

An Internet software platform based on SICStus Prolog (Position Paper)

Joakim Eriksson Fredrik Espinoza Niclas Finne
Fredrik Holmgren Sverker Janson Niklas Kaltea
Olle Olsson

Swedish Institute of Computer Science
SICS, Box 1263, S-164 28 Kista, Sweden
{joakime, espinoza, nfi, fredrikh, sverker, nicke, olleo}@sics.se

March 18, 1997

1 Introduction

During the last year we have been involved in several projects that have utilized the agent metaphor. A vital part of the work has been to identify and implement mechanisms/constructs that enabled us to work at a suitable level of abstraction. As a result several modules representing the basic functionality needed for agent oriented applications have emerged.

Currently we are working on refining and extending the existing modules into an agent platform, working name AgentBase. The platform is being developed in Prolog Objects, an object oriented extension of SICStus Prolog [1].

In our previous work we have focused on resource allocation and electronic markets, but our goal is to support the implementation of Internet aware agents, as well as simple web servers and proxys, in general.

2 The platform

The platform consists of a library of classes (or prototypes in SICStus Objects terminology). The intention is that the user chooses and combines the appropriate functionality given by the library according to the demands of the application. See Figure 1 for an overview of the architecture and an example of a web server. Below follows a technical description of some of the classes that currently realize the platform.

- **Multiple socket I/O Internet server class**

The Internet server class is the kernel when building Internet server applications. It listens to a number of sockets each associated with a specific object. When a connection to a socket is made, the object associated with that socket will be informed about the connection and the resulting stream.

- **HTTP generation and parsing libraries**

The HTTP libraries are for parsing and generating HTTP messages into and from an internal format. There is one library for parsing a message into the internal format and then there are libraries for accessing/parsing the different types of HTTP header fields.

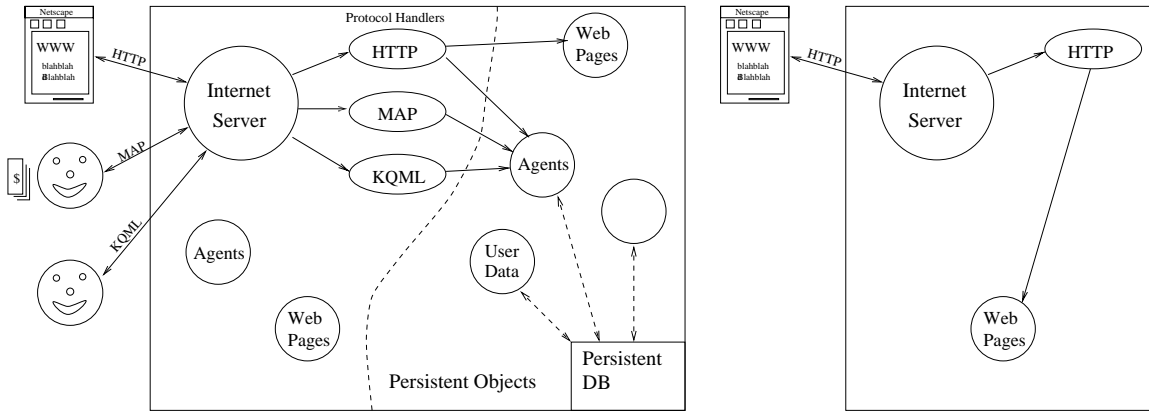


Figure 1: Example of a full agent system (left) and a web server (right) using AgentBase

- **HTTP client and server classes**

The HTTP client class gives objects inheriting from it the ability to send HTTP messages to servers. This is mostly used for fetching web pages and is useful when implementing web-proxys. The HTTP server uses the Internet server for handling incoming connections and when accessed it sends the message to the object that has the same name as the first part of the path. For example if someone accesses the page “http://mungo.sics.se:10000/tools/unix” it will be the object named tools that receives the message.

- **HTML generation and parsing libraries**

The HTML libraries are for generating HTML code from prolog terms. The generation library is designed to be as close to HTML as possible so that users do not need to learn a new language for describing web pages, but rather reuse their HTML knowledge.

- **HTML pages design classes**

The HTML page design classes are made to make it simpler to generate nice looking web pages. An object inheriting from one of these classes only have to define the real content of the page and a few attributes that the design class uses to generate the page. One example is a menu design where the design class defines a menu to the left of the page and lets the inheriting objects define which links the menu should contain by setting the attribute “links” to a list of links.

- **MAP (Market Agent Protocol) handler**

The MAP handler works like the HTTP server in the sense that it sends the incoming messages to objects with the same name as the receiver in the message. The MAP protocol is an agent protocol somewhat inspired by speech-acts that is based on negotiation within markets. It contains the communicative acts: *tell, ask, negotiate, offer, accept, refuse*

- **Persistent object class**

This mix-in class implements persistent attributes for objects. When a persistent object (or rather an instance) is not loaded into the system, and a method in it is called, the object will be created and its attributes will be loaded from the database. The attributes is stored in a database file using the “external database” library for SICStus prolog.

The platform is currently being expanded and will include classes associated with other conceptual levels and use. What follows is either under development or in the planning stages.

- **Security classes**

Currently there is a class for authenticating HTTP accesses using Netscape Cookies but other classes for digital signatures for authenticity and cryptography for confidentiality are planned.

- **KQML**

Previously we have gained some experiences of agent communication languages along the lines of KQML (Knowledge Query and Manipulation Language) [2].

Currently we have not implemented the full set of performatives as proposed in [3] but we are moving in that direction.

The basis of the current implementation can be regarded as a toolkit were you assemble your own performatives. This makes it possible to tailor an agent communication language in the style of KQML if needed. Each performative automatically belongs to some specific category depending on its meaning and the type of content it may carry. The core of the toolkit consists of the underlying functionality to realize not only basic categories for inter-agent dialogue but also support related to more complicated agent federations.

Besides the above classes that are under development, work are also being done on a multithreaded version of SICStus Prolog. This will make the platform even more interesting for Internet aware agent applications.

3 Application projects

AgentBase will be used within a number of different research projects at SICS.

MarketSpace. This project aims to automate parts of the market interaction using agents. The interaction protocol is based on market negotiation and the information used in the agent communication is interests describing sets of deals. Agents will be used in the roles of: *personal assistant, broker, auctioneer*, etc.

Adaptive and intelligent workflow systems We are currently working on an intelligent system for adaptive workflows, where we use agents to model actors and activities in a workflow system. A workflow consists of one or more agents that interact, either in a cooperative or a competitive manner, in order to perform a process. Agents advertise services they can perform and services they need through different facilitators. As soon as an agent need a services it can check with the facilitator for agents that can perform the requested service and negotiate with them for the best possible offer. What constitutes a best offer varies, but it can be based on price, speed, security, etc. With the introduction of negotiating agents we can dynamically optimize the process.

Resource monitoring and allocation. Previous work in these areas has (from an agent platform point of view) focused on the agent communication and architecture levels. Future work will concentrate on related issues in conversation and coordination.

References

- [1] Swedish Institute of Computer Science. *SICStus Prolog User's Manual*, 1995. Also available via <http://www.sics.se/isl/sicstus.html>.
- [2] Tim Finin and Richard Fritzson. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management*. ACM Press, 1994.
- [3] Yannis Labrou and Tim Finin. A proposal for a new KQML specification. Technical report, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, February 1997.