Generations of Technology in Industrial Automation Software

The 1980s brought the early adoption of the PC platform in automation systems and the first large projects, using the PCs to communicate with PLCs. In the following decades, supervisory and control systems technology evolved, creating several generations of software tools and automation products. A generation means an evolutionary step and a new platform, with a total change of the programming methods, user interfaces and paradigms, going beyond than the merely incremental improvements that are made during the maintenance life cycle of products; a new generation means renewing the internal architecture.

It is simple to identify those evolutionary steps when making historical analysis, but it is not as easy to have that picture clear when we are in the middle of one transition, as it is happening now. Similar to the transition from VAX/VMS platform to the PC platform in the 80s, or changing from DOS to Windows in the 90s, or the transition from CD media to MP3 in the past decade, we are now in the transition to a new generation of supervisory systems and industrial management software tools.

Among the factors that led to the advent of the new generation of systems, are:

- The increase of speed on communication networks and processing capacity.
- Higher integration between the shop floor and the corporate environment.
- Distributed access and the consequent demand for increased security.
- New computing environments such as.NET Framework.
- New programming languages like C # and VB.NET.
- Cloud computing and the service as the core product.
- Collaborative and remote engineering.
- New User Interface paradigms and Design concepts, influenced by Apple.
- New forms of interaction, such as tablets, smartphones and 3D models.
- Use of open standards for storage, data exchange, graphical files.
- Greater integration of systems of control with supervisory systems.
- Smart-grid deployment in power management.
- More customized production, with shorter production cycles.
- Production chains distributed in different suppliers and locations.

The migration through this transition is not achieved with minor improvements over existing platforms; it also requires new platform architectures, new software kernels, new concepts to be embraced and a new generation of technologies to implement them. In this article we will explore these new concepts and technologies, the new user interfaces and communication,

safety requirements and changes in corporate environment; factors that shaped the creation of a whole new generation of real-time distributed software for supervisory and control applications.

INTRINSIC SECURITY APPLIED TO SOFTWARE

One feature that remains unchanged, is the operational stability as the main requirement (* 1). The mechanisms related to increase the guarantee of stability are among the main architectural changes made possible by new technologies.

In field instrumentation, security is not solely guaranteed by internal procedures or manufacturers' warranty, but also and primarily by the system architecture, using voltages and currents which are "intrinsically safe" in the environment that instrumentation will operate.

The same concept applies to software. The previous generation of technology used C/C++, pointers, several modules sharing the same memory area, direct access to hardware and to the operating system resources, necessary procedures vis-à-vis computers and languages available at the time. However, we consider these to be intrinsically unsafe.

The new generation of software uses computational environments, such as the .NET Framework or JAVA, where processes are natively isolated between themselves and the operating system, regardless of the programmer, allowing better use of computers with multiple processor cores and ensuring greater operational stability, even in the face of potential drivers and hardware errors, or failures on individual modules of the system.

Another change in course, aiming for enhanced safety, is the replacement of the Scripting languages used on the software tools for project customization. The previous generation used proprietary scripts or interpreted languages, such as VBScript, VBA or proprietary expressions editors; the new generation relies on more modern and compiled languages such as C #, VB.NET, with object-orientation and more projection on the execution. With interpreted languages, you cannot do a complete code validation during the development stages, only when execution passes by the code that the final verification is made, which means many problems are only possible to test when running the project, not during the engineering configuration. For example, using variables without initialization, types and inconsistent parameters, those errors on interpreted languages are only identified during the execution. In addition, to increase the efficiency of the project development, the main reason why this concept is so important is to ensure operational safety. A typical project may have hundreds to thousands of possible execution paths for the code, the testing scenarios cannot test all those paths by exhaustion running all the possible cases, the ability to detect potential errors during the engineering and the ability to recover and isolate the errors during runtime are key elements for safety and operational stability which are only possible by migrating the legacy interpreted scripts to the new compiled and managed languages.

COMPLETE PROJECT CYCLE

Another concept of this new generation of supervisory systems is the focus on the full project cycle, not just on the software tool itself, but providing resources for all project phases, which includes: initial engineering specifications, project configuration, testing, field installation and maintenance. Each project phase has its own requirements and new software platforms shall provide tools to help on each of those phases.

Technology selection: execution threads and module processes should be independent and isolated from each other; scripting should use modern language with compiling validation and managed execution environment, secure web clients with no need to install legacy Active-X components (which are a flaw on the network security and requires operating system privileges), use of new technologies and standards, such as WPF, WCF, XAML and the consolidated ones such as SQL.

Project Configuration: enabled for engineering collaboration (multi-user and multi-project) using local, remote or cloud computing projects on the way; tags definition on the control systems and supervisory systems sharing a common unified list; native change management and version tracking; enhanced validation during the configuration.

Installation and pre-operation: native tools for testing, diagnostics, performance profile, project verification and publishing. Concurrent and remote access for the project configuration; project configuration to be centralized in one database file in opposition to previous generation tools where the project was split on hundreds of separate files without the guarantee of integrity.

Operation: ability to run testing scenarios in parallel with production on the same server; logging, historian and recipes in standard formats, such as SQL or XML, in opposition to closed systems; client server architecture, native redundancy, enhanced security system, easy integration of video, geo-information, 3d-models and remote web or tablet users.

Maintenance and evolution: Management of multiple product versions without requirement of multiple installations, complete remote access, ability to hot swap the project configuration without disconnecting operators or stopping the application; ability to run multiple concurrent projects on each server with multiple types of connected clients.

TECHNOLOGICAL UPDATE AND FULL USE OF THE HARDWARE

64-bit architecture, hardware acceleration, multi-touch, computers with multiple CPU cores, .NET Framework, C# and VB.NET languages, cloud computing, graphical hardware acceleration, are just some of the technologies that were not available when the internal architecture of previous generations of supervisory systems were created.

Although some degree of improvement can be done through upgrades and conversions, the full use of available technology usually demands a core design and architecture that is designed from its inception with the full knowledge of the available resources and the functional requirements.

Being able to open two projects simultaneously, automatically track configuration changes, allow remote access on the web by several engineers at the same time to the same projects, selecting displays by "preview" of the image rather than the name, all these functions are common to current text editors, but very often they were not incorporated in the previous generations of automation tools, some of them are not even running in 64-bit mode.

There are many features which have a tied connection with the technology and system architecture; therefore they are more effectively incorporated in a new design and a new generation of product; in general the tendency to add more advanced features on top of a core product created with old technology is very expensive, not reliable, only partially implemented and sometimes not even possible at all.

The following table lists some typical components of real-time and industrial automation systems, and how they benefit by the adoption of the new technologies.

Item	NEW GENERATION	LEGACY TECHNOLOGIES
Internal Programming	**C#/VB.NET/Java** Memory management is automatic, protected and greater independence of hardware and operating system protection.	**C++/C** Extensive use of pointers, required validation for each device, direct access to hardware and operating system.
Graphics Technology	**WPF, XBAP, Silverlight and XAML** Regardless of the resolution (vector) and uses hardware acceleration. Greater performance, native capacity for 3D and multi-touch.	**GDI/GDI+** Pixel-oriented, depend on the resolution of your monitor, distorted in conversion, less use of graphics hardware, limitations of dynamic animations.
Web client Technology	Native Web browser, without elevation or extra facilities.	Active installations, upgrading and need for security.
Vista Client Technology	WCF communication, standardized protocols, centralized installation and hot swap on the server.	Communication via proprietary protocols, installation on each client machine and no hot-swappable versions of project.
Editing and project execution	Multi-user, with editing and execution of multiple concurrent projects.	Single-user and mono project.
Remote access engineering	Native, multi-project, multi-user supported VPN environments and Cloud computing.	Use only in VPN through external utilities. Single-user normally.

Data model and Tag types	Data types reflect the models of processes, such as engine, valve and their properties.	Data Types reflect the memory of field equipment, such as byte, word, signed and unsigned.
Remote access to Runtime	Smart-client technology with centralized installation on the server or the WEB or Cloud, without installation of additional components. Standardized and secure protocols, such as WCF.	Local installation required for clients and WEB clients. Dedicated protocols with frequent need to free firewall ports.
Traceability of version control and configuration	Client-server architecture, SQL, databases centered with native traceability project versions and settings.	Architecture in multiple files and configuration and owners. Traceability performed manually or through external programs.
Functional modules and scripts at runtime	**Native Multiple processes and threads.** Each module and script execution thread .NET is natively protected from others. Architecture designed for effective use of multi-core processors. Exception control and memory protection is done by the operating environment.	**Single process multi-threaded or manually programmed** logical and sequential Execution of unified environment. Insulation of modules, parallel execution of scripts and protection exceptions, when it existed, was done through a dedicated programming with higher level of complexity.
Scripts	**Compiled (Vb.NET/c #)** Implementation of logic is between 10 x and 40 times faster than an interpreted script or owner. Performs more checks during configuration, is multi-threaded and handles exceptions, ensuring isolation of errors and increased performance. Full access to all functions in .NET Framework.	**Interpreted (VBA/VBScript or logical and mathematical proprietary)** Because they are interpreted, detecting many errors are possible only when you run the system. Most were mono-thread, meaning that slower functions or possible compromise of system errors. Sometimes with limited access to Windows functions.
Native platforms	**64-bit native. Support for 32-bit.** Better usage of hardware and more compatibility. The system was originally designed for 64-bit and to use components already present in the operating system.	**32-bit native. Support for 64-bit.** The 64-bit support is not possible, or where it exists, requires the installation of many additional components not native to the operating system.
Communication Drivers	Parallel execution with a capacity for multiple connections to each node. Automatic statistics, Diagnostics, Redundancy, syntax validation addresses field, integration of defining tags with the PLC, on multiport serial multi-protocol support, remote servers and pickup are regular functions.	Serial communication of network stations and only a TCP/IP connection to each node. Automatic statistics, Diagnostics, redundancy and other features mentioned were only partially available on some systems, were not yet the default minimum and regular systems.

History	Archiving to SQL with search optimizations, compression and management of daylight and time zone.	Owner history or archiving to SQL without optimizations. Common problems of daylight or access in different time zones.
Data exchange	Web Services, SOAP, XML, SQL queries. DDE, text files/CSV, COM and DCOM.	DDE, text files/CSV, COM e DCOM.
Alarms and Events	Distributed, with high flexibility.	Centralized, standardized targeting.
Hot swap projects	Enables online configuration.	Enables online configuration, but normally not allowed hot swap version running project

NEW USER INTERFACES, DESIGN AND CLOUD COMPUTING

Two themes that deserve their own article, consequently will be only briefly discussed here, are the new Design concepts for user interfaces and cloud computing. Many years ago there was already the concept of a tablet device, but it was only with the advent of the iPad that this technology was largely adopted; the major differential was the "Design". Much more than the simplified concept of appearance, Design determines the usability, the way to interact with the system. The new generation of automation software also brings the evolution of Design, not only the appearance but of a better usability of the configuration tools and projects. In the same way that changing from DOS for Windows changed the user interaction with programs, in this transition now, from Windows to the .NET Framework, there are also new User Interface paradigms to adopt, which bring the opportunity for configuration and programming interfaces to be more intuitive, productive, immersive, with more validation and why not, also more aesthetic.

As for Cloud Computing, it is clear that it will not replace control systems in the field, but it brings new features, both for the engineering and configuration. During the execution, there are now safer and more easily programmable interfaces to implement the distribution of real-time data to clients outside the corporate firewall, whether WEB clients or smartphones devices. During the project configuration and engineering phase the gain is the ability to provide easy collaboration, allowing distributed teams to work together. Previously, in order to allow the engineers to exchange project information, the method was to exchange emails with pictures, or FTP all the project files or plan a trip; with the cloud computing resources for collaborative distributed engineering, various teams in different locations can interact in real time, sharing the configuration, development and verification of the same project, with security of access and traceability of the modifications.

NEW SOFTWARE TOOLS

In some companies, at the same time it is a standard procedure on corporate IT to perform regular updates of their software systems, many industrial systems are relegated and keep using the same software tools from previous decades. Among several factors, some automation software was too tied to other elements of the automation, so the cost-benefit on upgrading to get marginal gains was not enough to justify the investment. This scenario also had significantly changed due to transition to this new generation of industrial automation systems.

The new technologies enable much more effective connectivity to legacy systems, in this way it not necessary to replace the control level to evolve your operator interfaces and to add more powerful management software. There are concrete and measurable gains, especially in security and flexibility, even keeping the field controls systems with the old components and evolving the HMI or MES level. If your current system is still based on legacy technologies, the most appropriate time to start planning to adopt new systems is exactly now, when the previous systems are still operable, not when its limitations due to old technologies raise to the point becoming your bottleneck in reliability, flexibility or evolution of the whole industrial process.

But just upgrading the latest version number of the same software tool is not enough, if that product was not created with the latest technologies. Use of current data migration techniques is very straightforward in changing your project configuration and your data from any legacy system to the new ones that are created on top of more updated technologies.

Finally, another important reason leading to this transition to new generation software tools is that the measurement of the gains in reliability, security, flexibility and functionality are not marginal percentages, but multiplicative factors. The adoption of the new systems has a clear ROI ensuring longevity and security for the facilities: the real-time graphical application managing a process is the front-end and visible link to the very large investment on the industrial assets being monitored, therefore leveraging the full advantages of a new software enable to get more from that whole system, what easily justify to adoption of the most current technologies for that front-end.

By: Marcos Taccolini, 2011
CEO at Tatsoft llc
www.tatsoft.com

REFERENCES

[1] Market Research Survey, August, 2010, held by Tatsoft and Control magazine.

[2] Zampronha, Roger, "the evolution of Supervisory Systems", 2009, technical article PCSoft.

[3] Vigiani, Roberto. "Quality and Project Management on SCADA systems", may, 2010, FactoryStudio User Guide, http://www.tatsoft.com

[4] Overview NET Framework http://msdn.microsoft.com/en-us/library/zw4w595w.aspx

[5] Peter Williams and Simon Cox. "Engineering in the Cloud: An Engineering Software + Services Architecture Forged in Turbulent Times", The Architecture Journal, June 2009\