Event Points: Annotating XML Documents for Remote Sharing

Olivier Beaudoux
CER-ESEO
Angers, France
olivier.beaudoux@eseo.fr

ABSTRACT

Collaboration is heavily based on sharing documents. However, most groupware toolkits do not directly support document sharing, but rather focus on supporting mechanisms such as remote concurrent access to shared objects. We propose the notion of *event point* as a single and unified concept for defining sharing capabilities of XML documents and introduce four types of event points for real-time groupware: replication, copy, echo, and synchronization. These event points support such collaborative features as real-time sharing, synchronization, telepointing, localization, and echo. The paper presents the concept of event point, its implementation in the DoPIdom toolkit, and some sample uses in our Sovigo drawing tool.

Categories and Subject Descriptors

H5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Computer-supported cooperative work*; I.7.2 [**Document and Text Processing**]: Document Preparation—*desktop publishing, markup languages, standards.*

General Terms

Design, Standardization, Languages

Keywords

Real-time groupware, XML documents, CSCW toolkit

1. INTRODUCTION

From the Xerox Star to the World Wide Web, electronic documents have become the main artifact for computer-mediated collaboration. However, today's groupware toolkits do not provide sufficient support for document sharing and exchange. They rather focus on extending single-user interface models and on allowing remote and concurrent access to shared objects. As a consequence, groupware applications must provide their own document or presentation sharing model.

DocEng'05, November 2–4, 2005, Bristol, United Kingdom.

Scene graphs were originally introduced in 3D and VR environments in order to abstract the presentation of 3D worlds in a simple and powerful way. Recent work has shown that scene graphs are also adapted to 2D drawing and may be broadly used by Web browsers [5]. An advantage of using scene graphs to represent documents is that they smoothly integrate with XML. Our work builds on this approach. It uses scene graphs as the document presentation model, links abstract documents with their presentations, and allows the documents, and thus their presentations as well, to be remotely shared.

This paper introduces the concept of *event point* as a single and easy-to-use object that provides real-time sharing capabilities for documents and/or presentations. It presents the various possible uses of event points, explains the key aspects of their implementation, and illustrates how presentation sharing was carried out in our Sovigo drawing tool.

2. THE DOPIDOM MODEL

The *DoPIdom* model and toolkit separates interactive systems into three levels: the *Document* level holds the semantic data, the *Presentation* level displays representations of the document, and the *Instrument* level defines how users interact with documents through their presentations [3]. This model is expressed as a DOM extension that includes the definition of XML *active transformations* (eXAcT), and an *interactive component* model (iDOM) that provides the remote sharing feature presented in this paper.

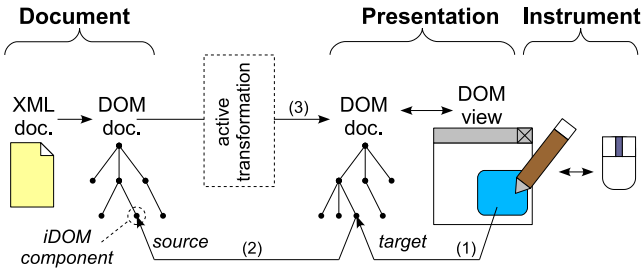

Figure 1: The DoPIdom model

Figure 1 illustrates the key principles of DoPIdom. The *document* (D) is an XML document which is accessed through the DOM. The *presentation* (P) is another DOM document displayed in a view. Following the MVC pattern [4], target elements of the presentation (1) are associated with source elements of the document (2). The active transformation [2] creates and maintains this association at runtime (3). An iDOM *interactive component* defines the *behavior* of the element. It is executable code that is dynamically linked to the source element. A user first interacts with the

document by handling an *instrument* (I). The instrument points to a target element of the presentation (1), and consequently to a source element linked to an interactive component (2). When a user interaction occurs, the instrument *produces* an action on the interactive component, then the interactive component *consumes* the action by modifying the state of the source element. Finally, the active transformation updates the target element of the presentation.

Our sharing model is based on event flow agents, called *event points*, that listen to this production / consumption cycle and replay it on remote elements. The production / consumption cycle defines the following new DOM event types: *begin*, *do*, *end*, *undo*, *redo*, *echo*, and *replication*. Technically, event points do not strictly depend on our DoPIdom model. They only rely on the DOM Event recommendation [7] and these new event types.

3. REPLICATION AND COPY POINTS

A replicated architecture is often considered as the best choice for real-time collaboration [6]. This is mainly motivated by the fact that interactive systems require a short response time.

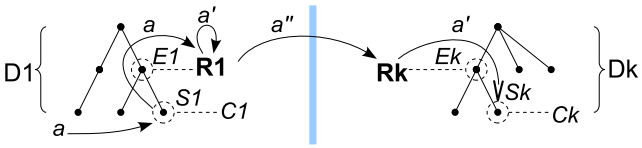

Figure 2: Replication of actions

The *replication point* is an event point dedicated to the *replication of actions* among remote DOM elements (figure 2):

1. An instrument produces an action for the source element *S1* of document *D1*, by *targeting* (in the sense defined by DOM Events [7]) the action event *a* on *S1*. The iDOM component *C1* linked to *S1* consumes the action by modifying *S1*.
2. The replication point *R1* is linked to the element *E1* of *D1*, such that *E1* contains *S1*. It is uniquely identified by its IP address and communication ports.
3. It listens to the *begin*, *do*, *end*, *undo*, or *redo* events that are targeted to *E1* or its descendant elements. This listening uses the *bubbling* mechanism of DOM Events [7].
4. When *R1* detects the action event *a*, it clones *a* into *a'* and encapsulates *a'* in a *replication event a"*. The event *a"* contains the action and the relative path *p* from *E1* to *S1*.
5. The replication event *a"* is sent over the network to all remote replication points *R2...n* paired with *R1*.
6. In turn, when a replication point *Rk* receives *a"*, it extracts event *a'* and targets it to *Sk*. The target element *Sk* is computed by extracting the path *p* from *a"*, then by finding *Sk* that matches *p* relatively to *Ek*.
7. The interactive component *Ck* linked to *Sk* finally consumes the action by modifying *Sk*.

Another way to achieve replication consists in replicating each state changes of shared elements by listening to *mutation events* (in the sense defined by DOM Events [7]). The *copy point* achieves this type of replication by directly modifying the shared element according to the listened mutation events rather than by replaying actions. Note that since remote sharing features, such as echo, require the replication of *actions*, copy points cannot be used for sharing documents. However, they can be used to implement awareness techniques at the presentation level.

4. APPLICATION TO REMOTE SHARING

4.1 Real-time sharing

We have developed the Sovigo SVG drawing tool in order to experiment the use of iDOM interactive components and event points. Sovigo's workspace was created using the Inkscape[1] native SVG drawing tools. It is a single SVG document where event points were statically defined[2].

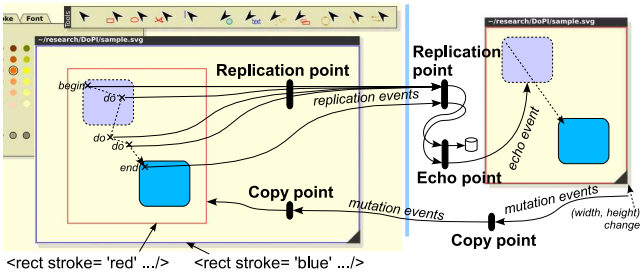

Figure 3: Window content sharing in Sovigo

The real-time sharing of elements is carried out by replication points that implement basic concurrency control algorithms using logical clock and undo-redo [6]. Figure 3 (left side) displays the shared content of the Sovigo's sharing window, which is defined using Inkscape as follows:

```
<g  id='sharing-window'> ...
    <g  id='window-shared-content'>
      <dpi:event-point  class= 'ReplicationPoint'>
        <dpi:integer value='4000'/>
        <dpi:integer value='4001'/>
        <dpi:string value='192.168.10.85 192.168.10.118'/ >
      </dpi:event-point> ...
    </g>
</g>
```

The replication point is defined with the <dpi:event-point> element as an instance of the ReplicationPoint class, with three parameters: the two port numbers used by the replication point, followed by a list of the group members' IP addresses[3]. The first group (<g>) defines the sharing window while the second defines its shared content. By linking the replication point to the second group, the window content becomes shared.

4.2 Echo

The echo technique relaxes the strong temporal coupling of real-time sharing by summarizing all intermediate steps of a long action into a single echo action, which typically consists of an animation [1]. For example, rather than replicating all intermediate translations when moving an object, an animated and straight translation can be played on remote elements (figure 3, right). The *echo point*, used in conjunction with a replication point, enables the echo technique. The following excerpt illustrates the use of an echo point within the sharing window:

[1]http://www.inkscape.org
[2]However, DoPIdom allows the dynamic insertion of event points.
[3]This paper does not address how the addresses are collected.

```
<g id='sharing-window>
    <dpi:event-point class='EchoPoint'/>
    <g id='window-shared-content'>
      <dpi:event-point class='ReplicationPoint'> ...
      </dpi:event-point> ...
    </g>
</g>
```

The echo point listens to *replication events* that are sent by the replication point. Whenever a replication event is received, the action event is extracted from it. If this event is a *begin* or *do action event*, the event is saved in a buffer, then is *captured* (in the sense defined by DOM Events [7]) so the action does not propagate to its target element. If the event is an *end action event*, all previously saved events are sent to the target element through one *echo action event* that encapsulates the saved events. The component linked to this target element consumes the echo action event by providing its echo effect.

4.3 Synchronization

The *synchronization point* alters the synchronization of replication points: synchronization is only performed on demand, so that a user may work temporally in a private context. The following excerpt illustrates how a synchronization point can be defined within the sharing window:

```
<g id='sharing-window>
    ...
    <g id='window-shared-content'>
      <dpi:event-point class='ReplicationPoint'> ...
      </dpi:event-point>
      <dpi:event-point class='SynchronizationPoint'/>
      ...
    </g>
</g>
```

The synchronization point listens to all action events. Whenever a *begin*, *do*, *end*, *undo*, or *redo action event* is received, the event is saved into a buffer that stores *local* actions. The event propagation is then stopped so that the event is not received by any other replication point: the related action thus remains local. Whenever a *replication event* is received, the event is saved into a second buffer that stores *remote* actions, and is captured so that remote actions are not consumed. Finally, when the user requests a synchronization, the events saved in both the local and remote action buffers are sent to their matching target so that they can consume the actions.

4.4 Awareness techniques

Awareness techniques such as *localization* [1] and *telepointing* are fundamental in groupware environments. Figure 3 shows a possible use of localization in the sharing window. Both remote users access the shared contents through their own sharing window. Since their sizes and scrolling locations differ, the first user may not be aware that the second user is also working on the same shared content. The localization technique provides rectangles that locally reflect the area viewed by remote users. The following excerpt illustrates how a *copy point* can handle localization rectangles (here, a red one and a blue one):

```
<g id='sharing-window'>
    <dpi:component class='SharingWindow'>
      <dpi:node path="g/g/rect[@stroke='blue']"/> ...
    </dpi:component>
    ...
    <g id='window-shared-content'>
      <dpi:event-point class='ReplicationPoint'> ...
      </dpi:event-point>
      <g id='localization-rectangles'>
        <dpi:event-point class='CopyPoint'>
          <dpi:integer value='4002'/>
          <dpi:integer value='4003'/>
          <dpi:string value='192.168.10.85  192.168.10.118'/ >
        </dpi:event-point>
        <rect stroke='blue' x='...' y='...' width='...' height='...'/>
        <rect stroke='red' x='...' y='...' width='...' height='...'/>
      </g> ...
    </g>
</g>
```

The SharingWindow component maintains a link with its blue localization rectangle, which is specified by the XPath expression provided by the <dpi:node> element. This component can thus adjust the *x*, *y*, *width* and *height* attributes of its localization rectangle whenever the component is resized or scrolled. Since the localization rectangles of remote sharing windows are all located within the copy point, all their attribute changes are remotely copied.

Telepointing is an other awareness technique used to remotely point at objects in a shared workspace. Telepointers can also be provided through copy points that replicate the location of remote cursors, in a way similar to the localization rectangles.

5. CONCLUSION

This paper presented the concept of event point as a single object dedicated to real-time collaboration, including features such as echo, synchronization, localization, and telepointing. We have implemented event points in the DoPIdom toolkit, and experimented their uses in the Sovigo drawing tool. Since predefined event points can be used directly in XML or DOM documents, flexible sharing is easy to achieve. For example, the original version of Sovigo, which did not provide any groupware feature, required only a few changes to turn it into a groupware tool. In addition, event points can be used independently of DoPIdom provided that the application uses DOM documents and handles the required new DOM event types.

The perspectives of this work follow two directions. We first plan to explore other uses of event points for groupware and for the Web, such as session management, and extend Sovigo so that it becomes a real groupware authoring tool. The second direction is to develop environments centered on XML documents by using the DoPIdom model and toolkit. This will lead to an extension of the DOM recommendation for supporting interaction and collaboration on XML documents.

6. REFERENCES

[1] M. Beaudouin-Lafon and A. Karsenty. Transparency and awareness in a real-time groupware system. In *Proc. of UIST'92*, pages 171–180. ACM Press, 1992.

[2] O. Beaudoux. XML active transformation (eXAcT): Transforming documents within interactive systems. In *Proc. of DocEng'05*, page [in press]. ACM Press, 2005.

[3] O. Beaudoux and M. Beaudouin-Lafon. DPI: A conceptual model based on documents and interaction instruments. In *Proc. of IHM-HCI'01*, pages 247–263. Springer Verlag, 2001.

[4] G. E. Krasner and S. T. Pope. A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80. *Journal of OOP*, pages 26–49, 1988.

[5] J. C. Mong and D. F. Brailsford. Using SVG as the rendering model for structured and graphically complex Web material. In *Proc. of DocEng'03*, pages 88–91. ACM Press, 2003.

[6] A. Prakash. Group editors. In M. Beaudouin-Lafon, editor, *Computer Supported Cooperative Work*, volume 7 of *Trends in Software Series*, pages 103–133. John Wiley & Sons, 1999.

[7] W3C. Document Object Model level 3 events specification. Normative recommendation, W3C, 2003.