Common Objects in Place-based Collaborative Environments

A Green Paper

Jeff Kurtz, jkurtz@mitre.org
The MITRE Corporation

Introduction

Place-based collaborative environments (PBCEs) facilitate group communication and information sharing between people who are not collocated. This type of environment usually presents a virtual place, such as a building or rooms, where people can meet, conduct conversations and share documents among other things.

Imagine a chat room application coupled with a document repository. Users enter rooms to chat with other people and they exchange documents via the repository. The environment consists of rooms, people, a text chat tool and documents. Most PBCEs maintain these types of elements.

This paper characterizes the common objects of PBCEs and puts forth a sample data model. There are four primary objects. *Context* is the setting for collaborative activities, which is usually a collection of participants, conversations and objects. *Participant* represents a user (human or software agent) of the environment. *Conference* is a shared, multi-participant session used by participants to communicate with each other or used by applications to share data. *Folder item* is a generic item that can be a container or leaf object such as a document.

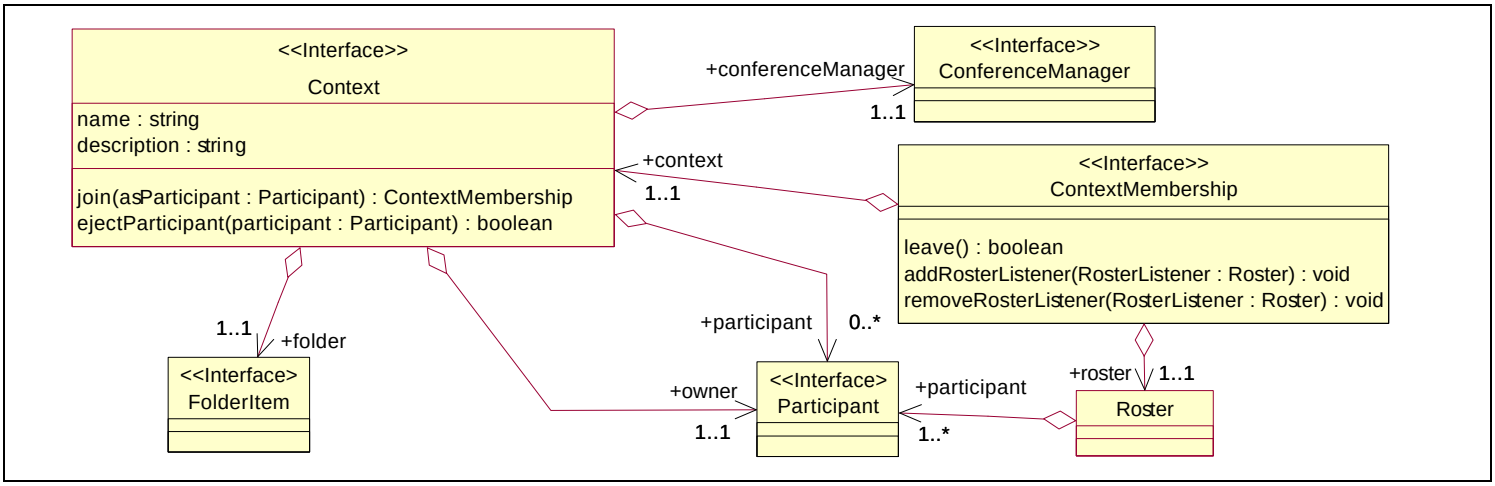
Context

Contexts are abstract, metaphor-neutral collections of participants, conversations, and documents. Contexts factor the total collaborative activity of the system into useful, comprehensible parts. This concept is a generalization of place-based systems that factor collaboration based on space. Contexts are gathering places for collaborations and can be bound to many different representations, metaphors, and purposes; for example, plan objectives, functions, units, common skills, geographic regions of interest, and so on. For example, a software development company might have contexts for design, development, testing, marketing or purchasing. In the testing context you could find people responsible for testing and test plan documentation.

Contexts are persistent—that is, the context and the documents, tools, and so on gathered there are continuously available—they do not go away when any or all of their users disconnect from the system.

Figure 1 shows the UML model for the context object. Joining a context returns a *ContextMembership*. The membership is used to track participation in the context and to leave the context. *Roster* is the set of participants joined to the context. *ConferenceManager* manages the set of conferences available when joined to the context.

Figure 1: Context UML



One participant is identified as the owner of the context. The owner could be the person who created it or it could be assigned to another participant.

The context's participants are all the users who have joined the context. Joined participants are reachable but not necessarily active in the context

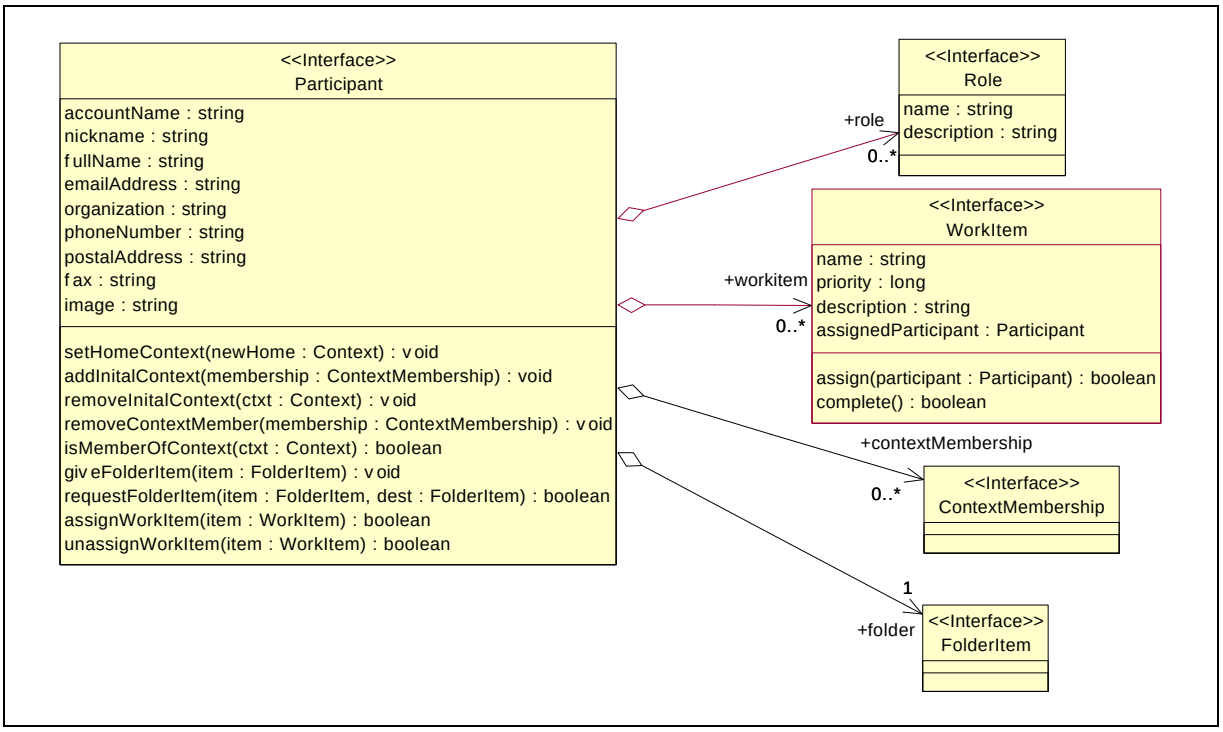
Participant

Participants represent users (human or software agents) of the environment. They track the contexts a user is participating in and the conferences being used in those contexts. Each participant may have one or more roles—much as in the workplace, where a single employee may work on more than one project and may even have different kinds of duties and different levels of responsibility associated with each project.

Roles are social cues that can be used to indicate a participant's responsibilities. For example, a participant in the environment may join one context as a committee member who needs to participate in discussions, and another context as the person with authority to sign off on logistic plans.

The UML model in Figure 2 shows the attributes and behavior of the participant object. Setting the home context identifies one context as the participant's default location. Initial contexts are contexts the participant will participate in as soon as they connect to the environment. *FolderItem*s and *WorkItems* can be exchanged via the participant. FolderItems are items such as documents and WorkItems describe a task to be performed by the participant.

Figure 2: Participant UML



When a participant joins a context they communicate to other people using the conferences managed by that context.

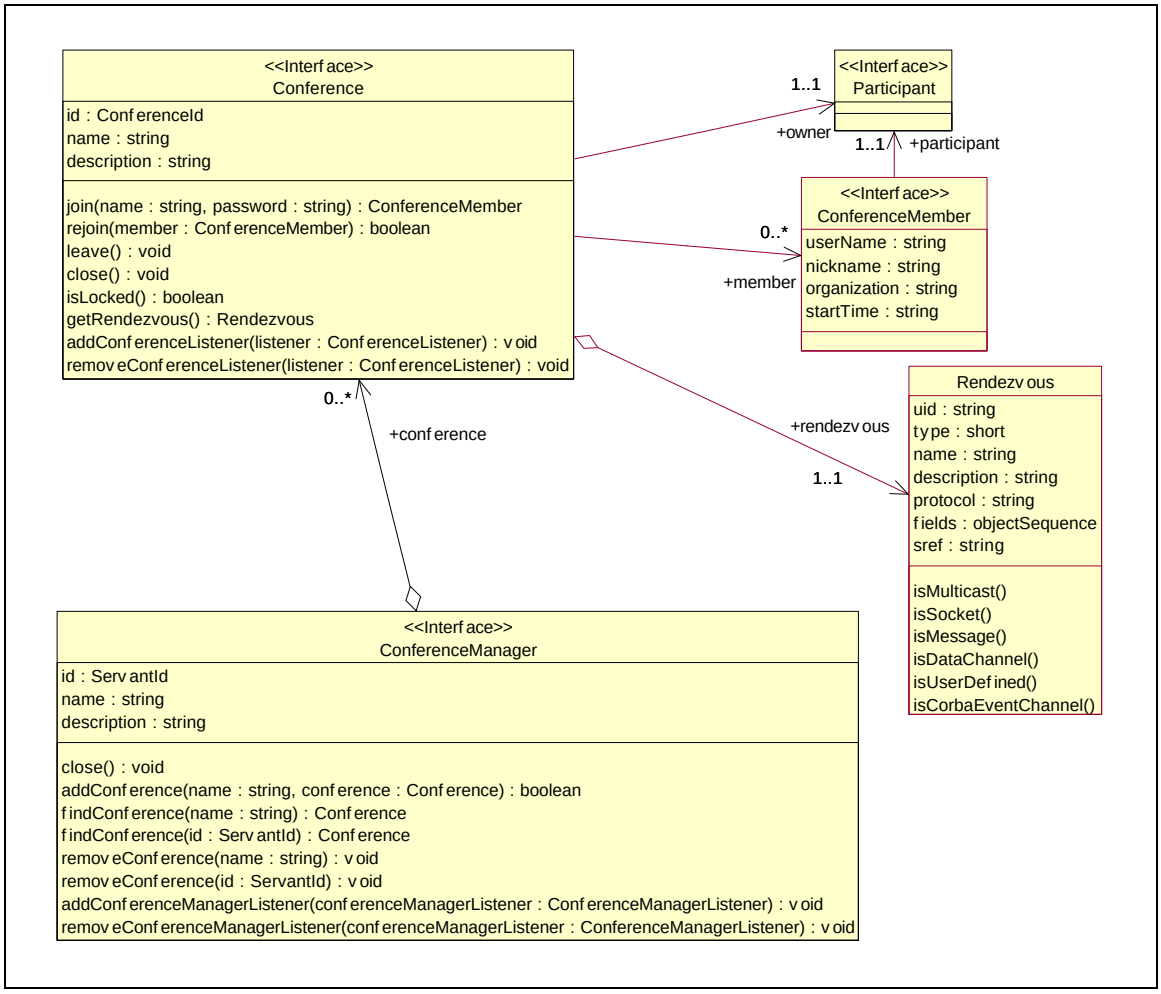
Conference

A conference is a shared, multi-participant session under a single point of control, either unimodal or multimodal. It is used by participants to communicate with each other within the virtual environment, and for applications to share data. Examples of kinds of conferences supported in the toolkit include text conferences, in which users communicate using basic “chat” capabilities; multicast conferences, in which participants can use audio and video media; and conferences that allow participants to share and collaborate on documents, data, and objects. Participants may be active in several conferences at once; for instance, participants collaborating on a logistics plan might be sharing data via one conference, and talking about that data via another.

As shown in Figure 3, each conference has a name, description, owner and list of members. After joining the conference the *rendezvous* is used by client software to connect to the underlying communication channel. The rendezvous describes the type and protocol of the conference.

ConferenceManager is found in contexts (see Figure 1) and manages the conferences for one context.

Figure 3: Conference UML

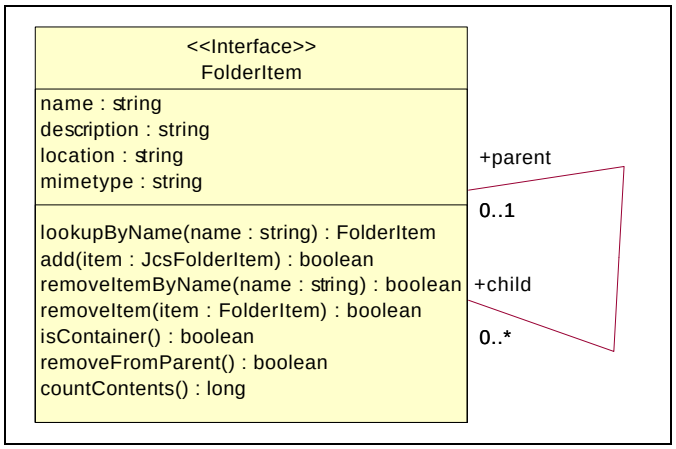


Folder Item

Participants and contexts have folders and documents associated with them. A *folder item* can be a container or a leaf object like a document. Folder items have locations and types. The location is a reference to the associated data in a repository. For example, the *location* could be a URL or a document repository identifier. This allows for distribution of objects across repositories. The *mimetype* field describes the type of the associated data.

The UML in Figure 5 shows that FolderItems can contain other folder items. This allows you to define hierarchies of information. The *isContainer* method returns true if the item is a container and false if is a leaf.

Figure 4: FolderItem UML



Participants can exchange folder items with other participants. They can also place the items inside a context's folder.

Folder items can refer to any type of information. Software clients for the environment can use the mimetype information to identify data handlers for presenting the information to the user.

Building Environments Using these Objects

Taken together these components can be used to build a variety of place-based collaborative environments. Different user experiences can be created using the same underlying components.

A complete collaborative environment will consist of component implementations, factories for instantiating components and directories for distributing and accessing the components. It will also include one or more client software implementations. Client software can be tailored to a group's preferred mechanism for coordinating activities.

Summary

The data model described here is just one abstraction of place-based collaboration. Undoubtedly, there are others. If you have any comments or alternate

approaches please contact Jeff Kurtz (jkurtz@mitre.org, 781-271-2291) or Henry Rothkopf (henryr@mitre.org).

The sample data model described in this paper is available in the Joint Collaboration Services (JCS) toolkit. The software development kit is written in Java and uses CORBA. See http://jcs.mitre.org. The appendix contains more UML models.

Appendix: More UML models

The *Login* component manages the connections participants make into the environment. Once a participant logs in they are able to act in the environment. Figure 5 shows the Login component and the *LoginConnection* object that is returned by the *login* method and used to logout from the environment.

Figure 5: LoginUML

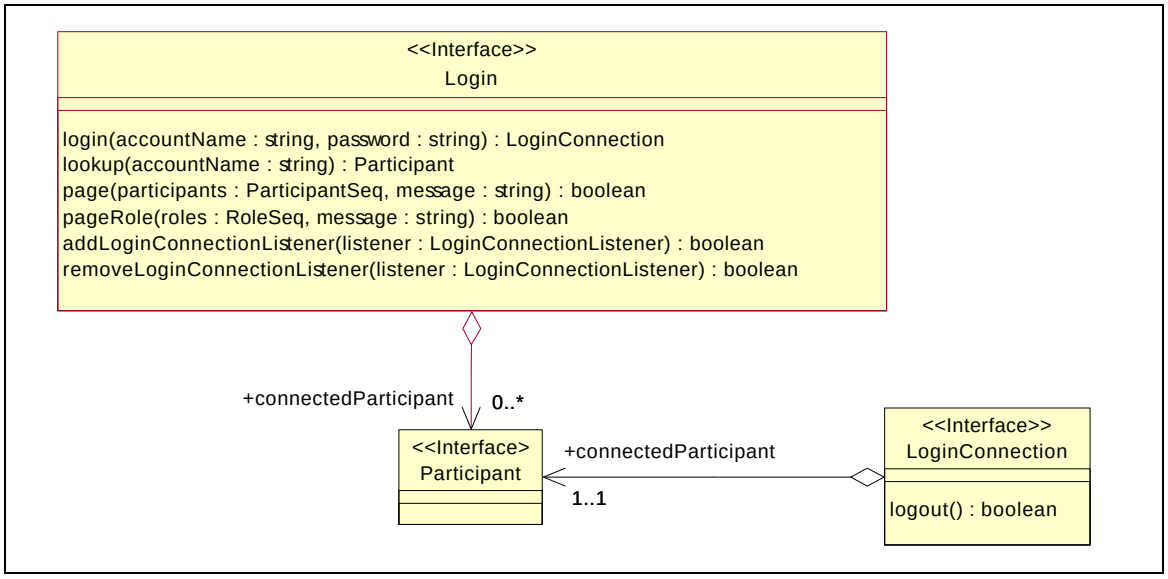


Figure 6 shows all the objects discussed above in one UML model. The operations are not shown to conserve space.

Figure 6: Combined UML model, without operations