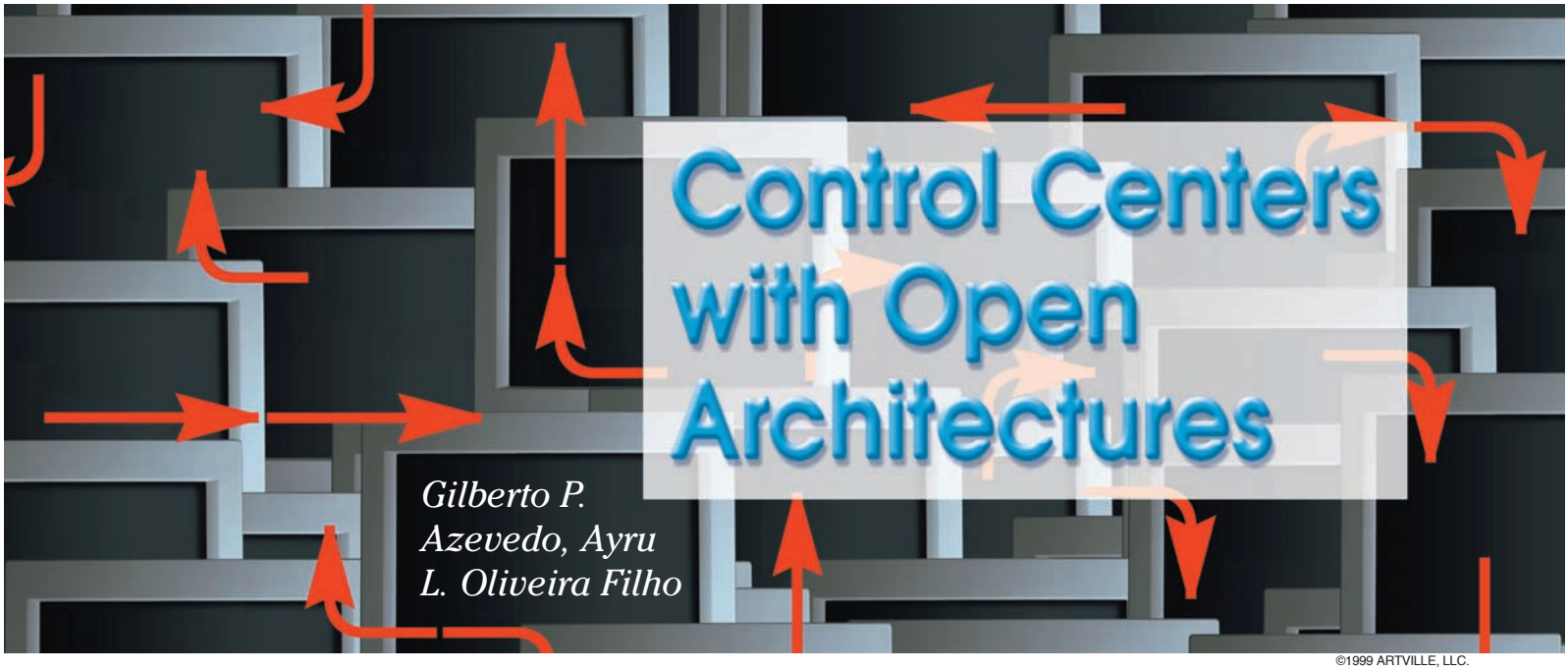


In the software development area, as in most fields of the computer industry, new technologies are trumpeted as revolutionary solutions almost daily, just to disappear silently some time later. This was not the case with open-architecture energy management systems (EMS). About 10 years after their conception, they have proven to be a successful technological approach. But this does not mean that all problems have been solved; in fact, this is a dynamic research area, in continuous evolution and still raising challenges for the near future.

Evolution of Energy Management Systems

The first generation of computerized power system control centers appeared in the 1970s and was obviously based on the computational architectures available at that time. The very limited performance of the computers (by today's standards) implied that software had to be intensively optimized and intimately connected to the operational system and even to the hardware. The result was that applications, person-machine interfaces (PMI), databases, operational systems, and hardware were all interconnected.

This generation of control centers has been successful in enhancing the quality of supervision and control of power systems. The advantages and the potential of using computers became clear, as power systems could operate closer to their limits without increasing operative risks. Investments in expansion could be postponed, and more complex operative configurations became manageable.

But the computational architecture of that generation also included some hidden problems. Computers and software soon became obsolete, and the dynamics of the market eliminated many of their original manufacturers. Meanwhile, to keep the control centers operating properly, utilities needed to replace defective components, add new functionality, expand databases, and increase the computational load. But the deep connection among software components and hardware made it very difficult, and often impossible, to update or expand both hardware and software. After a few years, some utilities had control centers based on obsolete hardware and software. Maintenance was increasingly difficult and expensive, with low performance and decreasing reliability. Those control centers were not able to follow the evolving operational requirements.

With Linux, the Web, and Java, portability is now easier to achieve than it was 10 years ago, but by no means it has become a less important goal

Those problems were common to other sectors of the industry. In the 1980s, the appearance of low-cost but powerful computers, together with the evolution of networks, allowed for the emergence of distributed processing. Networks of workstations and personal computers quickly replaced the old mainframes. This decentralization led to a rush for standardization in different fronts, because standards were essential to connect the various network computers. De facto standards soon appeared: C programming language, X-Window, UNIX, and TCP-IP are some of the best-known examples.

To avoid the problems of the first generation of EMS/SCADA systems, most control-center software developers adopted the generalized use of standards. This approach, together with the following design methods, led to software architectures for control centers that became known as *open architectures*.

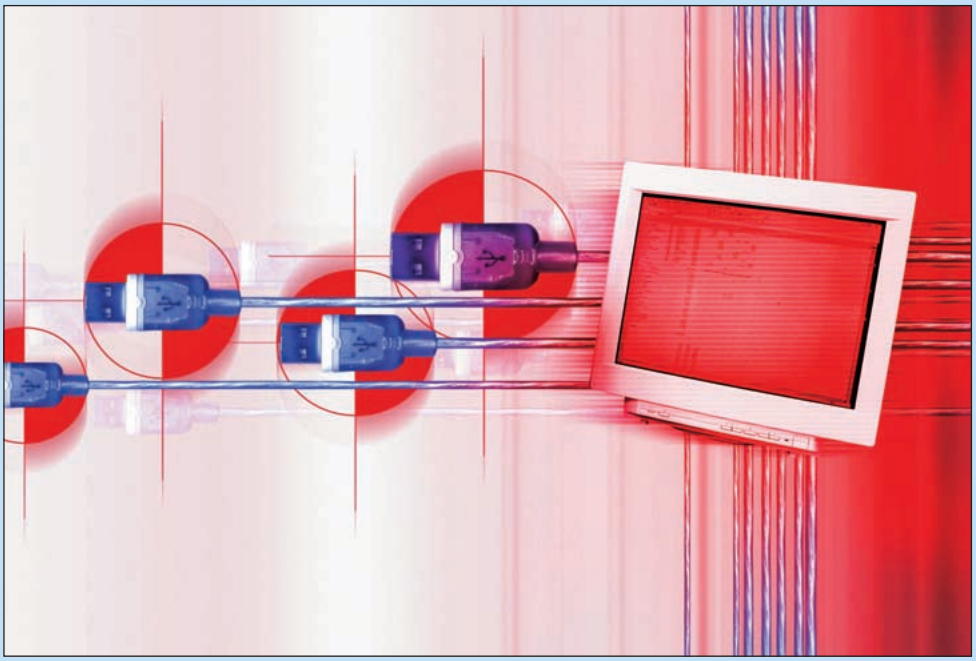
G.P. Azevedo and A.L. Oliveira Filho are with CEPEL, Rio de Janeiro, RJ, Brazil.

ISSN 0895-0156/01/$10.00©2001 IEEE

Open Systems: More than Just Software

Almost 10 years ago, when CEPEL started to design and develop SAGE, the commitment to open architectures was firmly defined. It was even included in the product's name: SAGE is an acronym for Sistema Aberto de Gerenciamento de Energia, an expression in Portuguese that means open energy management system.

The enforcement of those concepts proved to actually bring important benefits to users. Scalability has been especially useful in large utilities, by eliminating the need to maintain different SCADA/EMS systems in different control levels. The connection between those levels became straightforward. Portability and interoperability are a reality, and today SAGE software is able to run in virtually any UNIX platform (including PC and RISC architectures), which can be mixed in the same installation. Modularity has made the inclusion and evolution of software modules a routine task. Most of SAGE's installations are recent ones, but, even in this circumstance, their expandability characteristics have been necessary in some cases.



It also became clear that openness should not be only a software design guideline. The commitment to openness must embrace all phases of a product's life: design, development, training, implementation, and evolution. The benefits and drawbacks of direct contact between end-users and the development team are discussed in the article, but note that it has been important to fine-tune the product and to indicate new evolution guidelines. Users are also able to develop and include their own applications in real-time or, preferably, in the offline environment (the real-time environment is obviously much more sensitive to user errors).

The implementation of new control centers is another branch where this generalized open approach is emphasized. In many cases, users have been trained and stimulated to command and execute partially or totally the implementation of new installations, from database and screens development to software and hardware configuration. The intensive participation of users in implementation and testing led to a mean rate of a new control center each 3-4 weeks in the last 2-3 years. Most of SAGE's almost 70 installations are relatively small SCADA systems controlling a few substations or plants, but there are also several medium-size and large EMSs, including the Brazilian National Grid Control Center.

The good results of these approaches reinforce our view that, if openness is actually to be considered an important goal, it should not be restricted to software.

Life Cycle of Open Control Centers

Accompanying the long life cycle of a software product as large as an EMS/SCADA is a privilege that not many software developers have. At CEPEL (a Brazilian power system research center, Centro de Pesquisas de Energia Elétrica, in Rio de Janeiro), we have had the opportunity to participate in the development of SAGE (*Sistema Aberto de Gerenciamento de Energia*) since its initial design, in 1991, until the coordination of many implementation projects. Open architecture concepts were just starting to spread at the onset of the design phase, but they have been strictly enforced since the very beginning.

This rich experience taught us valuable lessons. The first one was that close contact between users and software developers is essential to fine-tune the product. Some sophisticated features that software developers supposed to be very useful have been ignored by end users; conversely, users often require the implementation of new features (often simple ones) that had not been even imagined by developers but are considered very important. Giving users access to software developers certainly causes some inconvenience (frequent interruptions, filtering of requests, change of priorities, etc.), but the benefits are clearly more relevant.

The most interesting lessons were derived from the use of open architecture concepts in every step of the design. In the beginning of the 1990s, those concepts were not yet completely clear, and the advantages and

risks of their use were not totally evident. But in a few years, they proved to be essential to extend the life cycle of the product.

Open architecture concepts (portability, interoperability, expandability, modularity, and scalability) are still evolving and will play an important role also in the design of the next generation of EMS/SCADA systems, which is expected to emerge in a few years.

Portability

In the context of control centers, *portability* refers to the possibility of running the same software on different hardware/software platforms. This feature eliminates one of the greatest problems of the previous generations of EMS/SCADA systems: the dependence on specific vendors. If a system is portable, it will be able to run on different platforms, from different vendors. This brings some important advantages:

- It is possible to choose, from a large range of alternatives, the equipment that is best suited for each specific application, considering cost, robustness and performance. Otherwise the utility could be forced to select from a much smaller range of alternatives provided by a specific vendor.
- Independence of proprietary hardware and software. Hardware and software vendors may suddenly leave the market, increase the costs of services and products, or just discontinue a certain product. Sometimes this has meant the end of the possibility for a control center to evolve.

Portability is directly dependent on the use of a range of standards, but those associated with the operating system are especially important. Some SCADA systems are based on versions of the Microsoft Windows operating system. Of course, they are not actually portable, but their dependence on specific vendors is limited to the operating system. In the recent past, some developers chose other proprietary operating systems, either because of previously accumulated experience or ease of use, but they often have experienced problems like those mentioned earlier. Since an EMS/SCADA system is expected to have a life cycle of more than 10 years, dependence on specific vendors does not seem to be a safe approach.

One good solution to this aspect of portability has been the adoption of the UNIX operating system. UNIX has many different versions, but only its standardized features should be used. Enhancements provided in specific versions of UNIX, no matter how attractive and comfortable they are, should be totally avoided. In the near future, the foreseeable adoption of Linux on nearly all hardware platforms may make this aspect of portability easier. Of course, total independence of the operational system would be the best solution, but the possible benefits don't seem to compensate the costs involved in such a strategy.

Portability is directly dependent on the use of a range of standards, but those associated with the operating system are especially important

Another important aspect of the portability is related to the programming languages used. Widespread languages should be preferred, and, programmers should resist the temptation to use enhancements provided by specific compilers that are not part of the standards of the language. Until a few years ago, C, C++, and even Fortran have been the languages of choice, but now Java (which is inherently portable across most platforms) is proving to be capable of replacing them with great advantages related to portability in a wide range of applications.

Portability must also encompass graphical interfaces. Until a few years ago, in the context of control centers, this usually meant that interface developers should use X-Windows and Motif. Web technologies are changing this picture dramatically, because common Web browsers may easily provide cross-platform access to graphical interfaces.

Databases are another area that affects portability. The use of SQL (avoiding as much as possible all nonstandard features) has eased the development of software capable of connecting to different commercial databases.

Our experience shows that portability must be a permanent goal. If it is taken seriously enough in all phases of software development, the migration of the EMS/SCADA software to a new platform may be painless. Nevertheless, it will rarely be an easy task.

With Linux, the Web, and Java, portability is now easier to achieve than it was 10 years ago, but by no means it has become a less important goal.

Interoperability

If an EMS/SCADA has been developed with full attention to portability issues (especially those associated with the strict use of standards), it is likely to be able to go one step further in this direction and reach interoperability.

Interoperability is the ability to run software modules (identical or not) on different platforms, in the same network, at the same time, all communicating and interacting with one another. That is, different hardware, operating systems, and software modules can coexist in the network, all being part of the same EMS/SCADA solution. A nice feature of systems with interoperability is the possibility of running exactly the same software on different hardware platforms, in the same network.

Table 1. Control center evolution		
Past	**Present**	**Future**
Centralized	Client/server and distributed	Distributed, based on components and agents
No use of standards	Use of operational systems, graphics, protocols, and programming language standards	All present standards plus database model and APIs for easier integration of applications
Vendor dependent (hardware and software)	Hardware vendor independent	Hardware and basic software (operating system, middleware, and database) vendor independent
No interoperability potential, not portable, and hardly expansible	Portable, expansible, modular, scalable, and interoperable	Completely open, with component and agent-based interoperability
Semigraphical PMI	Full-graphics PMI	Full-graphics, Web-based PMI, with advanced visualization resources
Rigidly organized interaction with other control centers belonging to the control hierarchy	Controlled interaction with other control centers, other sectors of the utility, and other entities	Intensive interaction with nonsubordinated external players in an open environment
Incipient information exchange with external entities	Some degree of information exchange with external entities	Intensive information exchange with external entities
Autonomous, with decisions made based on information gathered and processed by the control center itself	Autonomous, with low dependence on external data	Semiautonomous, with high dependence on external data
Technically oriented operation	Importance of nontechnical criteria starts to increase	Economic, commercial, and legal criteria become highly important
Life cycle dependent on hardware	Life cycle dependent on basic software standards	Life cycle dependent on components interface standards

Interoperability is not easy to achieve. Very few EMS/SCADA systems actually support it. But its benefits are highly relevant: utilities can mix different hardware platforms and thus are free to buy the solution with the best cost-benefit relation every time the system needs to be upgraded or expanded.

This implies that the true benefits of portability only become evident if interoperability is also achieved. Otherwise, utilities that need to upgrade or expand their systems will stay attached to the original hardware vendor or will have to replace the whole hardware platform. This is much like it was in the first generation of control centers.

The interoperability between different flavors of UNIX is easier to achieve than between the UNIX and Windows worlds, since, in this case, it is not possible to rely on the set of standards used to promote UNIX interoperability. Common object request broker architecture (CORBA) standards are a possible solution for such integration.

Another effort toward interoperability (and modularity too) is the definition of a common data model for power control systems. The objective here is to develop a distributed EMS/SCADA based on a standardized data model, so that it would be possible to aggregate third-party functions to the existing system. In this way, EPRI has proposed the Common Information Model (CIM) that is being jointly standardized by IEC TC57 WG13 and WG14.

The future poses great challenges to interoperability. It will not be restricted anymore to the EMS/SCADA software running on a local network. Transformations in the power industry increasingly will require that EMS/SCADA software interact with other software of the utility, its partners, or other players. Some software modules will have to deal with issues like negotiation of information and services, security, external communication, and aspects of uncertainty related to open environments, but they will have no direct control over the external players. Agent technology is a promising approach to the development of software able to interoperate in open environments, but some important subjects like security, reliability of external information, and communication languages still require careful research.

Expandability

The control center software must be able to efficiently support the expansion of the power system of the utility. Both the growth of the power system and the inclusion

of new software functionality must be easily accommodated, while keeping the performance at acceptable levels. In short, control center software must have *expandability*.

In the past, this usually meant that the system should be overdimensioned since its initial design, because it would be difficult to add new processing power later. Large investments in hardware had to be made immediately during the initial phase of the control center, only to provide enough processing power for possible future requirements. This obviously had a very negative impact on the initial costs of the control center. But, if this provision was not made, utilities would be totally dependent on the original hardware vendor for any necessary performance improvement. Without competitors, the vendor would be free to define the price of services and products.

Portability and interoperability together make expandability a problem that is relatively easy to solve in open architecture control centers. Investment in idle hardware is not necessary, because processing power can be added or upgraded later as needed. Distributed processing in networks of heterogeneous hardware allows for the safe addition of new computers or replacement of obsolete ones.

If a certain investment in hardware can be postponed for 2 or 3 years without affecting the performance of the control center, computer market dynamics provides that costs will be significantly lower when the acquisition is finally made. This implies that hardware costs are reduced, replacement of obsolete equipment becomes simple, and the control center can be expanded as needed at low cost and easily.

The importance of expandability to the utilities will certainly not decrease in the future. Fortunately, we can expect that foreseeable improvements in portability and interoperability will make expandability a goal that will be gradually easier to achieve. But portability and interoperability are not sufficient to achieve expansibility. All the EMS/SCADA software must be designed to allow for future expansions. The limits included in the software (maximum database size, for example) must depend essentially on the available processing power. The software must also be designed to accomodate growth in the number of clients accessing control system data, the number of computers attached to the real-time network, and expansion of the power system being controled.

Modularity

Modularity is related to the ability of modifying the EMS/SCADA software with a negligible impact on software components that are not directly involved. Software modules can be added, modified, replaced, and, in many cases, even removed without affecting other modules. This can be achieved by a careful design of system architecture, in a macro level, and enhanced by the use of object-oriented software development techniques.

The first generation of EMS/SCADA software was forced to use deeply connected (and thus nonmodular) code. Efficiency concerns often required that application code included explicit references to databases, operational system, PMI, and to other software modules. The resulting software was capable of running well on computers of that time, but the price of this success was high; it was often very difficult to make major changes in the EMS/SCADA software. Evolution became a problem, and obsolescence soon reached those EMS/SCADA systems.

In the 1980s, new inexpensive but powerful hardware platforms eliminated those constraints and allowed for the development of truly modular code. A very successful approach to modularity has been the decoupling of databases, PMI, and the other software modules. Some EMS/SCADA systems like SAGE developed general-use interfaces to visualize and interact with information stored in the real-time database. Those interfaces are decoupled from the applications that generate the data. This means that applications can operate on data without any concern with their use, and interfaces can display data regardless of the application that originated them. Modularity can be enhanced if applications use a standard data model for power system software, like EPRI's CIM.

Modularity made the addition of new modules become a routine task in today's EMS/SCADA software. New functionality can be added as necessary, and changes in existing software usually became straightforward on what relates to the impact on other modules. This made software development and maintenance a much easier task than it was before and largely extended the lifecycle of the software.

The future will take modularity some steps further. Some knowledge that is implicitly shared among different modules, like nonstandard database models, still represents obstacles to the improvement of modularity. Agent technology can provide help; converting modules into autonomous agents that communicate and exchange information using an appropriate language (the only implicit hypothesis being that agents can speak that language) may eliminate those last obstacles. As happened before, hardware and network evolution will compensate the increase on computational and communication load.

Scalability

Scalability is the ability to run essentially the same software in control centers of very different sizes and scope. The same database, user interfaces, communications software and other support modules are used either in a large EMS or in a small SCADA system. The main differences are simple database model customization and the selection of application modules that will be plugged into the system.

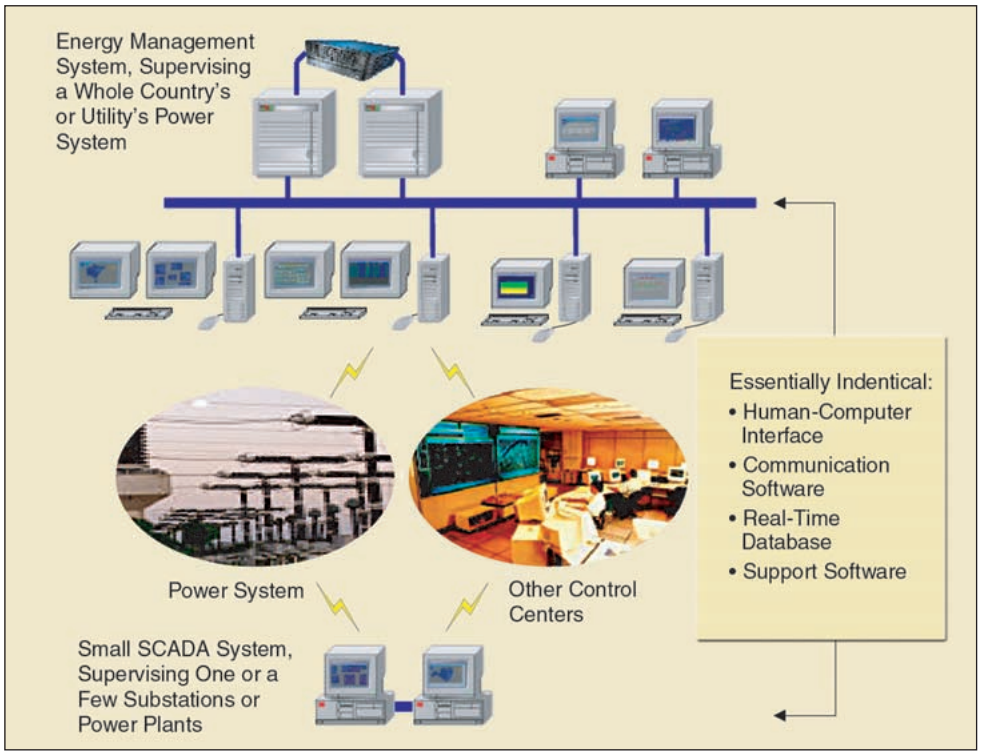


Figure 1. Scalability enables the same software to be used in control centers of very different sizes and scope

An EMS will require more complex database models and advanced network analysis modules that are not necessary in smaller SCADA-based control centers (Figure 1).

Scalability greatly simplifies maintenance problems and reduces training requirements. The same support team is able to deal with all control centers. This is especially important in large utilities, which may have up to three levels of control centers in their control hierarchy.

It is easy to see that scalability depends directly on modularity and expandability. The modules that are plugged into the basic software, together with the database model, define if the system is to be used in a small and simple or in a large and sophisticated control center.

If an EMS/SCADA is scalable, this almost certainly means that it is truly modular. Our experience is that scalability is possible and not hard to achieve, provided that modularity concerns are taken seriously in the design phase of the system. Scalability also proved to be useful and to have a significant positive impact on the long-term costs of support and maintenance of control centers.

The ever-increasing power system market rush toward efficiency gains and cost reduction will reinforce the importance of scalability. The economy that it brings comes without any drawback and derives exclusively from efficiency improvements.

Open Architectures: The Future

The importance of power system control centers increases in competitive scenarios. The centers are essential to safe and lucrative operation by enabling a more efficient use of existing power system resources. The centers deal directly with the main business of the utilities.

The new power system control scenario clearly shows that, in the near future, change will be a permanent state. New requirements, rules, laws, tools, businesses, opportunities, and problems will appear constantly. Those changes will influence the power system operation and consequently the EMS/SCADA systems. If the centers are not able to evolve fast and within acceptable costs, they will become obsolete in a short time.

But control centers cannot be discarded and replaced frequently. The implementation of a control center usually requires large investments, both direct (acquisition, installation, and customization) and indirect (training, adaptation to the existing infrastructure, and testing), and takes at least a few months.

Open-architecture control centers provide the solution to accommodate the necessary changes and to keep the software compatible with the evolving requirements. If in the past most changes have been related to the transformations in the computer industry, now they derive from the power system market. The revised principles of open architectures for control centers discussed in this article provide answers to those challenges too. Open architecture allows for smooth evolution and promises a long life cycle for control-center software. This will be fundamental to provide the conditions for successful actuation of utilities in the new competitive and dynamic scenario.

For Further Reading

G.P. Azevedo, B. Feijo, and M. Costa, "Control centers evolve with agent technology," *IEEE Computer Applications in Power*, vol. 13, pp.48-53, July 2000.

Biographies

Gilberto P. Azevedo received his B.Sc. in electrical engineering from PUC-Rio, M.Sc. in electrical engineering from COPPE-UFRJ, and D.Sc. in computer science from PUC-Rio. He has been with CEPEL since 1982. His current interest areas include power system operation and control, PMI, power system restoration, and artificial intelligence, with a special interest in agent technology. He may be reached by e-mail, gilberto@cepel.br.

Ayru L. Oliveira Filho received his B.Sc. in electrical engineering from UFJF, M.Sc. computer science from IME-RJ, and D.Sc. from in computer science from COPPE-UFRJ. He has been with CEPEL since 1988. His current interest areas include power system operation and control, distributed computing, communication protocols, and databases for power systems. He may be reached by e-mail, ayru@cepel.br.