

The Open Overlays Collaborative Workspace Environment

Chris Cooper¹, David Duce¹, Muhammad Younas¹, Wei Li¹, Musbah Sagar¹, Gordon Blair²,

Geoff Coulson², Paul Grace²

¹Oxford Brookes University

²Lancaster University

Abstract

Next-generation Grid applications will operate within and across many heterogeneous network types, will employ a wide range of device types ranging from supercomputers to sensor motes; and will require many more “interaction paradigms” than merely RPC and message-passing. The Open Overlays project proposes a middleware approach to satisfy these emerging needs. In this paper we describe an application scenario, wildfire management, that encompasses these needs, and our approach to constructing a component of this scenario, a collaborative workspace tool (svgCWE). This is being constructed using Web technologies: SVG and RDF. Information in the collaborative workspace is regarded as an annotation of the workspace resource which can be described with an RDF model. The architecture of svgCWE is designed so that a variety of different kinds of RDF repository (centralized, replicated and distributed) can be used. The paper describes the first prototype of svgCWE and draws some tentative conclusions about the advantages of the approach.

1. Introduction

This paper describes a collaborative workspace tool being developed at Oxford Brookes University as part of the Open Overlays project in the Fundamental Computer Science for e-Science programme. This project is undertaking fundamental research relating to advanced middleware and advanced networking — and, crucially, exploring an approach that integrates these two key areas. The approach integrates a component oriented middleware platform oriented toward next-generation Gridware with an extensible set of interaction types and advanced network services, and on an architectural framework for the internals of future Grid middleware.

There is a trend towards diversity in both end-systems and networked infrastructures. The spectrum of end-systems includes high performance supercomputers, desktop, laptop, PDA and miniature sensor devices. Networked infrastructure includes high speed LAN-based systems, lower speed WANs, infrastructure-based wireless networks, and *ad hoc* wireless networks. The latter range from relatively static to highly dynamic configurations.

The range of “interaction paradigms” used at the application level has also increased. Beginning with point-to-point interactions (e.g. RPC and SOAP messaging), the range of interaction paradigms is expanding to include, for example, reliable and unreliable multicast, media streaming, publish-subscribe, tuple-space

communication, and peer-to-peer based resource location or file sharing.

The project seeks to provide a Grid middleware infrastructure that can span and integrate the kinds of diversity just described. The essence of the approach is to place a flexible and configurable set of middleware frameworks over a layer of *overlay networks* and to construct the whole architecture in terms of a lightweight component model that can be implemented on a wide range of device types.

In order to exercise the middleware concepts, we are developing application scenarios. The first scenario is based on wildfire management. Section 2 describes the application scenario and section 3 describes the approach taken to providing a collaboration tool for this scenario. Section 4 discusses related work and section 5 discusses the achievement so far and future work.

2. The Application Scenario

We are considering (with advice from environmental scientists/geographers at Royal Holloway University of London and The Open University) a scenario based on a remote region with poor accessibility. Fire fighters in this scenario have very limited means at their disposal: aerial attack is not possible, the main instruments for fire-fighting are hand beaters and pre-cut fire breaks (which in general will have become overgrown). The fire-fighters have little idea where the fire boundary is and there is

no communication between different groups of fighters.

In the scenario, we posit a number of advances. Fire fighters carry PDA-like devices which can present data, video and audio information; these devices enable communication with other fire fighters and those directing operations (controllers). The locations of these devices, hence the locations of fire fighters are known.

We also posit the availability of sensors to provide information for controllers: in particular mobile cameras (for example attached to the PDAs carried by fire fighters) to give controllers a view of the fire, but more importantly sensors for environmental conditions. The most important factor is wind velocity; magnitude and direction; though rainfall and moisture content of the combustible materials are also important parameters in determining how fire spreads. These sensors are assumed to be portable; more may be brought into play as necessary and they may be destroyed by the fire. They are assumed to communicate through radio, i.e. data gathering does not require human intervention. The locations of the sensors are assumed to be known. Portable environmental sensors are available now commercially (though not packaged in the way a full realisation of this scenario would require). Research in sensor networks is addressing questions such as self-organisation of *ad hoc* networks of sensors; questions that the Open Overlays middleware infrastructure is also addressing. Hence it is valuable to have a sensor component in the scenario.

The scenario also assumes the availability of simulations to predict the spread of the fire. There are a number of wildfire simulators available, for example Farsite (Faresite, 2005). Fire simulators in general are resource intensive and this gives a fixed Grid component to the scenario.

The scenario contains a number of different types of actors:

- Controllers (mobile and fixed). We assume peer-to-peer rather than hierarchical relationships between controllers. Multiple controllers could be involved if the fire is large or crosses administrative boundaries. Controllers may have fixed or mobile locations.
- Field Worker (mobile). Field workers will have PDAs and their locations will be known. Field workers may split into different kinds of groups: for example, those

charged with fire-fighting and those charged with deploying sensors.

- Sensors (mobile - portable).
 - Environmental sensors: wind velocity - magnitude and direction. Currently these are valuable resources and hence field workers would move them in response to changing conditions.
 - Video cameras generating video streams. Video cameras may be portable (on the top of portable masts - moved around the site by field workers) or wearable (e.g. miner's helmet style).

The scenario contains a rich variety of types of communication:

- Audio: Telephone-like audio communication between field workers/controllers using PDA-like devices or computers.
- Video: Low rate video footage is exchanged between field workers and controllers.
- Location Information: Location of field workers (their PDAs); location of sensors (GPS information). Sensor Information: Video (see above), environmental conditions, wind velocity.
- Collaborative Workspace Environment: to support sharing ideas, issuing commands and helping decision making.

The scenario is completed by conceiving the fire-fighting groups and controllers as forming dynamic, mobile Grids, and the simulation facilities being provided by a dynamic Grid embedded in a wider fixed Grid infrastructure.

The scenario as described involves highly heterogeneous device and networking technologies, and it calls for a wide range of interaction paradigms, for example reliable *ad hoc* multicast for command propagation, stream-based multicast for group audio communication, publish-subscribe for sensor data collection, SOAP-based messaging for communication with objects in the fixed Grid, etc.

Figure 1 shows a mock-up of the kind of display that might be presented to a controller, showing the locations of controllers, field workers and sensors, overlaid on a map of the affected region.

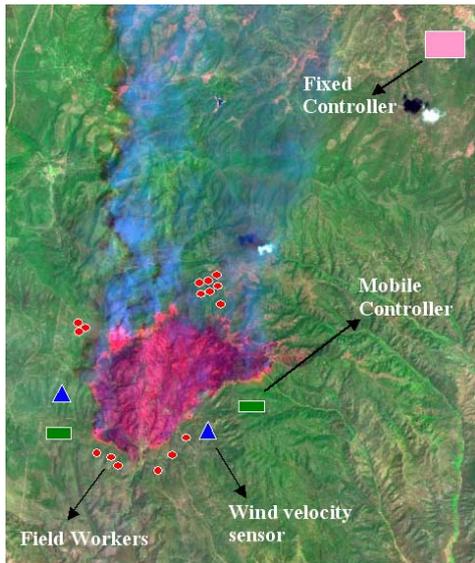


Figure 1: Mock-up of a controller's display

A key requirement is the ability to share such visual presentations between the human actors in the scenario, the controllers and field workers. This is the function of the collaborative workspace tool (svgCWE) that is the focus of this paper. svgCWE provides the basis for graphical communication between controllers, controllers and field workers, and field worker to field worker. svgCWE is used to present maps overlaid with visualizations of sensor information (including positions of actors), and output from simulations. Controllers and field workers need to be able to sketch on the drawing surface, for example to give an estimate of the local fire boundary, to highlight particular features, or to suggest changes to how resources are deployed.

In addition to the collaborative workspace tool, the scenario includes audio and video tools which will be assembled from existing components.

3. svgCWE Architecture and Implementation

The collaborative workspace tool is being developed using a variety of Web technologies, the main ones being Scalable Vector Graphics (SVG) and the Resource Description Framework (RDF). SVG is an XML application (markup language) for 2D graphics, providing a rich set of 2D drawing primitives and attributes, transformations and animation capabilities. RDF is essentially a data model. The basic building block of RDF is a statement asserting that a resource (the *subject*) has a given property

(the *predicate*) with a given value (the *object*). It is one of the Web technologies at the core of the so-called Semantic Web (Antoniou and van Harmelen, 2004). RDF makes no assumptions about the domain to which it is applied. The vocabulary used in RDF data models can be expressed in a schema language called RDF Schema (RDFS) or in more comprehensive languages such as the Web Ontology Language (OWL). Query languages for RDF data models are emerging.

SVG is used to represent graphical information for display in the workspace, which may be provided by any of the actors using the workspace and by applications. The design relies on one key idea: the notion that all information in the workspace is an annotation of the workspace that can be represented as an RDF triple stored in an RDF repository. Appropriate information for each actor (e.g. field worker or controller) can be displayed in the actor's workspace (effectively the result of a query of the triple store) and the content of the workspace can be replayed.

A snapshot of the first version of svgCWE is shown in figure 2. This shows a background map with sketched annotations highlighting particular features.

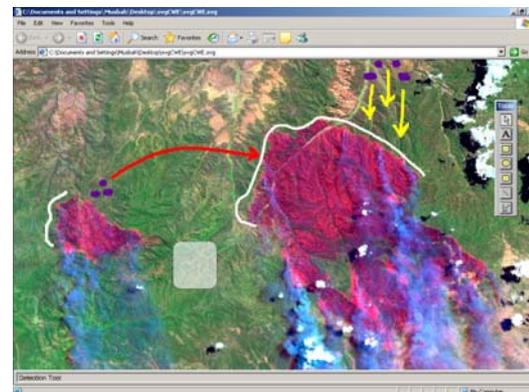


Figure 2: Snapshot of svgCWE

3.1 The RDF data model

The initial design of svgCWE associates a single workspace with a single group. Within a workspace, information is structured as a set of contexts. A context consists of background information provided by the application (for example a map, or the output of a simulation), and sketch annotation created by the members of the group.

The essence of the approach taken is that the workspace of a collaborative group is considered to be a web resource. Each member of the group has their own view of the

workspace. Members can add display information in their view which is then transmitted to the views of the other members of the group.

RDF is used to describe both the content and history of a workspace, the latter to facilitate replay. The structure of the RDF model is evolving. Figure 3 shows the RDF class structure used in the initial prototype.

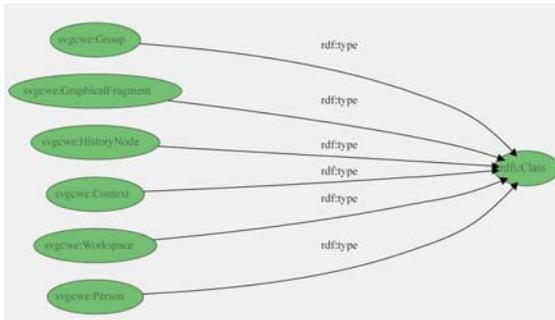


Figure 3: RDF class Structure

Classes are defined to represent persons, groups, workspaces, contexts within workspaces, history nodes (which record changes to the content of the workspace), and graphical fragments (elements of graphical content within the workspace). Properties are defined to relate, for example, graphical fragment to history node, history node to context, and context to workspace. For further details of this structure see Cooper *et al.*, 2005.

Whenever a change is made to the content of the workspace, for example by a member sketching on the workspace, the change is recorded as a history node object that is linked by a history-of assertion to the appropriate context. An RDF/XML representation of a history node is shown below.

```
<svgcwe:HistoryNode>
  <svgcwe:svgFragment
    rdf:parseType="Literal"
    xmlns:svgcwe="...">
  </svgcwe:svgFragment>
  <svgcwe:fillColor>turquoise
  </svgcwe:fillColor>
  <geom2d:x>25</geom2d:x>
  <geom2d:width>300</geom2d:width>
  <geom2d:y>50</geom2d:y>
  <geom2d:height>400</geom2d:height>
  <svgcwe:timestamp>2005-01-28T20:00:00Z
  </svgcwe:timestamp>
  <svgcwe:history-of rdf:nodeID="A0"/>
  <dc:creator rdf:resource=
    "http://svgcwe/person/1"/>
</svgcwe:HistoryNode>
```

The history node contains assertions about the graphical content of the sketch (in fact the object of the assertion is a piece of SVG markup that represents the sketch), assertions about the geometrical properties of the sketch, the creator of the sketch and a timestamp.

Geometrical properties are expressed using an RDF vocabulary called RDFGeom (Goad, 2004) which is closely related to the way geometry is expressed in SVG.

RDF content can be retrieved using an RDF query language such as RDQL. For example a query to find the timestamp, type, and geometric attributes of all the history nodes created by Fred Smith which make assertions about the workspace identified by the URI <http://svgcwe/ws/1> could be written as follows.

```
SELECT ?date, ?t, ?x, ?y, ?w, ?h
WHERE
  (?c svgcwe:context-of
    <http://svgcwe/ws/1>)
  (?geom svgcwe:content-of ?c)
  (?geom rdf:type ?t)
  (?g svgcwe:history-of ?geom)
  (?g geom2d:x ?x)
  (?g geom2d:y ?y)
  (?g geom2d:width ?w)
  (?g geom2d:height ?h)
  (?g dc:creator ?person)
  (?g svgcwe:timestamp ?date)
  (?person foaf:name "Fred Smith")
USING ....
```

The SELECT statement identifies the variables to return. The WHERE clause describes the graph patterns to match. Variable names are prefixed by '?'. Where the same variable name is used in different patterns, the value of the variable must be the same in each triple pattern for a successful match. The USING statement expands the namespace abbreviations (svgcwe etc.) and has been omitted.

The approach is extensible. For example we could easily add assertions about the location of each member of a group and then search for history nodes created by members within a particular region. For example the query:

```

SELECT ?xloc ?yloc ?date ?t, ?x, ?y, ?w, ?h
WHERE
  (?c svgcwe:context-of
   <http://svgcwe/ws/1>)
  (?geom svgcwe:content-of ?c)
  (?geom rdf:type ?t)
  (?g svgcwe:history-of ?geom)
  (?g geom2d:x ?x)
  (?g geom2d:y ?y)
  (?g geom2d:width ?w)
  (?g geom2d:height ?h)
  (?g svgcwe:timestamp ?date)
  (?g dc:creator ?p)
  (?p svgcwe:location-x ?xloc)
  (?p svgcwe:location-y ?yloc)
AND
  ?xloc >= 200 && ?xloc <= 300 &&
  ?yloc >= 200 && ?yloc <= 400

```

would search for history nodes created by any member located in the region 200 to 300 in x and 200 to 400 in y. The AND clause filters tuples that satisfy the WHERE clause based on the values of the location-x and location-y properties.

The development of the RDF data model has been based on the Jena RDF toolkit (Jena, 2005) which supports the RDQL query language. W3C are standardising an RDF query language called SPARQL and this may be used in later work.

In Cooper *et al.* (2005) we discuss how this approach extends to dealing with application level queries such as “show all the firebreaks in a given region”. It is clear that other kinds of sensor data, not just location information, can also fit into this general structure.

3.2 The Gridkit architecture

The architecture of svgCWE fits into the context defined by the Open Overlays architecture called Gridkit (Grace *et al.*, 2004; Coulson *et al.*, 2005). The Gridkit architecture is illustrated in figure 4. The following description focuses on the important points for the svgCWE architecture description in section 3.3.

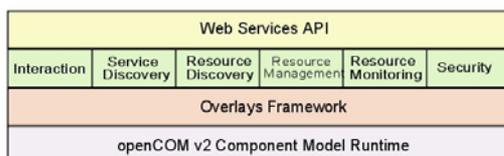


Figure 4: The Gridkit architecture

Gridkit is built in terms of a component model called OpenCOM v2, which employs a

minimal runtime environment to support the loading and binding of lightweight software components. The layer above OpenCOM is a framework to support the deployment of overlay networks. The vertical frameworks above this provide functionality in a number of areas. The top layer provides XML/SOAP/WSDL-based APIs to the underlying frameworks.

The OpenCOM component model is shown in figure 5.

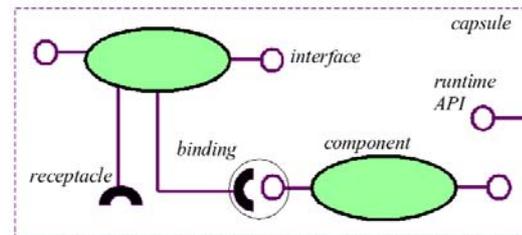


Figure 5: The OpenCOM component model

Components are language-independent encapsulated units of functionality and deployment that interact with other components exclusively through *interfaces* and *receptacles*.

For the svgCWE architecture description in section 3.3, it is the notions of receptacle, interface and *binding* that are important. Interfaces are expressed in terms of sets of operation signatures and associated datatypes. Components can support multiple interfaces: this is useful in embodying separations of concern (e.g. between base functionality and component management). Receptacles are *required* interfaces that are used to make explicit the dependencies of a component on other components. Finally, bindings are associations between a single interface and a single receptacle.

3.3 The svgCWE architecture

The current svgCWE architecture is shown in figure 6. svgCWE is split into two components, the client and the communication manager. The former is responsible for the user interface and presentation, the latter handles communication between participants. Each participant has their own copy of the client and communication manager components. The points to note are the bindings between the svgCWE client and communication component, and between the communications component and RDF repository and group abstraction interface over Gridkit. The group abstraction receptacle and RDF repository receptacle ensure that the svgCWE components are agnostic to the kind of overlay used to realize group communication

and the kind of RDF repository used (e.g. centralized, replicated, or distributed).

The svgCWE client is based on HotDraw (Johnson, 1992), a framework for structured drawing editors, originally developed in SmallTalk, but later ported to Java (Java HotDraw, 2005). A subset of the code has been ported to JavaScript to provide the foundation for svgCWE. We have developed an extensive set of JavaScript libraries which implement appropriate parts of the Java Foundation Classes and JavaSwing classes in order to provide GUI support (Sagar *et al.*, 2005).

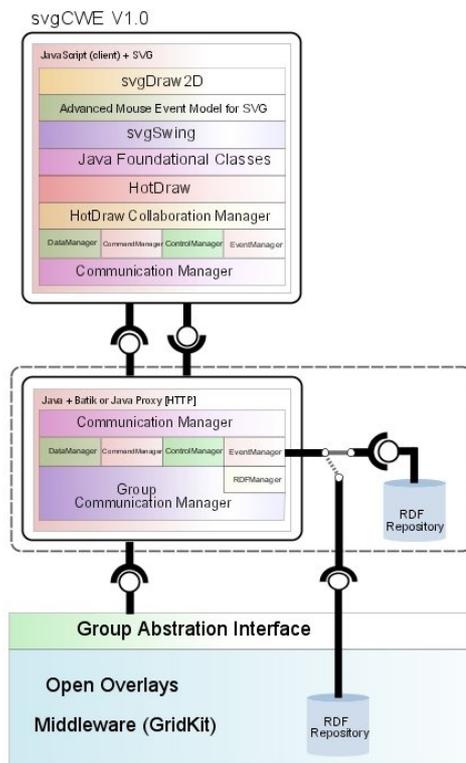


Figure 6: The svgCWE architecture

The svgCWE client can run either within a Web browser with appropriate SVG and JavaScript support, or as a Java component using the Batik SVG Java engine (which includes JavaScript support).

Communication between Java and JavaScript is achieved through a communication manager class which is implemented in both languages and supports a simple transport interface with send and receive methods. When svgCWE is running in a Web browser the communication is mediated through a Web server proxy.

The group interaction interface facilitates dissemination of updates to relevant actors. The initial development has used a centralised RDF

repository, each member of a group having their own copy of the repository. We are also considering both a centralised repository and a locally developed distributed repository, based on distributed hash tables, inspired by RDFPeers (Cai and Frank, 2004). The middleware framework should allow the shared workspace to move seamlessly at run-time from a centralised implementation of the RDF repository to a distributed implementation.

4. Related Work

Existing commercial products that support on-line collaboration such as an on-line meeting, typically use a client server architecture, with multiple clients accessing a single server. See Hassler (2004) for a recent review of commercial products. Such products typically provide audio/video conferencing, a shared whiteboard, and possibly application sharing. Architectures without a centralized server have been used for shared whiteboards, for example, wb (Floyd *et al.*, 1997) which used a reliable multicast protocol to transmit the streams of timestamped drawing operations generated by each member to the other members. Earlier systems seem to have made a commitment to a particular distribution architecture, in the svgCWE/Open Overlays approach, we are exploring the possibility of plugging in either a centralized, replicated or distributed repository, allowing the architecture to be tailored to particular circumstances.

The svgCWE approach builds on the authors' earlier work in distributed and collaborative visualization. For a review of this area see Brodli *et al.* (2004). Distributed collaborative visualization can be regarded as an example of application sharing. A particular concern is the level at which information is shared, for example as a rendered bitmap (using technologies such as VNC), as geometry or as application data (for example using a modular visualization environment such as NAG's IRIS Explorer). The svgCWE approach shares geometry, though since the geometry and application level data are regarded as resources, RDF statements can describe the relationships between them. From experience gained so far, this is a flexible and extensible approach, not least because the annotation is not "hard-wired" to the underlying resources, so different views of the application data can be built using different relationships. This is a strength of the RDF-based approach.

W3C's Annotea system (Kahan *et al.*, 2001) used RDF to model annotations of documents as

a class of metadata. Annotations were viewed as statements made by an author about a document and could be stored externally to the document in one or more annotation servers. Users could query the annotation server to retrieve, modify or delete existing annotations and add new annotations. The Annotea approach influenced our early thinking in the development of svgCWE.

Turning to our use of SVG, Qiu, Carpenter and Fox (2003) describe a shared SVG browser. They describe the decomposition of the Batik browser based on the Model-View-Controller (MVC) paradigm. The view corresponds to the user interface and the model to a Web service interface. The model and view are linked by a publish/subscribe messaging system called NaradaBrokering. The preparation and interpretation of messages together with the messaging system correspond to the controller component. Collaboration is supported by replicating Web services and delivering events generated on a master view client to all instances of the model, which then service the associated view components. They also describe an alternative approach which is to use NaradaBrokering to multicast the messages from a single model instance to all collaborating view components. The svgCWE approach differs from this work in two respects: firstly the Open Overlays middleware infrastructure is very different to the NaradaBrokering infrastructure; secondly Qiu *et al.* were not exploring the use of RDF in their work.

SVG is gaining in popularity, not least in the mobile phone arena where many products are using SVG as the base rendering layer. The svgCWE approach might extend to such devices, if not immediately, then in the future as device performance and capabilities improve.

5. Status and Future Work

A first version of the svgCWE tool has been implemented and a first integration with Gridkit has been achieved. The integration with Gridkit is described in detail in Coulson *et al.* (2005). The HotDraw functionality required for svgCWE is functioning. At present each member of the workspace has their own local RDF repository (a replicated architecture). RDQL has been used experimentally to retrieve information from the RDF repository, but this has been done to try out RDF model structures and is not yet integrated with svgCWE. Collaborative sessions can be replayed from the RDF repositories in the prototype, but this is done by traversal of the triple store rather than

by querying. Based on the experience so far, we are in the process of revising the RDF data model to incorporate application level concepts in a more systematic way than was done in the first approach described in this paper.

Future plans for svgCWE include experimenting with different kinds of RDF repositories, including DHT repositories, based on overlay plugins that have been developed in Gridkit.

We perceive the following benefits in our approach:

- SVG provides a convenient presentation environment for 2D graphics that is device-independent and can be built into stand-alone or Web based clients.
- RDF appears to provide a framework for managing the display of diverse kinds of information in a collaborative setting, where there will be relationships between the roles that actors take and the kinds of information that they require.
- RDF also provides a framework for expressing relationships between graphical and other kinds of information.
- Use of RDF also opens up the possibility for richer ways of managing graphical presentations in the future, for example through ontology-based semantic markup.

References

- Antoniou, G., van Harmelen, F. (2004). *A Semantic Web Primer*, The MIT Press. ISBN 0-262-01210-3.
- Brodie, K.W., Duce, D.A., Gallop, J.R., Walton, J.P.R.B, Wood, J.D. (2004). "Distributed and Collaborative Visualization", *Computer Graphics Forum*, **23**(2), pp. 223-251.
- Cooper, C, Duce, D., Younas, M., Li, W., Sagar, M., Blair, G., Coulson, G., Grace, P. (2005). "The Open Overlays Collaborative Workspace", *SVG Open 2005*. Proceedings available at <http://www.svgopen.org/2005/>
- Coulson, G., Grace, P., Blair, G.S., Porter, B., Cai, W., Cooper, C., Duce, D., Younas, M., Sagar, M., Li, J. (2005). "Open Overlay Support for the Divergent Grid" to appear in *Proceedings of AHM 2005*.

<http://www.allhands.org.uk/>

FARSITE (2005), Fire Area Simulator, <http://www.farsite.org/> [accessed 24 June 2005]

Floyd, S., Jacobson, V., Liu, C.-G., McCanne, S., Zhang, L. (1997). "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", *ACM Transactions on Networking*, **5**(6), pp. 784-803.

Goad, C. (2004), *RDFGeom (RDF Geometry Vocabulary)*, <http://fabl.net/lib/geometry/1.1> November 24, 2004 [accessed 24 June 2005]

Grace, P., Coulson, G., Blair, G., Mathy, L., Yeung, W.K., Cai, W., Duce, D., Cooper, C. (2004). "GRIDKIT: Pluggable Overlay Networks for Grid Computing", in *Proceedings of Distributed Objects and Applications (DOA'04)*, Lecture Notes in Computer Science, 3291, Springer-Verlag.

Hessler, V. (2004). "Online Collaboration Products", *IEEE Computer*, **37**(11), pp. 106-109.

Java HotDraw (2005), *HotDraw*, <http://c2.com/cgi/wiki?HotDraw> [accessed 24 June 2005]

Jena (2005). *Jena Semantic Web Framework*, <http://jena.sourceforge.net/>

Johnson, R.E. (1992), "Documenting Frameworks using Patterns", *OOPSLA '92*, pp. 63-76, also in *ACM SIGPLAN Notices* **27**(10), pp. 63-76.

Kahan, J., Koivunen, M-R., Prud'Hommeaux, E., Swick, R.R. (2001). "Annotea: An Open RDF Infrastructure for Shared Web Annotations", *WWW 10*, ACM Press, 2001, pp. 623-632, ISBN 1-58113-348-0.

Cai, M., Frank, F. (2004). "RDFPeers: A Scalable Distributed RDF Repository based on a Structured Peer-to-Peer Network", *WWW 2004*, pp. 650-657, ACM Press.

Qiu, X., Carpenter, B., and Fox, G.C. (2003). "Collaborative SVG as A Web Service", *SVG Open 2003*.
<http://www.svgopen.org/2003/paperAbstracts/CollaborativeSVGasAWebService.html>
[accessed 4 February 2005]

Sagar, M., Cooper, C.S., Duce, D.A. (2005). "Advanced Mouse Event Model for SVG", *SVG Open 2005*. Proceedings available at <http://www.svgopen.org/2005/>