# Supporting Virtual Organisations through Knowledge Fusion

Alun Preece

*University of Aberdeen, Computing Science Department*
*Aberdeen AB9 2UE, Scotland*
*Phone: +44 1224 272296; FAX: +44 1224 273422*
*Email: apreece@csd.abdn.ac.uk*

**Abstract**

The formation and operation of dynamic and open virtual organisations is a central concern in business-to-business e-commerce. Virtual organisations enable partner companies to develop and manufacture customised products with low costs and rapid delivery. Agent-based architectures are an effective platform for such virtual organisations because they provide mechanisms to allow organisations to advertise their capabilities, exchange rich information, and synchronise workflows at a high-level of abstraction. In this paper, we examine the KRAFT architecture and its features for supporting virtual organisations. In particular, we focus upon KRAFT's agent-based architecture, and its use of constraints as a knowledge exchange medium. We show how constraint fusion supports the design of customised products.

## 1 Introduction

One of the most promising areas of electronic commerce (e-commerce) is improved management of the supply chain, to streamline the production of goods, enable the rapid production of customised goods, and coordinate business processes among cooperating organisations [15]. In effect, suppliers, manufacturers and retailers are enabled to act as a single *virtual organisation*; the member companies integrate their complementary resources to create a more competitive whole. Providing technology to support virtual organisations is the primary theme of business-to-business e-commerce [25]. Successful integration requires that the members of the virtual organisation at least agree on mechanisms to exchange electronic documents and synchronise their workflows [19].

To minimise production times and product costs, the virtual organisation should be agile: relationships between members need to be dynamic and flexible [21]. Re-negotiations between suppliers, manufacturers and retailers will occur regularly. In this kind of agile organisation, there is competition between members, and members join and leave the organisation more regularly. The support of agile organisations is technologically challenging because the communication mechanisms must cope with both the cooperative and the competitive nature of the enterprise.

The commonly-accepted life-cycle for virtual organisations is as follows [8]:

1. *needs identification:* definition of the services or products provided by the virtual organisation;

2. *partner selection:* composition of the group of partners that, together, can meet the identified needs;

3. *operation:* conduct of the transactions by which the services or products are provided by the partners;

4. *dissolution:* disbanding of the group of partners, including any final settlement of payment or other closing transactions.

In current practice, the main technologies offered to support virtual and agile organisations are Electronic Data Interchange (EDI) and Extranets. EDI supports the exchange of structured documents along the supply chain (from requests for quotations to invoices). Unfortunately, current EDI systems are largely proprietary and limited in the form of information they can exchange; they are limited to the exchange of relatively simple relational data. The newer XML standard promises to address the former problem (for representative examples, see [5] and [24]), but it will not solve the latter: to be fully "self-describing", business data needs to have attached meta-knowledge in the form of rules or constraints on how the information can be used and combined with other information [13].

Extranets provide the low-level communication protocols to exchange EDI messages securely between the internal networks (Intranets) of the individual companies. Current Extranet technology is more concerned with basic message exchange and security than with supporting higher-level business operations. Recent standards for supporting open transactions such as the CORBA Services and Enterprise JavaBeans [20] provide a useful higher-level communication infrastructure, but rely on conventional rigid electronic data transactions models that cannot cope with the required flexibility demanded by agile organisations [26].

The KRAFT project (Knowledge Reuse and Fusion/Transformation) has an architecture that is well-suited to support virtual organisations in which members exchange information in the form of constraints expressed against an object data model [22]. The constraints allow member companies to design new products from components in their individual catalogues, and also to advertise the content of their catalogues. Constraints are exchanged via messages expressed in an agent communication language, supporting flexible transactions. The KRAFT supports the life-cycle of a virtual organisation as follows:

1. *needs identification:* customers' requirements can be expressed readily and naturally as a set of constraints; likewise, the capabilities of service and product providers can be expressed using constraints;

2. *partner selection:* by combining and checking the constraints from customers and service/product providers, a virtual organisation can be composed that has the potential to meet the customers' requirements;

3. *operation:* additional constraints will appear during the process of working to satisfy a specific customer's requirements — these may come from the customer itself, or from any of the suppliers; the constraint-solving process can easily accommodate these constraints, dynamically;

4. *dissolution:* the constraint solving process will yield a set of results that include the conditions that must be met when the virtual organisation is disbanded.

The remainder of this paper is organised as follows: the next section describes the way in which the KRAFT architecture uses constraints to support a network or customers and suppliers in a virtual organisation; Section 3 describes the KRAFT architecture as a whole; Section 4 examines the constraint-solving process at the core of a KRAFT system; Section 5 presents an example KRAFT application, where a virtual organisation is created to provide network data services in the telecommunications domain; Sections 6 and 7 present related work and future directions, respectively.

## 2 Constraints in Virtual Organisations

The KRAFT architecture was conceived to support configuration design applications among multiple partner organisations with heterogeneous knowledge and data models. This makes it suitable for the support of virtual organisations where the service or product to be supplied needs to be composed from components supplied by multiple component vendors. This model turns out to be very general, supporting not only the obvious manufacturing-type applications (for example, configuration of personal computers or telecommunications network equipment) but also service-type applications such as travel planning (for example, composing package holidays or business trips involving flights, ground travel connections, and hotels) and knowledge management (for example, selecting and combining business rules from multiple heterogeneous knowledge and databases on a corporate intranet).

Configuration design problems were originally tackled by rule-based systems (the best-known being DEC's XCON system, used for configuring VAX computers); now, they are more commonly seen as constraint satisfaction problems [9]. In the KRAFT architecture, the domains of many of the variables are entities stored in local databases held by individual partners in a virtual organisation [12]. Constraints on these entity types may be set by their makers, and stored with them in the database. KRAFT provides mechanisms by which local database contents can be advertised on the network, so that constraints can be selected and *fused* together by specialised middle agents, and passed to a constraint solver. The solver then has to find feasible values to satisfy the constraints. However, the problem is complicated by constraints that refer to related instances of other entity types, whose values must be extracted from some database and checked for compatibility.

In manufacturing-type applications, KRAFT solves the problem of finding a number of parts that fit together to make some product, or that work together in some way to deliver some service. Suppliers of these parts make their catalogues, in the form of database tables, available on the network. Locally to each supplier, the tables will be likely to have different semantics and hidden assumptions. In printed catalogues, these assumptions often appear as asterisked footnotes or "small print" in the catalogue. For example, the product catalogue for (fictitious) disk drive vendor Storage Inc may have the following small print associated with each of its range of Zip disk drives: *this Zip disk drive requires a PC with a USB-type port.* This kind of small print can readily be expressed as constraints in a database catalogue. For example:

```
forall z in zip_drives :
    port_types(connected_pc(z)) must include "USB";
```

This constraint would be stored in Storage Inc's catalogue database, but note that it refers to a property of the PC (`port_types`) to which the drive would be connected in a configured PC system (stored as the `connected_pc` attribute of the Zip disk drive `z`). In configuring a design to meet a customer's requirements, it is not enough just to make a distributed database query to find a list of possible components; it is also necessary to ensure that the components satisfy attached small print constraints.

A further complication arises from the fact that the locally-stored constraints will typically be expressed against a local data model for the product catalogue. In order to fuse together constraints from multiple heterogeneous product catalogues, it is necessary to translate the constraints to a common *constraint interchange format*, expressed in the terms of a *shared ontology* as is commonly done in knowledge-sharing systems [18]. The following constraint shows the above small print Zip disk constraint translated from Storage Inc's local constraint language to the KRAFT Constraint Interchange Format (CIF) language, expressed in the terms of a shared ontology for PC system configuration:

```
constrain
    each d in disk_drive
        such that name(vendor(d)) = "Storage Inc"
```

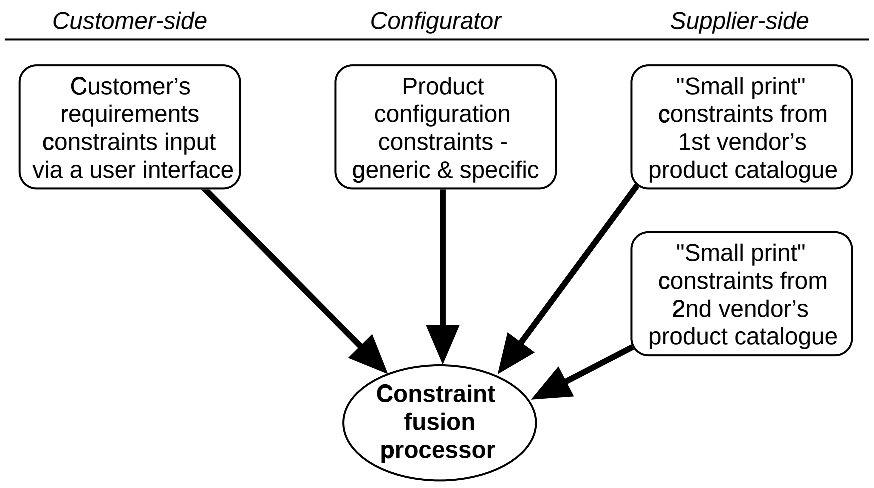

Customer-side
Configurator
Supplier-side
Customer's requirements constraints input via a user interface
Product configuration constraints - generic & specific
"Small print" constraints from 1st vendor's product catalogue
"Small print" constraints from 2nd vendor's product catalogue
Constraint fusion processor

Figure 1: Fusion of constraints from multiple sources.

```
          and type(d) = "Zip"
      at least 1 p in ports(host_pc(d))
  to have type(p) = "USB";
```

KRAFT CIF is based on the CoLan language used to express semantics in the object database P/FDM [11]. Some of the transformations needed here were:

- *Addition of contextual information.* In the local Storage Inc database, all constraints implicitly refer to this vendor's products; in the shared ontology, the name of the vendor of the component must be stated explicitly.
- *Mapping classes to attribute values.* In the local Storage Inc database, the type *Zip disk* is represented by the class `zip_drives`; in the shared ontology, "Zip" is the value of the attribute type of elements in the class `disk_drive`.
- *Coping with different granularities of description.* In the local Storage Inc database, the connected PC is modelled in less detail, with the types of ports being stored in the attribute `port_types` of the `connected_pc` entity; in the shared ontology, the connected PC of the disk drive (attribute renamed to `host_pc`) is modelled in more detail, with individual ports as entities in their own right, and `type` as an attribute of the `port` entity.

These transformations must be implemented by special-purpose software for the individual vendor, as part of the price to be paid for joining the virtual organisation. The architectural framework that incorporates these transformational software is described in Section 3 below. Once transformed, the small print constraints can be fused together with other constraints from various sources, as shown in Figure 1.

In the virtual organisation, some constraints will be provided by the customer; others will come from the vendors as discussed above; there will also be constraints coming from the partner who will act as the configurator of the product or service provided by the organisation. Typically, the configurator partner will be a value-adding reseller from the point-of-view of the component vendors. Note that there may be multiple configurator partners, each providing a different product or service; also, the supply chain may have additional stages, where one reseller sells to another reseller, each adding their own constraints to the final product or service. Details of how the constraint fusion process operates within the KRAFT architecture are given in Section 4.

## 3 The KRAFT System Architecture

The KRAFT system has an agent-based architecture, in which all knowledge processing components are realised as software agents. An agent-based architecture was chosen for KRAFT for the following reasons:

- Agent architectures are designed to allow software processes to communicate knowledge across networks, in high-level communication protocols; as constraints are a sub-type of knowledge, this was seen as an important feature for KRAFT.
- Agent architectures are highly dynamic and open, allowing agents to locate other agents at run-time, discover the capabilities of other agents, and form cooperative alliances; as KRAFT is concerned with the fusion of knowledge from available on-line sources, these features were seen as being of great value.

The design of KRAFT is consistent with several emerging agent standards, notably the de facto KQML standard [17] and the de jure FIPA standard [4]. Agents are peers; any agent can communicate with any other agent with which it is acquainted. Agents become acquainted by registering their identity, network location, and an advertisement of their knowledge-processing capabilities with a specific type of agent called a *facilitator* (essentially an intelligent yellow pages service).

When an agent needs to request a service from another agent, it asks a facilitator to recommend an agent that appears to provide that service. The facilitator attempts to match the requested service to the advertised knowledge-processing capabilities of agents with which it is acquainted. If a match is found, the facilitator can inform the service-requesting agent of the identity, network location, and advertised knowledge-processing capabilities of the service provider. The service-requesting agent and service-providing agent can now communicate directly.

It is worth emphasising that, while this model is superficially similar to that used in distributed object architectures such as CORBA and DCOM [20], the important difference is the semantic level at which interactions take place: In distributed object architectures, objects advertise their presence by registering method signatures with registry services, and communicate by remote method invocations.

In agent-based systems, advertisements of capabilities are much richer, being expressed in a declarative knowledge representation language, and communication uses a high-level conversational protocol build from primitive conversational actions such as *ask*, *tell*, *advertise*, and *recommend*. Distributed object architectures are in fact highly suitable for implementing agent-based architectures (for example, the ADEPT system used CORBA [14]) but the converse is not true.

### 3.1 Types of KRAFT Agent

An overview of the generic KRAFT architecture is shown in Figure 2. KRAFT agents are shown as ovals. There are three kinds of these: wrappers, mediators, and facilitators. All of these are in some way knowledge-processing entities. We describe each one below.

**Wrappers** These are agents that act as proxies for external knowledge sources, typically databases and knowledge-based systems. These are often legacy systems, so one task of a wrapper is to provide a bridge between the legacy system interface and the KRAFT agent interface. For example, the legacy interface of a relational database will typically be SQL/ODBC; the KRAFT wrapper will accept incoming request messages from other agents in the KRAFT agent communication language, transform these into to SQL queries, run them on the database, and transform the returned results to an outgoing message in the KRAFT agent communication language.

In a virtual organisation, wrappers are used to provide access to vendors' product databases as explained in Section 2. The wrapper software implements the constraint transformations necessary

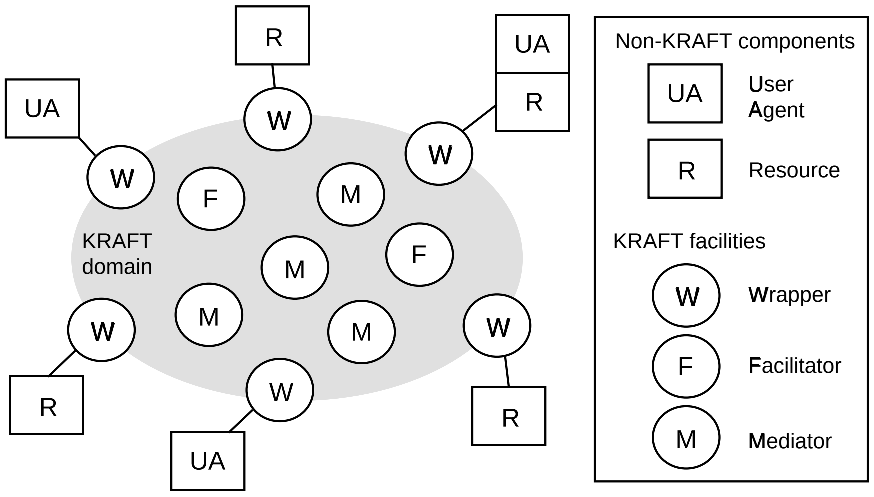


Figure 2: Overview of the generic KRAFT architecture.

to move small print constraints from the local product database to the CIF interchange format and shared ontology.

Wrappers also provide entry-points into the KRAFT system for user agents. User agents allow end-users access to a KRAFT knowledge processing system. A user agent will offer some kind of user interface, with which the user will present queries to the KRAFT network. The user agent will transform the users' queries into the internal knowledge representation language of the KRAFT system, and interact with other KRAFT agents to answer the queries. A user agent will typically also do some local processing on knowledge, at least to transform it for presentation.

In a virtual organisation, customers input their requirement constraints via user agents. Customers would not be expected to write the constraints themselves; rather, a friendly interface would allow them to enter the elements of the constraints, from which the user agent software would generate the corresponding CIF constraints.

**Mediators** These are the internal knowledge-processing agents of the KRAFT system: every mediator adds value in some way to knowledge obtained from other agents. Typical mediator tasks include filtering, sorting, and fusing knowledge obtained from other agents.

In a virtual organisation, the mediators are typically agents of the configurator partners or resellers. They perform constraint selection and fusion, and invoke the services of constraint solvers.

**Facilitators** These have already been mentioned above: these are the "matchmaker" agents that allow agents to become acquainted and thereby communicate. Facilitators are fully-fledged knowledge-processing entities: establishing that a service request "matches" a service advertisement requires reasoning with the declarative representations of request and advertisement.

In a virtual organisation, the facilitators accept advertisements from vendors and resellers, and broker connections between customers and resellers, and resellers and vendors.

## 3.2 KRAFT Communication Protocols

KRAFT agents communicate via messages using a nested protocol hierarchy. KRAFT messages are implemented as character strings transported by a suitable underlying protocol (for example, CORBA IIOP or TCP via sockets). A simple message protocol encapsulates each message with low-level header information including a timestamp and network information.

The body of the message consists of two nested protocols: the outer protocol is the agent communication language CCQL (*Constraint Command and Query Language*) which is a subset of the Knowledge Query and Manipulation Language (KQML) [17]. Nested within the CCQL message is its content, expressed in the CIF protocol (*Constraint Interchange Format*) — a superset of the CoLan constraint language shown in Section 2.

Syntactically, KRAFT messages are implemented as Prolog term structures. An example message is shown below. The outermost `kraft_msg` structure contains a `context` clause (low-level header information) and a `ccql` clause. The message is from an agent called `storage_inc` to an agent called `pc_configurator`. The `ccql` structure contains, within its content field, an encoded CIF expression (here, we see a "pretty-printed" CIF constraint; in the implementation, CIF expressions are actually transmitted in a compiled internal format).

```
kraft_msg(
    context(1,id(19), pc_configurator, storage_inc,
        time_stamp(date(29,9,1999), time(14,45,34))),
    ccql(tell, [
        sender : storage_inc,
        receiver : pc_configurator,
        reply_with : id(18),
        ontology : shared,
        language : cif,
        content : [
            constrain
                each d in disk_drive
                    such that name(vendor(d)) = "Storage Inc"
                    and type(d) = "Zip"
                at least 1 p in ports(host_pc(d))
            to have type(p) = "USB"
    ])
)
```

Use of Prolog term structures is chiefly for convenience, as most of the current knowledge-processing components in the KRAFT implementation are written in Prolog. However, the Prolog term structures are easily parsed by non-Prolog KRAFT components; currently there are several components implemented in Java, for example. (It is likely that the next version of the KRAFT implementation will use XML instead of Prolog term structures, as XML retains the ease of parsing, while being a more open interchange standard.)

As explained in Section 2, the terms used in the CIF part of the message are defined in the shared ontology. To support the representation, storage, and transmission of data instances, the ontology has schema-level information in addition to conceptual-level definitions.

## 4 Constraint Fusion Example

To demonstrate constraint fusion from different sources to support the manufacturing activities of a virtual organisation, consider a configuration problem where a PC is built by a reseller, combining components from vendors. As shown in Figure 1, constraints come from the user requirements, "small print" restrictions attached to components from different vendors, and the reseller's configuration design knowledge governing a workable configuration. The customers specify their requirements in the form of constraints through a user agent. In this example, a customer specifies that the PC must use a "Pentium 3" processor but not the OS named "Windows 2000":

```
constrain each p in pc
    to have cpu(p) = "Pentium 3"
    and name(has_os(p)) <> "Windows 2000"
```

For the components to fit together, they must satisfy certain constraints defining valid configurations. For example, the size of the OS must be smaller or equal to the hard disk space for a proper installation:

```
constrain each p in pc
    to have size(has_os(p)) <= size(has_disk(p))
```

Now the candidate components from different vendors may have “small print” conditions attached to them as constraints. In the vendor database of operating systems, the OS named “Linux” requires a memory of at least 32 megabytes:

```
constrain each p in pc
such that name(has_os(p)) = "Linux"
    to have memory(p) >= 32
```

In the virtual organisation, the reseller partner employs a constraint-fusing mediator which extracts and combines constraints from customer and vendors for problem-solving purposes. The constraints are assembled by the fusion mediator and pre-processed to compose a constraint satisfaction problem (CSP), which is then analysed and solved by a combination of distributed database queries and constraint logic programs. With the help of a facilitator, this approach allows the mediator to tailor its execution plan in a dynamic environment, depending on the capability and availability of online resources.

The sample constraints above can be pre-processed to give the following fused constraint, which describes the overall requirement on the variables involved:

```
constrain each p in pc
    to have cpu(p) = "Pentium 3"
    and name(has_os(p)) <> "Windows 2000"
    and size(has_os(p)) <= size(has_disk(p))
    and if name(has_os(p)) = "Linux"
        then memory(p)) >= 32
        else true
```

The reason for fusing the constraint fragments is to provide the basis for exploring how the CSP can best be divided into sub-problems of distributed database queries and sub-CSPs [11]. When a single piece of constraint is insufficient to solve a CSP effectively, the fusion mediator can combine information from multiple constraint fragments to arrive at a more solvable solution. It is from the fusion process that useful information can be inferred and captured for problem solving purposes.

The constraint fusion process composes a concrete description of the overall CSP in a declarative form. To solve a composed CSP efficiently, the fusion mediator feeds it into a problem decomposer which extracts selection information from the CSP description to generate distributed database queries, with the remaining constraints forming a smaller sub-CSP. The mediator then sends these database queries in multiple messages to different database wrappers to retrieve candidate data values.

Database query generation constitutes an important phase of pre-processing. It shifts part of the problem solving process to the distributed databases by composing data filters as database queries. This prevents unnecessary transportation of irrelevant data across the KRAFT domain, and thereby reduces network traffic in the distributed system. However, data filtering by database query generation is not sufficient to resolve all constraints. The amount of selection information which can be represented as database queries depends on the expressiveness of the database query language. The remaining sub-CSP has to be resolved by a more powerful constraint solver in the next stage. The final stage of the problem solving process is to feed data and constraints into a constraint solver so that solutions to the CSP can be obtained. In the current demonstration system, described in

Section 5, we use the finite domain constraint solver in the ECLiPSe constraint logic programming (CLP) system.[1]

## 5 An Example KRAFT Application

To demonstrate proof-of-concept, the KRAFT architecture has been tested on a realistic application in the domain of telecommunications network data services design; this application has been provided by BT. The network data services design problem considered by KRAFT is in the phase of network configuration from the viewpoint of a customer at a single site, allowing a BT network designer to select to meet the customers' requirements: (1) a suitable Point of Presence (POP) at which to connect to the BT network and (2) suitable Customer Premises Equipment (CPE) with which to service the connection (types of CPE include routers, bridges, and FRADs, though it was decided to focus initially solely on router products). This scenario tests the applicability of KRAFT to supporting virtual organisations: CPE vendors can join the network and advertise their products to the value-adding reseller of network data services. The reseller offers its services to the customer by means of user agents.

A conceptual view of the application architecture is shown in Figure 3(a). In the current implementation, all KRAFT agents (mediators, facilitators, and wrappers) are implemented in Prolog. The user interfaces (user agent and message monitor) are Java applications. The database resources are managed by independent instances of the P/FDM DBMS[2], each with its own local schema. The constraint solver is ECLiPSe. Inter-agent communication is implemented by asynchronous message passing using the Linda model [3].

The prototype application employs three sources of constraints:

- a database of POP information;
- two databases of CPE information: one for each of two competing router vendors.

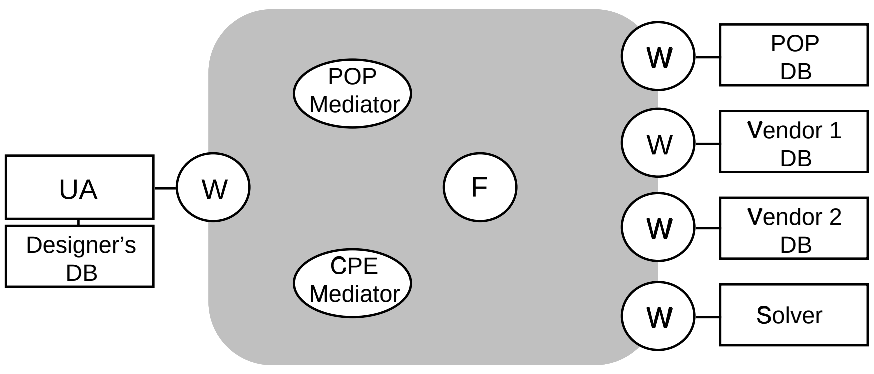
These information sources are considered to be pre-existing legacy databases. For the purposes of the prototype, simplified versions of these databases were created; however, care was taken to ensure that the databases of CPE information were created independently, so as to ensure realistic heterogeneity. Each of the databases was populated with data and constraints; for example, a vendor database was populated with data on the vendor's CPE products, and constraints defining the valid usage of each product. The main aim of creating the three resources was to test the feasibility of creating wrapper agents to transform between the internal knowledge representation (data and constraints) and the KRAFT CIF language.

In the prototype, the tasks of identifying potential POPs and CPEs are the responsibility of mediators. As the two tasks are independent in practice (it is possible to select a CPE on the basis of a customer's LAN and WAN requirements, without knowing which POP will be used, and vice versa), it was decided to provide a separate mediator for each task.

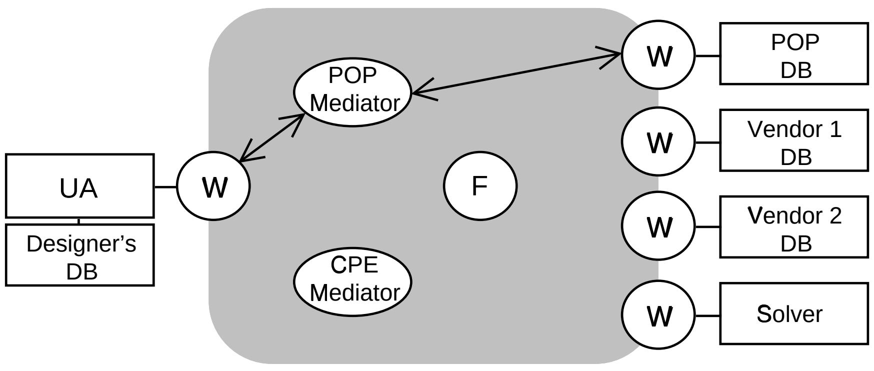
A user agent serves as the front-end to the test system, with a graphical user interface allowing a BT network designer to enter the customer's requirements, and launch two kinds of query into the KRAFT system:

1. For a POP query, the user specifies the location of the customers' site, and the customers' required wide-area network (WAN) services (for example, Frame Relay and ISDN). The user agent formulates the POP query as a KRAFT message, and attempts to locate an agent that can answer the query. It does this by contacting a facilitator, which in turn puts it in contact with the POP Mediator. Upon receipt of the user agent's query, the POP mediator obtains

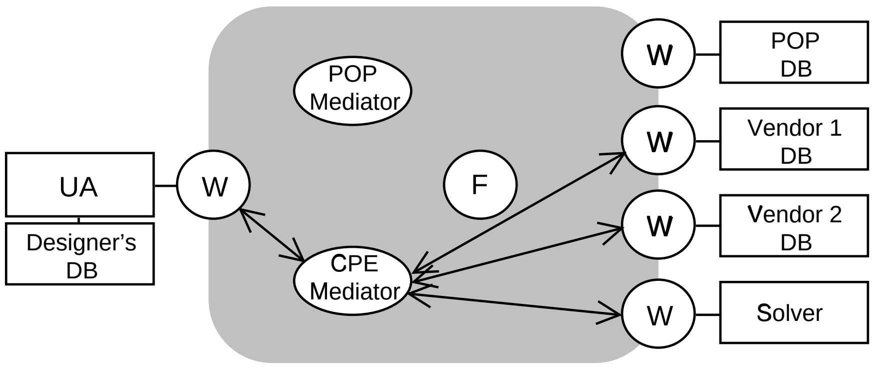
[1]`http://www.ecrc.de/eclipse/`
[2]`http://www.csd.abdn.ac.uk/~pfdm`



(a) KRAFT Network DataServices application architecture



(b) KRAFT Network DataServices application interaction 1: locate a POP



(c) KRAFT Network DataServices application interaction 2: choose CPE

NOTE In (b) and (c), interactions with the facilitator are not shown.

Figure 3: Network data services scenario architecture.

a list of POPs from the POP DB, and filters these according to the user's requirements. It then sends a reply to the user agent. If one or more suitable POPs were found these will be displayed to the user, ranked in order of proximity to the customers' site. These interactions are shown in Figure 3(b).

2. For a CPE query, the user specifies additional constraints on the type of equipment needed, including support for various LAN protocols used within the customer's site (TCP/IP, AppleTalk, 10 base T Ethernet, etc) and support for the required WAN services that determined the choice of POP (Frame Relay, ISDN, etc). Having acquired these constraints, the user agent issues a query to the KRAFT network as above. This time, the CPE Mediator interacts with vendors to select apparently-suitable products, together with any "small print" constraints on these. Fusing these constraints with those from the customer's requirements, the CPE Mediator then calls upon a constraint solver to identify actually-suitable CPE products. These it relays back to the user agent. These interactions are shown in Figure 3(c).

The implemented application was tested to demonstrate the essential functionality of all the components (facilitation, constraint transformation, and constraint fusion), and also to test the performance of a wide-area KRAFT network (with agents running at all four of the project sites across the UK). The tests showed that a virtual organisation of vendors, configurator, and customer could indeed form at run-time, and coordinate its activities to put together a workable, multi-vendor solution to the customer's problem, without any of the parties having any prior knowledge of one another. The facilitation/matchmaking mechanism allows any party to dynamically join or leave the KRAFT network.

While the functionality of a KRAFT network was proven satisfactorily, the performance proved to be rather sluggish, due largely to the choices of platform (Prolog and Java). Further details of the testbed application are available in[7].

## 6 Related Work

Agent-based architectures are proving to be an effective approach to developing distributed information systems [2], as they support rich knowledge representations, meta-level reasoning about the content of on-line resources, and open environments in which resources join or leave a network dynamically [28]. KRAFT employs such an agent-based architecture to provide the required extensibility and adaptability in a dynamic distributed environment. Unlike most agent-based distributed information systems, however, KRAFT focuses on the exchange of data and constraints among agents in the system.

Recent research in the area of software agent technology offers promising ways of supporting new kinds of distributed information system applications, but the area is still far from mature. Early projects such as PACT [6] and SHADE [16] showed that agent technology could support exchange of rich business information — using the Knowledge Interchange Format (KIF) — between organisations using heterogeneous technologies, with a limited amount of organisational agility — basic "matchmaking" brokerage connecting suppliers to consumers. While demonstrating the promise of the agent-based approach, these projects revealed problems: the complexity of the KIF representation has prevented it from gaining widespread use, while the limited brokerage models hinders the implementation of flexible negotiation schemes.

The ADEPT project offers a flexible environment for agile organisations, with an emphasis on the dynamic management of workflow between partner organisations [14]. Service agreements are negotiated, formed, and re-formed over time, supporting both competitive and collaborative interactions, albeit with rather limited forms of information exchange.

The design of the KRAFT architecture builds upon recent work in agent-based distributed information systems. In particular, the roles identified for KRAFT agents are similar to those in the

InfoSleuth system [2]; however, while InfoSleuth is primarily concerned with the retrieval of data objects, the focus of KRAFT is on the combination of data and constraints. KRAFT also builds upon the work of the Knowledge Sharing Effort [18], in that some of the facilitation and brokerage methods are employed, along with a subset of the 1997 KQML specification [17]. Unlike the KSE work, however, which attempted to support agents communicating in a diverse range of knowledge representation languages (with attendant translational problems), KRAFT takes the view that constraints are a good compromise between expressivity and tractability.

In its emphasis on constraints, KRAFT is similar to the Xerox Constraint Based Knowledge Brokers project [1]; the difference is that the Xerox work focusses upon the use of constraints to support querying of distributed data sources, rather than the extraction of constraints from distributed sources, and the use of these constraints in configuration design problem-solving. Also, KRAFT recognises the need to transform constraints when they are extracted from local resources, typically for reasons of ontological or schema mismatch [12].

Constraints are also used to advertise customers' requirements and supplier capabilities in the Matchmaker project, which is currently being applied to supporting virtual organisations [10]. Again, like the Xerox work, the Matchmaker project does not deal with the extraction of constraints from distributed sources, and their use in problem-solving.

The Smart Clients project [27] 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 KRAFT in that only data is extracted from the remote databases, no small print constraints come attached to the data; also, all the problem-solving is done on the client, rather than by mediator agents. 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, ongoing work at IBM's T. J. Watson Research Center is similar in concept to KRAFT's use of "small print" constraints [23]. The difference is that this work uses a rule-based formalism to specify contractual "fine print" in the form of business rules. Logic programming techniques are then used to reason with the rules.

## 7 Conclusion

The KRAFT network architecture was originally conceived to tackle the problem of gathering a specification for a configuration problem, including potential parts and their constraints. Its agent architecture therefore makes it very suitable to support virtual organisations, where the various partners (suppliers, reseller and customers) ally themselves together because they wish to interact by exchanging constraints. In order to do this, they are prepared to map data and constraints to an ontology that is shared but monotonically extensible [12]. Therefore, KRAFT is essentially restricted to cooperative interactions between agents.

Clearly, there is a cost associated with joining a KRAFT network, in that members must wrap their knowledge sources to conform to the shared protocols and knowledge exchange languages. However, KRAFT aims to demonstrate that the use of constraints offers an effective "middle way" between the off-putting complexity of KIF at one extreme, and the limited expressivity of the EDI approaches at the other extreme. To become successful, the KRAFT architecture would need to become a standard for establishing virtual organisations in which constraint fusion would be a primary mechanism for doing business. An organisation could then commit once to providing KRAFT wrappers for its own enterprise systems, in the knowledge that this would allow it to participate in KRAFT virtual organisations. In this context, KRAFT would become a layer that could exist above lower-level business information interchange formats based on XML described in Section 1.

The prototype network data services application has proven the concept of supporting virtual organisations by constraint fusion. In doing so, it has raised a number of issues that will be the subject of future work:

- the small scale of the prototype does not provide convincing evidence of the scalability of the KRAFT approach needed to cope with Internet-scale virtual organisations — further testing for scability on larger virtual organisations is needed;
- interactions within a KRAFT domain are currently fundamentally cooperative, except for the competition between vendors built-into the advertising and facilitation model — richer forms of competitive interaction need to be incorporated and tested;
- control is decentralised within the KRAFT network, and transactions are overly loose at present: more robustness and control is needed to support real-world ecommerce situations.

**Acknowledgements** KRAFT is a collaborative research project between the Universities of Aberdeen, Cardiff and Liverpool, and BT. The project is funded by EPSRC and BT.

## References

[1] J. Andreoli, U. Borghoff, and R. Pareschi. Constraint agents for the information age. *Journal of Universal Computer Science*, 1:762–789, 1995.

[2] R. Bayardo. InfoSleuth: agent-based semantic integration of information in open and dynamic environments. In *Proc. SIGMOD'97*, 1997.

[3] N. Carriero and D. Gelernter. Linda in context. *Communications of the ACM*, 32:444–458, 1989.

[4] L. Chiariglione. FIPA: agent technologies achieve maturity. *AgentLink Newsletter*, 1:2–4, 1998.

[5] CommerceNet. `www.commerce.net`, 2000.

[6] M. Cutkosky, R. Engelmore, R. Fikes, M. Genesereth, T. Gruber, W. Mark, J. Tenenbaum, and J. Weber. PACT: an experiment in integrating concurrent engineering systems. *IEEE Computer*, 26:8–27, 1993.

[7] N. J. Fiddian, P. Marti, J-C. Pazzaglia, K. Hui, A. Preece, D. M. Jones, and Z. Cui. A knowledge processing system for data service network design. *BT Technology Journal*, 14:117–130, 1999.

[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] E. Freuder and B. Faltings, editors. *Configuration: Papers from the AAAI-99 Workshop.* AAAI Press, Menlo Park, CA, 1999.

[10] E. Freuder and R. Wallace. Matchmaker agents for electronic commerce. In *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop*, Menlo Park, CA, 1999. AAAI Press.

[11] P. M. D. Gray, S. M. Embury, K. Y. Hui, and G. J. L. Kemp. The evolving role of constraints in the functional data model. *Journal of Intelligent Information Systems*, 12:113–137, 1999.

[12] P.M.D. Gray, A. Preece, N.J. Fiddian, W.A. Gray, T.J.M. Bench-Capon, M.J.R. Shave, N. Azarmi, and M. Wiegand. KRAFT: Knowledge fusion from distributed databases and knowledge bases. In R.R. Wagner, editor, *Eighth International Workshop on Database and Expert System Applications (DEXA-97)*, pages 682–691. IEEE Press, 1997.

[13] K. Jeffery. Metadata: an overview and some issues. *ERCIM News*, (35), 1999.

[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] R. Kalakota and A. Whinston, editors. *Electronic Commerce: A Manager's Guide*. Addison-Wesley, 1997.

[16] 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.

[17] Y. Labrou. *Semantics for an Agent Communication Language*. PhD thesis, University of Maryland, Baltimore MD, USA, 1996.

[18] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W.R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall 1991.

[19] D. E. O'Leary, D. Kuokka, and R. Plant. Artificial intelligence and virtual organisations. *Communications of the ACM*, 40:52–59, 1997.

[20] R. Orfali and D. Harkey, editors. *Client-Server Programming with Java and CORBA*. Wiley, 2nd edition, 1998.

[21] R. Plant and S. Murrell. The agile organisation: Technology and innovation. In *Using AI in Electronic Commerce: Papers from the AAAI-97 Workshop*, pages 26–32. AAAI Press, 1997.

[22] A. Preece, K. Hui, W. A. Gray, P. Marti, T. Bench-Capon, D. Jones, and Z. Cui. The kraft architecture for knowledge fusion and transformation. In *Research and Development in Intelligent Systems XVI (Proc ES99)*, pages 23–38. Springer, 1999.

[23] 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.

[24] RosettaNet. `www.rosettanet.org`, 2000.

[25] E. Schein. Innovative cultures and organisations. In *Information Technology and the Corporation of the 1990s*, pages 125–146. Oxford University Press, 1994.

[26] M. Singh. Commitments among autonomous agents in information-rich environments. In *Proc 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW)*. Ronneby, Sweden, 1997.

[27] 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.

[28] G. Wiederhold and M. Genesereth. The basis for mediation. In *3rd International Conference on Cooperative Information Systems COOPIS95)*. IEEE Press, 1995.