CERN School of Computing

The 7 Sins of Software Engineers in HEP

Ioannis Baltopoulos
CERN Summer Student 2004

CSC Student Lectures (Vico Equence)
Monday, 6th September, 2004

Outline

- The Context (HEP)
- The Sins!
  - Observation
  - Problem
  - Solution (definitely not exclusive)
- Conclusion

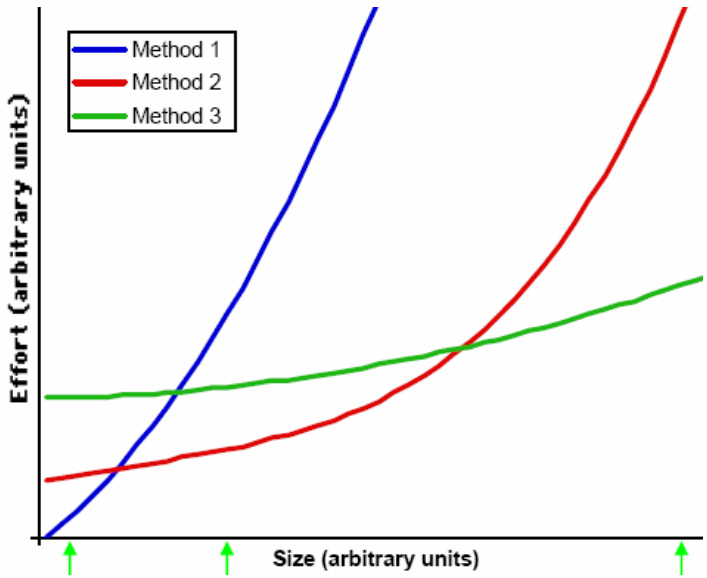
Establishing the Context

- Although these sins are **applicable in many** other situations, they are going to be addressed in the **context of HEP**.
- Agree that HEP Software Engineering is a **special case** because of the:
  - **Size** of projects
  - Amount of **money** invested
  - Amount of **people** involved
  - High **expectations** from research

Sin#1: Tool-coupled Productivity

Observation:

- We've used the **same tools** for the past 25 years
- Religious "wars" about the toolkit



Problem:

- Tools don't scale.
- They slow us down.

Solution:

- **New** free tools can do the "dirty" repetitive work for you.
- **Automate** as much as you can!

Sin#2: Code Infected

Observation:

- Software Engineers **love** writing code.
- Write it in small chunks; its **easier** to understand.

Problem:

- Easy to **write code**.
- Easy to **prototype** a system.

Solution:

- **Understand** the problem **&** the solution before coding.
- **Design** first, code later.

Sin#3: Process Infected

Observation:

- Very **detailed** and specific processes
- Quality Assurance is **postponed** or **flawed**

Problem:

- Reduce productivity/creativity
- Create unnecessary **overhead**

Solution:

- Process **Reengineering** from the SE themselves
- Have **bare minimum** that will save us from chaos
- Describe only the **functional aspects** of the work

Sin#4: Reinventing The Wheel

Observation:

- **Recurring** problems get solved all over again
- **Rewriting** algorithms (sorting?)
- Several projects that do almost the same thing

Problem:

- **Duplicated** effort, **wasted** man-power/months
- **Half completed** projects (80% maybe?)

Solution:

- **Use** patterns/templates/existing code/catalogs
- **Ask** for other peoples' experiences

Sin#5: Functional but not Usable

Observation:

- **Overexposing** interfaces
- **Information packed** applications/websites
- Inadequate help system/updated documentation

Problem:

- User's given **too much choice** and makes wrong one

Solution:

- **Get** feedback. Don't think for your users, ask them!
- Be **consistent**! Guide the users' actions.
- Spend that 10% of time to make the application shine!

Sin#6: Documentation Paralysis

Observation:

- **Huge** amount of documentation

Problem:

- Out of date
- Manually produced
- Not informational/relevant

Solution:

- Source code metadata documentation
- Reverse engineer diagrams during implementation

Sin#7: Change Resistant

Observation:

- Pieces are only added, never taken away
- “This is how we do it here, try and adapt.”

Problem:

- People like to stay in their **zone of comfort**

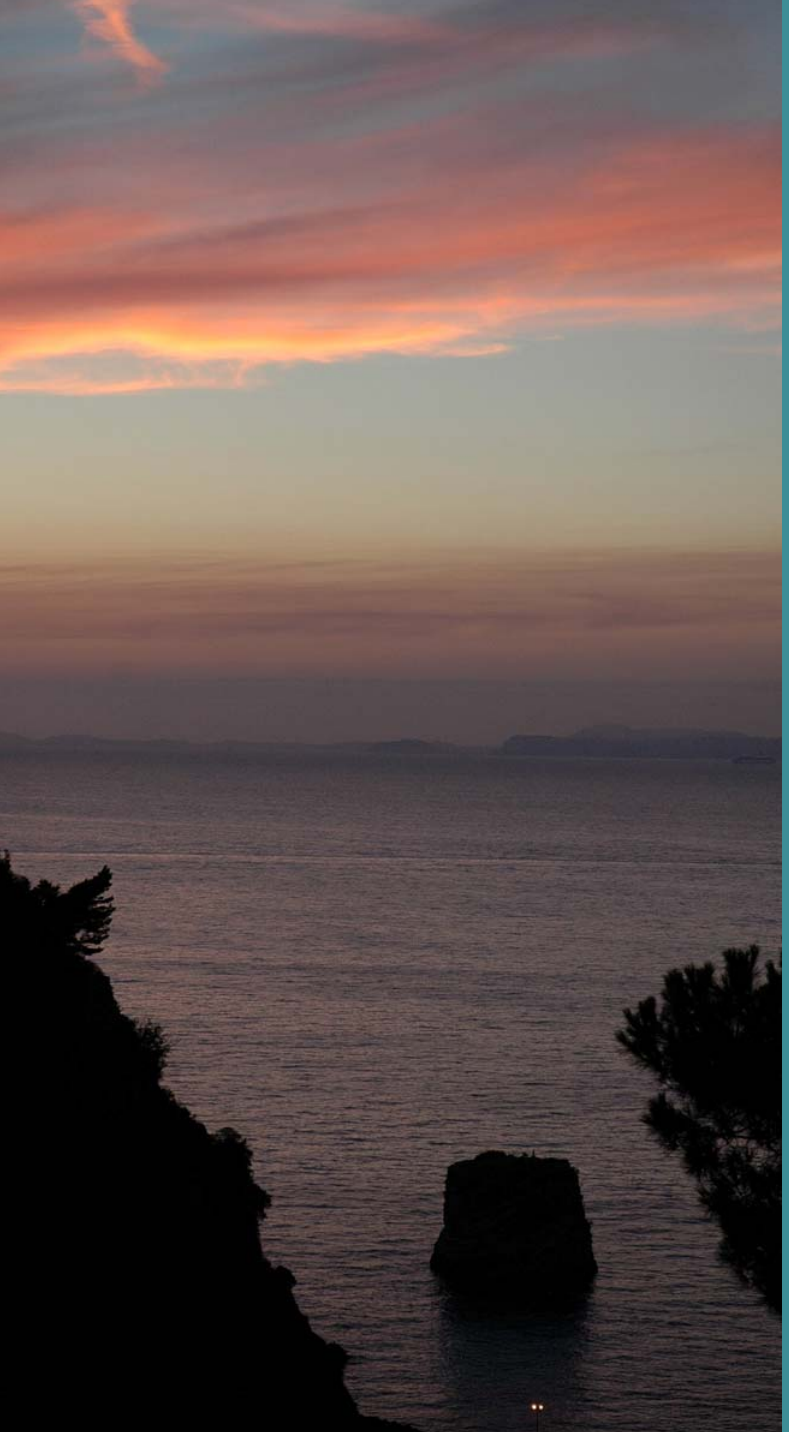
Solution:

- Be **flexible**!
- “The reasonable man adapts himself to the environment; the unreasonable man persists in trying to adapt the environment to himself. If any progress has been achieved it was due to the **unreasonable man**.”

Summary

Things to take home:

- Automate as much as you can.
- Over-design, under-engineer.
- Be lazy! Re-use code, Patterns, Templates.
- Looks matter!
- Be unreasonable! Change how things are done.



CERN School of Computing

Thank You!

Ioannis Baltopoulos
CERN Summer Student 2004

CSC Student Lectures (Vico Equence)
Monday, 6th September, 2004

Tools of the Trade: Build tools

- ANT (ant.apache.org)
- Maven (maven.apache.org)
- GNU Make (www.gnu.org/software/make)
- NAnt (nant.sourceforge.net)

Tools of the Trade: IDEs

- Eclipse (www.eclipse.org)
- NetBeans (www.netbeans.org)
- JCreator (www.**jcreator**.com)
- IntelliJ IDEA (www.jetbrains.com)
- Sun Java Studio Creator (www.sun.com/jscreator)
- JDeveloper (otn.oracle.com/products/jdev)
- Visual Studio (http://msdn.microsoft.com/vstudio/)

Tools of the Trade: Testing

- JUnit (www.junit.org)
- Clover (www.cenqua.com/clover)
- JCoverage (www.jcoverage.com)
- SQLUnit (sqlunit.sourceforge.net)
- DBUnit (dbunit.sourceforge.net)
- HTTPUnit (httpunit.sourceforge.net)

Tools of the Trade: Quality Assurance

- Checkstyle (checkstyle.sourceforge.net)

Tools of the Trade: Auto Documentation

- JavaDoc (www.sun.com)
- Doxygen (www.doxygen.org)
- yDoc (www.yworks.com)

Tools of the Trade: Design Tools

- ArgoUML
- Poseidon for UML
- Rational Rose
- MagicDraw
- Visio
- Together

Tools of the Trade: Version Control

- CVS (www.cvshome.org)
- Subversion (subversion.tigris.org)
- Visual SourceSafe (msdn.microsoft.com/ssafe)
- RCS

Source Code Resources

- Java Almanac
- Java Forums (forums.java.sun.com)
- Numerical Recipes in C
- Stony Brook Algorithm Repository

Design Patterns Resources