# A Middleware Model in Alloy for Supply Chain-Wide Agent Interactions

Robrecht Haesevoets, Danny Weyns, Mario Henrique Cruz Torres, Alexander Helleboogh, Tom Holvoet, and Wouter Joosen

Distrinet, Katholieke Universiteit Leuven, 3001 Leuven, Belgium
robrecht.haesevoets@cs.kuleuven.be

**Abstract.** To support the complex coordination activities involved in supply chain management, more and more companies have autonomous software agents acting on their behalf. Due to confidentiality concerns, such as hiding sensitive information from competitors, agents typically only have a local view on the supply chain. In many situations, however, companies would like to expand the view of their agents to share valuable information such as transportation tracking and service delays. Non of the participating companies, however, has enough knowledge or authority to realize such interactions in a controlled manner.
In this paper, we present an organization middleware that offers a collaboration platform and enables agents to interact across the boundary of local interactions. Policies and laws enable companies to define the scope of interactions of their agents and the restrictions on their exposed information. Using Alloy, we formally define the relation between the interactions offered by the middleware, the exposed information and the provided policies and laws. This allows us to guarantee a number properties which are of particular interest to companies using the middleware.

**Keywords:** Organisations and institutions; Social and organizational structure; Verification of MAS

## 1 Introduction

In today's competitive and globalized market, streamlined collaborations between business entities are a necessity. In the DiCoMas project[1], a joint research effort with academic and industrial partners, we have been studying the use of agents for managing collaborations between business entities in the domain of supply chain management. A key objective of this project is to improve integration and collaboration among supply chain partners.

Due to company-specific restrictions, such as hiding sensitive data from competitors or having clients exchange pricing info with subcontractors, companies typically only allow their agents to participate in local supply chain interactions [14]. As a result, agents only have a local view on the supply chain. Nevertheless, in many situations companies would like to extend the view of their

[1] DiCoMas: Distributed Collaboration using Multi-agent System Architectures: http://distrinet.cs.kuleuven.be/projects/dicomas/index.html

agents and allow them to participate in supply chain-wide interactions in a controlled manner. Examples are tracking containers throughout the supply chain or monitoring problems such as delays outside the local view of agents.

A typical way to structure such interactions between agents is by means of roles and organizations [9, 1]. In previous work [17], we have presented an organization model for collaborative multi-agent systems. Although the model is relatively simple, it is powerful enough to model controlled supply chain-wide interactions. A subset of the model is shown in Fig. 1. The core abstractions of the model are organization, role, and capability. Organizations, defined as a set of roles, specify the boundaries in which controlled interactions can take place. A role represents a concrete participation in the organization. It defines the agents that have access to the organization, and it defines the capabilities these agents have in the organization. Each capability represents a concrete interaction ability relative to another role in the organization.

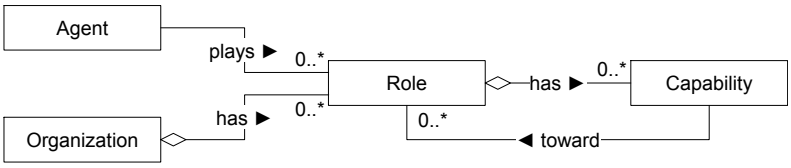


**Fig. 1.** A visual representation of the organization model.

Realizing organizations and managing their dynamics in a heterogeneous and distributed supply chain setting is a very complex task, for which none of the participating companies has enough authority or knowledge. Additionally, companies want guarantees before exposing confidential information or allowing their agents to collaborate with external parties.

To address these challenges we present an organization middleware approach. The middleware offers organizations and roles as a set of reusable programming abstractions to application developers. At run-time, the middleware realizes a collaboration platform. Agents provide the middleware with local information on the supply chain, and in return, the middleware offers managed organizations that enable agents to engage in supply-wide interactions in a controlled way. Companies can specify interaction laws to define the desired scope of interactions for their agents and a set of policies to restrict the information they expose, in order to deal with confidentiality concerns. These laws and policies will then be enforced by the middleware.

The use of organizational abstractions together with a middleware has a number of key benefits: (1) it allows to represent and structure supply chain-wide interactions at a high-level of abstraction; (2) it allows to separate the management of dynamic supply chain-wide interactions, performed by the middleware, from the actual functionality, provided by agents participating in the

interactions; (3) it allows to accurately restrict the interactions between agents according to provided policies in terms of capabilities.

The contributions of this paper are:

1. We motivate and specify a set of concrete requirements for supply chain-wide interactions in the domain of logistics for supply chain management.
2. We present a formal model in the Alloy specification language [8] of an organization middleware supporting supply chain-wide interactions. The model formally defines the relation between supply chain-wide interactions enabled by the organizations offered by the middleware and the local supply chain information exposed by the agents and the provided policies.
3. We assert a number of relevant properties offering companies formal guarantees in terms of confidentiality using the model and the Alloy Analyzer.

**Overview of this paper.** Section 2 introduces a running example together with a set of requirements for supply chain-wide interactions. The organization middleware is presented in Sect. 3 and illustrated in the running example. Section 4 presents the middleware model in Alloy and shows how the Alloy Analyzer can be used to assert a number of properties. Finally, related work is discussed in Sect. 5, and Sect. 6 concludes and reflects on future work.

## 2 Logistics in Supply Chain Management

In the domain of supply chain management, companies usually outsource their logistic activities to one or more specialized third-party logistics providers (3PL). To integrate and streamline the operations of different 3PLs, an extra level of outsourcing can be introduced, called fourth-party logistics providers (4PL). Figure 2 shows an example of a hierarchical outsourcing structure in a supply chain, used as a running example in this paper. In the example, several companies collaborate to realize the logistic needs of company 0. Company 0 has an outsourcing contract with company 1, which as acts as a 4PL and integrates the services of two 3PLs, company 2 and 3. Company 2, in turn, has two additional subcontractors, company 4 and 5. In the example, company 3 is currently carrying a container of company 0, and company 4 and 5 are expecting a delay.

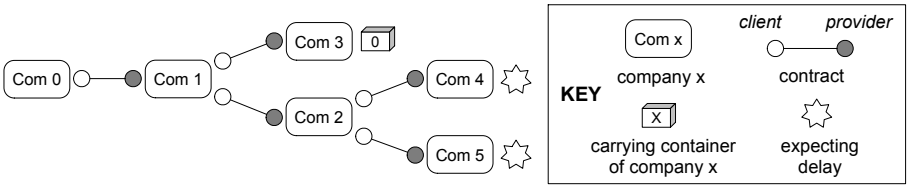


**Fig. 2.** Supply chain collaborations.

Due to confidentiality concerns, companies only allow their agents to participate in local interactions corresponding to active outsourcing contracts. As a

result, agents only have a local view on the supply chain. Typical supply chain flows, such as information and services, are propagated through the supply chain based on local interactions. In the DiCoMas project, we aim to enhance the integration and collaboration of the supply chain partners to improve information sharing and responsiveness. To realize this, agents acting on behalf of companies need extended views on the supply chain and have to interact across the supply chain in a controlled way. We give a number of concrete stakeholder requirements that motivate the need for supply chain-wide interactions. For clarity, the requirements are explained in the context of the running example.

**Collaborative Planning.** To create a planning in correspondence with the individual goals of each stakeholder, company 1 wants to use a collaborative planning approach. This requires agents of both clients, such as company 0, and subcontractors, such as company 2 and 3, to participate in coordinated planning and negotiation activities, while company 1 maintains a supervising position and can enforce the necessary restrictions on the involved interactions.

**Traceability.** Company 0 wants to track the location and status of its containers throughout the supply chain. Instead of having to contact its service provider, company 1, who in turn has to contact other service providers, company 2 or 3, and so on, company 0 requires it agents to directly interact with the agents of the current carriers of its containers, increasing responsiveness and reducing overhead. Using policies, intermediate companies such as company 1 should be able to restrict the information that can be exposed to company 0.

**Improved Responsiveness in Case of Problems.** As a 4PL, Company 1 wants its agents to be directly informed by agents managing third-party resources when serious problems occur, such as delays or decommitment. This enables company 1 to anticipate future problems at a supply chain-wide level and offer its clients a higher quality of service. Intermediate companies should be able to restrict the information exposed by their subcontractors.

## 3 The Organization Middleware

The previous section introduced a number of stakeholder requirements that underpin the need for supply chain-wide interactions. Such interactions can be modeled and coordinated using organizational abstractions we introduced in [17]. In this section we present an organization middleware that offers such organizations and roles as a set of reusable programming abstractions to application developers. At runtime, the middleware provides a collaboration platform and takes the responsibility of managing organizations and their dynamics, for which non of the partners in a supply chain has enough authority or knowledge.

Figure 3 gives a high-level overview of the approach. To participate, agents of supply chain companies have to provide the middleware with context information and a set of interaction laws. In return, the middleware offers agents a broader view on the supply chain and support for supply chain-wide interactions, while taking the responsibility of managing the interactions and their dynamics. Using a middleware allows us to separate the management of the organizations from

the agents, who can now focus on realizing the functionality in organizations. Internally the middleware can be realized using different technologies including agents. Agents using the middleware have to conform to certain communication standards, which are outside the scope of the current model.

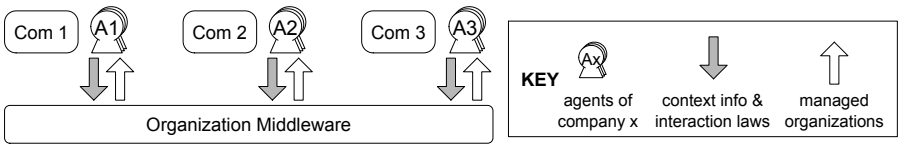


**Fig. 3.** High-level overview of the approach.

In the remainder of this section we first explain the notions of context and interaction laws in more detail. We then show how context and laws can be used by the middleware to offer organizations that enable controlled supply chain-wide interactions in the running example.

### 3.1 Context Information and Interaction Laws

Agents have to provide the middleware with local information on the supply chain, consisting of context and interaction laws. The completeness of the context depends on the amount of information exposed by the agents on behalf of the companies. Context includes information on companies, their dynamic properties, such as containers currently carried or expected delays, the current outsourcing contracts between companies, and a set of flow policies. Flow policies define the allowed supply chain flows between agents of particular companies. We currently consider two types of flows: information flow and service flow. These allow companies to specify which information exchange and which concrete service provision can take place between which specific companies. Flow policies are specified at the level of outsourcing contracts as allowed flows within outsourcing contracts as well as between different contracts. An example is shown in Fig. 4, illustrating how flow policies of different companies create a graph-like structure defining the allowed information and service flows at a supply chain-wide level.

Interaction laws allow companies to define in a declarative way the desired scope of the supply chain-wide interactions for their agents. In particular, an interaction law specifies a desired set of interaction partners whose agents should be allowed to participate in the interaction, such as "all providers of a company" or "all companies carrying a specific container", as well as the supply chain flows the interaction should enable between these partners.

### 3.2 Realizing Supply Chain-Wide Interactions.

The middleware uses the interaction laws together with the current context to provide a set of organizations supporting the desired supply chain-wide interactions. Each organizations enables a set of interactions, defined by the capabilities

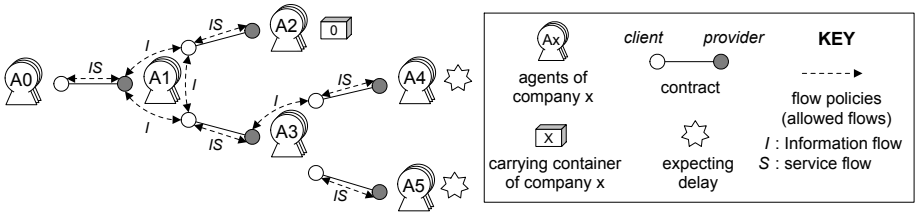


**Fig. 4.** Context consisting of flow policies and outsourcing contracts.

of its role and each capability enables a specific supply chain flow toward another role in the organization in correspondence with the current flow policies. As context or laws change, the middleware adapts the organizations accordingly.

Figure 5 illustrates a set of organizations realizing the requirements for supply chain-wide interactions introduced in Sect. 2 for the running example. Organization 1 illustrates collaborative planning, enabling the agents of client 0 to exchange planning information with the agents of subcontractors 2 and 3. Role capabilities, compliant with the flow policies, show that company 1, as a 4PL, remains in a supervising position, ensuring clients have no capabilities to make any direct service requests to subcontractors. Organization 2 shows the tracking of a container throughout the supply chain, enabling the agents of company 0 to interact with the carrier of their container, the agents of company 3. Improved responsiveness is exemplified by organization 3, allowing agents of company 1 to interact with the agents of company 4, which is expecting a delay. Because company 2 wants to hide its internal outsourcing strategy, it does not allow any flows between company 5 and other parties, as illustrated in Fig. 4. As a result, company 5 is excluded from organization 3, although it is also expecting a delay.

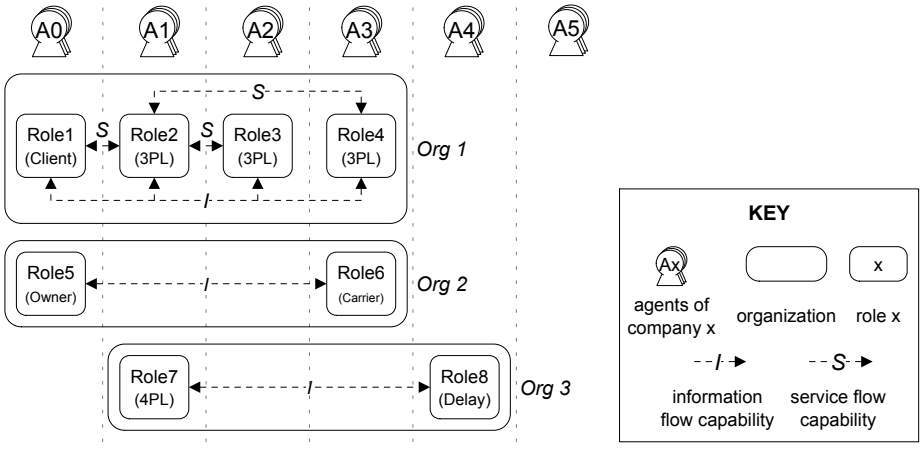


**Fig. 5.** Examples of organizations and roles realizing supply chain-wide interactions.

## 4 Middleware Model in Alloy

In this section we give a formal model of the middleware abstractions using the Alloy specification language. Alloy [8] is a structural modeling language based on first-order logic for expressing complex structural constraints and behavior in software systems. The Alloy Analyzer[2] is a constraint solver, supporting automatic simulation and checking of Alloy models within a specific scope. Simulation consists of finding instances satisfying a specification, while checking consists of finding counter examples violating certain assumptions about a model. The Alloy analysis is based on the notion of *small scope hypothesis* [8], assuming that assertions checked within a well-chosen scope will also hold for larger scopes. However, with a well-chosen scope and model, it can even be possible to do a complete analysis for a specific setting.

The purpose of our formal model in Alloy is threefold: (1) present a rigorous specification of the main concepts of the organization middleware; (2) formally define which supply chain-wide interactions the middleware can and should provide, given the context and a set of interaction laws; (3) show how this model can be used together with the Alloy analyzer to guarantee a number of properties in terms of confidentiality constraints.

### 4.1 Middleware Model

The middleware model is shown in Spec. 1. Some parts of the model are omitted, but can be found in Appendix A. An executable version of the model is also available for download[3]. Every concepts is represented by *signature*. In Alloy, a signature introduces a new set of atoms in the universe (*univ*) of the model (the universal set *univ* contains all atoms of the model).

**Context Information.** Context information consists of information on companies, their dynamic properties and their flow policies. We start by defining the signatures *Company* and *Contract* to represent companies and their outsourcing contracts. *Company* has one *field*, named *properties*, mapping each company to a set of properties, defined by the signature *Property*. *Contract* has three fields, two disjunct companies representing the client and provider in the contract, and a field *flows* mapping each contract to the set of supply chain flows that are allowed to take place within the contract. Supply chain flows are defined by the signature *Flow*. Subtypes *Info* and *Service* represent some of the typical supply chain flows, but more expressive subtypes can be introduced.

On line 11 the signature *context* defines the context of the middleware as a set of companies, contracts and flow policies. Flow policies are defined on line 14[4] as ternary relations which specify the allowed flows between different contracts.

[2] Alloy Analyzer 4 - `http://alloy.mit.edu/alloy4/`

[3] `http://people.cs.kuleuven.be/~robrecht.haesevoets/AOSE2010/`

[4] The field *flowPolicies* can refer to multiple flow policies. The Alloy syntax does not require the *set* keyword for relations.

**Specification 1** Middleware Model (partial)

```
sig Company{
  properties:set Property
}
sig Contract{
  disj client,provider:Company,
  flows:set Flow
}
sig Property{}
abstract sig Flow{}
one sig Info,Service extends Flow{}
sig Context{
  companies:set Company,
  contracts:set Contract,
  flowPolicies:Flow->contracts->contracts
}{
  all c1,c2:Contract | c1->c2 in flowPolicies[univ] implies
    some c1.(client+provider) & c2.(client+provider)
}
fun allowedFlows[context:Context]:Flow->Company->Company{
  {flow:Flow,com1,com2:Company | some c1,c2:context.contracts |
    flow in c1.flows & c2.flows and
    com1+com2 in (c1+c2).(client+provider) and
    c2 in c1.*(flows.flow<:context.flowPolicies[flow]:>flows.flow)}
}
sig Law{
  scope:Flow->Company->Company
}
fun propertyBasedSelection[p:Property, vp:Company, context:Context]:set Company{
  {c:Company | p in c.properties and Info->c->vp in allowedFlows[context]}
}
sig Role{
  company:Company,
  capabilities:Role->Flow
}
sig Organization{
  roles:set Role
}
fun enabledFlows[org:Organization]:Flow->Company->Company{
  {flow:Flow,com1,com2:Company | some r1,r2:org.roles |
    r1.company = com1 and r2.company = com2 and r2->flow in r1.capabilities}
}
sig MiddlewareModel{
  context:Context,
  laws:set Law,
  orgs:set Organization
}{
  enabledFlows[orgs] = laws.scope & allowedFlows[context]
}
```

For example, `Info->c1->c2` represents a flow policy allowing information to flow from contract `c1` to contract `c2`. A signature fact on line 16[5] introduces an additional constraint to ensure companies can only define flow policies between their own contracts. We also define a help function *allowedFlows* on line 19[6] which returns the supply chain flows that are allowed between companies by the contracts and flow policies in the given context.

**Example:** Specification 2 shows an example of context corresponding to the setting shown in Fig. 4. There are six companies and five contracts between these companies. All contracts allow information and service flows. *Company2* is carrying a container of *Company0* and both *Company4* and *Company5* are expecting a delay. The context also specifies a set of flow policies that allow information to flow between *Contract01* and *Contract12*, *Contract01* and *Contract13*, *Contract12* and *Contract13*, and *Contract13* and *Contract34*.

**Specification 2** An example of context specified in the middleware model.

```
1  ...
2  one sig Context1 extends Context{
3  }{
4    companies = Company0 + Company1 + Company2 + Company3 + Company4 +
5                Company5
6    contracts = Contract01 + Contract12 + Contract13 + Contract34 +
7                Contract35
8    Contract01.client = Company0 and Contract01.provider = Company1
9    Contract12.client = Company1 and Contract12.provider = Company2
10   ...
11   Contract01.flows = Info + Service
12   Contract12.flows = Info + Service
13   ...
14   Company2.properties = CarryingContainerOfCompany0
15   Company4.properties = ExpectingDelay
16   Company5.properties = ExpectingDelay
17   flowPolicies = Info->Contract01->Contract12 +
18                  Info->Contract12->Contract01 +
19                  Info->Contract01->Contract13 +
20                  Info->Contract13->Contract01 +
21                  Info->Contract13->Contract34 +
22                  Info->Contract34->Contract13
23 }
```

[5] The *box join* `a[b]` is the equivalent of the relational join `b.a`. The `+` sign represents the union of two sets while the `&` sign represents the intersection.

[6] The set comprehension `{a: A | constraint}` returns all elements of $A$ satisfying the given constraint. `*a` represents the reflexive transitive closure. `<:` and `:>` represent the domain and range restriction of a relation.

**Interaction Laws.** Interaction laws are represented by the signature *Law* on line 25. The field *scope* specifies the desired scope of interaction, as the set of supply chain-wide flows the interaction should enable between companies. To represent a meaningful scope of interaction, functions can be used which use the current context as input. An example is the property-based selection function on line 28, which returns all companies having a given property $p$ and that are visible from the given viewpoint $vp$.

**Example:** Specification 3 shows an example of three laws, corresponding to the organizations shown in Fig. 5. *Law1* specifies an interaction scope between *Company0*, *Company1*, *Company2* and *Company3*. *Law2* uses the property-based selection function to specify an interaction scope between *Company0* and the companies carrying its containers. *Law3* specifies an interaction scope between *Company1* and the companies expecting a delay.

**Specification 3** An example of interaction laws specified in the middleware model.

```
1  ...
2  one sig Law1 extends Law{
3  }{
4    scope = (Info+Service)->Company0->Company1 +
5            (Info+Service)->Company1->Company0 +
6            Info->Company0->Company2 +
7            ...
8  }
9  one sig Law2 extends Law{
10 }{
11   scope = Info->propertyBasedSelection[CarryingContainer,
12                                        Company0,
13                                        Context1]->Company0 +
14           ...
15 }
16 one sig Law1 extends Law{
17 }{
18   scope = Info->propertyBasedSelection[ExpectingDelay,
19                                        Company1,
20                                        Context1]->Company1 +
21           ...
22 }
```

**Roles and Organizations.** Roles and organizations are defined on lines 31 and 35. Each role has a field *company*, mapping the role to the company whose agents are allowed to play the role, and a field *capabilities*, representing the capabilities of the role in terms of supply chain flows allowed toward other roles in the organization. Organizations contain the field *roles* representing the current

roles of the organization. We also define a help function *enabledFlows* which returns the flows between companies that are enabled by a given organization.

**Example:** Specification 4 shows a specification of the organizations in Fig. 5. Each role has a company and a set of capabilities. For example, *Role1* is played by *Company1* and has capabilities for information and service flow with *Role2*, and capabilities for information flow with *Role3* and *Role4*.

**Specification 4** An example of organizations in the middleware model.

```
one sig Role1 extends Role{
}{
  company = Company0
  capabilities = Role2->(Info+Service) +
                 (Role3+Role4)->Info
}
...
one sig Org1 extends Organization{
}{
  roles = Role1 + Role2 + Role3 + Role4
}
one sig Org2 extends Organization{
}{
  roles = Role5 + Role6
}
one sig Org3 extends Organization{
}{
  roles = Role7 + Role8
}

```

**Middleware Model** The state of the middleware is represented by the signature *MiddlewareModel* on line 42. This state is defined as the current context and interaction laws, and the organizations offered by the middleware. A signature fact on line 47 uses the two help functions, we defined earlier, to specify the relation between the organizations offered by the middleware and the current context and interaction laws. The fact specifies that organizations offered by the middleware should enable those, and only those, supply chain flows between companies that are both defined by the scope of the interaction laws and allowed within the current context and its flow policies.

**Example:** Specification 5 the specification of a middleware model with the context and laws we specified in the previous examples. The application of the laws to the context results in a set of organizations (`Org1 + Org2 + Org3`), which were illustrated in the previous example.

**Specification 5** An example of a specific middleware model.

```
one sig MiddlewareModel1 extends MiddlewareModel{
}{
  context = Context1
  laws = Law1 + Law2 + Law3
}
```

### 4.2 Asserting Properties

Using the Alloy Analyzer, we can check a number of useful properties of our model. We focus on two relevant properties: (1) asserting that the middleware only offers organizations compliant with the current context; (2) asserting that companies *can* put forward a number of confidentiality constraints, by restricting the supply chain flows in the outsourcing hierarchy. The Alloy specification of these properties is shown in Spec. 6[7]. Both properties have been checked by the Alloy analyzer within a scope of 6 atoms for each type. Although this scope is limited, it covers more than all the possibilities in our running example.

The first property states that companies always need some direct or indirect contractual link, known to the middleware, before their agents can participate in any supply chain-wide interaction. The second property states that a company (*com3*) can restrict all supply chain-wide interactions between any two companies (*com1* and *com2*) that do not have a direct or indirect contractual link with each other independent from the restricting company (*com3*). This property ensures, for example, that 3PLs, such as company 2 in Fig. 4, can restrict the information their subcontractors can expose, such as company 4 and 5. In the example, company 2 allows company 4 to expose information in supply chain-wide interactions, but restricts this for company 5. As a result, the agents of company 1 can participate in an interaction with the agents of company 4, expecting a delay, but not with the agents of company 5, also expecting a delay.

## 5 Related Work

The approach presented in this paper intersects with several domains of related work. We focus on a number of representative approaches for business to business (B2B) integration in supply chain management, roles and organizations, organization middleware and formal methods for organizations in multi-agent systems.

**B2B Integration in Supply Chain Management.** Several approaches have been proposed to address the integration of business processes in supply chain management. Preist et al. [13] recognize the problems of setting up

[7] contractPath[com1,com2,context] returns true if a path from com1 to com2 exists in the contractual structure of the given context. indepContractPath[com1,com2,com3,context] returns true if a path exists independent from com3.

**Specification 6** Properties

```
check property1{
  all mw:MiddlewareModel, disj com1,com2:Company |
    !contractPath[com1,com2,mw.context] implies
    no role1,role2:mw.orgs.roles | role1.company = com1 and
    role2.company = com2 and role2 in role1.capabilities.univ
} for 6
check property2{
  all mw:MiddlewareModel, disj com1,com2,com3:Company |
    !indepContractPath[com1,com2,com3,mw.context] and
    (all c1,c2:(client+provider).com3 |
      no Flow->c1->c2 & mw.context.flowPolicies) implies
    no r1,r2:mw.orgs.roles | some r2->Flow & r1.capabilities
      and r1.company = com1 and r2.company = com2
} for 6
```

interactions between agents of different supply chain partners, and propose a Web service architecture providing automated B2B integration. Stefansson [15] stresses the importance of automated information sharing in supply chains, but also states the lack of scientific research covering the management of information flows within supply chains. Projects, such as CrossFlow [5], have explored the integration of business process between outsourcing partners using cross-organizational workflow management and virtual organizations. In contrast to the work presented in this paper, these approaches typically focus on the local integration of business processes, lacking explicit support for setting up and managing supply chain-wide interactions.

**Roles and Organizations.** Roles and organizations are generally acknowledged as valuable abstractions to structure complex interactions [9, 1]. Two particular lines of related research are electronic institutions [4] and Law-Governed Interactions [10]. Both approaches use laws, norms or policies to govern interactions among agents. Most of the existing approaches, however, put the responsibility of managing organizations with the agents, such as AGRE [3] and TuCSoN [11]. The organization middleware presented in this paper encapsulates the management of organizations as a reusable service. This greatly enhances the portability of our approach and can reduce the complexity of developing and maintaining the agents themselves. An interesting approach to support the development of the organization middleware is the A&A meta-model proposed by Omicini et al. [12].

**Organization Middleware.** A number of approaches propose middleware-supported organizations and interactions, such as AMELI [2], S-moise+ and ORA4MAS [7], and Law-Governed Interactions [10]. However, most other approaches take an agent-centric perspective in which agents are responsible for performing the functions in organization and managing life cycle of organizations. Novelty toward e-institutions and norm-based approaches is two-folded: (1) Flow policies can specify *local restrictions* on agent interactions. E-institutions

and norm-based approaches typically use global norms rather than company-specific and context-aware restrictions. (2) Implementations of norm-based approaches often rely on central entities enforcing norms, e.g. managers in AMELI and S-Moise+. Our model could also support decentralized realizations [18].

**Formal Methods for Organizations.** Formalization is recognized as a foundation for analyzing properties such as structure and stability of organizations [1, 16]. Most approaches focus on theoretical aspects of organizations, relying on heavyweight formal methods. Grossi et al. [6], for example, represent organizations as multi-graphs. By adding formal semantics to the graphs, different organizational structures can be compared in terms of performance, flexibility and efficiency. In this paper, we presented a model in Alloy and focused on the management of organizations and domain specific concerns, such as confidentiality. Because Alloy is limited, both in terms of expressiveness and the ability to analyze complex models, alternative approaches such as temporal logic and Petri nets may be more appropriate to explore run-time issues of organizations or complex interaction protocols.

## 6 Conclusions and Future Work

We have made the case for using an organization middleware to support supply chain-wide interactions in the domain of supply chain management. The organization middleware realizes a collaboration platform and offers organization and role as reusable abstractions to enhance the integration of different business processes. Although we applied our approach to a specific case in logistics management, we have shown how a limited set of organizational abstractions and a light-weight formal modeling language can be used to offer formal guarantees in terms of confidentiality constraints, such as the ability of companies to restrict the interactions between their subcontractors. These guarantees can contribute in establishing the trust of companies in such a middleware approach.

The organizational abstractions, used by the middleware, have proved powerful enough to structure supply chain-wide interactions at a high-level, and enable the separation of managing the interactions and their dynamics from providing the actual functionality provided in the interactions itself. But most importantly, they allow to accurately restrict the interactions among agents, according to company-specific confidentiality constraints.

A prototype implementation of the middleware is also available on the web[8], showing a visual representation of the approach within a controlled setting. Using a web-based GUI, users are able to set up a number of supply chain-wide interactions and dynamically alter the context, flow policies and laws.

**Future work.** A number of concerns are not addressed by our current model such as dealing with incomplete and incorrect information, security and authentication, and explicit support for interaction protocols, such as automated auctions. Other interesting future directions include a domain specific policy language and integrating the model into a development process.

[8] `http://people.cs.kuleuven.be/~robrecht.haesevoets/AOSE2010/`

## Acknowledgement

This research is supported by the Foundation for Scientific Research in Flanders (FWO-Vlaanderen), the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and the Research Fund K.U.Leuven.

## References

1. V. Dignum. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Information Science Reference, 2009.
2. M. Esteva, B. Rosell, J. Rodriguez-Aguilar, and J. Arcos. Ameli: An Agent-Based Middleware for Electronic Institutions. In *AAMAS'04*, pages 236–243. IEEE Computer Society Washington, DC, USA, 2004.
3. J. Ferber, F. Michel, and J. Baez. AGRE: Integrating environments with organizations. In *First International Workshop on Environments for Multi-Agent Systems*, volume 3374 of *LNCS*, pages 48–56, New York, NY, USA, 2005. Springer-Verlag.
4. A. Garcia-Camino, P. Noriega, and J. .Rodrguez-Aguilar. Implementing norms in electronic institutions. In *AAMAS '05*, pages 667–673, New York, NY, USA, 2005. ACM Press.
5. P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: Cross-organizational workflow management in dynamic virtual enterprises. *Computer Systems Science and Engineering*, 15(5):277–290, 2000.
6. D. Grossi, F. Dignum, V. Dignum, M. Dastani, and L. Royakkers. Structural aspects of the evaluation of agent organizations. *LNCS*, 4386:3, 2007.
7. J. Hübner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. *Autonomous Agents and Multi-Agent Systems*, pages 1–32.
8. D. Jackson. *Software Abstractions: logic, language, and analysis*. The MIT Press, 2006.
9. N. R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.
10. N. Minsky and V. Ungureanu. Law-Governed Interaction: A Coordination and Control Mechanism for Heterogeneous Distributed Systems. *ACM TOSEM*, 9(3), 2000.
11. A. Omicini and A. Ricci. Reasoning about organisation: Shaping the infrastructure. *AI* IA Notizie*, 16(2):7–16, 2003.
12. A. Omicini, A. Ricci, and M. Viroli. Artifacts in the A&A Meta-Model for Multi-Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 17(3):432–456, 2008.
13. C. Preist, J. Esplugas-Cuadrado, S. Battle, S. Grimm, and S. Williams. Automated business-to-business integration of a logistics supply chain using semantic web services technology. *LNCS*, 3729:987, 2005.
14. H. Stadtler. Supply chain management and advanced planning: basics, overview and challenges. *European Journal of Operational Research*, 163(3):575–588, 2005.
15. G. Stefansson. Business-to-business data sharing: A source for integration of supply chains. *International Journal of Production Economics*, 75(1-2):135–146, 2002.
16. E. Van Den Broek, C. Jonker, A. Sharpanskykh, J. Treur, and P. Yolum. Formal modeling and analysis of organizations. *LNCS*, 3913:18, 2006.

17. D. Weyns, R. Haesevoets, A. Helleboogh, and Holvoet. The macodo organization model for context-driven dynamic agent organzations. *ACM Transaction on Autonomous and Adaptive Systems*, 6(4), 2010.
18. D. Weyns, R. Haesevoets, A. Helleboogh, T. Holvoet, and W. Joosen. The MACODO Middleware for Context-Driven Dynamic Agent Organzations. *ACM Transaction on Autonomous and Adaptive Systems*, 5(1):3:1–3:29, 2010.

# A Omitted Parts of the Middleware Model

```
1  fun contractualLinks[context:Context]:Company->Company{
2    {disj com1,com2:Company |
3      some c:context.contracts | com1+com2 in c.(client+provider)}
4  }
5
6  pred contractualPath[com1,com2:Company, context:Context]{
7    com2 in com1.*(contractualLinks[context])
8  }
9
10 pred indepContractualPath[c1,c2,dependence:Company,context:Context]{
11   let indepContractualLinks =
12     (Company-dependence)<:contractualLinks[context]:>(Company-dependence) |
13   c2 in c1.*indepContractualLinks
14 }
```