Autonomous Agents in Inter-Organization Project Management

Jeremy Pitt* Matthew Anderton† Jim Cunningham
Department of Computing, Imperial College of Science, Technology, and Medicine,
180, Queen's Gate, London, SW7 2BZ, UK. Email {jvp,mta,rjc}@doc.ic.ac .uk

Abstract

The CEC Project GOAL (Esprit 6283) aims to develop generic software tools to support a new project management paradigm, in which projects are collaborative, decentralised and inter-organizational. To support inter-organizational interaction and communication in such projects, we are developing a framework for formalizing the flow of information between organizations, specifying access to and provision of project services, and structuring inter-organizational transactions. This framework is based on normalizing interactions between autonomous software agents by specifying messages and protocols for communication and cooperation with and between agents. This extended abstract summarizes the framework: its technological background, specification in the framework, and two prototype implementations used in case studies.

1 Introduction

The CEC GOAL Project (Esprit 6283) aims to develop generic software tools to support a new project management paradigm: "the extended enterprise". Some distinguishing characteristics of projects in the new paradigm are that they are collaborative, decentralised and inter-organizational. This means that while there are general issues in project management (such as risk and uncertainty) which take on an extra dimension, there are many new problems to be addressed, such as information access, dissemination, and security, the coordination of independent partners, and communication between partners at remote sites.

We are particularly concerned with the inter-organizational aspects of communication and interaction which enable participating organizations to complete shared tasks and to achieve shared goals – i.e. computer-supported cooperative work (CSCW) for inter-organizational projects. The GOAL Cooperation Services Framework (henceforth CSF) is proposed to enable the specification of project-wide standards for normalized interactions with and between autonomous software agents acting in a federated information system. The CSF provides mechanisms to:

i normalize inter-organizational communication and the flow of information between organizations;

ii enable each organization to provide/use project services required/offered by other organizations;

iii structure inter-organizational interactions with respect to contractual relationships and local working practices.

In the following sections of this extended abstract, we summarize the technological background of the CSF, and briefly review two prototype applications which illustrate the ideas.

*Supported by CEC Esprit Project GOAL (Esprit 6283) and CEC Esprit BRA Medlar II (Esprit 6471).
†Supported by CEC Esprit Project GOAL (Esprit 6283).

2 Technological Background

The GOAL Project

Broadly speaking, a GOAL system is a customisable kernel which provides a data repository, common interface and CSCW services. The data repository is an abstract representation of the project (people, tasks, timescales, etc.). The common interface enables users to access other software tools (word processors, spreadsheets, etc.) and to transfer information between such tools and the data repository. The aim of the CSF is to support advanced CSCW services, i.e. agreed working practices for performing shared project tasks and accessing distributed project information, whilst respecting the local working practices and various security policies of individual organizations. Each partner in a collaborative project operates a GOAL system. Autonomous software agents defined in the CSF framework and operating as part of a GOAL system enable project management procedures for inter-organizational projects to be normalized and executed.

Cooperation Protocols

The mechanisms by which project management procedures are executed is through autonomous agents interacting according to certain pre-defined protocols (cf. [Burmeister *et al.*, 1993]). In the CSF, an *agent* is a GOAL system application which enables two or more users and/or other GOAL applications to participate in inter-organizational tasks and activities. An agent specification defines both the communicative behaviour expected between two agents and the decision making processes that determine that behaviour. Agents communicate by *message passing*: message passing is specified as an action and a set of message specifications (cf. [Winograd, 1988]), which determine for each type of message the space of possible responses. A *cooperation protocol* is then the definition of a structured exchange of messages and constraints on those exchanges, which constitutes an agreed procedure for performing some inter-organizational task or activity.

Two-tiered Specification

We propose a two-tiered approach to the design and specification of cooperation services: one level for specifying agent interactions via protocols and message passing, and the other level for specifying internal processes which determine how an agent behaves in a protocol. The specification of a protocol at the first level is agent, application and domain independent, and protocols are then configured at the second level for particular applications and domains, by specifying the agents' decision-making processes and message parameters (cf. [Kaplan *et al.*, 1991]).

At the first level, we are exploring the possibilities offered by deontic action logic [Maibaum, 1993], which is a modal logic whose modal operators are deontic modalities $\mathcal{O}$ (ought to) and $\mathcal{P}$ (permitted to), and parameterised agent-action modalities, where if A is an agent and a is an action, then $[A, a]$ is a modality. Deontic action logic can be used for automatic generation of a standard 'network-style' specification from the message and protocol specifications, which can be extended and emended by interactive editors. It can also be animated for validation purposes, e.g. evaluation of the dynamic behaviour specified by the protocols (see [Costa *et al.*, 1990]), and can be used for reasoning about time, causality, and non-normative behaviour (i.e. what happens when something goes wrong, and what to do about it).

At the second level, we are experimenting with the specification language of [de Greef *et al.*, 1993], which includes constructs for describing 'plan-schemas' and message passing. This allows us to specify conversations between two or more agents, to blur the distinction between 'client' and 'server' agents, and to be more specific about the data being passed between agents.

Agent Prototyping

IC-Prolog II is a new implementation of Prolog that was designed to address distributed applications, and is also well-suited to multi-agent programming [Chu, 1992].

We have adopted IC-Prolog II as a demonstrator platform for a number of reasons. Firstly, IC-Prolog II is *multi-threaded*, where each thread is a distinct Prolog process. This mechanism allows independent programs and goals to be explicitly forked and to run *concurrently* and in pseudo-parallel. Secondly, the language also provides *high-level communication primitives*, so that threads can communicate efficiently by sending and receiving messages via mailboxes. Thirdly, it has an *object oriented extension*. This supports typical object-oriented notions such as inheritance, self/super and methods, and is a convenient and natural way to structure agents.

Thus the execution of an IC-Prolog II program will typically give rise to a set of explicitly forked sequential processes, such that these processes run concurrently and can communicate using asynchronous message passing. If a mailbox server is running, then a set of programs can cooperate and communicate by registering with the server. This allows programs (and their forked processes) which are executing concurrently and on separate workstations to communicate over a network. This makes IC-Prolog II particularly apposite for prototyping the interacting autonomous agents envisaged by the CSF.

3 Two Case Studies

Cooperative Answering

As a preliminary investigation into providing distributed cooperative services, we developed a system for an agent to provide cooperative answers to remote queries submitted to a local data repository, i.e. a trivial *inquire* protocol.

The motivation for using cooperative answering technology is to avoid the need for remote users to know the structure and content of a database in order to ask a query, and to retrieve all the facts relevant to a query or better still more information than just atomic facts. This enables a partner in an inter-organizational project to provide project information services. The agent responds to natural language queries, so that users can make queries using familiar names and concepts, and are not burdened with the need to learn a formal query language.

The prototype agent consists of an email server, configured to send and receive X400 standard email under UNIX; a natural language processor which converts English text into a logical query; and a cooperative responder which implements a number of techniques (topics of interest, plans and goals, and intensional and conditional answers) to try to find the most helpful answer. A language synthesizer (not yet fully integrated with the whole system) converts the logical answers back into natural language. The processing constraints of the present technology are compatible with turn-around times expected when using email with human agents. Full details of the system can be found in [Pitt *et al.*, 1994b].

Distributed Document Review

The scenario addressed in this study was that of quality control in a stereotypical Esprit Basic Research Action where the deliverables are largely technical papers. The Project Office aims to ensure the quality of these documents by having each paper reviewed by (say) three referees.

The procedure for allocating papers to referees begins with an announcement from the Project Office to potential reviewers of a particular deliverable with basic task information such as document identification (title, authors, etc.), a list of keywords and a proposed resource allocation of (say) five person days. Each reviewer accepts or rejects the task. If sufficient reviewers accept the initial terms then three are chosen, otherwise the Project Office selects what it thinks are appropriate reviewers, and engages in one-on-one negotiation with each in turn until either enough reviewers have been found or the set of potential reviewers has been exhausted. In the latter case an alert is raised.

The implementation consists of a single 'client' agent, the Chair, and multiple 'server' agents, the Reviewers. Each agent is constructed from a number of modules which correspond to elements of the CSF and the application domain. The generic components implement the basic client and server agents, an agreed message specification, *negotiate* and *announce* cooperation protocols, and a conversation handling mechanism. Application specific elements are added as knowledge bases for Chair and Reviewer agents, including domain knowledge, organizational knowledge, and knowledge of each individual agent. For full details see [Pitt *et al.*, 1994a].

4 Conclusions

We have given an overview of the GOAL Cooperation Services Framework, a framework for specifying agents and interactions with and between agents via protocols, which can support CSCW in inter-organizational project management. We have briefly described: a two-tiered approach to the specification of agents and protocols, the programming language IC Prolog II being used as a demonstrator platform, and two case studies in inter-organizational communication and interaction. While the implementations are relatively simple prototypes, we believe that they are a successful proof of concept and show how agent technology and distributed AI can be used in inter-organizational project management.

The challenges ahead range from short-term development to long-term research: we need to make the CSF operational in the GOAL system, to demonstrate genericity and configurability of the protocols by re-use in other management procedures (e.g. change management, meeting scheduling, etc.), and to investigate autonomous agents which are secure, self-configurable, and pro-active (cf. the July issue (vol. 37, no. 7) of the *Communications of the ACM* on "Intelligent Agents").

References

[Burmeister *et al.*, 1993] B. Burmeister, A. Haddadi, and K. Sundermeyer. Generic configurable cooperation protocols for multi-agent systems. In *MAAMAW'93, Neuchatel*, 1993.

[Chu, 1992] D. Chu. IC Prolog II: A Language for Implementing Multi-Agent Systems. In *Proceeding Special Interest Group on Cooperating Knowledge Based Systems, Keele*, 1992.

[Costa *et al.*, 1990] M. C. Costa, R. J. Cunningham, and J. Booth. Logical Animation. In *Proc. of 12th International Conference on Software Engineering*, pages 144–149, Nice, 1990. IEEE Computer Society Press.

[de Greef *et al.*, 1993] P. de Greef, K. Clark, and F. McCabe. Toward a specification language for cooperation methods. In *GWAI'92 16th German AI Conference*, volume 671, pages 313–320. Springer Verlag, 1993.

[Kaplan *et al.*, 1991] S. Kaplan, A. Carroll, and K. MacGregor. Supporting Collaborative Processes with ConversationBuilder. ACM, 1991.

[Maibaum, 1993] T. Maibaum. Temporal Reasoning over Deontic Specifications. In J. Ch. Meyer and J. Wieringa, editors, *Deontic Logic in Computer Science*, pages 141–202. Wiley, 1993.

[Pitt *et al.*, 1994a] J. Pitt, M. Anderton, and J. Cunningham. Normalized interactions between autonomous agents. m/s, Imperial College of Science, Technology and Medicine, (submitted for publication), 1994.

[Pitt *et al.*, 1994b] J. Pitt, J. Cunningham, and J. H. Kim. Cooperative answering to natural language email queries. In F. Anger, R. Rodrigues, and M. Ali, editors, *Proceeding 7th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 273–281. Gordon and Breach Science Publishers, 1994.

[Winograd, 1988] T. Winograd. A language action perspective on the design of cooperative work. In I. Grief, editor, *Computer Supported Cooperative Work: A Book of Readings*. Morgan Kaufman, 1988.