Corporate Memories as Distributed Case Libraries*

M.V. Nagendra Prasad
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003, USA
nagendra@cs.umass.edu

Enric Plaza
IIIA - Artificial Intelligence Research Institute
CSIC - Spanish Council for Scientific Research
Campus UAB, 08193, Bellaterra
Catalonia, Spain
enric@iiia.csic.es

Abstract

Rising operating costs and structural transformations such as resizing and globalization of companies all over the world have brought into focus the emerging discipline of knowledge management that is concerned with making knowledge pay off. Corporate memories form an important part of such knowledge management initiatives in a company. In this paper, we discuss how viewing corporate memories as distributed case libraries can benefit from existing techniques for distributed case-based reasoning for resource discovery and exploitation of previous expertise. We present two techniques developed in the context of multi-agent case-based reasoning for accessing and exploiting past experience from corporate memory resources. The first approach, called Negotiated Retrieval, deals with retrieving and assembling "case pieces" from different resources in a corporate memory to form a good overall case. The second approach, based on Federated Peer Learning, deals with two modes of cooperation called DistCBR and ColCBR that let an agent exploit the experience and expertise of peer agents to achieve a local task.

*The first author would like to acknowledge the support by the National Science Foundation under Grant Nos. IRI-9523419 and EEC-9209623. The second author's research reported in this paper has been developed at the IIIA inside the ANALOG Project funded by Spanish CICYT grant 122/93. The content of this paper does not necessarily reflect the position or the policy of the US Government, the Kingdom of Spain Government, or the Catalonia Government, and no official endorsement should be inferred.

1 Introduction

Rising operating costs and structural transformations such as resizing and globalization of companies all over the world have brought into focus the emerging discipline of knowledge management as one of the ways to raise productivity and profitability. Knowledge management is concerned with creating, maintaining and exploiting "knowledge infrastructures", and "organizational knowledge cultures", and "making knowledge pay off"(Pasahow 1996). Corporate Memories can play a crucial enabling role in making all this happen. Corporate memory represents the collective data and knowledge resources of a company including project experiences, problem solving expertise, design rationale etc.

In this paper, we discuss how viewing corporate memories as distributed case libraries can benefit from existing techniques for distributed cases based reasoning for resource discovery and exploitation of previous expertise. While it may not be the case that all resources in a company are amenable to this view, insofar as we can view a resource as a case base, we can exploit techniques developed in multi-agent systems and case-based reasoning communities to build powerful tools for knowledge access and manipulation. We present two techniques developed in the context of multi-agent case-based reasoning for accessing and exploiting past experience from corporate memory resources, that by their very nature are distributed. The first approach, called Negotiated Retrieval(Nagendra Prasad, Lesser, & Lander 1995; 1996), deals with retrieving and assembling "case pieces" from different resources in a corporate memory to form a good overall case. The second approach, based on Federated Peer Learning(Plaza, Arcos, & Martín 1996), deals with two modes of cooperation called DistCBR and ColCBR that let an agent exploit the experience and expertise of peer agents to achieve a local task.

2 Corporate Memory

Corporate memory consists of the sum total of the information and knowledge resources within an organization. Such resources are typically distributed and are characterized by multiplicity and diversity: company databases, machine-readable texts, documentation resources and reports, product requirements, design rationale etc.

A corporate memory facility that promotes an organization to leverage its existing information and knowledge assets through effective reuse can be crucial to handling internal and external pressures in an information-driven economy(Huynh, Popkin, & Stecker 1994). A number of benefits can arise out of a well thought out and implemented corporate memory infrastructure(CMI)(Huynh, Popkin, & Stecker 1994):

- Competitive pressures require quick and effective reactions to the ever changing market situations. The gap between the evolving and continuously changing collective information and knowledge resources of an organization and the employee awareness of the existence of such resources and their changes can lead to losses in productivity. CMI seeks to address this problem through "knowledge-empowerment of workers", thus enabling them to respond better to market opportunities.

- Timely availability of relevant information from resources accessible to an organization can lead to more informed decisions on the part of individuals (managers, project leaders etc), thereby promoting the effectiveness and viability of decentralized decision making.

- Promotes organizations to become learning systems and avoid repeating the same mistakes(Sharp & Lewis). Information about past projects - protocols, design specifications, documentation of experiences: both failures and successes, alternatives explored - can all serve as stimulants for learning, leading to "expertise transfer" and "cross-project fertilizations"(Vanwelkenhuysen 1996) within and across organizations.

Enablement of effective management of the know-how within a company mandates that a CMI incorporate characteristics like(Huynh, Popkin, & Stecker 1994):

- Semantically rich and flexible access mechanisms
- Automated management of potentially large-scale resource sets
- Efficient management of change and reuse given the dynamic nature of corporate information and knowledge resources
- Adaptability entailing learning from past experiences by recording them along with their context and re-instantiating them in similar future contexts to gain a level of predictability about these new situations.

The above requirements on CMI and the distributed nature of the resources comprising a corporate memory system provide compelling reasons for treating it as a distributed problem solving system. In the subsequent sections, we discuss the potential for distributed case-based reasoning approaches in dealing with access to corporate memory for semantically related but physically dispersed data and knowledge.

3 Distributed Processing vs Distributed Problem Solving

The task of information gathering in a distributed setting can be viewed in general terms as either distributed processing or distributed problem solving (DPS). Distributed processing is characterized by complete independence of subproblems. Agents need nothing other than local information to arrive at a subproblem solution of the required quality that can be synthesized with other agent subproblem solutions to arrive at a global solution. Distributed problem solving, on the other hand, is characterized by the existence of interdependencies between subproblems assigned to the individual agents, leading to a need for them to cooperate extensively during problem solving. They rely on communication to detect and exploit these interdependencies between subproblems. At the start, agents have only partial and incomplete views of global solution requirements. In spite of this deficiency in information, they may arrive at partial and tentative results that may be exchanged by the agents working on subproblems that are interdependent, to reduce the uncertainty that surrounds local problem solving. That is, agents may exploit the interdependencies between subproblems to their benefit(Lesser 1990; 1991).

In this paper, the information and knowledge resources comprising a corporate memory of an organization are viewed as distributed case bases. Doing so lets us map techniques for distributed problem solving into tools for knowledge manipulation in corporate memories.

The first technique is based on a method proposed by Nagendra Prasad, Lesser and Lander(Nagendra Prasad, Lesser, & Lander 1996) for retrieval from distributed case bases. This method can be seen as an instantiation of aspects of the cooperative information gathering(CIG)(Oates, Nagendra Prasad, & Lesser 1994) approach to intelligent information gathering from networked information resources. This approach relies on the "FA/C" paradigm(Lesser 1991) previously developed as a framework for distributed problem solving. Oates, Nagendra Prasad and Lesser(Oates, Nagendra Prasad, & Lesser 1994) provide an extensive discussion as to why it is better to treat information gathering in a networked environment as distributed problem solving. In a CIG task, potentially useful constraints may exist between different pieces of information. The discovery and exploitation of such constraints is necessarily a dynamic and incremental process that occurs during problem-solving and entails communication of partial results among agents in a timely and selective manner, to augment each agent's local view with a more global view. Given the incomplete nature of the local views of the individual agents, another important aspect of CIG is the explicit recognition of the role of solution and control uncertainty. Coupled with the fact that resources and time for conducting a search are limited in real-life problems, this leads to the notion of *satisficing search.* Another aspect of CIG is the explicit recognition and exploitation (or avoidance) of *redundancy*, leading to increased robustness or decreased resource demands depending on the context and the structure of the domain.

The second technique is the Federated Peer Learning-based cooperative Case-based Reasoning (Plaza, Arcos, & Martín 1996). Two modes of cooperative case-based reasoning are discussed: DistCBR where an agent can delegate its authority to another peer agent to solve a problem and ColCBR where an agent maintains authority while exploiting the experience of a peer agent. While remote evaluation capability supports DistCBR, ColCBR mode is supported by remote programming (or mobile code) capability of the underlying representation and communication framework. These modes let an agent exploit the collective memory in a distributed environment in a lazy, on-demand way.

In the following section, we first discuss why some of the resources comprising a corporate memory can be viewed as case bases. We then briefly discuss the Negotiated Retrieval Algorithm and the Federated Peer Learning-based cooperative Case-based Reasoning modes. Readers interested in further details are urged to refer to (Nagendra Prasad, Lesser, & Lander 1996; Plaza, Arcos, & Martín 1996).

4 Retrieval and Reasoning in Distributed Cases-bases

A Case-based Reasoning (CBR) system uses lazy "generalization" from past similar cases to the present task. Past similar cases are retrieved through similarity estimates between the current problem P and the precedent cases CB that the system has access to. The crucial assumption in a CBR system is that the more similar the current problem P is to a precedent $C \in CB$, the more similar the solution of P is to the solution of C. In recent times, CBR techniques have enjoyed an immense popularity among researchers and practitioners of AI, building intelligent tools for a number of applications(Aamodt & Plaza 1994). Techniques for distributed case-based reasoning are being developed with the aim of leveraging the insights gained from building and using such applications but at the same time coping with the distributed nature of the knowledge available in many applications. Viewing Corporate Memories as Distributed Case Libraries (or Distributed Case Bases) provides us with ways to exploit these techniques to develop semantically rich and flexible tools for knowledge management in distributed environments.

4.1 Corporate Memories as Distributed Case Bases

In its most general form, a case base is a source of complex data stored in specific formats. A number of knowledge and data sources that comprise a corporate memory could be defined as case bases in this sense. Case bases could arise from formated records of useful employee experience and/or expertise. Alternately, *Case-Knowledge Engineers* or content experts could design case bases of relevant experience by populating them with collections of records in appropriate formats. Certain unstructured databases like text databases can also be converted to case bases by generating semantic descriptors characterizing each of its documents. Much of the work in information extraction and text summarization concentrates on generating such descriptors(Lehnert *et al.* 1992). Given such descriptor generating capabilities, any set of databases with inter-related data can be treated as distributed case bases. Another promising alternative involves creation of *metadata*(Weibel 1995) that is an informative record attached to a document (structured or unstructured). A typical metadata record could contain elements like subject, author, title, object type, relationship to other elements, coverage etc(Weibel 1995)[1]. Development of metadata becomes especially feasible if the authors of a resource or a document could be encouraged to create such a description. Perhaps some of the most important sources of distributed case bases available to a corporation beyond its organizational boundaries are the distributed digital libraries with mutually related information, like PARTNET(Partnet) on the Internet. PARTNET is WWW-accessible distributed electro-mechanical component library developed by the University of Utah's Mechanical Engineering Department as a resource to connect designers and engineers with parts suppliers. In this paper, we view a case base as not just a passive data store but an active one that can reason about its contents and their appropriateness in the context of a query. Thus, when we use the term "agent" and a case base interchangingly, we mean that the agent is an active case-based reasoning entity that can reason about its local case knowledge.

[1]Metadata Workshop held on March 1-3, 1995, in Dublin, Ohio addressed the issue of appropriate metadata elements for document-like objects and identified an initial set of thirteen elements called the **Dublin Core**(Weibel 1995).

4.2 Negotiated Retrieval

4.2.1 Overview

Most of the literature on information gathering deals with locating, gathering and selecting the best response to a query from among a multitude of responses from different data repositories(Arens *et al.* 1993; Bowman *et al.* 1994; Oates, Nagendra Prasad, & Lesser 1994). Nagendra Prasad, Lesser and Lander(Nagendra Prasad, Lesser, & Lander 1995; 1996) introduced a different model of response to a query where no single source of information may contain the complete response to a query; necessitating piecing together mutually related partial responses from disparate and possibly heterogeneous sources. A complex query is presented to a set of agents, each of which is responsible for retrieving information relevant to a part of the query. The agents negotiate to piece together a mutually acceptable response to the query. This type of retrieval, defined as *Negotiated Retrieval*(Nagendra Prasad, Lesser, & Lander 1995; 1996), adopts the above view of a query to a set of distributed case bases in the corporate memory context. More specifically, a response to a query involves assembling related pieces of information from different case bases to form a composite case. The agents have to cooperatively retrieve mutually acceptable responses while negotiating compromises to resolve conflicts. Each agent retrieves subcases from its local case base and all agents together assemble a mutually acceptable overall case from these subcases to produce a response to the user's information needs.

Information requirements of many real life applications rely on such distributed case bases. Let us illustrate this with an example: a management consultancy firm is faced with the need to quickly build a cross-functional team by drawing from an organization-wide talent pool for the purpose of helping an inventor research the market for a product, analyze pros and cons of the competition, and locate interested venture capitalists. We can imagine an automated assistant for querying a multi-agent system that assembles the team by letting each agent access its own resume database of various experts for a particular aspect of the project: technical, management, sales, etc. In assembling the team, the agents need to consider interactions between the requirements of experts for different aspects like for example, all experts willing to work on that type of project or all technical experts on the team being familiar with a particular computing environment. The distributed case bases in this example are the resume databases for different expertise, with descriptor generators that extract features like *"project_types_willing_to_workon"* and *"familiar_computing_environments"*.

4.2.2 Negotiated Retrieval Algorithm

Below, we briefly present a summary of the Negotiated Retrieval Algorithm. Interested reader is referred to (Nagendra Prasad, Lesser, & Lander 1995; 1996) for a more formal treatment and empirical studies.

Negotiated Retrieval is viewed as a distributed constraint optimization problem where each agent has a set of subcase consistency constraints in addition to the local case base. These constraints arise in a number of ways:

- Constraints could arise from useful context information attached to the metadata specification of a resource or documents or as qualifiers attached to particular elements of the metadata.

- Constraints could arise from the domain-specific knowledge an agent has about the requirements of the context in which its local subcases can usefully participate.
- Constraints could also be derived from queries that may be required to specify them in addition to user's information needs.

A case can be viewed as a set of feature-value pairs. The set of constraints that an agent has may be defined on both "local" and "non-local" features. Local features of an agent are those features that the agent uses to represent its local subcases. Non-local features are those features that some other agent uses to represent its local cases. Thus, a constraint can be viewed as a relationship between subcases of one or more case bases and is enforced through interrelationships between features of these subcases.

A case has a number of associated attributes that involve measures of certain characteristics of a case and are functions of the feature values of a case. Examples of attributes include reliability, quality, uncertainty and cost. In addition to being acceptable to all agents, it is desirable that a case be optimized along the attribute set. These requirements lead to organization of an agent's constraints as *soft* constraints and *hard* constraints where the former set represents solution preferences and the later set represents those constraints that are relaxed only as a result of explicit recognition by the agents that the set is too constrained to lead to a mutually acceptable solution. Relaxing a soft constraint may only involve penalties in terms of loss of optimality in the desirable attributes. For example, relaxing a soft constraint may lead to less robust solutions. Softness of a constraint represents its degree of flexibility. On the other hand, hard constraints are generally not relaxed except with an explicit understanding that the resulting responses satisfy the query specifications only partially or are consistent with only a subset of the agents.

A *partial case* is a partially evolved response to a query. A partial case is obtained by composing the subcases from one or more agents but it is not yet acceptable to all agents (perhaps because it needs more subcases or because other agents have not checked it against their consistency constraints). *Projection* of a partial case on to a set of features is the set of feature-values pairs in the partial case for that set of features.
The agents execute the Negotiated Retrieval Algorithm (NRA) as follows:

Phase I: Local Retrieval

If an agent received feedback from other agents about previous violations, it first *assimilates* it (details of the assimilation process are provided later). Some of the agents, using the relevant portions of the user query and the presently available information on the problem-solving state (including previously tried solutions, conflicts they caused and feedback in the form of advice on violated constraints from other agents), retrieve the seed subcases around which the rest of the case evolves. Other agents have too poor a local view to perform retrieval without additional help and hence wait to extend partially assembled cases. If a seed agent fails to retrieve a case, it can relax some local constraints until it finds a case or gives up at certain point. In general, a locally retrieved subcase is re-instantiated in the present problem context during this phase. Re-instantiation could also involve adaptation of the retrieved subcase to the new context.

Phase II: Sub-case Integration, Partial Case Extension and Conflict Detection

Agents try to "merge" the local subcases to form larger partial cases. It involves checking for consistency and interactions among the sub-cases retrieved by different agents. In the situation where the agents are trying to physically represent the overall episode at a single central repository, the agents can easily obtain the relevant information for consistency checking by looking at the other agents' sub-cases in the central repository. However, integration need not necessarily lead to a combination of the sub-cases at a single physical location. In this situation, the agents have to exchange projections of partial cases. For example, agent A_1 wanting a consistency check on one of its partial cases by A_2 has to send to A_2, a projection of that partial case with respect to the relevant features of the constraints at agent A_2. Projection information alone is sufficient for Agent A_2 to check for the consistency of that partial case with respect to its constraints (more details can be found in (Nagendra Prasad, Lesser, & Lander 1996)). If constraints are not violated then the subcases are considered "merged" due to their mutual consistency. Those agents who could not retrieve subcases due to insufficient information in Phase I, try to extend these partial cases by adding their subcases. An agent intending to extend a partial case obtains an appropriate projection of that partial case to serve as an anchor for the local subcase retrieval.

If any violations are detected during a merge or an extend operation due to poor or infeasible values on local or non-local features, go to Phase III; else exit.

Phase III: Conflict Resolution through Negotiation

Each agent categorizes its violated local constraints into pre-enumerated classes of violations and uses a set of conflict resolution strategies associated with each of these categories to generate a set of advice as feedback to itself and the other agents involved in the partial case that led to the conflicts. The set of advice could range from domain independent strategies to highly domain specific ones(Nagendra Prasad, Lesser, & Lander 1996):

1. Some of the agents may do their local retrieval using similarity measures based on "closeness" of the retrieval requirements to the cases in the archive. Such agents can be advised to broaden their search by obtaining cases with poorer similarity values or different similarity measures.
2. Some of agents may retrieve a case and massage it using an adaptation strategy to fit the new situation. An agent could advise another agent to modify the retrieved case in a different way — use a different adaptation strategy.
3. In systems where it is possible for agents to be associated with some knowledge of the importance of a particular feature's values and constraints for the overall case, this knowledge can serve as basis for generating advice to relax a soft

constraint involving certain parameters. Alternately, the advice can be in the form of changes to the values or ranges of certain features in order to obtain better local solutions.

4. An agent detecting lack of progress either locally or at other agents (based on the projections it receives from those agents) could advise some of them to relax their hard constraints. This is expected to take the retrieval process to qualitatively different regions of the case base. Just as with soft constraints, the choice of which constraint to relax is based on system-wide knowledge or generic strategies associated with some or all of the agents about the importance of various types of constraints for the overall case.

5. Some of the agents may have capabilities to analyze particular features of the solution space that lead them to recognize opportunities for more efficient customized search strategies. They can together decide to play out specific roles in this kind of customized search. Lander(Lander 1994) presents a good example of a customized search called *linear compromise* where agents, upon recognizing the linear nature of their solution space, decide to exchange end points and extrapolate between them to find the intersection point as a mutual compromise solution.

Upon generating feedback, goto Phase I.

Assimilation of feedback advice from other agents enhances an agent's view of the non-local requirements. An agent assimilating feedback can indulge in a process as complex as the generation of feedback. It may involve relaxing a constraint or adding a new constraint to the local set of constraints. Assimilation may be context-sensitive, leading to constraints that are applicable only in specific contexts or to specific types of partial cases at the local agent. In addition, the assimilation process may also involve transformations where an agent uses the feedback from other agents to generate its own local constraints rather than directly incorporate the feedback.

The Negotiated Retrieval Algorithm is an asynchronous parallel distributed constraint optimization search to obtain a good overall episode assembled from case pieces. The asynchronous nature of the search arises from the fact that an agent could be in any phase of the NRA for a given case evolution at a given time. More than one partial case could be evolving simultaneously and an agent could be in different phases of Negotiated Retrieval for different partial cases at any given time. The NRA algorithm is very general and a system may go through only some or all of these phases to achieve coherent retrieval of good overall cases.

Another feature of Negotiated Retrieval Algorithm is its ability to work with heterogeneous agents. While we cast NRA as constraint optimization search, it is not essential that the agents represent these constraints in any particular form - they could be procedural or declarative and in multiple forms. Internally, the agents could be using disparate knowledge organizations or problem solving control organizations. Agent detects constraint violations based on projections that basically represent information about the features that this agent as well as some other agents know about. The only requirement is the ability of an agent to translate a projection or feedback into its local language or from its local language to the language of another agent. These types of translation mechanisms are commonly called "wrappers"(Genesereth, , & Ketchpel 1994) and are used to enable a set of heterogeneous or stand alone systems to function as a multi-agent system.

4.2.3 CBR-TEAM: A multi-agent design system

In this section, we present a brief summary of the CBR-TEAM system(Nagendra Prasad, Lesser, & Lander 1996) that uses negotiated retrieval to compose coherent design cases. Note however, that our experience with negotiated retrieval is still rudimentary and we will be able to give further insights into its effectiveness in future.

CBR-TEAM, whose core is derived from TEAM(Lander 1994), is a parametric design system that uses a set of heterogeneous cooperative agents for designing steam condenser components. It consists of three agents - **`motor-agent`**, **`pump-agent`** and **`vbelt-agent`** - that are responsible for the design of the **`motor`**, **`pump`** and **`vbelt`** components of a steam condenser. The user gives a problem specification that consists of minimum head size for the pump in the required design. The agents in CBR-TEAM retrieve and use suitable members from libraries of manufacturer-specified models for designing their components and use the negotiated retrieval strategy to arrive at mutually acceptable designs. When the components of the individual agents are being assembled, violation of constraints due to mismatches on shared parameters lead to information exchange followed by redesign. Interface parameters are those features of a component that are shared by more than one agent. All the relevant agents have to reach an agreement on the values of the shared parameters.

During the initial phase of retrieval, the agents may have only partial information on the requirements of other interacting components. So, each of the agents chooses the lowest cost component based on the information available to it. Trying to assemble these components into an overall design may lead to conflicts due to mismatches in the parameters that are shared by two or more components. For example, **`motor`** and **`pump`** components have **`required-pump-power`** as a shared parameter and both **`motor-agent`** and **`pump-agent`** impose their own set of constraints on this parameter. A mismatch on this parameter involves one agent assigning a value to the parameter that violates the constraints in another agent. When a conflict is detected, the agent detecting it sends feedback to the other agents involved. In CBR-TEAM, feedback involves communicating the relevant violated explicit constraints, all of which are single-parameter numerical-valued constraints. When an agent receives feedback from others, it assimilates the feedback. Assimilation involves adding the feedback constraints to the set of local constraints to effect further searches from there on. The agents iteratively perform further rounds of retrieval using the previous information and the new requirements from other agents to get better cases to be assembled into a design that does not produce the same conflict.

4.3 Federated Peer Learning

Plaza, Arcos and Martin(Plaza, Arcos, & Martín 1996) discuss two modes of cooperation among case-based reasoning (CBR) agents where an agent can leverage the learning capabilities or past experience of peer agents[2] to achieve a task or solve a problem. These modes are developed within the *Federated Peer Learning* (FPL) framework (Plaza, Arcos, & Martín 1996) that aims to study cooperative problem solving among agents possessing either same or different capabilities and incorporating potentially different knowledge and problem solving behaviors based on their individual learning and experience. Cooperative problem solving in such a system can result in bringing wide range of experience to bear on a task at hand in an agent. The approach taken here to achieve cooperation is through communication using the *Noos* representation language developed at IIIA for integrating learning and problem solving(Arcos & Plaza 1996). *Plural* Noos is an extension of Noos that allows communication and mobile (or "migrating") tasks and methods (to achieve these tasks) among agents that use Noos as a representation language. In particular, we will show two modes of cooperation among CBR agents: Distributed Case-based Reasoning (DistCBR) [3] and Collective Case-based Reasoning (ColCBR). Intuitively, in DistCBR cooperation mode an agent A_i *delegates its authority* to another peer agent A_j to solve a problem — for instance when A_i is unable to solve it adequately. In contrast, ColCBR cooperation mode *maintains the authority* of the originating agent: an agent A_i can transmit a mobile method to another agent A_j to be executed there. That is to say, A_i *uses the experience* accumulated by other peer agents while maintaining the control on how the problem is solved.

Each of the cooperating agents in DistCBR and ColCBR is capable of solving the overall task by itself (most of time) unlike in Negotiated Retrieval where agents are specialists at specific subtasks.

4.3.1 Representation and Communication

The approach taken to develop cooperative CBR is to extend Noos, a representation language for integrating learning and problem solving that has been used to develop several CBR systems. In this section we first present some basic notions of the language, and later the *Plural* extension that supports communication and cooperation among CBR agents using Noos.

Noos is a reflective object-centered representation language designed to support knowledge modeling of problem solving and learning(Arcos & Plaza 1994; 1996). Noos is based on the task/method decomposition principle and an analysis of knowledge requirements for methods — and it is related to knowledge modeling frameworks like KADS (Wielinga *et al.* 1993) or components of expertise (Steels 1990). A *method* models a way to solve a task. A method can be elementary or can be decomposed into subtasks. These new (sub)tasks can be achieved by their corresponding methods in the same way. For a given task there may be multiple alternative methods (alternative ways to solve the task). For

[2]By *peer agents* we mean agents that are capable of solving the task that another agent has at hand.

[3]Note that this mode of cooperation unfortunately carries the same name as our use of the term distributed case-based reasoning for agents more generally doing case-based reasoning in a distributed manner. The name for this mode of cooperation was introduced in (Plaza, Arcos, & Martín 1996) where cooperative CBR was used for agents more generally doing case-based reasoning in a distributed manner. In order to avoid confusion, in this paper, we use the acronym DistCBR for this specialized FPL cooperation mode and the terms cooperative CBR and distributed case-based reasoning are used interchangeably for the more general situation of agents doing case-based reasoning in a distributed manner.

instance, a `CBR` method(Aamodt & Plaza 1994) is decomposed into **`retrieve`**, **`select`** and **`reuse`** subtasks and there are several possible methods to access stored cases, select one of them according to some criteria, and finally reuse the solution. Retrieval methods allow Noos to inspect and analyze previous specific situations in the episodic memory. The reuse methods re-instantiate solutions to precedent problems in the current context or construct new solutions using the precedent solutions and the current problem. Decision-taking in Noos is modeled by a preference language that allows the specification of the conditions under which an alternative is better than the others. Reasoning about preferences permits an agent to select a method from a set of alternatives or to choose to cooperate with an agent from a set of associate agents (more on this later).

Noos is a representation language based on *descriptions*. A description is formed by a collection of features. The values of features are constants or other descriptions. This approach is close to the *ψ-term* formalism (Aït-Kaci & Podelski 1993). Domain knowledge is represented in Noos by descriptions of the concepts in that domain. Descriptions have a correspondence to labeled graph representations as shown in Figure 1 that is a description of an experiment in the chromatography domain(Plaza, Arcos, & Martín 1996).

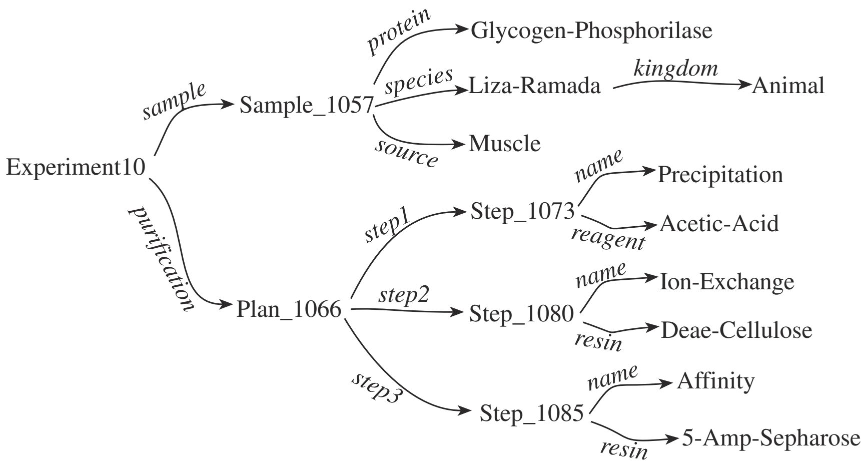

Figure 1: A case description in CHROMA.

Methods are also represented as descriptions. The features of a method description represent the subtasks into which that method is decomposed. Methods are defined by refinement from a set of built-in methods. The set of built-in methods in Noos are those of a general-purpose language plus some constructs enabling introspection. The uniform representation of methods as descriptions is what allows *Plural* Noos to transmit both entity descriptions and methods in the same way over the network.

4.3.2 Agent Communication with *Plural* Noos

Plural provides a seamless extension of Noos to support distributed scoping and reference for all the basic Noos constructs. An agent in *Plural* is a particular Noos application with a network address, and the acquaintances of an agent are those agents with addresses known to it — as in the actors model. The CBR cooperation modes use three *Plural*

Noos capabilities: network references, remote evaluation, and mobile methods. Using *Plural* Noos, arbitrary Noos descriptions (of entities and methods) can be transmitted over the network from one agent to another. In particular, cases and CBR methods can be transmitted from a CBR agent to another.

Network references extend Noos references to agents over the net. Syntactically, a reference to a feature in an **agent-i** like **(>> feature of entity)** once transmitted to a new agent **agent-j** becomes a network reference equivalent to **(>> feature of entity at agent-i)** in **agent-j**; and an identifier of an entity in **agent-i** like **entity55**, once transmitted to a new agent, becomes a network reference like **entity55@agent-i** in **agent-j**. Network references are transmitted over the network: if **entity55@agent-i** is a value of the feature **my-friend** of **entity99** in **agent-j** and a new agent has the reference **(>> my-friend of entity99 at agent-j)**, it will get the original network reference **entity55@agent-i**. Network references avoid the problem of maintaining state when objects with state are copied over the network. State is local to agents, and when a description is referenced by another agent, a network reference is transmitted.

If we view a Noos description as a labeled graph, transmission of a description starts at a selected node (for example, the root in Figure 1) and "copies" the graph to the destination as follows: if a graph node is a constant, (like a number or a string) a fresh copy is produced. Otherwise a network reference to that node is created. Since Noos performs lazy evaluation, not all the nodes in a graph are transmitted when the root is referenced, but only those needed by remote references. Path equality (sharing) and circularities in the graph are preserved.

Remote evaluation allows an agent to use a method owned by another agent — as in remote procedure call (RPC). Specifically, remote evaluation allows an agent **agent-i** to ask another agent **agent-j** to execute a specific method **method-k@agent-j** for a given **problem-n** of **agent-i**, as in the expression **(noos-eval (method-k@agent-j problem-n) at agent-j)**. In this process, **agent-j** receives the network reference **problem-n@agent-i** and applies **method-k** to it. During evaluation, further references in **agent-j** to features of **problem-n@agent-i** are interpreted as network references that automatically lead to communication to **agent-i** asking for the value of that feature. **Agent-i** is responsible for inferring that value and transmitting its network reference to **agent-j**.

For some cooperation modes it is necessary to support so-called *mobile code* or *migrating programs*. Mobile methods are supported by *Plural* Noos through the capability for transmitting method descriptions. A mobile method description is first defined in an originating agent **agent-i**. Then, the *Plural* Noos construct **jump** can be used to bind the mobile method with the appropriate references and to instantiate the method at the destination agent.

When a method jumps to a remote agent, the whole task/method decomposition of the mobile method is "copied" in the following sense: the name of the built-in of which the method is a refinement is transmitted, as well as its subtasks. Recursively, the methods defined in the originating agent for those subtasks are also "copied". The references of the mobile method in the originating agent are transformed to network references as explained previously.

4.3.3 Two Modes of Cooperation for CBR Agents

Cooperation among CBR agents involves exploiting the set of precedents in the collective memories of all the agents for use in similarity-based reasoning. There are two general ways to do so: DistCBR and ColCBR. Intuitively, both DistCBR and ColCBR are based on solving a problem by reusing the knowledge learned by other CBR agents. Given an agent (the *originator*) trying to solve a problem, the difference between both modes is in the similarity-based reasoning method used: that of the originator or that of the CBR agent that is helping the originator.

- DistCBR is based on an agent A_i transmitting the problem and the task to be achieved to another agent A_j. Agent A_j uses its own CBR method and its case base CB_j to achieve the task and send the results back to agent A_i. In case of failure, a failure token is sent back and A_i can iterate the cooperation tasks with the next agent of its preference.

- ColCBR is based on an agent A_i transmitting the method that is to be used to solve a task, in addition to the problem and the task to be achieved, to another agent A_j. Agent A_j will use its case base CB_j and the method sent by A_i to achieve the task and send back the results. In other words, the originator is using the memory of the other agents as an extension of its own — as a collective memory — by means of being able to impose on other agents the use of its own CBR methods. In case of failure, a failure token is sent back and A_i can iterate the cooperation tasks with the next agent of its preference.

From the standpoint of implementing these cooperation modes, we can say that DistCBR is supported by the remote evaluation capability and ColCBR is supported by remote programming (or mobile code) capability of *Plural* Noos.

4.3.4 Multi-agent Protein Purification

We now discuss the application of the two cooperation modes introduced above in a system that recommends chromatography techniques to purify proteins from tissues and cultures. There are a number of proteins and associated chromatography techniques currently in use in a larger number of industrial chemical labs. Each of these labs may face its own subset of problems that it routinely solves and in the process it develops expertise for handling them. It may also face problems that seldom occur at its location but occur frequently at other locations — leading to the developemnt of expertise at those locations. Different locations may thus have different methods for case-based reasoning that rely on knowledge modeling analysis of their particular problems and local expertise and biases. This gives rise to the need for cooperation to exploit peer expertise.

Our multi-agent CBR system consists of a number of CHROMA agents that can recommend chromatography techniques for protein purification. Each CHROMA agent can be *configured* using Noos. It allows the configuration of a CBR system through a knowledge model analysis of the domain(Arcos & Plaza 1994; 1996). Such a configuration is done with the component blocks provided by Noos — like generic retrieval methods — that are refined (or biased) in order to incorporate the domain knowledge that has been modeled. In CHROMA, the domain knowledge is used to characterize which features are more important for judging the similarity between a current problem and a precedent case. Noos allows the expression of such knowledge by means of retrieval methods and preference methods. Such an abstraction permits one to ignore implementation details like the indexing algorithms and, most importantly, permits the communication of such methods among CBR agents(Arcos & Plaza 1994; 1996). This allows a CBR agent to not only exploit the cases in its own case-base but also those cases known by other agents.

Learning in CBR is lazy: a CBR system imposes a partial order among (a relevant subset of) the past examples based on the current problem. The solution of a problem is determined by the solution of the case(s) that is maximal in the partial ordering established by preferences. Thus, solutions proposed by the system are a function of the individual experience of the CBR system plus the domain knowledge given by the system designers during the knowledge modeling stage. The CBR method in a CHROMA agent is configured as follows:

1. **Goal-driven Retrieval** This is a generic method that selects from memory all cases obeying a constraint declared as pattern. Intuitively, it retrieves all cases subsumed by (all cases that match) the pattern. Domain knowledge in CHROMA requires that only cases whose **`protein`** feature has the same value as in the current problem are retrieved. This form of retrieval is called goal-driven retrieval (since the protein is the goal in our process) and can be represented by a general method called **`retrieve-by-determination`**.

2. **Domain Selection Criteria** This component is a **`preference`** method that imposes a partial order among retrieved cases. In CHROMA there are three basic preferences:

   Preference n.1 Domain knowledge in CHROMA states that usually the most important criterion for similarity is having the same value for a feature in the **`source`** and in the current problem.

   Preference n.2 This preference method relies on the **`species`** feature i.e. the species of the sample tissue or culture from which the protein is purified. This preference discriminates the retrieved cases that are indistinguishable using preference n.1.

   Preference n.3 This method is a preference based on the **`kingdom`** taxon of the source. It is used to discriminate among the retrieved cases that are not distinguishable by the preceding preference methods.

3. **Reuse** Finally, the **`reuse`** method re-instantiates the purification plan of the most relevant precedent chosen using the above domain preferences.

In the multi-agent extension of CHROMA, each laboratory has a specific agent that can support this CBR method or a similar one. Different CBR methods can be derived by supplementing or substituting the general preference criteria with specific ones arising out of the kinds of problems an agent regularly solves. For instance, for a given tissue, the species criterion could be more relevant than the source criterion. Thus, each CBR agent possesses selection criteria adapted to its own experience.

We have seen that CBR methods are decomposed into three main tasks: `retrieval`, `selection` and `reuse`. The multiagent CHROMA application specifies the `reuse` task to be local to the agent involved in solving a problem, while `retrieval` and `selection` tasks can be delegated to other agents. In DistCBR an agent A_i has a new method encompassing the `retrieval` and `selection` tasks; this `retrieve&select` method is declared public. Another agent A_j can specify to the agent A_i to apply that method to a current problem and A_j will receive as result, the network reference of the best precedent case C_i^{best} in A_i memory. Then A_j can access the information of C_i^{best} in A_i (essentially the solution) and reuse or adapt that information to the current problem.

In Dist CBR, the precedent case C_i^{best} was selected using the criteria embodied in A_i `retrieve&select` method. On the other hand, in ColCBR an agent A_j uses its own CBR method on any federated agent—in essence accessing and using the memory of the another agent as if it was its own, and hence the name of *collective memory*. In ColCBR an originating agent A_j can transmit its `retrieve&select` method to another agent A_i who is responsible for using it on its own memory and experience and sending back the best precedent case C_i^{best} to A_j. Here C_i^{best} is the best according to the criteria of A_j and the experience of A_i.

In fact, the way ColCBR is performed is slightly more complex. In ColCBR we want to find the best precedent case in the collective memory of all the peer agents, but each agent may respond with the best case from its case base. The agent can not give any assurances about the goodness of the case with respect to the entire collective memory of all the agents. So, the following ColCBR method is used to select the the best precedent case:

1. **Retrieve & Select** This mobile method is sent to the peer agents, and each of the peers responds with the network reference of the (locally) best precedent case C_i^{best}.

2. **Global Select** The originating agent uses the preference criteria to rank the set of locally best cases C_i^{best} in order to select the globally best one(s).

3. **Reuse** The (globally) best precedent case is reused as before by the originating agent.

4.4 Discussion

The Negotiated Retrieval and the FPL-based cooperative CBR represent two attempts at exploiting previous experience, distributed within a corporation, for solving new problems. Each of the two approaches brings with it a set of conditions and problem features for which it is most appropriate.

Negotiated Retrieval is based on a model where the response to a query is derived from composing partial responses from distributed case bases. Tasks like assembling cross-functional teams or assembling a set of documents for various aspects of a project from the documentation set available from previously executed projects (to let a project leader derive leads from them for the present project) or accessing a set of relevant components from manufacturer-specified online component catalogues for various design assembly stages represent a few examples where NRA provides a useful tool for the users. On the other hand, DistCBR and ColCBR represent attempts to benefit from the collective experience of peers in a corporation. Each of the peer agents can solve the task on hand by itself. However, their experiences are different and diverse and thus each of them can potentially bring some unique experience to the task. Exploiting this diversity and richness is what these two modes of cooperation attempt to do. Tasks like a project manager's agent exploiting the experience of its peers for project cost estimates or an agent exploiting the experience of a group of expert agents for a new marketing initiative are examples where these modes of cooperation can come in handy[4].

DistCBR and ColCBR rely on knowledge modeling to flesh out a domain to capture the recursive structure of task-method-subtasks relationships and manage the conflicts and preferences among subtasks or submethods. This represents a knowledge intensive approach to cooperative CBR. On the other hand, Negotiated Retrieval is a search-intensive approach to cooperative CBR where preferences and harmful interactions are managed by augmenting the retrieval of individual subcases with search during integration.

Lastly, DistCBR and ColCBR have certain specific representational requirements like agent control being organized to be able to work with descriptions in Noos language. Thus resources in a corporate memory setting have to be augmented with such capabilities to be able to exploit these two modes of cooperation. However, this is not a serious limitation because Noos is a very general and powerful representation language. Negotiated Retrieval is designed to work with heterogeneous agents that can be equipped with "wrappers" to achieve cooperative communication without having to change the representations of their internal problem solving control.

5 Conclusions

Corporate Memory forms an important enabling component in any knowledge management strategy of a company. In this paper, we propose that such memories be viewed as distributed case bases in order to facilitate exploitation of techniques from multi-agent systems and case-based reasoning to build flexible and powerful tools for knowledge access and manipulation. Negotiated Retrieval deals with piecing together partial responses from different sources to evolve a coherent response to a query. It augments CBR agents with distributed constraint optimization search capabilities to avoid harmful interactions among case pieces and assemble "good" candidate overall responses from partial responses of individual agents. FPL-based cooperative CBR proposes two cooperation modes among CBR agents for exploiting "collective memories" of peer agents. In DistCBR, an agent transmits the task to be achieved to another agent, and the host agent uses its CBR methods and its local case base to achieve the task for the originating agent. In ColCBR, an agent transmits the task to be achieved and the method to be used to achieve this task to another agent, and the host agent uses its local case base along with the received method

[4]In these examples, when we talk of agents we mean computational agents that develop their case bases from the experiences of different employees and are equipped with automated tools for exploiting such experience like case-based reasoning tools. However, the methods presented here would also be applicable for mixed-initiative systems where computational agents and humans are treated as "agents" in an integrated multi-agent system.

to achieve the task for the originating agent. Both Negotiated Retrieval and FPL-based cooperative CBR are appropriate for different kinds of tasks and complement each other as tools for effective knowledge management, to be invoked for problems and tasks for which they are appropriate. As we gain further insights into the mechanisms for distributed case-based reasoning systems through our future research agenda, we hope to translate these into tools for knowledge management in the corporate memory context.

References

Aamodt, A., and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7(1):39–59. <http://www.iiia.csic.es/People/enric/AICom_ToC.html>.

Aït-Kaci, H., and Podelski, A. (1993). Towards a meaning of LIFE. *J. Logic Programming* 16:195–234.

Arcos, J. L., and Plaza, E. (1994). Integration of learning into a knowledge modelling framework. In Steels, L.; Schreiber, G.; and de Velde, W. V., eds., *A Future for Knowledge Acquisition*, number 867 in Lecture Notes in Artificial Intelligence. Springer-Verlag. 355–373.

Arcos, J. L., and Plaza, E. (1996). Inference and reflection in the object-centered representation language Noos. *Journal of Future Generation Computer Systems.* To appear.

Arens, Y.; Chee, C. Y.; Hsu, C.; and Knoblock, C. A. (1993). Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems* 2:127–158.

Bowman, M. C.; Danzig, P. B.; Manber, U.; and Schwartz, M. F. (1994). Scalable internet resource discovery: Research problems and approaches. *Communications of the ACM.*

Genesereth, M. R.; ; and Ketchpel, S. P. (1994). Software agents. *Communications of the ACM* 37(7).

Huynh, M.; Popkin, L.; and Stecker, M. (1994). Constructing a corporate memory infrastructure from internet discovery technologies. White paper, Marble Associates, Inc. <http://www.marble.com/cgi-bin/list-whitepapers>.

Lander, S. E. (1994). *Distributed Search in Heterogeneous and Reusable Multi-Agent Systems.* Ph.D. Dissertation, Dept. of Computer Sceince, University of Massachusetts, Amherst.

Lehnert, W.; Cardie, C.; Fisher, D.; McCarthy, J.; Riloff, E.; and Soderland, S. (1992). University of massachusetts: Description of the circus system as used for muc-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, 282–288.

Lesser, V. R. (1990). An Overview of DAI: Distributed AI as Distributed Search. *Journal of the Japanese Society for Artificial Intelligence* 5(4):392–400.

Lesser, V. R. (1991). A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6):1347–1362.

Nagendra Prasad, M. V.; Lesser, V. R.; and Lander, S. E. (1995). On retrieval and reasoning in distributed case bases. In *1995 IEEE International Conference on Systems Man and Cybernetics.*

Nagendra Prasad, M. V.; Lesser, V. R.; and Lander, S. E. (1996). Retrieval and reasoning in distributed case bases. *Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries* 7(1):74–87.

Oates, T.; Nagendra Prasad, M. V.; and Lesser, V. R. (1994). Cooperative Information Gathering: A Distributed Problem Solving Approach. Computer Science Technical Report 94–66, University of Massachusetts.

Partnet: The Distributed Component Information System. <http://part.net>.

Pasahow, E. (1996). Insider's viewpoint. *Computer Industry Daily.*

Plaza, E.; Arcos, J. L.; and Martín, F. (1996). Cooperation Modes among Case-based Reasoning Agents. In *Proc. ECAI'96 Workshop on Learning in Distributed Artificial Intelligence Systems.* <http://www.iiia.csic.es/Projects/FedLearn/CoopCBR.html>.

Sharp, C., and Lewis, N. Information Systems and Corporate Memory: Design for Staff Turn-over. *Australian Journal of Information systems* 1(1).

Steels, L. (1990). Components of expertise. *AI Magazine* 11(2):28–49.

Vanwelkenhuysen, J. (1996). Corporate Memory. <http://www.inria.fr/acacia/personnel/jvanwelk/projects/cm-project.html>.

Weibel, S. (1995). Metadata: The Foundations of Resource Description. *D-lib Magazine: The Magazine of the Digital Library Forum.* <http://www.dlib.org/dlib/July95/07weibel.html>.

Wielinga, B.; van de Velde, W.; Schreiber, G.; and Akkermans, H. (1993). Towards a unification of knowledge modelling approaches. In David, J. M.; Krivine, J. P.; and Simmons, R., eds., *Second generation Expert Systems.* Springer Verlag. 299–335.