Negotiations for Start-ups

LECTURER KHOLOVA GULNORA

Practitioner stories

“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 et al

What is “Agility”?

Effective (rapid and adaptive) response to change

Effective communication among all stakeholders

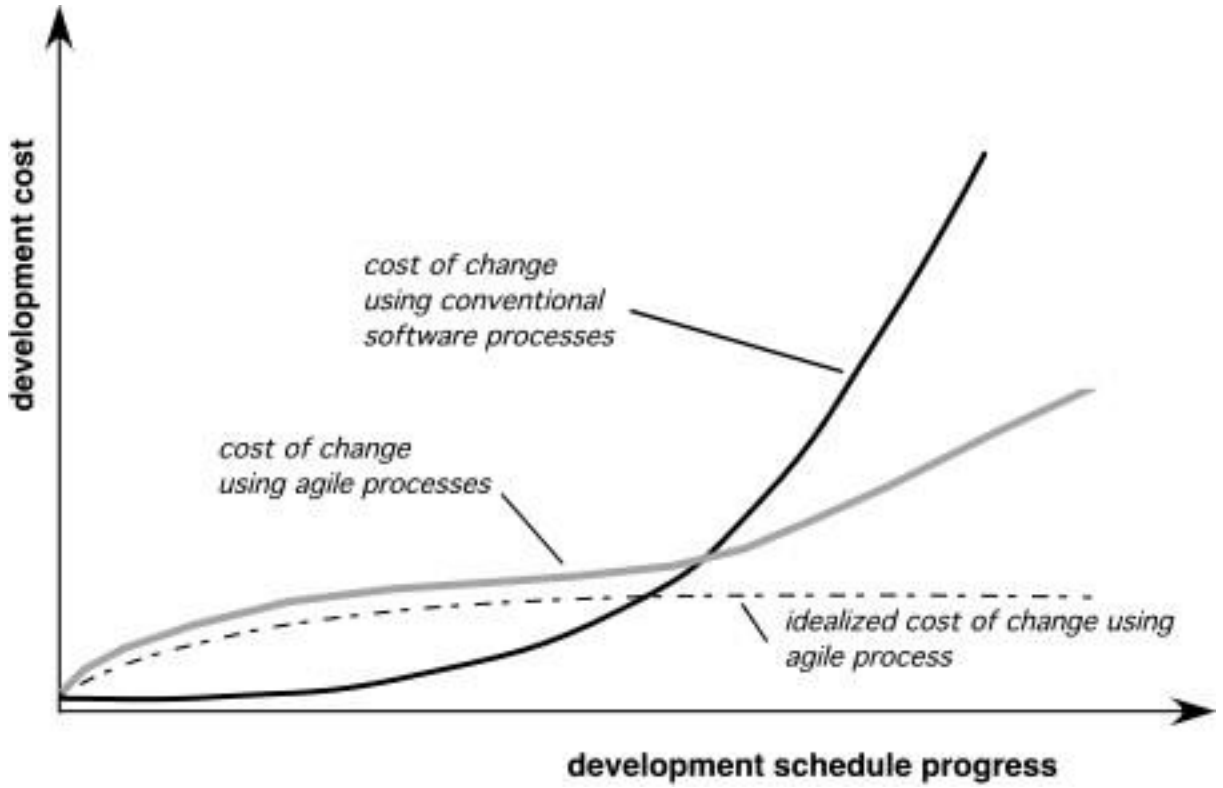
Drawing the customer onto the team

Organizing a team so that it is in control of the work performed

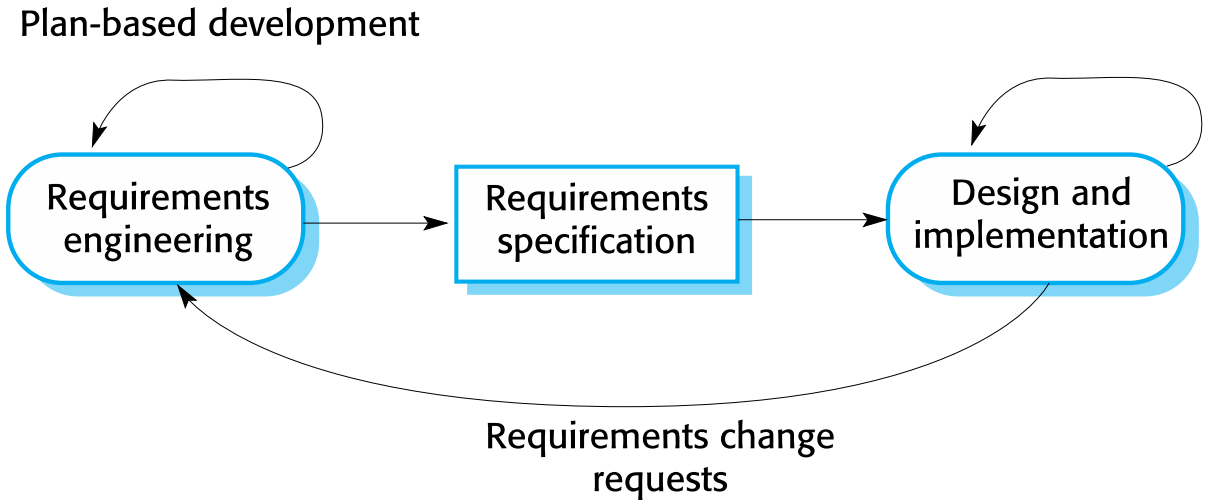
Yielding …

Rapid, incremental delivery of software

Agility and the Cost of Change



Plan-driven and agile specification



An Agile Process

Is driven by customer descriptions of what is required (scenarios)

Recognizes that plans are short-lived

Develops software iteratively with a heavy emphasis on construction activities

Delivers multiple ‘software increments’

Adapts as changes occur

Agility Principles - I

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.

Agility Principles - II

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self–organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
- Custom system development within an organization, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems.

Problems with agile methods

It can be difficult to keep the interest of customers who are involved in the process.

Team members may be unsuited to the intense involvement that characterises agile methods.

Prioritising changes can be difficult where there are multiple stakeholders.

Maintaining simplicity requires extra work.

Contracts may be a problem as with other approaches to iterative development.

Agile methods and software maintenance

- Most organizations spend more on maintaining existing software than they do on new software development. So, if agile methods are to be successful, they have to support maintenance as well as original development.
- Two key issues:
  - Are systems that are developed using an agile approach maintainable, given the emphasis in the development process of minimizing formal documentation?
  - Can agile methods be used effectively for evolving a system in response to customer change requests?
- Problems may arise if original development team cannot be maintained.

Plan-driven and agile development

Plan-driven development

- A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
- Not necessarily waterfall model – plan-driven, incremental development is possible
- Iteration occurs within activities.

Agile development

- Specification, design, implementation and testing are inter-leaved and the outputs from the development process are decided through a process of negotiation during the software development process.

Human Factors

the process molds to the needs of the people and team, not the other way around

key traits must exist among the people on an agile team and the team itself:

- **Competence.**
- **Common focus.**
- **Collaboration.**
- **Decision-making ability.**
- **Fuzzy problem-solving ability.**
- **Mutual trust and respect.**
- **Self-organization.**

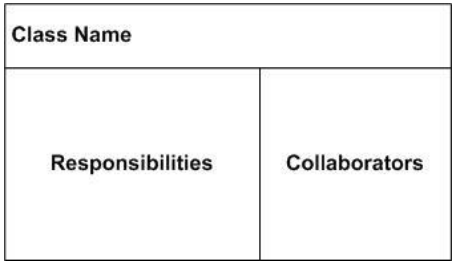
Extreme Programming (XP)

The most widely used agile process, originally proposed by Kent Beck

XP Planning

- Begins with the creation of "user stories"
- Agile team assesses each story and assigns a cost
- Stories are grouped to for a deliverable increment
- A commitment is made on delivery date
- After the first increment "project velocity" is used to help define subsequent delivery dates for other increments

Extreme Programming (XP)



XP Design

- Follows the KIS principle
- Encourage the use of CRC cards (see Chapter 8)
- For difficult design problems, suggests the creation of "spike solutions"—a design prototype
- Encourages "refactoring"—an iterative refinement of the internal program design

XP Coding

- Recommends the construction of a unit test for a store *before* coding commences
- Encourages "pair programming"

XP Testing

- All unit tests are executed daily
- "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

What is CRC card ?

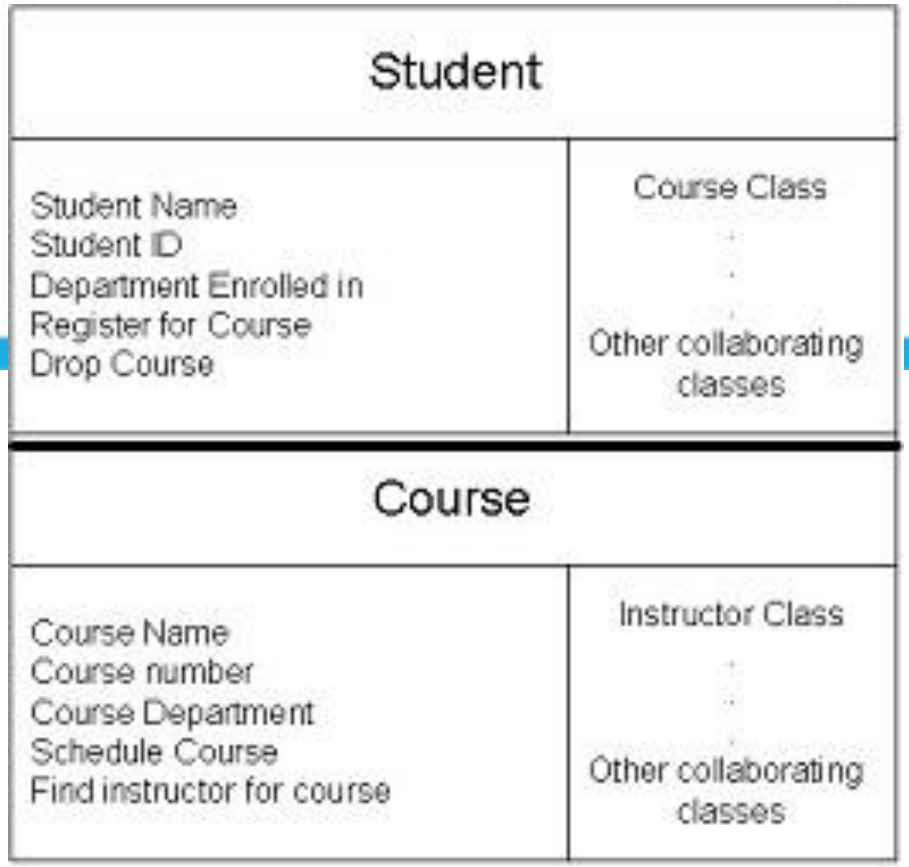
CRC stands for Class, Responsibilities, Collaborations

A brainstorming tool used in the design of object-oriented software.

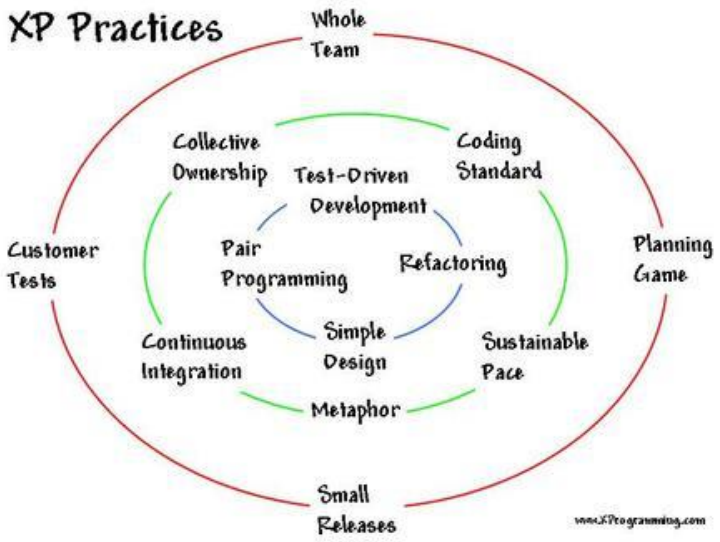
The cards are arranged to show the flow of messages among instances of each class.

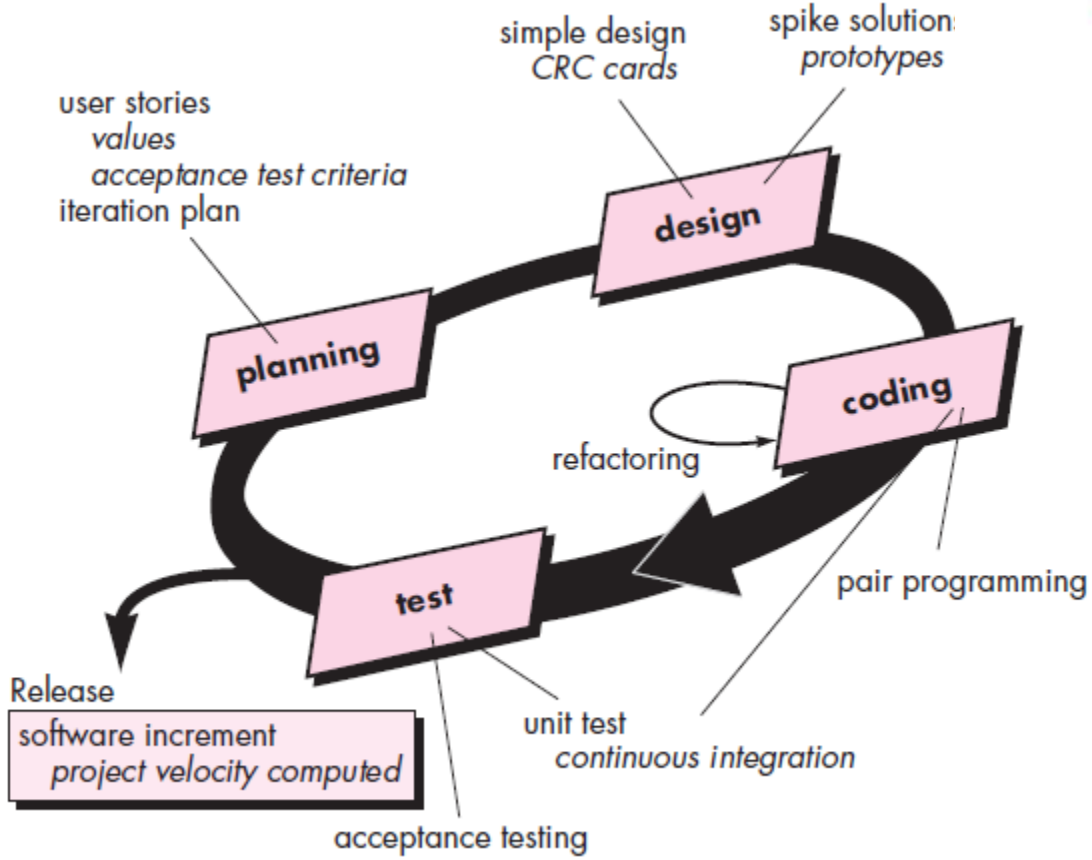
A collection of standard index cards that have been divided into three sections

1. **A class represents a collection of similar objects.**
2. **A responsibility is something that a class knows or does.**
3. **A collaborator is another class that a class interacts with to fulfill its responsibilities.**



Extreme Programmi





User Stories
Who (often called role) and what
Why (optional)
Easy to understand
Short
Indicates measures of success

User Story Examples (1 of 2)

As a smart phone user, I want to be able to install the application.

As a smart phone user, I want to be able to uninstall the application.

As a business owner, I want to be able to accept credit cards.

As a business owner, I want to be able to receive confidential customer feedback.

User Story Examples (2 of 2)

As a dog owner, I want the dog to notify me when it needs to go out.

As a dog owner, I want the dog to sit when asked.

As a dog owner, I don't want the dog to bite humans.

As a dog owner, I want the dog to come when called.

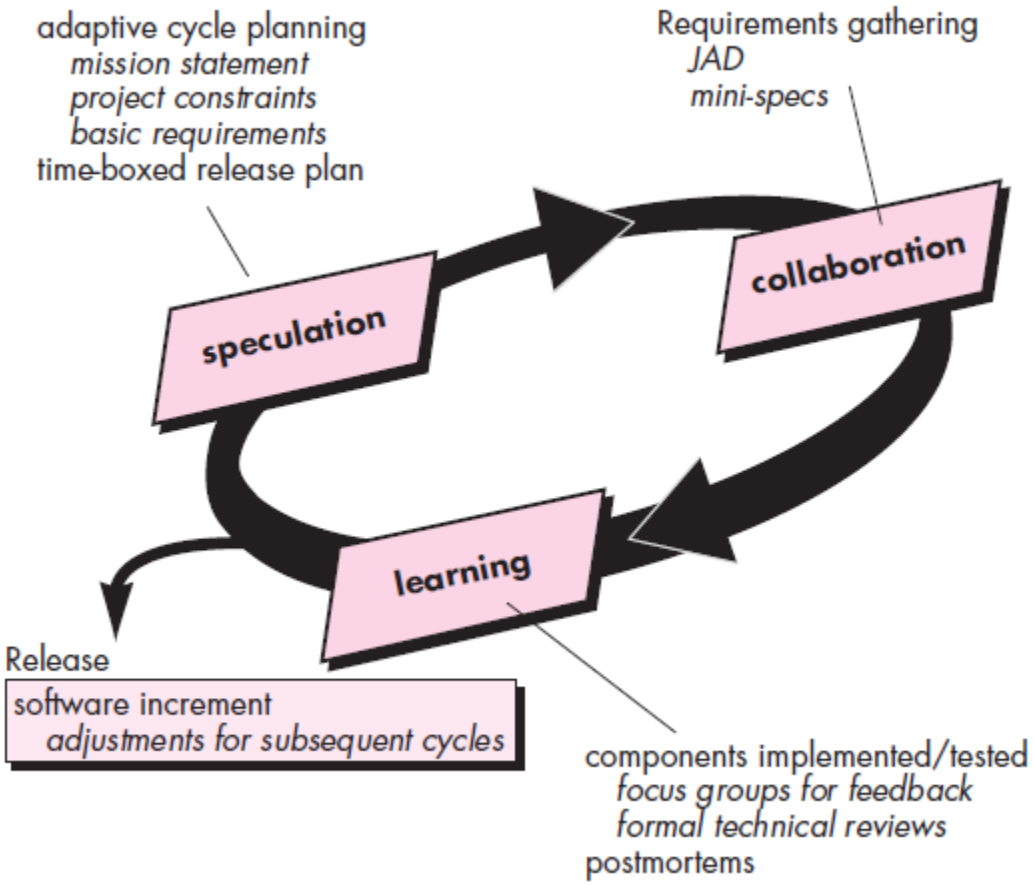
Adaptive Software Development

Originally proposed by Jim Highsmith

ASD — distinguishing features

- Mission-driven planning
- Component-based focus
- Uses "time-boxing" (See Chapter 24)
- Explicit consideration of risks
- Emphasizes collaboration for requirements gathering
- Emphasizes "learning" throughout the process

Adaptive Software Development



Dynamic Systems Development Method

Promoted by the DSDM Consortium (www.dsdm.org)

DSDM—distinguishing features

- Similar in most respects to XP and/or ASD
- Nine guiding principles
  - Active user involvement is imperative.
  - DSDM teams must be empowered to make decisions.
  - The focus is on frequent delivery of products.
  - Fitness for business purpose is the essential criterion for acceptance of deliverables.
  - Iterative and incremental development is necessary to converge on an accurate business solution.
  - All changes during development are reversible.
  - Requirements are baselined at a high level
  - Testing is integrated throughout the life-cycle.

Dynamic Systems Development Method

DSDM Life Cycle (with permission of the DSDM consortium)

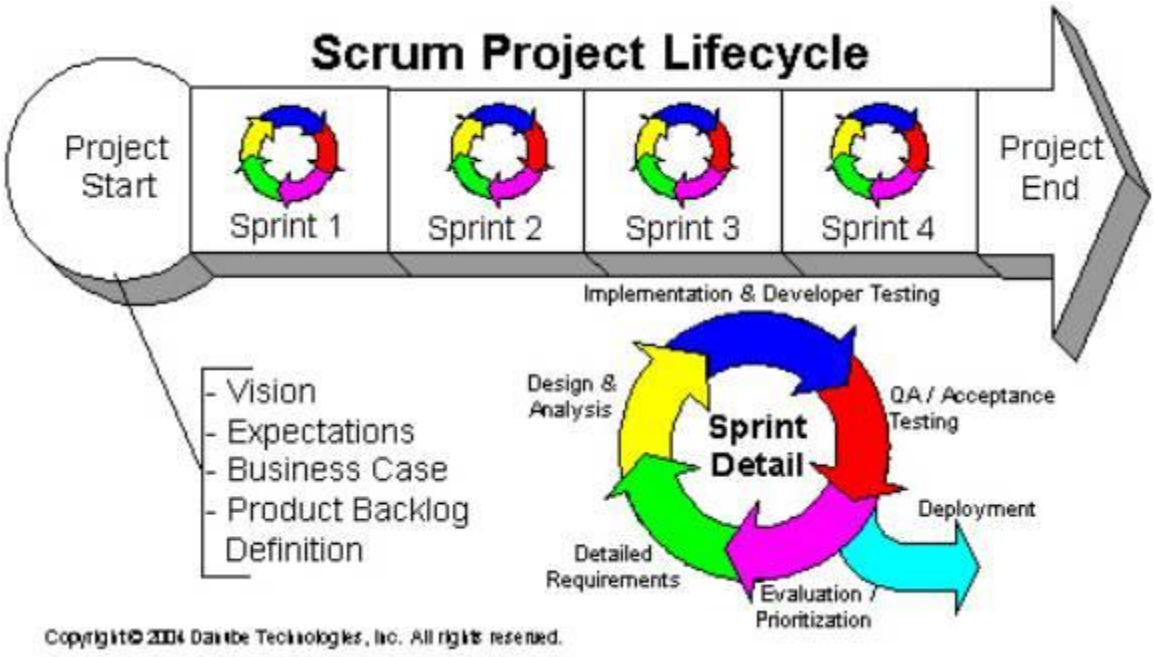
Scrum

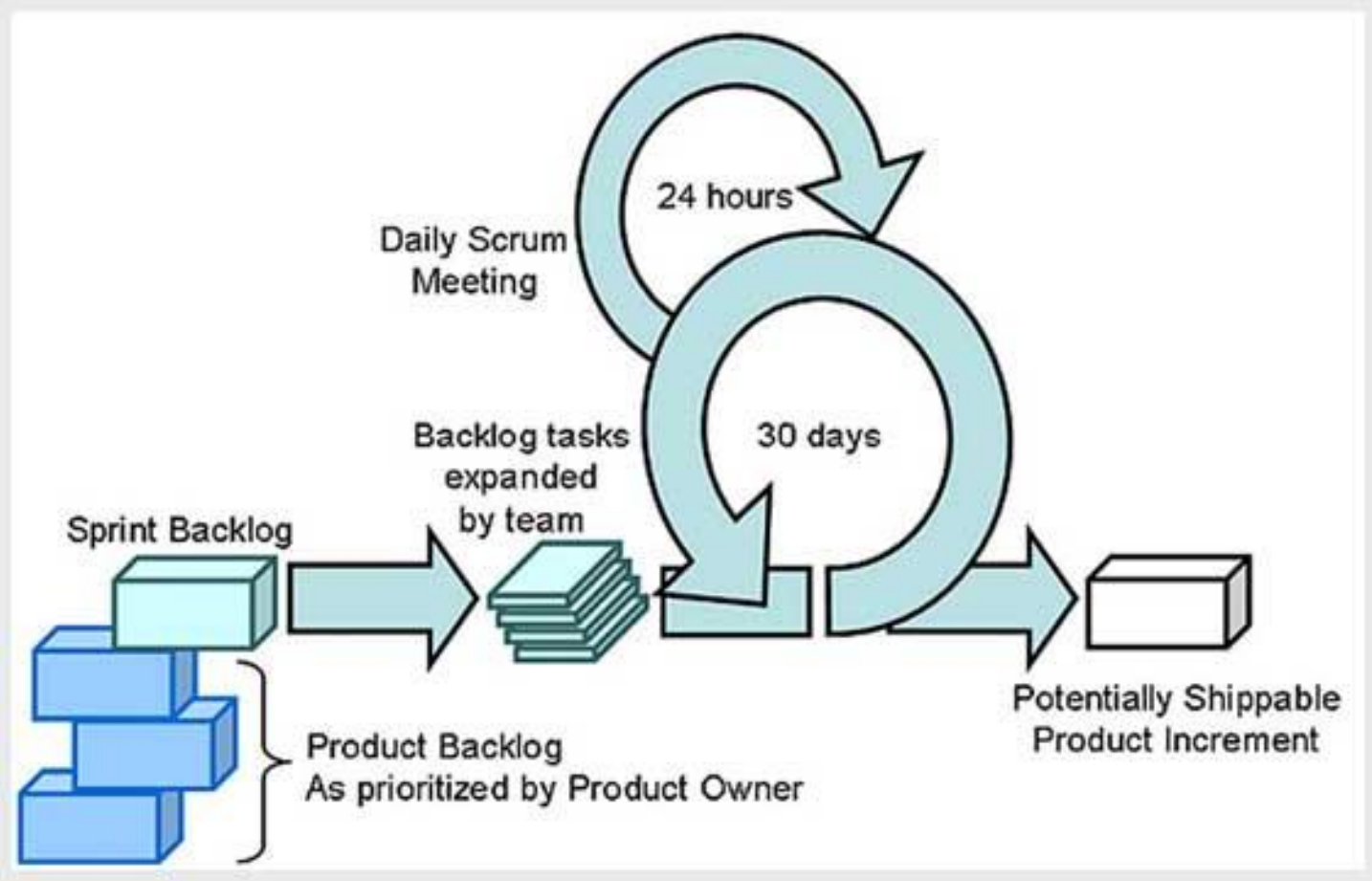
Originally proposed by Schwaber and Beedle

Scrum—distinguishing features

- Development work is partitioned into “packets”
- Testing and documentation are on-going as the product is constructed
- Work occurs in “sprints” and is derived from a “backlog” of existing requirements
- Meetings are very short and sometimes conducted without chairs
- “demos” are delivered to the customer with the time-box allocated

Scrum


Scrum Project Lifecycle
Project Start
Sprint 1
Sprint 2
Sprint 3
Sprint 4
Project End
Implementation & Developer Testing
- Vision
- Expectations
- Business Case
- Product Backlog Definition
Design & Analysis
QA / Acceptance Testing
Sprint Detail
Deployment
Detailed Requirements
Evaluation / Prioritization
Copyright© 2004 Danube Technologies, Inc. All rights reserved.


24 hours
Daily Scrum
Meeting
30 days
Backlog tasks
expanded
by team
Sprint Backlog
Potentially Shippable
Product Increment
Product Backlog
As prioritized by Product Owner

Crystal

Proposed by Cockburn and Highsmith

Crystal—distinguishing features

- Actually a family of process models that allow "maneuverability" based on problem characteristics
- Face-to-face communication is emphasized
- Suggests the use of "reflection workshops" to review the work habits of the team

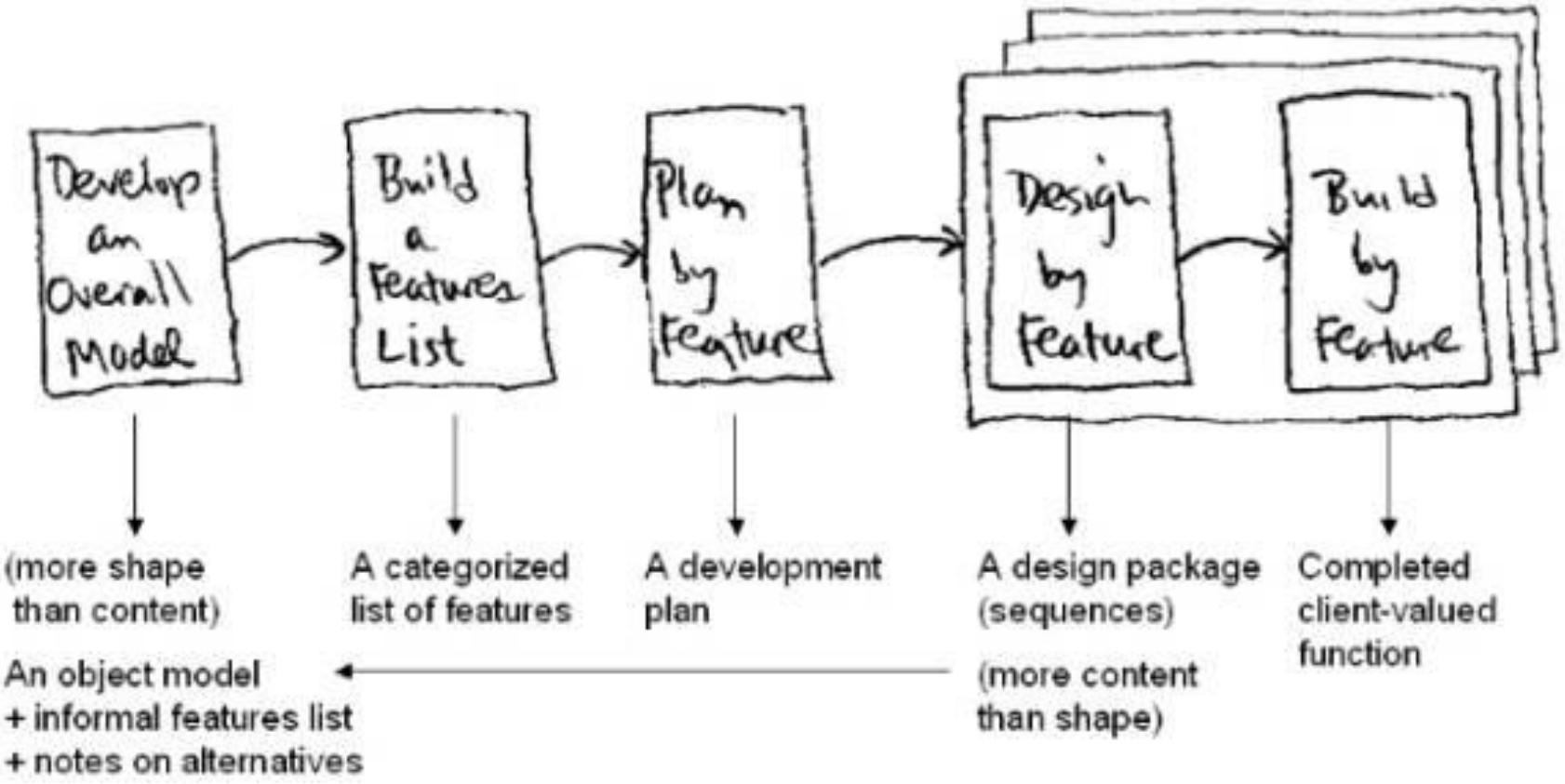
Feature Driven Development

Originally proposed by Peter Coad et al

FDD—distinguishing features

- Emphasis is on defining "features"
  - a *feature* "is a client-valued function that can be implemented in two weeks or less."
- Uses a feature template
  - <action> the <result> <by | for | of | to> a(n) <object>
- A features list is created and "plan by feature" is conducted
- Design and construction merge in FDD

Feature Driven Development



Agile Modeling

Originally proposed by Scott Ambler

Suggests a set of agile modeling principles

- Model with a purpose
- Use multiple models
- Travel light
- Content is more important than representation
- Know the models and the tools you use to create them
- Adapt locally

References and sourses

1. Contract Drafting and Negotiation for Entrepreneurs and Business Professionals – June 8, 2018 by Paul A. Swegle

2. Startup Law and Fundraising for Entrepreneurs and Startup Advisors – July 23, 2020 by Paul A. Swegle (Author)

3. Entrepreneurial Negotiation: Understanding and Managing the Relationships that Determine Your Entrepreneurial Success by Samuel Dinnar , Lawrence Susskind , et al. | Aug 16, 2018

4. Start-Up Saboteurs: How Incompetence, Ego, and Small Thinking Prevent True Wealth Creation by Ziad K. Abdelnour | Dec 31, 2020

5. 99 Negotiating Strategies: Tips, Tactics & Techniques Used by Wall Street's Toughest Dealmakers by David Rosen | Oct 4, 2016

6. Negotiating for Success: Essential Strategies and Skills by George J. Siedel | Oct 4, 2014

7. Venture Capital Deal Terms: A guide to negotiating and structuring venture capital transactions by Harm de Vries , Menno van Loon, et al. | Aug 1, 2016