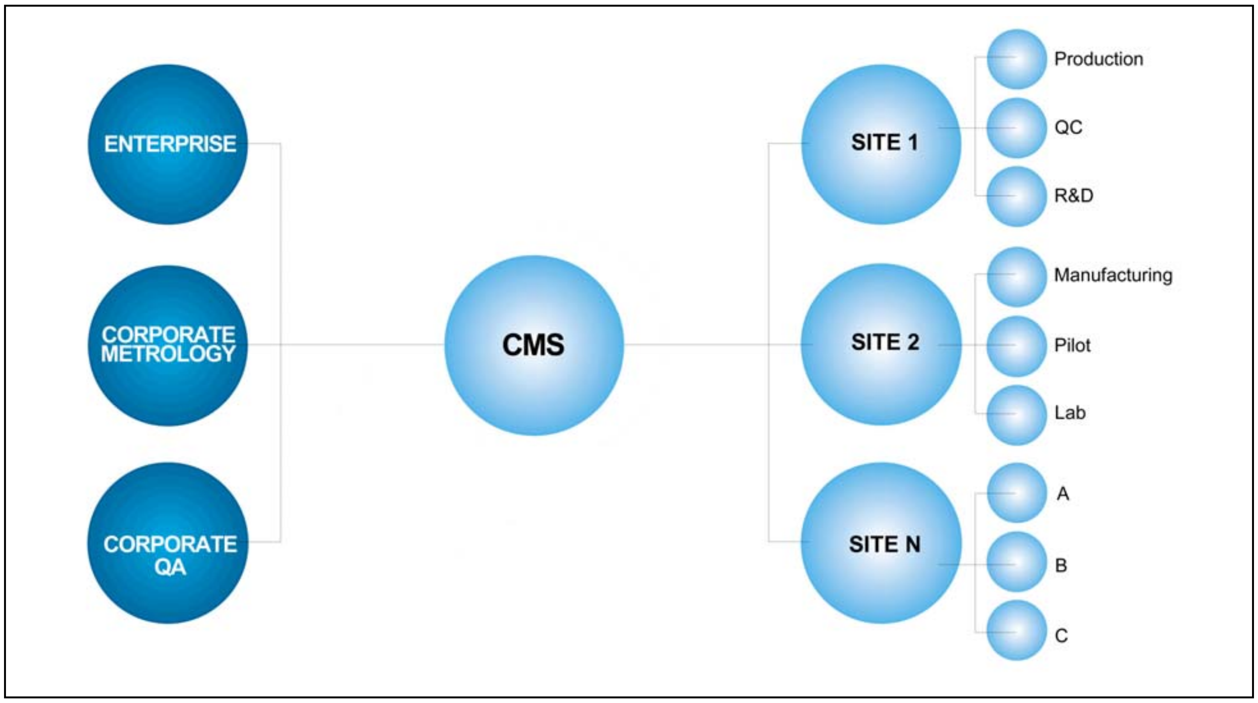
Architectural Benefits of Distributed Enterprise Calibration Management Systems

Speaker: Jim Erickson
Author: Jeff Rimland, Jim Erickson
Blue Mountain Quality Resources
208 West Hamilton Avenue, State College, PA 16801
Phone: (814) 234-2417; Fax: (814) 234-7077
www.coolblue.com

Measurement was once a localized process. The object to be quantified was compared to a standard and the results were written down for future reference. Today, this model is no longer applicable. The two chains of calibration, Management and Standards, both span the globe. Data that was once shared between dozens of people within an office must now be instantly accessible to hundreds of people spanning thousands of miles. Although standards may be used from anywhere on the planet, traceability must be instant and dependability must be absolute.

The Calibration Management System (CMS) must serve as a central hub between management, corporate metrology labs, corporate QA and the calibration sites. Just being connected is not enough. Interconnectivity and interoperability must be seamless and security controls must be precise enough to allow access to only the exact bits of data necessary, whether you are connecting to the system via LAN, WAN, VPN or the Internet, and regardless of your choice of platform. Whether you are working from a desktop, laptop or palmtop system, the CMS must be a seamless extension of the way you work. (See Figure 1)

Figure 1: Enterprise layout.



N-Tier Architecture

It takes a different kind of thinking to turn the dream of a system meeting these goals into a reality. The first step toward meeting this goal is letting go of the antiquated methodologies that are currently being implemented by many CMS designers. Traditional fat-client implementations simply cannot keep up with the growing need for global scalability and control. Even the simpler client-server paradigm falls short. Where these methods often fail, a web based, n-tier architecture excels for facilitating the worldwide measurement, distribution and sharing of metrology data.

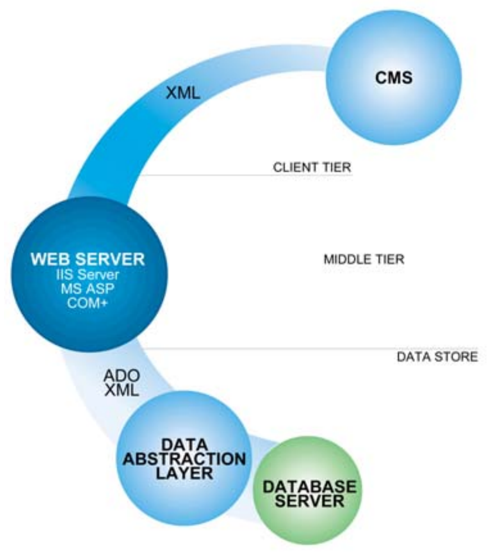
A web based, n-tier approach distributes the necessary user interaction, computation and storage tasks between the layers of the architecture. Although some latitude exists in the exact number and structure of layers, the system is typically broken up into a Client Tier, a Middle Tier and a Data Store. The Data Store is further segmented into a Data Abstraction Layer (DAL) and a Database Server. (See Figure 2)

The Client Tier of the system is responsible for interaction with the user. In the past, all users of an application saw the same interface. Today's users access their data from a variety of devices with differing screen sizes and input methods. The Client Tier must be accommodating regardless of what bridges the user to the system.

The Middle Tier, which is also known as the Business Logic Layer, is where the data is interpreted and where business rules are applied. Certain types of security and access checks are also done at this level. The Middle Tier is considered the "brain" of most n-tiered systems.

The Data Store is delegated the task of storage and retrieval of data. Once the choice of Database Server has been made, a DAL is created to provide the interface between the Middle Tier and the database itself.

Figure 2: Architecture design.



Implications of n-tier Design

On initial inspection, it would appear that this type of segmentation would add nothing but complexity to a system that must function both reliably and responsively. Although it is true that this approach can add complexity, its benefits in both performance and implementation time make up for the increased complication of the architecture.

Any type of architecture can be used to create a software-based CMS if no changes will occur after the software has been installed. However, it is most likely that changes to the volume of data, user interface or database server will occur at some point. In today's global metrology environment, change is daily and response time is critical. This is where the true beauty of an n-tier approach comes to the surface. With a conventional application, changes like adding a browser-based interface or switching from MS SQL Server to Oracle can take almost as long as the original implementation itself. Nearly the entire code base must be touched in some way, not to mention the testing and validation phases.

If n-tiered methodologies are used, adding a browser interface is as simple as creating ASP pages that can communicate with the original middle-tier objects. No additional changes or testing is required aside from that directly related to the added functionality. As long as the new functionality complies with the existing interface, it can simply be plugged in and switched on.

Switching between database servers can be handled in a similar manner. If an n-tier application has been written for MS SQL Server, for example, switching over to an Oracle database is a matter of modifying just the Data Store layer instead of having to modify, test, and re-validate the entire application.

Scalability and ease of utilization of emerging hardware technology are two of the most exciting benefits. If the number of records stored becomes too great or if the data needs otherwise exceed the capabilities of a single database server, simply plug in as many database servers as the situation requires. Likewise, if more users need to interact with the system, simply add Client Tier components. If wireless network technology is needed, just plug it in. Scaling essentially happens automatically. As new technologies such as retinal scan-based identification or other biometric devices become more feasible, n-tiered architecture offers the quickest path between the idea's conception and its implemented reality.

COM and the Middle Tier

Microsoft's Component Object Model (COM) is a perfect fit for handling the complexities inherent in the business logic required for measurement and calibration related tasks. The main tenant of COM is using software objects to represent objects in the real world. This fosters logical software development practices, reuse of code (which reduces bugs, complexity and code size) and easy upgradeability.

XML

Working with an n-tier component based architecture makes many aspects of a development project simpler. To the uninitiated, however, it can also cause some problems. With a traditional fat-client implementation, data is typically transferred directly between the application and the database server. This is very different from how things work in the n-tiered world. The n-tier methodology introduces the need for a data format that can be used to send sensitive data across insecure and error-prone networks; a format that needs to be readable by human eyes, yet capable of being quickly obfuscated beyond the recognition of man or machine. This data format is the Extensible Markup Language (XML).

XML, which shares the last two letters of its acronym with the Hyper Text Markup Language (HTML), is actually very similar to its ubiquitous cousin. HTML has proven itself as an effective method of transmitting information over the Internet. Its tag-centric structure is well suited for carrying the data needed to properly display items in the user's web browser. It is text-based, which means that it can be read or edited by any word processor. This also means that, unlike their binary counterpart, text files can be read, as-is, on any hardware or software platform. The limitation of HTML is that its tags can only be used to contain a specific type of information, which is essentially the formatting component of a web page.

XML breaks free of these limitations by allowing an unlimited combination of user-defined tags to contain anything from simple lists of data to complex and deeply nested heterogeneous hierarchies. These files can be encrypted and checksum-encoded to allow only authorized users to view or edit data that is essentially hidden in plain sight. XML schemas can provide a built-in validation step to ensure that the data is in the correct format; one more check to assure that your business keeps flowing smoothly. Furthermore, for all of its merits, XML is really just a text string. Because of this, it eschews firewall and binary compatibility issues that were the bane of earlier client-server and n-tier applications.

Summary

Today's business world is full of technologies that are implemented for their own sake. Companies blinded by the shine of their latest high-tech hammer begin to see every new challenge as a nail. On the other hand, certain applications are a perfect match for the latest benefits that recent paradigm advancements have to offer. Distributed Enterprise Calibration Management Systems and n-tiered, XML-based software architecture are a perfect example of two emergent trends whose paths are converging.