# Services for Virtual Teams Hosting : ToxicFarm Introduction

François Charoy, Claude Godart, Pascal Molli, Gérald Oster, Marc Patten, Miguel Valdes

# SERVICES FOR VIRTUAL TEAMS HOSTING

*ToxicFarm Introduction*

F. Charoy, C. Godart, P. Molli, G. Oster, M. Patten and M. Valdes
*ECOO Team, LORIA–INRIA Lorraine*
*BP 239, 54506, Vandœuvre-lès-Nancy Cedex, France*
{charoy,godart,molli,oster,patten,valdes}@loria.fr

**Abstract** Virtual teams provider is an emerging business on the Internet. It allows people to work together distributed across space, time and organizations. This paper overviews and discusses a set of generic services requested to host a virtual team. These services and their integration are illustrated thanks to the *ToxicFarm* environment developed by the authors.

**Keywords:** CSCW, Virtual Team, Awareness

## Introduction

If for most of people, Internet is huge database in which they can find information responding to a so huge spectrum of requests, it is already an inescapable working tool for some of us. And that does not concern only tele-work or electronic commerce, but complex business activities depending on the ability to solve problems through the collaboration of groups of people, within and across organizations. This includes co-design, co-engineering, knowledge sharing, decision making applications, with long term or ephemeral cooperation links.

And there is now a multitude of tools to support cooperative work in the world of Internet: TeamScope (TeamScope, 2000), BSCW (Bentley et al., 1997) , SourceForge (SourceForge, 2000) ... And some of them are now widely used: as example, SourceForge hosts more than 30 000 open source projects for more than 300 000 virtual members (this statistics makes reference to version 2.0 before privatization). These projects are mainly software design and development projects.

The objective of this paper is to overview a set of generic services requested to host a *virtual team*, i.e. a team distributed in space, in time and potentially in organizations, and to discuss their interactions.

The discussion is driven by author experiments in the development of the *ToxicFarm* environment (http://woinville.loria.fr), an environment in the vein of SourceForge, but with a particular focus :

- on virtual team *coordination*, a crucial problem in a virtual team where the possibility of informal discussion is lost,
- with consideration of a larger set of applications than only software engineering, including co-design and co-engineering applications in general..

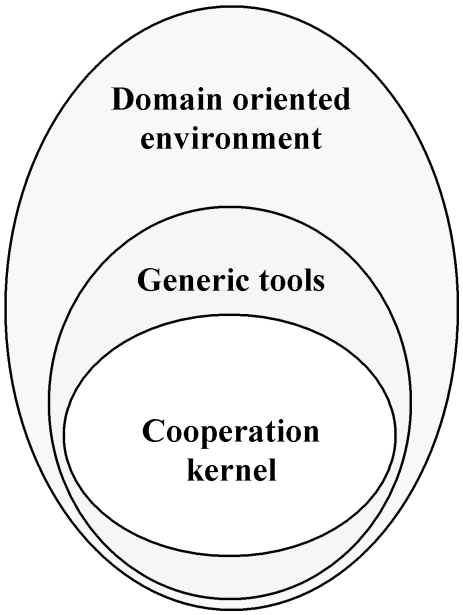


*Figure 1.* Cooperation architecture.

More precisely, we are concerned with the development of a generic kernel that is a basis for the development of several environments in different application domains, i.e. not for the development of only one specific environment in only one specific domain. That is what figure 1 illustrates.

Examples of applications in which we are, on the one hand searching for requirements and inspiration, and on the other hand experimenting our software, are :

- Co-design and co-engineering in the domain of AEC (Architecture, Engineering and Construction). We studied especially in this project the definition of cooperation bricks to support cooperation in realistic scenarios of buildings co-design and construction. This work has been developed in collaboration with architects of the Research Center in Architecture of Nancy and France Telecom R&D.

- Distributed software development in the PureSource project [1]. In this project the objective is to develop an environment for cooperative and distributed software development, in the vein of SourceForge, but with a particular focus on team coordination and software quality,

- Cooperative learning, in the RIAM Coopera project [2]. The objective is to use Internet and cooperative work for learning 10 to 12 years old children, not only to use Web and Internet, but also and especially to organize their work, including cooperative work between classrooms of different schools.

If these three applications clearly need three different domain oriented user interfaces, they share a lot of functionalities that can be grouped together in a kernel and presented as Web Services.

The paper is organized as follows: section 1 provides a rapid classification of virtual team hosting services; section 2 overviews these services and discusses one instance of them: our *ToxicFarm* environment (http://woinville.loria.fr/). Section 3 quickly discusses the Web Services based implementation of *ToxicFarm*. Section 4 concludes.

## 1. Classification of services

This section quickly classifies and overviews the services that we feel important for supporting a virtual team. Of course, this classification is debatable and the same tool can be seen in different classes by different persons, or by the same person at different time, but we feel that it is a good start for discussion. We distinguish between: project administration, object sharing, communication, coordination, mobility, audit, security and interoperability services. This section simply lists these services, the next section overviews them and concretizes them from the point of view of *ToxicFarm*.

**Project administration**. The first think to do, before to be able to start to work is to create a new project, including the registration of at least one administrator for this project. Once the project is registered, other members can register as member of this project under the control of

one administrator. At this time, all the structures necessary to support cooperation services need also to be initialized.

**Object sharing**. In most applications, object sharing is the minimum service to provide for cooperation support. This implies, in addition to object storage, at least the management of versions and of partner privacy, including strict access control and workspace management.

**Communication**. Team members need to talk to each other, to discuss, to show and to update shared artifacts. Most groupware tools can be useful, but they should be integrated as much as possible in the virtual team support environment. This includes chat, email, videoconferencing, white boards and so on.

**Coordination**. One important difficulty introduced by distribution is, due to the loss of easy meeting possibility, the coordination of the activities of team members. We address this problem in two ways (Godart et al., 2001):

- *task coordination* (or explicit coordination) based on the hypothesis that it is possible to define a process and to enforce this process on working sites,
- *group awareness* (or implicit coordination) based on the hypothesis that if the right information, about what other people do, is sent at the right time to the right people, this information will trigger communication between people that will result in an autocoordination of the virtual team.

**Mobility**. The ability for a user, on the one hand to easily and quickly connect to his ongoing projects from any point, on the other hand to work disconnected from the service provider server, is unquestionably requested.

**Audit**. Audit information, concerning as example the rate of transfers between workspaces, the number of new or completed tasks ... is of first interest information for measuring a project activity. The number of usage of a specific tool can also be used to measure the interest of maintaining this tool in the environment. More generally, such information is requested to support any usage analysis.

**Security management**. Security is an important issue for virtual teams members who want to work securely on secure data, especially if data are hosted on a site which is not a part of their organization, as that occurs in virtual teams. Trust in communication media, trust in other members identity, trust on how data are stored are essential.

**Interoperability**. Due to the multiplication of virtual team services providers, especially since SourceForge has been put in private hands, considerations of interoperability between virtual teams hosts becomes

valuable.
This concludes our list of services. It has not the pretension to be exhaustive, but we think it is representative of the problematic. In the following paragraphs, we refine these services and illustrates them from the point of view of *ToxicFarm*.

# 2. Cooperation services and *ToxicFarm* implementation

## 2.1 Project administration

The first action to undertake when creating a new distributed project is to register this project and to define at least one project administrator. This action must be as simple as possible and take minutes and not hours. This should consist mainly in filling web forms. A new user should also register as simply, in giving only a name and an electronic address through a Web form.

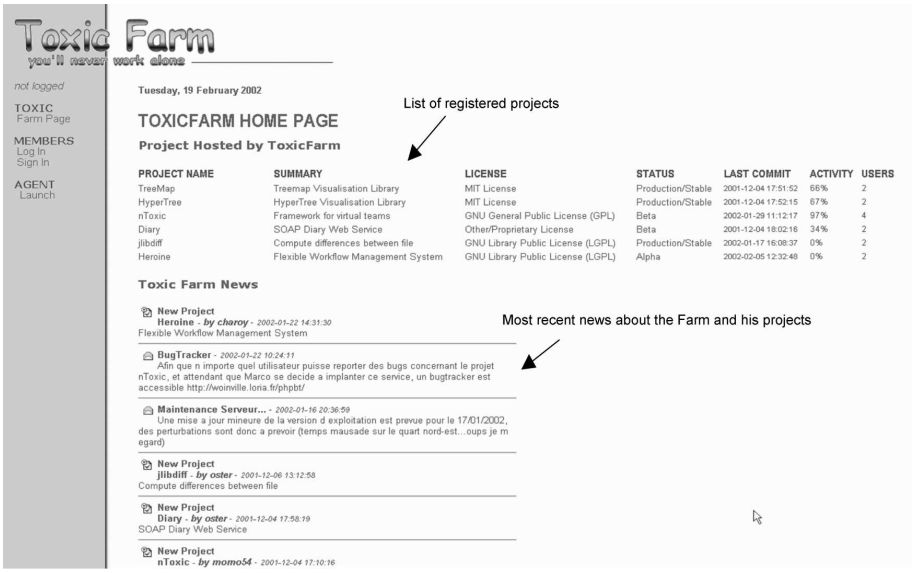


*Figure 2.* *ToxicFarm* homepage.

In *ToxicFarm*: when a user accesses the home page of the *ToxicFarm* system (see figure 2), he can log in. When logged in, he is displayed his personal home page. There, he can visualize the list of projects he is a *member of*, and read the related news. He can also easily create a new project which he becomes administrator of. The administrator of a project can add a new user in his project, giving him either administrator

or user role. If a user is not registered in the system, he can easily do it by giving a name and an email address.

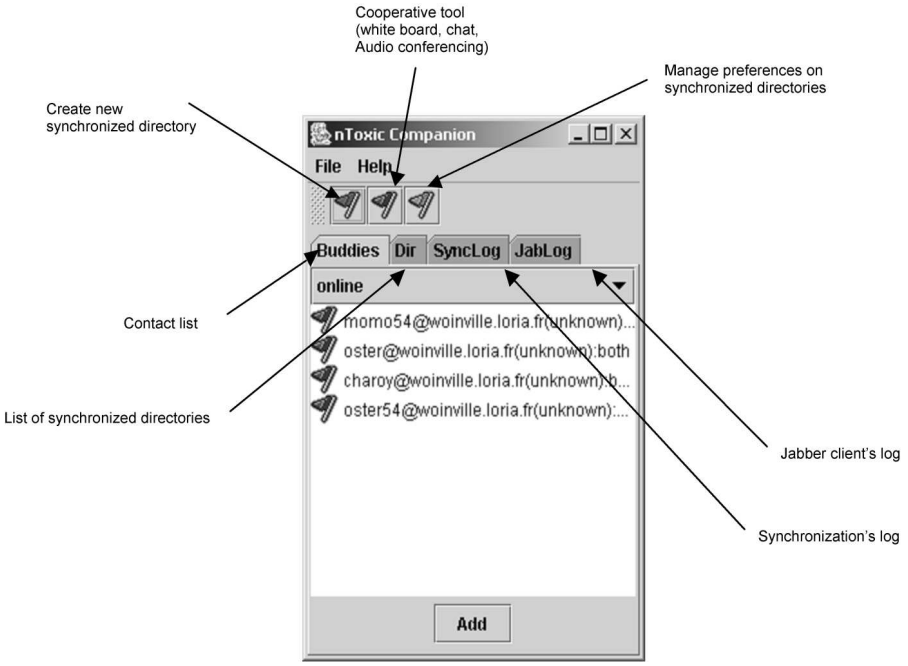


*Figure 3.* The Companion interface.

Selecting a project, a user is associated a *companion*. His companion is a very important "people": it accompanies him all along his work and allows him to be aware of his situation, with regards to his proper tasks but also, and above all, with regards to other project member work. We will deepen the companion role all along the next sections; see figure 3 for the look of the companion.

## 2.2 Shared object services

Sharing objects is the minimal condition for cooperation. This includes a lot of aspects.

**Objects and Dependencies**. Services must provide support for at least files and directories, in combination or not with a typed databases.

**Access Control**. Access control is also critical. At least Access Control Lists must be provided, allowing only authorized members to read or modify an object.

**Versioning**. Versions are used for various purpose: concurrent engineering, experimental development, variants, trace ability and so on (Conradi and Westfechtel, 1998).

**Concurrency**. By definition of sharing, concurrent accesses must be managed In the context of file version management, this can be achieved by more or less sophisticated methods (locking policies, *Copy-Modify-Merge* paradigm, transaction models ...).

**Notification**. This allows to track important events that occur on shared objects and that are relevant of the project life cycle. This notification is heavily used by awareness engines.

**Annotation**. Annotation facilities can be easily integrated to object sharing.

**Workspace management**. In the application we are concerned with, members of a team need different levels of privacy with regards to others. Complementary to access rights, the management of workspace is strongly requested: to modify a data, a user must check out it from a common shared workspace, modify it in its private workspace and then commit his change back to the common workspace.

**Heterogeneous data management**. A high value contribution that can be added in shared data services concerns the management of heterogeneous data (filters, transformers ...).

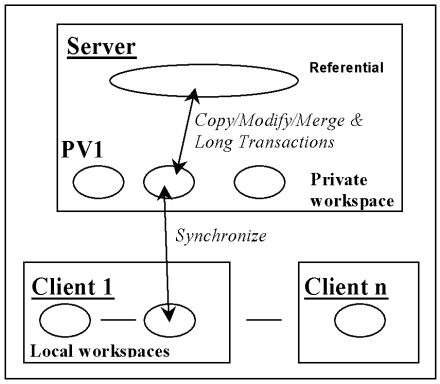


*Figure 4.* *ToxicFarm* architecture.

In *ToxicFarm*, there is one *referential directory* (common workspace) that contains all objects shared in a project (see figure 4). These objects are multi-versioned.

To be able to work, the first operation is to create a *private workspace*. This is simply done by clicking on the *create directory* button of the

*companion* (see figure 3). This results in a directory that is a copy, on the server machine, of the referential directory.

For facility and/or mobility purpose, a user can create a private *local workspace* on his own machine that is a copy of a private workspace on the server. This is simply done by clicking on the *synchronize* button of the *companion* and indicating a directory where to store the workspace copy on the local machine.

A user can modify an object either in his workspace on the server, through a web interface, or in his local workspace on his own machine. All the manipulations of objects in a local workspace are through local tools and local system. This allows a user to continue to operate objects with his usual tools.

Periodically, a user can synchronize his local copy with the corresponding server workspace. This is also simply done by clicking again on the *synchronize* button of the *companion*. Clicking on the SyncLog button of the companion, the user can see all the differences between his server workspace and his local workspace that have been synchronized. Synchronization between a server workspace and the referential is based on the *Copy/Modify/Merge* paradigm and the Long Transaction Model (Feiler and Downey, 1990) : a user checkout, update and commit a complete workspace, and not a file or a directory as in the simple Copy/Modify/merge paradigm (CVS (Berliner, 1990)).

This architecture distinguishing between two levels of workspaces actively supports mobility and disconnected work, as developed in section 2.7.

Events on objects and object states in different workspaces are the main support for awareness based coordination, as we will detail in section 2.4.

*ToxicFarm* provides also for traditional access rights control.

## 2.3 Communication services

All communication media (voice, video and typed documents) can be useful. Communication can be synchronous or asynchronous:

**Synchronous**. Video-conference tools (CUSeeMe, 2002), ip-telephony, application sharing (NetMeeting, 2002), collaborative tools, groupware toolkit (Greenberg and Roseman, 1999) and real time editors (Ellis and Gibbs, 1989) can be used.

**Asynchronous**. As experience demonstrates, asynchronous communication is very important because people working on the same project, but on different sites, generally do not work at the same time. And of course, this is yet more true if there is a time difference between work-

ing sites. Traditional e-Mail, mailing-list and web pages can be used. Object sharing itself is a basic way of communication.

Communications in *ToxicFarm* is based on traditional means: instant messaging, mailing list. They are coupled with objects states and the companion to inform users of principal events.

Voice conferencing coupled with a shared whiteboard allows users to discuss idea with other members during synchronous work session. This can be simply triggered by clicking on a companion button.

## 2.4 Coordination services

**2.4.1 Task coordination.** When people work together in a common objective, it exists some defined tasks that are more or less formalized. This can take different forms.

**Project Management Tools** define tasks and their synchronization (process). Tasks are defined but not enacted: execution of tasks is not controlled by the tool and the link between project management and working sites is weak.

**To-Do lists**. In this case, tasks are dynamically and opportunistically defined by members without any planning. If this can be an efficient cooperation support, it is inefficient for project management.

**Workflow** provides both process modeling and enactment, but we have to notice that current workflow technology is not adapted for all aspects of task management, especially for synchronization of the more creative tasks.

More generally, this topic is especially difficult, because:

- process is matter of *how people work*, and in general people does not like to say how they work, because on one hand it is difficult to precisely describe, on the other hand they are afraid to lost some "prerogatives",
- in a complex project, on the one hand, there is not only one process, but several process fragments that co-habit, and on the other hand, these processes are of different nature : some are very simple scripts, others are traditional workflow in the sense of the WfMC (WfMC, 2002), and finally other ones are gross grain processes with subtle interactions between their activities (we call them later *cooperative workflow*),
- cooperative workflow technology is not mature.

In *ToxicFarm*, task coordination is currently mainly based on To-Do lists. A project member can consult his list of tasks and their states,

he can create a new task and assign it to another project member, and follow the states of these tasks. All these operations are done through Web pages.

However, *ToxicFarm* will also provide shortly innovative support for process management in two ways :

- placeholders for script processes,
- COO-flow (Grigori et al., 2001) cooperative workflow system for the support of traditional workflow and cooperative workflow definition and enactment.

**2.4.2 Group awareness.** Awareness services keep people aware of project activity. They are deeply depending on shared data services and also task coordination services. Their role is to gather, filter, aggregate events coming from these sources and finally to deliver them to final users. Awareness must be pertinent and must not increase the cognitive overload of final users. How awareness is visualized by users is also an important issue. Different classes of awareness can be distinguished: *state awareness* based on the state of shared data, *availability awareness* based on the knowledge of the physical presence and current status of a member, and finally *process awareness* based on the knowledge of current activities situated in a global process. Of course, this list wants not to be exhaustive.

In *ToxicFarm*, as introduced above, selecting a project, a user is associated a companion that will help him all along his working sessions. Especially, it will inform him of the main events (new user, news about project, commit ...) and give him some means to react to some of these events.

One other central aspect of *ToxicFarm* awareness is *state awareness*, i.e. awareness about the state of shared objects: to each representative state is associated a color (as example, a yellow colored file in a user workspace means that the file is *locally modified*, i.e modified by the user himself in his current workspace; a pink colored means *remotely modified*, i.e. modified by another user in another workspace; if the file is locally and remotely modified, it is colored in orange, indicating a *potential conflict*). State awareness is dispatched in different ways:

- a colored square is associated to file names in directory listings,
- a synthesis of file states is displayed in the header of each workspace web page; to each meaningful color is associated the number of files in the corresponding state,

- colored Treemaps are used to visualize a very large amount of objects (Shneiderman, 1992). In the same time that state awareness (Molli et al., 2001) is displayed, information about conflicting people and about the quantity of divergence between conflicting objects can be displayed (as example the number of different bytes between the local value and the referential value).

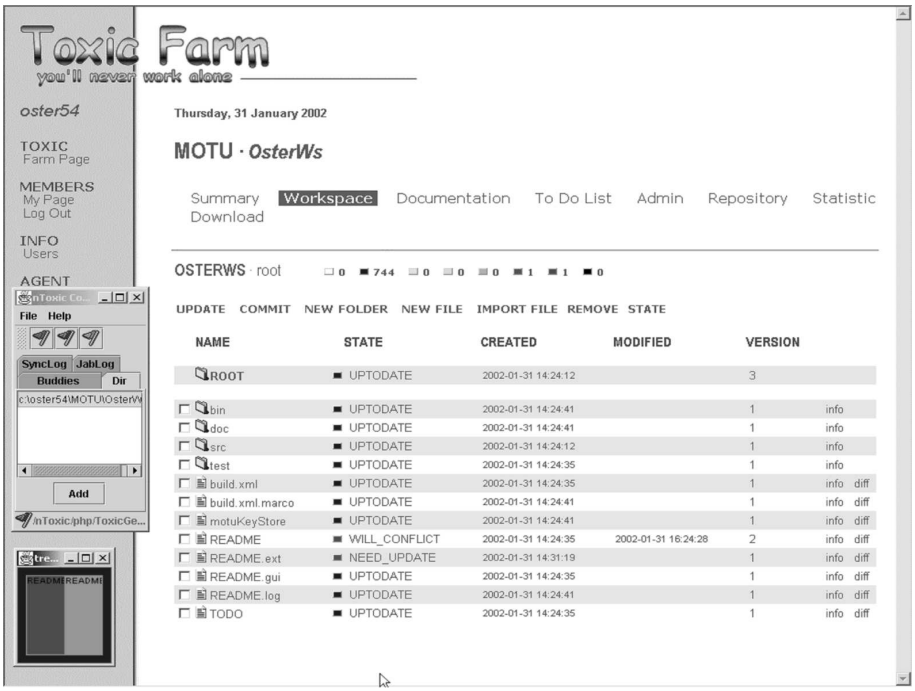


*Figure 5.* Workspace and state awareness.

This is partly illustrated by figure 5. This figure displays the content of the server workspace called *OsterWS*. It indicates that this directory content 744 files that are up-to-date with the referential, one file that *need_update*, i.e that has been modified in the referential since the last checkout of this file, and one file that *will_conflict*, i.e. that is currently modified by the owner of *OsterWS* and by another user.

In the right low quarter of the screen, there is a tree map that repeats this information. If the interest of a tree map is not clear in this case, because there is only two files being modified, its interest is clear when there is a lot of such files, as it allows to analyze in a quick look the degree of divergence in the project.

Users are aware of which members are connected or not thanks to Availability/Presence awareness provided by an instant messaging client (a ICQ-like tool supporting Jabber protocol (Jabber, 2002)). Moreover, status of users can give information about what team members are doing.

State awareness and presence awareness are easily accessible though the companion interface.

To-do lists is also a good source of process awareness. But process awareness can be more complex and more efficient if tasks are parts of a process and if the process status is used to display the status of interactions between project members. This will be driven by cooperative workflow.

## 2.5 Security management

Existing tools provide more or less services for security purpose. But often, identity of members and authentication are not well handled. If security is supported by the service provider, it may conflict with local security policies. In this case, tunneling, extra-authorizations can be required within the organization of virtual team member. But are such considerations compatible with the high-productivity nature of virtual teams. Anyway, security has to be ensured in a transparent way for the user. This requirement is more crucial if we take into account the fact that virtual team members can be mobile. If members are not sure to use virtual teams services after moving to another place, then virtual teams services lost a lot of interests.

In *ToxicFarm*, security is limited to user access control based on roles and is mainly based on role and encryption in order to guarantee confidentiality of data during network transmission.. Moreover, our environment supports the *Secure Sockets Layer* (SSL, 2002).

## 2.6 Audit services

A project member can have a good idea of project efficiency thanks to statistics. An important information is the activity rate of a project that allows to measure the dynamism and the effectiveness of the project and the corresponding software, but force is to note, that the universal algorithm to measure the activity of a project has not yet been defined.

In *ToxicFarm*, different kinds of statistics (number of accesses to project home page, number of downloading, number of commit operations ...) can be defined and visualized in different forms (pie charts and plot graphs).

An algorithm to evaluate the activity rate of a project is being designed.

## 2.7 Mobility

As introduced above, mobility is a quite sensitive capability that is often poorly considered in most systems.

In *ToxicFarm*, thanks to its server based architecture combined with its web interface, a user can find his working environment from any point in the world as soon as he can access to Internet. In addition, thanks to the duplication of a user workspace on the server and on his working machine, a user can easily disconnect himself from the server to work at home or in a train. When he comes back, he has only to synchronize his work with work done during his absence. And state awareness, news and update logs allow him to be aware of what happened during his absence.

## 2.8 Interoperability

Since SourceForge has been put in private hands, several initiatives have been developed to host software developments: Savannah (Savannah, 2002), TuxFamily (TuxFamily, 2002) ... In the same time, the need to support interoperability between them has occurred. One good representative of this work is the CoopX (CoopX, 2002) project, the objective of which is to develop a standard format based on XML to exchange information on projects hosted by such environments. With such a format a project manager should be able to migrate his project from one hosting platform to another, or to mirror it.

## 2.9 Other services

*ToxicFarm* also provides services for in line documentation and downloading.

# 3. Implementation

*ToxicFarm* platform has been designed and developed with largely used open source technology. As a consequence, it is easy to deploy and maintain on most representative systems. Thus the current version requests only a Web, a database and an instant messaging server for running *ToxicFarm*.

Web technology provides several advantages: availability, scalability, uniform HTML interface, and component based software engineering thanks to Web services technology.

The SOAP (SOAP, 2002) protocol is used, on the one hand to describe *ToxicFarm* services API, on the other hand to integrate new services presented using this protocol. As example, awareness based component uses the SOAP API of the Workspace Service to know the state of shared

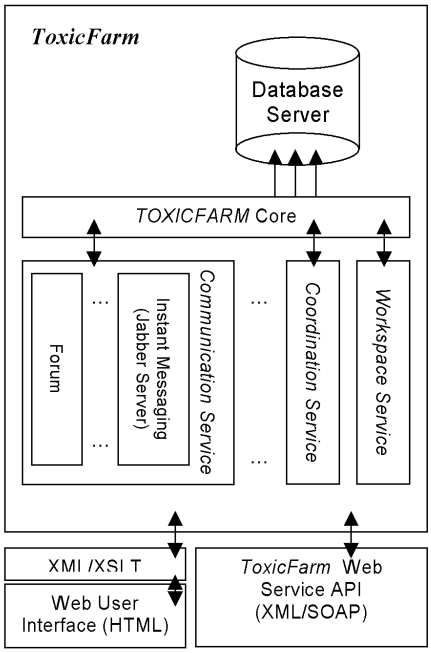


*Figure 6.* *ToxicFarm* Server architecture.

objects. Another example is the cooperative process management service that will be easily plugged into *ToxicFarm* because its API conforms to the SOAP protocol.

XML/XSLT (XML/XSLT, 2002) standards are also largely used for message and presentation normalization. This, combined with SOAP allows to present the same service in different ways depending on the application domain, or to change the implementation of a service without changing the service interface.

In the current version, the database server is MySql (MySQL, 2002), but it can be exchanged provided that the new database server is able to respect the corresponding SOAP API, that is the case of most database servers provided it is able to manage traditional transactions. The database server is in charge of administrative objects management : projects, users, workspaces, information about application objects ... Of course, contents of shared application objects are stored in some file management system(s).

The Core plays the role of coordinator between the different components. It is based on PHP technology (PHP, 2002).

## 4. Conclusions

Of course, the classification proposed in this paper can be discussed. Especially, may be that it points out social considerations (communication, coordination) to the detriment of security and networking aspects. But, we are force to notice that most of current tools on the topic are more concerned with improvement of awareness, workflow and groupware than security purposes.

Our *ToxicFarm* environment is a representative of the systems in the vein of SourceForge dedicated to manage a distributed, but with a particular focus on coordination and with a larger scope of applications. Its design based on Web Services technology makes it a flexible and adaptable system.

A test version is currently accessible online (http://woinville.loria.fr/) and a larger deployment is forecasted in a short time. Some usage analysis have started in domain of software engineering and e-learning.

## Notes

1. PureSource is a project of RNTL (French National Network for Software Technology), http://puresource.org/

2. Coopera is a project of RIAM (French Network for Research in Audio-visual and Multimedia), http://www.cnc.fr/riam/

## References

Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Sikkel, D. K., Trevor, J., and Woetzel, G. (1997). Basic support for cooperative work on the world wide web. *International Journal of Human Computer Studies: Special issue on Innovative Applications of the World Wide Web*, 46(6):827–846.

Berliner, B. (1990). CVS II : Parallelizing software development. In *Proceedings of USENIX*, Washigton D. C.

Conradi, R. and Westfechtel, B. (1998). Version models for software configuration management. *ACM Computing Surveys*, 30(2).

CoopX (30 July 2002). *Online http://coopx.eu.org.*

CUSeeMe (30 July 2002). *Online http://www.cuseemeworld.com.*

Ellis, C. A. and Gibbs, S. J. (1989). Concurrency control in groupware systems. In *SIGMOD Conference*, volume 18, pages 399–407.

Feiler, P. H. and Downey, G. F. (1990). Transaction-Oriented Configuration Management: A Case Study. Technical Report CMU/SEI-90-TR-23 ESD-90/TR-224, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Godart, C., Bouthier, C., Canalda, P., Charoy, F., Molli, P., Perrin, O., Saliou, H., Bignon, J.-C., Halin, G., and Malcurat, O. (2001). Asynchronous Coordination of Virtual Teams in Creative Applications (co-design or co-engineering) : requirements and Design Criteria. In *Information Technologies for Virtual Enterprises.*

Greenberg, S. and Roseman, M. (1999). *Groupware Toolkits for Synchronous Work*. John Wiley and Sons Ltd.

Grigori, D., Charoy, F., and Godart, C. (2001). Flexible Data Management and Execution to Support Cooperative Workflow : the COO approach. In *CODAS*. IEEE.

Jabber (30 July 2002). An open xml-based presence and instant messaging. *Online http://www.jabber.org*.

Molli, P., Skaf-Molli, H., and Bouthier, C. (2001). State Treemap : an Awareness Widget for Multi-Synchronous Groupware. In *7th International Workshop on Groupware - CRIWG'2001, Darmstadt, Germany*.

MySQL (30 July 2002). *Online http://www.mysql.com*.

NetMeeting (30 July 2002). *Online http://www.microsoft.com/netmeeting*.

PHP (30 July 2002). *Online http://www.php.net*.

Savannah (30 July 2002). *Online http://savannah.gnu.org*.

Shneiderman, B. (1992). Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Computer-Human Interaction*, 11(1):92–99.

SOAP (30 July 2002). Simple object access protocol. *Online http://www.w3.org/TR/SOAP*.

SourceForge (28 June 2000). Breaking down the barriers to open source development. *Online http://www.sourceforge.net*.

SSL (30 July 2002). Secure sockets layer. *Online http://netscape.com/security/ssl.html*.

TeamScope ((28 June 2000)). Software for collaborative project environments: a web-based collaboration tool. *Online http://cscw.msu.edu/scope.html*.

TuxFamily (30 July 2002). *Online http://www.tuxfamily.org*.

WfMC (30 July 2002). Workflow management coalition. *Online http://wfmc.org*.

XML/XSLT (30 July 2002). *Online http://www.w3.org/TR/xslt*.