Cooperation Services in a Structural Computing Environment

Samir Tata, David L. Hicks, Uffe K. Wiil

▶ **To cite this version:**

Samir Tata, David L. Hicks, Uffe K. Wiil. Cooperation Services in a Structural Computing Environment. 3rd International Workshop on Structural Computing - SC3'2001, 2001, Arhus, Danemark, 5 p. inria-00099400

HAL Id: inria-00099400
https://inria.hal.science/inria-00099400

Submitted on 26 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cooperation Services in a Structural Computing Environment

Department of Computer Science and Engineering
Aalborg University Esbjerg
Niels Bohrs Vej 8
6700 Esbjerg, Denmark
http://www.cs.aue.auc.dk

Abstract. This

1 Introduction

2 Architecture

3 Services design

3.1 Naming service

The Construct naming service allows Construct services to locate other construct components by a user definable name. The Construct service allows other Construct services to associate an object (i.e. a user, a group, a structural service, a foundation service) with a name. Once the name is associated with the object, any Construct service or application can access the object through the name.

The naming service is both very simple and very useful. It is simple because it just map Construct object references and abstract names. It allows in addition the service provision in Construct. News services can be registered with the naming service and so can be used by Construct services and applications.

The IDL specification of this service is given below.

```
interface Naming
{
  void register(in Namingable object);
  void remove(in string name);
  Namingable lookup(in string name);
};

interface Namingable
{
  string getName();
};
```

3.2 Session management service

A session is a context allowing cooperative users to start, stop, join, leave, and browse collaborative situations across the network[6]. Within a session, users can connect from various locations to the structural computing environment in order to work together on shared artifacts such as white board and use multimedia conferencing tools to communicate. The context of a text document editing within a shared editor and a video conferencing among a number of participants are examples of sessions.

In order to create sessions and allow users to works within sessions, the structural computing environment need a service to manage sessions. A session management service determines the way in which the users of a cooperative application joint together into a collaborative endeavor[2]. The major functions of this service are the session information management and the command management. In the following, we describe broadly these functions and we give their IDL specifications. We describe, after that, the interaction of the session management service with other service in the structural computing environment.

Session information management The *session information management* function creates sessions and updates information within active sessions. A simplified IDL specification of this function is given below. Among others, the session information management manage the list of online users working in a given session : adds/removes users to/from a given session...

```
interface SessionInformationManager
{
  attribute string name;
  attribute Vector onlineUsers;
  string getName();
  void setName(in string newname);
  void addOnlineUser(in string name);
  void removeOnlineUser(in string name);
  Vector getOnlineUsers(in string group);
  Vector getAllOnlineUsers();
  void close();
};
```

Command management The *command management* function opens and closes sessions. It allows users to log in and log out and provides to the logged users facilities to execute commands on shared data using any structural service in the structural computing environment (i.g. navigational service, meta-data service...). The IDL specification of this function is given below.

```
interface CommandManager
{
  string create();
```

```
    void close(in string name);
    void login(in string name, in string password,
               in string group, in string session);
    void logout(in string name, in string session);
    void handle(in Command cmd, in string session);
};
```

Interacting services The session management service depends on some services within the structural computing environment.

foundation service Like almost Construct services, the session management service depends on the store service. In fact, the store service is used to save objects representing session information (session names, online users...) and Construct services, which are used to execute command users. In addition, the session service may use the store service to keep user preferences, personal data, application defaults...

In order to control session accesses, the session management service use the access control service. In fact, it controls whenever a user has the permission to create or join sessions, to handle commands or to get the online users in a given session.

Naming service The session management service depends on the naming service. This later is used to select users and groups and to select and to locate Construct services (that the session service may use to execute user commands).

Other Construct services to allow users to work together on shared objects using several Construct services, the session management service interact with all structural services. It sends commands that will be handled by these services.

3.3 Awareness management service

In cooperative systems, users can be distributed in space, in time, and even over organizations. To interact effectively they need means that allow them to be aware of others: who does what? Coworkers need to be aware of the current status and the activity of others. For them, the awareness acts as an "integrating framework" by allowing them to talk to each other, discuss, show results and update them.

The need of supporting awareness in cooperative applications has been identified in several works. Some of applications that support awareness include Quilt [5] and GROVE [4] that are summarized in [1].

The term *awareness* has a common usage to indicate mindfulness, or being conscious of surroundings, situations, or other people. Since the literature has not set a technical definition of the term *awareness*, we can, like in [3], present a definition based on separating the concept into two components, which have been investigated in the literature:

– presence and location of coworkers
– previous and current tasks

To detect the presence and the location of coworkers and determine the previous and the current tasks, we propose here a construct service based on the naming, the session, and the notification Construct services. In instance, using the naming service and the operation `getOnlineUsers` of the session service the Construct awareness service can locate the online users. In addition using the notifications Construct service, it can get the posted messages reflecting the activity of coworkers.

The IDL specification of this service is given below.

```
interface awareness
{
  const int META_DATA_MASK = 2;
  const int DATA_MINING_MASK = 4;
  const int STORE_MASK = 8;
  ...
  const string META_DATA_PREFIX = "META_DATA";
  const string DATA_MINING_PREFIX = "DATA_MINING";
  const string STORE_PREFIX = "STORE";
  ...
  attribute java::util::Vector members;
  attribute java::util::Vector memberLevels;
  int getMemberLevel(in string member);
  void setMemberLevel(in string member, in int level);
  void postEvent(in string initiatorMember, in int prefix,
                   in string message, in int memberlevel);
  Vector getLoggedEventsPrefix(in int prefix);
  Vector getLoggedEventsMember(in string initiatormember);
  Vector getLoggedEventsLevel(in int level);
};
```

4 Future work

5 Conclusion

References

1. P. Dourish and V. Bellotti. Awarness and coordination in shared work space. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 51–58, Toronto, Canada, November 1992. ACM Press.
2. W. K. Edwards. Session management in collaborative applications. In *ACM CSCW'94*, pages 323–330, 1994.
3. W. K. Edwards. *Coordination Infrastructure in Collaborative Systems.* PhD thesis, Georgia Institute of Technology, College of Computing, Atlanta, GA., December 1995.

4. C.A. Ellis, S. Gibbs, and G. Rein. Groupware: some issues and Experiences. In *Communications of the ACM, 34 (1)*, pages 38–58, 1994.
5. Robert S. Fish, Robert E. Kraut, and Mary D. P. Leland. Quilt: A collaborative tool for cooperative writing. In *Proceedings of the Conference on Office Automation Systems*, Collaborative Work, pages 30–37, 1988.
6. R. D. Hill, T. Brink, F. Patterson, S. L. Rohall, and W. T. Winter. The rendezvous language and architecture. *Communication of the ACM*, 36(1):62–67, January 1993.