



Prescription for Healthy Code

by Alex Kriegel

(PMP, CSM, MCTS, TOGAF8 Certified Practitioner

PH OIS Enterprise Architect

October, 2009

Prepared for Software Association of Oregon Event

Code Quality

- “Software quality measures how well software is designed (*quality of design*), and how well the software conforms to that design (*quality of conformance*)”

Wikipedia

- Code Quality – necessary (but not sufficient) condition for **both**



Broken Window Syndrome

- “... the broken window effect can take root when [management] begins to tolerate downtime, constant work-arounds and broken processes.”*

John D. Halamka
CIO at CareGroup Healthcare System, CIO and
associate dean for educational technology
at Harvard Medical School

(just substitute developers for management, and
add some sloppy code to the mix...)

Creating a Code Quality Culture

- Educate developers
 - value of standards (and their limitations)
 - importance of development processes (and their limitations)
 - **meaningful** metrics as valuable feedback for code improvement
- Promote trust among the team members
- Use right tools for the job, and automate as much as you can
- Educate managers about value of code quality metrics

Code Quality

- First and foremost: it has to work!
- Start with the **Standards and Processes**:
 - coding standards (naming conventions, formatting, comments etc)
 - leveraging programming language
 - use of appropriate patterns
 - full traceability (requirements to code to bug fixes)
 - unit testing (incl. automated Regression testing)
 - meaningful metrics (Cyclomatic, NPath, Defects/KLOC etc. – **where and when appropriate**)
- Continue with **Change Control Process**

Tools of Trade

- Version Control
- Issue Tracking
- Standards Compliance
- Coverage Analysis
- Code Review Process
- Refactoring
- Documentation Compiler
- Logging Framework
- Continuous Integration/Build Management
- Automated Testing
- Change Control Board

Code Reviews

- Lead code review
- Peer code review
- “Automated” code review

Heterogeneous Development Environment (on shoestring budget)

- “Why not Visual Studio Team Edition?”
 - need to support both Java and .Net development
 - budget constraints – for a small-to-moderate team size, price for VSTS stack could easily run into tens of thousands of dollars
 - integration with existing enterprise components

Integrated Development Tools Stack (example)

Developer's Workstation



maven

Microsoft BUILD

FxCop



Hudson



CheckStyle



NCOVER
MSTest



Build Server Environment

Java Tools

- **CheckStyle**: compliance (coding standards)
- **PMD**: compliance (design, localization, performance, security)
- **EMMA/Cobertura** : code coverage
- **JUnit**: unit testing framework
- **Logging**: log4j
- **JavaDoc/Doxygen**: documentation compiler
- **Maven**: build management
- **Continuous integration**: Hudson
- **Jira**: issue tracking system
- **SVN/Subversion**: version control
- **Collaboration**: Atlassian Wiki

.Net/C# Tools

- **FxCop**: compliance (design, localization, performance, security)
- **NCover**: code coverage analysis
- **Doxygen**: documentation compiler
- **MSTest/NUnit**: unit testing framework
- **Logging**: log4net, Enterprise Library
- **MSBuild**: build management
- **Continuous integration**: Hudson
- **Jira**: issue tracking system
- **SVN/Subversion**: version control
- **Collaboration**: Atlassian Wiki

Other Free/Open Source Tools

.Net/C#	Java
<i>DevMetrics</i>	<i>JavaNCSS</i>
<i>PartCover</i>	<i>Cobertura</i>
<i>SourceMonitor</i>	<i>Coverlipse</i>
<i>CodeMetrics</i>	<i>CodeCover</i>
etc...	etc...

But what does it *mean*?!

(an example of applied metrics)

“Make everything as simple as possible but no simpler...”

Albert Einstein

For instance, high *Cyclomatic* complexity indicates **possible** poor code and/or design flaws

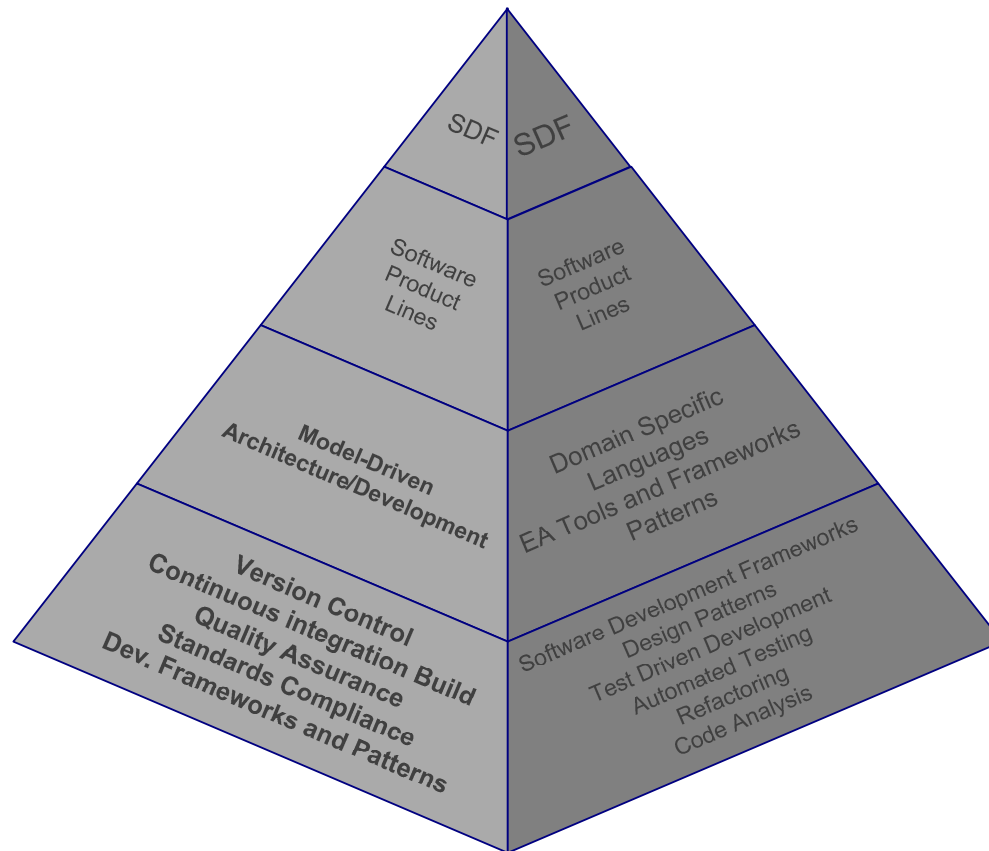
- Essential complexity
 - domain complexity, cannot be reduced
- Incidental complexity [might be]
 - technology induced
 - code/design problems

Train Your Tools Well

- Learn proper use of a tool – *then* tune up/customize it to produce metrics ***meaningful for your specific environment***
- No tool is an island – integrate tools into the development process (and SDLC)
- Each tool has to have a master ...
orphaned tools wither and die

Bigger Picture...

Assembling Applications with Patterns,
Models, Frameworks, and Tools



Software Development Factory

Questions ?

