INTERACTIVE RESOURCE SHARING ON THE WEB

Simon Schubiger and Oliver Krone*
IIUF – Institute for Informatics, University of Fribourg
Chemin de Musée 3
1700 Fribourg
Switzerland
http://www-iiuf.unifr.ch/~schubige

1 Introduction

The Web, an inherently distributed application, is the ideal research playground to take advantage of concepts developed within the area of a new research discipline, called coordination theory [10]. For example Berlinda [13] based on the Linda [1] coordination model, or Shade [2] based on an agent rewrite technique, successfully coordinate applications on the Web. It is the main goal of these tools to provide a platform which allows the user to benefit from the computational potential offered by the Web. However, most of them fail to provide an appropriate environment for heterogeneous general purpose application sharing, a topic we address in this paper.

The idea for this work is to combine coordination theory with Web technology and use the Web as an interactive meeting point for resource sharing. At interface level, the user of our environment accesses available resources such as files, CPUs, and the like or whole applications via typical operations known from classical desktops. The visibility of resources on the Web can dynamically be extended or reduced, therefore enabling collaboration functionality by changing the accessibility of formerly local resources onto a system-wide, global level.

The rest of this extended abstract is organized as follows: Section 2 sketches the overall software architecture of our approach, Section 3 shows a typical example and Section 4 concludes this paper.

2 Architecture

The Software Architecture of our approach is depicted in Figure 1. There are three components involved: a resource set server as the implementation of a coordination space to manage shared resources, user interfaces for interaction purposes with the Web, and machine-local

*Author's current address: Digital Equipment Corporation, DEC, Postfach, CH-3001 Bern, Switzerland, Oliver.Krone@digital.com
resource servers used to export local resources to the global Web. Note that local user interfaces and local resource server can be used independently. A user benefiting from resources on the Web only typically invokes a user interface only, whereas machines which provide certain resources automatically, that is without user intervention, may only run a local resource server.

Figure 1: An overview of the interactions between a resource set, two resource servers and two user interfaces.

2.1 Resources

By a resource we understand everything that can be manipulated by a given machine. Resources may have some physical representations such as a printer or a file. But also abstract concepts such a CPU-power, that is a time dependent function, are considered as resources. Due to the heterogeneous nature, there exists no common interface to a resource. However, resources can easily be classified. An open set of properties will describe each resource in addition to an open set of actions. These actions may just be operations which can be applied to a certain resource or operations that the resource can perform itself. A printer for example can be described by some properties like the type and vendor of the printer. An action that may be applied to a printer is a status query and one of the operations that a printer can perform is printing a page.

All these features map very well to an object oriented model. In fact, this is the reason why a class hierarchy is used to organize the different kinds of resources. Fields and methods implement properties and actions, respectively. Using a platform independent object framework such as Java allows us to turn the previously heterogenous world of resources into a well organized, homogenous world of objects.

The object framework acts just as a “wrapper” for the native resources. So the full power and features of the native software and hardware are kept. Since the object framework is open and extensible, there are no restrictions on fields and methods, legacy software as well as new technologies can be integrated.

Figure 2 shows a resource and its four parts in more detail:

- A presentation provides an interactive interface to a resource on the user level. A resource has at least two standard presentations: “Iconic” and “Inspector”. Both of them are shown on the screen-shots (3). The “Iconic” presentation is used to manipulate a resource similar to the file and program icons found in today’s desktop interfaces. The
```
RESOURCE

Presentations (User Interface)
- Iconic
- Inspector
- ...

Properties
- read(<property>)
- write(<property>,<value>)
- set_readonly(<property>)
- notification(<property>)

Inputs (Methods) | Outputs (Stubs)
- ...
- ...
```

Figure 2: The resource interface

"Inspector" presentations allows a user to view and change properties of a resource. The right hand side of figure 3 for example shows the "Inspector" presentation of the UNIX unzip tool. Generic implementations are provided for the standard presentations, but they can be extended or replaced by a particular resource. Resources can add additional presentations in order to provide a specialised user interface.

- The properties represent the state of a resource. The resource framework automatically collects all public fields of a class used to implement a resource. These fields are then published as properties and are accessible from the outside through the appropriate read and write calls. In addition, a property can be marked read-only and notifications allow a client to be informed upon a change of a property.

- The inputs of a resource are all public methods of a resource. Like the properties, the inputs and their corresponding calling interfaces are collected and published by the resource framework automatically.

- The outputs are stubs provided by the implementer of a resource. Outputs are typed in the same way as inputs. An output can be linked at runtime to one or more input if the interface matches. An execution of an output stub by a resource will result in a parallel execution of all linked inputs.

The resource framework uses the runtime class information available through the `java.lang.reflect` classes to collect and publish the features of a resource. Therefore, no extra work is required in order to implement a resource.

### 2.2 Sharing

Locally available resources are managed by a resource server running on the machine where these resources are accessible. The resource server is responsible to publish the local resources
and their interfaces on the network. A resource server might also act as meta-server. Instead of accessing a resource directly, it will translate the calls to some other protocol such as the WOSRP defined in [6].

In order to allow sharing of resources, the local resources published through the resource server have to meet in a globally accessible place. Such a place is called a resource set. A resource set can be seen as a coordination space [5], a medium which enables sharing through the coordination of distributed applications.

As far as the implementation is concerned a resource set can be located anywhere on a network and may be distributed and replicated. Resources can dynamically be added to and removed from a resource set. The resource set is responsible for tracking the location and the availability of resources. For example, if a local resource server breaks down, the resource set has to temporarily remove the resources as they are no longer accessible.

It turns out that the resource set is a network wide accessible directory of resources located on that network. Usually resource sets will be created to solve a specific problem. This means that many resource sets may exist simultaneously. Because resource sets will be reused and modified over time, they are persistent.

For purpose of homogeneity communication takes place through Java RMI [12].

2.3 Interaction

The success of the Web browser interface has shown the need for a graphical representation tool for resources on a network. Therefore, a graphical user interface permits the interaction with a resource set. The user interface follows the desktop metaphor. Today's users know how to manage their local resources through a desktop-interface. This understanding will be used to manage resources in a similar way.

A new feature not found in the current interfaces is resource linking. The pipe operator of the UNIX shells is, although limited, an implementation of resource linking. In the interface used for the interaction, this linking will be done graphically and goes far beyond input/output redirection. Since all resources publish their interfaces, properties, inputs and outputs can be matched and interconnected arbitrarily. Resources can be linked together to form a data-flow network by connecting their inputs and outputs. Such a data-flow network acts as a transaction in a resource set and can be used for problem solving.

Again, the graphical interface supports a wide range of platforms, thanks to Java.

3 Example

This example will show the interaction of the architectural parts as well as the representation on the user side.

Imagine two users A and B located thousands of miles away from each other. How could A print a document on user's B printer? Let's first suppose that both have access to the Internet, but their only way to communicate is email, which is a very common situation. So the user A that owns the document attaches it to an email that is sent to user B. B will receive the email, extract the document - but unfortunately - it is in a format user B never heard of. The sender of the email will tell B to use this and this converter that user B should
have locally but which has to be recompiled for user B's environment. You might know the end of this story.

Using resource sets makes the situation more comfortable. To start, a resource set will be created or reused. This resource set might already exist because it represents a project on which both users are working anyway. All user B needs to do is to add his printer to the resource set. Then the other user can add his converter program and link it to the printer.

To print the document the user can simply drop it onto the converter program. This will initiate the data-flow through the network. The document will end as a printed version in user B's hands.
Figure 5: Documents can now be converted and printed by dropping files.

What happens behind the scenes? First of all, both parties have to locate the resource set. A resource set might be located through a well-known location such as a Web-page or some dynamic mechanism, similar to a DNS-lookup or the eductive engines of [7].

When the resource set is found, the users can connect to it. From now on, the resource set knows about the location of the users, and more important, where the resource server of each user is running. Adding resources is now straightforward.

The user links the resources by choosing an output of the converter program and linking it to a type compatible input of the printer. Dropping the file onto the converter will initiate the data-flow. The converter resource will start a process on A's machine. The output of this process (stdout or files) will then be passed by the converter resource to the printer resource. Finally the printer resource performs the printing by starting the printing process on B's machine with the received data.

4 Conclusion

This paper presented an approach inspired by coordination theory for interactive resource sharing on the Web. Every user can share his/her local resources with other users and shared resources are collected in a persistent resource set which implements a coordination space of [5]. Users contributing resources to a resource set can combine and use the resources in the resource set for interactive problem solving. Collaboration of geographically dispersed people is facilitated by using the well known desktop metaphor augmented by a powerful link technique to interact with the resource set.

We are currently in the early development stages of a first prototype. Using Java [4] as the implementation language has many advantages. Beside its good support of and integration in web technologies, the availability of runtime class information and RMI simplifies the implementation. Java RMI [12] [9] in special proved to be very stable.

We are convinced that our object oriented "wrapper" technique to integrate resources into the Web is a powerful mechanism to share heterogeneous resources on the Web. The user
interface part of our prototype has been already developed and we are planning to integrate
the upcoming JavaSpaces platform [11], a system which implements the Linda coordination
model in an object oriented way, as our main implementation platform for resource sets.

References


Proceedings ACM Symposium on Applied Computing (SAC ’98) Special Track on Co-
ordination Models, Languages and Applications, Atlanta, GA, U.S.A., Feb. 27 – March

Departement of Math and Computer Science, Emory University, 1996.

Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, CA 94042-1100 U.S.A., 1.0 edi-
tion, August 1996.

Coordination Model for Massively Parallel Systems. In Proceedings of the ’95 SIPAR-
Workshop on Parallel and Distributed Computing, Biel, Switzerland, October 1995.

[6] Peter Kropf. WOS Request Protocol. Université Laval, Département d’Informatique,
December 1997.

Technical report, Université Laval, Département d’Informatique, Sainte-Foy (Québec),
Canada, G1K 7P4, 1997.

(WOS). Technical report, Université Laval, Département d’Informatique, Sainte-Foy
(Québec), Canada, G1K 7P4, 1997.

Method Invocation. Technical report, Computer Science Department, University of


Avenue, Mountain View, CA 94042-1100 U.S.A, 0.4 edition, June 1997.

tems, Inc., 2550 Garcia Avenue, Mountain View, CA 94042-1100 U.S.A, 1.4 edition,
February 1997.