BINUS UNIVERSITY

People
Innovation
Excellence

Course	: COMP6100/Software Engineering
Effective Period	: Desember 2017

The Software Process

Session 03

BINUS UNIVERSITY

Acknowledgement

These slides have been adapted from Pressman, R.S. (2015). *Software Engineering : A Practioner's Approach. 8th ed*. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157. Chapter 3 , 4, 5 and 6

People
Innovation
Excellence

Learning Objectives

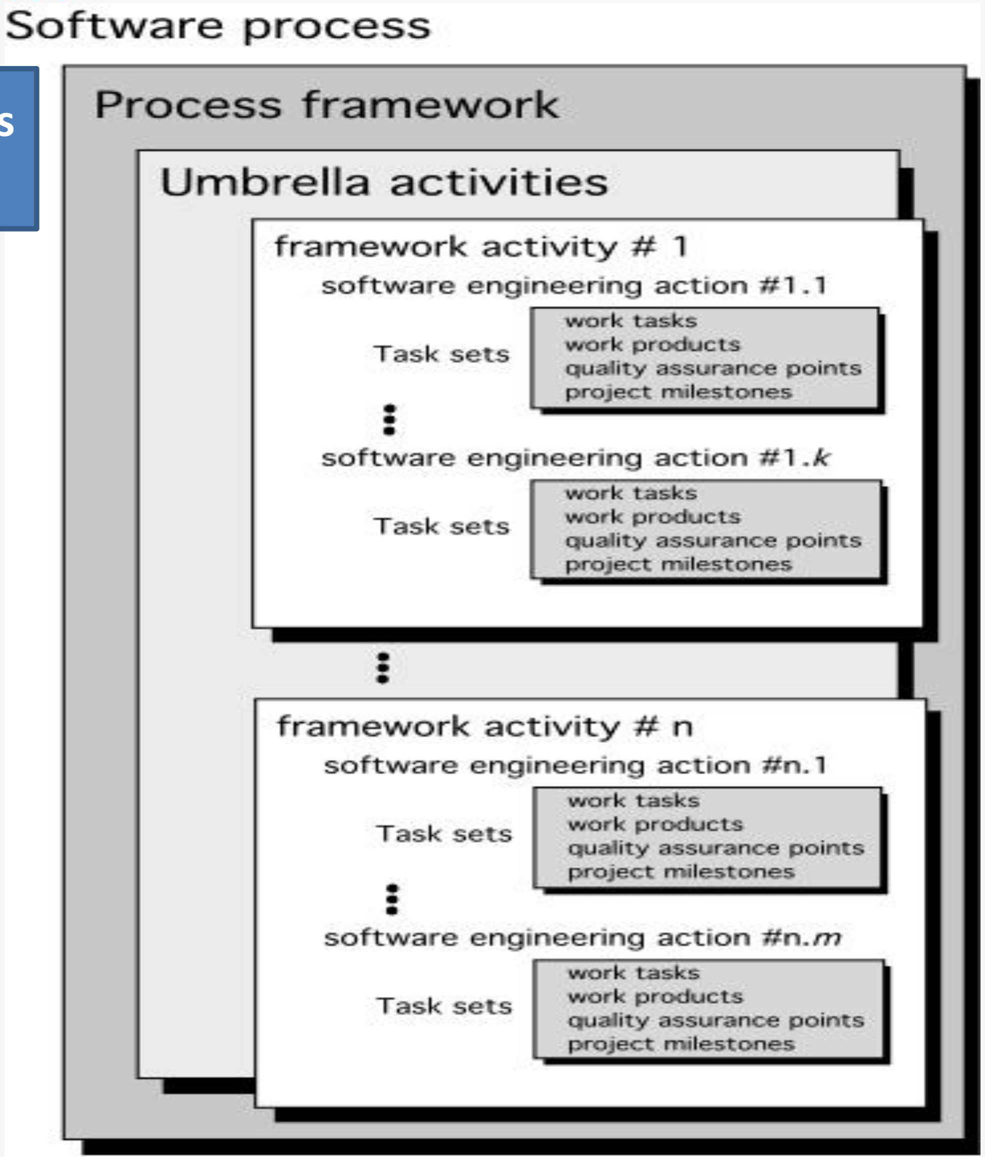
LO 1 : Describe the concepts of software process models and the opportunity for potential business project

Contents

- Software Process Structure
- Process Models
- Agile Development
- Human Aspects of Software Engineering

Software Process Structure

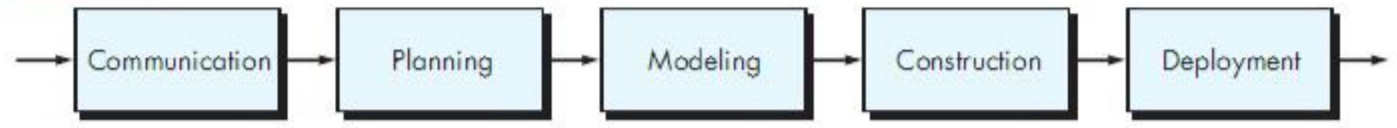
Software Process Framework



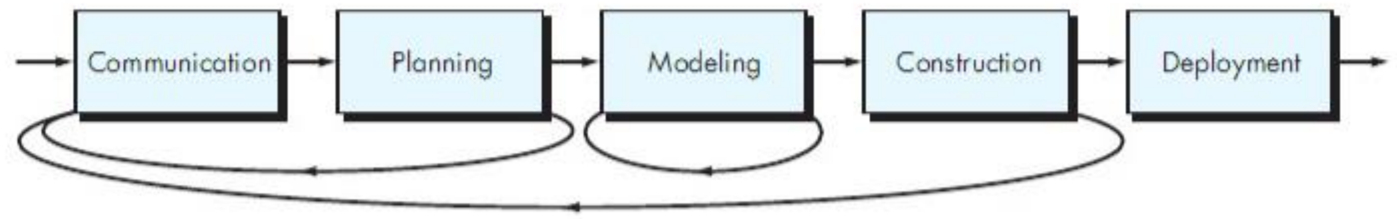
People
Innovation
Excellence

Software Process Structure

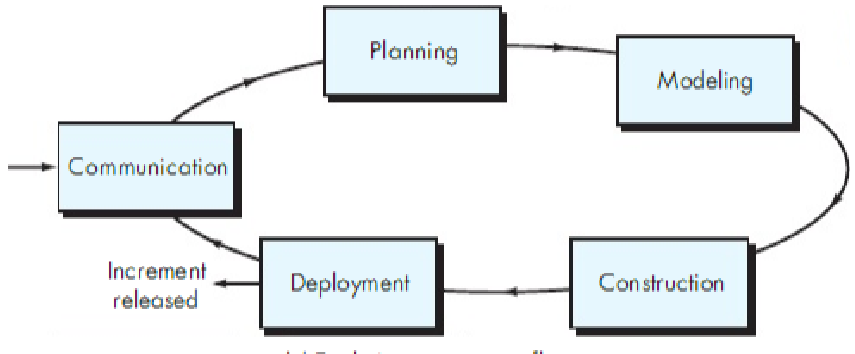
Process Flow



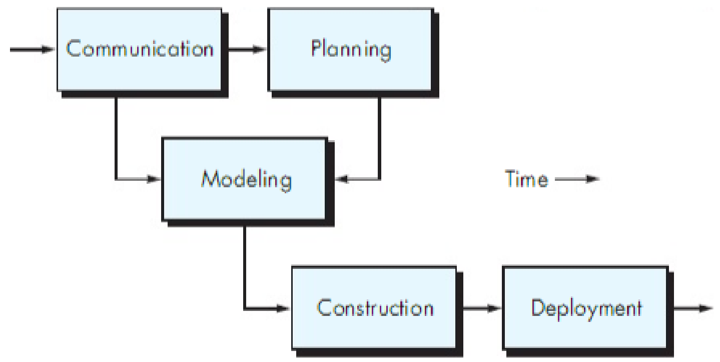
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

Software Process Structure

Identifying a Task Set

- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.
  - A list of the task to be accomplished
  - A list of the work products to be produced
  - A list of the quality assurance filters to be applied

BINUS UNIVERSITY

Software Process Structure

Process Patterns

- A *process pattern*
  - describes a process-related problem that is encountered during software engineering work,
  - identifies the environment in which the problem has been encountered, and
  - suggests one or more proven solutions to the problem.
- Stated in more general terms, a process pattern provides you with a *template* [Amb98]—a consistent method for describing problem solutions within the context of the software process.

Software Process Structure

Process Pattern Types

- ***Stage*** *patterns*—defines a problem associated with a framework activity for the process.
- ***Task*** *patterns*—defines a problem associated with a software engineering action or work task and relevant to successful software engineering practice
- ***Phase*** *patterns*—define the sequence of framework activities that occur with the process, even when the overall flow of activities is iterative in nature

Process Models

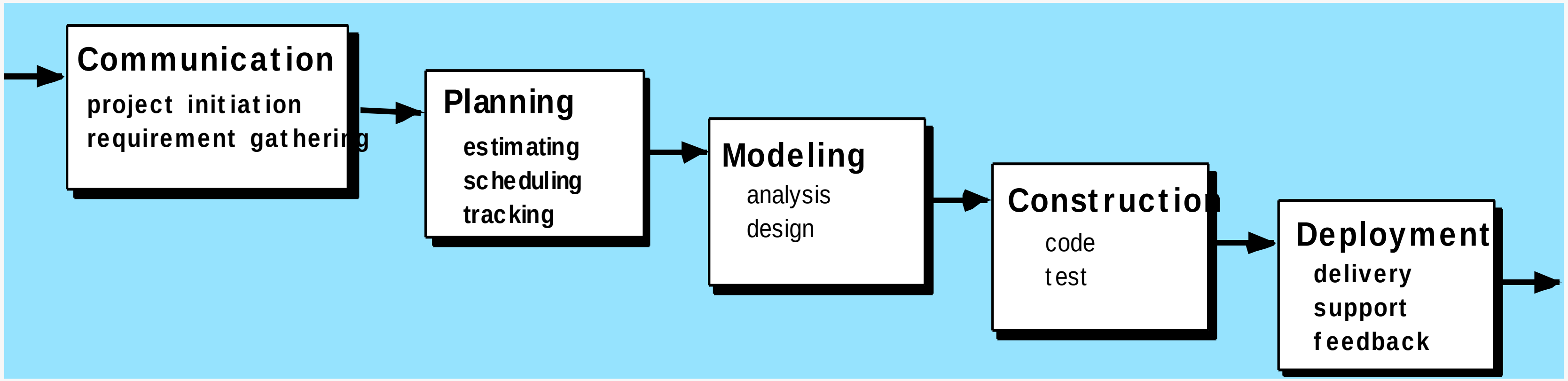
- Prescriptive process models advocate an orderly approach to software engineering

That leads to a few questions …

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

Process Models

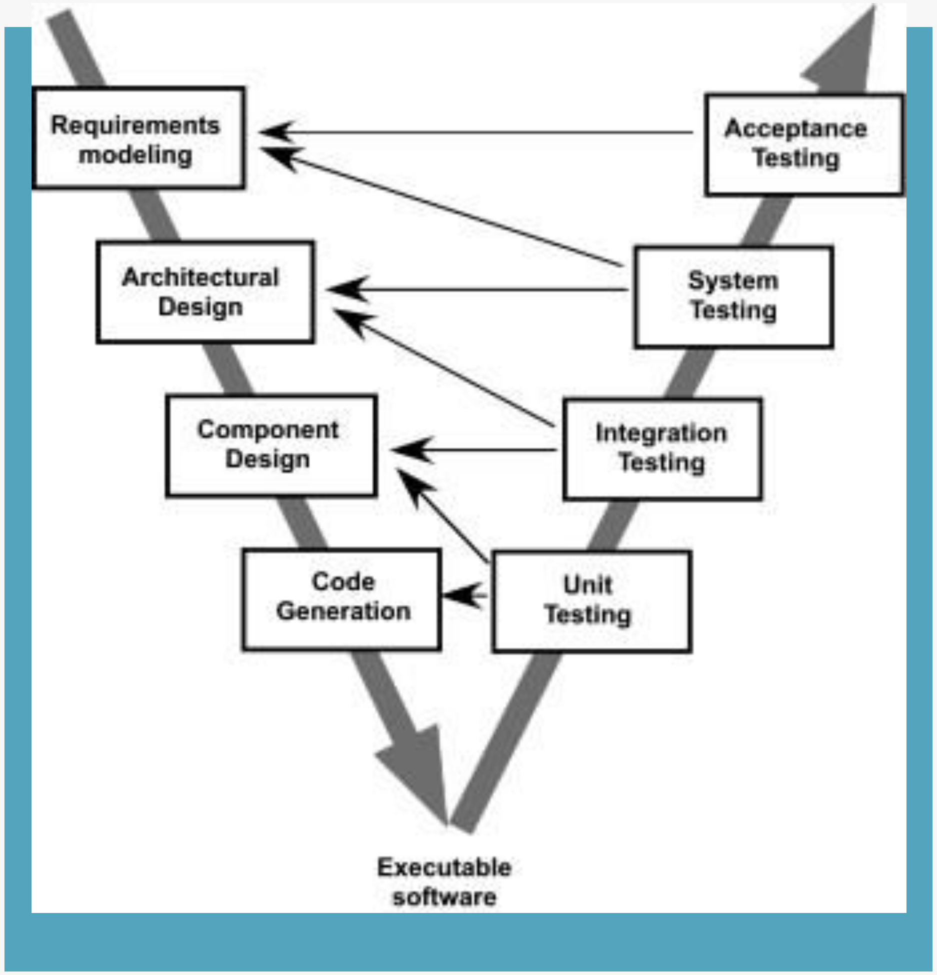
The Waterfall Model



People
Innovation
Excellence

Process Models

The V-Model

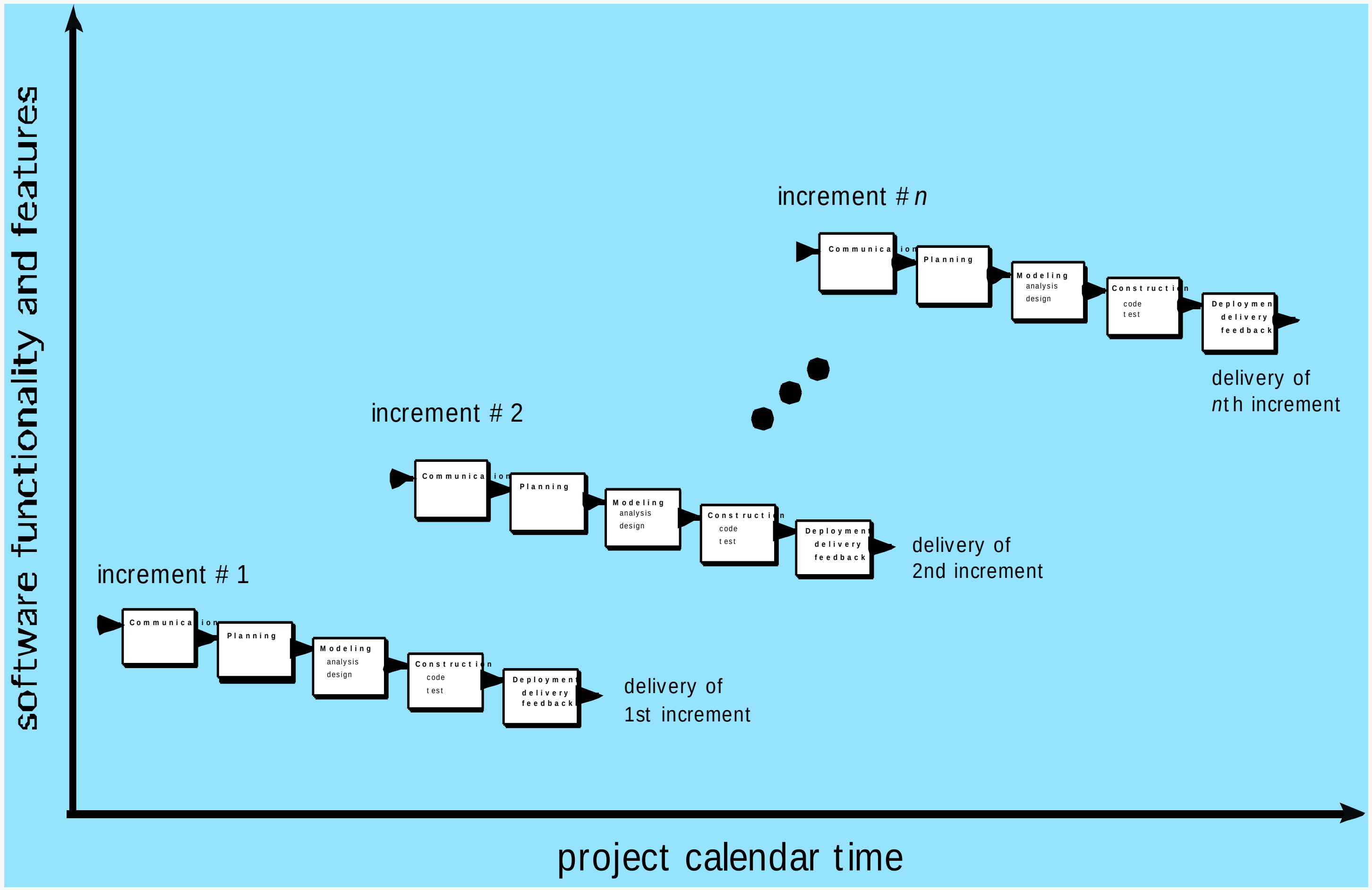


BINUS UNIVERSITY

People
Innovation
Excellence

Process Models

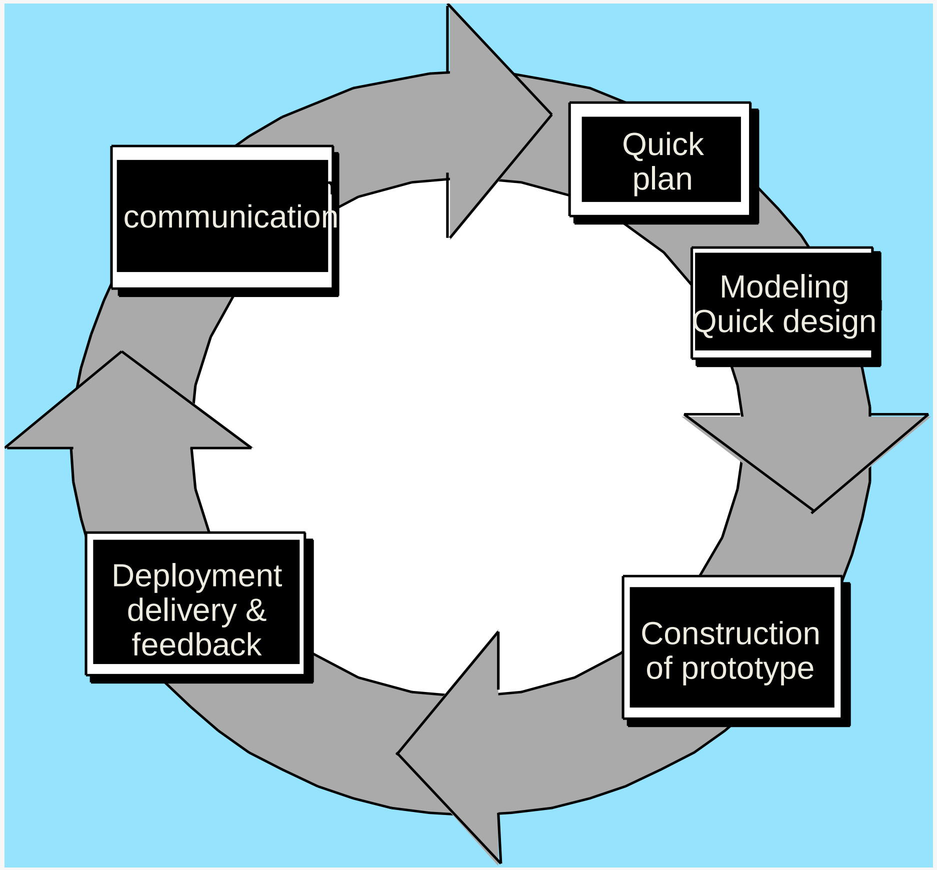
The Incremental-Model



Process Models

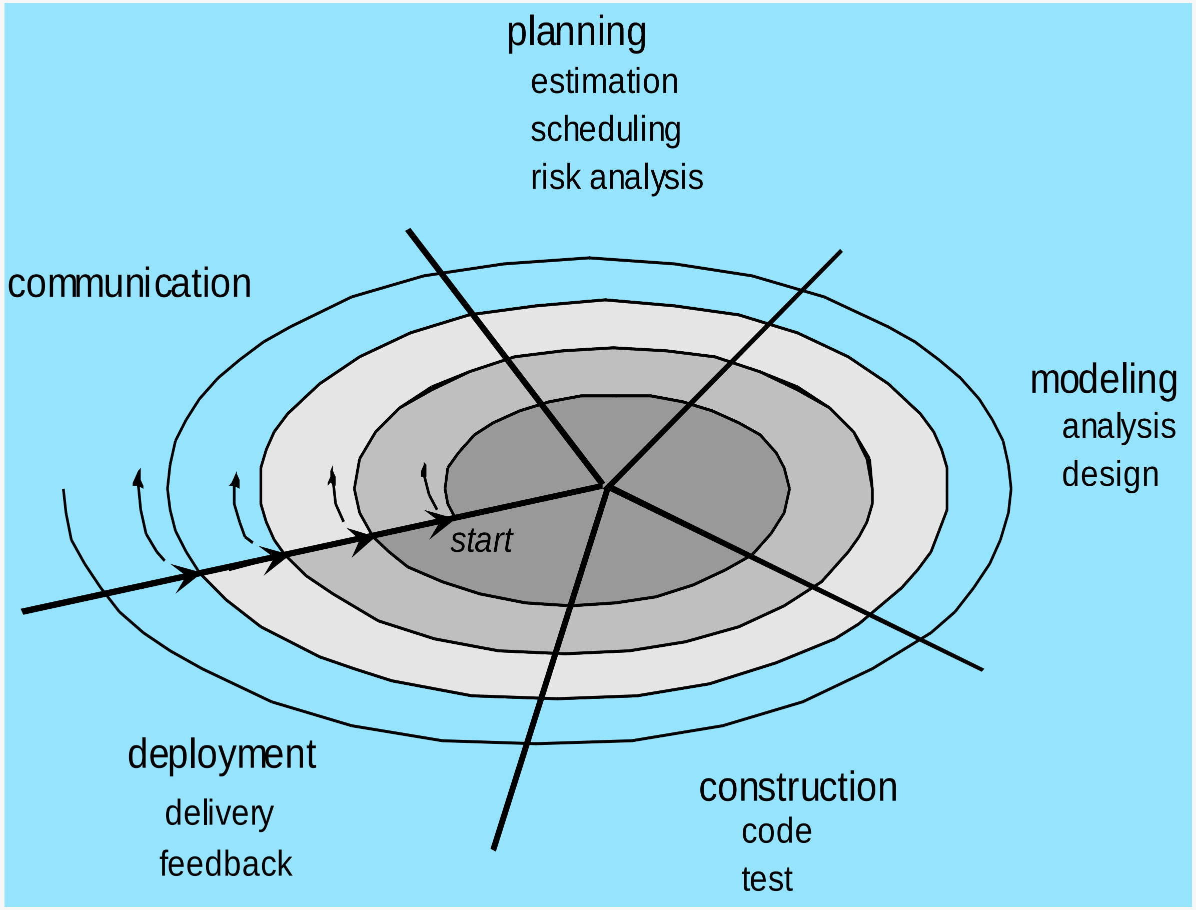
People
Innovation
Excellence

Evolutionary Models: Prototyping



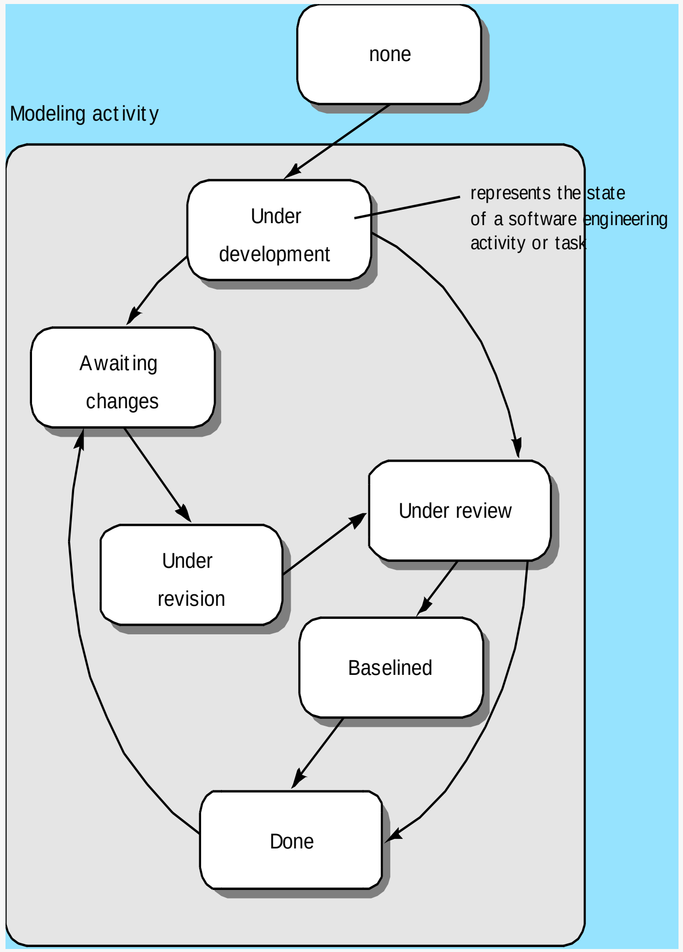
Process Models

Evolutionary Models: The Spiral



Process Models

Evolutionary Models: Concurrent



Specialized Process Models

- **Component based development**—the process to apply when reuse is a development objective
- **Formal methods**—emphasizes the mathematical specification of requirements
- **AOSD**—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- **Unified Process**—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

People
Innovation
Excellence

Agile Development

What is "Agility"?

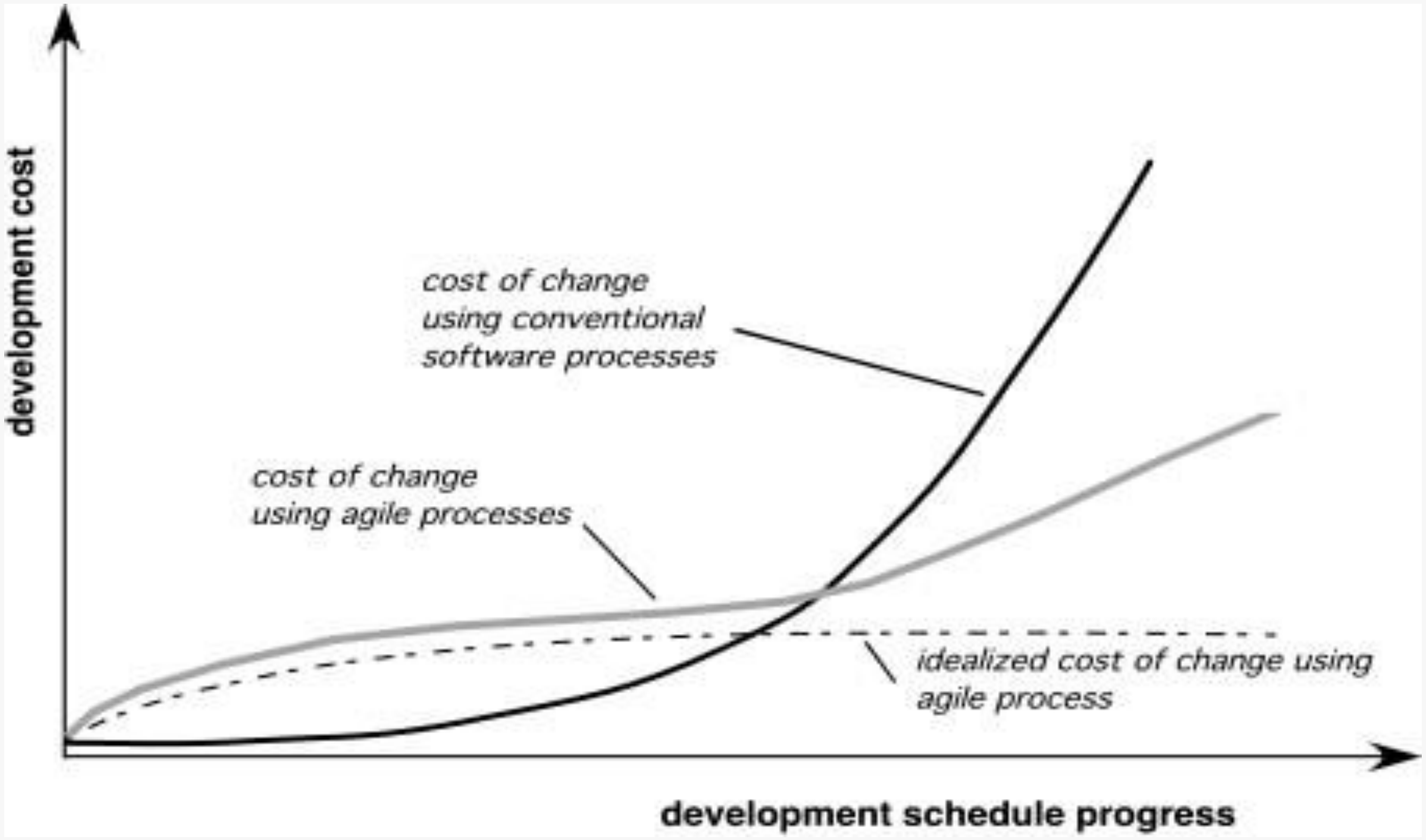
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

Agile Development

Agility and the Cost of Change



People
Innovation
Excellence

Agile Development

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

BINUS UNIVERSITY

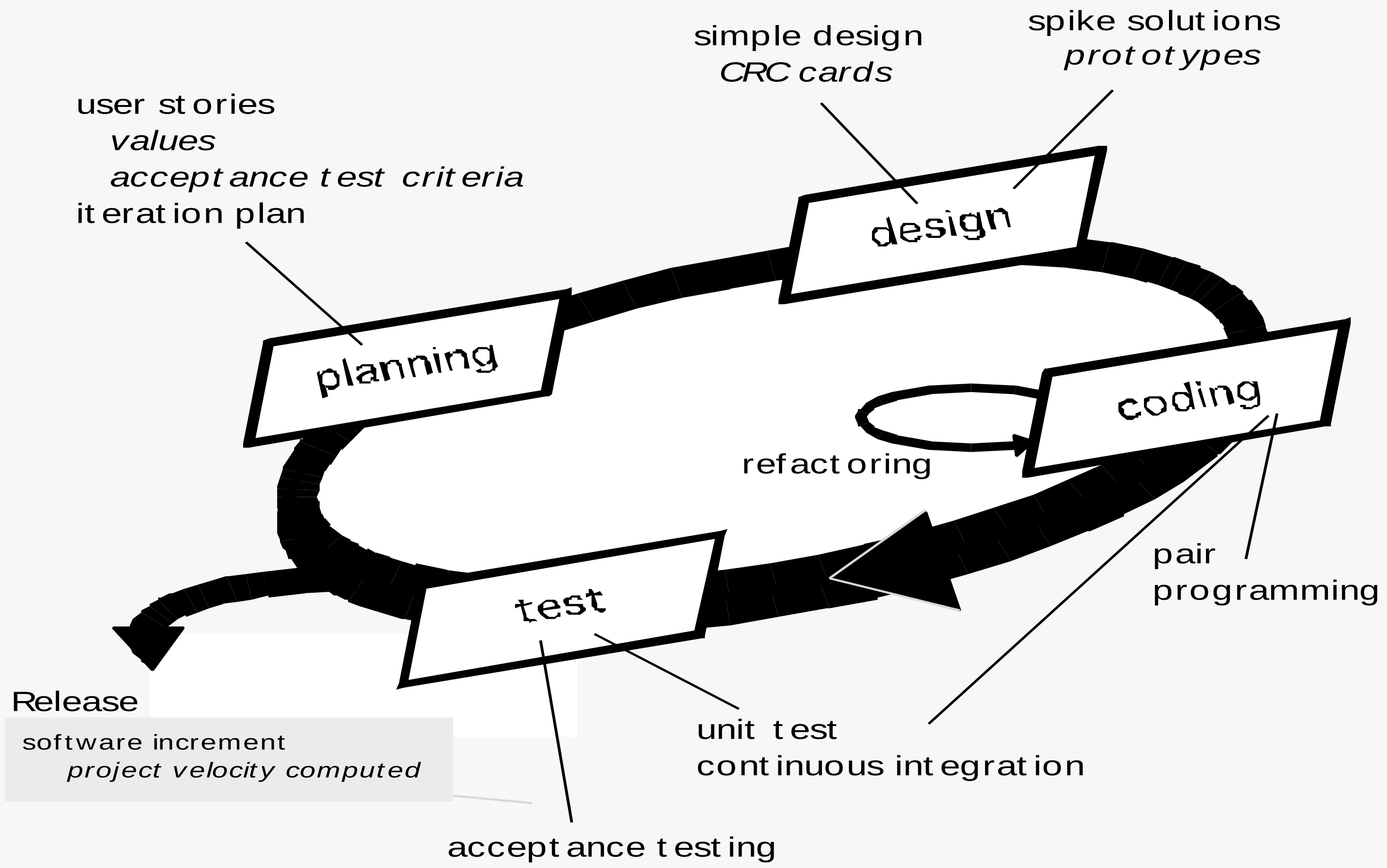
Agile Development

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

Agile Development

Extreme Programming (XP)



People
Innovation
Excellence

BINUS UNIVERSITY

Agile Development

Adaptive Software Development

- Originally proposed by Jim Highsmith
- ASD — distinguishing features
  - Mission-driven planning
  - Component-based focus
  - Uses “time-boxing”
  - Explicit consideration of risks
  - Emphasizes collaboration for requirements gathering
  - Emphasizes “learning” throughout the process

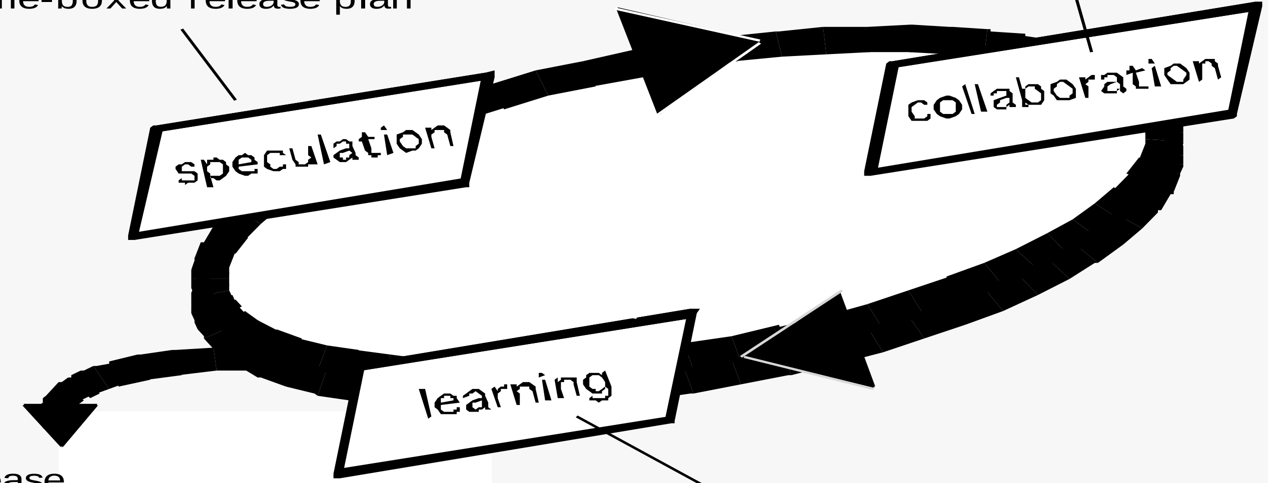
People
Innovation
Excellence

Agile Development

Adaptive Software Development

adaptive cycle planning
uses mission statement
project constraints
basic requirements
time-boxed release plan

Requirements gathering
JAD
mini-specs



Release
software increment
adjustments for subsequent cycles

components implemented/tested
focus groups for feedback
formal technical reviews
postmortems

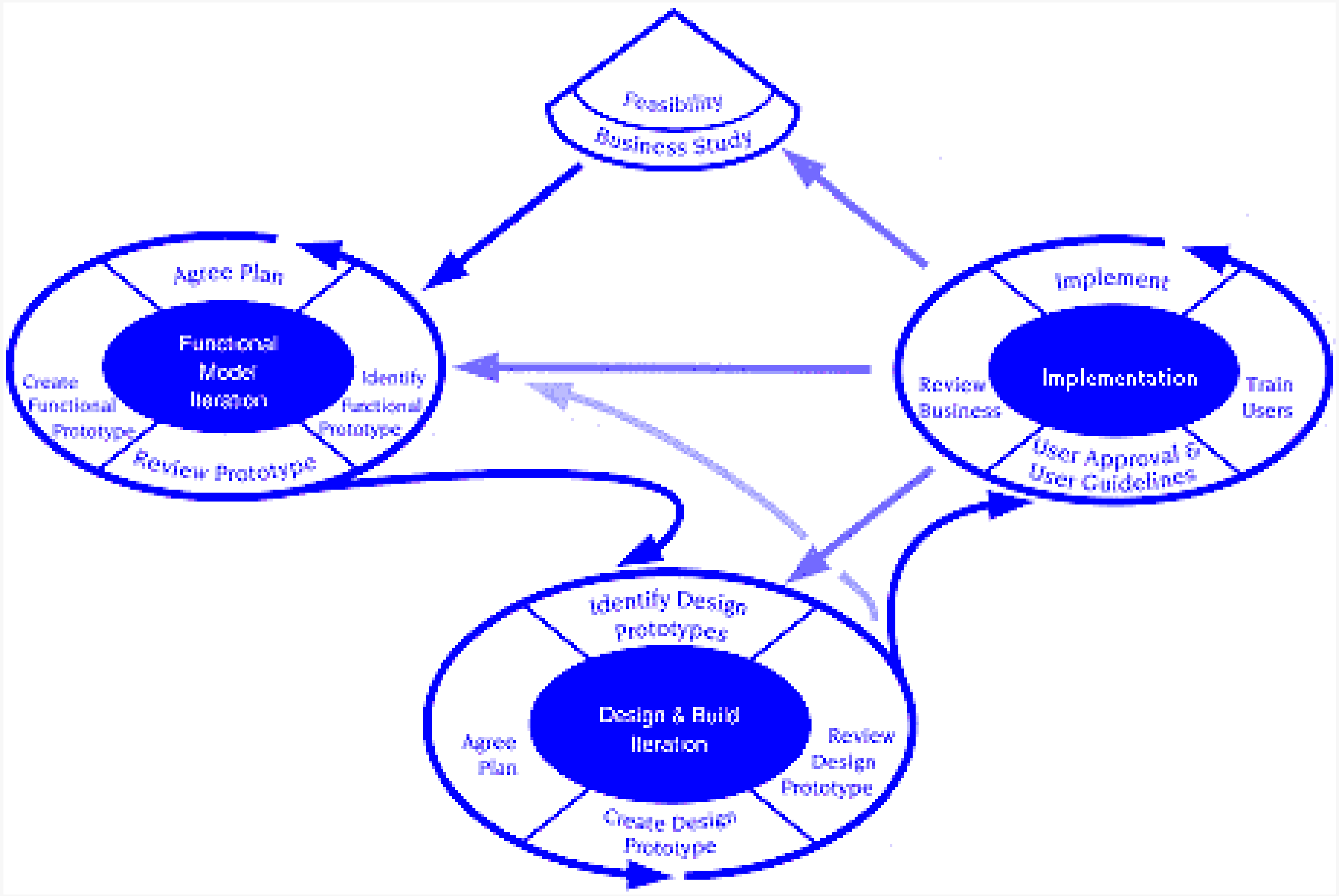
Agile Development

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

Agile Development

Dynamic Systems Development Method



Agile Development

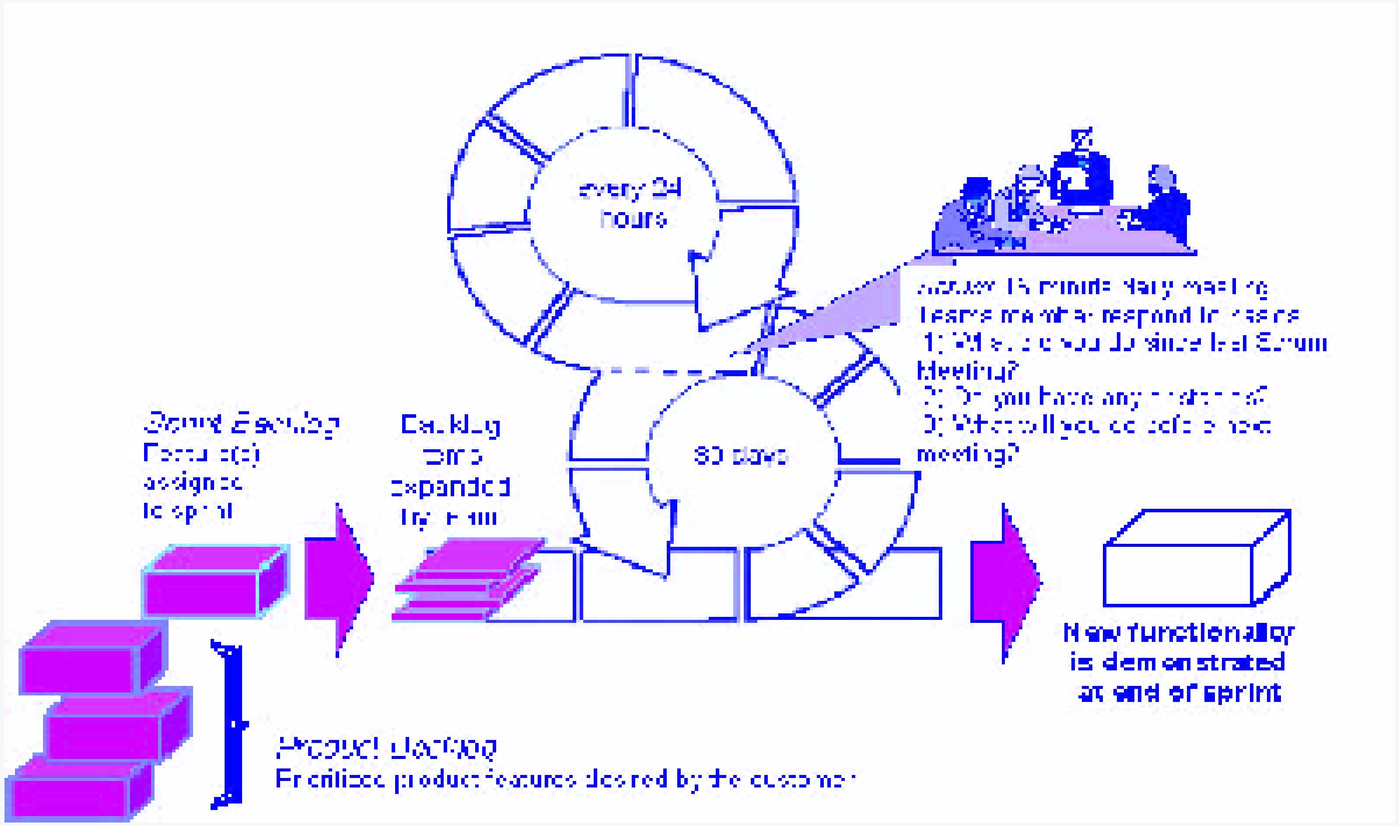
Scrum

- 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

People
Innovation
Excellence

Agile Development

Scrum


every 24 hours
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s)
assigned
to sprint
Backlog
items
expanded
by team
30 days
New functionality
is demonstrated
at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Scrum Process Flow (used with permission)

Agile Development

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

People
Innovation
Excellence

Agile Development

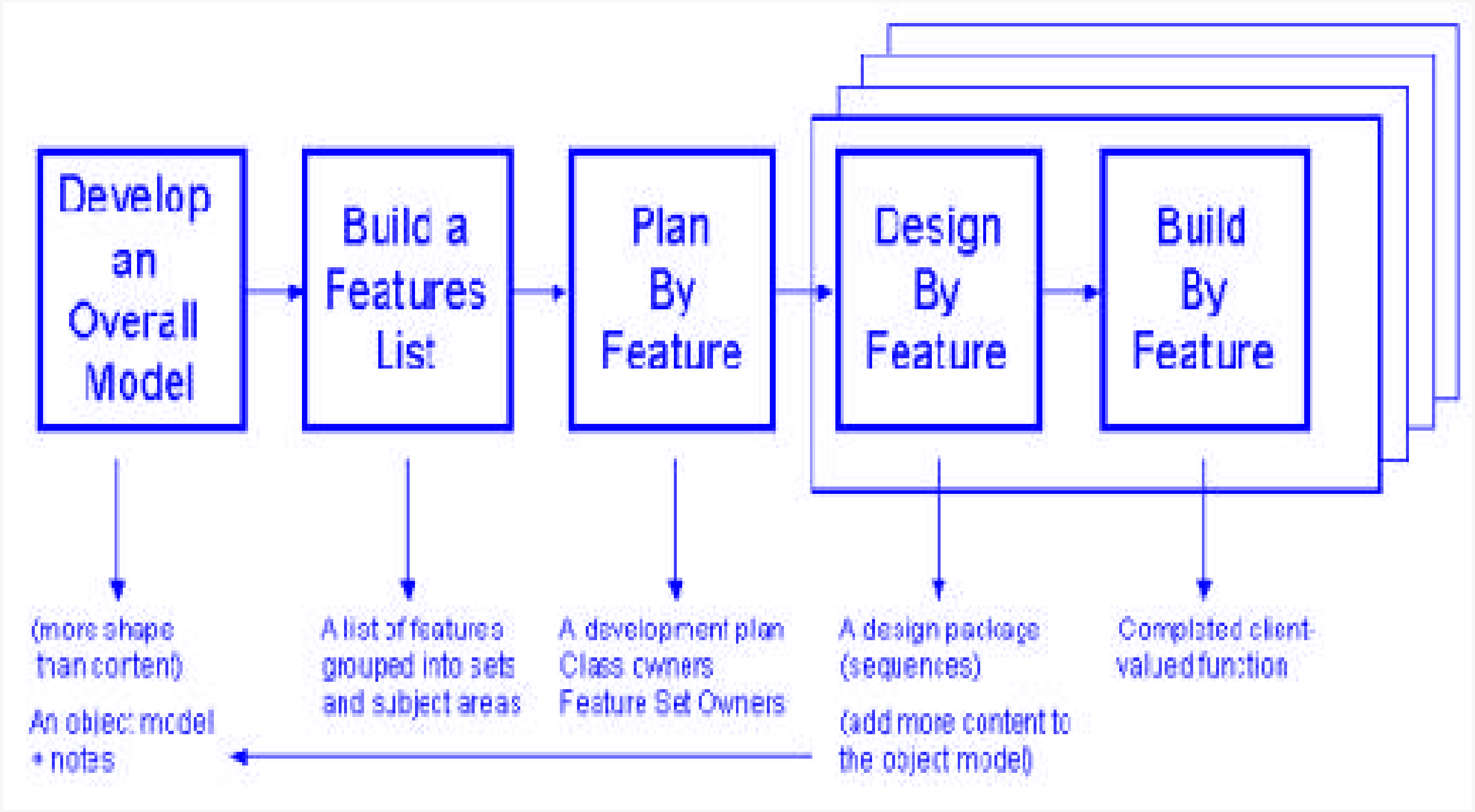
Feature Driven Development

- 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

People
Innovation
Excellence

Agile Development

Feature Driven Development


Develop an Overall Model
Build a Features List
Plan By Feature
Design By Feature
Build By Feature
(more shape than content)
An object model + notes
A list of features grouped into sets and subject areas
A development plan Class owners Feature Set Owners
A design package (sequences)
(add more content to the object model)
Completed client-valued function

Reprinted with permission of Peter Coad

People
Innovation
Excellence

BINUS UNIVERSITY

Agile 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

People
Innovation
Excellence

Human Aspects of Software Engineering

BINUS UNIVERSITY

People
Innovation
Excellence

Characteristics of Software Engineer

Erdogmus [erd09] identifies seven traits that are present when an individual software engineer exhibits "superprofesional" behavior.

An effective software engineer :

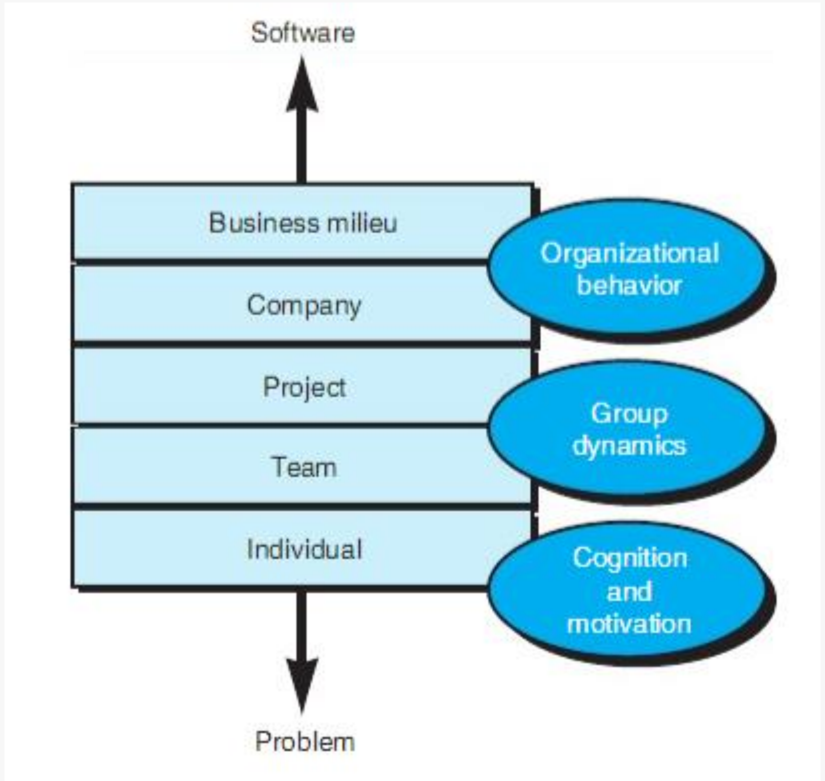
- has a sense of individual responsibility
- has an acute awareness
- is brutally honest
- exhibits resilience under prerssure
- has a heightened sense of fairness
- exhibits attention to detail
- is pragmatic

People
Innovation
Excellence

Human Aspects of Software Engineering

The Psychology of Software Engineering

In a seminal paper on the psychology of software engineering, Bill curtis and Dave Walz [Cur90] suggest a layered behavioral model for software development.



BINUS UNIVERSITY

Human Aspects of Software Engineering

The Software Team

- An effective team should foster a sense of trust
- Software engineers on the team should trust the skills and competence of their peers and their managers.
- The team should encourage a sense of improvement by periodically reflecting on its approach to software engineering and looking for ways to improve their work

People
Innovation
Excellence

Team Structure

Constantine [Con93] suggests four “organizational paradigms” for software engineering teams

1. **A closed paradigm;** a team along a traditional hierarchy of authority
2. **A random paradigm;** a team loosely and depends on individual initiative of the team members
3. **An open paradigm;** a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
4. **A synchronous paradigm;** relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication amonjg themselves

BINUS UNIVERSITY

References

People
Innovation
Excellence

- Pressman, R.S. (2015). ***Software Engineering : A Practioner's Approach. 8th ed***. McGraw-Hill Companies.Inc, Americas, New York. ISBN : 978 1 259 253157.
- Multimedia : Video Water Fall, V model, http://www.youtube.com/watch?v=KaPC0gsEQ68
- Software Engineering Incremental Model, http://www.youtube.com/watch?v=9cBkihYP1rY
- The Strengths and Weaknesses of Extreme Programming, **http://www.youtube.com/watch?v=LkhLZ7_KZ5w**
- Agile project management tutorial: What is agile project managemen, http://www.youtube.com/watch?v=MJR-EgHTA4E

Q & A

People
Innovation
Excellence