software testing processes) or automatically by deriving information gathering functionality from the specifications.

The oracle procedures analyze the test results (execution trace) against the oracle information to verify behavioral correctness [14]. If an execution trace violates a specification, an oracle may construct a formula describing the violation. This information provides valuable advice for system designers about possible flaws in the software.

Requirements monitoring addresses *software engineering issues* emerging in situations where

environment of deployed software or the requirements of software itself does not remain static [18]. Requirements monitoring is a technique where an operational system is actively monitored for situations where the behavior of the system deviates from assumptions of the software [17].

Requirements monitoring is needed for example in long-lived, dynamic enterprise-application environments [18]. Concrete examples of such operational environments are for example FlexLM, a software license manager, and Lotus Notes, an intra-enterprise collaboration software. Software developed for such environments needs requirements monitoring functionality to guarantee that its assumptions of resource and operating needs are respected by the operational environment [18].

Requirement monitors are installed to gather and analyze information about the system's runtime environment, not the executing program itself [18]. Thus the surrounding environment must provide necessary functionality to emit information about interesting system properties, such as size of user population, status of the platform or processor load.

Specifications of what to monitor are given to the monitor, such that divergences from the assumptions are detected. Specifications are derived using a non-formal requirements refinement methodology. Ideal, high-level requirements specified in software design documentation are manually subdivided into finer-grained requirements. From each subdivided requirement assumptions are identified. Assumptions emerging from the requirements are candidates for monitoring [18].

An important feature in the requirements monitoring approach introduced in [18] is the notion of evolution support. Each assumption identified from the requirements of software is attached with a compensatory remedial action. Remedial evolutions provides actions to be taken when mismatch between assumptions and the current environment is developed [18]. For example an assumption for license manager "*User population* $< k$" could be attached with a remedial action "*Purchase more licenses or reduce user population*". In Table 6.1 is an example subdivision of a license manager requirement "Users should not have to wait unduly long for a license" into its subdivided requirements and their corresponding assumptions and compensatory actions, which should be taken if assumptions are violated [18].

Subdivided requirement	Assumption	Remedy
Licenses sufficient for user population	User population < k	Purchase more licenses or reduce user population
	No more that $x\%$ of user population wants to use program at once	Purchase more licenses or reduce user population
Licenses to individual users are served fairly	The longest waiting user gets license first	Have license manager maintain a queue of waiting users
	Users do not hog licenses	Issue time-bounded licenses
		Revoke licenses of current users
Users are not kept waiting if licenses are available	License manager on a reliable platform	Relocate license manager to a more reliable platform
		Employ more robust license manager design
	License requests do not become backlogged at license manager	Subdivide license manager and licenses across several platforms

Table 6.1: Division of the requirement "Users should not have to wait unduly long for a license" [18].

Remedial actions are specified to support manual system evolution by its maintainers, i.e. the system administrator might receive an e-mail of what assumption has been broken and what should be done. If the assumptions derived from the requirements and remedial actions are described formally, even automatic evolution (or reconciliation) is possible to some extent [17]. Requirements monitoring systems may also provide a feedback loop between the runtime and development time environments [24], resulting in an evolutionary software development process.

Quite a few monitoring platforms have been implemented for software testing and requirements engineering purposes. TAOS (Testing with Analysis and Oracle Support) is a toolkit which provides test oracle based support for testing processes [46]. Execution profiles are extracted by instrumenting the source code with a tool called Artemis [46]. After the execution of each test case, TAOS applies all oracle procedures, which basically compare the execution trace and input / output values to those specified in oracle information.

Bridget is an agent-based requirements monitoring environment which was developed to support evolutionary prototype-based software engineering processes and to bridge the gap between developers and end-users in analysis and design phases [24]. In Bridget behavior of end users working with a prototype system is monitored by software agents. Mismatches between the expectations made by developers and actual system usage are reported back to developers. Bridget was developed for monitoring use cases of user interfaces and initiating a dialog between developers and end-users [24].

AMOS/FLEA is a prototype monitoring system and requirements specification language developed for runtime monitoring of system requirements and assumptions [10, 17]. FLEA (Formal Language for Expressing Assumptions) language provides constructs for expressing temporal combinations of events [17]. Monitoring code is generated from FLEA descriptions automatically. AMOS/FLEA system comprises a historical database management system for storing event information, an inference mechanism for concluding conformance between events and requirements, and a communication mechanisms to gather events from the actual system and distributed notifications of event combinations [17].

Monitoring approaches for software testing concentrate on formal runtime verification of a software artifact. The object of observation is usually a testing phase software system and its functional properties. Especially methods for derivation of testing oracles from temporal specifications have been developed and successfully applied in software testing. Another important method addressed in the test-oracle based monitoring approach is the instrumentation of program code. Results and methods studied in test-phase monitoring are also applicable for virtual organization monitoring purposes.

Requirements monitoring mainly concentrates on non-functional aspects of the operational environment. Typical objects of monitoring are different performance values such as processor load or the number of users in a system. What makes research conducted under requirements monitoring area is the identification and formalization of a notion of reflection between the operational environment and its maintainers [18, 17]. A similar reflection mechanism can also be used in virtual organization environments for execution of compensatory actions needed for reconciliation of e.g. contract breaches.

Monitoring in virtual organizations is typically a process of contract supervision [39]. Contracts in virtual organizations express properties for a community, such as business goals, organizational roles, legal agreements, technical bindings and behavioral specifications and constraints [39]. Virtual organization contracts and contract management in general is discussed in Chapter 7. We will concentrate from now on to the *behavioral aspects* of contracts and monitoring of *conformance* between actual system (community) behavior and its specification.

6.2 Monitoring concepts

Monitoring system contains several phases and functionalities, such as observers and specifications. In this section we will introduce the basic concepts of typical monitoring systems; these are illustrated in Figure 6.1. The conceptualization is slightly adapted from [20] and specialized for purposes of virtual organization monitoring and follows the ideas of e.g. [10, 17, 48, 40].

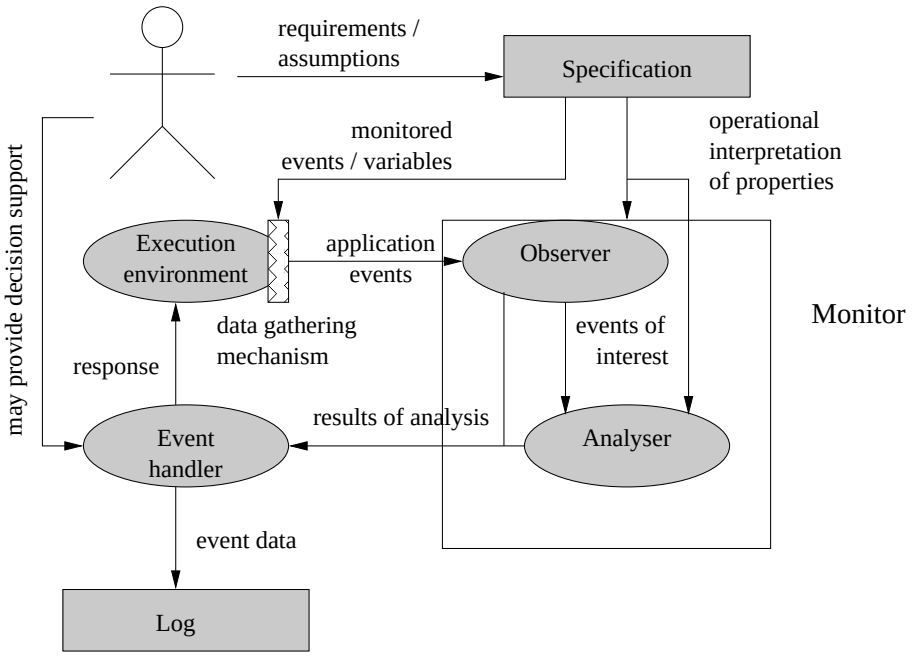


Figure 6.1: Concepts of system monitoring.

Monitoring requirements and assumptions are first provided by the user. These specifications are basically behavioral descriptions or constraints that must be respected by the monitored system. Different kinds of formalisms have been used to specify monitoring assumptions and requirements, such as event calculus [36], process algebras [5], linear temporal logic [3, 19] or interval logic [14]. Specifications are given an operational interpretation in form of a finite state automaton and then forwarded to the observer and analyzer components.

Monitoring assumptions are behavioral descriptions of either the environment or the system-under-test itself. For virtual organizations a process description language with formal semantics is typically used to describe monitoring assumptions. Process algebras, such as CSP, CCS and π-calculus, Petri nets [42] or finite automata [48] can be used for this purpose. Behavioral descriptions in virtual organizations are used as a basis for a contractual agreement on joint behavior. For monitoring purposes, these behavioral descriptions are interpreted as finite state machines whose transitions are labeled by e.g. communication events. Simulating a finite state machine parallelly with actual system execution provides a mechanism to guarantee conformance to the expected, mutually agreed behavior.

Monitoring requirements define constraints over system behavior, usually in form of temporal or action-based logic. As monitored event traces are linear sequences of state changes, linear temporal logics are typically used to provide constraints over system behavior [3, 19]. Temporal logics are used to define safety and liveness properties for system behavior which state that “nothing bad happens” and “something good happens”, respectively [7].

It should be noted that usually, especially in traditional monitoring approaches, no behavioral descriptions per se are used but only linear temporal logic (LTL) or other temporal logic is used as a specification. The use of temporal and modal logics for software verification originates from *model checking* [7] where temporal logics are used as a specification language and finite state automata are used as an abstract model of implementation. When temporal logics are used as a specification language for runtime verification (monitoring) purposes, the event traces act as the abstract model of computation. Virtual organizations and their contracts, however, benefit from both kind of behavioral descriptions. Behavioral descriptions are used as high-level business process execution instructions and behavioral contracts whereas temporal logics (e.g. LTL) are used to express constraints and policies for the community [39, 40]. We will introduce some specification formalisms and their use in Section 6.4.

A *Data gathering mechanism* provides a trace of events for the actual monitor. These events might be triggered by message passing actions or function calls [48]. An Observer uses this information to validate conformance between specifications of a program and its actual behavior. Information flow between an observer and monitored system may be implemented with mechanisms like logging [1, 48] or event-based methods as in [34] or [17].

The Data gathering mechanism is typically enabled by instrumentation of actual system implementation [48, 20] or by intercepting communication or execution of the actual system [40]. Program instrumentation makes the actual program to emit events of important activities whereas interceptors are part of the environment. Interceptors can be implemented for example as a part of the service proxies in execution platform middleware or as part of a general API, if all services are expected to use a common API-framework. Interception of communication is especially suitable for process execution platforms, for example as an integrated part of a workflow execution engine. We will not, however, discuss the interception-based approach further in this paper mainly because instrumentation-based monitoring approaches are more general. Instrumentation of program code is further discussed in Section 6.5.

The *Observer* is responsible for interpreting the core events coming from the data gathering system, i.e. for translating the execution platform and application specific event information to concepts comparable with the specification language. The Observer provides this event information to the analyzer, if necessary. More importantly an observer is a testing device which observes the system under test (SUT) and checks whether the event stream emitted from the system conforms to the behavioral specifications [49]. An event stream (trace) conforms to a specification A, if A can generate the corresponding trace [49].

The observation of events may happen either actively or passively. In active observation, the observer is provided with a query interface to the actual system. When the observer needs to know the state of the system, it makes a query which returns the state. In passive observation, the observer receives state information via a callback routine or an event mechanism. Active observing is less demanding for the infrastructure but requires more activity and functionality from the observer itself. An example scenario of active observation would be a system where the SUT writes its actions to a log file and the observer periodically (or after the execution of the program has finished) fetches the log of events from the file. Passive observation is more demanding for the infrastructure as it needs some additional services, such as event mediators or callback-functionality.

The *Analyser* is a monitoring functionality that performs more complex analysis of system functionality. In distributed systems these tests are typically some kind of concurrency analysis such as deadlock or data race detection [27] or trust analysis (see Chapter 8). In serial systems an analyser may provide for example resource usage analysis for load balancing purposes. Where an observer may only tell if an action has violated requirements, given a behavioral descriptions

and an event trace, the analyser may do more complex tests, such as testing if the action just taken may be about to violate requirements. Results of an analyser might then be used in the future to prevent non-conforming behavior.

Results of the analysis from observer and analyser components are given to the *Event handler*. Event handler is responsible for storing analysis results for later use (e.g. as an evidence of contract breach / conformance) and for making decisions about how system should evolve especially in case of behavioral violations [20]. In situations where automatic decisions are not possible or are too risky, external decision support may be consulted.

6.3 Conformance validation

Verification techniques have been used for decades to ensure that a software or hardware artifact satisfies its specifications. Theorem proving and model checking techniques have been developed to mechanize proofs [23] but they are still in their infancy when considering their use in practice. Verification of even a simple Java program requires quite a lot of mathematical knowledge and a complete verification of program behavior is even theoretically impossible (Turing completeness). In the context of virtual organizations complete static verification of community behavior is practically impossible, due to dynamic nature of organizations' policies and evolving services.

Light-weight formal techniques, such as simulation and testing, have been proposed for bringing mathematical formalisms into everyday software engineering processes. Runtime monitoring is also a light-weight formal method [23]. Runtime monitoring may include simulation over behavioral descriptions and testing for temporal specification conformance.

Runtime verification is based on the same techniques that have been used for automaton based model-checking, e.g. generation of a test automaton or on-the-fly techniques for validating that the model conforms to the temporal specification [7]. Where in model checking the models are infinite sequences of events (usually represented as a Büchi automaton), in runtime verification the models are finite sequences of events [23].

The most common way to implement a runtime monitoring system is to use event logging mechanisms as method of gathering execution trace information [1]. Another common way is to use event-based mechanisms [48]. Log-based mechanisms are more applicable for local monitoring systems and they need less infrastructure services. Event-based monitoring systems can be effectively distributed but their use requires that the execution environment provides necessary messaging functionality in the form of for example publish-subscribe mechanisms or message oriented middleware.

Runtime verification of program execution is an automata theoretic verification method. Instrumented code emits events which represent the program execution trace and finite state automata are used to validate that the execution trace corresponds to the specification. In the following we will briefly introduce this approach using a log-based analysis technique as a concrete example.

In log-based analysis actual programs output messages of their progression into a logging system, usually a file or a database [1]. Events of interest are typically such things as inputs and outputs, messages sent and received, parameters and return values of important functions, and changes of state of key variables. Information recorded to the logging system is then consumed by log-file analysers.

A program which is to by analysed should write events to a logging system following some well-defined, agreed-upon logging policy [1]. Logging-policy states "formally" what kind of information should be written to the logging system and under what conditions.

A log file analyser is implemented as a set of finite-state machines [1]. Each automaton represents a distinct requirement in program specification. Finite-state machines analyse the program execution by interpreting the lines of the execution log as transitions between automaton states. Every line in a log-file represents a transition in one or more finite-state machines. The log-file is read line-by-line and states of the automata are updated if their transitions correspond to the event in the log-file. A log file (sequence of lines) is accepted by an analyser machine, if the state machine evolves from its initial state to a final state. An analyser accepts a log, if all its constituent machines accept it [1].

Another example implementation of log based runtime-verification is ReqMon introduced in [48]. ReqMon has an interesting approach which combines both model-checking and runtime monitoring to provide a platform where violations of specifications can be detected before they actually have happened. The ReqMon platform is implemented for Java-language platforms and the monitored programs are instrumented with additional bytecode to generate a stream of events that a monitor can interpret [48].

The monitor program of ReqMon continuously checks the program event log and creates a modified Java program from the original source code of the program and the program event log. The modified Java program represents the method call trace of the actual program [48]. A formal model described in the Promela language is generated from the modified Java program using a *java2spin* program for verification purposes. The generated Promela model is then verified against program specifications with the model checker Spin [28].

6.4 Behavioral specifications

Specifications given to a monitoring system are based on formal semantics and propositional logics. In traditional testing and requirements monitoring approaches temporal logics are used as the only specification language [47, 14]. Virtual organizations, however, usually include process-like behaviour which is described using process description languages such as WSCI [53], BPEL4WS [41] or WS-CDL [31].

For monitoring purposes, formal semantics for the process description should be defined. Formal semantics enables for example automatic deadlock analysis or interoperability validation. Process algebras and Petri nets are popular formalisms used as semantic framework for several business descriptions languages, process execution engines and desing tools. For example XLANG and WS-CDL are based on π-calculus semantics where as tool providers are more usually more accustomed to the executable nature on Petri nets [33].

Temporal specifications describe constraints on the order in which events may occur during the execution of a program [13]. Linear Temporal Logic (LTL) [7] is a propositional logic extended with future or past time operators [27]. The syntax of LTL logics is given in Table 6.2. A is a set of propositions (atomic names corresponding to the primitive concepts of specialisation language) and $\otimes$ is an exclusive or operation [27].

$F ::=$	$true \mid false \mid A \mid \neg F \mid F \vee F \mid F \wedge F \mid F \otimes F \mid F \rightarrow F \mid F \leftrightarrow F$	(prop. op)
	$\bigcirc F \mid \Diamond F \mid \Box F \mid F \mathcal{U}_s F \mid F \mathcal{U}_w F$	(future time op)

Table 6.2: Syntax of future time LTL logic formulas

Temporal operators of LTL are read as following: $\bigcirc F$ "next F", $\Diamond F$ "eventually F", $\Box F$ "always F", $F_1 \mathcal{U}_s F_2$ "F_1 strong until F_2" and $F_1 \mathcal{U}_w F_2$ " F_1 weak until F_2" [27]. Formula $\bigcirc F$ holds if F holds in the next step of the trace, $\Diamond F$ holds if F holds in some future time point, $\Box F$

holds if F holds in all future time points. The temporal formula $F_1 \mathcal{U}_s F_2$ holds is F_2 holds some time in the future and until then F_1 holds. Formula $F_1 \mathcal{U}_w F_2$ holds if F_1 holds in all future time points or if F_2 holds in some future point and until then F_1 holds [27].

Temporal logics are usually interpreted over infinite state models [7]. However the models of runtime verification (execution traces) are *finite* and the temporal formulas must generally be interpreted over finite sequences of states [19]. A finite trace t of length n is a sequence of events $s_0 s_1 s_2 \dots s_n$. A (sub)trace starting from event i, $s_i s_{i+1} \dots s_n$ is denoted $t^{(i)}$, $1 \leq i \leq n$. Interpretation of future time LTL for finite sequences is given in Table 6.3 [27].

$t \models a$	iff	$a(s_1)$ holds
$t \models \bigcirc F$	iff	$t' \models F$ where $t' = t^{(2)}$ if $n > 1$ and $t' = t$ if $n = 1$
$t \models \Diamond F$	iff	$t^{(i)} \models F$ for some $1 \leq i \leq n$
$t \models \Box F$	iff	$t^{(i)} \models F$ for all $1 \leq i \leq n$
$t \models F_1 \mathcal{U}_s F_2$	iff	$t^{(j)} \models F_2$ for some $1 \leq i \leq n$ and
		$t^{(i)} \models F_1$ for all $1 \leq i \leq j$
$t \models F_1 \mathcal{U}_w F_2$	iff	$t \models \Box F_1$ or $t \models F_1 \mathcal{U}_s F_2$

Table 6.3: Interpretation of future time LTL logic formulas

In Table 6.4 is an example of requirements for the dining philosophers in the KAOS language [48]. KAOS is a framework developed for goal-based modeling and reasoning [4]. The given requirement has an LTL-formula which states that it should be never possible to enter a state where four forks are held by four philosophers.

SafetyGoalAvoid$[FourHeldLeftForks]$
InformalDef
"Avoid four philosophers simultaneously holding their left fork"
FormalDef
$\forall p1, p2, p3, p4 : Philosopher,$
$f1, f2, f3, f4 : Fork$
$\Box\neg(HoldingLeft(p1, f1) \wedge$
$HoldingLeft(p2, f2) \wedge$
$HoldingLeft(p3, f3) \wedge$
$HoldingLeft(p4, f4))$

Table 6.4: An example KAOS requirement for dining philosophers.

6.5 Program Instrumentation

Instrumentation is a process where informative statements are inserted into program code for the purpose of monitoring [48]. When the actual program is executed, these informative statements provide information which can be interpreted by an observer. Instrumentation is one possibility to implement the data gathering mechanism introduced briefly in Figure 6.1.

Instrumentation of program code can be divided into static and dynamic counterparts, where static instrumentation refers to approaches where "hooks" are inserted into code before execution. Dynamic instrumentation happens during the execution of the actual program.

There are basically two kinds of static code instrumentation. The first one is the source code based approach where pre-compilers are used to inject event emitting code to necessary places in

source code. Source code based methods are limited to situations where source code is available, e.g. for testing purposes and very static environments.

The second kind of static instrumentation is done while loading binary programs into the system. Load time instrumentation of code can be implemented also for programs which we do not have source code and for that reason it is applicable to wider selection of programs and situations. Practically an interpreter-based execution platform (e.g. Java Virtual Machine [35] or .Net [15]) and higher abstraction level byte code programs are needed for load time instrumentation. Where static instrumentation happens before program execution, dynamic instrumentation may happen during the execution of a program. Dynamic instrumentation provides injection of new rules and constraints into the system under operation time. Especially in virtual organizations' environment this is beneficial as for example a company may change its policies while some business transactions and programs execution are still under operation. In Aspect Oriented Programming dynamic aspect weavers based on Java Debugging Interfaces have been used [44, 43, 50] to instrument code with join point hooks. Similar approach has been applied to runtime verification in the `jassda` framework [5].

6.6 Observers

Observers consume events emitted by data gathering system and perform trace based analysis of system operation. The data gathering system emits events from the execution platfrom. These actions are usually related to fine grained actions, such as function calls, communication or variable handling.

Specifications given to the monitor might be expressed using constructs and notations that are more abstract or otherwise differing from core events [47]. An observer must first somehow interpret the core events coming from the data gathering system using an interpretation function. An interpretation function is a mapping from core events of an execution platform to the concepts of specification language. These mappings are execution platform dependent and they must be given explicitly as an indistinguishable part of an environment. For example a mapping from platform specific notation to an action-based specification logic might express that a function call of the form $f(x, y, ...)$ corresponds an input action $in(x, y, ...)$ in the specification logic.

After the core application events are interpreted the observer can validate the conformance between system operation and specifications. For this purpose finite state machines derived from specifications are used [47, 14, 51, 26]. If an automaton accepts the trace constructed by application events then the application conforms to its specification, otherwise a violation of specification has happened. In the following we will shortly present the basic concepts and techniques used to generate observers for monitoring of specifications expressed in Linear Temporal Logic (LTL) [7].

A *tableau construction* method is usually applied to translate a temporal logic formula into a non-deterministic finite state automaton. The resulting automaton accepts precisely all the models (finite execution traces) of the formula [13, 21]. Traditional tableau-based translation methods to non-deterministic automata result in exponential blowup [52]. In practise, however, the resulting automata can be kept quite small for temporal formulas most frequently used in verification [54].

An efficient algorithm to produce a testing automaton for runtime verification was presented in [23]. It uses an approach similar to the one developed for on-the-fly model checking [22] which constructs a Büchi automata (automaton for infinite words) from an LTL formula; the algorithm is modified for finite words. Temporal logic used in [23] is LTL-X, a variant of LTL with no "next"-operator. Developed algorithm is used in JPaX framework which is a generic platform for the runtime analysis of Java programs [23, 25].

Automaton construction proceeds in two phases [23]. In the first phase a graph that presents the final automaton without final states is constructed. In the second phase of the algorithm final states are selected. Two-phase construction is necessary to guarantee satisfaction of eventualities (pending "until"-clauses) in finite trace semantics [23].

The core of the algorigthm is based on *expanding a graph node* [22, 23]. A graph node is a data structure which represents a state in the execution trace and includes the following fields [22, 23]:

NAME: A unique name for the node.

INCOMING: The set of nodes that lead to this node, i.e. which have incoming edges to this node.

NEW: The set of LTL formulae that must hold on the current state but have not yet been processed.

OLD: The set of LTL formulae that have already been processed. Each formula in NEW that gets processed is transferred to OLD.

NEXT: The set of LTL formulae that must hold at all immediate successors of this node.

The algorithm removes "proof obligations" from the NEW set until the set comes empty. The algorithm to translate a LTL-X formula ϕ starts with a single node which represents the initial state; its INCOMING set includes a dummy value `Init` to represent this fact. The initial edge has one obligation, namely the formula ϕ, and the sets OLD and NEXT are empty [22].

With a current node N the algorithm checks if there are obligations left in the set NEW. When NEW set is empty, the node is ready, i.e. it represents a state in the final automaton [22]. If the NEW set was not empty, a formula μ is extracted and removed from the set. If μ is an atomic proposition (p) or negation of a proposition ($\neg p$) and $\neg\mu$ is not in the set OLD, then μ is moved to the set OLD. Otherwise node contains a contradiction and it is discarded [22].

If the formula μ is a conjunction $\phi \wedge \psi$ then both ϕ and ψ are moved to the set NEW [22]. Otherwise the current node is split into two distinct nodes, such that for example for disjunction $\phi \vee \psi$, one node makes ϕ true and the other makes ψ true. For temporal operators obligations need to be pushed to the immediate successors, i.e. formulas will be added to the NEXT set [22, 23].

In the second phase of the automaton construction algorithm final states are selected for the automaton so that the satisfaction of *eventualities* is guaranteed. The algorithm designates only those states as accepting which do not contain a $\mathcal{U}$ formula in their NEXT state [23].

In [26, 27] observer generation algorithms based on rewriting logic [12, 37] were introduced. Worst-case complexity of algorithms is exponential but in practise performance was quite tolerable when using memorization optimizations [27]. The system was implemented in the Maude platform [9] and consisted of ca. 300 lines of code.

Alternating automata can be used to translate linear temporal logic formulas to automata with a linear size [52]. Three different algorithms, which use alternating automata for validating reactive system runtime behaviour against LTL, were presented in [19]. Another approach to decrease the complexity of the LTL translation operation is to use condinductive methods. The minimal deterministic automaton that accepts an LTL formula can be constructed using coinductive methods represented in [51].

6.7 Monitoring platform implementations

Several monitoring platforms and tools that can be used to monitor system conformance have been developed. In this section we will introduce some of these tools briefly.

Some methods and tools that could be used for Java-based load time instrumentation are binary component adaption (BCA) [32], the Java Object Instrumentation Environment (JOIE) [11, 29], the Byte Code Engineering Library (BCEL) [2] and JavaAssist [6]. For example in [48], JOIE is used for instrumenting Java programs with necessary statements to log necessary events in program execution.

jMonitor is an application library and runtime utility for monitoring Java programs [30]. Event generating code is bound to method calls of a binary Java object through byte-code instrumentation. Events can be attached to different points of the execution trace, such as method calls, field read and write operations or exceptions by definition of event patterns [30]. Event patterns use regular expressions for identifying a set of execution points which can be then associated to program code with *doBefore*, *doAfter* and *doInstead* methods. Each event pattern is associated with zero or more monitors which get called when the corresponding event happens.

ReqMon is a monitoring system implemented for requirements monitoring purposes [48]. ReqMon the uses Joie framework to instrument Java class files with event-emitting code. Requirements are specified formally using linear temporal logic (LTL). The validation of requirements is done with the Spin model checker [28]. ReqMon uses so called "suspect conditions" to find suspicious behaviour before a requirement violation happens. Conceptually suspect conditions are weakened conditions of requirement violations, such that for example for requirement "Avoid four philosophers simultaneously holding their left fork" migth have a suspect condition "Three philosophers simultaneously hold their left fork" [48]. If a suspect condition is met, a notification is usually given.

Temporal Rover is a commercial tool for runtime monitoring of applications written in C/C++, Java, Verilog or VHDL [16]. Temporal Rover can verify program conformance over past or future time temporal logics, LTL or Metric Temporal Logic (MTL) but it requires the user to manually instrument the program code. Instrumentation instructions are given in source code comments from where a pre-compiler extracts the specifications and generates the necessary monitoring code [16]. Temporal Rover parser converts an instrumented source code into a new file which includes the necessary code for runtime verification.

Java Path Explorer (JPaX) is a runtime verification tool for validating conformance between Java programs and temporal logic specifications [27]. In addition to temporal logic verification JPaX also provides functionality to analyse concurrency errors from programs, such as deadlocks or data races. Code instrumentation is done automatically in byte-code level. Verification functionality in Java Path Explorer is implemented by using the Maude rewriting framework [8].

Maude is a high-level language and system supporting executable specification and declarative programming in rewriting logic [8]. Rewriting logic [12] provides a general, flexible semantic framework for concurrency [38] which can be used to describe different logical characterisations of concurrency features. JPaX uses Maude both as an event driven monitoring process and as a tool to produce observer automata from temporal specifications [27].

A framework for monitoring service oriented systems described in [36]. The monitoring system uses event calculus as its semantic framework to describe behavioural properties and assumptions. Behavioural properties are initially extracted from BPEL4WS [41] processes and they describe properties of the composition process. Assumptions are defined by system providers and they describe additional properties for behaviour of the system, the agents in its environment or individual services [36]. Events are generated by the process execution engine, thus no instrumentation is needed.

A Business Contract Architecture (BCA) and a Business Contract Language (BCL) for contract based communities and their monitoring was described in [39, 40]. BCA is basically an event based monitoring platform and BCL is a contract language which essentially includes be-

havioural descriptions and temporal contraints that must be monitored during the operation of a community [40].

BCL defines the structure, roles, behavioural interactions and policies for a community [40]. Behavioural interactions describe the allowable ordering of actions between participants in a community. Interactions are described with event patterns. Policies are deontic statements, inspired by deontic logic, a branch of modal logic developed for reasoning with notions of obligation, permissions, prohibitions, authority and so on [39]. From the deontic statements it is possible to derive operational constraints that must be satisfied by the participants of a community.

The main purpose of the Business Contract Architecture (BCA) is to execute BCL contracts and monitor that contract policies are not violated [40]. The executing entity in the architecture is called a Business Monitoring Activity (BAM) engine, basically a rule-based, reactive process execution platform. An Interceptor component is located between a participiant and the rest of the community and provides non-intrusive interception of business messages exchanged between trading partners [40]. The Interceptor creates business event notifications for BAM and the Contract Monitor, which performs evaluation of contract policies.

6.8 Conclusions

Inherent properties of virtual organizations such as heterogeneity of technological platforms, autonomicity of participiants and dynamicity of both service and collaboration network evolution make it necessary to monitor the operation of a community. By monitoring we mean procedures that are used for the extraction of necessary information from the community describing its operation and behaviour, validation of conformance between current community operation and the expected behaviour, and mechanisms to react to possible behavioural violations.

In this chapter we introduced the concepts needed for creation of virtual organization monitoring platforms. Section 6.1 described traditional monitoring approaches used for software testing and requirements engineering.

Monitoring as a tool for software testing concentrates in the formal verification of an software artifact: usually a specification in temporal logic is given and the program code is instrumented to emit necessary events. Especially methods of deriving test oracles from temporal specifications and mechanisms for code instrumentation developed for softare testing purposes can be applied quite straightforwardly to virtual organizations' context.

Requirements monitoring concentrates on the supervision of the execution environment. When requirements and assumptions are specified and given to a monitor the program can be notified when its environment has been changed into a state which violates the program's expectations. Methods and mechanisms usable in the context of virtual organizations are: 1) the notion of a reflective feedback loop between a program and its environment, and 2) compensatory actions or processes when a requirement has been violated.

In Section 6.2 we identified the basic concepts found in general monitoring systems. Formal specifications are needed to describe behavioural patterns that must be followed by a program or a community. A data gathering mechanism must be present in every platform where monitoring is going to be implemented. This mechanism can be implemented using program code instrumentation or messaging interception. A monitor itself consists of an observer and an analyser. The observer implements behavioural conformance validation between a program execution trace and program specification whereas an option analyser can be also present for more complex analysis, such as deadlock, data race or trust analysis. An event handler and logging mechanism is used to propagate results of the monitoring system to e.g. administrators, analysists or to the execution

platform itself.

A general introduction to runtime conformance validation was given in Section 6.3. An automata theoretic approach to verification of temporal properties was introduced briefly through an example of log-based analysis. In Section 6.4 behavioural specifications were discussed briefly. The specification language choosed as an example was Linear Temporal Logic (LTL) due to its prevalence and applicability for the verification of finite execution traces.

Program instrumentation, mainly the instrumentation of Java based programs, was shortly discussed in Section 6.5. Two basic instrumentation tactics were identified, namely static and dynamic instrumentation. Static instrumentation takes places before program execution and is implemented usually in source code in the pre-processing phase or during the byte code loading phase. Dynamic instrumentation is done during the execution of the program. Techniques used for dynamic weaving in Aspect Oriented Programming are also usable for monitoring purposes.

Observer generation and its basic functional responsibilities were discussed more deeply in Section 6.6. Observers must first interpret the core application events to concepts used in the specification language. For this purpose an interpretation, or mapping function must be provided. After the interpretation of events the observer can validate the behavioural conformance of program operation to its specification. For this purpose an automaton which accepts only conformant execution traces must be provided. For this purpose so called tableau methods are usually used. A very brief description of a tableau algorithm was given.

In Section 6.7 some tools and platforms that can be used for implementing a virtual organization monitoring platform were introduced.

Bibliography

[1] ANDREWS, J. H., AND ZHANG, Y. Broad-spectrum studies of log file analysis. In *Proceedings of the 22nd international conference on Software engineering* (2000), ACM Press, pp. 105–114.

[2] APACHE SOFTWARE FOUNDATION. The Byte Code Engineering Library. `http://jakarta.apache.org/bcel`, 2003.

[3] BARRINGER, H., GOLDBERG, A., HAVELUND, K., AND SEN, K. Program monitoring with LTL in EAGLE. In *18th International Parallel and Distributed Processing Symposium* (Apr. 2004), IEEE, pp. 264–271.

[4] BERTRAND, P., DARIMONT, R., DELOR, E., MASSONET, P., AND VAN LAMSWEERDE, A. GRAIL/KAOS: an environment for goal driven requirements engineering. In *ICSE'98 - 20th International Conference on Software Engineering* (Apr. 1998), IEEE-ACM.

[5] BRÖRKENS, M., AND MÖLLER, M. Dynamic Event Generation for Runtime Checking using the JDI. *Electronic Notes in Theoretical Computer Science 70*, 4 (Dec. 2002), 1–15.

[6] CHIBA, S. Load-Time Structural Reflection in Java. *Lecture Notes in Computer Science 1850* (2000), 313–.

[7] CLARKE, E. M., GRUMBERG, O., AND PELED, P. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[8] CLAVEL, M., DURÁN, F., EKER, S., LINCOLN, P., MARTÍ-OLIET, N., MESEGUER, J., AND QUESADA, J. F. The Maude system. *Lecture Notes in Computer Science 1631* (1999), 240–.

[9] CLAVEL, M., DURÁN, F., EKER, S., LINCOLN, P., MARTÍ-OLIET, N., MESEGUER, J., AND QUESADA, J. F. Maude: specification and programming in rewriting logic. *Theor. Comput. Sci. 285*, 2 (2002), 187–243.

[10] COHEN, D., FEATHER, M. S., NARAYANASWAMY, K., AND FICKAS, S. S. Automatic monitoring of software requirements. In *Proceedings of the 19th international conference on Software engineering* (1997), ACM Press, pp. 602–603.

[11] COHEN, G., CHASE, J., AND KAMINSKY, D. Automatic Program Transformation with JOIE. In *1998 USENIX Annual Technical Symposium* (1998), pp. 167–178.

[12] DERSHOWITZ, N., AND JOUANNAUD, J.-P. *Rewrite systems*. MIT Press, 1990, pp. 243–320.

[13] DILLON, L. K., AND RAMAKRISHNA, Y. S. Generating oracles from your favorite temporal logic specifications. In *Proceedings of the 4th ACM SIGSOFT symposium on Foundations of software engineering* (1996), ACM Press, pp. 106–117.

[14] DILLON, L. K., AND YU, Q. Oracles for checking temporal properties of concurrent systems. In *Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering* (1994), ACM Press, pp. 140–153.

[15] Microsoft .Net information web-page. `http://www.microsoft.com/net/default.asp`.

[16] DRUSINSKY, D. The Temporal Rover and ATG Rover. In *7th SPIN Workshop* (2000), vol. 1885 of *Lecture Notes in Computer Science*, pp. 323–330.

[17] FEATHER, M. S., FICKAS, S., VAN LAMSWEERDE, A., AND PONSARD, C. Reconciling System Requirements and Runtime Behavior. In *Proceedings of the 9th International Workshop on Software Specification and Design* (1998), pp. 50–59.

[18] FICKAS, S., AND FEATHER, M. S. Requirements monitoring in dynamic environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (1995), IEEE Computer Society, p. 140.

[19] FINKBEINER, B., AND SIPMA, H. Checking Finite Traces Using Alternating Automata. *Formal Methods in System Design 24*, 2 (Mar. 2004), 101–127.

[20] GATES, A. Q., MONDROGON, O., PAYNE, M., AND ROACH, S. Instrumentation of intermediate code for runtime verification. In *28th annual NASA Goddard Software Engineering Workshop (SEW'03)* (Dec. 2003), IEEE, pp. 66–71.

[21] GEILEN, M. On the Construction of Monitors for Temporal Logic Properties. In *RV'01, Runtime Verification* (July 2001), K. Havelund and G. Roşu, Eds., vol. 55 of *Electronic Notes in Theoretical Computer Science*, Elsevier.

[22] GERTH, R., PELED, D., VARDI, M. Y., AND WOLPER, P. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV* (1996), Chapman & Hall, Ltd., pp. 3–18.

[23] GIANNAKOPOULOU, D., AND HAVELUND, K. Automata-Based Verification of Temporal Properties on Running Programs. In *ASE '01: Proceedings of the 16th IEEE International Conference on Automated Software Engineering* (2001), IEEE Computer Society, p. 412.

[24] GIRGENSOHN, A., REDMILES, D. F., AND SHIPMAN, F. M. Agent-Based Support for Communication Between Developers and Users in Software Design. In *9th Knowledge-Based Software Engineering Conference* (Sept. 1994), IEEE Computer Society Press, pp. 22–30.

[25] HAVELUND, K., AND ROSU, G. Java PathExplorer — A runtime verification tool. In *Proceedings 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, ISAIRAS'01, Montreal, Canada, June 18–22, 2001.* (2001).

[26] HAVELUND, K., AND ROSU, G. Synthesizing Monitors for Safety Properties. In *Tools and Algorithms for Construction and Analysis of Systems* (2002), vol. 2280 of *Lecture Notes in Computer Science*, pp. 342–356.

[27] HAVELUND, K., AND ROSU, G. An overview of the runtime verification tool Java PathExplorer: Special issue on selected papers from the First International Workshop on Runtime Verification Held in Paris, July 2001 (RV01). *Formal Methods in System Design 24*, 2 (Mar. 2004), 189–215.

[28] HOLZMANN, G. J. The Model Checker SPIN. *IEEE Transactions on Software Engineering 23*, 5 (May 1997), 279–295.

[29] INTERNET SYSTEMS AND STORAGE GROUP, DUKE UNIVERSITY. The Java Object Instrumentation Environment. http://www.cs.duke.edu/ari/joie/, 2003.

[30] KARAORMAN, M., AND FREEMAN, J. jMonitor: Java runtime event specification and monitoring library. In *RV'04, Fourth Workshop on Runtime Verification* (2004), Electronic Notes in Theoretical Computer Science, Elsevier.

[31] KAVANTZAS, N., BURDETT, D., AND ET AL., G. R. *Web Services Choreography Description language*. W3C, Oct. 2004. http://www.w3.org/TR/2004/WD-ws-cdl-10-20041012/, Working draft.

[32] KELLER, R., AND HÖLZLE, U. Binary Component Adaption. In *ECOOP'98 – Object-Oriented Programming* (1998), vol. 1445 of *LNCS*, Springer-Verlag Heidelberg, pp. 307–329.

[33] KINDLER, E., AND WEBER, M. The Petri Net Kernel: An infrastructure for building Petri net tools. *International Journal on Software Tools for Technology Transfer (STTT) 3*, 4 (Sept. 2001), 486–497.

[34] KUNZ, T., TAYLOR, D. J., AND BLACK, J. P. Poet: Target-System-Independent Visualizations of Complex Distributed Executions. In *Proceedings of the 30th Hawaii International Conference on System Sciences* (1997), IEEE Computer Society, p. 452.

[35] LINDHOLM, T., AND YELLIN, F. *The Java Virtual Machine Specification*, 2 ed. Sun Microsystems, Aug. 1999.

[36] MAHBUB, K., AND SPANOUDAKIS, G. A framework for requirents monitoring of service based systems. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing* (2004), ACM Press, pp. 84–93.

[37] MARTÍ-OLIET, N., AND MESEGUER, J. Rewriting logic: roadmap and bibliography. *Theor. Comput. Sci. 285*, 2 (2002), 121–154.

[38] MESEGEUR, J., AND ROSU, G. Rewriting Logic Semantics: From Language Specifications to Formal Analysis Tools. In *Automated Reasoning: Second International Joint Conference, IJCAR 2004* (2004), D. Basin and M. Rusinowitch, Eds., Lecture Notes in Computer Science, Springer-Verlag, pp. 1–44.

[39] MILOSEVIC, Z., AND DROMEY, R. G. On Expressing and Monitoring Behaviour in Contracts. In *Proceedings of the Sixth International ENTERPRISE DISTRIBUTED OBJECT COMPUTING Conference (EDOC'02)* (2002), IEEE Computer Society, p. 3.

[40] MILOSEVIC, Z., GIBSON, S., LININGTON, P. F., COLE, J., AND KULKARNI., S. On Design and Implementation of a Contract Monitoring Facility. In *Proceedings of the First IEEE International Workshop on Electronic Contracting (WEC'04)* (2004), IEEE Computer Society, pp. 62–70.

[41] OASIS. *Business Process Execution Language for Web Services*, May 2003. http://www.ibm.com/developerworks/library/ws-bpel/.

[42] PETERSON, J. L. *Petri Net Theory and the Modeling of Systems*, 1 ed. Prentice Hall, Englewood Cliffs, 1981.

[43] POPOVICI, A., ALONSO, G., AND GROSS, T. Just-in-time aspects: efficient dynamic weaving for Java. In *Proceedings of the 2nd international conference on Aspect-oriented software development* (2003), ACM Press, pp. 100–109.

[44] POPOVICI, A., GROSS, T., AND ALONSO, G. Dynamic weaving for aspect-oriented programming. In *Proceedings of the 1st international conference on Aspect-oriented software development* (2002), ACM Press, pp. 141–147.

[45] PRATT, V. R. A practical decision method for propositional dynamic logic (Preliminary Report). In *Proceedings of the tenth annual ACM symposium on Theory of computing* (1978), ACM Press, pp. 326–337.

[46] RICHARDSON, D. J. TAOS: Testing with Analysis and Oracle Support. In *Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis* (1994), ACM Press, pp. 138–153.

[47] RICHARDSON, D. J., AHA, S. L., AND O'MALLEY, T. O. Specification-based test oracles for reactive systems. In *Proceedings of the 14th international conference on Software engineering* (1992), ACM Press, pp. 105–118.

[48] ROBINSON, W. Monitoring Software Requirements Using Instrumented Code. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9* (2002), IEEE Computer Society, p. 276.2.

[49] SADDEK BENSALEM, MARIUS BOZGA, M. K., AND TRIPAKIS, S. Testing conformance of real-time software by automatic generation of observers. In *RV'04, Fourth Workshop on Runtime Verification* (2004), Electronic Notes in Theoretical Computer Science, Elsevier.

[50] SATO, Y., CHIBA, S., AND TATSUBORI, M. A selective, just-in-time aspect weaver. In *Proceedings of the second international conference on Generative programming and component engineering* (2003), Springer-Verlag New York, Inc., pp. 189–208.

[51] SEN, K., ROSU, G., AND AGHA, G. Generating Optimal Linear Temporal Logic Monitors by Coinduction, 2003.

[52] VARDI, M. Alternating automata: Checking truth and validity for temporal logics. In *14th intl. Concerence on Automated Deduction* (1997), vol. 1249 of *LNCS*, Springer-Verlag.

[53] W3C. *Web Service Choreography Interface (WSCI)*, Aug. 2002. `http://www.w3.org/TR/wsci/`, W3C Note.

[54] WOLPER, P. *Constructing automata from temporal logic formulas: a tutorial*. Springer-Verlag New York, Inc., 2002, pp. 261–277.

Chapter 7

Contract management

Anna-Kristiina Ritola

When organizations do business together they need to agree on the nature of their business and bind themselves to it. That is why contracts are needed to base a legally enforceable agreement between the parties. Contract management is then needed to make sure that the parties fulfill the contract during the whole life cycle of it. Contract management can be seen both as a proactive and reactive process [9].

Traditional contract management can be seen as consisting of the issues concerning service delivery management, relationship management and contract administration [9]. The purpose of *service delivery management* is to take care that the exchange between the parties is handled as promised and the quality and performance corresponds to the agreed values. *Relationships between the parties* are managed to prevent problems or solve them if they arise. The last one, *contract administration*, is required to deal with the governance issues and change management.

In the area of virtual organizations and B2B the contract management issues concentrate on automating the contract management. The design of architectures and mechanisms that support the whole life cycle of the contract is important and widely studied. In addition to the need for the contract management systems there is a need for standard contract templates [4].

In this chapter the contract management is surveyed. First in Section 7.1 the contracts are discussed. After that the life cycle of a contract is observed in Section 7.2. The mechanisms for contract management are handled in Section 7.3, after which contract management architectures are discussed in Section 7.4.

7.1 Contracts

Standardizing the contracts brings lots of benefits. Standardizing means saves in money and trouble, when contracts or templates can be reused [4]. It also increases the trust among the partners. In addition to standardizing the contracts the contract clauses can be standardized as well. From the contract management's point of view standardizing is important, because the contracts and their elements and the clauses can be used as components in B2B systems. A good contract, covering all the issues needed, forms the basis for the success of contract management [9].

Contracts are needed to establish an agreement between the parties to form a basis for their business. According to [4] there are four elements which form a valid contract: 1. agreement, 2. consideration, 3. capacity (or competence) and 4. legal purpose. There needs to be an agreement between the parties about the terms and conditions of the contract. Consideration is used to express why the contract is made and what the exchange between the parties is. In the contract this means

money, services, property or individual rights. Also, capacity (or competence) expresses that the contract is signed by the representatives of parties both being authorized by their organizations. The last one, legal purpose, refers to the need of legality of the contract, because the contracts must be legal to make contracts enforceable.

The contents of the contract are expressed by clauses. By means of contract clauses the behaviour norms agreed are expressed [2]. There are three different types of behaviour norms: obligation, permission and prohibition. Obligation refers to an action obliged, permission to an action permitted and prohibition to an action prohibited to execute at a certain time. To ensure the effectiveness of obligations sanctions need to be enclosed. The relationship between the clauses and the elements of validity is that the four elements must be visible in the clauses of the contract [4].

The behaviour obligations and rights belonging to the roles of the parties can be described as normative statements. This can be done in the following way, which is based on the deontic logic. Normative statements are the following kind of statements [1]:

"If certain condition holds then subject is {obliged, permitted, prohibited} {to/by} beneficiary party to achieve certain action or the state-of-affair to bring about {before/until} deadline holds true."

In expressing norms and normative statements of the contract Deontic Logic is typically used [1, 2]. Deontic Logic is proposional logic in which is added obligations, permissions and forbiddance to express the task allocation and process coordination [10]. Anyway, the Deontic Logic is not discussed more in this text.

The norms in the contract can be seen as a hierarchy of institutional, constitutional and operational norms [2]. The purpose of this normative framework is to provide background for contract validation, monitoring and enforcing. Institutional norms form the basis for the contracts by means of law. The institutional norms describe the regulations and default values for circumstances that are not explicitely expressed in the contract. Contract templates can be predefined with institutional norms, too. The cooperation between the parties is expressed using constitutional norms. Lastly, the operational norms describe the implementation of the constitutional norms. In Figure 7.1 it is shown how constitutional norms are validated against the institutional norms, when contracts are formed. When contracts are excuted the fulfillment of the contract can be monitored by examining the operational norms against the constitutional norms.

The contract must be unambiguous [8] and the definition and interpretation of the terms used in the contract must be provided [4]. Also, the contract must be uniquely identifiable [1], because there might be many contracts, even nested and overlapping ones, executed at the same time [7]. For the contract management's needs also the definition of the QoS parameters and the means of measuring them must be provided [6].

In the contract the conditions for entering and leaving the virtual organization must be expressed [2]. The parties may want to leave because of their own will or they might be expelled. In the case of parties leaving earlier than the contract comes to an end there is a need for expressing the possible indemnities in such situations.

According to [2] the simple form contracts describing a one time relationship between the parties have been most addressed and the contracts concerning more complex business relationships are little studied. In [6] the relationships between the parties are divided into chains of relationships. When it comes to contract this means that contracts can be seen as bilateral, which removes the need for complex multi-party contracts.

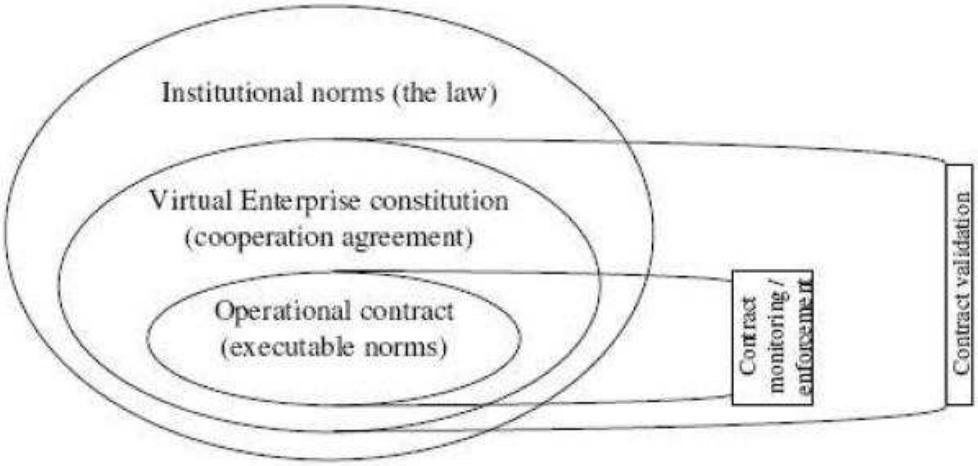


Figure 7.1: Normative framework [2].

7.2 Contract life cycle

The contract's existence from the beginning to the end can be described as a life cycle model. The life cycle of a contract can be seen as a process which consists of three phases [1]. 1. In the *drafting* phase a contract is drafted from a contract template. 2. In the *contract formation* phase the draft is made a complete contract. 3. In the *contract fulfillment* phase consideration is carried out. In this phase the operation of the parties are monitored to see if the terms and the conditions of the contract are accomplished. The lifecycle of the contracts can also be divided into two phases: 1. *contract establishment* and 2. *contract execution* [8]. To provide support for the contract management, some mechanisms have been presented.

The following mechanisms support the *contract establishment* phase [8]:

Storing contract templates promotes the reuse of contracts and the use of contracts of a standard form in the drafting phase. The elements of the contracts, such as contract clauses, can be of the standard form to promote automation, too.

Contract validity checking ensures the validity of a contract. The institutional norms are the ones against which the contract is validated [2]. Respectively, the operarational norms can be validated against the agreed contract.

Negotiation mechanisms are needed to reach an agreement of the contract among parties. There are two ways of reaching the agreement. The options are to use a predefined contract template by a third party or negotiate the structure and the contents of the contract among the parties. Using contract templates forms a basis for negotiation and enables approaches to automate negotiation [2].

Storing contracts that have been signed is important, because it is the basis for the contract management. In the simplest case the expiration of the contract can be notified.

Next, the following mechanims are seen to support the *contract execution* phase:

Contract monitoring makes it possible to compare if the parties are following the behaviour

agreed in the contract or not in the execution phase. The behaviour of the parties can be monitored so that if significant events which indicate the violation of the contract are noticed, actions are taken.

Contract notifications can be used in various ways. Notifications can be given to parties which have been detected or are suspected to violate the contract. Notifications can also be given as reminders. Also, notifications can be used to report the execution of a certain action to the parties interested in knowing of it [7].

Contract mediation intends to find a resolution in a dispute between the parties. If it can be reached the result may be an amended contract.

Contract arbitration tries to find a resolution in a dispute with the help of an arbitrator. There are approaches to use agents and the information provided by reputation systems or such as arbitrators.

Lastly, the following ones are to support *trust* during the whole life cycle of the contract:

Reputation systems can be used to collect and distribute feedback of the behaviour of the parties.

Insurance services can increase the trust between the parties. This means third parties to insure the risks of the business.

In addition to the previous mechanisms *Registration* and *Brokering* mechanisms can be seen supporting the very beginning of the contract life cycle [2]. In these mechanisms it is expected that there exists some kind of breeding environment to enable the partner matching. It is sometimes referred to as a pre-contractual phase.

7.3 Contract management

For the contract management's needs the purpose of a contract is to direct how and when the fulfillment of the contract should be verified [2]. The fulfillment of the contract is checked by monitoring the operational norms. The contract management can be handled by all the parties, one of them or it can be delegated to the third parties [6].

7.3.1 Contract performance monitoring

The contract performance monitoring deals with the issues concerning what the parties should be doing according to the contract at a certain time, are they following the behaviour predescribed and if not, are there any mechanisms to correct the execution and find a solution [3]. To find out the compliance to the performance agreed in the contract requires extracting the workflow information from the contract and comparing them to the actual workflows. The importance of being aware of the contract performance is that the parties need some assurance and checking that the operation corresponds to the promised [7]. Most of the research concentrates on detecting a violation which has already happened and requires contract enforcement which is discussed later. Anyway, the need for pro-actively detect the violation or threat of it has been adressed in [10] so that parties could be warned or reminded beforehand.

The contract can be used to identify the obligations of the parties. It can also be used in the scheduling of actions or identifying situations of violation which need responses [7]. By means of a stored contract the parties are able to know which are the actions permitted and obliged at a certain time. There is also a need for knowing the required actions following a certain time and the time allowed to accomplish them. The consequences of failure and non-compliance should also be observable.

In [7] the basis for the contract monitoring is to recognize event patterns. The correct behaviour leads to a final state. It can also recognize some midstages of importance. The wrong behaviour might be failures or events in the wrong context of execution. These event patterns not corresponding to the contract are signaled and the required actions according to situation are enclosed.

The mismatch between the monitored and the agreed behaviour defined in the contract may result from the different levels of abstraction caused by the parties' different ways of performing actions or delegation [7]. That is why it is important to determine the dynamic binding of the behaviour of the parties in contract execution already in the contract negotiation phase to prevent problems in contract monitoring. The solutions suggested to determine the binding were pre-registration and inspection of parameterisation of initial exchanges.

If a violation to a contract is detected, sanctions defined in the contract can be applied [2]. There is a need for a third party to bring trust and to pass a sentence because of the fact that the parties will not voluntarily comply with the sanctions. In [6] the need for determination of the source of violation is addressed as well as the need for initiating corrective actions.

The importance of timing in contract monitoring is addressed in [7]. The distributed nature of contract execution brings problems for the reconstruction of the sequence of events because of transmission delays and various local clocks. Using time signaled by a third party prevents frauds committed by manipulation of clocks or transmission delays.

7.3.2 Contract enforcement

A contract is said to be enforceable if it is possible to verify the compliance to the contract by examining the actions of the parties [2]. The starting point for the contract management is the idea that there is always non-compliance to the contracts and that is why contract monitoring and contract enforcement is needed [8]. The way in which the contract enforcement is done is based on the detection of violation, after which enforcement mechanisms can be applied to reach for the compliance to the contract.

In real life the legal system has the power to enforce the contracts, but in the B2B area the slowliness, costs and applicability are seen as problematic [8]. The mechanisms that try to avoid falling into using the legal system when disputes arise are called alternative dispute resolution mechanisms (ADR). These ADR mechanisms try to solve the disputes without going to court.

The starting point for contract enforcement in [3] is that both parties have a subjective opinion of their compliance to the contract. In case of a dispute the opinions of the parties are different and resolution is needed. A judge of some form is needed to resolve the dispute based on the evidences.

In [8] contract enforcement is used as a definition for the mechanisms attempting to ensure the compliance of the parties to the contract. These mechanisms include among others mediation and dispute resolution. The following mechanisms are described as discretionary, because they are used to solve the violation or dispute which has already arisen, but do not try to prevent it from happening, in which case they would be non-discretionary or pro-active. Anyway, according to [8] the use of contract enforcement process has many benefits. In addition to having a defined process to handle with the violations and disputes, it can be seen as pro-active in that sence that the awareness of such a system itself may prevent from intentional deviation and violation of the contract.

The contract enforcement process is described as a model which contains 5 levels [8]. The purpose of the levels is to describe if the contract execution is consistent with the contract. The first level symbolizes the fulfillment of the contract. Despite of the fulfillment of the contract devi-

ations may happen as long as due to compliance to the notifications received the parties violating the contract correct their behaviour. In the second level notifications have been ignored and the execution is not consistent to the agreed. If the solution and an amended contract is found with the help of a Mediator or Negotiator, the execution returns back to the first level. In the third level the mediation upon the amended contract did not work and an Arbitrator is needed to look for a solution. If found, the execution can return back to the first level. In the fourth level penalties are given by the Discretionary Enforcement Moderator (DEM) consisting of two roles, Mediator and Arbitrator, because the help of Arbitrator did not solve the dispute. In the fifth level even the DEM was not able solve the situation, which means that the contract execution is over and the dispute might continue outside the contract management system, meaning the traditional legal systems. The contract enforcement process described above is presented in Figure 7.2.

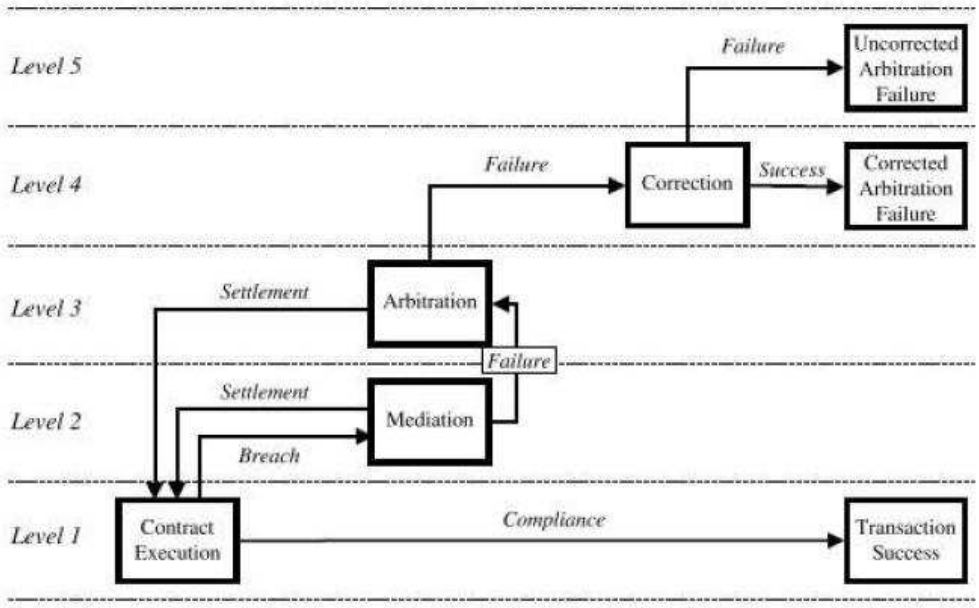


Figure 7.2: Contract enforcement process [8].

7.3.3 Resolution

In the Workflow-Contract-Solution model (WCS) a problem solution process is described. The problem solution process is undertaken to get over the contract violation or exception, which may be an action or inaction of the agreed workflow process [5]. The first stage of the problem solution process is the *Problem Report Stage* (1) in which the problem is reported by a party in the form of a statement describing the problem. The simplest problems may terminate at this stage if a solution is found. The second one is a *Situation Analysis Stage* (2) in which the reporter of the problem gives evidence. The evidence given are formalized as situation and contract processes. The counter partner may reply with his own evidence. If the situation is agreed, the process continues to the *Solution Discussion Stage* (3) in which the solution is seeked using abstract processes and semantic links to the workflow processes. Justification to the solution is done with the rights and obligations processes. If a solution was found the process passes to the forth and final *Solution Execution*

Stage (4). If the solution was noticed to be improper, the problem solution may start over from the beginning. Also, if the problem escalates, it falls outside the WCS system and the dispute might continue in the traditional legal system. This may happen if the situation is not agreed on at the situation analysis stage, no solution is found at the stage of Solution Discussion or lastly at the stage of Solution Execution.

7.4 Contract management architectures

7.4.1 Role-based

A role-based architecture has been proposed to the management of contracts supporting the whole life cycle of a contract [8]. The architecture is reasonably similar to the architectures described in [1, 4]. The components for the *contract establishment* are: Negotiator, Validator, Notary and Contract Forms Repository. The contract templates and contract clauses are stored in a Contract Forms Repository. The Negotiator can be used in contract negotiation enabling making offers and counter-offers between the parties. The Validator is used to ensure the validity of the contract, after which an established, valid contract is stored in a Notary.

The components for the *contract execution* are: Monitor, Notifier, Enforcer and Discretionary Enforcement Moderator (DEM), which consists of two roles, Mediator and Arbitrator. The Monitor monitors the activities of the parties and compares them to the agreed values of the contract stored in the Notary. If the Monitor detects a violation, it can signal the Notifier to send notifications to the parties or signal the DEM, if mediation or arbitration is needed. The DEM makes reasoning of the performance of the parties and may pose the roles of a Mediator or an Arbitrator. The Enforcer is responsible for the enforcing actions ensuring the compliance to the contract. The Enforcer may also give penalties to the parties.

The components for the *trust establishment* are: Guarantor, Advisor, Feedback Collection and Reputation Rating Centre (FCRRC) and Intermediary. The Guarantor assures the risks of a non-compliance of a party. The Guarantor may be for instance a credit company. The Advisor gives an opinion of the reliability of the party. One example of an Advisor is a Reputation System. An Intermediary is a party intervening in the contractual relationship. An Intermediary party may bring assistance in establishing business relationship or contract negotiation. An example of an Intermediary is an information portal.

In another paper [6] the proposed architecture contains three roles of the Management Service Providers (MSPs): measurement, violation detection and management services. The measurement services provide functions to measure the relevant metrics of the contract execution which can be done by probing or interception of client invocations. Violation detection services compares the measures to the agreed ones in the contract each time a new value is avalailable or periodically. The management services determine and resolute the problems.

The contract management system [6] contains the following components: Contract Repository, Measurement, Violatation Detection, Management, Business Entity and Deployment. The Contract Repository stores all the contracts of an organization. The Measurement component collects measures of contract execution. Violation detection compares the collected measures against the agreed values in the contract and if violatation is detected it notifies the management component. When the management component is notified, it asks for advice from the business entity componet for corrective actios and triggers the appropriate actions. The Business entity makes the decisions of corrective actions. Human involvement may be included in reasoning. The last component, called Deployment, is for setting up the architecture. In this architecture the Measurement component corresponds to the Monitor component and the cooperation of Management and the

Business Entity corresponds to the cooperation of DEM and Notary described before.

Another monitorable contract model has been presented as well to manage the contracts and the business execution [10]. The difference to the models presented above is that this model supports in addition to the detection of contract violation also the pro-active detection of threat of violation and possibly forthcoming one. The contract model enables finding out the expected actions of the parties following a certain state of contract execution by using guards of contract constraints. Based on the former, it can also detect if the contract violation is imminent and parties should be warned or reminded by using the guards of the contract constraints and the deadline of the action. Lastly, the parties responsible for the contract violation can be solved by using a commitment graph.

An action is the most important element of the monitorable contract needed in the architecture [10]. It is defined so that an action has a name, sender, receiver and deadline to be performed. Commitments are used to express consideration as causal relationships between the parties. A commitment is defined as having a name, receiver and a series of actions and their attributes. A contract constraint is used to specify the occurrence order of the actions. Lastly, a guard of the contract constraints is used to define the state of the process execution. This is done so that the guard for a certain action defines which actions must have occured, which have not yet, which are and which are never expected to happen.

Typically the elements of the contract such as normative statements are defined using Deontic Logic. In this approach the constraints and guards are implemented using Temporal Logic [10]. The difference to the former is that by means of Temporal Logic the status of the occurrence of an action can be find out, too. In Temporal Logic the truth and falseness of an proposition depends on time.

In [10] the monitorable contract model for multiparty contracts contains two components, the Monitorable Element (ME) and the Monitoring Mechanism (MM). In the Monitorable Element (ME) the trading process is described using actions and commitments of the parties to form a contract. In this component the relationship between the actions is described by the contract constraints and the right order of the actions is specified by guards, too.

In the Monitoring Mechanism (MM) [10] there are two components to provide the contract monitoring. The monitoring module is used to monitor the contracts by using commitment graphs and two algorithms for guarding the actions and pro-active detection. The reactive module is used to remind, warn, trace and compensate the actions. Each commitment can be expressed as a commitment graph which defines the relationship among the commitment. The actions form the edges of the commitment graph. The code for the edge is formed of the commitment which the action belongs to and of the sequence of the action concerning the commitment. The parties form the nodes of the commitment graph. The use of commitment graphs is to help identifying the parties violating the contract especially in multiparty contracts. The guards are used to monitor the contract and keep track of the execution. Because of guards it is possible to inform the parties by means of reminding and warning messages.

7.4.2 Agent-based

In multi-agent technology approaches each agent represents the enterprise it belongs to [2]. All the agents are autonomous even though they cooperarate for the whole virtual organization. The agent-based architectures describe well the distributed nature of virtual organizations containing autonomous enterprises. The cooperation of agents is needed for the coordination and distributed problem solving.

In [3] is described an agent-based architecture for monitoring the contract execution and rea-

soning the possible deviation. The controllers (agents) are used to collect evidence and Subjective Logic is used in the evidence-based reasoning. In the proposed (e-market) architecture the e-market controller is aware of the obligations of the contract. That's why the controller and the agents of the parties co-operating with it also know the expected execution defined in the contract. The actions of the parties are logged and the compliance to the contract can be monitored. If a dispute arises, the agents of both parties give their opinion of the situation. The controller represents a judge to form a resolution from the parties' views. The controller agent can also use second hand information sources such as certification authorities or other controllers to help in the resolution of disputes.

The opinions are presented using Subjective Logic in which a measurable belief of the truth or falsity of an opinion is formed when no certain information is available [3]. In the case of contract monitoring the opinions are formed of the compliance to the contract, which can be for example occurences of events. Subjective Logic is based on classical logic and on a theory of subjective probabilities.

Subjective Logic can be used in describing the subjective opinions of the agents about a proposition [3]. The opinions of the agents consist of the composition of three values. The first value measures the belief that the proposition is true. The second value measures the disbelief of the proposition to be false. Lastly, the third value measures the uncertainty of the proposition to be either true or false. There are many ways of calculating opinions in different situations, but they are not discussed in this paper.

The state of the contract execution can be observed with the help of a state diagram [3]. In a state diagram each state represents a proposition of the obligations of the parties. The transactions represent the actions performed by the parties. So, the states can be labeled as good, tolerably unacceptable or intolerably unacceptable ones. This way the state of a contract execution in a given time can be checked. If the current state of execution is detected tolerably unacceptable, the controller should reach for a resolution between the parties and try to get the execution back to the good state.

Workflow-Contract-Solution model A totally different kind of approach has been proposed in [5] for the business process and contract management. The approach highlights the human involvement in the process of finding solutions to the exceptions and violations occurring during the contract execution. The prototype architecture proposed for the Workflow-Contract-Solution model is based on top of the UML modeling tool (Konesa) which allows realtime collaboration of parties. The elements used in the problem solution process are UML diagrams which can be edited concurrently. The architecture promotes the cooperative nature of human direction and decision making.

7.5 Conclusions

To ensure the success of contract management there is a need for contracts and contract management systems of good quality. The automation of whole contract life cycle is addressed a lot. To enable the contract management and the contract performance monitoring sharing of confidential information between the parties is required. Still, one of the great challenges of contract management is to get the parties to expose their performance metrics to other parties [6]. The distributed nature of operation also requires the compatibility of various systems used in management purposes in the organizations of the parties. There are also problems concerning the use of agents in contract execution automation dealing with complex legal issues and subjective judgement of agent compliance [2].

Bibliography

[1] BOULMAKOUL, A., AND SALLE, M. Integrated contract management. technical report HPL-2002-183. Tech. rep., Hewlett-Packard Development Company, 2002.

[2] CARDOSO, H. L., AND OLIVEIRA, E. Virtual enterprise normative framework within electronic institutions. In *Proceedings of the 5th International ESAW Workshop on Engineering Societies in the Agents World* (2004).

[3] DASKALOPULU, A., DIMITRAKOS, T., AND MAIBAUM, T. Evidence-based electronic contract performance monitoring. *Group Decision and Negotiation 11*, 6 (2002), 469–485.

[4] GOODCHILD, A., HERRING, C., AND MILOSEVIC, Z. Business contracts for B2B. In *Proceedings of the CAISE00 Workshop on Infrastructure for Dynamic Business-to-Business Service Outsourcing* (2000), pp. 63–74.

[5] IWAIHARA, M., JIANG, H., AND KAMBAYASHI, Y. An integrated model of workflows, e-contracts and solution implementation. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing* (2004), ACM Press, pp. 1390–1395.

[6] KELLER, A., KAR, G., LUDWIG, H., DAN, A., AND HELLERSTEIN, J. L. Managing dynamic services: A contract based approach to a conceptual architecture. In *Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS)* (2002), IEEE, pp. 513–528.

[7] LININGTON, P. F. Automating support for e-business contracts. In *Proceedings of the CoALa workshop on Contract Architectures and Languages* (2004), Z. Milosevic and G. Governatori, Eds., IEEE Digital Library.

[8] MILOSEVIC, Z., JØSANG, A., DIMITRAKOS, T., AND PATTON, M. A. Discretionary enforcement of electronic contracts. In *Proceedings of 6th International Enterprise Distributed Object Computing Conference (EDOC)* (2002), IEEE Computer Society, pp. 39–50.

[9] Office of Government commerce (OGC): Contract management guidelines, 2002. URL `http://www.ogc.gov.uk/sdtoolkit/reference/ogc_library/generic_guidance/contractmgtguidelines.pdf` [11.12.2004].

[10] XU, L., AND JEUSFELD, M. A. A concept for monitoring of electronic contracts. technical report series no. 10. Tech. rep., Tilburg University Infolab, 2003. URL `http://infolab.uvt.nl/research/itrs/itrs010.pdf` [20.12.2004].

Chapter 8

Trust management

Sini Ruohomaa

Trust is an important factor in our daily coexistence with other people. Their free will makes them unpredictable, and trust helps reduce the uncertainty caused by this to an endurable level. In the fast-growing market of electronic commerce, trust plays an even more important role. In order to get customers and make profit, a business should keep its system as open as possible, and make its services easy to use. However, too much openness can quickly become a security nightmare.

In a world where customers and partners may be out of reach of the local laws or not even have a known representation outside the network, it is bad for business to rely on strict discrimination between those who can certainly be brought to justice for breaking contracts and those who maybe cannot. Decisions to trust despite the risks are already being made simply to cope with the situation, but without a framework for organized trust management, the decisions tend to end up being made "on the fly" by people whose interests lie elsewhere and who may never see the long-term consequences of those decisions.

8.1 What is trust?

Trust and reputation, a closely related term, are firmly rooted in sociology. Mui *et al.* argue that when building a computational trust model, these roots should not be forgotten [29]. We begin with an overview on how humans trust, and work our way towards how systems should.

Trust seems to essentially be a means for people to deal with uncertainty about the future and their interaction partners. Stephen Marsh considers the protection of law, a lack of options for possible outcomes and other kinds of limitations, reducing the aforementioned independence of actors, as examples of factors reducing the need to trust [26]. In a more technical environment, "trusted" hardware for monitoring [1] or cryptographically secure communications [9] also work towards reducing uncertainty, which in turn reduces the need to "take things on faith".

A trusting decision is related to a situation where it has some sort of an effect on the trustee. A more general decision may be made beforehand, but usually it is made with a class of applicable situations in mind: "I (will) trust *X* if/when it comes to *Y*", where X is an actor, the *trustee* in this case, and Y is the context, including in particular the attempted action. The effect on the trustee varies: the decision may be *authorisation-related*, determining whether a particular action is allowed at all. If authorisation is granted, further adjustments are possible. *Observation-related* effects cause the tightening or loosening of control measures, most importantly observation, while effects related to *resource allocation* may for example determine how much bandwith, CPU cycles or external resources will be tied to the action.

The effect of trust comes with a risk: an authorization decision means that we expose something in our control to attack or abuse, while reduced observation means misbehaviour may proceed undetected and more resources allocated means more of them may go to waste or be misused. The connection between risk and trust is emphasised by many researchers [26, 27, 12].

Although human trust seems easy to dismiss as somewhat irrational and overly complex, it is an important research topic for computer scientists as well. After all, there is a human actor somewhere behind the user agents and client programs, making trust-involving decisions eg. on whether to use our trust management system or not. If the human users disagree with their trust management system more often than not, they will probably change the system accordingly. Research on how to appear trustworthy to users shows that trustworthiness estimates are determined by much more than just past success or failure to comply to expectations [6, 11, 18, 20]. It has been suggested that contracts, as they make implicit expectations explicit, can encourage more trust [18, 20]. An agreement on social norms does help in building human trust, but binding contracts can also be used to reduce uncertainty in electronical commerce. In case of a breach of contract, a means for dealing with the violation may be given by the contract itself, with the backing of law, if necessary. Contracts are in this sense another measure of control, like insurances, and they may well reduce risk even more than encourage more trust to balance with the remaining risk. The effect of contracts in trust management has been discussed, related to transaction modelling in the e^{3}value project [14, 34].

When two people trust each other through computer programs, new complications arise in the trust relationship. Besides trusting the person at the other end of the communication link, the trustor must also place some, often implicit, trust in everyone who can get in between: the programmer of the software they are using, a malicious third party on the Internet who has a trojan horse installed on either one's computer running the communication software, and anyone who has physical access to the aforementioned computers and a chance at breaking into them and possibly pretending to be the victim of the break-in in a later collaboration. Jøsang argues that computer systems, being rational entities as opposed to their passionate human owners, cannot trust anyone or anything; they only implement trust-emulating policies placed by passionate entities, i.e. people. Similarly, a passionate entity, a person, does not trust a rational entity in any other measure than its ability to withstand manipulation by another passionate entity [21].

8.1.1 Definitions

Trust management is involved in the upkeep, analysis, evaluation and correlation of two central measures, trust and reputation. **Trust** is defined as *the extent to which one party is willing to participate in a given action with a given partner in a given situation, considering the risks involved* (adjusted from Mayer, Jøsang and Lo Presti [27, 23]). Trust is only used in bridging the gap between *risk* and applied countermeasures to that risk, such as closer observation or binding contracts. The **trustor** is a service provider practicing electronic commerce on the Internet, and the **trustee** is either a business partner or an individual requiring access to the trustor's services, as represented by an identifiable agent in the network. There are several other kinds of trust as well, such as that required of the trustee here to be willing to request access to the trustor's services in the first place or the trust placed by trustors in the system to protect themselves from misuse. We limit ourselves to this one kind of trust, however. In the virtual organization context, the **actions** relate to including the trustee in a collaboration, that is, make them a part of the virtual organization, and possibly the actual collaboration with them. The risks may include for example loss of currency or a security breach in the trustor's system. Trust is context-dependent, affected by experience and divisible into a base value per trustee and an adjustment term set per action.

A **trust decision** is binary and based on the balance between trust and risk. **Risks** are tied to assets, as pointed out in [6]. If money or system security are assets, the related risks are losing money or experiencing a breach in security. The risks considered here are limited to those known to the trustor; balancing against something unknown may prove difficult. The business value or importance of an action affects trust similarly to good reputation, increasing the willingness to make a positive trust decision.

A **recommendation** is simply *an attempt at communicating a party's reputation from one community context to another*. A poor recommendation may be detrimental to one's reputation, and there is no separate term for "negative recommendation". The word *attempt* should remind us that the source and target communities are seldom compatible enough to be able to use a recommendation directly. Instead, it may be tuned down by various means, including tagging it as "uncertain information" or giving it a lowered weight in appropriate calculations. In order for reputation to exist in larger than the trivial one-member communities, the members must come to an agreement about what to perceive as that reputation for each given party in that community. Various reputation systems suggest different ways of coming to this agreement. There is no objective truth to be found—or lost—in reputation; some perceptions merely come closer to the target party's real intentions and norms than others.

McKnight and Chervany have made an attempt to classify types of trust [28]. In relation to this classification, this definition of reputation comes close to a *trusting belief*, while the definition of trust is closer to a *trusting intention*, which implies an extent of willingness to depend on a party. While a belief can be formed without active cognition, an intention is something which can be decided. It can also be based on something else than attributes of the trustee: McKnight and Chervany point out that it can also be based on *system trust*, a belief that the environment (e.g. the trust management system) protects the trustor from harm, or *dispositional trust*, which is directed to people in general, either due to a belief that they are trustworthy or as a strategy to make the trustees more likely to cooperate. A *situational decision to trust*, which measures the extent to which the trustor intends to depend on a non-specific party in a given situation, is represented by the measure of action importance given here. The importance is not trustee-specific. The sixth category, *trusting behaviour*, is represented by the consequences: allowing a particular action to happen if sufficient *trusting intention* is present.

8.1.2 Towards managing trust

Trust management research has its roots in authentication and authorization. In the context of authentication, trust is established by means such as digital certificates. These certificates are proof of either identity directly or membership in a "trusted" group. Authentication-related trust is discussed for example in [36, 9]. Policy languages, such as REFEREE [8], PolicyMaker [5] and Keynote [4], then make it possible to automatically determine whether certain credentials are sufficient for performing a certain action, to authorize the trustee. Credentials are sufficient when the system is either convinced of the trustee's identity or knows her to be a member of some sufficiently trusted group—or one of the credentials may be an authorization certificate, signed by someone with the right to delegate such authority. At the authentication level, trust is static and attached to a certain identity or membership. Updating the level of trust in someone based on their actions is not yet considered; it is presumably done out of band by eg. issuing more certificates, or by manual adjustments.

To make trust more dynamic, the performance of the trustee should be considered as well. In the year 2000, monitoring users could be achieved by intrusion detection systems, but the information gleaned was not being used to re-evaluate trust as such; as Grandison and Sloman

note in their survey that year, none of the existing systems then yet covered monitoring and re-evaluation of trust [15]. Since then, behaviour history collection has been included in one form or another in numerous trust models. Behaviour information can be gathered locally [35, 23], or it can be recieved as third-party observations through a reputation system [31]. Involving third parties, however, requires some sort of trust in their statements, as well as comparability between the trustor's views on reputation and the third party's. Reputation systems are discussed in more detail in subsection 8.2.1.

Much work has also gone to identifying factors which either are considered to affect trust directly or which are used together with trust decisions. It was mentioned earlier that uncertainty is involved in increasing the need for trust. Uncertainty is not always problematic to the trustor, however, but mostly when it is related to risk. While the exact relationship between risk and trust is not entirely clear [27, 12], it is agreed that increased risk and an increased need to trust go hand in hand [17, 22]. Risk is relative to the risktaker; for example, the risk of losing a specific monetary sum is less important if the sum is only a small fraction of the usable funds. It has also been noted that increased potential profits in making a decision to trust encourages coping with relatively higher risk [23]. Potential profits can be considered as a part of the action importance mentioned in the definitions subsection. The protection of law and other factors limiting the need to trust according to Marsh may also be considered means to reduce risk [26].

Applications where a more dynamic trust management is beneficial may have a rapidly-changing user base. Newcomers create a problem for a trust management system based on behaviour history alone. The system must determine how much these unknown individuals should be trusted, sometimes without knowing anything about them. While certification may provide a means to introduce an initial trust out of band, it may not be plausible for some applications. Similarly, reputation systems are only helpful if the user has interacted with other systems gathering reputation before. For fully unknown users, a default level of trust must be determined.

This default level of trust affects how open and easily accessible the system is. In an environment where actors can easily change identities, it is also effectively the lowest level of trust any actor has to accept. For virtual organizations, however, identities may be bound to e.g. certificates granted by some third party, and a new identity would be difficult to acquire. If the default level of trust is too high, unknown actors may cause problems in the system and the aforementioned possibility of "starting over" allows misbehaved actors to get rid of their reduced level of trust by creating a new identity which is more trusted. If it is set too low, the user may not be allowed access to the system at all, which makes proving trustworthiness through one's actions rather difficult [2].

8.2 Initialization and upkeep of trust relationships

A trust relationship begins by finding a suitable partner to work with. In a Web Services environment, the partner may be found e.g. via an UDDI (Universal Description, Discovery and Integration) registry [3]. A search in the registry may result in a plethora of potential partners, some of which may be incompetent or even malicious. Tools are needed for partner selection. After the best partner has been chosen, they are assigned a default level of percieved trustworthiness or reputation. This is at best a good guess, and beginning from the first decision to trust the partner to complete a particular action, experience should be gathered concerning their behaviour in practice. This experience can then be used to update the reputation to produce more educated trust decisions in the future.

8.2.1 Reputation systems

One way of finding out more about a potential partner's background is to ask a reputation system. A reputation system stores information about the past behaviour of a group of entities in the form of the community's perceptions of them. This may include information from book reviewers's percieved fairness to online companies' percieved competence and reliability. The construct has been found to help markets of computer-aided human-to-human interaction, by reducing the level of uncertainty about new acquaintaces to an endurable level [31, 22]. Reputation systems use both centralized and distributed strategies in information collection and retrieval. For example, eBay [10], an online marketplace, asks its users to rate their trading partners on a fixed scale and leave additional comments about their performance. The information gathering is distributed, but the result is stored and retrieved from a central server [22]. Organizations like The International Chamber of Commerce, who act as advisors and certifiers in first-trade situations and thus provide information about other organizations' reputation, represent a centralized approach [33]. Everyone wishing to use a reputation system must be able to trust the information provider to not insert false ratings or omit information at will. While this may be a small challenge in a centralized approach, where one party must be trusted by everyone, it is considerably more difficult to achieve for a fully distributed approach, where nearly everyone should be able to trust everyone else as a reputation information source.

In a reputation system where information gathering is distributed, anyone can claim anything about anyone. The problem escalates when reputation is more valuable; competitors may be given bad ratings to disrupt their business, or malicious users may cooperate to form a circle of positive feedback passing, regardless of their actual performance. Some sort of control over the trustworthiness of reputation statements is needed, as especially new users are unable to tell the difference between honest and dishonest statements about other users [2]. Obreiter suggests the use of evidence in the form of trade reciepts, which can be used as a sort of certificate of having behaved well at some point [30]. Similarly, any certificate proving membership in a group of high repute can be used as a form of presentable "proof" of good behaviour.

Dishonesty in the expression of perceptions should be somewhat difficult to detect, let alone prove, but in the context of reputation systems we can understand dishonesty as either too agreeing or too disagreeing to be likely to be useful for others, as presented in the Pinocchio trustworthiness service for a recommendation system [13]. Kalcklösch and Herrmann apply statistical methods in ad hoc networks where trust information communication is automatic [24]. While these methods may not be the approach of choice for an established web service provider looking for partners, they serve well in their context. One must note that solutions on different levels, from infrastructure to service to community level, have very different needs.

Even if the recommending party is known to be honest and knowledgeable, their statements may be useless if the principles behind them are not comparable to those of the reciever. A good reputation as a trader in an online auctioning system does not necessarily mean that the user should be given eg. wider access priviledges in a distributed software development project. This makes representing trust or reputation as a single numeric value somewhat problematic: If a reputation statement says that a user is trustworthy by "3 on a scale from 1 to 5", what does it mean in the reciever's context? Has the default been 1 or 3 and how easily is it increased or decreased? If this is a trader's reputation, should I trust them to sell a car to me if they got their reputation by selling pocket lighters [31]? This causes difficulties for porting ratings from one system to another as well. Recommendations remain an *attempt* at communicating reputation information between communities.

If human reviewers are involved, their special tendencies should also be considered. For ex-

ample, most people tend to avoid giving negative feedback on eBay and prefer to resolve conflicts out-of-band *before* giving their final judgement [31]. On the other hand, it is common that after a person has dealings with a company, e.g. a store, good experiences may be at most shared with closest friends, while bad experiences are discussed more freely.

Resnick *et al.* identify three properties that are required from a successful reputation system: First, the entities must be long-lived and have use for reputation. Second, feedback must be captured, distributed and made available in the future. Third, the feedback must be used to guide trust decisions [31]. The first property implies some problems that newcomers have with reputation systems. Besides the problem of finding a trustworthy information provider, they must gain a reputation themselves [2].

The usability of reputation information from outside sources is not limited to choosing a partner. It can also be included as a factor in the trust estimate of a partner, along with their locally gathered reputation based on first-hand experience. Initially, as there is no local information about the partner's behaviour, external reputation information may hold considerable weight in a trust decision. A trust management system also needs some sort of a default level of reputation to assign to complete strangers. This default may be raised or lowered based on a general view of the world the system operates in. If the average partner seems to be a somewhat unreliable opportunist, the default may be reduced. On the other hand, if the system operates in an environment of honest cooperation, the default may be increased, easing the burden of "proving oneself" for the newcomer and thus making the collaboration easier to participate in. On the other hand, as it is difficult and arduous to stop the creation of new identities, the default level of reputation is also the lowest reputation any users willing to go through the trouble of creating a new identity will have to endure. With a high default level of reputation, the efficiency of low reputation as a sanction is therefore reduced.

8.2.2 Observation

Observation can be done in two different roles: either as an active participant of a collaboration of two, or as an outsider, a silent third party. In the first case, which this paper concentrates on, the actions of the observed party are seen through a personal context, and as a result can be analysed in more depth. In the latter case, there may be unknown factors affecting the collaboration. Some context information may be revealed only to an internal observer, while external observers must draw their conclusions without that support. The difference probably remains rather small in practice, as we can consider observer software an outsider as well, while software providing the actual service is the active participant.

Traditionally, observation has been studied in the form of intrusion detection. The research can be put to use in observing users or partners in a trust management system as well. Intrusion detection software benefits from "insider" information as well. The traditional approach to intrusion detection looks at system calls or network traffic e.g. at the transport (TCP/UDP) level, while application intrusion detection adds a higher level of understanding to the analysis by being aware of particular applications observed instead of trying to understand network traffic or system calls for the entire system.

Thorough observation ties up resources, which may make it simply impossible to keep close track of what every user is doing at all times. Herrmann and Krumm, who study monitoring and trust directed towards system components, suggest adjusting the intensity of monitoring and behaviour checks according to the level of trust in the observed component, its hosting environment and its vendor [19]. Some components are not necessarily considered trustworthy in their system, and are therefore monitored with the help of more trusted observer components, wrappers, placed

around them.

Suspicious activity can in the most straightforward case be actual misbehaviour in the form of breaking system policy or not following other forms of orchestration. It can, however, also be an action which either should only be taken by actors in a different role or is merely highly unusual behaviour for the observed party. A change in an actor's behaviour may give reason to suspect that communications with the actor have been compromised, either on the way or in the source by subverting the actor or its representee somehow. The exact reason behind the unusual behaviour may not be of consequence; the actor is not behaving as it should, and the observing system wants to protect itself against these possibly malicious influences.

Besides detecting suspicious activity, an observation system could be used as a witness of "normal" behaviour. Good experiences lead to better or at least more "certain" reputation in many reputation systems where the users themselves act as witnesses. On the other hand, if a reputation estimate includes a measure of confidence, ie. how certain the estimate is, a lengthy period of observation showing behaviour in agreement with the current reputation may be taken as increased confidence in the reputation estimate.

8.2.3 Reaction to new evidence

When an observation system has detected suspicious activity, a decision must be made on what to do with the information. If behavioural analysis is done in real-time and at some points transactions stay on hold until a go-ahead is sent by the observation system, it becomes possible to prevent attacks instead of only detecting them. An intrusion prevention system (IPS) extends the concept of intrusion detection by also considering preemptive measures. However, automated reaction to detected attacks requires very accurate, real-time intrusion detection. The anomaly intrusion detection approach, which is based on profiling what is considered normal behaviour and detecting deviations from the profile, is infamous for its relatively high rate of false positives. It would therefore be a sub-optimal strategy if used alone in intrusion prevention. Misuse intrusion detection, which is based on recognizing known attacks by their descriptions, generally causes less false positives. It will also miss some attacks due to not knowing them beforehand, however. Specification-based anomaly detection [32] combines the rulesets of misuse intrusion detection with the focus on normal, accepted behaviour in anomaly detection. It may not be feasible, however, for applications beyond a small set for which specifications are straightforward to write.

The idea of preventing policy-breaking or otherwise suspicious activity is not new. Various system management tools from access control lists to intrusion prevention systems have been functional without any explicit notion of trust. However, while a decision to prevent something from being done can be done strictly based on the action itself and the user's permanent access rights, trust could be used as a basis for determining more dynamic access rights instead. As for intrusion prevention, a wider background and context information gained by attaching a reputation-gathering system to the observation artillery can go a long way to reducing the need for requesting for system administrator aid whenever it is impossible to certainly determine whether a user is attempting an attack or merely doing something out of the ordinary.

The evolution of reputation and trust stands at the heart of a trust management system. It also seems to be a subject which is seldom discussed in detail in the practical context. One reason for this may be the need for configurability; research should not impose any particular policy on trust updates upon its applications. Some detailed examples in the right context can prove invaluable, however.

Mathematical models give tools and formulae for dealing with experience as it is represented as a binary for "cooperated vs. defected" [29] or by scalars [25]. Systems closer to practical

implementation have not yet gone that far. The SECURE project seems to be the closest to a practical implementation, building a trust management framework applicable for nearly any mobile or ubiquitous computing need [7]. They provide a formal model of incorporating experience into new trust values, but the source of experience is still unclear. The Sultan project has included an experience (or "evidence") collection module as a part of their system [35, 16], but do not seem to have implemented or described the means through which experiences translate into updates in trust or reputation levels. The concept of experience seems to be considerably more complicated in practice than in theory, especially if trust and reputation are represented as something more complex than a scalar to begin with.

As the user's reputation is updated based on their actions, information about the change can be sent to reputation systems spanning larger communities, such as those used by the local reputation system to estimate the initial reputation of newcomers. The information, passed in the form of recommendations[1], can then be used to adjust the user's reputation in the target community as well. This requires that the recommendation includes a representation of the user's identity that is recognized in both communities. It is noteworthy that a recommendation sent from one reputation system to another is not a direct adjustment of the target system's reputation scales. It is information about a perception change in the sender system, and the reciever system decides what to do with it. Communicating reputation changes across systems involves agreements on how the information is dealt with, and the topic is central to the development of reputation systems.

8.3 Conclusions

Trust can be used as a tool to counter uncertainty, which is always evident in virtual organizations due to their dynamic nature. A trust decision balances the trustor's willingness to increase their dependence on the trustee against the risks inherent in doing so, and may involve a requirement for e.g. additional constraints to reduce the risks involved. Automating routine trust decisions at a service level can save considerable amounts of man-hours when compared to the work required to achieve a similar level of dynamism and flexibility by manual adjustments to traditional access control and authorisation systems.

A trust management system is not infallible, however. Human intervention should be possible in cases of exceptions of the rule or new kinds of situations where routine rules are difficult to apply. Especially in the area of observation and reputation updates, the research on trust management is only getting started. While reputation systems have been researched for a while now, they mostly require constant experience input from humans and are designed for computer-aided collaboration between people instead of functioning in the context of independent software agents. Reputation systems also often direct trust in an actor in general, and do not relate reputation or trust to a particular action.

[1] While the word recommendation is positive, a poor recommendation may well have a negative effect on reputation. The word is used for the communication of reputation information in general.

Bibliography

[1] BALDWIN, A., AND SHIU, S. Hardware security appliances for trust. In *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings* (May 2003), vol. LNCS 2692/2003, pp. 46–58.

[2] BARBER, K. S., FULLAM, K., AND KIM, J. *Challenges for Trust, Fraud and Deception Research in Multi-agent Systems*, vol. 2631/2003 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003, pp. 8–14.

[3] BELLWOOD, T., CLÉMENT, L., EHNEBUSKE, D., HATELY, A., HONDO, M., HUSBAND, Y. L., JANUSZEWSKI, K., LEE, S., MCKEE, B., MUNTER, J., AND VON RIEGEN, C. *UDDI Version 3.0. UDDI Spec Technical Committee Specification, 19 July 2002*. UDDI.org, July 2002.

[4] BLAZE, M., FEIGENBAUM, J., AND KEROMYTIS, A. D. KeyNote: Trust management for public-key infrastructures (position paper). In *Security Protocols: 6th International Workshop, Cambridge, UK, April 1998. Proceedings* (Apr. 1998), vol. LNCS 1550/1998, Springer-Verlag, pp. 59–63.

[5] BLAZE, M., FEIGENBAUM, J., AND LACY, J. Decentralized trust management. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 1996), IEEE.

[6] BRÆNDELAND, G., AND STØLEN, K. Using risk analysis to assess user trust - a net-bank scenario -. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LCNS 2995/2004, pp. 146–160.

[7] CAHILL, V., GRAY, E., SEIGNEUR, J.-M., JENSEN, C., CHEN, Y., SHAND, B., DIMMOCK, N., TWIGG, A., BACON, J., ENGLISH, C., WAGEALLA, W., TERZIS, S., NIXON, P., SERUGENDO, G. D. M., BRYCE, C., CARBONE, M., KRUKOW, K., AND NIELSON, M. Using trust for secure collaboration in uncertain environments. *Pervasive Computing 2*, 3 (Aug. 2003), 52–61.

[8] CHU, Y.-H., FEIGENBAUM, J., LAMACCHIA, B., RESNICK, P., AND STRAUSS, M. REFEREE: Trust management for Web applications. *Computer Networks and ISDN Systems 29*, 8–13 (1997), 953–964.

[9] DJORDJEVIC, I., AND DIMITRAKOS, T. Towards dynamic security perimeters for virtual collaborative networks. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 191–205.

[10] The eBay online marketplace, 2004. URL `http://www.ebay.com` [5.11.2004].

[11] EGGER, F. N. *From Interactions to Transactions: Designing the Trust Experience for Business-to-Consumer Electronic Commerce*. PhD thesis, Eindhoven University of Technology, 2003.

[12] ENGLISH, C., TERZIS, S., AND WAGEALLA, W. Engineering trust based collaborations in a global computing environment. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 120–134.

[13] FERNANDES, A., KOTSOVINOS, E., ÖSTRING, S., AND DRAGOVIC, B. Pinocchio: Incentives for honest participation in distributed trust management. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 64–77.

[14] GORDIJN, J., AND AKKERMANS, H. Designing and evaluating e-Business models. *IEEE Intelligent Systems 16*, 4 (2001), 11–17.

[15] GRANDISON, T., AND SLOMAN, M. A survey of trust in Internet applications. *IEEE Communications Surveys and Tutorials 3*, 4 (Dec. 2000), 2–16.

[16] GRANDISON, T. W., AND SLOMAN, M. Sultan - a language for trust specification and analysis. In *Eighth Workshop of the HP OpenView University Association, Berlin, June 24-27, 2001* (June 2001), HP OpenView University Association. URL `http://www.hpovua.org/PUBLICATIONS/PROCEEDINGS/8_HPOVUAWS/Papers/Paper01.2-Grandison-Sultan.pdf`.

[17] GRAY, E., SEIGNEUR, J.-M., CHEN, Y., AND JENSEN, C. Trust propagation in small worlds. In *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings* (May 2003), vol. LNCS 2692/2003, pp. 239–254.

[18] GRIMSLEY, M., MEEHAN, A., AND TAN, A. Managing Internet-mediated community trust relations. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 277–290.

[19] HERRMANN, P., AND KRUMM, H. Trust-adapted enforcement of security policies in distributed component-structured applications. In *Proceedings of the 6th IEEE Symposium on Computers and Communications. Hammamet, Tunisia* (2001), IEEE Computer Society Press, pp. 2–8.

[20] ISHAYA, T., AND MUNDY, D. P. Trust development and management in virtual communities. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 266–276.

[21] JØSANG, A. The right type of trust for computer networks. In *Proceedings of the ACM New Security Paradigms Workshop* (1996), ACM.

[22] JØSANG, A., HIRD, S., AND FACCER, E. Simulating the effect of reputation systems on e-markets. In *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings* (May 2003), vol. LNCS 2692/2003, pp. 179–194.

[23] JØSANG, A., AND PRESTI, S. L. Analysing the relationship between risk and trust. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 135–145.

[24] KALCKLÖSCH, R., AND HERRMANN, K. Statistical trustability (conceptual work). In *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings* (May 2003), vol. LNCS 2692/2003, pp. 271–274.

[25] LIU, J., AND ISSARNY, V. Enhanced reputation mechanism for mobile ad hoc networks. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 48–62.

[26] MARSH, S. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Department of Computer Science and Mathematics, 1994.

[27] MAYER, R. C., AND DAVIS, J. H. An integrative model of organizational trust. *The Academy of Management Review 20*, 3 (July 1995), 709–734.

[28] MCKNIGHT, D. H., AND CHERVANY, N. L. The meanings of trust. Tech. rep., University of Minnesota, MIS Research Center, 1996. Tables included as attachments in the online version.

[29] MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. A computational model of trust and reputation. In *35th Annual Hawaii International Conference on System Sciences (HICSS'02)* (Jan. 2002), vol. 7, IEEE Computer Society.

[30] OBREITER, P. A case for evidence-aware distributed reputation systems overcoming the limitations of plausibility considerations. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 33–47.

[31] RESNICK, P., ZECKHAUSER, R., FRIEDMAN, E., AND KUWABARA, K. Reputation systems. *Communications of the ACM 43*, 12 (Dec. 2000), 45–48.

[32] SEKAR, R., GUPTA, A., FRULLO, J., SHANBHAG, T., TIWARI, A., YANG, H., AND ZHOU, S. Specification-based anomaly detection: a new approach for detecting network intrusions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA* (2002), pp. 265–274.

[33] TAN, Y.-H. A trust matrix model for electronic commerce. In *Trust Management: First International Conference, iTrust 2003, Heraklion, Crete, Greece, May 28–30, 2003. Proceedings* (May 2003), vol. LNCS 2692/2003, pp. 33–45.

[34] TAN, Y.-H., THOEN, W., AND GORDIJN, J. Modeling controls for dynamic value exchanges in virtual organizations. In *Trust Management: Second International Conference, iTrust 2004, Oxford, UK, March 29–April 1, 2004. Proceedings* (Mar. 2004), vol. LNCS 2995/2004, pp. 236–250.

[35] WAGEALLA, W., CARBONE, M., ENGLISH, C., TERZIS, S., AND NIXON, P. A formal model on trust lifecycle management. In *Workshop on Formal Aspects of Security and Trust (FAST2003) at FM2003*, vol. IIT TR-10/2003. IIT-CNR, Italy, Sept. 2003, pp. 184–195. URL `http://www.iit.cnr.it/FAST2003/fast-proc-final.pdf` (TR-10/2003).

[36] WINSBOROUGH, W. H., SEAMONS, K. E., AND JONES, V. E. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00. Proceedings* (Jan. 2000), vol. 1, IEEE, pp. 88–102.

Appendix A

AO4BPEL

The Service-Oriented Computing paradigm (SOC) [8] and the Web Services [11] technology has provided software vendors and business enterprises concepts and methods to overcome problems related to the integration of heterogeneous business information systems. The SOC paradigm is based on a service oriented architecture (SOA) which provides means for describing, publishing, discovering and binding autonomous computational elements, services. Services can be composed to more complex services which are orchestrated in process-like fashion [8]. Web Services technology provides a simple, low-cost communication middleware platform based on open TCP/IP- and XML-based standards.

BPEL [1] is a XML-based language for describing business processes composed out of activities. Each activity is implemented as a Web Service method. BPEL is used both as a execution language, describing actual behaviour of a participant, and as an abstract language for business protocols, describing mutually visible behaviour of participants in a collaboration [1]. Business processes are modeled in as BPEL monolithic entities which include all the actions needed for communication and coordination between composed services as well as actions related to business rules, which are statements or constraints defining some aspects of a business [9].

Monolithic approach to business process definition implies that if business rules change then every BPEL process influenced by that rule must be re-implemented. As it happens to be, business rules may change even quite rapidly which means that business functional logic, the purely operational part of business process, and business rules should be separated. If process functionality can be separated from business rules, then the evolution of rules does not render the whole process useless. This can be achieved by following a aspect oriented approach.

Aspect-oriented programming is a programming technique that was developed to capture properties that cross-cut the system's basic functionality and to improve separation of concerns [7]. The goal of aspect oriented programming enables programmer to cleanly separate components and aspects from each other, thereby increasing reuse and maintainability of components and aspects.

AO4BPEL [3, 4] is a business process orchestration language and engine based on the BPEL-language [1] and principles of aspect oriented programming, AOP [7, 5]. In AO4BPEL the aspect oriented programming paradigm is used to increase the modularity and reusability of business processes by separating functionality from business rules [3]. As in AspectJ [6], AO4BPEL uses join points (where we can put aspects), point cuts (selecting join points) and advices (code to implement aspect functionality) to define aspects. Every activity (e.g. <invoke> [1]) is a possible join point in AO4BPEL and advices can be executed *before*, *after* or *around* an activity. Point cuts are defined using the XML query language XPath [10], since BPEL is XML-based [3]. Advice functionality is defined using the BPEL language.

The behaviour of business processes can be adapted even during their execution in AO4BPEL [3].

By using dynamic aspect weaving techniques provided by orchestration engine (as in AO4BPEL) or an underlying Java Virtual Machine (e.g. [2]), new business rules can be applied to processes that are already under execution. This is a useful feature especially for long-running business processes and when we want for some reason change a composition to e.g. include additional functionality.

In Table A.1 is an example aspect from [3]. This example defines an aspect which collects audit data from all method invocations that are used to search flights of Lufthansa.

```
<aspect name="Counting">
 <partnerLinks>
  <partnerLink name="JavaExecWSLink" &/>
 </partnerLinks>
 <variables>
  <variable name="invokeMethodRequest" &/>
 </variables>
 <pointcutandadvice type="after">
  <pointcut name="Lufthansa Invocations">
   //process//invoke[@portType ="LufthansaPT" and @operation ="searchFlight"]
  </pointcut>
  <advice>
   <sequence>
    <assign>
     <copy>
      <from>increaseCounter</from>
      <to variable="invokeMethodRequest" part="methodName"/>
     </copy>
    </assign>
    <invoke partnerLink="JavaExecWSLink" portType="JavaExecPT"
     operation="invokeMethod" inputVariable="invokeMethodRequest"/>
   </sequence>
  </advice>
 </pointcutandadvice>
</aspect>
```

Table A.1: Example aspect in AO4BPEL platform [3].

The separation of business process functionality and business rules can be achieved using AOP paradigm, as presented in [3, 4]. A separation of concerns into modular units provides better reuse of functional and logical parts. If business rules can be handled as first-class entities, they can be more easily be reused in different processes and situations. Business rules can be also declared in a more declarative fashion using logical languages and rule engines [4]. The use of rule engines and logical inference provides formal and well known declarative method to describe domain knowledge which is better suited for users not comfortable with programming languages. The downside is a more complex (i.e. slower) orchestration engine and the fact that it can be sometimes hard to cross the boundary between terms of a purely functional language, such as BPEL, and those of pure logic.

Bibliography

[1] ANDREWS, T., CURBERA, F., DHOLAKIA, H., ET AL. *Business Process Execution Language for Web Services Version 1.1*, May 2003. URL: `http://www.ibm.com/developerworks/library/ws-bpel/` [27.1.2005].

[2] BOCKISCH, C., HAUPT, M., MEZINI, M., AND OSTERMANN, K. Virtual machine support for dynamic join points. In *Proceedings of the 3rd international conference on Aspect-oriented software development* (2004), ACM Press, pp. 83–92.

[3] CHARFI, A., AND MEZINI, M. Aspect-Oriented Web service Composition with AO4BPEL. In *Proc. ECOWS 2004* (2004), L. J. Zhang, Ed., vol. 3250 of *LNCS*, Springer.

[4] CHARFI, A., AND MEZINI, M. Hybrid Web Service composition: Business processes meet business rules. In *ICSOC04 2nd International Conference on Service Oriented Computing* (2004), ACM. Accepted. To be published.

[5] ELRAD, T., FILMAN, R. E., AND BADER, A. Aspect-oriented programming: Introduction. *Commun. ACM 44*, 10 (2001), 29–32.

[6] KICZALES, G., HILSDALE, E., HUGUNIN, J., KERSEN, M., PALM, J., AND GRISWOLD, W. G. An overview of AspectJ. In *Proceedings European Conference on Object-Oriented Programming* (Berlin, Heidelberg, and New York, 2001), vol. 2072 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 327–353.

[7] KICZALES, G., LAMPING, J., MENDHEKAR, A., MAEDA, C., LOPES, C., LOINGTIER, J.-M., AND IRWIN, J. Aspect-oriented programming. In *ECOOP'97—Object-Oriented Programming* (1997), M. Akşit and S. Matsuoka, Eds., vol. 1241 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 220–242.

[8] PAPAZOGLOU, M. P. Service-oriented computing: Concepts, characteristics and directions. In *4th International Conference on Web Information Systems Engineering (WISE 2003), 10-12 December 2003, Rome, Italy* (dec 2003), IEEE Computer Society, pp. 3–12.

[9] THE BUSINESS RULES GROUP. Defining Business rules, What are they really?, July 2000. URL: `http://www.businessrulesgroup.org` [27.1.2005].

[10] W3C. XML path language (XPath) version 1.0 – W3C recommendation. Available at http://www.w3.org/TR/xpath.html, 2000. URL: `http://www.w3.org/TR/xpath.html` [27.1.2005].

[11] W3C. Web services activity, 2004. URL: `http://www.w3.org/2002/ws` [27.1.2005].

Appendix B

CrossFlow

CrossFlow was a research project concerning dynamic contract development and contract enactment to support workflow management among virtual enterprises. The project took place in Europe and lasted for a couple of years (1998-2000). The main participants in the project were IBM with its e-business group, development laboratory and workflow software development group, GMD-IPSI, University of Twente, Sema Group and some partners to experimenting. During the project about twenty papers were published.

The motives for the project were demanding and fast changing requirements of the market environment and a continually growing complexity of the products, which requires outsourcing and tighter cooperation among cooperating enterprises. According to this, advanced workflow management and process controlling are required.

The CrossFlow project aims at supporting the processes and workflows of dynamic B2B through electronic contracts. Contracts are seen as the basis for the life cycle of dynamic virtual enterprises. Based on the contracts the cooperation between the enterprises are initiated, controlled, monitored and eventually finished.

The CrossFlow technology consists of a matchmaking facility for dynamic service outsourcing through contract templates and enactment infrastructure to form the cooperation environment based on the contract. The tehcnology is based on a workflow management platform MQSeries Workflow by IBM. A contract model was also developed to specify for instance the concepts used and processes involved. Contracts were designed to be based on XML. In the contracts it is also possible to define the processes and workflows with constraints and requirements so that during the enactment quality of service can be monitored, enactment can be controlled and changes made dynamically. The development of a more sophisticated contract framework was seen as a future challenge.

Literature

Grefen, P., Aberer, K., Hoffner, Y. and Ludwig, H., CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises; International Journal of Computer Systems Science & Engineering, Vol. 15, No. 5, 2000; pp. 277-290.

For more information see `http://www.crossflow.org`.

Appendix C

E-ADOME

The Internet is a global common platform for communication among organizations and individuals. It is used for various commercial activities and provides value-added services. Services provided via the Internet are called E-services. In the current Internet environment there is a clear need for cross-organizational workflow support to these activities. Many organizations have already been using some kind of workflow technologies. There exists several advanced workflow management systems (WFMS) for modeling cross-organizational activities with workflows. E-ADOME [3] is one of these.

One of the basic requirements in order to support workflow interoperability is a mechanism to let only authorized external parties to access the related and relevant parts of a workflow. E-ADOME proposes the use of workflow views as a fundamental mechanism for cross-organization workflow interaction. The components of a workflow view include the process flow graph, input/output parameters, objects, rules, events, exceptions and exception handlers derived from the original workflow. A workflow view model can be defined with Unified Modeling Language (UML). A simple workflow view definition language has also been developed to express the UML model. This language can be expressed also with the XML schema.

Each workflow view must be specified with one or more accessible roles. A role represents a collection of agents of similar properties and therefore the roles can also be used in specifying the security context. The role concept serves the purpose of classifying business partners into groups, enabling personalized business processes.

The cross-organization interoperation model uses an interoperation protocol which consists of workflow views, communication graphs between these views, and a set of interoperation parameters. This information should be stored at each of the parties involved in the business process. In the E-ADOME interoperation model, the interoperation parameters capture a set of attributes (for ex. Accept, Offer, Goal, Schedule, Payment, Documents, QoS, Exception_Rules, Commit) whose values describe the necessary information for the business process, which is usually in the form of parameters. During interoperation of the business process, besides the parameters, the two parties have to agree on a common workflow, task assignments, and cross-organizational message exchanges. In the E-ADOME workflow protocol model, the necessary balance between trust and security is achieved through the workflow view mechanism. Each party specifies a view of its internal workflow that is accessible to the other party.

In addition to each party's workflow mechanism also cross-organizational communications are required in order to support the specific business process. These communications consists of information exchange, control exchange and synchronization. In the E-ADOME interoperation model, there are some tasks in each workflow view, called communicating tasks, through which two parties communicate. The cross-organizational control and information flow is speci-

fied within communicating tasks and their associated communication links. Each communicating task receives and sends a set of messages in specified order.

When two parties want to start interoperation they form an interoperation protocol and they have to decide on the value of the interoperation parameters. Then, each party has to present the view as specified in the interoperation protocol model, in order to allow access to workflows of each other, and to incorporate the protocol requirements on the data and control flow. On the basis of these workflow views both parties can identify the communicating tasks of both workflow views. Then, by matching these input and output messages, both parties can derive the necessary communication links and their order. At the same time, this process also detects possible mismatches. There is no need for centralized control.

Web services provide a new interoperable platform for Internet applications. Web services perform functions which can be anything from simple requests to complicated processing of business data. The Web Service architecture is suitable for cross-organizational collaboration in a highly dynamic environment as it supports just-in-time integration, encapsulating and true interoperability. This allows the implementations to be programming language-neutral and communications mechanism-independent. E-ADOME extends a flexible, web-enabled workflow management system ADOME-WFMS [1, 2], in order to provide support for specifying, executing and monitoring composite e-services. In particular, E-ADOME strengthens the external interface layer to interact with different types of agents over the Internet more effectively. The E-ADOME environment can be divided into the following layers:

- The ADOME / OODBMS Layer. ADOME was developed to enhance the knowledge-level modeling capabilities of OODBMS models, so as to allow them to more adequately deal with data and knowledge management requirements of advanced information management applications, especially workflow management systems (WFMS).

- The ADOME-WFMS Layer is a flexible WFMS built upon ADOME facilities, supporting effective management of agents, on-line workflow evolution, and automatic and cooperative exception handling.

- The Internet Interface Layer is the enhancement layer to the WFMS for ADOME-WFMS to interact with various types of external agents through the Internet. An Internet Message Sender sends alerts to users and agents via ICQ or E-mail. This module also sends out requests to other software agents using a compatible API. Internet Event Interceptor receives responses or alerts from software agents through a compatible API and translates them to ADOME events. An agent may be internal or external to the organization and may itself be another WFMS. An Access Security Layer is added to handle external communications. Web Script Processor enables the E-ADOME to initiate an automatic conversation script with other interactive, web-based service providers without compatible software API, including most on-line ordering web pages or service report forms. The newly added Web Service Interface module enables E-ADOME to communicate with other WFMSs to allow for more advanced task execution and control in foreign WFMSs. It enables human agents or users to interact with E-ADOME, to access the database, and to report work progress, in addition to programmed web interface. The Web Service Interface module can be integrated without changing of underlying layers.

The enabling power of E-ADOME for e-service applicability mainly relies on the additional Internet interface layer on top of ADOME-WFMS. This interface layer can send and receive messages through the Internet, in order to communicate with distributed users and other service agents. Arrival of incoming messages can be detected as events to trigger actions of the WFMS

specified by both regular workflow specification and Event-Condition-Action rules. In particular, the WFMS interface module, now equipped with Web Service mechanism, further facilitates crossorganizational e-service (workflow) enactment. As the E-ADOME extension is built on top of ADOME-WFMS, it is perceived that the techniques presented are applicable to other similar WFMSs or other information systems.

Bibliography

[1] CHIU, D., LI, Q., AND KARLAPALEM, K. A meta modeling approach for workflow management system supporting exception handling. *Information Systems 24*, 2 (1999), 159–184.

[2] CHIU, D., LI, Q., AND KARLAPALEM, K. Web interface-driven cooperative exception handling in adome workflow management system. *Information Systems* (2001).

[3] CHIU, D. K., CHEUNG, S.-C., KARLAPALEM, K., LI, Q., AND TILL, S. Workflow view driven cross-organizational interoperability in a web-service environment. In *Web Services ,E-Business, and the Semantic Web: CAiSE 2002 International Workshop, WES 2002, Toronto, Canada* (May 2002), vol. LNCS 2512, Springer-Verlag, pp. 41–56.

Appendix D

FETISH-ETF

The tourism industry has a history of regional clustering and co-operation among service providers to create value add services for their customers. Value add services, meaning a composition of number of low-level services into a single service, are an important part of the tourist industry since they allow the arrangement of customized travel packages that help companies differentiate their services. A value add service might be, for example, a combination of car rental, hotel booking and restaurant information services that create a single travel planning service. By forming regional clusters and providing value add services, even smaller tourism industry related companies can remain competitive in a global scale.

The Federate European Tourism Information System Harmonization - Engineering Task Force (FETISH-ETF) [2] was launched to help the tourism industry create virtual organizations and value add services flexibly by providing a platform that allows interoperability between new and old services, processes and applications. The project ran for two years between 2000 and 2002 within the IST Programme and was a co-operation between number of European universities and companies.

The project's main goals were [3]:

- To increase the availability of tourism related services.
- To improve the interoperability among service providers and clients.

To these ends the project takes advantage of the Internet and service federation methods. The idea of service federation is to support a network of autonomous nodes providing services [1]. These nodes can be in charge of their parts of the system but they also belong to a common federation that allows a distributed access to their services by the use of common interfaces, service registration, lookup and invocation. With such federation present, its members can find and use available services and combine them into higher-level services to add value to their customers.

The FETISH-ETF project has developed a layered, Java based architecture to help in these goals (figure D.1). The architecture consists of service, directory and VAS support layers.

- The service layer contains the actual services developed by the federation members.
- The directory layer provides service descriptions and look-up services.
- The VAS support layer offers support services such as access rights and business process management. In addition, it can be used to create value add services by combining multiple lower level services into a high-level service.

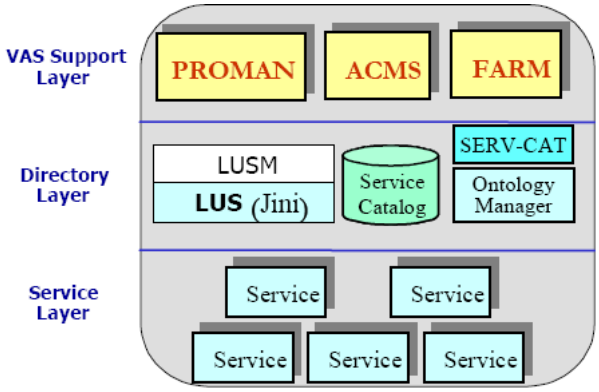


Figure D.1: Architecture [3].

New services are created by selecting an existing interface definition or submitting a new service type. If an interface exists for this type of service already, the service providers must implement it. This is to support interoperability between service providers. Once a service has been developed according to the interface, it is registered to a service catalog. Other members of the federation may then query the catalog for a service handle. This handle can be used to invoke the service.

The FETISH architecture also provides means to create value add services in a form of distributed business processes. Business process templates are process types that are filled with context related information to create business process models which are instantiated into business process instances. The templates and models are shared very much in the same way than services are. The templates of a business process are usually common property among the federation but models of them are the property of the developer and are used to execute the process. Templates and models are developed using a graphical language that describes, among other things, activities, services and state transitions. When the model is executed, its transition path is followed and its activities and service calls are invoked. This way service providers can form aggregate services from the services provided by other members of the federation.

This architecture and its parts are developed in a distributed way mainly in Spain, Portugal, Italy and Netherlands by universities and companies and will be released as open source.

Bibliography

[1] AFSARMANESH, H., AND CAMARINHA-MATOS, L. M. Future smart-organizations: A virtual tourism enterprise. In *Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00)-Volume 1* (2000), IEEE Computer Society, p. 456.

[2] The FETISH-ETF project homepage, 2004. URL: `http://www.fetish.t-6.it/` [27.1.2005].

[3] KALETAS, E. C., CARDOSO, T., CAMARINHA-MATOS, L. M., AND AFSARMANESH, H. Service federation in virtual organizations. In *PROLAMAT '01* (Nov. 2001). URL: `http://carol.wins.uva.nl/~kaletas/publications/prolamat12.pdf` [4.10.2004].

Appendix E

MASSYVE

MASSYVE was a three-year Keep-In-Touch project based in Brazil, and a part of the International Cooperation with Developing Countries (INDO-DC) EU programme. The project started in 1997 and finished in 2000. The participants were:

- New University of Lisbon, Portugal: Luis M. Camarinha-Matos, project coordinator
- Federal University of Santa Catarina, Brazil: Ricardo J. Rabelo, Alexandra P. Klen, André Jacomino, Michel B. Geszychter
- University of Amsterdam, the Netherlands: Hamideh Afsarmanesh
- CSIN (a business partner): A. Valentim Santos-Silva.

All participants except for Geszychter and Santos-Silva also participated in PRODNET-II, which finished a year earlier, in 1999.

MASSYVE stands for Multi-Agent Manufacturing Agile Scheduling Systems for Virtual Enterprises. The project started with an agile scheduling system (the HOLOS framework) prototype, previously developed at the New University of Lisbon, and aimed to both evaluate the prototype and apply the Multi-Agent Systems (MAS) paradigm to build an advanced layer on top of it. The target field was manufacturing systems.

A multi-agent system consists is a network of partially autonomous processor nodes, which are often heterogenic. The network aims to solve a global problem through the nodes' independent processing and intercommunication. In MASSYVE, these nodes formed virtual enterprises to, among other things, base scheduling decisions on negotation instead of pre-planning, in the context of manufacturing. The problem of scheduling, a classic in manufacturing research, consists of finding a suitable assignment of manufacturing resources to specific tasks within a specific time window, while coping with a set of constraints [6]. The negotiation approach allows for dynamic reaction in case of a conflict, roughly meaning that something not previously foreseen happens, whereas a pre-planned approach must consider and include instructions for all possible circumstances or it will not be able to recover gracefully. Complexity grows as the manufacturing resources consist of several working units (robots, machinery, workers), some of which may be in repair and others uncapable of performing a given task or simply unwilling.

The basic procedure begins by the announcement of a task (an "enterprise activity") through the MAS network. After that agents representing the resources exchange information such as capabilities (if the task is doable by the agent in general) and other constraints (if the resource the agent is representing is temporarily unavailable, e.g. in repair). Eventually, an agent is selected to perform the given task [6].

The project also produced the Massyve kit, a small, interactive tool for quickly developing simple applications for multi-agent systems. The user can design reference architectures and use them to derive particular systems. The kit is available from the project homepage [1].

The project implemented a prototype system and considered it to show their approach feasible in the area of shop floor scheduling [6]. Some possible points to continue on were mentioned, one of them being handling agents that are not necessarily totally cooperative. This may be unusual for e.g. a set of paper machines working in unison, but in the general context of virtual enterprises there can be partners that have their own interests competing with global optimization, or even competitors sending false information on purpose. Similarly, negotiation with incomplete or imprecise information was discussed.

The project's publications [2, 3, 5, 4, 6] are available at the MASSYVE homepage, `http://www.gsigma-grucon.ufsc.br/massyve/`.

Bibliography

[1] The MASSYVE project homepage, 2004. URL: http://www.gsigma-grucon.ufsc.br/massyve/ [5.11.2004].

[2] RABELO, R. J., AFSARMANESH, H., AND CAMARINHA-MATOS, L. M. Applying federated databases to inter-organizational multi-agent scheduling. In *MAS'99 1st International IFAC Workshop on Multi-Agent Systems in Production, Vienna, Austria, 2-4 Dec 1999* (Dec. 1999). URL: http://www.gsigma-grucon.ufsc.br/massyve/files/MAS99-Final.zip [15.10.2004].

[3] RABELO, R. J., AND CAMARINHA-MATOS, L. M. Balanced industrial automation approaches for agents integration. Tech. rep., GSIGMA, Dec. 1998. URL: http://www.gsigma-grucon.ufsc.br/english/files/Tech-Rep-00198.zip [5.11.2004].

[4] RABELO, R. J., AND CAMARINHA-MATOS, L. M. Generic framework for conflict resolution in negotiation-based agile scheduling systems. In *IMS'98 - 5th IFAC Workshop on Intelligent Manufacturing Systems, Gramado - Brazil, November 1998* (Nov. 1998). URL: http://www.gsigma-grucon.ufsc.br/english/files/HOL-IMS98.zip [15.10.2004].

[5] RABELO, R. J., CAMARINHA-MATOS, L. M., AND AFSARMANESH, H. Multiagent perspectives to agile scheduling. In *Basys'98—IEEE / IFIP International Conference on Balanced Automation Systems, Prague, Czech Republic, 26-28/08/98* (Aug. 1998), IEEE. URL: http://www.gsigma-grucon.ufsc.br/english/files/MassBASYS.zip [14.10.2004].

[6] RABELO, R. J., CAMARINHA-MATOS, L. M., AND AFSARMANESH, H. Multiagent-based agile scheduling. *International Journal of Robotics and Autonomous Systems, Special Issue on Multi-Agent Systems Applications 27* (1999), 15–28. URL: http://www.gsigma-grucon.ufsc.br/english/files/MassJournal.zip [15.10.2004].

Appendix F

METEOR-S

The increasing interest in Web Services makes the finding of relevant ones a big challenge. One approach is to add semantics to Web Service descriptions [4]. A project called METEOR-S [2] focuses on this and its related issues.

The project is based on another project called METEOR [2] which stands for Managing End-to-End Operations. This project focused on workflow management techniques for transactional workflows. METEOR-S, like its predecessor, is a project of the Large Scale Distributed Information Systems laboratory in Computer Science department of University of Georgia. The letter S in the project name stands for Semantic Web services and Processes. The project's key issue is the use of Semantics in all stages of Semantic Web processes.

METEOR-S builds upon existing standardized Service Oriented Architecture (SOA) and Semantic Web techniques with extensibility features whenever possible [3]. In addition to this the project seeks to influence to existing standards. For example semantic extensions to WSDL have been presented by the research team.

The team has constructed frameworks for supporting all stages of the Semantic Web Process lifecycle. The stages are classified as follows:

- Semantic Annotation and Publication of Web Services
- Abstract Process Creation
- Semantic Discovery of Web Services
- Orchestration and/or Composition of Web Services.

The semantic character is present in each stage of the process but it appears in different ways.

The research team has identified four kinds of semantics related to these phases of the Semantic Web Process [1]:

- data -
- functional -
- Quality of Service (QoS)- and
- execution semantics.

Data semantics is related to Web Services communicating with each other. Functional semantics of a Web service consists of its data semantics and classification of its operations' functionality. Web Services' QoS specifications describe different forms of performance. Finally, execution

semantics are related to the late binding of services. Effort is currently been put on formalizing these concepts. The project has for example generated a formal way of describing and evaluating qualitative and quantitative aspects in stages of semantic annotation and semantic discovery of Web Services.

Bibliography

[1] AGGARWAL, R., VERMA, K., MILLER, J., AND MILNOR, W. Constraint driven web service composition in meteor-s. In *Proceedings of the IEEE SCC* (2004).

[2] Meteor project homepage, Oct. 2004. URL: `http://lsdis.cs.uga.edu/projects/METEOR/` [18.10.2004].

[3] Meteor-s project homepage, Oct. 2004. URL: `http://lsdis.cs.uga.edu/projects/METEOR-S/` [18.10.2004].

[4] SIVASHANMUGAM, K., VERMA, K., SHETH, A., AND MILLER, J. Adding semantics to web services standards. In *Proceedings of the 1st International Conference on Web Services (ICWS'03)* (June 2003), pp. 395–401.

Appendix G

PRODNET

PRODNET is a layered/transaction-oriented infrastructure for VEs. It is geared towards providing basic communications, information exchange and application integration [2]. In the following we first summarize, based on [1] (unless otherwise noted), the PRODNET architecture followed by summary of the information classification used over the database schema that provides thecommon element for VE interoperability. The complete PRODNET publications are available from the project website `http://www.uninova.pt/~prodnet`.

The architecture of PRODNET is based on a broker. One node is the *VE coordinator* (broker) and the other nodes are *VE members* that report their activities to the VE coordinator. The role of the node can be a supplier, producer or final consumer for the given VE, supporting a supply-chain topology. A node can participate to multiple VEs and multiple enterprises seem to able to participate through one node. The node architecture is provided in Figure G.1.

The figure notes three distinct infrastructure components (1-3) and two communications protocols (4-5). The infrastructure components provide the following functionality:

- The *Internal Module* (1) represents the enterprise systems that provide the required information for the node. PRODNET is geared towards product-based manufacturing industries and as such the Production Planning and Control (PPC) would be most important of these. Internal Modules exist per enterprise communicating with the given PRODNET node. As such, PRODNET is not based on deploying a node per enterprise involved.

- The *Product Cooperation Layer* (PCL) (2) provides the interoperation between PRODNET nodes and possibly between multiple Internal Modules. It contains a *Distributed Information Management System* (DIMS) that is manages queries and updates to the database schema and configuration of the views allowed to the schema. The PCL also provides a *Local Coordination Manager* (LCM) that contains a Workflow Engine (Local Coordination) for event coordination and a message parser (Protocol Handler/Message Identifier) [1]. The PCL also provides a Coordination Kernel that allows parametrization of the VE.

- The *Advanced VE Coordination Module* (3) provides additional functionality including VE partner selection, negotiation, contract definition and configuration. It also provides "advanced coordination" of VE activities.

The PRODNET Communications Infrastructure (PCI, [2]) contains communication protocols and an API for support services to handle e.g. STEP and EDIFACT. The PCI completes the cooperation layer. Two communications protocols are provided: one for node local communication

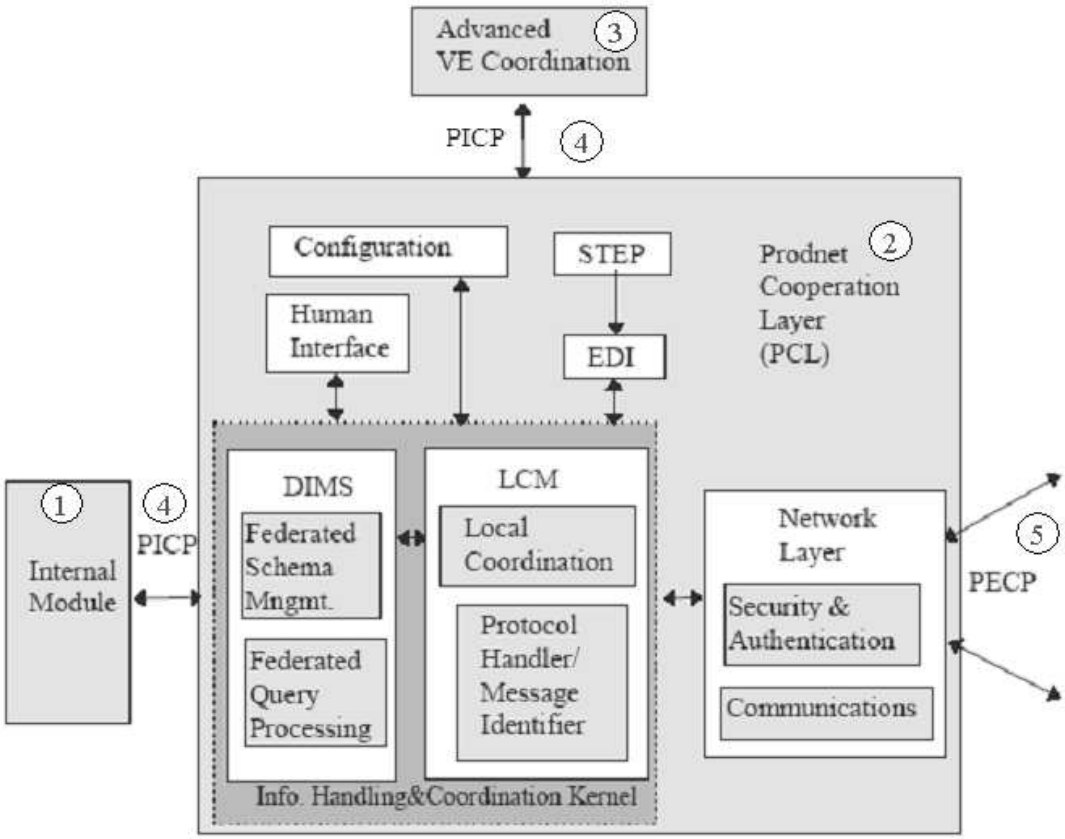


Figure G.1: PRODNET node architecture.

and the other for non-local. The *PRODNET Internal Communication Protocol* (PICP) (4) is used to manage communications between PCL and either Internal Module or Advanced VE Coordination module. The *PRODNET External Communication Protocol* (PECP) (5) is used to manage communication to other PRODNET nodes and to members outside the PRODNET VE.

Integration between PRODNET nodes (and Internal Modules) is based on a common, object-oriented database schema located in the PCL that is utilized by each node. As such the schema provides means for *federated information management* (i.e. PRODNET does not require the DBMS to be provided by a specific vendor as long as it conforms to provide the functionality required), although originally PRODNET was built on top of Oracle. This presumably does set a certain "dialect" of SQL that—on PRODNET level—may require rewriting some SQL-statements to port it to support other RDBMS (or potentially leads to restrictions with different vendor-specific "RDBMS gateway" -products). The schema itself provides a unified view of each node's PCL, describing both VE information (describing e.g. PRODNET architecture related information, VE configuration related information and directory information of nodes) and PPC information (describing e.g. clients and client orders of given products).

PRODNET categorizes information on three main levels: Self, Acquitance and VE. These describe the relationship of information to the given PRODNET node. Specific information in a specific category can be private, restricted or public. This is depicted in Figure G.2.

Acting on the information requires specifying the levels and the categories to limit the visibility of data. *Locally generated* information is categorized as self and private. This information can

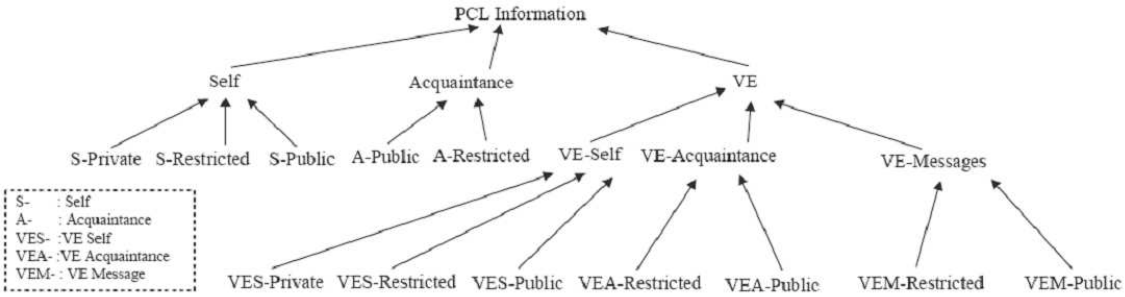


Figure G.2: PRODNET Classification of Information [3].

be *exported* to make it available for another enterprise or enterprises in the local node, an acquaintance node or the whole VE or similarly to another VE. Likewise a member can *import* exported data as long as it is exported in a fashion that makes it visible for the importer [1]. Modifications to information categorization affect the View Hierarchy Configuration of PCL [2].

In conclusion it should be noted that PRODNET has been piloted in an environment of four enterprises (two in Brazil, two in Europe). The enterprises were involved in bicycle design or manufacturing. In addition to noting non-technical issues that contributed to making the implementation non-trivial, also the PRODNET infrastructure was seen as complex to implement due large amount of configuration required for each tool and technology in-scope of integration [2]. Finally, the PRODNET project has included the use of process modelling to enable focusing on business integration fit between parties.

Bibliography

[1] AFSARMANESH, H., GARITA, C., HERTZBERGER, L., AND SANTOS-SILVA, V. Management of distributed information in virtual enterprises - the PRODNET approach. Tech. rep., Uninova, 1998. URL: `http://www.uninova.pt/~prodnet/papers.html` [27.1.2005].

[2] CAMARINHA-MATOS, L., AND AFSARMANESH, H. Elements of a base ve infrastructure. *Journal of Computers in Industry 51*, 2 (June 2003).

[3] GARITA, C., AFSARMANESH, H., AND HERTZBERGER, L. The PRODNET cooperative information management for industrial virtual enterprises. Tech. rep., Uninova, 1998. URL: `http://www.uninova.pt/~cam/ev/dims.pdf` [27.1.2005].

Appendix H

Projects at the University of Tilburg

MEMO stands for MEdiating and MOnitoring electronic commerce [1]. The goal of the project was to prototype, demonstrate, and evaluate a lightweight electronic commerce environment. The environment consists of smaller scale and diverse electronic commerce applications. Perspective for the project was given by ABN AMBRO bank which is planning to introduce a similar environment for SMEs (Small and Medium size Enterprises).

The older EDI is both limited and inherently inflexible in it's ability to enable a value chain [3]. It can be used to communicate basic information about business transactions but it cannot adapt to a rapidly changing environment and changing market conditions. To improve the situation, e-commerce platforms should include functionality for a few major elements. These are business process compatibility, adaptibility of business processes, leveraging legacy assets, support for business transactions and network security services.

Building cooperative applications is still a demanding and time consuming task. Typically low level programming is required to achieve service compositions. Main problems in composition are: large and highly dynamic service space, lack of underlying framework to describe and present services in efficient way, and multiple similar services with small differences. To solve these problems Papazoglou, Yang and Heuvel have developed a service description language called SDL. The language is designed to be able to describe the semantic relatedness (i.e. how close two services are), capability analysis (i.e. what kind of functionality the services have), and syntactic analysis (i.e. what is the syntax of the interface) of two services. The language is based on the idea of dividing large service components to the smallest possible parts. These parts are the functions that services offer. Then by combining these small parts together they are able to create more complex services. While complex services are composed from smaller parts they can be compared with each other by comparing the subservice trees and determine how close the services are syntactically and semantically [2].

Services can be composed in multiple ways. 1. Data oriented service composition. This means composing services dealing primarily with operationa data from different sources. 2. Sequential process oriented service composition. In this case the components are invoked sequentially and one service cannot start before the previous service has finished. 3. Parallel process oriented composition. In this case the services can be executed independently for each other. 4. Alternative composition in parallel or in sequence. In this case there are multiple services. These services can be executed each in turn or in parallel until the desired outcome is achieved [5].

Papazoglou and Yang have also produced a framework for managing the entire life-cycle of service compositions. They introduce the concept of *service component* that packages together complex services. Service component gives an unified and consistent presentation to interfaces and operations of the service. The life-cycle of a service component consists of abstract definition,

scheduling and construction to excecution. The planning phase assists in determining the series of service operations in order to achieve a given service request. The definition phase allows abstract composing of services. The scheduling phase will determine which services and in which order they must be executed in order to produce the wanted service. In the construction phase the actual service is composed of smaller services and is unambiquous. In the execution phase the bindings of the composite service are implemented and the service produced [4].

Bibliography

[1] MEMO project homepage, Oct. 2004. URL: http://www.uvt.nl/infolab/prj/memo/ [27.1.2005].

[2] VAN HEUVEL, W., YANG, J., AND PAPAZOGLOU, M. Service representation, discovery, and composition for e-marketplaces. In *Proceedings of 6th IFCIS International Conference on Cooperative Information Systems (CoopIS2001)* (2001), vol. 2172, Springer-Verlag, pp. 270–284.

[3] YANG, J., AND PAPAZGLOU, M. P. Interoperation support for electronic business. *Communications of the ACM 43* (June 2000), 39–47.

[4] YANG, J., AND PAPAZOGLOU, M. Service components for managing the life-cycle of service compositions, 2003. URL: http://citeseer.ist.psu.edu/yang03service.html [27.1.2005].

[5] YANG, J., VAN DEN HEUVEL, W.-J., AND PAPAZOGLOU, M. P. Tackling the challenges of service composition in e-marketplaces. In *RIDE* (2002). URL: http://citeseer.ist.psu.edu/yang02tackling.html [27.1.2005].

Appendix I

The WISE project

I.1 The project

WISE stands for Workflow based Internet SErvices. The project's final object is to try to lower the cost and effort of setting up enterprise networks between small and medium enterprises. The initial step has been the development of a model for electronic commerce. WISE introduces this model as the design principle.

I.2 The WISE model

The model defines three concepts:

1. Virtual business process
   A virtual business process is like any process but it is not tied into only one organization. It can span over multiple organizations and couple them together. The process does not need to know exactly how its subprocesses are composed. It only needs a unified interface for binding the components together.

2. Virtual enterprise
   This concept defines the context of a virtual business process. The context is a set of goals, rules, requirements, constraints and resources. Another way to define a virtual enterprise is to see it as the organization behind any given virtual business process. It is not always easy to define the organization behind such a process, because it can have the following properties: decentralized, multiple enterprises, multiple software platforms and complicated ownership.

3. Trading community
   The two previous concepts defined: what to do and where to do it. The third concept answers the question, who will do it. Trading community defines the actors for a virtual enterprise. An actor is basically every company, who is participating in the virtual business process and/or providing components for building it.

I.3 The WISE approach

Virtual business processes do exist today but they are static solutions, which means that they have been programmed case by case. The WISE project sees the real challenge in producing a software framework which would be capable of adapt dynamic changes in the process and accept new

actors own the fly. These changes in the virtual enterprise, the process and the trading community should be possible to do without programming. The reason for this new approach is the fact that the today's solution is too time consuming and expensive. Key points of the WISE approach:

1. Virtual business process life cycle
   The WISE approach defines life cycle as the base process for defining and controlling the three main concepts which are described in previous section. The life cycle includes services (building blocks for the process), the process (how to use these blocks) and finally the continuous process improvement.

2. The software platform
   The platform for executing a WISE system is build on four different specifications. The *Process definition* language to define the process and supporting environment. *Process enactment* includes the virtual process engine to execute the defined process. *Monitoring and Analysis* is for logging the entire process execution. The collected data could be used as a base for load balancing and QoS (Quality of Service), but also for process improvement. *Coordination and communication* describes the interface needed between different actors in the virtual business process. The interface should also provide a possibility for human interaction to accomplish the manual steps of the process.

I.4 The WISE architecture

The idea in WISE is to allow companies to continue using existing business modelling tools like the one included in the project (Structware's IvYFrame). The running WISE system should convert the modelling tool's representation to OCR (Opera Canonical Representation). OCR is the internal language used in the WISE system. The WISE engine runs the process defined by OCR. The engine works as any state-of-art workflow manager including persistence state information, monitoring and subsystem adapters for the different components used in the virtual business process. WISE introduces a communication chanel between parties in a process to solve possible problems with data inconsistency. This is based on another project called CoBrow [1].

Bibliography

[1] LAZCANO, A., ALONSO, G., SCHULDT, H., AND SCHULER, C. The wise approach to electronic commerce. *International Journal of Computer Science and Engineering* (2000).