Software Engineering

An **Oxymoron** in Your Organization?

Some Observations…
from Jonathan Kern

June 15, 2000

2

Topics

- The State of Software Development
- Mini-Survey of the Current Attendees
- A Look at How Things Get Engineered

    - Concrete & Steel
    - Software
- A Better Way
- Feature-Driven Development
    - FDD Details
    - Planning/Reporting Samples
- Extensions to UML to Support FDD

Goals

- Raise awareness
- Foster consensus that there is a problem
- Outline some solutions
- Treat Software Engineering for real
- Encourage growth
    - Developer
    - Management
    - Higher Education
- Spur action

4

Current State, Ugh (i)

- Some organizations are 10 (even 600) times more productive than others(1)
- Most Software Projects Fail
- Some definitions for Failure:
  - Missed schedule
  - Missed functionality
  - Missed budget
  - Too fragile for usage demands
  - Defect rates too high once in production

(1) Steve McConnell, *After the Gold Rush.* Redmond, WA. Microsoft Press, 1999.

June 15, 2000

5

Ugh (ii)

- In 1995, only 16% of software projects were expected to finish on time and on budget.(1)
- Projects completed by the largest US organizations have only 42% of

originally proposed functions.(1)

- An estimated 53% of projects will cost nearly 190% of their original estimates.(1)
- In large companies, only 9% of projects will be completed on time and on budget.(1)

(1) Standish Group International Report, “Chaos”, as reported in March ‘95 Open Computing. Copyright 1995 SPC.

June 15, 2000

6

Ugh (iii)

- Canceled projects—$81 billion loss to US in 1995(1)
- Average MIS—1 year late, 100% over budget(2)

(1) Standish Group International Report, “Chaos”, as reported in March ‘95 Open Computing. Copyright 1995 SPC.

(2) Capers Jones, *Applied Software Measurement*, McGraw-Hill, 1991.

June 15, 2000

7

Engineering vs. Science(1)…

- Scientists:
  - Learn what is true
  - How to test hypotheses
  - How to extend knowledge
- Engineers:
  - Learn what is true
  - Learn what is useful
  - Learn how to apply well-understood knowledge to solve practical problems

(1) Steve McConnell, *After the Gold Rush.* Redmond, WA. Microsoft Press, 1999.

June 15, 2000

8

Ad Hoc Survey (i)

[and some answers from UML World attendees via unscientific show of hands]

- How Many have a SWE Degree? [0]

- How many of you practice Software Engineering? ~10%
- Think of the last year's projects
  - 3+ months, or
  - Project team size >= 5
- Percent Successful Projects
  - 1% 100%, __ >75%, __ >50%, __ <25%, __ 0%

June 15, 2000

9

Ad Hoc Survey (ii)

- For failed projects, what went wrong?
  - Ill-defined requirements? [50%]
  - Ill-managed requirements changes? [30%]
  - Poor software development methodology? [25%]
  - Unrealistic schedule? [25%]
  - Poor project management? [20%]
  - Lack of user involvement?
  - Lack of stakeholder involvement?
  - Lack of qualified people?
  - Lack of tools?
  - Corporate politics get in the way?
  - Poor or no architecture?

- Lack of measurements and controls?

June 15, 2000

10

Is this How It Should Be?

- Why do we allow this to happen?
- Would you like your next <item> built like this?
  - Item car, home, airplane, bridge, ICU monitoring s/w
- Engineering and construction firms don't build things in the way most software gets built…
- Why is software allowed to so often be done without any engineering? Many Reasons…
- Can we do anything about this? Yes!
- Are there some better ways? Yes!!

June 15, 2000

11

Current Practices are Broke!

- We can do better!
- We must do it better!
- We should employ engineering methods

*Everyone can enjoy random success, but one is advised **not** to build a career on it!*

-- Bicknell, 1993

June 15, 2000

12

The Core Triad for Success

- People
- Process
- Technology
- People + Process + Technology
  - Process is no substitute for synaptic activity

- You still need to think and use common sense
- And a lot of that is not a book or course-learning exercise. It takes street smarts.
- That's where mentoring and consulting come in and add significant value

June 15, 2000

13

Engineering Evolution

- Mechanical Engineering, for example, has been around for centuries
- Task: Build a Bridge
  - Easy to describe features (length, width, load) and to know when "Done" (can drive over it :=)
  - Design discipline is well-understood by the architect and licensed civil/mechanical engineers, builders
  - Other stakeholders have "bought-in" (DOT)
  - All are able to read engineering & mgt. artifacts (e.g., blueprints, stress diagrams, WBS & Gantt charts)
  - Progress easy to measure, obvious milestones/goal

- Project can be tackled by forming teams of skilled subs

Engineering Evolution (ii)

- People:
  - Licensed Architects, Mechanical/Structural Engineers, Civil Engineers, Project Managers, *etc*.
  - Certif'd specialists: welders, concrete, site layout, *etc*.
- Process
  - Well-worn steps to designing & making a proposal
  - Tried-and-true (profitable) management techniques
- Technology
  - Improved design and analysis tools
    - Allow for safety factors and cost optimizations
  - Improved materials
    - Dramatically affects building design
  - Automated management tool support

15

Engineering Evolution (iii)

- Continuous Improvement
  - We learn from past mistakes
  - We require certification
  - Processes have improved
  - Infrastructure is in place
    - Higher education
    - Licensing
    - Codes of Ethics
    - Professional Organizations

June 15, 2000

16

S/W Evolution

- People
  - Position Descriptions of the 70s:
    - Programmers
  - 80s
    - Programmer/analysts
  - 90s
    - Developer
    - Architect

    - Business Analyst
  - 00s ?
    - UI Designer
    - Bean Provider
    - Assembler
    - Deployer

+ Architect
+ Modeler
+ Implementer
+ Project Manager
+ System Analyst
+ Config. Mgr.
+ System Tester
+ Test Designer

June 15, 2000

17

Software Evolution (ii)

- Processes
  - Ad Hoc
  - Waterfall

- Spiral
- Incremental
- Iterative
- RAD/JAD
- Unified Process
- Extreme Programming (XP)
- Feature-Driven Development

June 15, 2000

18

Software Evolution (iii)

- Technology
  - Languages: Assembler , Procedural, Structured, Object-Oriented
  - 3GL/4GL/CASE
  - Life Cycle Tools
    - Requirements, Architecting, Building, Testing
    - Configuration Management/Version Control
    - Round-trip Engineering (manual steps)
    - **Simultaneous** round-trip tools
  - Modeling:
    - Structured, DFD,
    - Coad, OMT, Booch,

- UML

June 15, 2000

19

Software Evolution (iv)

- Still facing same problems as the last 25+ years
  - Ill-defined requirements
  - Demanding schedules
  - Fast-moving technology
  - Fast-moving business needs
  - Large-scale projects
- Not much widespread improvement
- But there is hope…

June 15, 2000

20

Differences Between Software & Hardware Engineering

- Primary difference is that it *is* indeed “soft”

- S/W is viewed as being more changeable
  - But is it really?
- Users demand much more flexibility in S/W
  - Spectrum of cost: point solution generic/flexible ($$$)
- We haven't reached the level of H/W standards
  - Not many real software "Integrated Circuits" to allow quick build-up of system functionality
  - Components still not widespread
    - Except for UI components
  - Patterns/models not as accepted as engineering/arch. blueprints
  - No "ComponentDepot" -- yet

June 15, 2000

21

Some Fundamental Issues

- Software is very complex today
  - Hard for one to understand it all
  - Difficult to express in terms all stakeholders understand
- Business drivers add pressure
  - Shrinking business cycle

- Competition increasing
- Ever rising user expectations

- "Soft" Requirements
  - A common threat to schedule, budget, success
  - Too much change can cause failure

Fundamental Issues (ii)

- Flexibility and resilience to change is key
  - Ability to adapt to sudden market changes
  - Design is solid enough that change does not impact the core design in a destabilizing way
  - Willingness to re-architect as required
- Most projects are unpredictable
  - Lack of knowing where and what to measure
  - Lack of yardsticks to gauge progress
  - Requirements creep is common
  - Scrap and rework is common
  - Interminably 90% done

23

Fundamental Issues (iii)

- Lack of quality people results in failure
  - …in spite of best processes and tools!
  - Managing people is difficult
- Major change is organizationally difficult
- Reusing software artifacts is rare
    - Architectures/Designs/Models
    - Estimating processes
    - Team processes
    - Planning & Tracking procedures and reports
    - Software construction & management
    - Reviews, testing
    - *etc.*

June 15, 2000

24

What do we Fix?

- Common Problems Today
  - Requirements unclear
  - Requirements changes cause disruption
  - Poor quality, maintenance nightmares
  - Late breakage

- Delaying risk mitigation
- Performance woes
- Non-repeatable processes
- Poor architecture
- Lack of testing

- Blown schedules
- Built the wrong thing
- Filled an outdated need
- Outright abrupt cancellation
- Doesn't deliver ROI
- Staff is a revolving door

June 15, 2000

25

Requirements *Still* Important (1)

- Probably still true today (10 years later)
- Ordering defects by severity

levels 1 (low) to 4 (highest):

- Requirements dominate level 4

- Design & requirements share level 3
- Design dominates level 2
- Code dominates level 1

- Do we need 400-page Requirement Specs?

- Jones, Capers, Applied Software Measurement: Assuring Productivity and Quality, New York: McGraw-Hill, 1991, p136

June 15, 2000

26

Defects…

- Average distribution of defect types in the U.S. (corporate IS and DOD orgs)

6

10

Bad Fixes

7

5

Documents

35

30

Coding

27

25

Design

25

30

Requirements

DOD%

MIS %

June 15, 2000

27

Basic State of Affairs

- The majority of the quality defects occur during the requirements & design phases
- This is not unique to US firms
- (Actually not even unique to software!)
- Need to improve capturing requirements while simultaneously:
  - Integrating those requirements into system designs
  - Providing continuous integration testing
  - Delivering frequent, working, tangible results
  - Gracefully handling the inevitable change
- Need to remain focused on customer

June 15, 2000

28

Some Possible Solutions

- A Modern S/W Engineering Process
- Using UML Within the Process
- Feature-Driven Development Methodology(1)
- Extensions to UML to support FDD
- Extreme Programming (XP)
  - XP sets about trying to find out how development *should* proceed if you had *enough* time(2)

- I feel there's a kindred, pragmatic spirit between FDD & XP

- Coad, Peter; De Luca, Jeff; Lefebvre, Eric. *Java Modeling in Color with UML,* Upper Saddle River, NJ. Prentice Hall, 1999.
- Beck, Kent. *Extreme Programming Explained.* Reading, MA. Addison Wesley Longman, 2000.

June 15, 2000

29

Putting Engineering into Software Development

- Applications need to be architected and designed, not just "built"
- Hardware/construction engineering uses
  - Processes/Process Improvement
  - Tools (but no silver bullets)
  - Standards
  - Highly Skilled, licensed/certified People
  - Employs System Integration concepts
  - Things aren't just "built" with no planning
- Just because you have a hammer, *everything* isn't a nail!

June 15, 2000

30

Putting Engineering into S/W (ii)

- Software engineering should be no different than hardware engineering
- Habits of Successful Projects
  - Employ an iterative process
  - Use requirements-driven approach
  - Do up-front visual architecting/design
  - Perform continuous integration, use metrics & QA
  - Work with frequent, tangible artifacts (running code)
  - Have solid team support and communication

June 15, 2000

31

Putting Engineering into S/W(iii)

- Successful Project Habits (cont'd)
  - Use tools to automate tasks
  - Are able to reuse items
  - Have quality people

        - Management
        - Development
        - Specialists
    - Have good scheduling (realistic, accurate)
    - Frequent, positive involvement with stakeholders
    - Progress reporting
    - Requirements well understood (at least over time)

June 15, 2000

32

Analysis

Assess

Design

Build

A Modern Iterative Process

- Architecture first
    - The Object Model rules
- Iterative
    - Run through ever-increasing depth of features, performance, and quality
- Component-based

- Reuse, visual modeling
- Change Management
  - Metrics, trends, monitoring
- Round-trip Engineering
  - Modeling Tools, integrated environments

Activity flows are truly multi-directional, not just spiral!

June 15, 2000

33

Process (ii)

- Strive for:
  - Frequent results stemming from architecture-centric approach where the model is the code
  - Architecture driven by client requirements
  - Strong team that communicates through graphical artifacts, code, and working prototypes to avoid miscommunication
  - Improved process and measurable, automated quality control (testing, audits, metrics)
  - Quality at every turn

June 15, 2000

34

Process (iii)

- Avoid *both* extremes:
  - "Analysis Paralysis"
    - Too many paper studies
    - Too few tangible, working baselines
    - Too many business analysts
  - Getting hordes coding too soon
    - Chaotic design decisions "on the fly"by non-architects
    - Prolific solo coders get too far ahead with poor direction
    - Continuous hacking, scrap, and rework, is a symptom
  - Always ask: "Am I moving the project ahead or the completion date back by doing this?"

June 15, 2000

35

The Venerable Waterfall Process

Analysis

Design

Implementation

Testing

Maintenance

- Postpones Confronting Risk
- Late Design Breakage

- Can work on well-defined efforts
- Can work in smaller efforts
- Great for reporting apparent progress

June 15, 2000

36

A Solution

- Feature-Driven Development
  - Client-centric
  - Architecture-centric
  - Repeatable process
  - Pragmatic, livable methodology
  - Great for developers
  - Great for managers
  - Great for the application!

June 15, 2000

37

FDD and Software Engineering

People:

Processes:

Tools:

- **FDD**
- **Testing**
- **SQA**

- **Project Mgt**

- **Class Owners**
- **Chief Programmers**
- **Feature Teams**

- **Together® (Model+)**
- **"Parking Lot" Reports**
- **Version Control**
- **Build Management**
- **UI Builders**

FDD terms are in RED

June 15, 2000

38

Why the FDD Process?

- To enable and enforce
  - the **repeatable** delivery of
  - **working** software in a
  - **timely** manner with
  - highly **accurate** and meaningful **reporting** to
  - all key **stakeholders** inside and outside a project.

June 15, 2000

39

The FDD Process in Brief

FDD #1
Develop
an
Overall
Model

FDD #2
Build
a
Features
List

FDD #3

Plan
by
Feature

FDD #4

Design
by
Feature

.5 Major Steps/Activities

FDD #5

Build
by
Feature

Requirements

& Design

Design,

Some
Code

Code,

Initial

Testing

Test
& Put
in Build

Sched.
&
$$ Est.

Build

Promote
to
Build

Prioritized Releases

June 15, 2000

40

#4

- #1 & #2 are Reqt's & Design through Modeling/Coding/ Prototyping to get it right
- #4 is very granular Detailed Design/Code

- #5 is Detailed Code and Test in very,very granular chunks of client-valued functionality

FDD & Typical SDLC Phases

Analysis

Design

Implementation

Testing

Maintenance

#1

#2

#5

June 15, 2000

41

Reminder:

- Why FDD #1 & #2 Processes' Emphasis on Requirements & Design?
  - Studies have shown conclusively that it pays to do things right the first time
  - Unnecessary changes are expensive
  - TRW: a change in requirements-analysis cost 50-200 times less than same change later in the cycle (construction-maintenance). **Boehm, Parpaccio 1988**
  - IBM: removing an error by the start of design, code or unit test allows rework to be done 10-100 times less expensively than during unit test or functional test. **Fagan 1976**

June 15, 2000

42

A S/W Eng. View of FDD

- FDD #1 & #2
  - Look at the larger scope
  - Determine tools and architecture needed to get the job done as simply and as soon as possible
  - Eschew the "Code-and-Fix(1)" Approach
  - Emphasize planning and process up front
    - While simultaneously focusing on client needs
    - And reducing the need for lots of rework
- Conforms to good Engineering Practices

(1) Steve McConnell, *After the Gold Rush.* Redmond, WA. Microsoft Press, 1999.

June 15, 2000

43

An XP View of FDD

- FDD #1 & #2

- Work with domain experts to elicit requirements
- Record (at least) in the form of an object model
  - With Together® true round-trip engineering, this means real **source code**
- Confront riskier sections of domain with deeper coding as required
- Demands using your head, knowing when to stop, re-trace steps, and go down new path
- Re-architect as required to adapt to requirements

- Fits in with XP philosophy

June 15, 2000

44

XP Basic Principles(1) & FDD

Frequent, fine-grained iteration through reqts to coding and back

Incremental Change

Focus on big picture, detailed view driven by client-valued features, pragmatism rules

Assume Simplicity

Tangible, working results, testable, know if it works

Rapid Feedback

FDD

XP

- Beck, Kent. *Extreme Programming Explained.* Reading, MA. Addison Wesley Longman, 2000.

June 15, 2000

45

XP Basic Principles & FDD (ii)

Frequent, working results are non-ambiguous. Razor sharp reporting of being “done” leads to habits of being successful. Working in teams, peer reviews help to ensure quality.

Quality Work

Work with running system sooner, soliciting early client feedback, encouraging future feature discussions

Embracing Change

FDD

XP

June 15, 2000

46

Typical Iterative Development

Elaboration

Requirements

Analysis

Design

Coding

Testing

Inception

Construction

Transition

Iterations (1-n)

PRODUCTION

ENGINEERING

June 15, 2000

47

- Where/how does it fit into s/w development practices
  - In modern software development methodologies, there is a large degree of iterative development.
  - Productive modeling environments automatically synchronize source code and class diagrams
  - This implies that you take successive passes at the system, adding new information and capability along the way.

Fitting UML into Development

Inception

Iterative
Elaboration

Iterative
Construction

Transition

June 15, 2000

48

The Utility of UML

- Helps to promote standard communication

- But does not supplant human contact!
- Class Diagrams are very useful
- Sequence and Activity help with dynamics
- Use Cases
  - Useful if your organization has meaningful way to apply them—no silver bullet
  - Can be replaced by other means of capturing features/requirements

June 15, 2000

49

FDD Artifacts

FDD
#1

Breadth of App:
Class Diagram,
w/out

much content

FDD
#2

FDD

#3

List of
hi-level
features

Class Diagram
w/ more shape;
greater domain
understanding

List of
prioritized features

Initial Plan plus

Reports

Assign CPs
choose class
“owners”

S/W Artifacts

Mgt Artifacts

People

Architects,
Domain
Experts

Architects,
Dom.Exp’ts,
Chief
Prog’rs (CP)

June 15, 2000

50

FDD Artifacts (ii)

FDD
#4

Model gets more
content. Detailed
design in code.
Add Seq. Diags.

S/W Artifacts

Mgt Artifacts

People

FDD
#5

Running Code & Tests

Model gets
built, feature
by feature.
Promote to build,
release…

Build
and
Test

Detailed Models, Seq. Diag. As req'd

Form
Feature
Teams

June 15, 2000

51

FDD "Engineering" Outputs

FDD #1
Develop
an
Overall
Model

FDD #2
Build

a
Features
List

FDD #3

Plan
by
Feature

FDD #4

Design
by
Feature

FDD #5

Build
by
Feature

An object model
(more shape than content)

A categorized list of features

A Develop-ment Plan

June 15, 2000

52

FDD “Construction” Outputs

An object model
(more shape than content)

FDD #1
Develop
an
Overall
Model

FDD #2
Build
a
Features
List

FDD #3

Plan
by
Feature

FDD #4

Design
by
Feature

FDD #5

Build
by
Feature

A categorized list of features

A Develop-ment Plan

A design pack-age (sequences)

(more content than shape)

A client-valued function

June 15, 2000

53

FDD UML Extensions

- Report progress to management and clients
- Very high level (major feature sets)

- These reflect UML extensions made in Together®

June 15, 2000

54

FDD UML Extensions (ii)

- Next level down (feature sets)

Chief Programmer

Features

% Complete
(color too)

Due Date

Overdue!

Dec 2001

Completion Percentage:

Completion Status:

Completed

Targeted Completion Month

Example:

Feature Set:
Making Product Assess'ts –
Work in Progress

CP-1 is the Chief Programmer's initials

(14) there are fourteen features that make up this feature set

75% Feature Set is 75% complete

Target is to complete in **Dec 2001**

Overall Status:

MY

Progress bar

Work in progress

Attention (*ie*, Behind)

Completed

Making

Product

Assessments

(14)

75%

Not yet started

CP-1

FDD UML Extensions (iii)

June 15, 2000

56

Product Sale Management (PS)

Invoicing

Sales

(33)

Dec 2001

CP-1

Setting up

Product

Agreements

(13)

Dec 2001

Selling

Products

(22)

Nov 2001

CP-1

Shipping

Products

(19)

Dec 2001

CP-1

10%

Delivering

Products

(10)

Dec 2001

CP-3

30%

Making

Product

Assessments

(14)

Dec 2001

75%

99%

3%

Customer A/C Mgmt (CA)

Evaluating

Account

Applications

(23)

Oct 2001

95%

Logging

Account

Transactions

(30)

Nov 2001

82%

Opening

New

Accounts

(11)

Oct 2001

100%

Inventory Mgmt (IM)

Establishing

Storage Units

(26)

Nov 2001

100%

Moving

Content

(19)

Nov 2001

82%

CP-3

Accepting

Movement

Requests

(18)

Nov 2001

97%

CP-3

KEY: Work In Progress **Attention** **Completed** **Progress Bar** **Not Started**

CP-2

CP-1

CP-2

CP-2

CP-2

CP-3

FDD Sample Feature Sets

June 15, 2000

57

Milestones

"Milestones must be concrete, specific, measurable-events defined with a knife-edge sharpness"

"A programmer will rarely lie about milestone progress, ***if*** the milestone is so sharp he can't deceive himself"

"Getting the status is hard since subordinate managers have every reason not to share it"

Fred Brooks

June 15, 2000

58

Milestones (ii)

- Assigning % permits accurate reporting
- A feature that is still being coded is 44% complete
- Domain walkthrough 1% + Design 40% + Design

inspection 3% = 44%

Plan

Plan

Plan

Plan

Plan

Plan

Actual

Actual

Actual

Actual

Actual

Actual

Promote to Build

Code Inspection

Code

Design Inspection

Design

Domain Walkthrough

1%

40%

3%

45%

10%

1%

June 15, 2000

59

Milestones (iv)

- The Plan and Actual dates are completion dates
- An entry in the actual column for a milestone signifies that milestone has been achieved
- This triggers the percentage calculations for that feature, its feature set and so on
- Intervals aren't recorded – but they could be

June 15, 2000

60

FDD Tracking

- Granular level of a feature with planning and tracking properties used to calculate %complete
- These reflect UML extensions made in Together®

June 15, 2000

61

FDD Plan View

June 15, 2000

62

FDD Feature View

June 15, 2000

63

FDD Weekly Summary

June 15, 2000

64

FDD Weekly Summary (ii)

June 15, 2000

65

FDD Trend Reporting

June 15, 2000

66

FDD's Contribution to S/W Project Success Factors*

Process works to minimize creep

<10% requirements creep

Not the mission of FDD

Automated config. control

Informal thru continuous integration

Formal risk management

Yes, for modeling/building, FDD doesn't cover testing

Formal development methods

Object model architecture (shape) is key part of process

Formal architecture planning

Granular and summary (roll-up) reporting

Formal progress reporting

Continuously adjust estimates if required

Continuous planning

Plan/estimate by feature

Use of estimating tools

Tracks completion by feature

Accurate s/w measurement

Patterns of Software Systems Failure and Success (Jones 1996)

June 15, 2000

67

FDD's Contribution (ii)

Not the mission of FDD

Formal database planning

Informal thru architect

Significant reuse

Part of process to glean granular enough features and model to a sufficient depth

Controlled and measured complexity

Not part of FDD, but part of tool

Use of suitable languages

Not part of FDD, but part of tool

Automated design and specs

No. Relies partially on process to build in quality and frequent builds to test

Formal testing

Part of the process, a milestone

Formal code inspections

Part of the process, a milestone

Formal design reviews

Patterns of Software Systems Failure and Success (Jones 1996)

June 15, 2000

68

FDD's Contribution (iii)

Not the mission of FDD

Use specialists

FDD team collaboration links lead developers with junior developers

Capable project staff

FDD should help to make project managers more successful

Capable project management

Not the mission of FDD

Experienced Sr. Execs

FDD process covers team collaboration and radically improves communication

Team communications

Not the mission of FDD

Congruent Mgt. Goals

Part of the modeling process: involve client early and often.

Cooperation with clients

FDD presents clear picture of needs, should help ensure understanding

Exec understanding of est.

Plan/estimate by feature

Realistic Schedules

Patterns of Software Systems Failure and Success (Jones 1996)

June 15, 2000

69

Defect Removal Efficiency

- 80%—Formal structured peer reviews (e.g., Fagan inspections) (1)
- 60%—Code walkthroughs (2)
- 36%—Integration test
- "Formal design and code inspections have the highest defect removal efficiency of any technique yet noted, and average about twice as efficient as most forms of testing." (3)

(1) US Army Information Systems Command, "Software Quality and Testing: What DoD Can Learn From

Commercial Practices",

AIRMICS Report # ASQB-GI-92-012, 8/31/1992, p10

(2) Boehm, Barry, Industrial software metrics top 10 list, IEEE Software, Sept. 87

(3) Jones, Capers, Patterns of Software Systems Failure and Success, Thompson, 1996, p129

Additional Topics for Discussion

- The Worth of Quality Inspections
  - Code reviews
  - Audits
- Test real system continuously
- Metrics – have to measure your projects

Summary

- Most Software Development lacks engineering rigor today
- There are pockets of success, some even repeatable – we need more
- New processes exist (*e.g.*, FDD) to help achieve success
- Able to extend UML to support FDD

June 15, 2000

72

Further Info

- Coad, De Luca, Lefebvre. *Java Modeling in Color with UML*. Upper Saddle River, NJ. Prentice Hall, 1999.
- Royce, Walker. *Software Project Management.*
- McConnell , Steve. *After the Gold Rush.* Redmond, WA. Microsoft Press, 1999.
- Jones, Capers. *Applied Software Measurement.* McGraw-Hill, 1991.

- Beck, Kent. *Extreme Programming Explained.* Reading, MA. Addison Wesley Longman, 2000.
- Many other assorted articles

- Contact: **Jon Kern** (jk@TogetherSoft.com)