

# Integrated Computational Paradigms for Flexible Client-Server Communication

Jean-Marc Andreoli and Remo Pareschi  
Rank Xerox Research Centre, Grenoble, France

The basic insight behind client-server computing is separating the aspects of access and of management of computational resources. This simple model offers simplicity in closely matching the flow of data with the control flow and promotes modular, flexible and extensible system designs. It is, however, limited by its focus on clients requesting individual services. In fact, clients may need to invoke multiple services, coordinated to reflect how those services interrelate and contribute to the overall application. Important examples include task allocation and event notification in collaborative workgroup systems [4] and task sequencing and routing in workflow applications [6]. As a consequence, there have been several proposals for extending the basic client-server model in order to provide flexible support of client interactions with multiple servers. (See for instance [1] for a discussion and a classification of possible extensions.)

In the Coordination Technologies Group at the Rank Xerox Research Centre we have addressed this problem by identifying the computational paradigms that can best provide client-server computing with the required flexibility to meet the demands of workflow and workgroup applications. As a result, we have designed and implemented the Coordination Language Facility (CLF) [3, 2], a middleware tool that extends the basic one-to-one client-server model of communication with capabilities for many-to-many interactions between clients and servers. The CLF achieves this flexibility by combining two different computational paradigms:

- *Constraint Programming*, to provide flexibility in the issuing of requests and in the selection of answers on the client side.
- *Object Oriented Programming*, to provide flexibility in the management of requests on the server side.

In this position statement we give a characterization of the CLF from the point of view of the requirements of workflow applications.

Work processes, central to workflow management applications, describe how resources may be passed between the components of a software system to complete a package of work. For example, a workflow management system might wait for a set of files to arrive and then launch a text editor to merge those files into a report that, in turn, is stored with a document manager. The workflow management system is not concerned with the implementation of the editor or the document manager but with how their public interfaces communicate and pass data.

There is a lot of interest in workflow software, rooted both in business and in technology. In fact, business executives have realized that organizations are best viewed and analyzed in terms of their processes, and not just in terms of simple business activities [5]. At the same time, the coming age of networked enterprises provides the natural infrastructure to automate work processes by coordinating multiple distributed components. Software constructs that give full flexibility to the development of workflow management systems are still needed, however, as are complementary applications that provide direct support for the notion of process. The CLF is a process-oriented extension of client-server computing intended to fill this gap.

The CLF integrates two computational constructs: constraints and objects. This integration allows fine-tuning of the communication mechanism of client-server architectures, which in its basic form is implemented through simple procedure calls, in a way suitable to support work processes involving complex negotiations between multiple participants. From OOP, the CLF exploits the encapsulation of data accessed through well-defined interfaces, which gives direct support to client-server computing by imposing a strict separation between invocation and implementation of methods. From constraint programming, the CLF exploits non-determinism and incremental refinement of variable bindings: a variable can be (partially) instantiated by any token satisfying the constraints put on the variable itself; multiple bindings are resolved by accumulation of constraints. Indeed, a variable acts as a communication channel through which a negotiation is established between different actors which must agree on the

value of the variable. The CLF makes use of such a feature to implement negotiations across active objects, where several alternative potential agreements can be explored in parallel during the negotiation before a commitment is taken. Thus, under this scheme, server objects can provide alternative answers to requests invoked by client objects which can then select some or all of the proposed answers. This capability is particularly useful whenever a given request can be satisfied in a number of different ways: eg, in workflow management users can choose their own way to satisfy requests that correspond to tasks graphically displayed in their desktop environment.

The CLF implements this behavior via the interaction of entities of two kinds: coordinators and participants. In its basic form, a coordinator defines a minimal process unit for a course of action to be negotiated among, and enacted by, multiple participants. Each participant is an autonomous component responsible for its own part of the plan. The coordinator requests performance of actions to participants, and ensures that there is no conflict between the ways of enacting the actions chosen by the different participants. Once everything is agreed upon, the overall plan can be executed. Reaching this stage may involve several rounds of negotiation. For instance, two different flight databases may be requested to produce, respectively, a flight that departs from an airport in Northern Italy and reaches an airport on the US East Coast, and a flight that departs from an airport on the US East Coast and reaches an airport on the US West Coast; they are also requested to agree about East Coast arrival airport and East Coast departure airport to be the same. There is a clear relationship between agreement of this kind and the notion of consistency provided by transaction processing systems, and indeed the CLF negotiation protocol between coordinator and participants incorporates a standard transactional two-phase commit protocol. There are however some relevant differences too, mainly the fact that participants can choose which answer to return to a given request, and that a coordinator may reject answers that are conflicting and inappropriate. Thus, rather than superimposing transactional consistency over passive objects, coordinators perform *long-lived inquiries* leading to final agreement among autonomous components.

Participants correspond to active objects and can be accessed through a variety of communication protocols: the current implementation of the CLF supports CORBA (the communication protocol for distributed object infrastructures designed by the Object Management Group) and HTTP (the WWW communication protocol). Coordinators are currently implemented as production rules; other kinds of notations are however possible, eg procedural or graphical. A crucial feature of the semantics of coordinators is their compositionality, that allows both modeling of complex coordination behavior and achieving de-centralized and dynamically reconfigurable coordination. Thus, coordinators can be composed, either to chain given process units into larger units, or to merge different negotiation policies over the same participants; conversely, complex coordinators can be split into smaller coordinators that are distributed over different physical locations; moreover, coordinators can be dynamically added and removed during execution.

## References

- [1] R.M. Adler. Distributed coordination models for client/server computing. *IEEE Computer*, 28(4), 1995.
- [2] J-M. Andreoli, S. Freeman, and R. Pareschi. The coordination language facility: Coordination of distributed objects. *Theory and Practice of Object Systems*, 2(2), 1996.
- [3] J-M. Andreoli, H. Gallaire, and R. Pareschi. Rule-based object coordination. In *Proc. of ECOOP'94 Workshop on Coordination*, Bologna, Italy, 1994.
- [4] E. Bender. Workgroup computing. *PC World Magazine*, January 1995.
- [5] T.H. Davenport. *Process Innovation*. Harvard Business School Press, Boston, Ma, U.S.A., 1993.
- [6] L. Fischer and T. White. New tools for new times: The workflow paradigm. Technical report, Future Strategies Inc., Alameda, Ca, U.S.A., 1994.