

# Project Management Course Book

T++ Technical Skills Training Program

CUNY Institute for Software Design & Development (CISDD)

New York Software Industry Association (NYSIA)

Aron Trauring

Class 3 October 12th, 2004

# Class Agenda

**1:30-2:30** Traditional Methods — Historical Background '70s and '80s (Presentation)

**2:30-2:45** Break

**2:45-3:15** Exercise 3 — Applying FourM

**3:15-3:45** Break Out Presentations

**3:45-4:00** Break

**4:00-5:30** Agile Manifesto (Presentation)

## Traditional Methods - Historical Background '70s and '80s

### Structured Methods — Software Development Lifecycle (SDLC)

- Waterfall method — Winston Royce 1970
- Structured Methods \* Software Engineering
- Edward Yourdon, Peter Coad, James Martin, and Tom DeMarco

### SDLC — Project Phases

1. Requirements — What the Customer wants
2. Analysis — Understanding the Requirements — Functional Specification
3. Design — Software Architecture/Decomposition — Technical Specification
4. Development/Implementation — Writing the code — Programs
5. Unit Test — working unit
6. System Test — working system
7. Acceptance Test — Working application

### SDLC — Motivations

- Lower cost of hardware rising cost of software
- More complex systems became possible
- Industry in it's infancy
- Human nature — false analogy to manufacturing
- Response to ad hoc development of '60s
- Adopted as a military standard
- Gives management what it thinks it wants
- Good descriptive model

### SDLC — Snake Oil

- Ignores complexity
- Analysis Paralysis
- Pushes risk to end where it is more costly
- Ignores creativity and unpredictability
- Today: experience, research, standards, and experts all oppose

**Innovations of the Seventies and Eighties — Computerized Tools**

- Computer Aided Software Engineering (CASE)
- Modern: Rational
- Integrated Development Environments (IDE)
- Turbo Pascal - Phillippe Kahn, Borland '83
- Modern: Visual X.Net, Eclipse
- Fourth Generation Languages (4GL)
- FOCUS - Gerry Cohen, Information Builders '73
- Modern: SAP, Oracle
- Application Generators \* dBase II - Ashton Tate, '83
- Modern: Access, Application Servers (Cold Fusion, Zope)

**Innovations of the Seventies and Eighties — Object Orientation (OO)**

- Allan Kay — “The best way to predict the future is to invent it”
- 1971 - Smalltalk - Model the 'real' world
- Xerox Palo Alto Research Center (PARC) in seventies and eighties
- Steve Jobs — WIMP
- Personal Workstations

**OO — Key Concepts**

- Objects basic building blocks — data and associated functions
- Objects interact through exchanging messages
- Internal behavior is hidden
- Objects know protocols of communication

**OO — Advantages**

- Easier to decompose the problem
- Easier to map analysis to design
- Easier to reuse components

**OO — Two Schools**

- Ada - Information Hiding, Static - C++
- Smalltalk - Messaging, Dynamic - Java, Python

**Innovations of the Seventies and Eighties — CMMI**

- Capability Maturity Model Integration
- Based on research at CMU funded by the DOD
- Categorize the processes used by an organization
- Scale runs from ad hoc (chaotic) to mature (disciplined)
- Each process has key practice areas (KPA) needed to achieve its goal
- Organizations are classified into five maturity levels
- Highest level: software development is predictable

**CMMI — Critique**

The underlying assumption — good processes lead to good results — is both a questionable assumption and one that leads to too much introspection about process and too little focus on actual results.

— Jim Highsmith

- CCM levels 2 and up apply to a narrow domain
- More sophisticated maturity models needed

**Important Lessons**

- Avoid the single-pass waterfall
- Nothing new under the sun
- Tools move Brooks' conceptual problem up the chain
- Adopt the Smalltalk model to the extent possible



# Agile Manifesto

## The Agile 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.

## Agile Manifesto — What?

- Meeting in Snowbird, Utah in February, 2001
- 17 gurus of divergent methods
- Agree on the need to respond to change - agility
- Agree on four core values
- Agree on twelve statements
- Disagree on project tactics — this is not a unified method

## Agile Manifesto - Commentary

- 'Uncover' - not invented and not silver bullets
- 'Doing it' - practice what you preach
- 'Value the Right' - We come to praise Caesar not to bury him
- Industry still in its infancy

## Individuals and Interactions

- Attend to the people not the roles in the process chart
- Process serves people
- Quality of interactions matter
- Good community is key to success

## Working Software

- “Running code is ruthlessly honest” — Alistair Cockburn
- Goal is customer satisfaction
- Real versus promised

**Customer Collaboration**

- “There is no ‘us’ and ‘them’ - only ‘us’” — Alistair Cockburn
- Amicable relations
- Joint decision-making
- Good communications

**Responding to change**

- Change is normal
- Sticking blindly to a plan is tightening a noose

**Agile Values and FourM**

- Brooks not sufficiently acknowledged
- Stress on people
- Stress on communication
- Stress on working code
- Stress on responding to change

**Value Statement One**

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- Allows for rapid feedback and mid-course changes
- Early delivery mitigates risk
- Continuous production

**Value Statement Two**

Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

- Delivering stale requirements is valueless
- Change is inevitable

**Value Statement Three**

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time-scale.

- FourM IID
- Time-frame is project and customer dependent

**Value Statement Four**

Business people and developers must work together daily throughout the project.

- FourM communication
- Waste is extremely costly, rework undesirable
- Great resistance within organizations to this

**Value Statement Five**

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- FourM Surgical Team
- People over process

**Value Statement Six**

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Minimizes waste and lead time
- Improves communication — fewer misunderstandings
- Symptom of relations within the team

**Value Statement Seven**

Working software is the primary measure of progress.

- Measure of delivered value
- Not Milestones, Not Documentation
- No other measure is trustworthy

**Value Statement Eight**

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- Maximize ROI
- Constant pace means no overtime — overtime wears out people

**Value Statement Nine**

Continuous attention to technical excellence and good design enhances agility.

- FourM Conceptual Integrity
- Rework and waste are costly
- Not contradictory with frequent delivery — decomposition and surgical team

**Value Statement Ten**

Simplicity — the art of maximizing the amount of work NOT done—is essential.

- Increase production - built more quickly, less faults, less testing
- Simplicity is clearly defined:

Simple clear purpose and principles give rise to complex and intelligent behavior.

Complex rules and regulations give rise to simple, stupid behavior.

– Dee Hock

**Value Statement Eleven**

The best architectures, requirements, and designs emerge from self-organizing teams.

- Architecture and design, like specification, change over time
- Self-organizing is ambiguous
- FourM Surgical team is not 'anything goes'
- Empower team members — don't micro manage

**Value Statement Twelve**

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

- Culture of continuous learning and improvement - CMM compatible
- Self-correcting system