Theme:

Title: **An Internet-based Building Simulation Quality Assurance System**

Author(s): Shengjiang Lu[1], Robert Amor[1] and Michael Donn[2]

Institution(s): [1]University of Auckland, Auckland, New Zealand

[2]Victoria University of Wellington, Wellington, New Zealand

E-mail(s): trebor@cs.auckland.ac.nz

Abstract: *Building environmental design decision support tools in architecture are not well used, even though there exists a wide range of tools for thermal, lighting, structural, etc simulation. Previous work has looked at the issues which inhibit the use of these decision support tools (Donn et al. 2001) and determined that a simulation quality assurance system would help practitioners to trust the predictions of a simulation system. This paper describes the development of a distributed building simulation quality control system.*

The basic premise behind this work is that we can determine the reliability of the results of an individual simulation through comparison with previous quality assured simulations. This requires a test to determine whether an e-building is real, where real is some measure relating the behaviour of the e-building to know real building behaviour. To make this test we collate a case base of quality assured e-buildings with which we can compare the submitted e-building. The development of the case base is drawn from the Internet by developing the semantic web concept. This requires simulationists to provide meta-data describing their simulations which is then harvested across the WWW, along with the simulations, to form a global and distributed simulation case base.

This paper describes the Internet-based system which has been developed to collate the case base from interested simulationists (comprising XML-based meta-data and simulation files) and utilise it in comparing with a submitted e-building. The comparison system utilises XML-based information management approaches tied to case-based reasoning to create this high-performance decision support system for architects.

Keywords: *Simulation, quality assurance, CBR, Query Language, Internet*

Introduction

Previous work has established a need for a system able to provide assurances of the suitability of an e-building to the simulation task it will be applied to (Donn et al. 2001, Maunder et al. 2001). The development of such a system will help address problems such as:

- Lack of quality control systems that allow users to self-calibrate their predictions, ensuring the recommendations they are making are relevant.
- Lack of tools for summarizing and detecting patterns within the simulation "output".

In order to solve these sorts of problems a SimQA system (building simulation quality assurance) was proposed, building upon the ideals of the semantic web (Donn et al 2001, Maunder et al. 2001). In the SimQA system, in order to check the quality of building simulations, simulationists can compare their simulations with qualified cases harvested from the Internet. The SimQA system is responsible for accepting e-building information from users, matching the e-building with other quality assured e-buildings through a range of user selectable similarity measurements, and presenting e-building recommendations back to the user. The majority of the work on SimQA has been carried out at Victoria University of Wellington and can be accessed at http://www.aecsimqa.net/. This paper reports on work purely concerned with feature-based matching of e-buildings in support of the SimQA goals.

In this paper we begin by discussing the basic architecture of the SimQA system and then consider e-building matching approaches. We look at the use of XML for a distributed Internet-based system and the development of a query language to allow e-building features to be defined for matching purposes. finally, we describe the user interface required to support such a matching process.

SimQA Architecture

SimQA is an Internet-based system comprising a multi-tier distributed system architecture based on the considerations of performance, scalability and security (Grundy et al 2002). Figure 1 provides a high level outline of the various components in the SimQA system. The user is able to submit a simulation file (e-building) across the Internet to SimQA through their WWW browser software. This submitted e-building is matched against a case base of quality assured e-buildings (held in SimQA's database) through a range of predefined features (or user specified features through the SimQA query language). E-buildings similar in type and performance to that submitted are returned to the user for further evaluation. At the top of Figure 1 the spider component is shown. This crawls the Internet to identify e-buildings which have been made available for general use and will add these e-buildings to a case base of quality assured buildings if they meet a particular set of quality criteria.

Figure 1 shows that SimQA runs over several tiers, including the user's PC, HTTP server, application server and database server. Administrator programs run between 2 tiers, including the administrator's PC and database server.

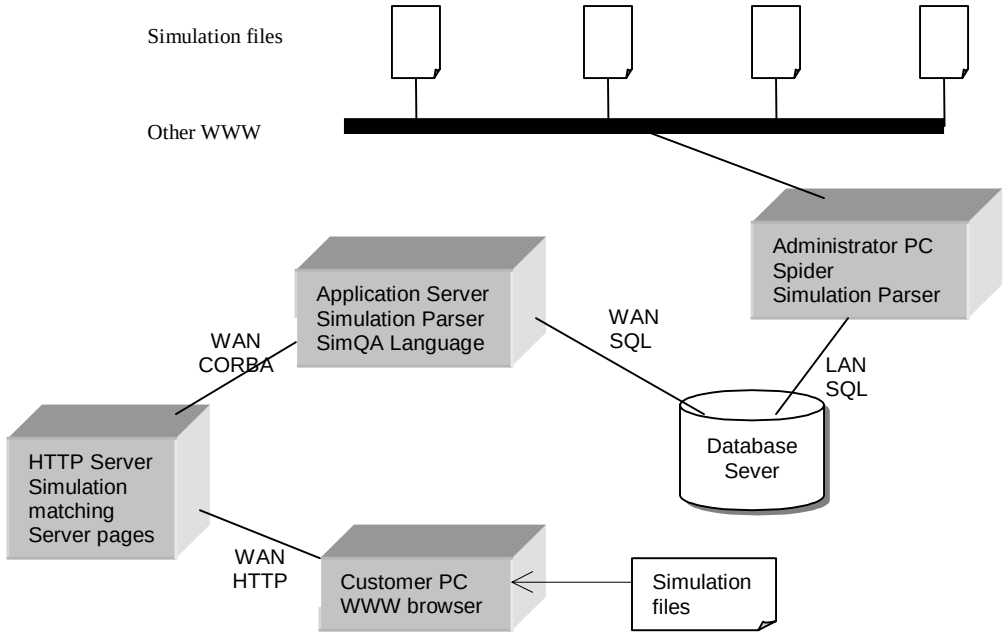

***Figure 1**. System architecture*

The user programs are actually distributed in four layers as shown in Figure 2. In the *presentation layer*, a user submits an e-building and a SimQA matching request. The presentation layer also displays the results back to the user. These functions are achieved with a graphic user interface (Geary and McClellan 1998) comprising a range of static HTML pages and server pages which are enhanced by DHTML and XML (St. Laurent and Cerami 1999).

The *component layer* contains a set of object models that encapsulate the SimQA logic. The main objects connect directly to the server pages layer. Some of the programs such as 'Simulation to XML' and the 'Data structure builder' need to connect to the *persistent layer*. This layer contains database server programs that are responsible for keeping e-building features and meta-data persistent for the component layer.

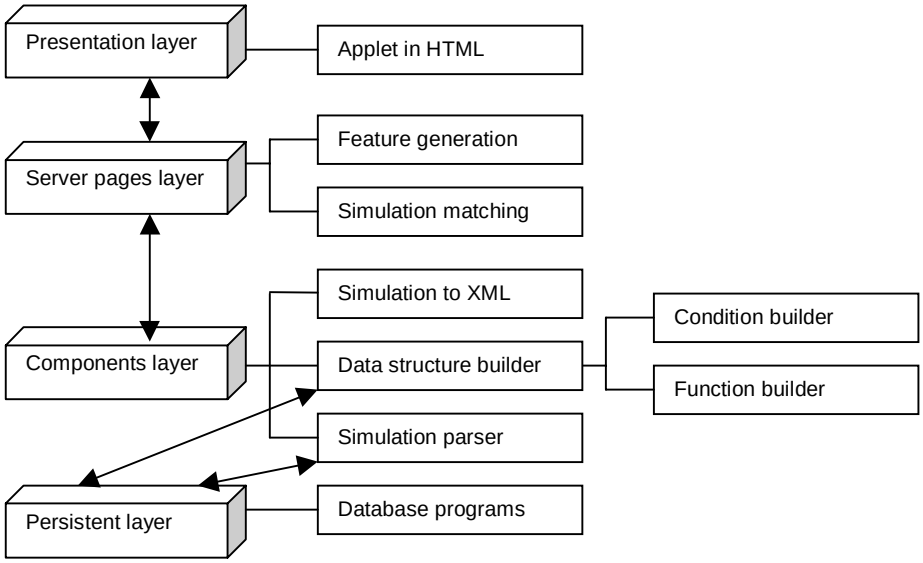

***Figure 2**. Four layer customer programs*

XML-based data management

The SimQA system utilises XML as its data transport and communication mechanism. XML is a friendly "container" able to represent and manage many different formats of simulation files and commands to be processed upon them. XML is also supported by a rich set of searching capabilities based on the Document Object Model or Simple API for XML (Moller 2001, St. Laurent 2000, St. Laurent and Cerami 1999).

The XML-based data management supports the capabilities outlined below:

- Data exchange: The SimQA case base is stored in XML containers. These cases can be parsed from many different kinds of building simulation input and output files.
- Data processing: The exchanged XML data can summarised and selected by a query language.
- Data presentation: The result of a XML query can be formatted and presented on web pages.

Figure 6 shows e-building matches for a user's query, created from XML and returned by the matching process.

E-building matching

One of the most important components of the SimQA system is its ability to compare a user-supplied e-building against a case base of quality assured e-buildings. The e-building matching is feature-based, which fits well with the underlying CBR (case-based reasoning) mechanism (Watson 2000, Watson 2001). An e-building is a simulation of the properties of a real, or proposed, building. In order to compare two e-buildings, it is necessary to identify features which can realistically be used as a basis for comparison. Raw information, such as the area of each individual space, is usually at too low a level to be of benefit in a comparison. However, information aggregated under certain constraints is more likely to be useful in a comparison of two e-buildings. Examples of features which can easily be constructed from an e-building include: total sun-facing room area; total sun-facing glazing area; window to wall area ratio; main building activity; climate zone of the building; etc. E-building matching is the process of using these features to find similarity. The closeness of fit between two e-buildings is based upon some defined set of these features.

SimQA utilises three types of e-building matching:

1. Default matching: A predetermined set of features is applied to an e-building. Users do not need to define or select features. When an e-building is submitted a comparison is made with the known case base.

2. Option matching: Users are able to select features from a public features library to drive the e-building matching process. This suits users who need to identify similar e-buildings on a select set of criteria.
3. Advanced matching: In addition users may define their own features through a feature specification interface. These new features can be used in conjunction with other library features and may be stored within the library for future use.

Whichever mode of matching is employed, the set of features to be compared is applied to the e-building supplied by the user and then across all e-buildings in the case base. A closeness of fit measure is calculated for each feature across the set of features used and then combining the results of each individual comparison. In the future, a weighting may be offered to allow prominence to be assigned to particular groups of features.

SimQA feature specification language

In order to define a new feature to be applied to an e-building, the user needs some language to describe the information they want to compare. SimQA Query is a high level and declarative language which allows the user to specify a feature based upon the XML data model loaded for their e-building. SimQA Query is based on the syntax of other XML query languages (Deutsch et al. 1998 and Chamberlin et al. 2000) but extended to handle query representations unique to this system.

Syntax overview

The SimQA Query system needs to work with XML data and hence requires a range of functions to manipulate portions of a hierarchical XML data structure. The SimQA Query grammar consists of the following components:

- Path expressions and variables

  A path describes where in the hierarchy of XML data the information to be accessed can be found. To define a path we use @ between element names and attribute names to denote the movement down the hierarchy.

- Functions

  Functions provide for the aggregation of data in a XML file. They include sum, count, average, max, min. All these functions (and conditional expressions) can be followed by a "group by" clause to allow the aggregated data to be grouped by a particular attribute (e.g. hour, day, month, year).

- Conditional and logical expressions

  Conditional expressions limit the data processed to calculate the feature. The form of a conditional expression is "where …". When multiple conditions are required, they can be joined by Boolean operators and relations. The conditional "contains" can be used to join two sets of XML data based upon a particular condition.

- Feature naming

  A feature can be named and then used in another feature expression. For example, "sum(Windows@WINHGT * Windows@WINLONG) save_as glazing_area".

  The value of feature "glazing_area" can be used in another feature specification such as "$glazing_area / $wall_area"

A feature example

A simple sounding feature may require a complex query containing many logical expressions and element pointers. Figure 3 shows an example of the process for defining north-facing room area. To determine the area we sum the ZAREA attribute of the zones element for all north facing zones (2). A zone is north

facing if there is a surface facing between 315 and 45 degrees and which has an element in it (4) which connects through to the zone (3).

north-facing room area ‖ sum(zones@ZAREA) where zone is north-facing ‖ North-facing zone names collection contain this zones//NAME ‖ (windows@WEXTSURF=surfaces@NAME and (SURFACES@SURFAZIM **<=** 45 or SURFACES@SURFAZIM **>=** 315) get windows@WINZONE)	(1) (2) (3) (4)

Figure 3*. Defining a feature for north-facing room area*

An example of how this query is solved is shown in Figure 4. We need to iterate through window surfaces to check whether they are north-facing. From the connection of windows to zones all north-facing zone names are collected at (4). Finally, the resultant zone area is calculated at (1).

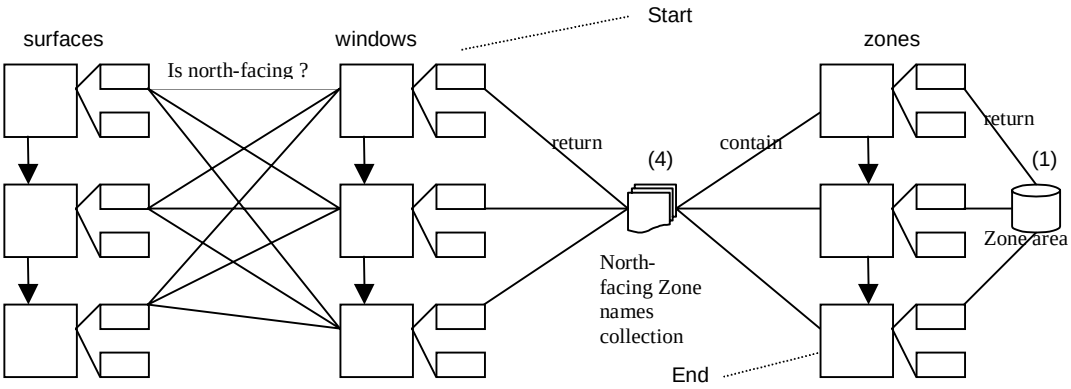

Figure 4*. Calculation of the north-facing room area*

Interface Design

A user interface had to be created that could cope with the complexities of defining features through SimQA Query, in order to manage the specification and matching of features. For the default matching, the interface needs only submit the e-building file and display the result. For the option matching the user interface needs to allow a selection to be made from the library of existing features. However, for the advanced matching, the features may be very complex and to write a query from scratch is difficult. A graphical interface that can be used to specify a feature query has been designed (see Figure 5).

To demonstrate how this interface works an example of defining north facing glazing area (similar to the feature shown in Figures 3 and 4) will be used. Figure 5 shows the interface being used to create this feature. The first line of the feature generator is for describing functions and the second line for describing conditions. In the attribute areas the attributes “Windows@ WINHGT”, “Windows@WINLONG”, “Windows@WEXTSURF”, “Surfaces@NAME” and “Sufaces@SURFAZIM” are selected from a tree representation of the e-building data model which is shown to the right of the figure. The sum function and all other operators are selected from pull down boxes on the screen. After entering all necessary information, and pressing the “create” button, a new feature will be created and displayed on the SimQA Query text area at the bottom of the figure.

The expression space can be expanded to create more complex expressions. For example, when defining the total north-facing window area feature we need a complex condition which includes "SURFACES@SURFAZIM <=45" as well as "SURFACES@SURFAZIM >=315". However, there is not enough room in the basic interface for this set of conditions. To do this the "Expand" button in the condition line can be pressed to add another line for further conditions.

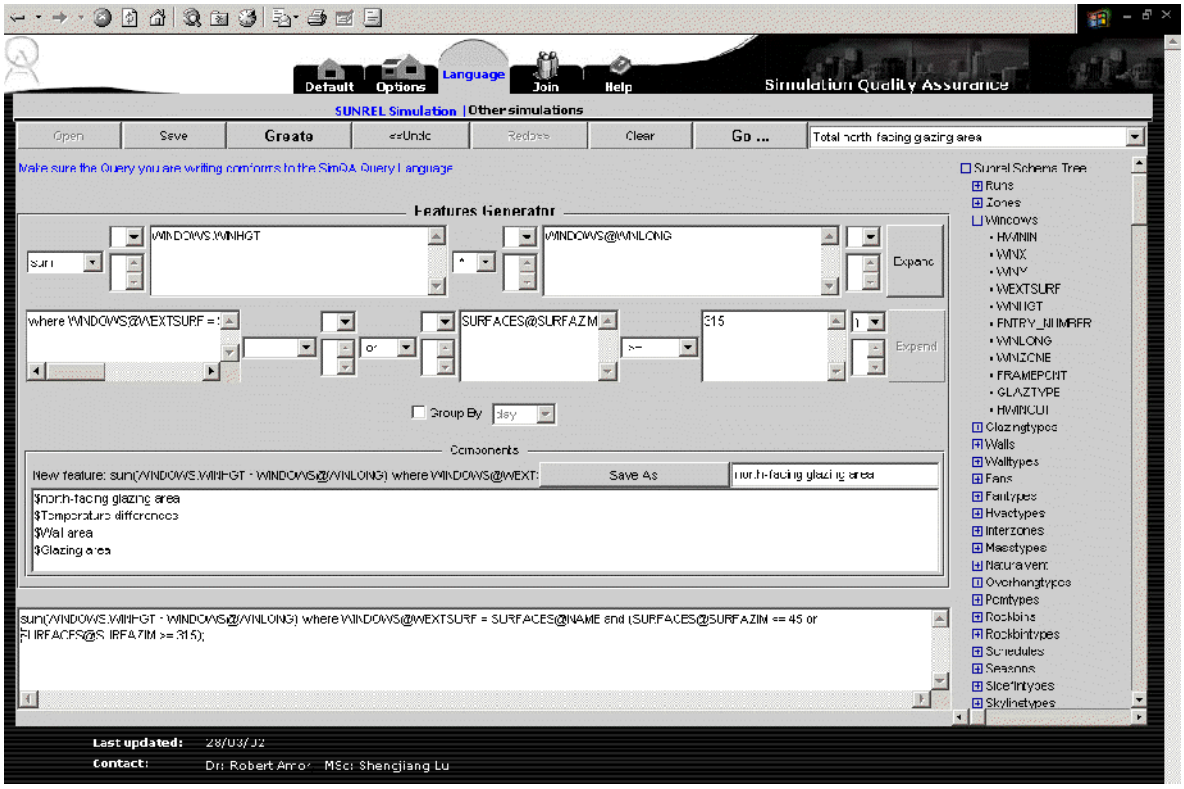

***Figure 5**. Defining a feature*

The interface shown in Figure 5 also provides the following functions:

- Public Features library

  This contains many typical features used to match e-buildings. A user may select them to use directly in the matching, or use them within a new feature specification.

- Component maker

  A newly created feature can become a component and be moved into the components list to be saved for later reuse. This means that users can have their private features library.

- Join and login

  Users can join the SimQA system and store their preferences, including individually defined feature components.

- E-building schema tree viewer

  E-building attributes and elements are displayed in a schema tree; they can be selected and moved to the feature generator by simple mouse clicks. The interface is able to load XML-based schema trees from multiple sources allowing data from several simulation systems to be used within SimQA.

Once all the features required to make a comparison are selected or specified, the user can ask SimQA to find matching buildings. Results of this comparison are as shown in Figure 6 where the similarity

measure for e-buildings in the case base is shown, along with the values calculated for each of the features used.

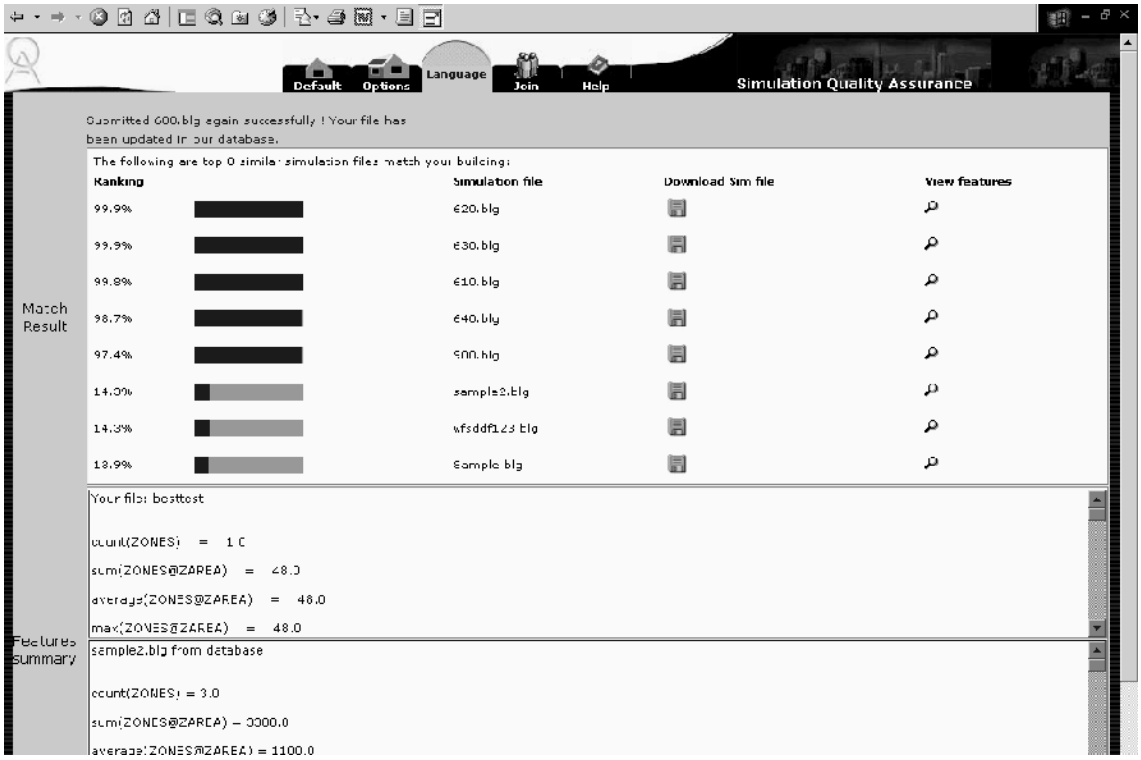

***Figure 6**. Results from searching for a matching building*

Summary

The SimQA system looks at providing simulationists with a level of confidence in their e-buildings. This is achieved by building upon the semantic web concept and pooling together resources from across the web to help users identify similar simulation problems to the one they are looking to solve.

The SimQA system described here provides a high-performance mechanism supported by a XML-based, multi-tier distributed system architecture to help users match e-buildings. A SimQA XML-based query language was developed as a core component of this system to allow users to define the features they require to match e-buildings. A user interface for SimQA has been designed to provide a rich public features library and allows the creation of a private features library for simulationists. E-building matching brings results immediately after a user submits their e-building through the SimQA interface.

The work described here is still in progress and there are many concerns which have not yet been addressed, but are certainly important issues for the future, including:

- Supporting more heterogeneous simulation packages (currently only thermal simulation has been examined).
- Providing more user control on the weighting of features used in the e-building matching.
- Optimising the quality assured case base which is distributed throughout the WWW.

References

Chamberlin, D., Robie, J. and Florescu, D. (2000) Quilt: An XML Query Language for Heterogeneous Data Sources, IBM Almaden Research Center, http://citeseer.nj.nec.com/chamberlin00quilt.html, last accessed March 2002.

Deutsch, A., Fernandez, M. and Florescu, D. (1998) A Query Language for XML, University of Pennsylvania, AT & T Labs, http://www.w3.org/TR/NOTE-xml-ql/, last accessed March, 2002.

Donn, M., Amor, R. and Harrison, D. (2001) A design proposal for an internet based simulation quality control tool, Proceedings of the IPBSA conference, Building Simulation 2001, Rio, Brazil, 13-15 August.

Geary, D.M. and McClellan, A.L. (1998), Graphic Java 1.2, Mastering the JFC: AWT, Volume 1, the Sunsoft Press Java Series, ISBN: 0130796662.

Grundy, J., Wang, X. and Hosking, J. (2002) Building Multi-Device, Component-Based, Thin-Client Groupware: Issues and Experience, Proceedings of the 2002 Australasian User Interface Conference, Melbourne, Australia.

Maunder, R., Donn, M., Curtis, S. and Lee, S. (2001) What does it look like? – Quality control in simulations of lighting appearance, Proceedings of the IPBSA conference, Building Simulation 2001, Rio, Brazil, 13-15 August.

Moller, A. (2001) The XML revolution – technologies for the future Web, http://www.brics.dk/~amoeller/XML/, last accessed March 2002.

Shannon, B., Hapner, M. and Matena, V. (2000) Java 2 platform, enterprise edition - platform and component specifications, Addison-Wesley, ISBN 0-201-70456-0.

St. Laurent, S. (1999) XML: A Primer, 2nd Edition, M&T Books, ISBN 0-7645-4771-1.

St. Laurent, S. and Cerami, E. (1999) Building XML Application, McGraw-Hill, ISBN 0-07-134116-1.

Watson, I. (2000) A Case-based Reasoning Application for Engineering Sales Support using Introspective Reasoning, proceedings of 17th. National Conference on Artificial Intelligence (AAA!-2000) and the 12th Innovative Applications of Artificial Intelligence Conference (IAAI-2000), July 30 - August 3, Austin Texas, pp. 1054-1059, AAAI Press, ISBN 0-262-51112-6.

Watson, I. (2001) A decision support system for local government regulatory advice, proceedings of FLAIRS-2001. 14th Int. FLAIRS Conference, Key West Florida May 21-23 2001. AAAI Press, Menlo Park CA, US, pp. 329-333.