THE JEVIS SYSTEM - AN ADVANCED DATABASE FOR ENERGY-RELATED SERVICES

Peter Palensky
Institute of Computer Technology, Vienna University of Technology
Gusshausstrasse 27-29/384, A-1040 Vienna
Austria

Abstract

The JEVis Framework is a database system for realizing services in the domain of energy-related data management. It combines a heterogeneous portfolio of technologies and services by mapping them onto a common database. This database is the integrative part and the starting point of all service design which happens in a new and innovative date-centered way. It connects to multiple sources and sinks of data and offers a versatile platform for implementing various web-enabled services. This paper describes the principal system structure and gives a basic overview on the possibilities of the JEVis Framework.

Key Words

Database, Services, Automation Technology, Data Acquisition

1. Introduction

The JEVis System is the commercial exploitation of a number of projects done by the JEVis Team at the Institute of Computer Technology (ICT) at the Vienna University of Technology (Austria) and Envidatec GmbH in Hamburg (Germany). JEVis is short for Java Envidatec Visualization, which origins from the first versions of this system. As a result of the IGUANA project [1], distributed Java-enabled embedded Web-servers were used to visualize data that was logged and acquired from automation networks.

Meanwhile the JEVis system grew in both complexity and functionality. It now consists of an Oracle 9i relational database (DB) and a number of server side programs that act as an Internet portal (see http://www.my-jevis.com).

Envidatec's main business and expertise are energy-related services and applications like energy management and energy data management while ICT contributes knowledge about and experience in fieldbus systems, security, database systems and general communication technology.

The task of the system is to integrate various state-of-the-art technologies for providing web-enabled services for SCADA (supervisory control and data acquisition) and home/building-automation purposes. Typical applications are energy-related services like remote metering, billing, benchmarking, statistics, or energy management and remote-home functionalities for increased efficiency, security, safety and comfort.

The paper describes the structure of the JEVis system and how the individual parts interact. After giving an example for services that are or can be implemented on top of the JEVis platform the paper finishes with an outlook on future research and development.

2. System Overview

The environment of the JEVis System is the domain of web-enabled services and add-on services (using one infrastructure for a number of different services). These services are based automation networks, the Internet and other communication technologies. Industry offers many different flavors of such technologies which results in a broad spectrum of proprietary solutions. The JEVis System tries to play an integrating role in this heterogeneous environment (Fig. 1). The data sources and data sinks of the JEVis environment are automation networks with their respective Internet-gateways, other databases and a variety of web-based user interfaces.

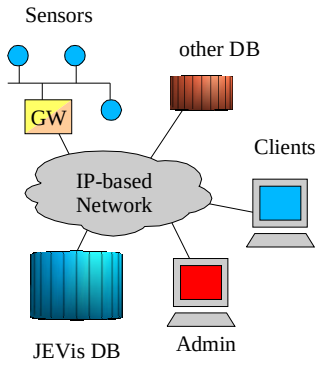

Fig. 1: The JEVis environment

JEVis interconnects all these sources of data with the sinks of data: user interfaces, destination databases and the like.

A typical example for the usage of the JEVis system might be "benchmarking" in facility management. Benchmarking in this context means evaluating the properties of buildings: Certain attributes of buildings are mapped onto each other in order to get key-values that can be compared to each other (Measurement Values, Statistics, Standards, Classifications, Properties of Objects, etc.).

A supermarket chain might for instance be interested in comparing the energy consumption of their subsidiaries. This only makes sense if the - let us say monthly - measured energy consumption is normalized to the size of the supermarket or to the number of customers, otherwise one would have to deal with absolute consumption values and a large market would compare bad to a small one.

Therefore the "benchmarking service provider" needs a certain set of key data like:

- Energy consumption E; e=E/E0 (normalized value)
- Area of the building A; a=A/A0
- Temperature T, t=T/T0
- Number of customers N, n=N/N0
- etc.

These values are generally subject to change over time. This might sound natural for measurement values like the temperature, but can also apply to attributes like the size of a building. So the JEVis System collects data values, equipped with additional attributes like a time-stamp, a validity and the like (see later in this document) and performs a number of more or less sophisticated calculations in order to get the desired benchmarking value. One simplified Example could be:

b=e/(ka*a+kt*t)

Here the energy consumption e is correlated to a weighted mix of area a and temperature t. The choice of the factors ka or kt demands heuristic insight into the particular benchmarking problem and are not subject of this work.

The JEVis System offers an open platform to implement data-related services of almost any kind, like the benchmarking example above. Core of the system is the JEVis database.

3. The JEVis Database

The main task of the JEVis database is to acquire data and information from various data sources and to generate services out of it by processing it. The data can be of some certain data type like polled measurement values, downloaded or received contents of data loggers, asynchronous alarm messages, configuration data, parameters, and the like. All that is stored in and processed by the JEVis database. There are three main types of interaction partners this database:

- Data Inputs (DI)
- Data Outputs (DO)
- Data Processors (DP)

These DB-peers attach themselves to the database and interact with it as depicted in Fig. 2.

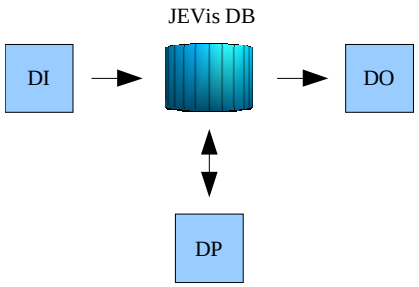

Fig. 2: Interfaces to the JEVis DB

Data Inputs are naturally responsible for adding data into the database. The way how this is done (interactively by some user, automatically via some network device, etc.) is encapsulated by the Data Input component. All necessary parameters needed by the Data Input component can be found in the database as well.

Data Outputs offer preprocessed data, that is stored in the database, in some specialized way. Typical examples for DOs are Web-frontends with statistical results or DOs that automatically generate faxes, printouts or E-Mails with reports.

DPs are the aggregating and processing parts of the system. DPs perform jobs on the data in the database, while the description and parameters for theses jobs are stored in the DB themselves.

DIs and DOs may interface to human operators, customers, to other databases and other data processing systems. Therefore they can be of more or less interactive character or take part in a client/server relation with other systems.

4. Data Acquisition and Dissemination

Feeding the database with data is, as already said, done via Data Input components. DIs can fall into one of three categories, namely

- server DIs,
- client DIs, and
- interactive DIs.

Server DIs are a server frontend for client data-sources. These clients (typically other computer systems or programs) can "upload" data to the database system via the server DIs by using some certain protocol. One server DI can serve multiple clients, while the JEVis database can serve multiple Dis.

Client DIs are also part of a client/server relation, but in contrast to the previous DI the client DI initiates the transactions (Fig. 3). It contacts some server application and requests data. The server is subsequentially supposed to provide this data in a response to the request.

Client DIs are triggered by the database itself or by some functional part of the database that is in need of some certain data that this particular DI is able to retrieve. Typically client DIs connect themselves to Fieldbus/IP-Gateways that provide historical and on-line measurement data. These gateways can be operated by

some IP-based protocol. See [2] and [1] for details on Fieldbus/IP gateways.

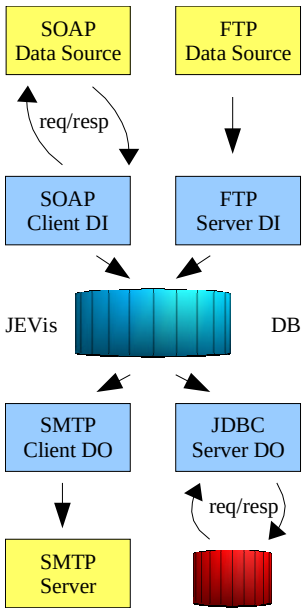

Fig. 3: Various DIs and DOs

Interactive DIs (Fig. 4) are typically forms, where authorized individuals can enter data into the database. In contrast to the first two types of DIs the interactive DI is interfacing a human being. Therefore things like usability, response-time and design of the user interface are much more an issue for the interactive DI than for the previous ones.

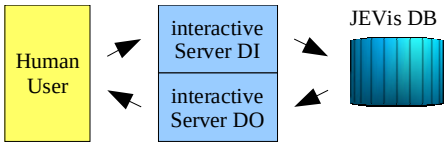

Fig. 4: Interactive DIs

The main type of interactive DI are certainly web-enabled forms, based on servlet/applet-technology or some comparable technology. Interactive DIs are in fact Server DIs but are treated separately because of the requirements resulting out of their highly interactive nature.

All configuration data for DIs, DOs and DPs like authentication data for used protocols, schedules for DPs or network addresses for client DIs is stored in the DB itself. So the database does not only store the collected data but also every other type of data that occurs in the JEVis framework.

5. Internals of the Database

SCADA databases have to store "engineering values", i. e. retrieved measurement values that are slightly more than plain numbers (see the ISO 16484 standard for an impression of the variety of datapoints that can be found in a modern building.). They have some real-world background that must be brought into an abstract form. DIs fulfill two important tasks in this context:

- Functional Abstraction
- Syntactical Data Abstraction

Functional Abstraction means, that the DIs take care about the technology-dependent details of data acquisition, like using a certain protocol for data retrieval, applying special addresses and searching hierarchies of data sources. See [3] for examples of and differences between automation networks and for examples how data is represented on control networks.

Syntactical Data Abstraction means, that data is brought into one common form and syntax. Such a common form was subject of research projects like NOAH (Network Oriented Application Harmonisation , an Esprit-Project of the European Union 1998-2000) [4] or various standardization efforts in automation control [5].

The main task of this syntactical data abstraction is to unify the engineering values which generally consist of two parts, namely

- the value itself (i. e. a floating point value), and
- value metadata.

The metadata make an engineering value out of a float value. They are attributes like physical unit, physical meaning, accuracy (absolute or relative), time-stamp of measurement, validity and quality. Sometimes there are even attributes that add semantical information to the value. Different data-sources might provide a different set of such attributes - some might offer information about the accuracy of the time-stamp while others might not or might offer a different way of describing this attribute. There are attributes that are apparently constant for one data-source like the physical unit while others differ from measurement sample to measurement sample like the time-stamp. Therefore the data structures inside the database must offer enough flexibility to meet the requirements and differences of the various data-sources that deliver data.

This abstracted data was the starting point of the whole system design. Questions like "what data does a service need and what data does a network device provide?" were the design principles for the database. Services and data acquisition must both interface to this data structures. In contrast to this, traditional design methods would be called

- bottom-up (starting from the automation nodes and network devices), or
- top-down (starting from the services).

The data-centered design as it was and is done in this project might be called "middle-out" in this terminology. It does not purely design a technology (like with bottom-up) that can be used to realize services and it also does not define services that must be implemented afterward. It defines a data-platform that tries to be flexible enough to work with various technologies and that is capable to serve various services.

The reason for this way is obvious: The technologies and services that will be used in some years are unknown now. Orienting on the currently used and needed ones

only could lead to a system design with serious disabilities. We expect more flexibility by having an abstract data layer as a basis.

- This data centered method consists of two central data structures:
- data point registry
- samples

The data point registry stores all data sources and sinks together with their respective attributes. Even parts inside the JEVis system that produce data (like calculation modules) are registered here. The samples table stores the samples which are actually the non-constant attributes of data sources. These two data structures are abstract enough to process all different types of services that appeared until now.

Another important aspect of the JEVis System is data security whose details, however, are not subject of this document. The basic parts of the security concept are

- secure data transport between all communication entities
- secure data storage
- powerful access right methods
- usage of smart card security
- secure customer frontends

See [6] for details on data security in automation technology and [7] for an introduction into smart card security.

There are two interesting branches of database technology that would be suitable for the JEVis system and might be of relevance in future. The first one is object oriented databases (OODB). Currently a “plain” relational database system is used to store all data samples, SCADA devices, configuration data, etc. This works fine as long as the system does not have to be flexible in terms of what to store. If the type and structure of the data samples are of a very different nature in future, it might be necessary to change the DB system (DBS) to an object oriented one. Even now it might happen that a particular data source can offer a set of data attributes that does not fit into the existing data structures. The current answer to this is to extend the structure of the DBS to store the respective data.

Another useful development in DBS technology are active databases (ADBMS) [8]. Active databases and especially active real time databases are preferably used for job shop scheduling, work flow management and production processes [9]. See also [10] for an introduction into ADBMS. Reactive behavior, triggered actions and the action scheduling features of ADBMS actually ideal for the JEVis services – the JEVis system also reacts on various changes in its environment.

The decision for Oracle was a commercial and a technical one. The chosen product offered a good package of support and professional maintenance tools. Additionally the JEVis system is intended to add several thousands of samples per second, which requires a high-performance DBS with a good scalability and possibilities for clustering.

6. Examples for Services

Envidatec's primary businesses are energy-management and energy-related data services for small and medium companies like statistics and data acquisition.

Energy management (EM) usually means demand-side-management (a.k.a load scheduling) of electrical energy. Modern EM systems use control networks to operate electrical consumers and to schedule their operation. There is even research on distributed algorithms and the usage of artificial intelligence to achieve fault-tolerant and flexible EM [11]. These systems are, however, still "local". They rely on on-line data, permanently available, provided by some automation network and processed by one node in this network like a PLC or some industrial PC. Our strategic direction, however, goes towards “Global Energy Management” (GEM).

GEM groups physically distributed energy consumers like the facilities of a supermarket chain to one community that is supposed to be collectively controlled. This can have economical reasons (a group of consumers appear as "one" customer on the liberalized energy market and are billed according to one load schedule) or physical reasons (the consumers are for instance supplied via some weak common part of the distribution grid).

GEM has to work via long distances, within whole regions and countries. The members of the GEM group are spatially distributed and the data channels within this virtual community of energy consumers are normally non-permanently available. One way is to connect a number of local EM-systems via dial-up IP connections that connect the individual EM-nodes on-demand. This principal difference to the local EM systems results in a completely different approach.

While local EM systems switch the energy consumers asynchronously “on-demand”, GEM systems have to plan and schedule the usage of energy. The historical behavior and energy consumption of the normally-offline GEM members are analyzed and interpreted by specialized GEM algorithms.

The basis for this is the JEVis DB that provides these historical measurement values and statistical properties of the energy consumers. The system has the possibility to learn and to estimate the future behavior of the consumers. Consequently it calculates schedules for the individual members that try to satisfy the global optimization goals [12].

GEM systems are a new and young research topic with only a few commercially available systems. The GEM application once again showed that the add-on service idea works with JEVis: existing data and shared infrastructure can be used for new services.

7. Conclusion and Future Work

The Project is now, which is the fourth quarter of 2002, at the beginning of the test phase. The data-centered design phase has proven its usefulness. Complex and complicated aspects of the system fit into the abstract design and allow a reasonable amount of flexibility. The choice of the relational DBS was oriented on the large amounts of data that are to be stored. The flexibility of OODBs and the functionality of ADBMSs were of secondary importance. This might change in future.

One very important part of the database is the structure of the data itself. Time series of measurement values and data derived from such time series need some special treatment, similar to what was done in [13], though JEVis will try to stay within the world of standard SQL and standard universal databases. The structure, transmission and representation of measurement data was and is also subject of research in CORBA (common object request broker architecture), SOAP (simple object access protocol), OSGI (open services gateway initiative) or OPC (object linked embedding for process control) which influenced and will further influence the JEVis project. Further contributions to the design of the database will come from research done in related domains like energy distribution and classical SCADA [14, 15], especially the usage of UML (universal modeling language). Such syntactical and semantical aspects of the data will be subject of further research.

Currently, the typical user of JEVis is a utility company, a supermarket chain or a similar multi-site customer that uses JEVis as an analysis and SCADA and/or on-line facility management tool. The next steps and services will lead to a wider bandwidth of usage and to further research activity in the fields of distributed computing, replication and availability of data.

References

1. M. Lobachov & P. Palensky, Bringing Energy-related Services to Reality, *Proceedings of the International Conference on Energy Economics (IEWT01)*, Vienna, Austria (2001).
2. T. Sauter & P. Palensky, A closer look into Internet-fieldbus connectivity"; in Elektrotechnik & Informationstechnik (e&i) 117, Austria (2000).
3. Loy, Dietrich & Schweinzer (Eds.), *Open Control Networks, LonWorks/EIA 709 Technology* (Kluwer Academic Publishers 2001).
4. C. Raibulet & C. Demartini, Mobile Agent Technology for the Management of Distributed Systems - a Case Study, *Proceedings of TERENA Networking Conference 2000*, Lisbon, Portugal, 2000.
5. *LonMark Application Layer Interoperability Guidelines, Version 3* (LonMark Interoperability Association, USA, 1996).
6. T. Sauter & P. Palensky, Security Considerations for FAN-Internet connections, *3nd IEEE Workshop on Factory Communication Systems*, Barcelona, 2000.
7. R. Bright, *Smart Cards - Principles, Practice, Application* (Ellis Horwood Ltd. 1988).
8 ACT-NET Consortium, The Active Database Management System Manifesto: A Rulebase of ADBMS Features, *ACM Sigmod Record* 25(3): 40-49, 1996.
9 G. Kappel & W. Retschitzegger, The TriGS Active Object-Oriented Database System - An Overview, *ACM Sigmod Record*, 27(3):36-41, 1998).
10 A. P. Buchmann, Architecture of Active Database Systems in Norman Paton (ed) *Active Rules in Database Systems*, (Springer-Verlag, New York, 1998).
11. P. Palensky & M. Gordeev, Demand Side Management by using distributed artificial intelligence and fieldbus technology, *Proceedings of the Intelligent and Responsive Buildings Conference (IARB99)*, Brugge, Belgium 1999.
12. P. Palensky, *Distributed Reactive Energy Management (*PhD. Thesis, Vienna University of Technology, Austria 2001).
13 A. Wolski, J. Kuha, T. Luukkanen & A. Pesonen, Design of RapidBase-An Active Measurement Database System, *Proceedings of the 2000 International Database Engineering and Applications Symposium (IDEAS00)*, 2000.
14 R. Podmore, D. Becker, R. Fairchild & M. Robinson: Common Information Model - A Developer's Perspective, *Proceedings of the 32nd Hawaii International Conference on System Sciences*, USA 1999.
15 S. Poulsen, Electrical Network Supervisor Status Report, *Proceedings of 4th Workshop CHAMONIX*, Cern, 2001.