ENERGY BASED DECISION SUPPORT SYSTEM FOR FACILITIES MANAGEMENT: INTEGRATION OF DATA/WEB MINING, KNOWLEDGE BASE AND THERMAL SIMULATION

Ali Malkawi and Ravi Srinivasan

T. C. Chan Center for Building Simulation & Energy Studies, School of Design
University of Pennsylvania, Philadelphia PA, USA 19104

ABSTRACT

This paper presents the development and implementation of an Energy-based Decision Support System (EDSS) that will enhance the selection process of replacement building features. EDSS includes dynamic databases that utilize data/web mining concepts, Knowledge Base System (KBS) and a thermal simulation engine. The system allows decisions to be made based on a long-term vision incorporating energy cost savings rather than immediate needs.

A stand-alone application was implemented based on EDSS. It allows the user to specify building features (e.g., windows) that are connected to a virtual database and a KBS, and integrated with a thermal simulation engine. Based on specific goal requirements for windows such as visual transmission, energy requirements and color, the application locates a set of windows from local databases and/or web-based vendors that best matches these goals using the KBS. These window specifications are then sent to a thermal simulation engine which computes energy usage. The performance results are provided to the user for selection. The results can be used as part of the evaluation process of the selection of replacement building features based on energy cost savings.

KEYWORDS

Energy, Decision Support System, Facilities Management, Data/Web Mining, Knowledge Base, Thermal Simulation.

1. INTRODUCTION

Facility management tools for design, maintenance, and operations collect individual facility data to improve the overall efficiency and resource management. Such tools primarily focus on inventory of assets, and planning and evaluation of work. They address deferred maintenance, measure future re-capitalization in asset planning, and improve stewardship and accountability. In addition, they aid in maintaining the condition of assets at an acceptable level, improve asset planning, and capture total cost of ownership of assets.

Facility management tools can be broadly classified as Computerized Maintenance Management Systems (CMMS) and Enterprise Asset Management Systems (EAMS). A CMMS is a complete system utilized to plan, schedule and track maintenance management activities, including resources, materials, labor hours, and costs. EAMS allows users to make informed decisions regarding facility infrastructure condition, multiyear capital budgeting, capital project planning, and functional adequacy based on information that has been collected through a building condition analysis. Facility management tools can be implemented either as stand-alone or web-based. Stand-alone tools can be expensive and may be cost-prohibitive for clients. In a stand-alone implementation, several individual applications are required for the initial installation to satisfy each resource. Much of the effort is in the integration and scalability work for ongoing maintenance and upgrades. On the other hand, web-based facility management tools continue to gain popularity. In this scenario, users store their data at a physically separate data center and access the system via the Internet.

Facilities are major energy consumers for heating and cooling. There is a need to reduce energy consumption and become more sustainable with energy conservation through effective design and decision making (Hodges, 2005). This is particularly true for large-scale operations. One approach is by conscious selection and integration of energy saving building features during the life of the buildings. Such an approach requires knowledge in the field of building energy performance and decision theory (Keeney and Raiffa, 1993; French, 1993; Cross 1994; Roozenburg et al., 1991).

Present day facility management tools provide extensive data management tools for effectively managing various facilities and their assets. Nevertheless, these applications do not allow for the selection and replacement of building features (windows, doors, etc.) based on energy cost savings. Such an approach will permit energy conservation and when practiced by large-scale operations, can conserve enormous amounts of energy.

This paper presents the development and implementation of an EDSS that will enhance the

decision-making process of selecting replacement building features. The system includes dynamic databases that utilize data/web mining concepts, knowledge-based and goal-driven systems to search and filter the databases based on user preferences, and a thermal simulation engine that predicts the performance of the building features. These three components were integrated using a computational manager module that was developed to permit communication among various components. A web-based interface was developed as a demonstration. The interface allows the user to access databases, change preferences, choose and interact with data, and select the building feature based on its energy consumption.

2. ENERGY-BASED DECISION SUPPORT SYSTEM

The Energy-based Decision Support System includes three components – building features database module, KBS module, and thermal simulation module, figure 1. The building features database module utilizes data/web mining concepts to retrieve and store features data from both architectural catalogs and the Internet respectively. The KBS module performs the search and filter functions based on user preferences, while the thermal simulation module conducts simulations to predict the performance of building features for decision support.

Based on specific goal requirements for windows such as visual transmission, energy requirements, and color, the implemented system is able to locate a set of windows from web-based vendors and/or local databases that best matches these goals. These window specifications are then sent to a thermal simulation engine and the energy results are provided to the user. The energy results are used as part of the evaluation process of the selection of replacement building features.

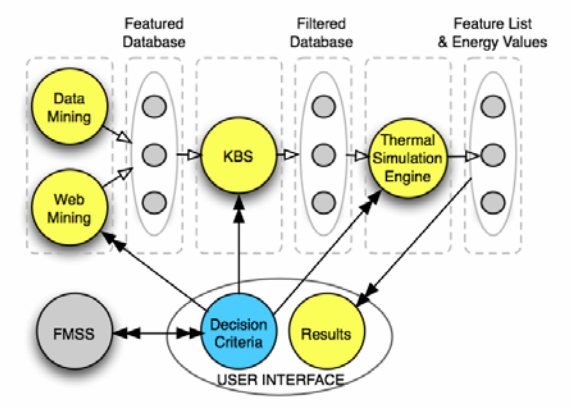

Figure 1. EDSS structure.

2.1 BUILDING FEATURES DATABASE MODULE

Building features are constantly improved owing to the technological advancements in manufacturing processes and materials research. An elaborate building features database is essential for effective functioning of EDSS. Relevant building features data is mined from local storage (data mining) and via the Internet (web mining). While data mining allows access to local storage formats, web mining performs Internet-wide search to retrieve relevant information.

For the web-based application, a window database was built to store general and energy related data. General information includes vendor / manufacturer information, window model type, color, and minimum / maximum size availability. Energy-related information includes Shading Coefficient (SC), U-Value, and visibility. While the general information stored provides users with building features available in the market, the energy-related material attributes are sent to the thermal simulation engine to compute respective energy values.

2.1.1 Data Mining from Local Storage

The first data mining approach allowed the retrieval of relevant information from local storage formats such as CDs (“Sweets 2004 Catalog”). The catalog CDs consisted of building feature data in several file formats including HyperText Markup Language (HTML), Microsoft Excel, and Autodesk AutoCAD. In addition, the folders and data files used a non-descriptive numbering system. The availability and organization of building feature data differed and a few lacked specific product data beyond a brief description and contact information. In order to overcome these irregularities, a scalable computational data mining algorithm was developed. The retrieved values are stored online using PostgreSQL.

Implementation

Two tables, “vendors” and “windows,” contain the windows database. The “vendors” table is comprised of vendor information and the “windows” table includes feature specification data, table 1. In order to maintain data integrity, constraints were added to the tables. For example, each entry of the “vendors” table must have a unique vendor name and the vendor name should be stored in the “vendors” table prior to adding window data. The database can be extended to hold doors, walls, roofs, and other building features.

Excel files were read, converted to Excel 5.0 format, decompressed, and parsed into the computational module, figure 2. A Java-based application “ExcelToSQL” based on JExcel API (JExcel, 2005) was employed to read, convert and decompress Excel

4.0 files. Java.sql API along with PostgreSQL 8.0 JDBC 3.0 API (PostgreSQL, 2005) was utilized to parse relevant data to the database. "ExcelToSQL" parses the files for data availability by first checking which of the different table structures is used by the current file. JExcel API reads data from individual cells containing the information to be mined. A connection to the PostgreSQL server is then made, and SQL statements are formed using the collected data from the Excel sheets to insert new entries into the tables.

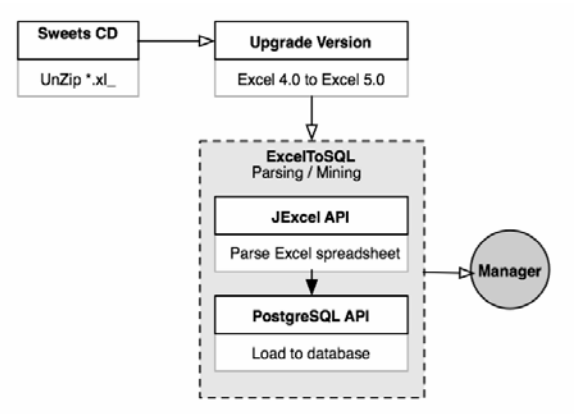

Figure 2. Implementation of building features database using data mining.

2.1.2 Web Mining from the Internet

To extract window attributes and vendor information from the Internet, a web mining module was developed. Web mining follows two main tasks – resource discovery and information extraction (Etzioni, 1996; Kosala and Blockeel, 2000). Resource discovery involves locating documents and services on the web while information extraction extracts specific information from newly discovered web resources.

Implementation

The web mining module utilizes Google API (Google, 2005) to search the Internet for specific buildng feature data, figure 3. A Java class "GoogleSearchEngine" along with PostgreSQL 8.0 JDBC 3.0 API was developed to perform this function. The class uses "GoogleSearchEngine" to connect to Google API to retrieve the search results in the form of URLs. The results are sent to the "WebMiningUtils" class already developed. This class acts as a manager for two other classes, "HtmlParserService" and "WindowDao." The "HtmlParserService" class connects to every URL and extracts the HTML content. In addition, "HtmlParserService" class parses the HTML to find specific keywords which are defined in the system. If the "HtmlParserService" Class locates any matching results, it will return them to the "WebMiningUtils" class and save them in the database using the "WindowDao" class.

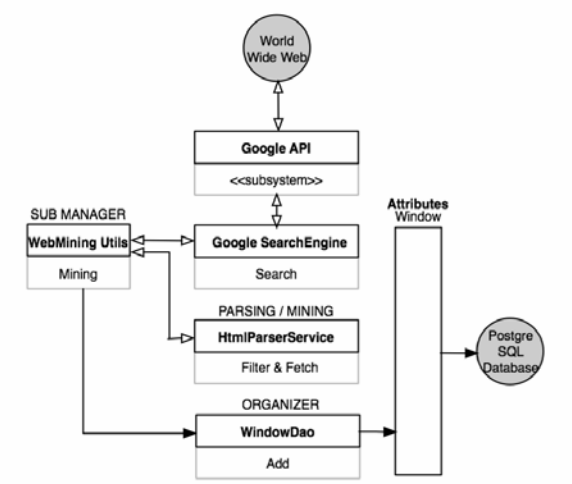

Figure 3. Implementation of building features database using web mining.

2.2 KNOWLEDGE BASE SYSTEM (KBS) MODULE

To search and filter the building materials database, a KBS was developed and integrated with EDSS. It converts user input into specific values that will be employed to narrow the list of building features that is passed to the thermal simulation engine. Based on specific goal requirements for windows such as visual transmission, energy requirements, and color, the system locates windows from the database that best match user preferences. To provide enhanced results, flexibility was added to KBS. Even if the specified criteria are too stringent to produce any result, the system returns results that closely match the criteria. For example, if the user specifies preferences that no windows match, the knowledge base will automatically lower its constraints and return the closest matches. With the constraints being lowered during each iteration, the system resembles a finite state machine in which the system passes into the next state along its selected path each time the knowledge base runs through its rule-set.

Implementation

The KBS module was implemented using CLIPS, an expert system shell that provides a complete environment for the construction of rule-based and/or object-based systems. CLIPS handles a variety of knowledge that supports several programming paradigms such as rule-based, object-oriented, procedural, etc. (CLIPS, 2005). In CLIPS, "if-then" logic statements are referred to as rules. The "if" portion of the rule describes facts that make the rule applicable while the "then" portion of the rule executes the applicable rule. In other words, CLIPS matches facts against patterns and determines rules that are valid for a given situation. The inference engine of the expert system runs these rules until no applicable rules remain. Examples of CLIPS integration include entering context information and

user actions using semantic representation (Billinghurst and Savage, 1995); assisting lime growers and extension agents (LIMEX, 1996); domain-independent context representation development for electronic patient records (Muller, 1997), etc.

User preferences for energy value include "EN-HIGH" (high energy value), "EN-MID" (medium energy value), and "EN-LOW" (low energy value). Based on user selection, KBS processes necessary logic statements. If no exact matches were found in the first run, the system continues to run using secondary rules to identify any matches. Once all rules are exhausted, it ends in "failure," implying that the constraints cannot be lowered any further. The rules in this system are designed to resemble a finite state machine that would move through the rule set in a way that supports uncertainty, figure 4. Communication between the database and KBS was enabled using PostgreSQL API. The final building feature list is sent to the thermal simulation engine to predict the thermal behavior for the exact asset being investigated.

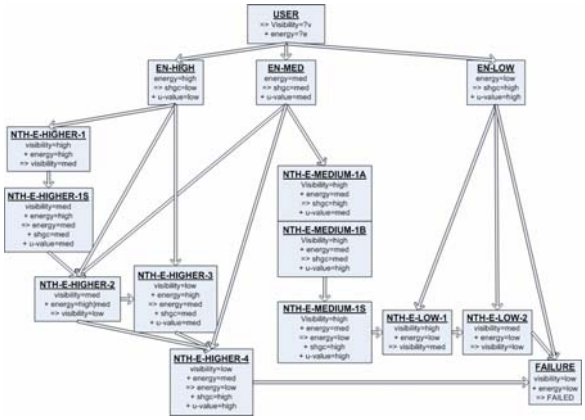

Figure 4. KBS rule structure.

2.3 THERMAL SIMULATION MODULE

The thermal simulation module determines thermal loads for the building features processed by the KBS. For this project, the module was developed based on detailed Transfer Function Method (TFM) and implemented as an object that can interact with the KBS output in a web-based environment. TFM was first introduced by Stephenson and Mitalas (Stephenson and Mitalas, 1967; Mitalas, 1973). This procedure is based on response factors and the interplay of heat exchange between various surfaces and sources of heat gain (Romine, 1992). These response factors are defined as an infinite series that relates a current variable to past values of other variables at discrete time intervals. A transfer function converts the theoretically infinite set of response factors into a finite number of terms that multiply both past values of the variable of interest and past values of other variables (McQuiston and Spitler, 1992). TFM contains three levels of analysis in its calculations. These include, first, the determination of the heat gain or loss the building produced based on the origin of the elements that produce these loads; second, the conversion of this heat gain or loss into cooling or heating loads; and finally, the determination of heat extraction rates. For the purpose of this project, only heating gains were used for predicting the thermal behavior of the windows.

Implementation

The thermal simulation module implemented in Java and Visual Basic takes into account the changing conditions in the temperature of the outside air, using the concept of sol-air temperature when calculating the difference between the outside and inside temperatures. These calculations are dependent upon the hourly calculations of solar intensity for each exterior surface. Solar heat gain, a combination of Transmitted Solar Heat Gain (TSHG), Absorbed Solar Heat Gain (ASHG), and Conductive Solar Heat Gain or loss (CSHG) is computed by "glazing_sh()" and "glazing_conductance()" function routines.

The transmitted and absorbed solar heat gain for each hour for each window is calculated based on the transmitted and absorbed solar heat gain factors. Conductive heat gain, on the other hand, depends on the flow of heat due to the change of temperature between outside and inside. Exterior shading for glazing is also taken into account within the process. And finally, the total heating gain or loss for the building is calculated, figure 5.

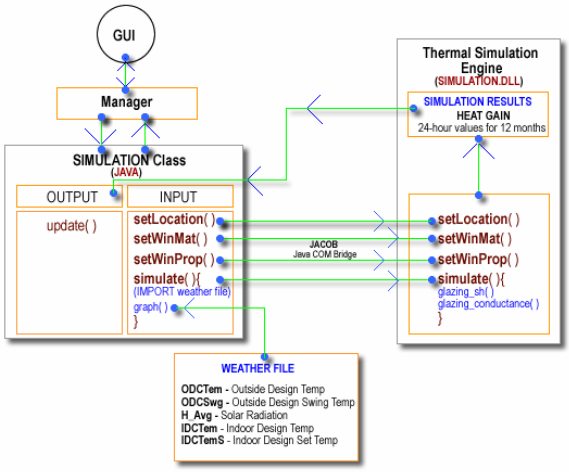

Figure 5. Thermal simulation module.

The simulation engine is integrated with EDSS via Java Native Interface (JNI). The output of the simulation is passed to the user interface along with window information for further manipulation.

2.4 INTEGRATION OF MODULES

The database, the KBS, and thermal simulation modules are integrated together via the "Manager," figure 6. The Manager allows communication among

the modules through the use of various APIs. The Graphical User Interface (GUI), on the other hand, passes the user input values directly to the manager class using Java classes. The Manager returns to the GUI a list of windows with energy values that matched the user's input, which is then displayed in the GUI for the user to view.

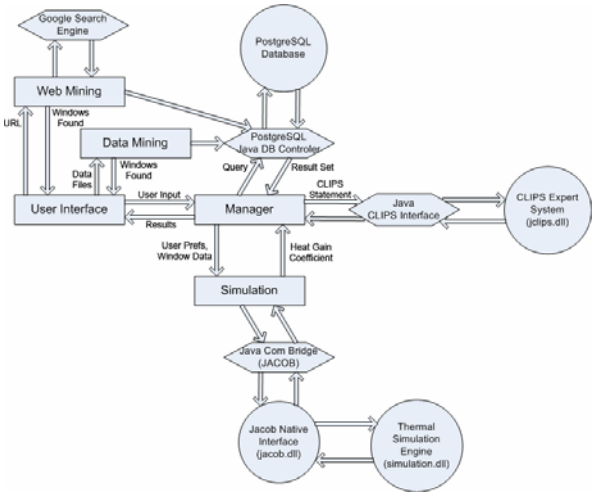

Figure 6. Integration of modules using the Manager.

Implementation

The Manager connects to the PostgreSQL database using PostgreSQL 8.0 JDBC 3 driver and the java.sql API. The JDBC driver contains code that allows the user to connect to a PostgreSQL database on a remote server. Using the driver along with the java.sql API, a connection is created to the database. Queries are executed using Java by passing the Query to SQL as a Java string. The results are returned to Java as multi-dimensional arrays of strings that can be read from Java. The Manager communicates with CLIPS knowledge base via the JCLIPS API (JCLIPS, 2005). The JCLIPS API allows the Manager to communicate with CLIPS using methods such as "reset( )" and "run( )" for executing commonly used CLIPS commands.

The Manager creates a connection to both PostgreSQL and JCLIPS. User input is passed from the GUI to the Manager using the "getWindows( )" method. The Manager then passes the user input values for visibility and energy preferences into CLIPS and runs the rules based on that input. When a rule is found true, it returns the result to the Manager via the JCLIPS "send-to-java( )" command. The Manager's "update( )" method picks up the results and uses them, along with the rest of the user input, to form an SQL query. If the query returns empty, the knowledge base is run again and another SQL query is made. If the query returns some matching entries, then a simulation is performed on each of those entries.

To run the simulation engine object, the Manager uses the JAva COm Bridge or JACOB (Adler, 2005), that uses the JNI API to communicate with native Win32 libraries (.dll files). JACOB comes in two parts, the API which allows the user to access its native code in Java, and the "jacob.dll" file, which is itself a native Win32 library used to communicate with other libraries. A simulation class was created in Java that uses the JACOB API to create a wrapper for the simulation object. The Manager takes the results from the simulation, adds them to the window data, and returns the windows list and simulation results back to the GUI.

3. SYSTEM INTERFACE AND DEPLOYMENT

The system interface allows the user to input feature information and preferences, and to access output. The interface consists of two tabs, decision criteria and results. The decision criteria tab allows users to input feature dimensions, specify preferences, and select or modify databases. The results tab displays the final results for user selection.

The decision criteria tab allows the user to input window dimensions, select preferences, and access database settings, figure 7. Window dimensions include height and width (in inches). Preferences consists of color, visibility, and energy levels. Glazing color choices include "clear," "blue," "yellow," "red," and "green." Visibility and energy level options are "low," "medium," and "high." Users may also specify the relative importance of each category. The system mines data from local storage and / or from the Internet. The user can select either one or both options for data retrieval. Users can also modify web database settings to suit their feature selection by clicking the "modify" button in the databases section. This activates the web data editor that allows input of new URLs and /or keywords for Internet searches, figure 8. This list can also be changed based on search criteria. Database settings can also be saved for future use.

With necessary inputs, the system starts to communicate with KBS to filter and search relevant data from the database, and parses it to the thermal simulation engine. As the system progresses, the information is displayed in real time in the "result" tab, figure 9. Real time information displays include a listing of the currently running process and the total number of feature data added based on data / web mining. The final feature list includes vendor information, feature model, color, dimensions, visibility, and energy values. Currently, the window feature is implemented and can be expanded to include a variety of building features.

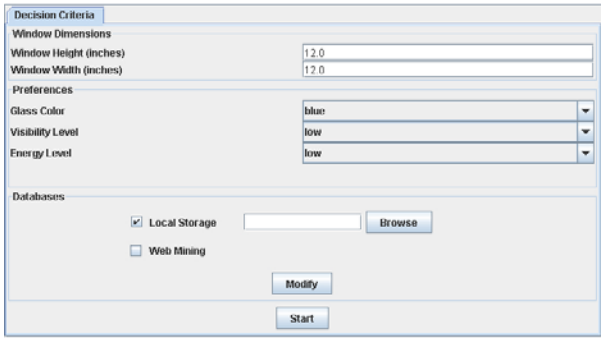

Figure 7. User Preference Interface.

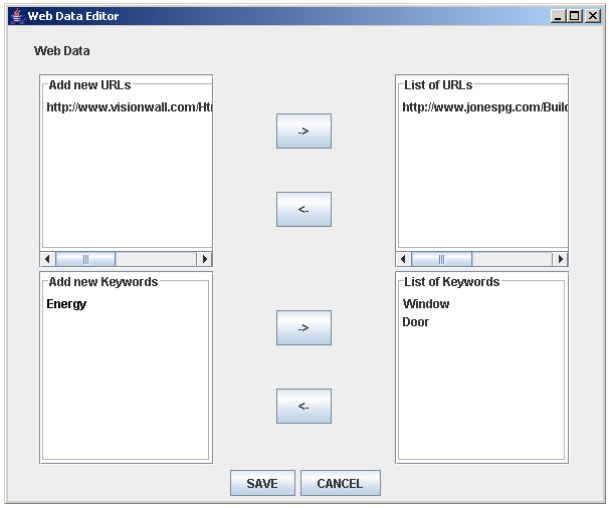

Figure 8. Data / Web Mining Interface.

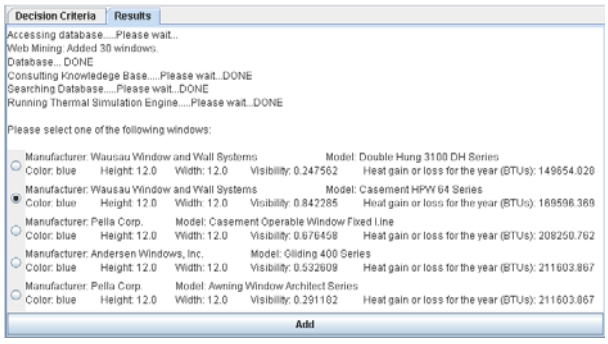

Figure 9. Results Interface.

EDSS was deployed as a Java Web Start (JWS) application. A JWS application accesses system resources and native libraries and executes independently from the web browser, thus circumventing any security disparities between the browser and JRE. It uses the Java Network Launching Protocol (JNLP) to download and install an application from a web server, and to run it locally. This is done through the use of a JNLP file written in eXtensible Markup Language (XML) and integrated into a HTML webpage through hyperlink to the JNLP file. The XML code consists of information needed to download, install necessary libraries, and run the application on the client machine without difficulty.

4. CONCLUSIONS

The paper discussed the development of an Energy-based Decision Support System that will enhance the decision-making process for facilities management. The prototype demonstrated opportunities that facilities management software can utilize to increase its efficiency as a tool (the web mining, the KBS, the integration with simulation, etc.). Although the current implementation demonstrated the advantage of the energy-based decision support model for windows, the system can be expanded to include other building features to increase long-term savings. As the system expands in regard to the features that it supports, its overall structure in regard to mining data can remain similar. This structure includes data mining from local libraries or the Internet, the use of the KBS to filter and assist the user, and integration of the features with a thermal simulation engine. Features identified for such scalability include wall, doors, roof, etc.

The development demonstrated the advantages of using the KBS module. This module can be enhanced to include more of the user preferences. As more features will be included, separate KBS modules can be added that will function in a similar way (as independent entities/agents). A sub-manager to allow for communication among the different agents will be need at that stage. Different models do exist to allow communications without interfering with the agent's autonomy.

ACKNOWLEDGMENT

Funding for this research was provided by University - National Park Partnership Program through sub-contract from Rochester Institute of Technology (RIT). The authors would like to thank Dr. James Winebrake at RIT, Jean Marra at the National Park Service and graduate students Jhairo Erazo and Cheng-Long Yeh.

REFERENCES

Adler, D. The JACOB Project: A Java-COM Bridge. http://danadler.com/jacob

Billinghurst, M. and Savage, J. 1995. Directive Interfaces for Virtual Environments: Unpublished technical notes. Presented at ACM Symposium on User Interface Software and Technology (UIST '95), New York, NY.

CLIPS Delivery Expert System. http://www.ghg.net/clips/WhatIsCLIPS.html

Cross, N., Engineering Design Methods, second ed., 1994. John Wiley & Sons, Chichester.

Etzioni, O. 1996. The world wide web: Quagmire or gold mine. In: Communications of the ACM, 39 (11):65-68.

French, S., Decision Theory – An Introduction to the Mathematics of Rationality, second ed. 1993. Ellis Horwood Limited, Chichester.

Google API. http://www.google.com/apis/

JCLIPS API. http://www.cs.vu.nl/~mrmenken/jclips/

JExcel API. http://www.andykhan.com/jexcelapi/

Keeney, R.L. and Raiffa, H. Decisions with Multiple Objectives – Preferences and Value Tradeoffs, 1993. Cambridge University Press, Cambridge.

Kosala, R., and Blockeel, H., 2000. Web Mining research: A survey. In: ACM SIGKDD 2 (1): 1 – 15.

LIMEX.http://www.claes.sci.eg/claes/limex/limex.html

Malkawi, A., 1994. Building Energy Design and Optimization: Intelligent Computer-Aided Theoretical Design, Ph.D. Dissertation, Georgia Institute of Technology, Atlanta.

McQuiston, F.C. and Spitler, J.D. 1992. "Cooling and Heating Load Calculation Manual," 2nd ed. Atlanta: American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc.

Mitalas, G.P. 1973. "Calculating Cooling Load Caused By Lights," ASHRAE Journal; June: 37-40.

Muller, R. 1997. The CLINICON Framework for Context Representation in Electronic Patient Records, Proceedings of the AMIA Annual Fall Symposium

PostgreSQL 8.9 JDBC 3.0 API. http://jdbc.postgresql.org/download.html/

Primikiri, E. and Malkawi, A. 2001. “Distributed Simulations: An Object-Oriented Approach.” In Proceedings of the Seventh International Building Performance Simulation Association (IBPSA) Conference Held in Rio de Janeiro, Brazil.

Romine, T.B., Jr. 1992. "Cooling Load Calculation: Art or Science?," ASHRAE Journal; January:14-24.

Roozenburg, N. and Eekels, J. Productontwerpen, Structuur en Methoden, Lemma. 1991. Utrecht.

Russell, S. J., and Norvig, P.. Artificial Intelligence A Modern Approach. 2002. Prentice-Hall, New Jersey.

Sowell, E.F. 1998. "Load Calculations for 200,640 Zones," ASHRAE Transactions; Vol.94, Pt.2:716-736.

Stephenson, D.G. and Mitalas, G.P. 1967. "Cooling Load Calculations by Thermal Response Factor Method," ASHRAE Transactions Vol. 73, Pt.1.