SAMS: Synchronous, Asynchronous, Multi-Synchronous Environments

Pascal Molli, Hala Skaf-Molli, Gérald Oster, Sébastien Jourdain
ECOO Team
LORIA, INRIA Lorraine
BP 239
54506 Vandoeuvre-les-Nancy, France
{molli,skaf,oster,jourdain}@loria.fr

Abstract

In the context of cooperative work, a team alternates divergence phases where each member works in insulation on copies of objects and convergence phases during which the group reconciles and validates data. To support this style of working, we propose the concept of SAMS environments. A SAMS environment allows team members to work in Synchronous, Asynchronous or Multi-Synchronous mode while ensuring the coherence of shared data.

1. Introduction

"Virtual teams work across space, time and organizational boundaries with links strengthened by webs of communication technologies" [10].Virtual teams are useful because they can be quickly brought together to produce a business objective within limited time and resources. This provides the opportunity for different organizations to cooperate by leveraging their core competencies. One can completely sets up a virtual team to carry out software development, book writing or building design. We can imagine companies like Bull France, IBM USA and Hitachi Japan delegate a few engineers to build quickly a prototype for a new promising technology. These engineers do not work in the same place, at the same time and do not belong to the same company.

Team members can interact in various ways :

Synchronous members work at the same time on the same data. Modifications on one shared object are carried out immediately and observed in a real time by other team members. Shared application's tools, like NetMeeting, allow synchronous work.

Asynchronous members work at the same time or postponed on the same data. Modifications on shared objects are carried out immediately and are observed by other members either immediately if they are connected, or delayed until they reconnect themselves[14]. Online web pages editors allow asynchronous work.

Multi-synchronous each member has a copy of the shared data. They modify their copies in parallel. This allows them to achieve their objective quicker. Of course, that does not go without posing problems of coherence between the various copies of the shared data. Work is a cycle of divergence and convergence. During divergence phases, each participant works in insulation. During convergence phases, participants synchronize their different copies to reestablish a common view of the data. Further individual activities will cause divergence again, necessitating further synchronization and so on [5, 6, 12]. Configuration Management tools [8] like CVS [3], ClearCase [1], NSE [9] are multi-synchronous environments for software development.

Existing tools and environments support synchronous and/or asynchronous or multi-synchronous mode separately. But no one provide all those three modes in a single environment.

However, it is interesting to have this kind of multi mode environment. Synchronous work seems to be suitable for conflicts resolution phases. Asynchronous work is more suitable for integration phases. Multi-synchronous work is adequate for production phases.

We develop an original concept of environment allowing working in Synchronous, Asynchronous and Multi-Synchronous modes (SAMS) [4]. Users of SAMS environment can choose interaction mode according to their needs, and the environment will ensure the coherence of data. In this paper, we give the main principles of such environment, which allow you to build your own SAMS environment.

We have implemented the first SAMS environment. This environment has two editors: CRC cards editor [17] and HTML editor. It is independent of the type of manipulated data. We can thus build a SAMS editor for

HTML, XML, text, CAD document or even more largely for calendar, bookmarks ...

The paper is organized as follows: The next section presents SAMS editors for CRC cards and for HTML. Section 3 gives the main principles of SAMS environments. The last section concludes with some pointers on future works.

2. Examples of SAMS Environment

Figure 1.[1] presents the SAMS editor of CRC cards. CRC cards (Class, Responsibility, Collaboration) are used in objects oriented design to define classes and components of a software system.

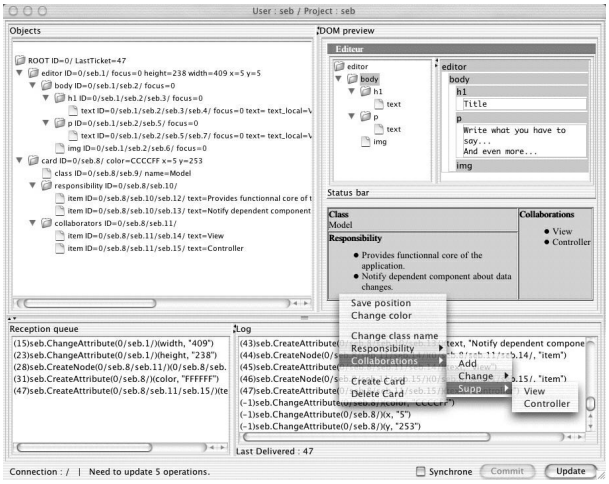

Figure 1. SAMS-CRC editor

The part *DOM preview* introduces the editor itself. It allows creating and manipulating the cards.
The part *Objects* shows the local state of the shared objects. In our case, it is an XML tree.
The part *Log* represents the log of operations applied to local objects. It contains all executed operations in a site.
The part *Reception queue* shows operations received from other sites that have not yet been integrated.
Finally, *Synchrone, Commit* and *Update* commands allow to choose the interaction mode with the other team members.

When the user checks the *Synchrone* box, his operations are immediately propagated to the other sites. Received operations from other sites are also immediately integrated. This is the synchronous mode.

If the *Synchrone* box is not selected, then the multi-synchronous mode is activated. Local operations are sent to the other sites when the user clicks on *Commit*. Received operations are integrated when the user clicks on *Update*. The user can send his local operations only if he has already integrated all the operations submitted by the other users.

If the user is not connected, then all the operations sent to him are stored in a persistent, fault-tolerant queue of messages. This is the asynchronous mode.

Our SAMS environment is based on XML object model. Editors available in the environment are viewers and controllers of a single given model. To illustrate our aim, the figure 2 shows the same SAMS environment where we replace the functions of edition of CRC cards by an editor of structured HTML document.

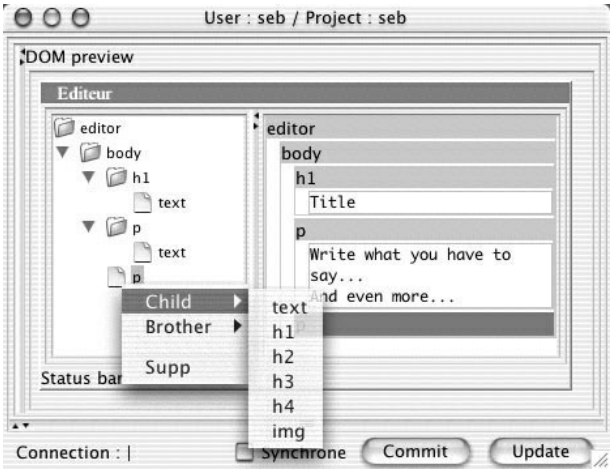

Figure 2. SAMS-XML editor

It is also possible to have the editor of CRC cards and the editor of HTML document within the same environment as shown in the figure 3.

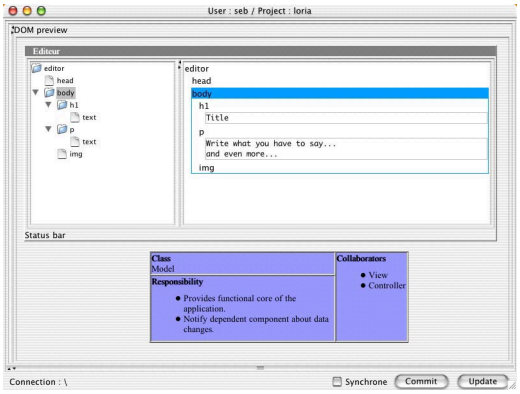

Figure 3. SAMS-XML environment

[1] The editor can be tested online at the following address: http://woinville.loria.fr/simu/

3. SAMS Environment Working Principles

A SAMS environment is based on typed objects (here XML) and log of operations. Each site has its own log. While working, a user modifies his copies. Trace of these operations is reported in his log. To propagate those operations to other sites, he has to integrate the concurrent operations before. The integration phase can generate conflicts that the environment will try to solve automatically.

Two principles drive this environment : Generating local operations and integrating distant ones.

3.1. Generating Local Operations

Each user has a copy of shared objects. In our case, the shared object is an XML tree provided with the following operations:

```
CreateNode(n,tn):nid;
DeleteNode(n):void;
CreateAttribute(n,a):void;
DeleteAttribute(n,a):void;
ChangeAttribute(n,a,v):void;
```

n is the identifier of XML node, *tn* is the name of the XML marker, *a* is the name of an attribute. *v* represents the value of the attribute.

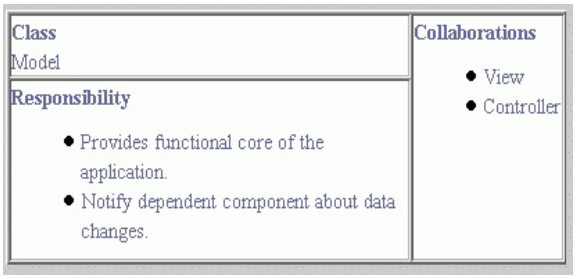

Figure 4. CRC Card

When a user creates a CRC card, he generates a sequence of elementary operations. These operations are executed immediately on his site. For example, the creation of CRC card illustrated in the figure 4 generates in the local log the following sequence of operations :

```
CreateNode(1,"Class")
CreateNode(2,"Responsibility")
CreateNode(3,"Collaborations")
CreateAttribute(1,"Model")
CreateAttribute(2,"Provides functional core of the
application")
CreateAttribute(2,"Notify dependent component about
data")
CreateAttribute(3,"View")
CreateAttribute(1,"Controller")
```

and an XML tree illustrated in the figure 5.

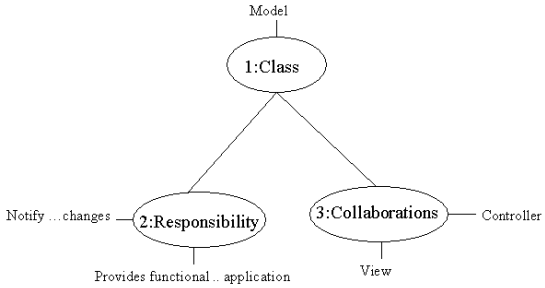

Figure 5. XML Tree

3.2. Integrating Distant Operations

The main difficulty of the SAMS editors resides in the integration phase. Indeed, *when an operation is received, the local state of the shared objects can be different from that observed during its generation*. Integrate, in our context, means transform the distant operation so it can be merged with the local ones. This transformation is not obvious. Similar problems have been treated in synchronous groupware.

In *synchronous groupware*, operational transformation algorithms are used in distributed real-time collaborative environments [15, 7, 2, 16]. In those environments, each site keeps a copy of shared objects. Operations that are locally executed on one site are broadcasted to all other sites where they will be executed. Consistency problems will occur when conflicting concurrent operations are produced in parallel. An operational transformation algorithm allows to re-establish a consistent state by merging, in real-time on each site, the locally executed operations and the concurrent ones. Merging is done while preserving intention, causality and ensuring convergence [15, 7, 16].

Causality If an operation op1 precedes an operation op2 on a site, then op1 precedes op2 on all sites.

Convergence Copies of the shared objects are identical at all sites at quiescence (i.e., all generated operations have been integrated and executed at all sites).

Intention Preservation If an operation has to be transformed, then the result of the transformation must respect the semantics of the operation.

Transformation algorithms are independent of objects types. To use a transformation algorithm, one must define his typed objects and the corresponding transformation functions. In our example, the typed object is an XML

tree provided with the operations *CreateNode, DeleteNode* ... To integrate concurrent operations, it is necessary to define the concurrent behavior of all the couples of operations; exactly 25 transformations in our example.

Now we can give some examples of transformation functions. They use the operational transformation algorithm of SOCT4 [16]. We use the following notation:

T(distant operation (not executed), local operation (executed)): transformed operation.

Function *T* takes as parameter a distant operation and a concurrent local one. The result of the transformation is a new operation.

The following function defines how to transform the received CreateNode (op1) operation considering that the operation DeleteNode (op2) was executed locally.

```
T(CreateNode(n1,t1),DeleteNode(n2)):-
  if( n1 ChildOf n2)
        return noop        /* nothing to do */
  else
        return CreateNode(n1,t1)
```

If *n1* is not a child of *n2*, then there is no conflict between op1, op2 i.e. the sequential execution of op1 *o* op2 is equivalent to op2 *o* op1. The result of the transformation is op1.

If *n2* is deleted locally and *n1* is a child of *n2*, then it is not possible to execute op1. The result of the transformation function is the null operation. Another possible solution could have been to cancel the local deletion of n2 and to execute op1. Simply we do not define the operation *undelete* for our XML tree. By writing this transformation, we make a choice for conflict resolution. We estimate that this choice respects the intention of the operation *CreateNode*.

Transformation functions depend on the transformation algorithm that we use. For example, to ensure copies convergence, an algorithm as SOCT4 [16] obliges the transformation functions to verify the following condition:

op1*o*T(op2,op1)=op2*o*T(op1,op2) [C1]

This condition ensures that: Starting with the same state, the execution (on one site) of op1 followed by the transposed of op2 with respect to op1 produces the same state as the execution (on other site) of op2 followed by transposed of op1 with respect to op2.

The following transformation defines the concurrent behavior of two *ChangeAttribut*e operations:

```
T(ChangeAttribute(n1,a1,v1),ChangeAttribute(n2,a2,v2)):-
    if n1=n2 and a1=a2 and  v1=v2
          return noop /* nothing to do */
    if n1=n2 and  a1=a2 and v1<>v2
          return ChangeAttribute(n1,a1,max(v1,v2))
    else
          return ChangeAttribute(n1,a1,v1)
```

If two users modify the same attribute of the same node with two different values then the transformation function will choose the maximum value automatically. This choice respects the property [C1]. Of course, this choice is arbitrary, one could choose the minimal value.

As we see, transformation functions make sometimes-arbitrary decisions to ensure copies convergence. However, if the state of convergence does not satisfy the users, they can continue to interact. Work in synchronous mode seems to be completely adapted to converge towards a state accepted by all users.

4. Conclusions and perspectives

A SAMS environment is an original concept. In this environment, a team member can use a working style according to his needs and the environment still ensures the consistency. Multi-synchronous mode is suitable for production phases where user wants to work in insulation and synchronous mode is suitable for discussion phases where user needs to work with others to converge towards a state that satisfy all people.

A SAMS environment is independent of shared objects types. We present in this paper our SAMS environment based on XML document. We developed in this environment two editors: a CRC cards editor and HTML editor. We could very easily add an SVG editor, UML, CAD editor…

As this environment is flexible, we can develop a SAMS environment for text editors, drawings, diaries....

However several limits remain:

Operational transformation algorithms are adequate for short periods of divergence (about a second). In multi-synchronous mode, the divergence can increase beyond this period. In this case, the arbitrary side of the transformation will tend to increase divergence. A state of convergence will be reached. Simply, it is likely not to satisfy anybody.

We currently work on a semi-automatic and collaborative resolution of the conflicts.

In asynchronous mode, the periods of disconnections can be significant. In this case, the size of the log of stored operations risks to be very significant.

We currently evaluate algorithms of log compression to overcome this problem.

SAMS environments rely on the availability of the log. It is not easy to re-use the existing tools within the environment.

We work on *a posteriori generation* of the log by using *diff* algorithms [18, 13, 11].

5. Acknowledgement

Many thanks to Marc Patten for his help.

6. References

[1] Larry Allen, Gary Fernandez, Kenneth Kane, David Leblang, Debra Minard, and John Posner. ClearCase MultiSite: Supporting geographically distributed software development. *Software Configuration Management: Selected Papers of the ICSE SCM-4 and SCM-5 Workshops, number 1005 LNCS*, October 1995.

[2] M. Beaudouin-Lafon and A. Karsenty. Transparency and awareness in a real-time groupware system. *In 5th Annual ACM Symposium on User Interface Software and Technology*. 1992.

[3] B. Berliner. CVS II : Parallelizing software development. *Proceedings of USENIX*, 1990.

[4] Abdelmajid Bouazza and Pascal Molli. Unifying coupled and uncoupled collaborative work in virtual teams. *ACM CSCW'2000 workshop on collaborative editing systems,* Philadelphia, Pennsylvania, USA, December 2000.

[5] Paul Dourish. A divergence-based model of synchrony and distributed in collaborative systems. *Technical Report EPC-1994-102, Rank Xerox Research Centre,* Cambridge Laboratory, 1994.

[6] Paul Dourish. The parting of the ways: Divergence, data management and collaborative work. *Fourth European Conference on Computer-Supported Cooperative Work,* 1995.

[7] C. A. Ellis and S. J. Gibbs. Concurrency control in groupware systems. In *SIGMOD Conference*, volume 18, 1989.

[8] Jacky Estublier, editor. Software Configuration Management: *Selected Papers of the ICSE SCM-4 and SCM-5 Workshops, number 1005 in LNCS*. Springer-Verlag, October 1995.

[9] Peter H. Feiler and Grace F. Downey. 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, November 1990.

[10] Jessica Lipnack and Jeffrey Stamps. *Virtual Teams: Reaching Across Space, Time, and Organizations with Technology.* Wiley, 1997.

[11] Webb Miller and Eugene W. Myers. A file comparison program. *Software Practice and Experience*, 15(11):1025–1040, 1985.

[12] P. Molli, H. Skaf-Molli, and G. Oster. Divergence awareness for virtual team through the web. *International Conference on Integrating Design and Process Technology (IDPT'02),* June 2002.

[13] Eugene W. Myers. An o(nd) difference algorithm and its variations. *Algorithmica*, 1(2):251–266, 1986.

[14] Mark Roseman and Saul Greenberg. Teamrooms: Network places for collaboration. *Conference on Computer Supported Cooperative Work*, 1996.

[15] C. Sun and C. Ellis. Operational transformation in real-time group editors: Issues, algorithms and achievements. *Computer Supported Cooperative Work, 1998.*

[16] N. Vidot, M. Cart, J. Ferrié, and M. Suleiman. Copies convergence in a distributed real-time collaborative environment. *Computer Supported Cooperative Work, 2000.*

[17] Nancy Wilkinson. Using CRC Cards: An Informal Approach to Object-Oriented Development. *SIGS Books*, New York, 1995.

[18] XMLDiff. Xml diff and merge tool. Online http://alphaworks.ibm.com/, (28 June 2000).