

<http://tadsessions.espace.com.eg>

**Agile Software Development
&
Extreme Programming (XP)**

Introduction and Practices

By: Ali Maher



- Software Development Process: is the total set of **software engineering activities** necessary to develop and maintain software products.

- Requirements specification (Analysis)
- Design
- Construction (implementation or coding)
- Integration
- Testing and debugging
- Installation (deployment)
- Maintenance

- Prescribes the construction of **initially small** portions of a software project to help **all those involved** to uncover important issues **early** before problems or faulty assumptions can lead to **disaster**.
- Agile software development processes are **built on** the foundation of iterative development. Agile processes use **feedback**, rather than planning, as their primary control mechanism.

- 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.
- Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

- Principle #1 (Customer Satisfaction): Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.

- Principle #11 (Requirement Changing) :
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Principle #III (Frequent Delivery):
 - **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
 - **Working software** is the primary measure of progress.

- Principle #IV (**Agile Team**):
 - Business people and developers must work together daily throughout the project.
 - Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
 - The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
 - The best architectures, requirements, and designs emerge from self-organizing teams.

- XP is a **style** of software development focusing on **excellent application** of programming **techniques**, clear **communication**, and **teamwork** which allows us to accomplish things we previously could not even imagine. [*Extreme Programming Explained, Kent Beck, 2nd edition*]
- *XP is a **pragmatic** approach to program development that emphasizes **business results first** and takes an **incremental**, get-something-started approach to building the product, using **continual** testing and revision.*

- The life time of the project is divided into small time slices called **iteration**. A typical iteration time is one to three weeks.
- Customer describes the feature he want to be added to the system and his description is documented by the team in a **Story**.
- The story is divided into a list of **Tasks**. Each task is **estimated** by the assigned (sub) team.

- Design Patterns
- Prototyping
- Simple and uses as fewest classes and methods as possible.
- Mocks
- Spike solutions

- One writes **automated tests**.
- Coding is done by **pair** of programmers.
- Coding is done when it **passes all tests**.

- **Refactoring:** is a change to the system that leaves its behavior unchanged, but enhances some nonfunctional quality – simplicity, flexibility, understandability, and/or performance.
- Design and architecture emerge out of **refactoring**, and come **after** coding.
- Refactoring is done by the **same** people who did the code.

- The **incomplete** but **functional** system is **deployed** or demonstrated for (some subset of) the users (at least one of which is on the **development** team).
- At this point, the practitioners start again on writing tests for the next most important part of the system.

- Planning Game
- Small Releases
- Metaphor
- Simple Design
- Functional Testing
- Unit Testing
- Refactoring
- Collective Ownership
- Coding Standards
- Continuous Integration
- On-site Customer
- Forty Hour Week
- Pair Programming

Why XP is extreme?

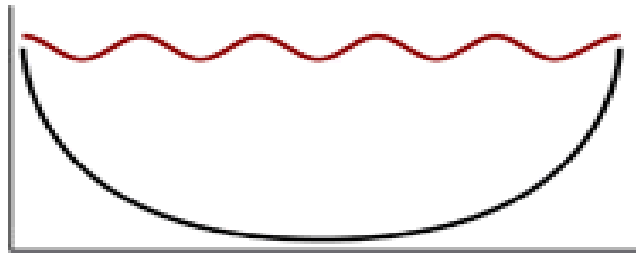


XP takes **commonsense** principles and practices to **extreme levels**.

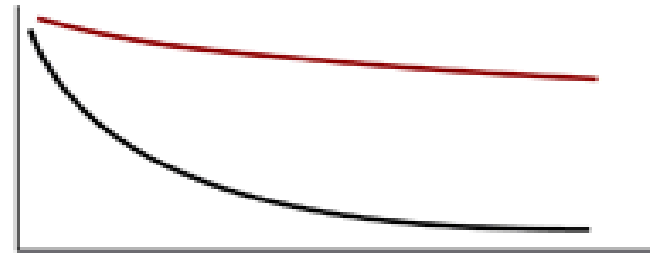
- Review code all the time (**pair programming**).
- Everybody will test all the time (**unit testing**), even the customers (**functional testing**).
- Design is part of everybody's business (**refactoring**).
- Always leave the system with the simplest design that supports its current functionality (**simplest thing that could possibly work**).
- Everybody works defining and refining the architecture all the time (**metaphor**).
- Integrate and test several times a day (**continuous integration**).
- Iterations really, really short – seconds and minutes and hours, not weeks and months and years (**the Planning Game**).

Agile Development Value Proposition

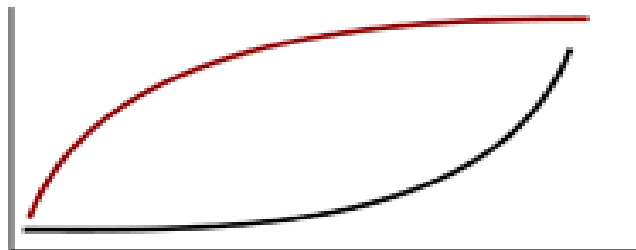
VISIBILITY



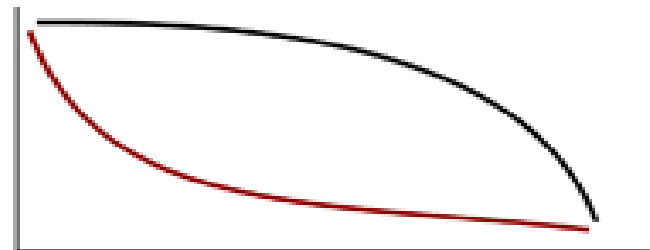
ADAPTABILITY



BUSINESS VALUE



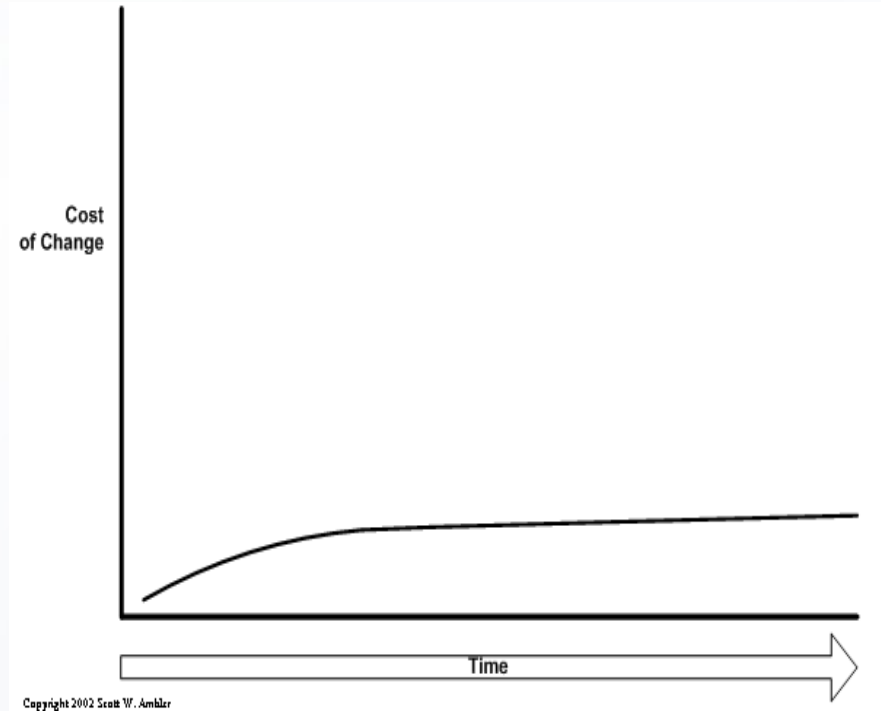
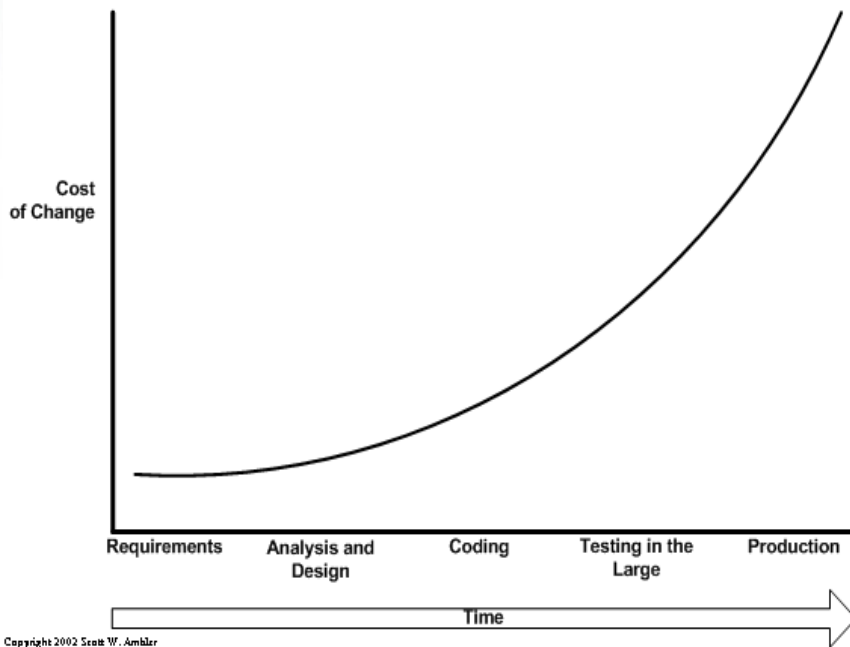
RISK



— AGILE DEVELOPMENT

— TRADITIONAL DEVELOPMENT

Extreme Programming Improves Cost of Change



- Increased productivity:
 - 2006: 10% or higher improvements reported by 87% of respondents
 - 2007: 10% or higher improvements reported by 83% of respondents
 - 2006: 25% or higher improvements reported by 55% of respondents
 - 2007: 25% or higher improvements reported by 55% of respondents

- Reduced software defects:
 - 2006: 10% or higher improvements reported by 86% of respondents
 - 2007: 10% or higher improvements reported by 85% of respondents
 - 2006: 25% or higher improvements reported by 55% of respondents
 - 2007: 25% or higher improvements reported by 54% of respondents



- Reduced cost:
 - 2006: 10% or higher improvements reported by 63% of respondents
 - 2007: 10% or higher improvements reported by 28% of respondents
 - 2006: 25% or higher improvements reported by 26% of respondents
 - 2007: 25% or higher improvements reported by 28% of respondents

- Other interesting improvements reported, among them:
 - Enhanced ability to manage changing priorities
 - Alignment between IT and business goals
 - Improved team morale
 - Reduced project risk

