

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 4 October 14th, 2004

Class Agenda

1:30-2:45 IID Concepts (Presentation)

2:45-3:00 Break

3:00-3:30 Exercise 4 — Compare and Contrast: IID vs. SDLC

3:30-4:00 Break Out Presentations

4:00-4:15 Break

4:15-5:30 The Project+ Game

IID Concepts

IID is NOT Software Engineering

“...[P]eople mostly use the word 'engineering' to create a sense of guilt for not having done enough of something, without being clear what that something is. “

– Alistair Cockburn

- Model Building - not working code
- Rigorous Software Methods(RSM) — goes against need for change

Software Development as a Game - Alistair Cockburn

- Characteristics: Finite, Goal-directed, Co-operative, Resource Limited
- Motivations: Fun, Challenge
- Success: Improvisation, Skill-sensitive, Talent, Tools, Team-work
- Type: Invention (core concept, solving in technology and implementation) and Communication
- Primary Goal: Deliver useful working software
- Secondary Goal: Set up for the next game (alter the current one, or start a neighboring game)

Characteristics of Predictable Manufacturing

1. First specify then build
2. From specifications can reliably estimate effort and cost
3. It is possible to identify, define and schedule all activities
4. Change rates are relatively low

Characteristics of New Product Development

1. up front, not possible to create unchanging, detailed specs
2. up front, no reliable empirical data for estimates
3. up front, not possible to identify all detailed activities
4. Change rates are high

Key difference between hardware and software

- Hardware: In the implementation phase components are immutable and inorganic
- Software: Human thought remains the only component throughout all phases
- Software manufacturing will never be totally predictable

Iterative Development

- Overall life-cycle is composed of multiple iterations
- Each iteration goes through all waterfall activities
- Iteration release at end of each cycle - stable, tested, integrated, partially complete

Iterative Incremental Development

- Each iteration adds new features
- Number of iterations varies with project and method
- Not the same as prototyping
- IID is 100% FourM compatible
- All Agile methods are IID
- Agile <> IID
- Older IID methods (e.g. RUP) are adapting to/adopting Agile concepts

Types of IID

- Risk-Driven - choose riskiest, most difficult tasks for early iterations
- Client-Driven - choose tasks client perceives as highest business value

Apply both. Clients do not always perceive what is technically hard or risky. Developers do not always appreciate what has business value.

Four variables of an iteration

1. Time
2. Scope (requested tasks)
3. People
4. Quality

Time-boxed Iterations

- Fixing iteration end date and not allowing it to change
- If chosen scope can't be met, reduce the scope
- Don't use overtime to increase people - quickly causes reduced quality
- Length of time-box varies with project and method
- All methods encourage time-boxing
- No external change of scope by stake-holders within specific time-box

Evolutionary and Adaptive Development

- Evolutionary: Requirements, plans and solutions evolve over time
- Nothing frozen up-front
- Compatible with unpredictable discovery and change of New Product Development
- Adaptive: Elements adapt in response to feedback from prior work
- Feedback comes from users, developers and testers
- Feedback-response mechanism of systems

E & A Analysis

- Requirement discovery and refinement mostly takes place in early iterations
- Earliest attention to understanding most architecturally significant or high business-value requirements
- Start with vision statements and top-ten high-level requirements
- Early technical analysis of architecturally influential factors: e.g. load, usability, internationalization
- Series of requirements activities during early iterations to expand and refine requirements
- Don't need to know all functional requirements to build architecture

E & A Planning

- Initial high uncertainty — use past data and experience
- Cone of uncertainty — estimates improves further along into iterations
- Selling management: exploratory iterations replace up-front analysis phase
- Adaptive planning — detailed schedules with short time horizon
- Level of detail and commitment commensurate with quality of information

Fixed-price contracts

- Two contract phases
- First phase exploratory iterations
- Results of first phase open to all bidders
- Partial working system created in first phase, not just documents

E & A Delivery

- Repeated delivery to end-users before project end
- Not at the end of every iteration
- Wider feedback and more robust integration testing
- Helps avoid the well-tuned testing platform syndrome
- Almost universally used in FOSS projects

Degree of Ceremony

- Methodology Size — Number of control elements — deliverables, standards, activities, quality measures, techniques
- Methodology Ceremony — amount of precision and the tightness of tolerance
- Methodology Weight — size times ceremony
- Agile stresses 'barely sufficient' ceremony and lightweight methods

System criticality — Cockburn's classification

- Amount of damage from undetected defects
- Loss of comfort (C) — no veggie meals available at lunch — fixed by buying an alternative
- Loss of discretionary monies (D) — mistakes in invoice — fixed by human backup
- Loss of essential monies (E) — mistakes can't be fixed by a phone call
- Loss of life (L) — mistakes can't be fixed, period
- Amount of Methodology weight proportional to system criticality and number of people
- XN metric (X - criticality N - number of people): $D_{10} < D_{20}$; $D_{10} < E_{10}$

Structure of a Methodology

- Roles — who you employ, what they do and what skills they need — project manager, designer...
- Teams — groups of roles
- Techniques — procedures used to accomplish tasks — writing a use case...
- Activities — how people spend their time — planning, programming...
- Process — how activities fit together with pre- and post-conditions — do design review two days after X
- Work products — what someone constructs — program, user manual...
- Deliverable — work product that gets passed across organizational boundaries
- Milestones — events marking progress or completion
- Standards — conventions the team adopts for tools, products and policies — using portable Java, using Eclipse...
- Values — aggressive, cautious...

Exercise 4 — Compare and Contrast: IID vs. SDLC

1. Compare and Contrast IID vs. SDLC

- a. Define each in one sentence.
- b. List what they have in common.
- c. List how they differ.

2. Discuss with group and make an agreed upon definition for a and list of five-ten key points raised for b and c.

3. Someone different presents to the class.

The Project+ Game

The instructor will read questions from the IT Project+ Study Guide (by William Helman and Lona Cram, Second Edition, Sybex, 2004 ISBN 0-7821-4318-0). The class will discuss and respond. This will be allow review of concepts required for CompTIA's Project+ exam. (Due to copyright restrictions we cannot include materials here). Students interested in taking the exam are encouraged to purchase the book.