Agility and Quality
(or) What CMM Level is my XP Project?

Herb Krasner
Director, UT Software Engineering Industry Affiliates
Director, UT Software Quality Institute
President, Krasner Consulting
hkrasner@ece.utexas.edu

Abbreviations used in this presentation:
  SW - software
  SWE - software engineering
### Who is Herb Krasner?

<table>
<thead>
<tr>
<th>Year</th>
<th>Institution/Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>'71- '79</td>
<td>U. of Mo. - Rolla</td>
<td>Asst. Prof. of CS</td>
</tr>
<tr>
<td>'81</td>
<td>Clemson Univ.</td>
<td>Developed Grad. pgm in SWE</td>
</tr>
<tr>
<td>'83</td>
<td>Harris Gov’t Systems</td>
<td>Research in DB theory</td>
</tr>
<tr>
<td>'88</td>
<td>MCC SW R&amp;D</td>
<td>DBA for Medicaid Claims Processing DBMS System</td>
</tr>
<tr>
<td>'90</td>
<td>Lockheed Research</td>
<td>Senior Technologist - Ada/SWE methods development</td>
</tr>
<tr>
<td>'91</td>
<td>SAIC</td>
<td>Project Lead - Empirical studies of SW design</td>
</tr>
<tr>
<td>'02</td>
<td>Krasner Consulting</td>
<td>Chief SW Technology Officer</td>
</tr>
<tr>
<td>'04</td>
<td>UT</td>
<td>Division General Manager</td>
</tr>
</tbody>
</table>

**Getting educated in CS/EE**

- Teaching CS Undergrad education
- Researching DB design methods
- SWE - developing PAX Res. System
- ML, AL and PL/1 expert
- A “systems” person

**Researching DB methods**

- Asst. Prof. of CS
- Developed Grad. pgm in SWE
- Research in DB theory
- DBA for Medicaid Claims Processing DBMS System

**Working with large projects**

- Working with large projects in C3I, R/T Satcom & Data Mgt.
- Ada expert

**Project Lead**

- Various lead roles in new SW R&D org.
- Director, Recruiting

**Chief SW Technology Officer**

- SW Process R&D group leader
- Leonardo SEE Reqts. team leader
- SW Tech. Xfer strategies

**SEI Assessments & SEPG development across Lockheed Co.**

- PI R&D projects: IMG, SPMS, Metrics, AI/ SW Proj. Mgt.

**Excellence Coach**

- “The Process Doctor”
- Organizational, project & product evaluations
- Subject matter expert
- Topical speaker & frequent writer, pub. reviewer
- Teacher SPI, SQI & SWE Seminars

**Division General Manager**

- New business develop.
- ARPA STARS
- Austin Quality Award Examiner (MBNQA)
- UT SWE Education

**Sr. Lecturer in CS**

- Director, UT SWE industry affiliates

---

*My mission - stamp out poor quality software, enable superior software*
Blatant Advertisements

• TX Symposium on Software Engineering - Aug. 27-28, 2004
  - Critical issues in SWE facing us in the next decade
  - Who wants to help?
  - Does ASEE want to be a co-sponsor?

• UT’s new Software Engineering Programs
  - Research: ARISE, see www.ece.utexas.edu/arise/
  - Education: MS in SWE, new BS in SWE in 2004
  - Outreach: new SWE Industry Affiliates Program
    • Looking for Charter Member Companies - can you hook me up?

• The rebirth of the Software Quality Institute
  - Back in the short courses business
  - SWPM and new certificate programs being conceived
Software is important and still problematic

- SW is the engine of innovation in our connected and embedded world. It’s the glue and the enabler of our applications and systems.
- Although there is much useful software produced, there is also much useless software produced. Commercial software products today are riddled with defects. The best commercial SW companies remove about 95% of all known defects prior to shipping⁴.
- Today’s experienced SWEs inject about 100 defects per 1000 lines of code⁵.
- The software project success rate today is ~30%⁶ - runaway or failed projects are still common - late deliveries dominate
- The best software organizations outperform the worst by at least 5-10X (measured by productivity, cost/schedule performance, quality)⁷.
- Reducing the cost of SW development and improving SW quality are important objectives of the U.S. SW industry.
- Let’s look at the national trends in SW
The Economics of SW - Terms

• **Software sales** - the amount of revenue generated by the sales of SW products and related services

• **Software development costs** - the total amount of money consumed by the process of defining, designing, producing, delivering and supporting SW products and related services for sale.

• **Software quality costs** (CoSQ) - that portion of the development costs consumed by producing less than ideal SW (e.g. useless, poor, late, off-target, etc.). There are 3 key components to CoSQ; the costs due to: external problems, internal problems, and enabling (better) quality.

• **SWE costs** - are found in both development and CoSQ.
The size and complexity of the SW needed to support the U.S.'s computerized economy is increasing at an alarming rate.

- **SW Sales**
  - 2000: $100 B (7.55)
  - 2003: $140 B (10.89)
  - Projected: $180 B (14)

- **Cost of SW development**
  - 2000: $100 B (7.55)
  - 2003: $140 B (10.89)
  - Projected: $180 B (14)

- **Cost of SW Quality**
  - 2000: $100 B (7.55)
  - 2003: $140 B (10.89)
  - Projected: $180 B (14)
The Impact of Poor SW Quality

- Canceled and off-track projects lead to wasted resources
- Lack of quality focus leads to late deliveries that cause penalties, over-runs, lost market share, lost revenues
- False feature rich SW leads to disuse
- Over committed projects lead to lots of rework, low productivity, turnover, poor complexity management, organizational stress
- Post delivery problems lead to huge bug costs, development instability, maintenance drain, early system retirement, unavailable and unused systems
- Lost user productivity due to faulty software leads to customer dissatisfaction
- Persistent poor performance leads to dissatisfied customers, tarnished reputation, lost business, corporate demise
What is SW Quality?

Although no standard industry definition exists for what constitutes good quality in software, it is generally taken to mean that a software product/system provides value to its users, makes a profit, generates few serious complaints, and contributes in some way to the goals of humanity (or at least doesn’t do harm).

**Customer Satisfaction**
- Achievement of requirements & needs
- Customer and user satisfaction levels
- Fitness for use (when & where)

**Properties (“illities”)**
- Usability
- Reliability (MTBF)
- Portability
- Maintainability
- Complexity

**Defectiveness**
- Defect levels
- Defect severity
- Defect removal

**Perspectives**

**Value to Stakeholders**
- profit
- competitive advantage
- personnel satisfaction

**Development Execution Link**
- Schedule & budget performance
- features & functions delivered
- process quality
- Effectiveness
The Software Quality Manifesto

- Software quality means different things to different people - the conflicts must be worked out among all stakeholders.
- The customer ultimately decides what is "good" quality - this depends on "expectations" which are mutually established and continually managed. Knowing what the customer needs does help.
- The characteristics of software quality must be defined and quantitatively managed for every product/project development.
- The development organization must have internal quality standards.
- The development team focus is on managing and preventing defects, not on human mistakes, must drive out the fear of defects.
- Quality is continuously injected into the emerging and evolving software - not inspected or tested in later.
- Continuous quality improvement is institutionalized and measurement based.
- Definition of quality, process and other factors lead to the selection of technology.
In general, it is the quality of being ready and able to move with quick, easy grace. In SW it is the ability to develop valuable software quickly, in the face of rapidly evolving requirements.

The principles of agile SW development are:
1. Satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Harness change for the customer’s competitive advantage.
3. Deliver working software frequently - from a couple of weeks to a couple of months - prefer shorter timescale
4. Business people and developers work together daily throughout the project
5. Build projects around motivated people. Give them the environment and support they need. Trust them to get the job done
6. Face to face conversation is the most efficient and effective method of conveying information within the team.
7. Working software is the primary measure of progress
8. Promote sustainable development - should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design
10. Simplicity - the art of maximizing the amount of work not done - is essential
11. Self-organizing teams create the best architectures, requirements and designs
12. At regular intervals, the team reflects on how to become more effective, and then adjusts its behavior accordingly
Agile SW Development Manifesto

• “We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
  – Individuals and interactions over processes and tools
  – Working software over comprehensive documentation
  – Customer collaboration over contract negotiation
  – Responding to change over following a plan
• That is, while there is value in the items on the right, we value the items on the left more.”
• Signed by the members of the agile alliance, see
  – http://www.agilealliance.com/home
• Methodologies include: FDD, Scrum, ADP, Crystal, eXtreme Programming (XP) - convergence is predicted!
eXtreme Programming (XP)

• XP is an agile methodology made up of simple, yet interdependent practices which work together to enable success. XP has:
  – Philosophy/assumptions
  – Values
  – Principles
  – Activities, Practices and Roles
• Software development is viewed as an evolving dialog between the possible and the desirable
• XP rejects the notion that it costs less to make changes earlier in the development cycle, rather than later.
• Project goals include the success variables: cost, time, scope and quality. In the agile approach the customer/management gets to pick 3, the team determines the 4th.
• XP is described by an iterative, incremental process model
• In just a few years, it has proliferated rapidly - there are many examples of XP success stories
**XP Iteration Development Process**

- **Iteration plan**
  - Choose a story
    - Create acceptance tests
      - Run and pass acceptance tests
        - Release
        - Create latest version
  - Choose a task
    - Create unit tests
      - Do pair programming
        - Refactor as needed
        - Integrate with whole system
          - Run and pass all unit tests
            - Create latest version

The 12 XP Practices

1. Planning Game – quickly determine the scope of the next iteration. Customers do the planning based on feedback from the developers.
2. Small Releases – take baby steps in each iteration. Rank iterations according to those which deliver the most valuable business requirements.
3. Metaphor – define a simple story of how the system will work. It should be enlightening.
4. Simple design – few classes and methods, no duplicated logic
5. Test Driven Development – Developers write unit tests, Customers write functional tests before coding
6. Refactoring – revisiting code with rules that simplify the code. “When the system requires that you duplicate code, it’s asking for refactoring.”
7. Pair Programming
8. Collective Ownership – anyone can change any code at any time.
9. Continuous Integration – code is integrated frequently. Integration is putting new code with the current system.
10. Sustainable pace - a sane work week
11. On-team customer – customer needs to be around, actively involved
12. Coding standards that all coders follow
Issue: Agility Vs Quality

- It might appear on the surface that these 2 concepts are mutually exclusive, but only if
  - You believe that quality is an absolute, and that quality/process control trumps satisfying the customer in his timeframe
- A popular myth is that “quality software necessarily takes longer”
- How does this fit with a previous focus on process maturity (e.g. CMM)?
- Beating the competition to market, and delivering features on or ahead of schedule is also a quality goal - right?
- Meeting the real needs of customers in a timely way while dealing with unknowns, changes and uncertainties is the claimed strength of agility
Agility and Quality

- The commercial software product marketplace is largely innovation driven
  - Package applications, O/S, telecomm, EDP, CAD, etc.
- Value is defined by product capabilities and market introduction timing leading to “market share”
- Quality is defined by the relative advantage gained, and the rapidity of responses to the challenges of competitors (during feature wars and product maturity/refresh)
- Agility in development is **more important** than stability
  - Tell that to your classic project manager!
  - The heart of a culture clash
  - Must agility lead to instability?
How do we measure them both?

• **SW Quality**
  - Customer satisfaction ratings
  - Product defectiveness
  - others

• **SW Agility**
  - Customer satisfaction ratings
  - Feature to market cycle time (with acceptable quality)
  - others

• The relationship depends on the type of software being developed
SW Quality and Cycle Time

for commercial shrink-wrapped software ONLY

pre-release defect removal rate

development time

most projects

fastest projects
Sub Issue: Agility Vs Discipline

- **Discipline** - following a repeatable, optimized process to achieve a well-defined and stable goal set (i.e. a planned process)
- The current notion of proper discipline in SWE comes from where?
- There is a natural tension between these two concepts, especially when you look at the process control movement of the last 10 years (e.g. CMM, IEEE stds)
  - Does agility promote hacking?
  - Does a heavyweight process stifle innovation?
  - SWE is not manufacturing, it is arts and crafts 101
- The interpretation of the meaning of quality process from these two perspectives is different
- Beware, culture clash ahead!!
The Frameworks & STDs Quagmire

Also see www.software.org/quagmire
CMM Process Maturity
- definitions and assumptions -

- **Process** - a series of actions leading to some desired result
- **Maturity** - the emergence of behavioral characteristics through growth processes, leading to becoming fully developed.
- **CMM process maturity concept is based on:**
  a) “I wanted to get software organizations to adopt Deming's approach to continuous improvement, but I realized it had to be done in stages.”
  b) “The quality of a software system is governed by the quality of the process used to develop it”
  c) “A process must be defined, measured and tracked in order to be *in control* and improvable.”
**SW Capability Maturity Model (CMM)**

**Objective** - characterize, control and evolve an organization’s software development process capability

---

**Initial Process**
- Ad Lib SWE & Mgt.
- Basic Project Mgt. areas under qualitative control
- KPAs: RM, SPP, PTO, SSM, SQA, SCM

**Defined Process**
- Qualitative SPI by org. learning with a defined process
- KPAs: OPF, OPD, OTP, ISM, SPE, IC, PR

**Repeatably Managed Process**
- Quantitative process control with measurement basis

**Quantitatively Managed Process**
- Continual SPI with quantitative process change mgt.
- KPAs: Defect Prevention, Org. Process Improvement, Org. Improvement Deployment

---

CMM v2.0 C - KPA means Key Process Area

Copyright@2003 - Herb Krasner - all rights reserved
Culture Clashes Within An Organization

Culture “P”  Culture “A”  Culture “R”

Culture “E”
Sub Issue: Agility Vs Plan-ability

✓ **Plan-driven** - following a detailed formulation of a program of actions to achieve a goal

- There is a natural tension between these two concepts. Counter beliefs include:
  - uncertainty is inherent and inevitable in software development
  - for a new software system, the requirements will not be completely known until after the users have used it.
  - it is not possible to completely specify an interactive system
  - The Waterfall approach does not work on wicked problems

- The interpretation of the meaning of quality from these two perspectives may be different

- *If you can’t plan well, then plan often*
Emphasis on Planning

Agile methods

IEEE/ISO std process

CMM/TSP/PSP

Ad Lib/hacking Scrum XP RUP

Ironclad, micromanaged contract

Risk driven

Milestone driven

Agility home ground

• Premium on rapid value creation
• Empowered, active customer
• Knowledgeable, collocated, collaborative developers
• Smaller teams, products
• Emergent requirements, rapid change
• Architected for current requirements
• Restructuring is inexpensive
• Reliance on tacit interpersonal knowledge
**Emphasis on Planning**

- Agile methods
- IEEE/ISO std process
- CMM/TSP/PSP

**Ad Lib/hacking**  Scrum  **XP**  **RUP**  **Ironclad, micromanaged contract**

**Plan-driven home ground**
- Premium on high assurance SW
- Mix of customer capability levels
- Plan-oriented developers with a mix of skills
- Larger teams, products
- Knowable, upfront requirements with little change
- Architected for current and foreseeable requirements
- Restructuring is expensive
- Reliance on explicit documented knowledge (e.g. process)
Emphasis on Planning

Application type examples

- Agile methods
  - XP
  - RUP
- IEEE/ISO std process
  - CMM/TSP/PSP
- Business IT infrastructure
  - Grid Computing
  - Telecomm
- Personal games & tools
  - e-Services
  - e-Commerce
- Life-critical systems
  - Grid Computing
  - Telecomm
- Safety-critical, embedded systems

Ironclad, micromanaged contract

Risk driven

Milestone driven

Ad Lib/ hacking Scrum
### Specific Project Drivers for Agility

<table>
<thead>
<tr>
<th>Driver Area</th>
<th>Pivot Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>Time-to-market pressure in months, coupled with application type goals</td>
</tr>
<tr>
<td>Developers</td>
<td>Highly talented, jelled team</td>
</tr>
<tr>
<td>Customers</td>
<td>Committed, competent and engaged</td>
</tr>
<tr>
<td>Requirements</td>
<td>&gt; 1% rate of change/month</td>
</tr>
<tr>
<td>Architecture</td>
<td>Ability to scale-up to foreseeable future requirements, no deal “breakers”</td>
</tr>
<tr>
<td>Restructuring</td>
<td>Cost of change/fix is not exponential</td>
</tr>
<tr>
<td>Size</td>
<td>&lt;= 15 persons or added layer of coordination/mgt. process</td>
</tr>
</tbody>
</table>
## Specific Project Drivers for High Quality

<table>
<thead>
<tr>
<th>Driver Area</th>
<th>Pivot Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectives</td>
<td>High levels of assurance needed due to criticality of system</td>
</tr>
<tr>
<td>Developers</td>
<td>Diverse skill sets and talent levels</td>
</tr>
<tr>
<td>Customers</td>
<td>Diverse and busy</td>
</tr>
<tr>
<td>Requirements</td>
<td>$\leq 1%$ rate of change/month</td>
</tr>
<tr>
<td>Architecture</td>
<td>Built upfront to suit all foreseeable future requirements</td>
</tr>
<tr>
<td>Restructuring</td>
<td>Cost of change/fix is exponential</td>
</tr>
<tr>
<td>Size</td>
<td>Teams of $&gt;15$ persons or multi team organizations</td>
</tr>
</tbody>
</table>
Requirements Change Management - the big issue -

Traditional approach:
1. “control” changes
2. Formal process for analyzing and responding to change requests
3. Resist late changes
4. Assumes that \( Q = \) conformance to reqts

Agile approach:
1. Welcome late changes
2. Accept changes at increment boundaries (frequently)
3. Informal process for analyzing and responding to change requests (during the planning game)
4. Assumes that \( Q = \) customer satisfaction

Who is exercising discipline and control?
Conclusions

• Quality and agility may represent two cultures at odds - The “process people” are not winning in some organizations

• Higher quality and faster cycle time are not mutually exclusive; and can be mutually enabled with the right approach in the right settings
  - Specify your project performance objectives at the beginning to establish the context - > including quality goals

• If you want to become an excellent supplier of quality SW to your customers then you must adapt to change; there are very few settings in which you can really “control” it
  - A common key practice is delivering early prototypes to users

• There are many organizations struggling with the issues of what to do in order to survive and/or thrive - Agility is the next big fad - you better understand it

• For more on XP see my 13ICSQ tutorial next month
References

1. US. Department of Commerce, 2000 - “annual SW sales in the US is estimated at $180B”, “growing at 15% per year”.
2. US Bureau of Labor and Statistics database, 2002, there are 2,169,000 software developers in the US, determined by aggregating several relevant job categories
3. NIST Planning Report 03-2, May, 2002, “The overall cost of poor SW quality in the US is just under 1% of GDP” - the US GDP is ~ $10.4 Trillion
5. K. Smith, The Software Industry’s Bug Problem, Quality Digest, April, 2003
9. Estimates determined by population density proportions, no TX data available