

A Distributed Platform based on Web Services for Multimedia Resource Discovery

Sabin C. Buraga
Faculty of Computer Science
“A.I. Cuza” University of Iași
Berthelot, 16 – Iași, Romania
busaco@infoiasi.ro

Petrică Găbureanu
Faculty of Computer Science
“A.I. Cuza” University of Iași
Berthelot, 16 – Iași, Romania
peter@infoiasi.ro

Abstract

The paper describes ITW – a multi-platform and multi-language architecture used to discover (multimedia) resources. The ITW system is based on Web services and software agents, exploiting the relations between Web sites' resources. To represent these relations, a high-level RDF model is adopted, using Interval Temporal Logic.

1. Introduction

The actual World-Wide Web space is primarily composed on pages (markup documents) with information in the form of natural language text and multimedia intended for humans to read and understand. Computers are used mainly to render this hypermedia information. Information retrieval has become ubiquitous with the WWW's development and information needs no longer to be intended for human readers only, but also for machine processing, enabling intelligent information services, personalized Web sites, and semantically empowered search engines – this is the seminal idea of the *Semantic Web* [15, 31].

Agent paradigm is one of the promising technologies for information retrieval in general and for multimedia resource discovery in particular. Another emerging technology is denoted by *Web services*, collections of operations that are network-accessible through standardized XML messaging [31].

The paper will present *ITW* – a multi-language distributed platform used for multimedia resource discovery, using a hybrid architecture composed from software agents, Web services, and other entities such as CGI scripts. The ITW system exploits the temporal relations established between Web sites' resources and uses a RDF/XML-based model for semantic representation of metadata and additional information that involves time.

2. ITL Model

2.1. Background

One of the most important problems of any distributed system (especially, the World-Wide Web) is the representation of time. According to Allen [2], different representations of time are usable depending on the assumptions about the temporal information to be represented.

There are many temporal formal models in order to capture different properties of the distributed real-time systems (such as database management systems, synchronized multimedia systems, or WWW space). In 1977, Pnueli [23] introduced a simple and elegant temporal logic for reasoning about concurrent programs. Over the years, this approach is followed – among others – by *Linear-time Temporal Logic (LTL)* [18, 22], *Interval Temporal Logic (ITL)* [1, 4], *Timed Propositional Temporal Logic (TPTL)* [5, 6], *Temporal Description Logic (TDL)* [7], *Temporal Logic of Actions (TLA)* [19] or branching time temporal logic *Computation Tree Logic (CTL)* [16]. All of these proposals are mainly focused on theoretical aspects of modeling distributed systems and their components. It is still difficult to give a machine-understandable description of these formal methods in order to implement different software applications (e.g., Web agents, Web mediators or Web services) for activities such as resource discovery and queries that involve time.

2.2. Short Description of ITL

The temporal structure introduced by *Interval Temporal Logic (ITL)* is a simple linear model of time introduced by Allen [2, 1, 4]. Given any two temporal intervals, there are thirteen mutually exclusive relationships. To model these relations, one primitive relation *Meets* is introduced. Intuitively, two periods m and n meet if and only if m precedes

n , yet there is no time between m and n , and m and n do not overlap. The axiomatization of time periods is presented in figure 1, where i, j, k, l , and m are logical variables restricted to time periods.

In the ITL model, every period of time has a period that meets it and another that it meets:

$$\forall i \exists j, k : Meets(j, i) \wedge Meets(i, k) \quad (1)$$

Time periods can compose to produce a larger period. For any two periods that meet, there is another period that is the “concatenation” of them. We are able to write $j + k$ to denote the interval that is the concatenation of intervals j and k .

$$\begin{aligned} \forall i, j, k, l : & Meets(i, j) \wedge Meets(j, k) \\ & \wedge Meets(k, l) \supset \\ & \exists m : Meets(i, m) \wedge Meets(m, l) \quad (2) \end{aligned}$$

Periods of time define an *equivalence class of periods* that meet them. Particularly, if i meets j and i meets k , then any period l that meets j must also meet k :

$$\begin{aligned} \forall i, j, k, l : & Meets(i, j) \wedge Meets(i, k) \\ & \wedge Meets(l, j) \supset Meets(l, k) \quad (3) \end{aligned}$$

These equivalence classes also uniquely define the periods. If two periods both meet the same period, and another period meets both of them, then the periods are equal:

$$\begin{aligned} \forall i, j, k, l : & Meets(k, i) \wedge Meets(k, j) \\ & \wedge Meets(i, l) \wedge (j, l) \supset i = j \quad (4) \end{aligned}$$

We can introduce now an *ordering axiom*. Intuitively, this axiom asserts that for any two pairs of periods, such that i meets j and k meets l , then either they both meet at the same “place”, or the place where i meets j precedes the place where k meets l , or vice versa. In terms of meet relation, this can be axiomatized as follows (the symbol \otimes means “exclusive-or”):

$$\begin{aligned} \forall i, j, k, l : & (Meets(i, j) \wedge Meets(k, l)) \supset \\ & Meets(i, l) \otimes (\exists m : Meets(k, m) \wedge \\ & Meets(m, j)) \otimes \\ & (\exists m : Meets(i, m) \vee Meets(m, l)) \quad (5) \end{aligned}$$

It can be proved that no period can meet itself, and that if one period i meets another j then j can not also meet i (finite circular models of time are not possible).

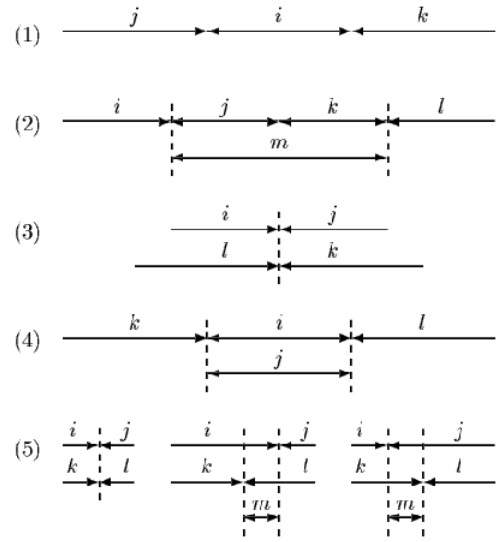


Figure 1. Axiomatization of time periods

With this system, we can define the complete range of the thirteen intuitive relationships that could hold between time periods. For example, one period is before another if there exists another period that spans the time between them, for instance:

$$Before(i, j) \equiv \exists m : Meets(i, m) \wedge Meets(m, j) \quad (6)$$

Figure 2 illustrates each of these relationships (the equality relation is not shown). We can use the following symbols to stand for commonly-used relations and disjunctions of relations [4]:

$$\begin{aligned} Meets(i, j) & i : j \\ Before(i, j) & i \prec j \\ During(i, j) & i \sqsubset j \\ Before(i, j) \vee Meets(i, j) & i \prec\!:\! j \\ During(i, j) \vee i = j & i \sqsubseteq j \end{aligned}$$

Between two periods of time, a *Disjoint* relation can be established: two intervals are *disjoint* if they do not overlap in any way. We can write this as “ $i \bowtie j$ ” and define it by $i \bowtie j \equiv i \prec\!:\! j \vee j \prec\!:\! i$.

A period can be classified by the relationships that it can have with other periods of time. A period that has no sub-periods can be called a *moment* and a period that has sub-periods, an *interval*. Also, we can define a notion of *time point* by a construction that specifies the beginning and ending of periods (moments and points are distinct). A *discrete time model* can be given, where periods map to pairs of integers $\langle I, J \rangle$, where $I < J$. Moments correspond to pairs

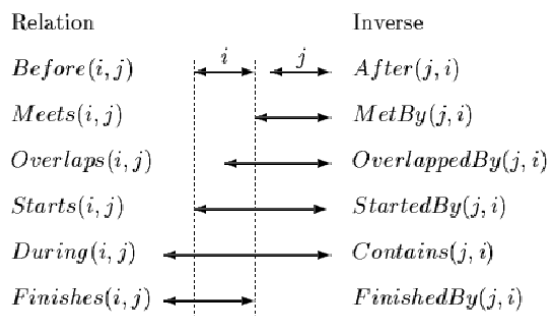


Figure 2. The possible relations between time periods (equality not shown) [3]

of the form $\langle I, I + 1 \rangle$, and points correspond to the integers themselves. A similar model build out of pairs of real numbers does not allow moments.

The computational properties of the interval calculus and algorithms for maintaining networks of temporal constraints are presented in [1, 2, 25]. Also, a formal representation of events and actions based on interval temporal logic is presented in [3] and a qualitative theory of motion based on spatio-temporal primitives is described in [24].

3. Expressing Temporal Relations in RDF

In this section, we'll give a Resource Description Framework (RDF) [20] model to express the temporal relations established between resources. RDF is a standardized basis for processing metadata. The used metadata format should allow us to reason about data. The RDF is intended to be used to capture and express the conceptual structure of information offered in the Web. The RDF assertions can be considered as a data model for describing machine processable semantics of data (including time) to build the infrastructure for Berners-Lee's *Semantic Web* [15, 31].

3.1. RDF Model

RDF consists of a model for the representation of named properties and property values. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources and therefore a RDF model can resemble an entity-relationship diagram.

To facilitate the definition of metadata, RDF is based on *classes*. A collection of classes, typically designed for a specific purpose or domain, is called a *schema* [9]. Through the sharability of schemas, RDF supports the reusability of metadata definitions. The RDF schemas may themselves be written in RDF.

The basic model of RDF consists of three object types:

resources All objects being described by RDF expressions are called *resources* and they are always named by Uniform Resource Identifiers (URI) [31] plus optional anchor identifiers. Using URI schemas (i.e. `http`, `ftp`, or `file` schemas), every type of resource can be uniformly identified.

properties A *property* is a specific aspect, characteristic, attribute, or relation to describe a resource, as stated in [20]. Each property has a specific meaning, defines its permitted values, the type of resources it can specify, and its relationship with other properties (via a RDF Schema).

statements A specific resource together with a named property, plus the value of that property for that resource is an RDF *statement*. These three individual parts of a statement are called, respectively, the *subject*, the *predicate*, and the *object*. The object of a statement (e.g., the property value) can be another resource or a literal.

RDF also specifies three types of container objects:

- *Bag* – an unordered list of resources or literals,
- *Sequence* – an ordered list of resources or literals,
- *Alternative* – a list of resources or literals that represent alternatives for the single value of a property.

The collections can be used instead of *Description* elements. The containers may be defined by a URI pattern. RDF can also be used to make statements about other RDF statements (higher-order statements).

The RDF data model provides an abstract, conceptual framework for defining and using metadata. Currently, there are several proposals of model-theoretic semantics for RDF and RDF Schema [14, 15, 31]. The concrete RDF syntax is based on XML (the grammar of the RDF/XML constructs can be found in [20]).

3.2. Temporal Relation Specification Language

To model different temporal relations as RDF assertions, an XML-based language – *Temporal Relation Specification Language (TRSL)* [12] – is defined to specify the time model presented in section 2. This language will express the relations *Before*, *Meets*, *Overlaps*, *Starts*, *During*, and *Finishes* that can be established between Web resources. These temporal relationships can help to determine the dynamics of that Web sites' content and can be used, among others, by Web agents in mirroring/discovering activities. The updating or querying actions can depend on the

temporal relations that can be expressed by TRSL and RDF constructs. The TRSL language will allow intervals of time (beginning and ending of periods) and, of course, moments.

For each time relation, TRSL offers an element that corresponds to a specific relation (e.g. `<Meets>` element for *Meets* relation). The beginning and ending of time periods are denoted by `begin` and, respectively, `end` attributes that are inspired by Synchronized Multimedia Integration Language (SMIL) [8]. Also, TRSL defines the `dur` attribute for specifying a known or predictive time period (this will allow Web agents to reason about different actions that may need to be performed). The source and destination (viewed as operands) of a temporal relation can be expressed by RDF constructs, too.

The syntax (specified by an XML schema) and the semantics of our defined TRSL language are detailed in [12].

4. ITW Architecture

The key idea in ITW's design rationale process is to offer a flexible multi-language and multi-platform architecture for (multimedia) resource discovery based on *semantic information* associated with the resources.

In the discovery process, one of the important issues is to deal with time. Using the RDF-based model presented in section 3, the software agents involved in the process of resource discovery will be able to reason about the spatio-temporal relations established between certain resources localized in different Web sites. Of course, our approach can be applied to (semantic) service discovery, in the context of Grid computing [17].

The general architecture of the proposed system consists of two main entities:

- *ITW agents* – their role is to effectively discover distributed multimedia resources stored on different sites; these agents are intended to be implemented within a multi-agent system, such as *Omega* [11];
- *ITW Web services* – their role is to give desired information about the resources and the access to these resources; the Web services can be invoked by different software applications (e.g., user agents – such as Web browsers or special clients –, other agents or Web services).

Actually, our interest is focused on the process of designing and implementing the ITW Web services. From this point of view, the idea is to offer a heterogeneous interoperable infrastructure based on Web services. Through a Web portal-like interface, the user will be able to formulate complex queries that involve time. The information and the associated RDF metadata offered by ITW system

will be stored on independent servers. Even if one server is shutdown, the system will be able to continue its execution, providing the same capabilities. The system is flexible enough to interrogate – “on-the-fly” – newly plugged-in running services.

4.1. ITW Web Services

The Web services part of the application consists of the following distributed components:

- one Web interface based on Extensible Markup Language (XUL) [21] to provide a flexible query user-interface;
- a number of m local Web services to provide information about the resources stored on the local Web (i.e. the intranet or the public Web site of an organization);
- a number of n external Web services, developed by third-party organizations (e.g., Google's public Web service – see section 4.2).

The physical localization and execution of the $(m + n)$ Web services are transparent for the final user. These Web services can be considered as one entity (very similar with a computational Grid [17]) by the ITW system.

If the IP addresses of the local Web services are fixed, then these services can be independently invoked by other external entities. Of course, the information exchanged by Web services and their clients is stored as Simple Object Access Protocol (SOAP) messages [31]. The ITW system provides the Web Services Description Language (WSDL) [31] descriptions of the local designed Web services and uses the available WSDL documents in order to invoke external Web services.

The information and metadata regarding the involved multimedia resources are stored by RDF documents. Associated metadata and localization of the resources are written as *XFiles* [10, 11] documents. Temporal information concerning the relations between resources is stored by TRSL files and RDF assertions (see section 3).

In the future versions of the system, these documents will be automatically generated by the ITW agents. In the present, a simple Web robot regularly runs to update this information about the local multimedia resources of a certain Web site.

Example

An example of a metadata XML-based file associated to a video resource (e.g., a film trailer or a speech excerpt) follows:

```

<?xml version="1.0" ?>
<rdf:RDF>
  <rdf:Description
    rdf:about="http://www.site.org/
      Video.mpeg">
    <!-- temporal information -->
    <temporal:link
      begin="2003-05-09T17:15:00"
      end="2004-05-09T10:00:00"
      linkType="temporal">
    <temporal:Before dur="7D">
      <rdf:Description
        about="http://mirror.org/
          video/1.mpeg">
        ...
      </rdf:Description>
    </temporal:Before>
  </temporal:link>
  <!-- metadata -->
  <meta:Properties>
    <meta:Location
      ip="193.231.30.225"
      port="80">
      www.infoiasi.ro
    </meta:Location>
    <meta:Auth type="basic">
      UsersGroup
    </meta:Auth>
    <meta:Owner>
      <meta:Login>
        web
      </meta:Login>
      <meta:Password
        method="md5">
        ...
      </meta:Password>
    </meta:Owner>
  </meta:Properties>
</rdf:Description>
</rdf:RDF>

```

The XML namespace declarations of the presented constructs are omitted.

The document defines a *Before* relation between two resources. The video resource denoted by `http://www.site.org/Video.mpeg` is considered the original one, but a copy is located at the Web address (URI) `http://mirror.org/video/1.mpeg`. Each update of the original video implies another update of the copy after 7 days. The original resource will be available between the dates specified by `begin` and `end` attributes.

Metadata information includes the address of the storage machine, the owner of the resource and the authentication

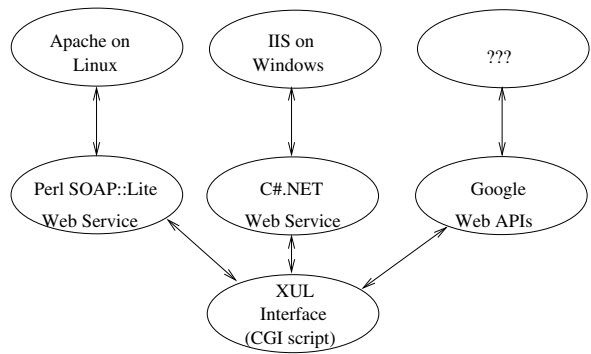


Figure 3. ITW Actual Architecture

mechanism. Of course, any other useful information – such as DCMI (Dublin Core Metadata Initiative) or RSS (Rich Site Summary) elements [29] – can be specified.

4.2. Actual Implementation

The ITW's actual implementation – in prototyping stage – includes the following components (see figure 3):

- one CGI script written in Perl that generates the XUL interface and functions as a generic client for the involved discovery Web services; the XUL documents are supported by Mozilla/Netscape browser. Using Extensible Stylesheet Language (XSL) [31] transformations, the XUL document can be transformed in other XML-based languages, such as XHTML or WML, in order to support any user-agent;
- two ITW Web services available on Linux and Windows platforms; one Web service is implemented in Perl by using *SOAP::Lite* module [26] and Apache Web server; the other is implemented in C# language on Microsoft .NET Framework [28], using IIS Web server;
- one external Web service freely provided by Google [27] in order to discover world-wide multimedia resources.

The implemented elements were tested on Windows XP – using .NET Framework 1.1 – and different Linux distributions (RedHat 9 and Mandrake 9.1) – using Perl 5.8 and Apache 2.0. For XML processing, the SAX (Simple API for XML) functionalities of the *XML::Parser* Perl module were used. The access to Google's Web service was given through a special unique key and the WSDL file provided by the search engine [27].

The Web robot that regularly updates the metadata information about the local resources is written in Perl language and uses *LWP* and *XML::Simple* modules [26].

5. Conclusion and Further Work

A multi-language architecture, named ITW, for multimedia resource discovery was presented. The ITW system consists of an ensemble of Web services, software agents and other components (described in section 4) and uses a RDF/XML model (see section 3 and [12]) for semantic representation of metadata and temporal relations between Web (multimedia) resources. This model is inspired by ITL logic [1, 4], shortly described in section 2.

The actual implementation was focused on the development of the local Web services – deployed on Windows and Linux platforms. For remote access to resources, the Google's search Web service is used (see section 4.2).

Instead of the XUL user-interface, another more flexible approach is given by the XSL (Extensible Stylesheet Language) [31] transformations, that can be applied on the server-side (e.g. using Apache AxKit [32] facilities) in order to offer a optimal interface for each user-agent (Web client or Web service).

To gain processing speed, in the next version of the system the XML/RDF generated constructs can be stored within an XML native database, such as Apache XIndex [32]. Another important aspect is to cache the results obtained from an interrogation in order to satisfy the client requests with the highest level of quality. To accomplish this goal, further work will be focused on serving differentiated content by using a tiered XML-based database storage service [13].

Also, we intend to enrich the presented model with more-complex RDF assertions about temporal events and actions, following the theoretical approaches stated in [3] and [19]. This will allow us to design the ITW agents for semantic resource discovery and to place the ITW project in the context of Semantic Web [15, 31] and Semantic Grid [17, 30].

References

- [1] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 1983.
- [2] J. Allen. Time and time again: The many ways to represent time. *International Journal of Intelligent Systems*, 6(4), 1991.
- [3] J. Allen and G. Ferguson. Actions and events in interval temporal logic. Technical Report 521, University of Rochester, New York, 1994.
- [4] J. Allen and P. Hayes. Moments and points in an interval-based temporal logic. *Computational Intelligence*, 5(4), 1989.
- [5] R. Alur and T. Henzinger. Logics and model of real time: a survey. In J. de Bakker et al., editors, *Real-Time: Theory in Practice*, Lecture Notes in Computer Science – LNCS 600. Springer-Verlag, 1992.
- [6] R. Alur and T. Henzinger. A really temporal logic. *The Journal of the ACM*, 41, 1994.
- [7] A. Artale and E. Franconi. Temporal description logics. In L. Vile et al., editors, *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. MIT Press, 1999.
- [8] J. Ayars et al. Synchronized multimedia integration language (SMIL 2.0). W3C Recommendation, World-Wide Web Consortium, Boston, 2001. URL: <http://www.w3.org/TR/smil20>.
- [9] D. Brickley and R. Guha. Resource description framework (RDF) schema specification. W3C Candidate Recommendation, World-Wide Web Consortium, Boston, 2000. URL: <http://www.w3.org/TR/rdf-schema>.
- [10] S. Buraga. A model for accessing resources of the distributed file systems. In *Lecture Notes in Computer Science – LNCS 2326*. Springer-Verlag, 2002.
- [11] S. Buraga, S. Alboaic, and L. Alboaic. An XML/RDF-based proposal to exchange information within a multi-agent system. In D. Grigoraş et al., editors, *Proceedings of NATO Advanced Research Workshop on Concurrent Information Processing and Computing*. IOS Press, 2003. To appear.
- [12] S. Buraga and G. Ciobanu. A RDF-based model for expressing spatio-temporal relations between web sites. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE 2002)*. IEEE Computer Society Press, 2002.
- [13] H. Chen and A. Iyengar. A tiered system for serving differentiated content. *World-Wide Web: Internet and Web Information Systems*, 6, 2003.
- [14] W. Conen and R. Klapsing. *A Logical Interpretation of RDF*, Linköping electronic articles in computer and information science edition, 2000. URL: <http://www.ida.liu.se/ext/epa/cis/2000/013/tcover.html>.
- [15] J. Davies, D. Fensel, and F. van Harmelen, editors. *Towards the Semantic Web*. John Wiley & Sons, England, 2003.
- [16] E. A. Emerson. Temporal and modal logic. In J. von Leuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Methods and Semantics*. Elsevier Science and MIT Press, 1990.
- [17] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kauffmann Publishers, 1999.
- [18] L. Lamport. “Sometime” is sometimes “not ever”: a tutorial on the temporal logic of programs. In *Proceedings of the 7th Annual Symposium on Principles of Programming Languages*. ACM SIGACT–SIGPLAN, 1980.
- [19] L. Lamport. *Specifying Systems. The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2003.
- [20] O. Lassila and R. S. (eds.). Resource description framework (RDF) model and syntax specification. W3C Recommendation, World-Wide Web Consortium, Boston, 1999. URL: <http://www.w3.org/TR/REC-rdf-syntax>.
- [21] I. Oeschger. *XUL Programmer's Reference Manual*. Mozilla.Org. URL: <http://www.mozilla.org/xpfe/Xulref.zip>.
- [22] S. Owicki and L. Lamport. Proving liveness properties of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 4(3), 1982.

- [23] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Symposium on the Foundations of Computer Science*. IEEE Computer Society Press, 1977.
- [24] L. Vieu. *Sémantique des relations spatiales et inférences spatio-temporelles*. PhD thesis, Université Paul Sabatier, Toulouse, 1991.
- [25] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: a revised report. In *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, CA, 1990.
- [26] CPAN – Comprehensive Perl Archives Network. URL: <http://www.perl.com/CPAN/>.
- [27] Google APIs. URL: <http://www.google.com/apis/>.
- [28] Microsoft .NET Framework. Microsoft Corporation, 2003. URL: <http://www.microsoft.com/net/basics/framework.asp>.
- [29] Purl Organization. URL: <http://www.purl.org>.
- [30] Semantic Grid Project. <http://www.semanticgrid.org>.
- [31] World Wide Consortium's Technical Reports, 2003. URL: <http://www.w3.org/TR/>.
- [32] XML Apache Project. URL: <http://xml.apache.org>.