Supporting Virtual Organisations using BDI Agents and Constraints

Stuart Chalmers, Peter M.D. Gray, and Alun Preece

University of Aberdeen;Kings College, Aberdeen, AB24 3UE
{schalmer,pgray,apreece}@csd.abdn.ac.uk

Abstract. Virtual organisations underpin many important activities in distributed computing, including e-commerce and e-science. This paper describes a new technique by which software agents can intelligently form virtual organisations to meet some pre-specified requirements. Our approach builds on work in BDI agents and constraint satisfaction techniques. Using a realistic service-providing scenario, we show how an agent can use constraint solving techniques to explore possible virtual organisation alliances with other agents, based on its beliefs and desires. The agent can choose the best among several possible virtual organisations to form in order to meet a customer's requirements. Our approach is to use a deliberative process to construct possible worlds, each corresponding to a potential virtual organisation, and each configured using constraint satisfaction techniques. We also show how an agent can take account of pre-existing virtual organisation relationships in its deliberations.

1 Introduction

Support for virtual organisations is emerging as one of the most significant requirements in modern distributed computing, underpinning many activities viewed as of enormous strategic importance to industry and government. These activities include business-to-business e-commerce, where virtual organisations enable electronic value chains [1], and e-science, where virtual organisations are seen as the fundamental organising principle of the "grid" [9]. The key aspects of virtual organisations that make them such an attractive paradigm are:

- they are composed of a number of autonomous entities (representing distinct individuals, departments, organisations, etc) each of which has a range of problem solving capabilities and resources at their disposal;
- the entities co-exist, collaborate, and sometimes compete with one another in a ubiquitous virtual space (representing a marketplace, meeting room, laboratory, etc);
- service-providing entities may advertise their capabilities to their peers, and then enter into service agreements or contracts with service-requiring entities;
- where appropriate, groups of entities may form a coalition in order to provide some amalgamated service, or carry out some overall activity cooperatively;

– all of the above aspects are highly dynamic: entities may come and go, capabilities may change over time, and coalitions may form, dissipate, and reform.

In the KRAFT project[1] we developed an architecture that offered limited support for virtual organisations, through the following main features [18]:

– entities are represented by *software agents*, which interact with one another in a virtual space defined by *shared communication protocols*;
– an individual entity (agent) may view the world in terms of a *local ontology* (or data model), but to communicate with other entities it must align this local ontology with a *shared common ontology*;
– coordinated activity is achieved by the exchange of *quantified constraints* between entities, and the solving of sets of these constraints; these constraints represent both requirements to be satisfied, and candidate solutions.

Although the KRAFT architecture has been successfully demonstrated in a realistic business-to-business e-commerce scenario [7] it lacks a number of features necessary for adequate support of virtual organisations:

– KRAFT supports very limited agent autonomy, as it assumes the entities are basically cooperative — in particular, *competition* between agents is not supported, because the agents do not have explicit agendas that may be in conflict;
– KRAFT does not allow for *negotiation* between entities over requirements and candidate solutions — there is no general mechanism for relaxing or trading-off constraints in the constraint-solving process.

In this paper, we describe how we are extending the KRAFT architecture to address these limitations, by incorporating recent work using constraint-solving within a *BDI (Beliefs, Desires, Intentions)* agent framework [4]. By adding BDI layers to the KRAFT agents, we aim to support both cooperative and competitive behaviours, and allow agents to negotiate over points of conflict. We have sought to integrate the BDI approach with our constraint-solving approach to coordinated agent interaction, in order to combine the benefits of both. In the context of supporting virtual organisations:

– *beliefs* are the state of the world, as modelled locally by an entity (agent) in the virtual organisation;
– *desires* are the goals of an entity in the virtual organisation;
– *intentions* are the actions an entity plans to carry out.

This paper reports on work-in-progress, specifically on the *formation* of a virtual organisation using BDI agents and constraint satisfaction techniques. In a realistic service-providing scenario, we aim to show how an agent can use constraint solving techniques to explore possible virtual organisation alliances with other agents, based on its beliefs and desires. We also show how an agent can take

[1] http://www.csd.abdn.ac.uk/research/kraft

account of pre-existing virtual organisation relationships in its deliberations.
The paper is organised as follows. Section 2 describes our motivating scenario. Section 3 describes the role of constraints in the BDI model, Section 4 describes the design and implementation of the constraint based architecture and Section 5 describes ongoing work on constraint relaxation.

2 Motivating Scenario

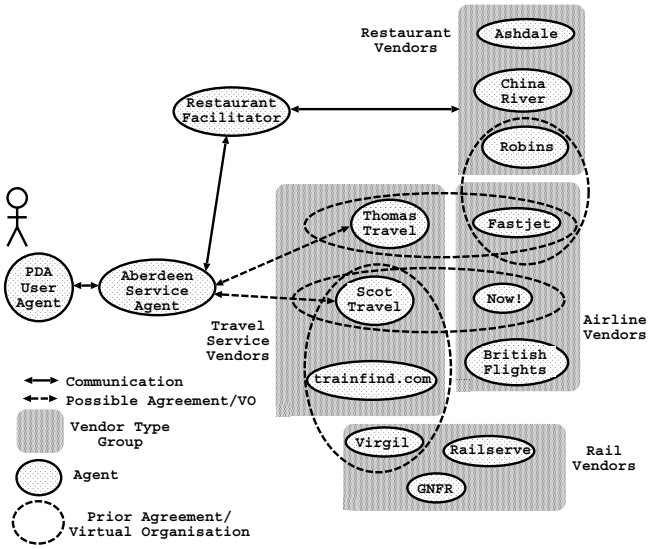

Fig. 1. *Motivating Scenario. The Aberdeen Service Agent has to decide whether to enter into a Virtual Organisation relationship with either Thomas Travel or Scot Travel.*

A lecturer is visiting Aberdeen University for a research meeting. He wishes to travel up from London on the train and arrive in time for his meeting at 2pm on Thursday afternoon. He also wants to eat at a restaurant and travel home that evening on the overnight sleeper train. He would also prefer the cheapest deal, as his funds are extremely limited!
He gives this information to a user agent on his PDA, specifying the following desires:

- Arrive in Aberdeen by 2pm Thursday
- Return to London by 8am Friday
- Eat at a restaurant on Thursday evening
- Get cheapest deal possible
- Travel by train *preferable*

In this scenario we have a number of vendors of different types, which can interact and form *virtual organisations* to provide services and facilities. From the diagram we can see the vendors grouped into service types (airline, rail, etc.) and that some have prior agreements with other vendors (e.g. a relationship exists between Scot Travel and Now!). In this situation the *Aberdeen service agent*, from the information given by the user's PDA, as well as from any relevant information available on other vendors, has to decide on the choice of vendor to enter into a virtual organisation relationship with, that will satisfy as many of the user specified desires as possible. This relationship is then in place for other such requests, or can be renegotiated if any other requirements or services are needed.
In Fig.1 the user agent contacts the *Aberdeen service agent*, which is able to gather information from various resources on local Aberdeen restaurants, and is able to communicate with various *mediator service agents* who can provide possible travel solutions to and from Aberdeen, by consulting train and airline agents for information. There also exists a *restaurant facilitator* which can recommend and book restaurants in the Aberdeen area.
The mediator service agents, Thomas Travel and Scot Travel are asked by the

Name	Travel Type	Price	Arrive	Depart
Now!	Air	€70	3:00PM	9:30PM
trainfind.com	Sleeper Train	€85	10:00AM	11:30PM
Fastjet*	Air	€90	1:00PM	9:30PM

Table 1. Options from the travel mediator agents (* indicates 25% off Robins restaurants)

Aberdeen service agent to provide travel details for return journeys from London to Aberdeen. It is their task to find a suitable deal for the service agent.
For Aberdeen/London flights, Thomas Travel are partners of a *virtual organisation* and have negotiated a deal with Fastjet, while Scot Travel have formed two relationships, the first for Aberdeen return flight deals with Now! and the second with trainfind.com and Virgil trains for sleeper services to Aberdeen. Solutions returned from the three mediators are shown in Table 1. After initial contact with restaurant facilitator about restaurant availability in Aberdeen, it receives the options shown in Table 2. From this information the service agent builds

Name	Price	Bookable From:
China River	€40	Fully Booked
Robins	€40	7:00PM
Ashdale	€45	8:30PM

Table 2. Options from the Restaurant Agent

Package	Price	Book From:
Virgil / Ashdale	€130	Scot Travel/trainfind.com
Virgil / Robins	€125	UK travel/trainfind.com
Now! / Ashdale	€115	Scot Travel
Now! / Robins	€110	Scot Travel
Fastjet / Ashdale	€135	Thomas Travel
Fastjet / Robins	€120*	Thomas Travel

Table 3. Possible package Solutions (*%25 Robins/Fastjet discount offer)

a list of possible travel/restaurant combinations (Table 3). From the desires expressed by the user, the service agent decides that the solutions that use Now! are not viable, as the flight would arrive in Aberdeen after the meeting start time. The trainfind/ Robins solution meets the requirements, but the agent decides to commit to the Fastjet/Robins package (Fig. 2), as the virtual organisation relationship between Fastjet and Robins can be exploited *within* this new virtual organisation to provide overall cheaper meal and travel, although the preference of travel by train needs to be relaxed.

We view this decision making process as a constraint problem, where the *avail-*

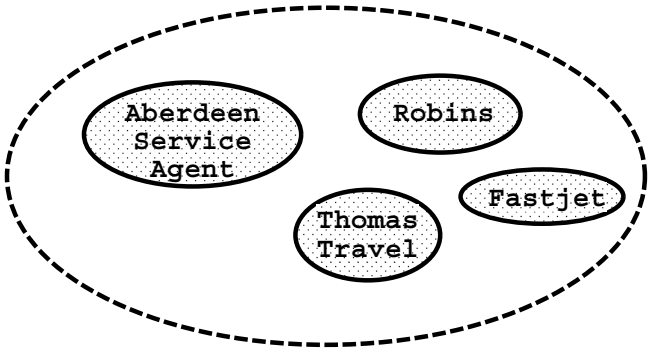

Fig. 2. *The VO formed by the Aberdeen Service Agent*

able, choices of travel and restaurant form the set of *possible solutions*, while the users *desires* and *preferences*, as well as information on services and their relationships in *virtual organisations* form the *constraints* over the possible solutions that influence the eventual decision. The *relaxation* of these constraints can also form the basis for negotation between the vendors in the VO, as in the example relaxtion of the method of travel described here.

This architecture is also used as a basis for the vendor agents when negotiating and forming virtual organisations. In particular, we describe how the autonomy provided by combining CLP with a BDI architecture provides an open and adaptable method for operating in competitive environments.

3 Constraints as Knowledge in an E-Commerce Scenario

Recent work has seen constraints being used for knowledge representation in a distributed agent-based environment [5, 12]. Our research takes this idea and focuses on using constraints to represent different sources of knowledge, but also to exploit this representation and to use these constraints along with CLP (Constraint Logic Programming) to provide an agent with intelligent decision-making capabilities that can be used in dynamic environments and situations where the choice of solution provided depends on commitments which are transient and can be changed or renegotiated at any time. In doing this, we endeavour to preserve agent autonomy and local decision making.
In the example in Section 2 we are using constraints to represent user desires specifying properties of the solution required, and also to represent specific instances of relationships between companies (e.g. the discount package between Fastjet airline and Robins restaurant). Both these aspects must be combined and used when deciding on a solution.

3.1 Constraints and Agents

Constraints are used to model a description of part of a *desired* goal state which the agent is free to achieve in any number of different ways. In doing so, the agent can take account of other constraints it has undertaken to satisfy by combining all the information into a CLP (Constraint Logic Programming) problem. Therefore an agent in a virtual organisation can take account of commitments with other agents when trying to provide a service or negotiate to join another virtual organisation.
This representation means that the agent is not just responding to a sequence of messages, but is able to deliberate on and *plan* its behaviour by taking into account both the message and other constraints. Further, if the desired state is impossible to achieve (over-constrained) because of too many different desires, it has to relax some constraints, by delegating or exchanging tasks with other agents (See Section 5), which is an important aspect of agent behaviour in multi-agent systems. The constraints thus become mobile [10].
The constraints can refer to the configuration in space of a number of objects, constrained by various relationships and inequalities described in predicate logic. This has been successfully used in the KRAFT project [18], but there the agents were 'information seeking', extracting the constraint information from databases and returning them to a central planner. Here the agents are themselves capable of planning and interacting with other agents and can use temporal as well as spatial constraints[4].
We can transform constraints to use a local ontology or data model as a common basis [13]. Once in this form we can treat the constraints as a CLP and thus provide a decision making process which can take into account many disparate factors, which is vital when providing autonomy in a highly dynamic environment.

```
Do
     options := option_generator(event_queue,B,D,I)
     selected-options := deliberate(options,B,D,I)
     update-intentions(selected-options,I)
     execute(I)
     get-new-external-events();
     drop-succesful-attitudes(B,D,I)
until quit
```

Fig. 3. *Main BDI loop*

4 BDI Agents

To provide an architecture to contain this CLP mechanism we have constructed a BDI agent, as its logical reasoning basis allows for easy integration with the logic approach of the CLP mechanism.
Fig. 3 shows a pseudo-code version of a BDI agent algorithm [19]. We use this as a basic mechanism for controlling the actions and decision making for each individual agent. This algorithm controls the agents planning and its interaction with others.
In a BDI loop, the agent has internal events from the desires (such as required services) and external events (such as information on the nature and type of services available) on the `event_queue` and uses these, along with its current beliefs (`B`), desires (`D`) and intentions (`I`) to generate a set of possible next options, represented as executable plans. It then selects one of these possible options based on the status of its B, D and I values. This selection is represented by our constraint mechanism, which takes the Beliefs and maps them to finite domains in a constraint problem. It then uses the desires and intentions (represented as constraints) in the constraint solving mechanism to choose and then execute the best action from those available. Finally, any new external events (caused by the execution of the chosen action or otherwise) are added to the `event_queue` and the agent checks to see whether it can mark as completed any of its goals that have been achieved, before carrying on.
The algorithm is adopted from the standard BDI loop [19]:

- *Beliefs* represent the views held about the current state of the world, as perceived by the agent.
- *Desires* are the goals and aims you wish your agent to achieve.
- *Intentions* are the actual processes being executed so that desires may be fulfilled (or sub-goals which are needed for the completion of a desire).

The process of choosing and executing a succession of actions gradually alters the agent's beliefs and provides newer and updated solutions. The agent may receive new desires in the form of constraints as it progresses but must take into account all existing commitment constraints until a solution has been found, when they can then be dropped and marked as being succesful (the `drop-succesful-attitudes` function in Fig.3).

4.1 Representing Beliefs, Desires and Intentions using CLP - A functional Data Model Approach

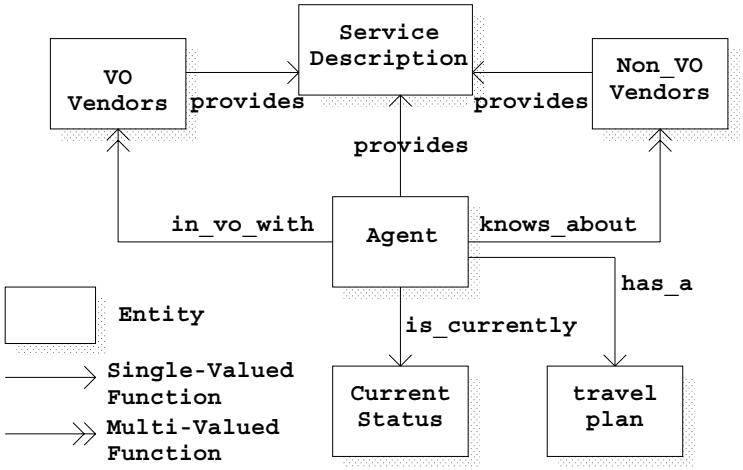

Fig. 4. *Agent's Belief data model*

We use an ER-like data model representation for the agent's beliefs (Fig. 4). This holds information on the agents current capabilities, as well as information on the capabilities of others and information on any relationships or coalitions formed. The entities hold the actual data, while the functions represent relationships between the entities. Our data model representation of the agent's beliefs is constructed using P/FDM[2], an object database constructed on semantic data modelling principles from the functional Data Model [21]. This database is constantly updated to reflect changes in the agent's status and current commitments as and when they happen. The advantage of this approach is that the data model can be used to express the semantics of the agent's beliefs as well as the beliefs themselves, and can express complex relationships between objects and data values in those beliefs.
The P/FDM constraint language, Colan [3] (based on Shipman's Daplex language), can be used to express desires in terms of beliefs. Initially Colan was used for specifying database integrity constraints, but since it is based on a range-restricted first order logic, it can be used to describe specifications or mobile fragments of specifications. The following shows an example Colan constraint, based on the scenario described in Section 2. It specifies that if an agent creates a travel plan for a journey, and the journey involves a sleeper train, then the arrival time of the sleeper train must be before 8.30am (For the purposes of simplifying the exposition we have left out contextual information such as `date`).

[2] http://www.csd.abdn.ac.uk/~pfdm

```
Colan Constraint
Constrain all t in travel_plan such that
travel_method(t) = train and....
travel_class(t) = sleeper_train
to have arrival_time(t) < 0830

First Order Logic Version
(∀t,a) travel_plan(t) &
travel_method(train, t) & ....
travel_class(sleeper_train,t) & arrival_time(a,t)→ a < 0830
```

This is an example of a constraint which, if given to a vendor, must be satisfied, along with any commitments or restrictions the vendor might have. The vendor agent would use the beliefs it has about its own status, and that of other agents capabilities, together with any existing constraints (e.g. from current VO relationships) to try to satisfy the given user constraint and provide a solution. There has been extensive work carried out on integrating P/FDM data model frameworks into finite domain CLP problems in the KRAFT project [13] and in using First Order Logic to express and generate CLP code in terms of this data model. Our techniques for transforming the information from the desires and beliefs into a finite domain constraint problem are largely based on this work, but in KRAFT constraints are used as problem specifications to find solutions to complex distributed design problems.
All components and agents are FIPA compliant, and we have RDF definitions for the COLAN constraint language, as well as for the Daplex database query language [11]. This is possible because of the entity-relationship model shared by both RDF and P/FDM.

4.2 Constructing possible worlds

The beliefs are a reflection of how the agent views the current state of the world so they naturally form the basis for the agent's deliberation process. Since we are using a data model approach to hold the agent's beliefs, and to provide the semantics to describe the domain, we formulate agent desires in terms of the entities, attributes and relationships contained in the data model and use these desires as *specifications* of the required tasks.
The advantage of using this data model approach is that each entity class in the data model can be easily viewed as a finite domain, with the object instances themselves as the elements in that domain. The object attributes and functions can then be used to form the basis for variables in the constraints.
The agent deliberation process must take into account current commitments and its current actions (intentions) when looking at any new collaborations. Given that the current environment is constantly changing, and that agents can negotiate and renegotiate commitments with each other at any time, we are unable to plan out and predict the agent's actions. what we *can* do is plan to a specified

depth of lookahead and then refine and replan as we encounter changes.
Although the current situation may change, the agent still maintains long term, high level desires which will remain unaffected (e.g. a vendor must be found that can provide travel requirements, but the specific choice of vendor is not stated). What *will* be affected by the changes is the way in which these high level desires are carried out. Thus we are using the CLP process to allow the agent to exercise a degree of *autonomy* in carrying out the high level desires. We provide it with the various implementations of each high level solution and give the agent an intelligent method for choosing the most appropriate implementation.
When the agent begins its deliberation process it first constructs a possible worlds model [19] which contains all possible solutions and the interim states needed to provide those solutions in a connected possible worlds tree like structure. To populate the tree with candidate solutions, we find all the actions possible from the current state, then apply these actions and add these as new branches to the tree. Therefore if the agent is currently at time t, that node of the tree will lead to x_1 possible states the agents beliefs could have at time *t+1* if a specific action towards the completion of a plan was carried out.
We then take these x_1 states and find the actions available from each of them. We can then apply these, and add the resulting new states to the tree (x_2 solutions at time *t+2*). This process can be continued to a given depth (i.e. how far we want the agent to plan ahead). This also means that we can limit the tree size to avoid a combinatorial explosion with the number of nodes.
The resulting populated structure contains all the possible states of the agents beliefs for the next n specified steps (what states the agent's belief data model would take were it to choose a specific way of carrying out all possible ways of doing these n steps towards completing the high level solution). We therefore have a possible worlds tree-like structure which preserves the information about the hierarchical position of each belief state in relation to the others, with each state containing the agent's beliefs, as they would be if the actions leading to the desired goal state were carried out.
Essentially this can be viewed as a *belief-accessible* world [19], where we can derive the *desire-accessible* and *intention-accessible* worlds, given the assumption that the agent will not desire or intend what it doesn't believe to be true. This can be achieved by constraining the *belief-accessible world* by applying the desires or intentions (represented as constraints).

4.3 From Possible Worlds to CLP

To construct the deliberation process as a CLP, we transform this structure into finite domains in ECLiPSe [3]. The constraints are then posted against these finite domains and any invalid states are eliminated. Fig. 5 shows an example of the constraints that can be posted in the CLP. The constraints come from various sources, but can all be combined to influence the agent's reasoning process.
Each node in the possible worlds structure holds a populated data model (Fig.

[3] http://www.icparc.ic.ac.uk/eclipse/

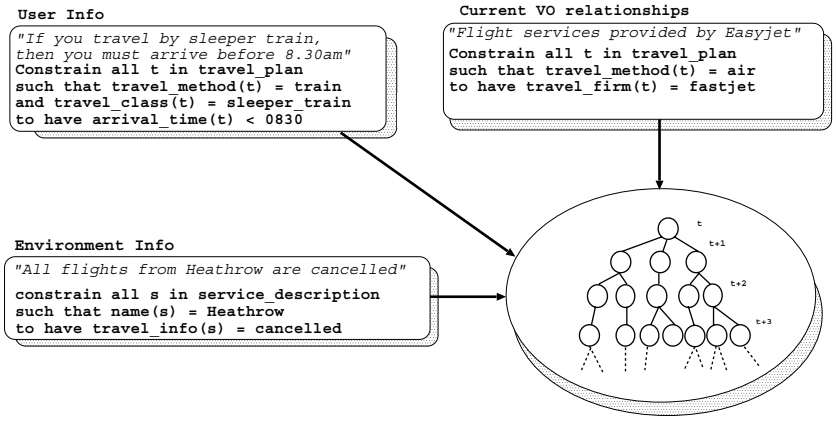

Fig. 5. *The Agent's Deliberation Process as a CLP*

4), representing the agent's beliefs for that particular state. The code fragment below shows part of the P/FDM data representation for a small part of one of these states. The entities and their values in the data model are represented using the `object/3` construct, and the functions and their relationships to other objects are shown by the `fnval/3` construct. The representation of the objects and the object functions as term structures gives us a uniform representation of the agent's beliefs, and allows for easy modification given any additions or changes to the data model representation.

```
object(travel_plan,agent1).
fnval(travel_method,agent1,train).
fnval(travel_class,agent1,sleeper_train).
fnval(arrival_time,agent1,0758).
fnval(vendor_used,agent1,vendor12).

object(vendor,vendor12).
fnval(vendor_name,vendor12,fastjet).

object(vendor,vendor10).
fnval(vendor_name,vendor10,Scot Travel).
```

4.4 Solving the CLP using the ECLiPSe Propia Library

The ECLiPSe *Propia* library supports *generalised constraint propagation*. Using the `infers most` operator we can specify the finite domains for the CLP problem using the following ECLiPSe goals:

```
object(travel_plan,AGENT_NAME) infers most.
fnval(travel_method,AGENT_NAME,METHOD) infers most.
```

```
fnval(travel_class,AGENT_NAME,CLASS) infers most.
fnval(arrival_time,AGENT_NAME,TIME) infers most.
fnval(vendor_used,AGENT_NAME,VENDOR_ID) infers most.

object(vendor,VENDOR_ID) infers most.
fnval(vendor_name,VENDOR_ID,VENDOR_NAME) infers most.
```

This makes the possible values of each attribute of each object (as well as the variables representing the objects themselves) into a finite domain CLP variable. For example, the above code fragment would create finite domains for each object and for each `fnval` construct (so `VENDOR_NAME` from the above `fnval` would represent the finite domain containing all vendor names from every state in every node in the tree that contains similar constructs). We can now post constraints from our various sources over these domains. Any elements which are eliminated from the solution space by the constraints will propagate through the rest of the domains, eliminating all belief states which do not satisfy the required constraints. The remaining belief states are the valid options available to the agent in the possible worlds model. What is left therefore is a pruned version of the possible worlds tree structure containing the remaining valid states given the constraints imposed.
From the remaining tree structure a valid chain of actions can be found that, when executed, will lead the agent to the desired state, and therefore the solution to the given task. This can then be returned by the deliberation process as the chosen plan to execute in the main BDI loop. If the problem is over-constrained there is the possibility that no solution is available for this plan, in which case the agent will try to relax the constraints.

5 Constraint Relaxation

While the constraint mechanisms described provide a way of providing autonomy in the decision making process, the agent needs to be flexible so that it can remove or relax possible constraints when the problem is over-constrained and a solution cannot be found.
An agent can find situations where the constraints imposed mean that no valid solution can be found to provide any alternatives for the decision making process. What the agent must do then is *relax* or re-negotiate the constraints it has. In the example shown in section 4.1, the agent has a constraint that if `travel_class` is `sleeper_train` for a journey, then `arrival_time` must be less than 0830.
Suppose this constraint was applied, and no solution can be found. The agent tries *relaxing* the constraint, and tries again using separately the following versions:

```
Constrain all t in travel_plan such that
travel_method(t) = train
```

```
and travel_method(t) = sleeper_train
to have arrival_time(t) < 0930

Constrain all t in travel_plan such that
travel_method(t) = train
to have arrival_time(t) < 0830
```

The first alternative relaxes a *constant*, so that the person only needs to arrive before 9:30am; the second alternative removes a *restrictive term*, so that the person could take a train which was not a sleeper.

6 Current and Related Work

In this paper, we have shown how an agent can choose the best among several possible virtual organisations to form in order to meet some customer requirement. Our approach is to use a deliberative process to construct possible worlds, each corresponding to a potential virtual organisation, and each configured using constraint satisfaction techniques. We are also investigating the scalability of such an approach, given the complexity of constraint satisfaction problems, in a realistic virtual organisation scenario.

Research and development work on technologies to support virtual organisations is not new [16], although it is currently receiving greater attention in the context of Grid computing [9]. In terms of explicit modelling of organisational properties (partner cabailities, service agreements, life cycle stages) the work that has been done using intelligent software agents [8] is well ahead of the mainstream distributed computing work that currently underpins Grid computing. Recent research in the agents area relevant to virtual organisations includes work on coalition formation [2] and formal specification of the rules constituting "electronic institutions" [6]. All of this work is complementary to ours.

As stated earlier, our current work builds on the results of the KRAFT project [18] which in turn was heavily influenced by early multi-agent projects such as SHADE [15], and ADEPT [14]. The novelty in our approach lies in the use of constraints to give a formal yet flexible business knowledge, enabling both the formation and operation of virtual organisations [17].

The Smart Clients project [22] is related to KRAFT in the way they conduct problem-solving on a CSP dynamically specified by the customer, using data extracted from remote databases. Their approach differs from ours in that only data is extracted from the remote databases, no constraints are therefore transmitted across the network; conversely, it is the constraint solver that is transmitted to the client's computer, to work with the constraints specified locally by the customer.

Finally, the Business Rule Markup Language is similar in concept to KRAFT's use of constraints [20]. The difference is that this work uses a rule-based formalism to specify business rules. Logic programming techniques are then used to reason with the rules.

Our current research draws in part on work being done in the context of the Advanced Knowledge Technologies project (AKT) — a multi-site collaboration focussing on the knowledge management life-cycle[4]. AKT shares some basic CLP mechanisms with our current work, as described in 4.1, and is developing RDF definitions for the P/FDM schema language and the Colan constraint language [11].
Our further work in this area will be done as part of the CONOISE project[5], a joint collaboration between Aberdeen, Cardiff and Southampton universities, and BTexact Technologies. CONOISE will look at agent interaction and deployment in open systems, and in particular the interplay between constraint-solving and negotiation processes in the formation and operation of virtual organisations. Initially, we are exploring how Southampton's work in online auctions can be effectively combined with our CSP approach in virtual organisation formation, and how Cardiff's work on service descriptions can enrich the information available to agents during the formation process.

Acknowledgements

The basis of this work is from research supported by an EPSRC grant under the supervision of Prof. P.M.D. Gray and on work previously completed during the KRAFT project, in particular the work of Kit-Ying Hui and Alun Preece. The KRAFT project was funded by grants from BT and EPSRC. The CONOISE project is funded by BTexact Technologies.

References

1. H. Akkermans. Intelligent e-business: From technology to value. In *IEEE Intelligent Systems July/August*, volume 16, pages 8–10, 2001.
2. P. Anthony, W. Hall, V. Dung Dang, and N. R. Jennings. Autonomous agents for participating in multiple online auctions. In *IJCAI01 Workshop on E-Business & the Intelligent Web*, pages 54–64, 2001.
3. N. Bassiliades and P.M.D. Gray. Colan: a functional constraint language and its implementation. In *Data and Knowledge Engineering 14*, pages 203–249, 1994.
4. S. Chalmers and P.M.D. Gray. Bdi agents and constraint logic. In *AISB Journal, Special Issue on Agent Technology*, volume 1, pages 21–40, 2001.
5. P. Eaton, E. Freuder, and R. Wallace. Constraints and agents: Confronting ignorance. In *AI Magazine 19(2):50-65*, 1998.
6. M. Esteva, J.A. Rodriguez, C. Sierra, P. Garcia, and J.L. Arcos. On the formal specification of electronic institutions. pages 126–147. Springer-Verlag, 2001.
7. N.J. Fiddian, P. Marti, J-C. Pazzaglia, K. Hui, A. Preece, D.M. Jones, and Z. Cui. Application of KRAFT in data service network design. In *BT Technology Journal*, volume 17. Chapman and Hall, 1999.

[4] http://www.aktors.org
[5] http://www.conoise.org

8. K. Fischer, J. Muller, I. Heirnig, and A-W. Scheer. Intelligent agents in virtual enterprises. In *Proc 1st International Conference on Practical Applications of Intelligent Agents*, London, 1996.
9. I. Foster, C. Kesselman, J. M. Nick, and S. Teucke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *http://www.globus.org/research/papers/ogsa.pdf*, 2002.
10. P.M.D. Gray, S. M. Embury, and G. J. L. Kemp. The evolving role of constraints in the functional data model. In *Journal of Intelligent Information Systems*, volume 12, pages 113–117. Kluwer Academic Press, 1999.
11. P.M.D. Gray, K. Hui, and A. Preece. An expressive constraint language for semantic web applications. In *IJCAI01 Workshop on E-Business & the Intelligent Web*, pages 46–53, 2001.
12. P.M.D. Gray, K. Hui, and A. D. Preece. Finding and moving constraints in cyberspace. In *Proc. AAAI Spring Symposium on Intelligent Agents in Cyberspace (SS-99-03)*, pages 121–127, 1999.
13. K. Hui and P.M.D. Gray. Developing finite domain constraints – a data model approach. In *Proceedings of the 1st International Conference on Computational Logic*, pages 448–462. Springer-Verlag, 2000.
14. N. Jennings, P. Faratin, M. Johnson, T. Norman, P. O'Brien, and M. Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5:105–130, 1996.
15. D. R. Kuokka, J. G. McGuire, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, and G. R. Olsen. SHADE: Technology for knowledge-based collaborative engineering. *Journal of Concurrent Engineering: Applications and Research*, 1(2), 1993.
16. D. E. O'Leary, D. Kuokka, and R. Plant. Artificial intelligence and virtual organisations. *Communications of the ACM*, 40:52–59, 1997.
17. A. Preece, P.M.D. Gray, and K. Hui. Supporting virtual organisations through knowledge fusion. In *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop*, Menlo Park, CA, 1999. AAAI Press.
18. A. Preece, K. Hui, A. Gray, P. Marti, T. Bench-Capon, Z. Cui, and D. Jones. KRAFT: An agent architecture for knowledge fusion. In *International Journal on Intelligent Cooperative Information Systems (IJCIS)*, 2000.
19. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems*, 1995.
20. D. Reeves, B. Grosof, M. Wellman, and H. Chan. Toward a declarative language for negotiating executable contracts. In *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop*, Menlo Park, CA, 1999. AAAI Press.
21. D. W. Shipman. The functional data model and the data language DAPLEX. In *ACM Transactions on Database Systems 6(1)*, pages 140–173, 1981.
22. M. Torrens and B. Faltings. Smart clients: constraint satisfaction as a paradigm for scaleable intelligent information systems. In *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop*, Menlo Park, CA, 1999. AAAI Press.