KRAFT: Supporting Virtual Organisations through Knowledge Fusion

Alun Preece, Kit Hui & Peter Gray

Department of Computing Science
University of Aberdeen
Aberdeen AB24 3UE
Scotland, UK
{ apreece | khui | pgray }@csd.abdn.ac.uk
www.csd.abdn.ac.uk/research/kraft.html

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 use of constraints as a knowledge exchange medium, and show how constraint fusion supports the design of customised products.

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 [Kalakota & Whinston, 1997]. 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 [Schein, 1994]. Successful integration requires that the members of the virtual organisation at least agree on mechanisms to exchange electronic documents and synchronise their workflows [O'Leary et al, 1997].

To minimise production times and product costs, the virtual organisation should be *agile*: relationships between members need to be dynamic and flexible [Plant & Murrell, 1997]. 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.

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 new XML standard promises to address the former problem, 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 [Jeffery, 1998].

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 [Harkey et al, 1998] 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 [Singh, 1997].

The KRAFT project (Knowledge Reuse and Fusion/Transformation) has an architecture that is suitable to support virtual organisations in which members exchange information in the form of constraints expressed against an object data model [Gray et al, 1997]. 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.

Configuration Design Problems

KRAFT was conceived primarily 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.

Configuration design problems were originally tackled by rules-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. In the KRAFT architecture, the domains of many of the variables will be entities stored in local databases held by individual companies. 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 found by specialised *mediator* agents and passed to a constraint solver together with other problem-specific restrictions. The solver then has to find feasible values to satisfy the constraints, as is common in engineering problems. 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.

Usually, configuration problems are solved by specially written pieces of software including pre-programmed constraints that take their parameter values from a number of data files prepared by the designer. The KRAFT architecture generalises this to allow both the parameters and the constraints representing the problem to be searched for and selected and brought together, over a network of nodes that may develop in various unanticipated ways. The agent architecture looks to be the best hope for coping with evolutionary change and the autonomy of different resource nodes.

For an example of the use of a KRAFT system, consider the problem of finding a number of parts that fit together to make something, or that work together in some way. Suppliers of these parts make catalogues, in the form of database tables, available over the network. However, the tables may have different semantics and hidden assumptions. These assumptions are often contained in an asterisked footnote or *small print* in the catalogue, for example: *this part must be mounted in a housing of adequate size*. Thus, it is not enough just to make a distributed database query to find a list of possible parts; we must also ensure that these parts satisfy various constraints.

It is the knowledge in these constraints which we aim to reuse by transforming it to work in the context of a *shared ontology* that is being used to integrate the data. Thus we might have a constraint stored as metadata in the database for the AbComponents catalogue:

```
constrain each w in widget
to have width(housing(w)) >= width(w) + 5
   and width(housing(w)) =< width(w) + 15;
```

This constraint is expressed in the KRAFT Constraint Interchange Format (CIF), based on the CoLan language used to express semantics in the object database P/FDM [Embury & Gray, 1995]. However, within the AbComponents database, the constraint might actually have been represented in some other form (as a trigger on a frame structure, for example); it must be translated into a CIF constraint before it can be used by the KRAFT network. To make use of widgets from the AbComponents catalogue, we must translate this constraint into a form consistent with a shared ontology. This requires an understanding of the different terminologies used in the AbComponents database and the shared ontology:

```
constrain each w in wotsit
   such that source(w) = "AbComponents"
to have distance(left_neighbour(w),
      right_neighbour(w)) >= width(w) + 2
   and distance(left_neighbour(w),
      right_neighbour(w)) =< width(w) + 6;
```

There are various ways to use the transformed constraint; in a design, for example, it could be transformed and *fused* with another constraint on a particular usage of the widgets/wotsits as parts of containers:

```
constrain each c in container so that
   each p in parts(c) such that p is a wotsit
      and source(p) = "AbComponents"
has internal_diameter(c) >= width(p) + 2 and
   internal_diameter(c) =< width(p) + 6;
```

Alternatively we could represent the fused constraints as a collection of clauses in normal form. We can now use this fused information in various ways as explained later.

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 [Labrou, 1996] and the de jure FIPA standard. 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 [Harkey et al, 1998], 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 [Jennings et al, 1996]) but the converse is not true.

A conceptual view of the KRAFT architecture is shown in Figure 1. KRAFT agents are shown as ovals. There are three kinds of these: user agents, wrappers, and mediators. All of these are in some way knowledge-processing entities.

Wrappers 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.

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.

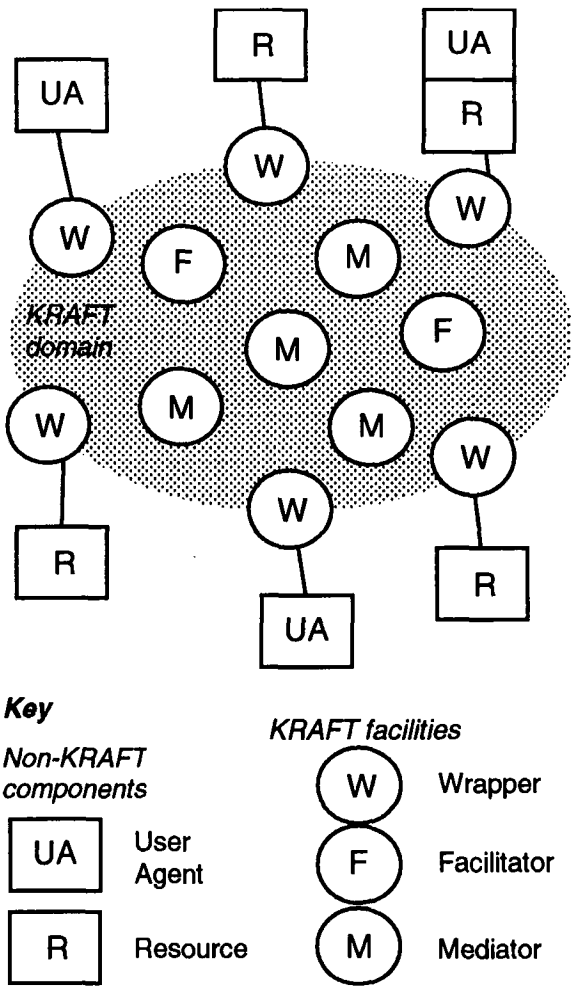

Figure 1 A conceptual view of the KRAFT architecture

Mediators 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.

Facilitators 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.

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) [Labrou, 1996]. Nested within the CCQL message is its content, expressed in the CIF protocol (Constraint Interchange Format).

It is worth noting that, syntactically, KRAFT messages are implemented as Prolog term structures. This is chiefly for convenience, as most of the knowledge-processing components 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.

Constraint Fusing 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 combining components from vendors. 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 pentium2 processor but not the win98 OS:

```
constrain each p in pc
   to have cpu(p)="pentium2"
   and name(has_os(p)) <> "win98"
```

For the components to fit together, they must satisfy certain constraints originating in the designer's knowledge base. 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 instructions attached to them as constraints. In the vendor database of operating systems, winNT requires a memory of at least 32 megabytes:

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

When we fuse all constraints together, we get the description of the overall constraint satisfaction problem:

```
constrain each p in pc
   to have cpu(p)="pentium2"
   and name(has_os(p)) <> "win98"
   and size(has_os(p)) =< size(has_disk(p))
   and if name(has_os(p))="winNT"
      then memory(p)) >= 32 else true
```

Related Work

Agent-based architectures are proving to be an effective approach to developing distributed information systems [Bayardo et al, 1997], 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 [Wiederhold & Genesereth, 1995]. KRAFT employs such an agent-based architecture [Gray et al, 1997] 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 virtual and agile organisations, but the area is still far from mature. Early projects such as PACT [Cutkosky et al, 1993] and SHADE [Kuokka et al, 1994] 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 customers. 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 model 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 [Jennings et al, 1996]. 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 [Bayardo et al, 1997]; 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 [Neches et al, 1991], in that some of the facilitation and brokerage methods are employed, along with a subset of the 1997 KQML specification [Labrou, 1996]. 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 [Andreoli et al, 1995]; the difference is that KRAFT recognises the need to transform constraints when they are extracted from local resources, typically for reasons of ontological or schema mismatch [Gray et al, 1997; Visser et al, 1997].

Current and Future Work

The KRAFT network architecture is being applied to the problem of gathering a specification for a configuration problem, including potential parts and their constraints. Its agent architecture makes it very suitable to support virtual organisations, where various vendors and potential customers ally themselves together because they wish to combine information. In order to do it they are prepared to conform (or map data and constraints) to an ontology that is shared but monotonically extensible [Gray et al, 1997]. 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.

Currently, and in the immediate future, work is focusing upon testing and evaluating the KRAFT architecture in a realistic business-to-business e-commerce scenario.

Acknowledgements

KRAFT is a collaborative research project between the Universities of Aberdeen, Cardiff and Liverpool, and BT. We acknowledge support from BT for Kit Hui working on the KRAFT project, which is also supported by EPSRC. We would like to thank Graham Kemp (Aberdeen), Zhan Cui (BT) and other KRAFT project partners for interesting discussions. Figure 1 is adapted, with permission, from [Gray et al, 1997].

References

Andreoli, J.; Borghoff, U.; and Pareschi, R. 1995. Constraint Agents for the Information Age, *Journal of Universal Computer Science* 1:762-789.

Bayardo, R. et al. 1997. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In Proc SIGMOD'97.

Cutkosky, M.; Engelmore, R.; Fikes, R.; Genesereth, M.; Gruber, T.; Mark, W.; Tenenbaum, J.; and J Weber, J. 1993. PACT: an experiment in integrating concurrent engineering systems", *IEEE Computer*, 26 (1): 8-27.

Embury, S. and Gray, P. 1995. Planning Complex Updates to Satisfy Constraint Rules Using a Constraint Logic Search Engine". In Proc 2nd International Workshop on Rules in Database Systems, 216-224. Springer-Verlag.

Gray, P.; Preece, A.; Fiddian, N.; Gray, W.; Bench-Capon, T.; Shave, M.; Azarmi, N.; and Wiegand, M. 1997. KRAFT: Knowledge Fusion From Distributed Databases and Knowledge Bases. In Proc 8th International Workshop on Database and Expert System Applications (DEXA-97), 682-691. IEEE Press.

Orfali, R. and Harkey, D. 1998. *Client-Server Programming with Java and CORBA*, 2nd ed. Wiley.

Jeffery, K. 1998. Metadata: an Overview and Some Issues. *ERCIM News*, No 35.

Jennings, N.; Faratin, P.; Johnson, M.; Norman, T.; O'Brien, P.; and Wiegand, M. 1996. Agent-Based Business Process Management. *International Journal of Cooperative Information Systems*, 5:105-130.

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

Kuokka, D.; McGuire, J.; Weber, J.; Tenenbaum, J.; Gruber, T.; and Olson, G. 1994. SHADE: Knowledge-Based Technology for the Re-engineering Problem.

Labrou, Y. 1996. Semantics for an Agent Communication Language, PhD Thesis, University of Maryland, Baltimore MD, USA.

Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; and Swartout, W. 1991. Enabling Technology for Knowledge Sharing", *AI Magazine* 12:36-56.

O'Leary, D.; Kuokka, D.; and Plant, R. 1997. Artificial Intelligence and Virtual Organisations", *CACM*, 40:52-59.

Plant, R. and Murrell, S. 1997. The Agile Organisation: Technology and Innovation. In AAAI-97 Workshop on Using AI in Electronic Commerce, 26-32. AAAI Press Tech Report WS-97-02.

Schein, E.. 1994. Innovative Cultures and Organisations. In *Information Technology and the Corporation of the 1990s*, 125-146. Oxford University Press.

Singh, M. 1997. 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.

Visser, P.; Jones, D.; Bench-Capon, T.; and Shave, M. 1997. An Analysis of Ontology Mismatches: Heterogeneity versus Interoperability. In Proc AAAI Spring Symposium on Ontological Engineering.

Wiederhold, G. and Genesereth, M. 1995. The Basis for Mediation. In Proc 3rd International Conference on Cooperative Information Systems (COOPIS95).