HOW TO SURVIVE THE UPGRADE TREADMILL

Kirsten Hinsch, Ursula Laaströer, Rüdiger Schmitz, Winfried Schütte - DESY Hamburg, Germany

Abstract

Here at DESY we have nine large particle accelerators and correspondingly a large and complex control system. It consists of subsystems of many generations of hard and software. Until recently we even had to support Norsk Data mini computers so called NORDs. The number of applications is considerably more than a thousand. The control team consists of a dozen people aged typically in the fifties. This large, diverse and understaffed system introduces a considerable amount of (healthy) inertia. On the other hand we have a fast hard- and software upgrade cycle in the general computing scene. There is every few years a major operating system upgrade. This conflict produces tension and stimulation.

1 INTRODUCTION

Everybody who works a few years in our field physically feels the technology driven upgrade pressure. The useful lifetime of control system hardware and software is just too short. We want to improve the quality of our service for the operations crew, the machine physicists, the accelerator hardware specialists, the accelerator customers and more than often find ourselves just upgrading.

So lets take a closer look what drives us:

2 THE DRIVING FORCES

One of our PC dealers claims any part as old as six weeks is antique. The hardware is continuously changing. It is nice to have for a while the same main board, the same graphics card, the same Ethernet card. In this paradise the software can use the same driver and you can keep your zoo of systems in synchronization.

Even worse are technological changes of the hardware like the introduction of USB. Native USB support requires a recent operating system. On the other hand we again and again depend on things like a frame grabber PC card that has only drivers for 16 bit Windows.

Microsoft makes a major operating system change every other year. And a change that one can not ignore (16 bit vbx to 32 bit COM component to .Net) roughly every six years. The window of software usability is definitely shorter than the lifetime of any of our accelerators. Our hard and software assets turn within five years to liabilities: obligations to upgrade

Changes between hardware and software are interrelated. They form a complex fabric. Both constitute the core of the continuously moving treadmill: our current platforms.

Things get even worse: the administration with all its system support is also platform dependent. Next our applications and components used to run our control system depend again on each platform. We are most vibrantly trapped.

Lets look a little closer on the impact of changes in each part of our environment.

3 IMPACT OF CHANGES

3.1 Changes in the accelerator

This is a true place of peace and tranquility. Of course a new accelerator needs a new control system. A closed one does not need one anymore. Still a major change in the accelerator does not necessary have any impact on the technologies used. It is just a good and most welcome opportunity to change things now.

3.2 Network

Network upgrades have moderate impact on the other areas (fair degree of orthogonality). The major problem we have now is CISCOs mediocre IPX support. We expect this to get worse.

3.3 Computer and Associated Hardware

This is one of the two major driving forces.

At the upgrade from the NORD we could not keep the operating system family. Everything had to be written new. Very important: the concepts, architectures and designs could be kept. Both the users of the NORDs and the hardware controlled by the NORDs did not change after all.

The impact on the “little” hardware stuff like boards, plug in cards, busses is what tends to drive truly crazy. It puts pressure on the administration, maintenance and software systems.

3.4 Operating System

Major upgrades in the operating system require a change in the development environment. In the case of upgrading from Sintran, the NORD operating system, we had to switch to an entire new language.

3.5 Development Environment and Programming Language

Changes within the same programming language are usually relatively easy. A change within VB1 to 3 just required an expert assisted recompile and usually no changes to the source code at all. Both versions could even run side by side on the same operating system without any problems. The change from 16 to 32 bit Visual Basic is a bit more involved but at least an upgrade wizard assists (see chapter 5 for a concrete example).

On the other hand a complete change of the programming language with its associated standard libraries would require a quite lengthy training period, complete rewrite of the software and a rather long testing phase to actually achieve accelerator grade quality. In our case

JAVA has a lot of interesting properties that Visual Basic does not have. Especially its platform ubiquity. On the other hand the power of Java is comparable to VB.Net, which has a wizard supported standard upgrade path. A lot of effort and time can here be saved with respect to leaming, writing and testing. The good new features can than be introduced in an adiabatic way leading to a hopefully high gain with only little disruptions. Also there is a good chance that .Net will be faster spread over other platforms [1] then we are able to "side grade" to Java

3.6 Distributed System Support and Special Purpose Graphics

Up until roughly now control systems needed/used special components for communication across computers and for the efficient display of our data (plots, histograms,…) Usually this is done by one or two C experts. An upgrade should involve only work on their part. All the applications programmers are concerned of is the case of interface changes and even more philosophy changes of these services. In the future the establishment of a few industry standard ways of supporting distributed computing (like web services) will inevitably result in those deep consequences.

3.7 Architecture and Design

The ideas how to write a control system kept very constant over the decades. This lead to pretty much the same architecture and design. Any change herein would not only change the code, but also the way people thing about the control system.

We expect minor changes in the architecture due to the increasing necessity of new client platform support (SMS, Web Interface, Palms, embedded devices or whatever). Also a GAN (global accelerator network) [2] requires some modifications.

3.8 Demography

During the next ten years most of us will retire. New capable personnel are usually young and will only work with reasonable recent technologies.

4 A STRONGLY SIMPLIFIED OVERVIEW OF OUR UPGRADE SCENARIO

For a general feeling of the size of the upgrade problem we face you find in Appendix A a list of the computers we purchased during the last years. You see a strong increase in numbers. Anyhow you should also consider that not all Pentium IIs are equal (different boards, cards,…).

On the software side we had roughly 232 NORD programs (10-20kB each) and have now 583 windows programs running (DORIS 141, LINAC 2/DESY 2 173, LINAC 3/DESY 3 115, PETRA 109, System 45) and more programs on HERA.

You can find a synopsis of "A strongly simplified overview of our upgrade scenario" in the Appendix B.

5 A CONCRETE EXAMPLE

e are still in the multi year process of transferring large parts of the control system from Windows 3.11 with Visual Basic 3.0 applications to Visual Basic 6.0 SP3 on Windows NT 4.0. Yes, we do know NT 4.0 is not supported by Microsoft anymore. So lets look at the tasks in closer detail:

5.1 Provision of the New Software Environment

- ?? All developers get PCs with NT4.0 and VB 6.0.
- ?? Server and client PCs are doubled or can be dual booted in both systems.
- ?? All self made system tools are supplied for the new environment. All further developments on them will be done for both types now.

5.2 Preparation of the Applications

- ?? Try to set all properties of self made tools at run and not design time. Those properties will not be upgraded.
- ?? Try to use only original Microsoft Visual Basic Extensions. Sheridan 3D tools for example were supplied by Microsoft, but have problems during upgrade with non integral font sizes.
- ?? Copy all utility forms and modules in the appropriate new folders.
- ?? All files of your VB project have to be stored in text and not in binary format.
- ?? All project files are writeable.

5.3 The Actual Upgrade

Load the project into VB6.0 and let the upgrade wizard do its work. With a lot of luck everything is fine now.

- ?? The larger the number of self made, third party etc visual basic extensions (vbx) one uses, the more picture boxes one has.
- ?? Load the corresponding new COM components (dll, ocx, …) into the project.
- ?? Replace the dummy picture boxes with the corresponding new graphical components. Use the same name. If you managed to keep the interfaces constant, you can reuse your code completely (properties, methods, event routines).

5.4 Tidying it up

- ?? Adopt to the interface changes.
- ?? Make some of the design and architecture changes you always wanted to make. Like generalizing the application to more than one accelerator.
- ?? Incorporate the new tool of the day.
- ?? Test, test, test, don't stop too early, test, test, …

5.5 Findings

- ?? Practice is necessary and helps to gain efficiency
- ?? There is an upgrade wizard for the core language transformation. The bulk of the work has to be done by an actual person.

6 PRINCIPLES

In this section we will present some helpful principles to deal with those situations.

6.1 A Story of Nomads

We are like nomads and have to move within a quickly changing world. So we have to act like nomads:

?? Use as little as possible and as much as necessary.

?? Keep it as simple as possible but not simpler.

?? Keep things stable as long as sensibly possible.

6.2 Three Strategies

6.2.1 Master of the Environment

Keep things stable over the estimated lifetime of the accelerator (let us say ten years). Here we buy enough parts and spare parts and licenses to keep us going for a decade. We truly leave the treadmill.

6.2.2 Efficient Mover

We try to keep the supported upgrade versions to at most three ones. A single mainstream environment, the always present left behind one and one future environment. No other - especially no competing ones - are kept.

6.2.3 Anarchic Management

Give up on managing the environments and jump from problem to problem. Some people call it paradise other nightmare. Here everybody has a lot of freedom. The relative size of the upgrade management problem gets reduced by introducing an even bigger problem.

6.3 A Statement of Truth

Upgradability is a quality attribute of the (control) system and a quality attribute of the corresponding management. In this respect upgradability is not distinct from other quality attributes like maintainability, reliability, security, … [3].

7 CONCLUSION

Good luck.

8 REFERENCES

[1] There are at least three projects porting DotNet to non windows platforms:
http://www.go-mono.com/ (Ximian),
http://www.oreillynet.com/pub/a/dotnet/2002/03/27/archtour.html (by Corel for Microsoft),
http://www.southern-storm.com.au/ (BSDGNU)

[2] For example: Albrecht Wagner in
http://www.cerncourier.com/main/article/42/6/22 and
http://www.cerncourier.com/main/article/40/5/15/1

[3] Any good book on systems, software architecture or technical management. We recommend: Len Bass et al "Software Architecture in Practice", Addison-Wesley 1997

APPENDIX A: NUMBERS AND TYPES OF NEWLY ACQUIRED COMPUTERS

Just to give you an impression of the floods and tides of incoming computers. Here are our numbers of the past few decades:

Year	Number	Type
1973	3	PDP
1974		
1975	7	NORD 10
1976		
1977		
1978		
1979		
1980	ca. 10	NORD 100
1981		
1982		
1983		
1984		
1985		
1986	16	NORD 110
1987		
1988		
1989		
1990	60 + 6	NORD 120 + 5000
1991	ca. 80 + 5	I386 + I486
1992		
1993	ca. 62 + 10	I486 + Pentium
1994		
1995	ca. 67	Pentium
1996		
1997	132	Pentium
1998	108	Pentium
1999	50	P II
2000	63 + 35	P II + P III
2001	57	P III
2002	35	P III

APPENDIX B: A STRONGLY SIMPLIFIED OVERVIEW OF OUR UPGRADE SCENARIO

Era	Mini Computer	16 Bit OS	32 Bit OS		Possible Future
Network	Pocal on top of laboratory made network on top of XNS pre Ethemet standard	IPX IP Novell Router	Cisco Catalyst switched network		At least 100MB Ethernet everywhere
Computer **(and Associated Hardware)**	Norsk Data Mini Computer	PC: 386 486 Pentium Pentium II	PC: Pentium II Pentium III Pentium IV		PCs Workstations embedded devices Browser
Operating Systems	Sintran	DOS Windows 3.1 Windows 3.11	Windows NT 4.0 Linux		Windows XP Windows CE Linux, …
Development Environment, **Programming Language**	Pocal Interpreter	Visual Basic 3 Some C (system development) Visual Basic 6	Visual Basic 6 Some C (system development)		Java 4 J2EE CORBA Components VB.Net …, ???
		File locations could be kept			
Distributed System Support and Special Purpose Graphics	Laboratory made task and listen, plots, histograms and design editor	Laboratory made visual basic extensions (vbx)	Laboratory made COM components (ocx)		Less self made stuff in favor of industry standards (see entries above)
Architecture and Design	Pre emptive multitasking	Co operative multitasking	Pre emptive multitasking		any tasking multithreading
	Client/Server Equipment Functions Click on Screen Event driven				Multi tier: Many Views Many field busses Published Services GAN enabled