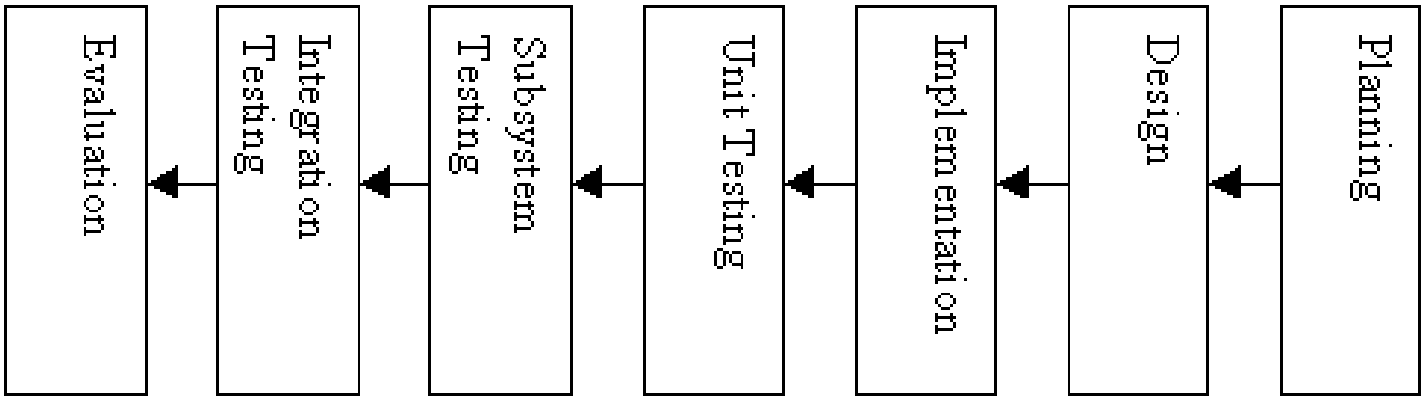
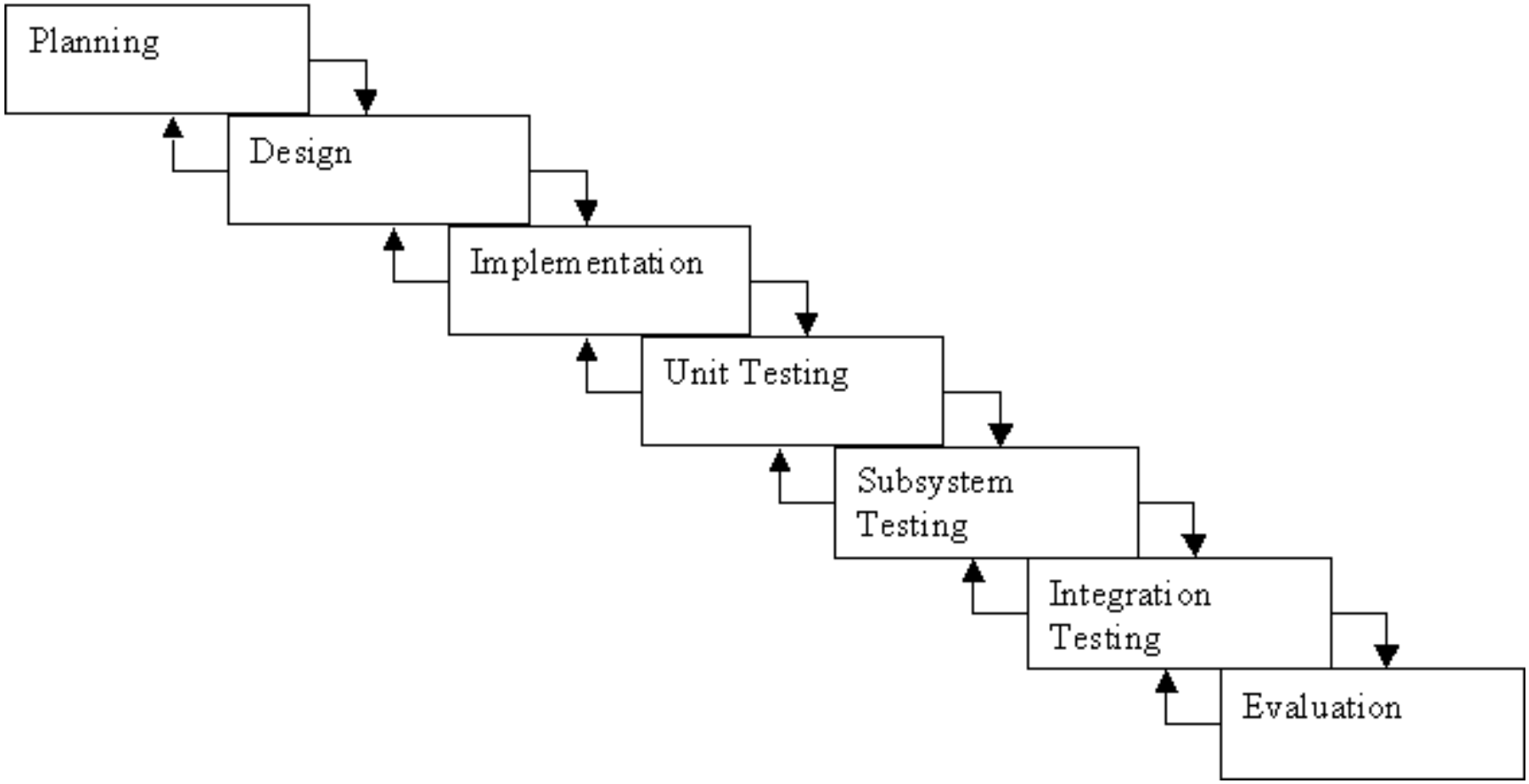
Software Engineering Methodologies and XP

See also: Chapter 6

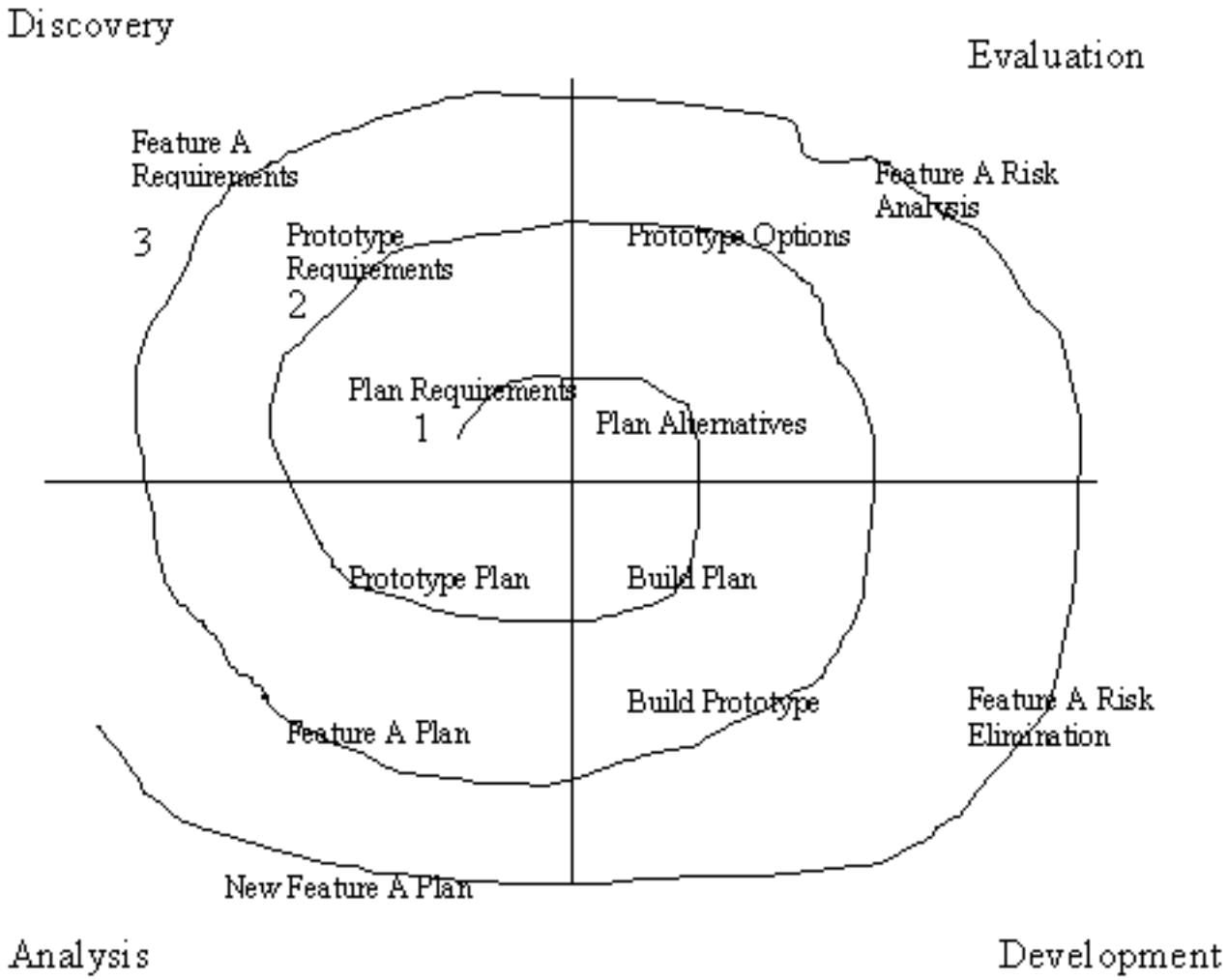
The Stagewise Model



The Waterfall Model



The Spiral Method



Extreme ~~Beach Volleyball~~ Programming (XP)

Extreme Programming is a collection of best practices and development guidelines that collectively make up a methodology with a really hokey name. See also: Agile Software Development

Why XP?

XP Recognizes that:

- Software development is difficult, unpredictable.
- Writing software isn't like designing a building.
- Software developers are in the best position to manage the process.

Plan As You Go

The notion of planning a large release, like "version 2.0" is frowned upon in XP.

Non-XP	Agile / XP
- Long kickoff meeting - 6 month+ release cycle - Frantic development towards the end - Time vs. Features vs. Bugs - Big party, T-Shirts	- Features presented weekly/monthly - < 5 day tasks - Always in release mode - Sustainable development - Fewer T-Shirts

Small Releases

… because it wasn't originally called "Windows *95*"

- Customers prefer smaller releases
- Easier to update these days (Automatic updates, web-based)
- Less risk of missing deadlines, easier to respond to feedback and new requirements

Share a Common Metaphor

Teams work better if they have a common mental picture of the software and shared terminology.

Simple Designs

XP Aphorism: *Avoid Speculative Generality*

Create the simplest possible thing that will get the job done.

Software is meant to be dynamic – you can change it later.

Test Constantly

The XP Logic:

Unit tests are good, so:

- We should require unit tests for a feature to be complete.
- We should write the test before the feature to prove that the feature is complete.
- We should define features in terms of their *acceptance tests*.
- We should have automated tests that will immediately tell us when something is wrong.

Refactor When Necessary

If I knew then what I know now…

Traditionally: Refactor or rewrite at the start of a major release

XP: Refactor the moment you see something that shows signs of needing to be refactored:

- Special cases
- Bug fixes
- Messy code
- No longer does a single task
- Code that nobody wants to touch
- Too general or too specific

Code in Pairs (the hippie part)

When people code together they:

- Do things properly (comment, unit test, check for NULL, etc.)
- Help each other think things through (one person codes)
- Catch more mistakes
- Bring expertise about a particular area of code
- Share knowledge
- Do less instant messaging / web surfing :-(

So….

XP says we should code in pairs all the time. Claims that the 50% reduction in effective coding staff is balanced by the increased speed of development.

Share the Code (aka Collective Ownership)
(the other hippie part)

When everybody is responsible for all the code:

- Anybody can fix a bug (bus insurance)
- There's less finger pointing
- No one person gets overworked
- Subsystems work well together
- Nobody turtles

Integrate Continuously

Integrate: To join distinct subsystems together, or to merge new code into the existing code base.

Traditional integration: painful, time-consuming

XP integration: protected by unit tests, more frequent, but less painful.

Work Sane Hours

You mean all-nighters + coffee *doesn't* result in good code?

Have a Customer On Site

All requirements should come from the customer (as directly as possible). Otherwise, why are we writing this software? And what are we doing in this handbasket?

Note: Your "customer" is often represented by a Product Manager.

Share Common Coding Standards

See also: Collective Ownership

XP: An "all or nothing" methodology?

Build Small Releases
Share a Common Metaphor
Simplify Your Designs
Test Constantly
Refactor When Necessary
Code in Pairs
Share the Code
Integrate Continuously
Work Sane Hours
Have a Customer On Site
Share Common Coding Standards

Roll Your Own Methodology

- Try different methodologies
- Be open to new ideas
- Keep what works
- Listen to the team
- Fine-tune with each iteration
- Give the process a chance