2

Service eXtensions for Instrumentation (SXI)

Vítor Viegas*†, P. Silva Girão†‡, Miguel Pereira*†

*ESTSetúbal/LabIM, Instituto Politécnico de Setúbal, Setúbal, Portugal, vitor.viegas@estsetubal.ips.pt
†Instituto de Telecomunicações, Lisboa, Portugal
‡DEEC/IST, Universidade Técnica de Lisboa, Lisboa, Portugal

2.1 Aim

The Chapter presents a service-oriented middleware platform that takes advantage of the communication model provided by WCF services and the information model defined by the IEEE 1451.1 clause. The proposed platform was used to control a physical process equipped with all the instrumentation needed to run control loops for pressure, level, flow and temperature, quantities widely found in the process industry. Tests were made to evaluate the performance of the system in terms of sampling frequency, communication delays and behavior of control loops. The methodology of each experiment is described, results are analyzed and conclusions are extracted.

2.2 Introduction

Over the years, the evolution of measurement and control systems has been done by incorporating advances in other fields of knowledge, in particular the fields of microelectronics and information technologies (see figure 2.1).

Service eXtensions for Instrumentation (SXI), 1–29.

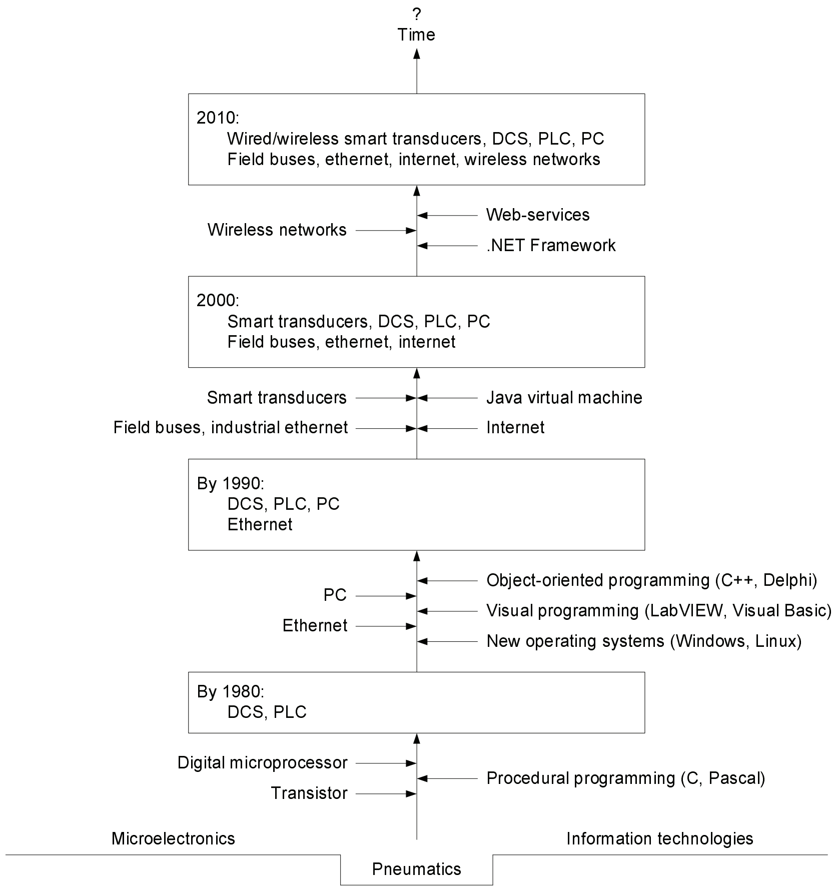

Figure 2.1 Evolution of measurement and control systems.

The introduction of the digital microprocessor drove to the replacement of old pneumatic controllers by modern Programmable Logic Controllers (PLC) and Distributed Control Systems (DCS). Since then, the Personal Computer (PC) gained its place in non-critical tasks (such as configuration, supervision and data analysis) and the ethernet became the preferred network for data communications. The increasing performance of the microprocessor (in terms of processing power, size, consumption and price) enabled its installation near to the primary element of measurement/actuation, all embedded in the same package, leading to the concept of "smart transducer". The transducers became smart enough to execute routines of self-identification, self-calibration and self-diagnostics, as well as to communicate with each other through a wired or wireless fieldbus.

In the field of information technologies, new development frameworks were released to address the demand for more and better software. These frameworks integrate all the tools needed to implement and execute applications, such as Object-Oriented (OO) languages (more productive than traditional procedural languages), visual editors (like those available in Visual Basic and LabVIEW), pre-built code libraries, intuitive debugging tools and high-performance virtual machines. A good example is the .NET Framework [1, 2], released in 2002 by Microsoft for Windows operating systems. The .NET Framework consists of three main components:

- **Programming languages**: The .NET Framework includes two last-generation programming languages: C#, which borrows heavily from C and C++ but adds high-level facilities such as automatic memory management, security and threading; and VB.NET, which is the evolution of Visual Basic. The high-level code is compiled into an intermediate language known as "Microsoft Common Intermediate Language" (MSIL).
- **Common Language Runtime** (CLR): Virtual machine that takes the MSIL code and compiles it to native processor instructions. The CLR makes .NET applications platform-independent because it provides an abstract execution environment that is separated from the underlying processor and operating system.
- **Class libraries**: The .NET Framework provides a vast set of dynamically loadable libraries that can be used by the programmer to perform common tasks. Hundreds of pre-built classes are available to construct graphical user interfaces, access files, interact with databases, handle network communications, and so on. Correlated classes are grouped in software packages according to their functional affinities.

The gains in terms of software productivity were followed by the development of cross-platform web-service-oriented applications [3, 4]. These applications cooperate in heterogeneous environments (like the internet) by calling remote methods (services) between them. Interoperability is achieved by imposing standards that describe the behavior of the service and the way to access it, regardless of its underlying implementation. This idea is not new, but new is the fact that web

- The temperature controller is able to track the setpoint but tends to oscillate due to the long dead time of the process. The anticipation capability introduced by the derivative component is crucial to limit the amplitude of the oscillations.

2.6 Conclusions

The Chapter focus on the SXI platform, a service-oriented middleware solution targeted for instrumentation. The SXI platform was tested in order to know its capabilities and limitations. The tests were conducted on a real plant equipped with all the instrumentation needed to run control loops for pressure, level, flow and temperature, quantities widely found in the process industry. The loops were closed by two control stations and supervised by one engineering station. The three stations were linked together through a 100 Mbit ethernet network. The results obtained led us to the following conclusions:

- The SXI platform is a feasible solution to implement distributed measurement and control systems. The system as a whole, and the control loops in particular, behaved well. All stations worked as expected.
- The SXI platform supports relatively low sampling frequencies (in the range of 10 Hz). This is mainly due to the time it takes to access the transducers. In each access, the hardware of the DAQ board is reconfigured leading to delays that can easily reach tens of milliseconds.
- The SXI platform is not ready for real-time applications. First, because the .NET Framework and the Windows XP operating system are not ready themselves for real-time applications [30, 31]; and second, because communications across ethernet are not deterministic by nature.
- The SXI platform does not guarantee the delivery of publications when the contention in the network is very high.

The limitations of the SXI platform can be overcome by taking the following actions:

1. Implement faster transducer blocks, preferably compliant with the IEEE 1451.0 clause.
2. Quit the .NET Framework and migrate the SXI platform to a real-time environment, such as C++ over the Windows CE operating

system [32, 33], or Java over a real-time Java virtual machine [34, 35]).

3. Use industrial ethernet switches [36, 37] to make communications deterministic.

The lack of support for real time stands out as the major limitation of the SXI platform at the present, and the first issue to be addressed in the future.

References

[1] David S. Platt, "Introducing Microsoft .NET", 3rd Edition, Microsoft Press, USA-WA, 2003, ISBN 978-0735619180

[2] Jeffrey Ritcher, "Applied Microsoft .NET Framework Programming", Microsoft Press, USA-WA, 2002, ISBN 978-0735614222

[3] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, "Web Services - Concepts, Architectures and Applications", Springer, Germany, 2010, ISBN 978-3642078880

[4] Adam Freeman, Allen Jones, "Microsoft .NET XML Web Services Step by Step", Microsoft Press, USA-WA, 2003, ISBN 978-0735617209

[5] http://msdn2.microsoft.com/en-us/netframework/aa663324.aspx

[6] Juval Lowy, "Programming WCF Services", O'Reilly, USA, 2007, ISBN 978-0596526993

[7] Justin Smith, "Inside Windows Communication Foundation", Microsoft Press, USA-WA, 2007, ISBN 078-0735623064

[8] http://ieee1451.nist.gov

[9] Engene Y. Song, Kang Lee, "Understanding IEEE 1451 - Networked Smart Transducer Interface Standard", IEEE Instrumentation & Measurement Magazine, Vol. 11, No. 2, pp. 11-17, April 2008

[10] "IEEE Std 1451.0 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats", IEEE Instrumentation and Measurement Society, USA-NY, 2007, ISBN 0-7381-5597-7

[11] "IEEE Std 1451.2 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats", IEEE Instrumentation and Measurement Society, USA-NY, 1997, ISBN 1-55937-963-4

[12] "IEEE Std 1451.3 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems", IEEE Instrumentation and Measurement Society, USA-NY, 2004, ISBN 0-7381-3822-3

[13] "IEEE Std 1451.4 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats", IEEE Instrumentation and Measurement Society, USA-NY, 2004, ISBN 0-7381-4007-4

[14] "IEEE Std 1451.5 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats", IEEE Instrumentation and Measurement Society, USA-NY, 2007, ISBN 0-7381-5599-3

[15] "IEEE Std 1451.7 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats", IEEE Instrumentation and Measurement Society, USA-NY, 2010, ISBN 978-0-7381-6246-1

[16] "IEEE Std 1451.1 - IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model", IEEE Instrumentation and Measurement Society, USA-NY, 1999, ISBN 0-7381-1767-6

[17] Vítor Viegas, J. M. Dias Pereira, P. Silva Girão, "A Brief Tutorial on the IEEE 1451.1 Standard", IEEE Instrumentation & Measurement Magazine, Vol. 11, No. 2, pp. 38-46, April 2008

[18] `http://www.w3schools.com/soap/soap_summary.asp`

[19] `http://msdn.microsoft.com/en-us/library/ms951274.aspx`

[20] `http://msdn.microsoft.com/en-us/library/ms751494(v=VS.90).aspx`

[21] `http://www.ni.com/dataacquisition/nidaqmx.htm`

[22] `http://zone.ni.com/devzone/cda/tut/p/id/4392`

[23] `http://zone.ni.com/devzone/cda/tut/p/id/2925`

[24] `http://zone.ni.com/devzone/cda/tut/p/id/4470`

[25] `http://www.yes01.co.kr/resite/yes01/plint_cat/AT1/thumbs/sect10/te34.htm`

[26] `http://www.codeproject.com/KB/testing/stopwatch-measure-precise.aspx`

[27] `http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)`

[28] Gerd Keiser, "Chapter 5. Basics of Queueing Theory", "Local Area Networks", pp. 167-201, McGraw-Hill, 1989, ISBN 0071003800

[29] Karl Johan Åström, Richard M. Murray, "Chapter 10. PID Control", "Feedback Systems: An Introduction for Scientists and Engineers", pp. 293-314, Princeton University Press, USA-NJ, 2008, ISBN 978-0691135762

[30] Michael H. Lutz, Phillip A. Laplante, "C# and the .NET Framework: Ready for Real Time?", IEEE Software Magazine, Vol. 20, No. 1, pp. 74-80, February 2003

[31] Martin Timmerman, Jean-Cristophe Monfret, "Windows NT as Real-Time OS?", Real-Time Magazine, Vol. 5, No. 2, pp. 6-13, June 1997

[32] http://www.microsoft.com/windowsembedded/en-us/evaluate/windows-embedded-compact-7.aspx

[33] Douglas Boling, "Programming Windows Embedded CE 6.0, Developer Reference, Fourth Edition", Microsoft Press, USA-WA, 2008, ISBN 978-0735624177

[34] http://java.sun.com/javase/technologies/realtime/index.jsp

[35] Peter C. Dibble, "Real-Time Java Platform Programming, Second Edition", BookSurge Publishing, USA, 2008, ISBN 978-1419656491

[36] http://www.iebmedia.com/ethernet.php

[37] R. A. Hulsebos, "Industrial Ethernet", eBook available in http://www.enodenetworks.com/specials/IndustrialEthernet.epub, 2010