

# Software Tools Software Development Processes

## Part II - Lecture 2

# Today's Outline

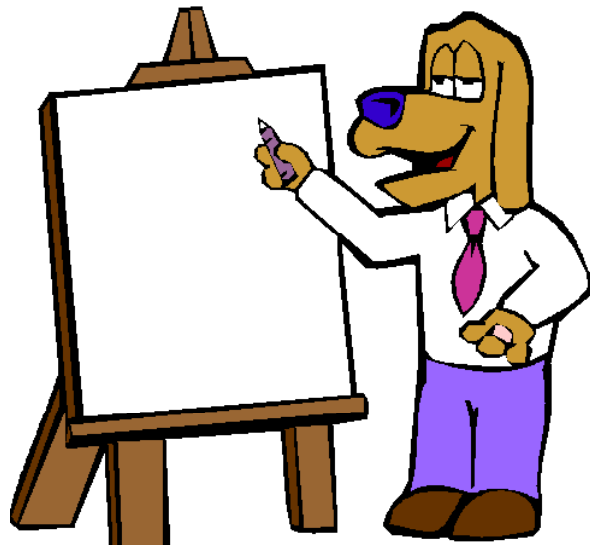
2009  
YEAR

COMPSCI 732

The University of Auckland | New Zealand

- Introduction to Software Development Processes
- eXtreme Programming (XP)
- Rational Unified Process (RUP)

# Software Development Processes



*He who fails to plan,  
plans to fail  
(Proverb)*

# Software Development Process

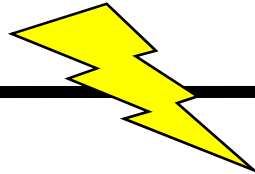
Generic plan for a software project

1. **What** has to be done? (-> tasks/activities/steps)
  2. **Why** do a task? (-> outcomes, produced artifacts)
  3. **When** should it be done? (-> schedule)
  4. **Who** does it? (-> people, roles, responsibilities)
  5. **How** should it be done? (-> methods, standards, tools)
- Many different processes exist
  - No single process suitable for every project (no "one size fits all")
  - Using a process can improve the quality of the product

# Adaptive vs. Predictive Processes



**Adaptive**



**Predictive**



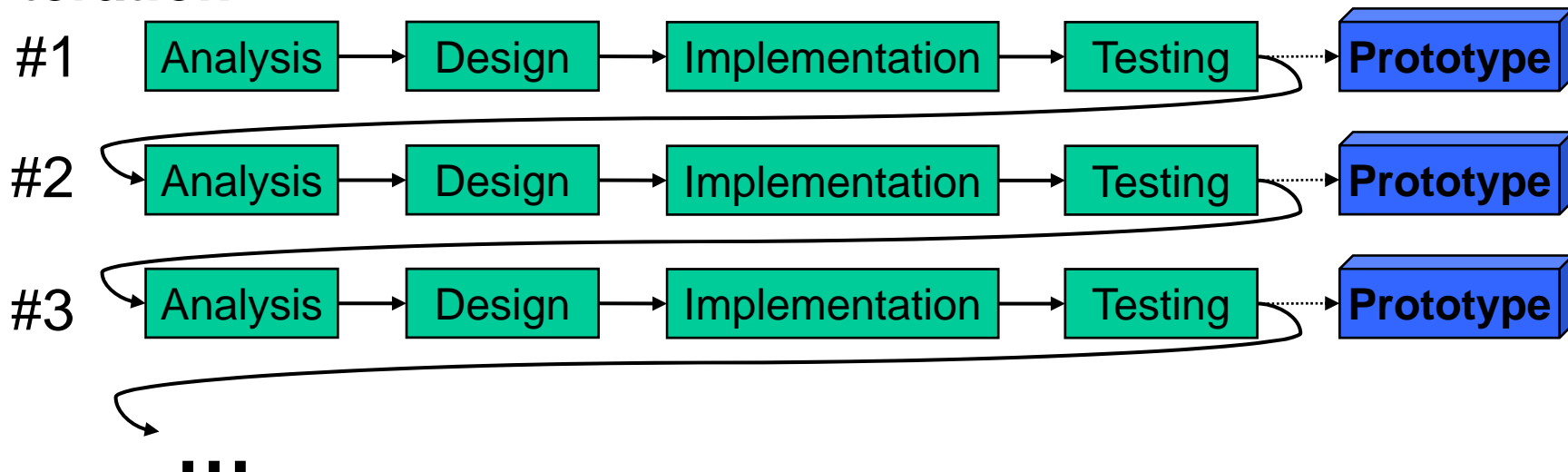
- Lightweight, 'agile'
- Control by feedback
- Many short iterations (weeks)
- Small scale (<10 developers)
- Face-to-face communication
- Code- & people-centric
- Egalitarian
- Problems:
  - Long-term results hardly predictable
  - Needs good project foundation
  - Cowboy-coding chaos
- E.g. XP

- Heavyweight, 'traditional'
- Control by planning
- Few long iterations (months)
- Large scale (>30 developers)
- Written documents
- Rule-centric
- Authoritarian
- Problems:
  - Inflexible with changing requirements
  - High integration and testing effort
  - 'Control freak' bureaucracy
- E.g. waterfall, RUP

# Agile Software Development

- Evolved in mid 1990s as part of a reaction against heavyweight methods
- Many short iterations (weeks), 'prototyping':

## Iteration



- Control by feedback: reevaluation & revision of project after each iteration

# eXtreme Programming (XP)

# XP Overview



*„Instead of cowboy coders we have software sheriffs; working together as a team, quick on the draw, armed with a few rules and practices that are light, concise, and effective.“*

(James D. Wells, [extremeprogramming.org](http://extremeprogramming.org))

- XP=eXtreme Programming:  
Nomen est omen, a code-centered approach
- **XP culture**: not just about getting work done
- Set of day-to-day **best practices** for developers and managers that encourage and embody certain values
- 5 values, 12 practices/rules



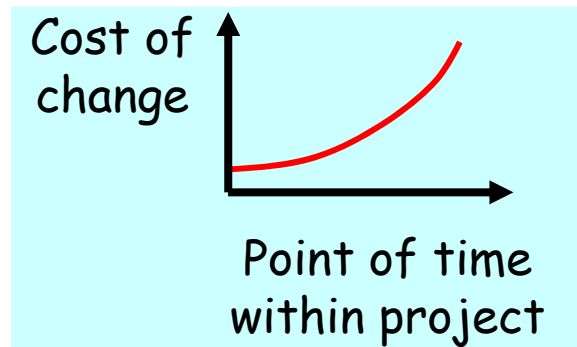
# The 5 XP Values

## 1. Communication

- Teamwork: consistent shared view of the system
- Open office environment: developers, managers, customers
- Verbal, informal, face-to-face conversation

## 2. Feedback

- Find required changes ASAP to avoid cost
- From the customer, through early prototypes & communication
- Testing, code review, team estimates



## 3. Simplicity

- Build the simplest thing that works for today
- No work that might become unnecessary tomorrow
- Simple design easier to communicate

## 4. Courage

- To change and to scrap, "embrace change"
- Better change now (cheaper)
- Never ever give up!

## 5. Respect your teammates and your work

# The 12 XP Practices

2009  
YEAR

COMPSCI 732

The University of Auckland | New Zealand

## Fine scale feedback

1. **Pair Programming**  
Programming in teams of two: driver and navigator
2. **Planning Game**: method for project planning with the customer
3. **Test Driven Development**
  - First write test cases, then program code
  - For each defect, introduce new test case
4. **Whole Team**: teamwork of customer, developer/manager

## Shared understanding

5. Use an agreed **Coding Standard**
6. **Collective Code Ownership**  
Everybody is responsible for and can change all code
7. **Simple Design**
8. **System Metaphor**  
Consistent, intuitive naming of program parts

# The 12 XP Practices

## Continuous process

### 9. Continuous Integration

- Work with latest version
- Integrate local changes ASAP

### 10. Refactoring

- Improve design whenever possible
- Remove clutter & unnecessary complexity

### 11. Small Releases

## Programmer welfare

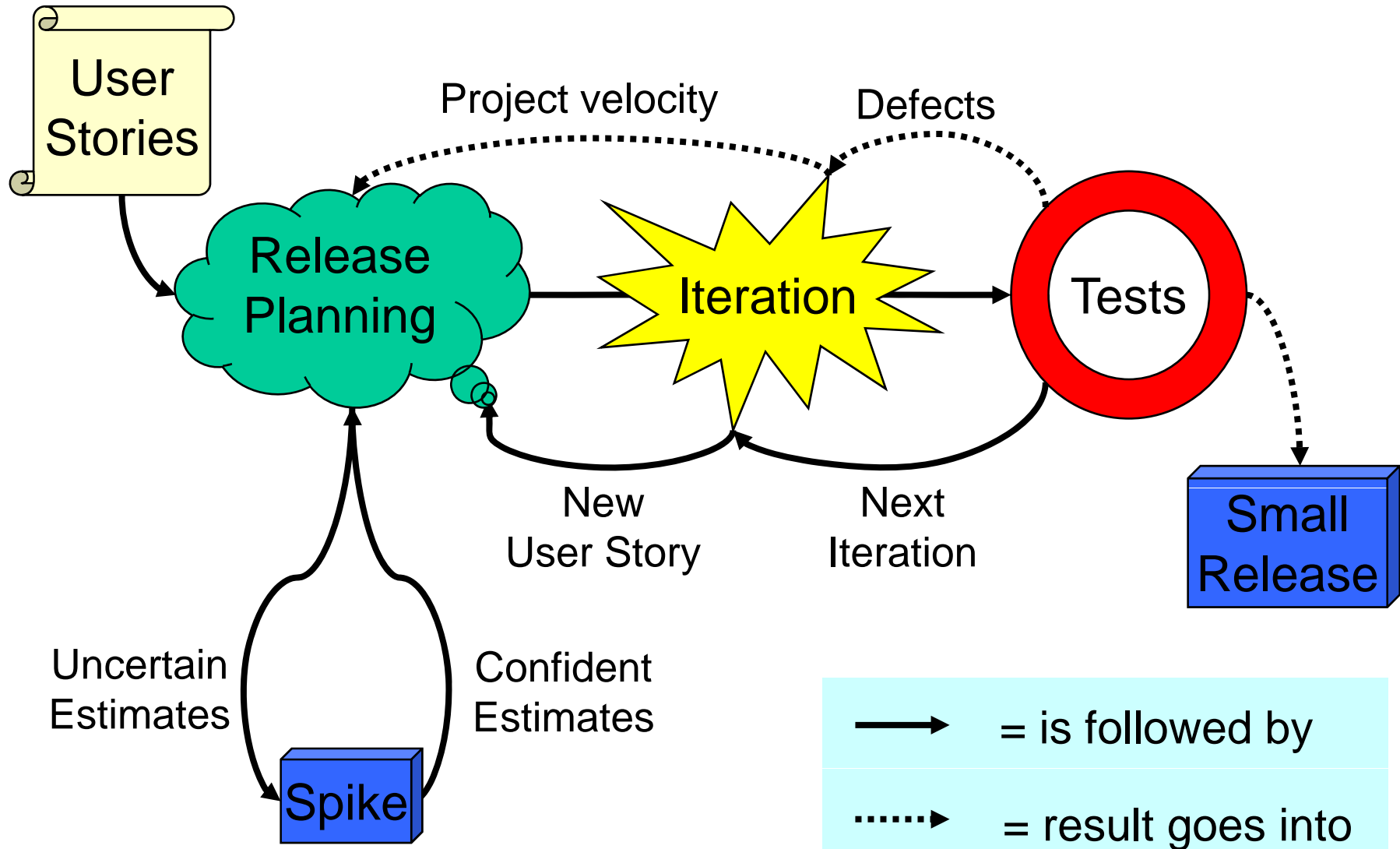
### 12. Sustainable Pace

No Overtime - change timing or scope instead

# Some XP Terminology

- **User story**
  - Things the system needs to do for the users
  - Written on a card in a few sentences
  - Should take 1-3 weeks to implement
- **Release**: running system that implements important user stories
- **Spike**
  - Small proof-of-concept prototype
  - Explores the feasibility of an implementation approach
- **Iteration**
  - Phase of implementation, 1-3 weeks long
  - Consists of tasks, each of which is 1-3 days long
- **Project velocity**: used to estimate progress
  - Either  $\#stories / time$  (time)
  - Or  $time / \#stories$  (scope)

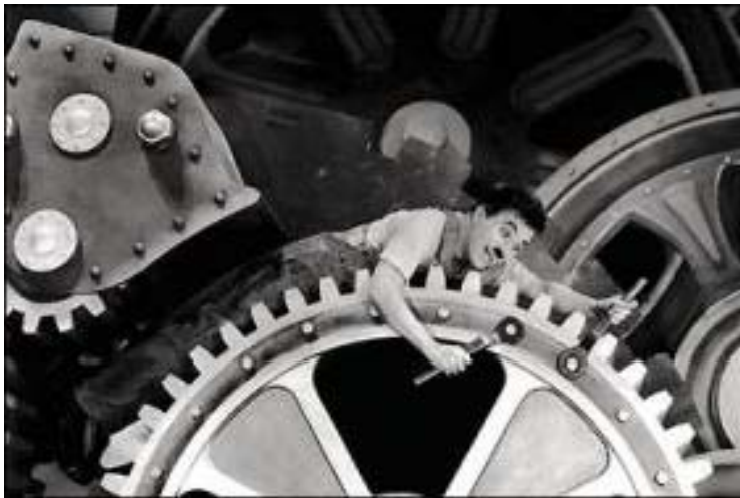
# XP Workflow Overview



# XP Criticism

- Relies on **on-site customer**
  - Single point-of-failure  
(-> source of stress, lack of technical expertise)
  - May not be representative for all users (-> user conflicts)
- **Unstable Requirements** because of informal change requests instead of formal change management (-> rework, scope creep)
- **Lack of documentation**, e.g. tests instead of requirements documents
- **Incremental design on-the-fly** (-> more redesign effort)
- **Pair-programming** required
- **Interdependency of practices** requires drastic organizational changes
- **Scalability? Distributed development?**

# The Rational Unified Process (RUP)



# RUP Overview

- Extensible, customizable **process framework**
- Created by the Rational Software Corporation in the 1980s and 1990s, which was sold to IBM in 2003
- Now software process product of IBM
- IBM sells RUP tools, e.g. Rational Method Composer for authoring, configuring and publishing processes
- **Business-driven** development
- Tied to UML
- Heavyweight, i.e. of considerable size, but recent changes influenced by lightweight, agile processes



# 6 RUP Best Practices: The RUP ABC

## **A**dapt the process

- right-size the process to project needs
- adapt process ceremony to lifecycle phase
- continuously improve the process
- balance project plans and associated estimates with the uncertainty of a project

## **B**alance competing stakeholder priorities

- understand and prioritize business and stakeholder needs
- center development activities around stakeholder needs
- balance asset reuse with stakeholder needs

## **C**ollaborate across teams

- motivate individuals on the team to perform at their best
- encourage cross-functional collaboration
- provide effective collaborative environments

# The RUP ABC Cont'd

**D**emonstrate value iteratively

- incremental value to enable early and continuous feedback
- adapt your plans
- embrace and manage change
- drive out key risks early

**E**levate the level of abstraction

- reusing existing assets
- leverage higher-level tools, frameworks, and languages
- focus on architecture

**F**ocus continuously on quality

- the entire team owns quality
- test early and continuously
- incrementally build test automation

# RUP Lifecycle

- 4 phases divided into a series of timeboxed *iterations*
- Each iteration results in an *increment* (release)
- *Disciplines* (like traditional phases) which happen with varying emphasis in every phase

## 1. Inception Phase

- Justification or business case
- Project scope, use cases, key requirements
- Candidate architectures
- Risks, preliminary project schedule, cost estimate

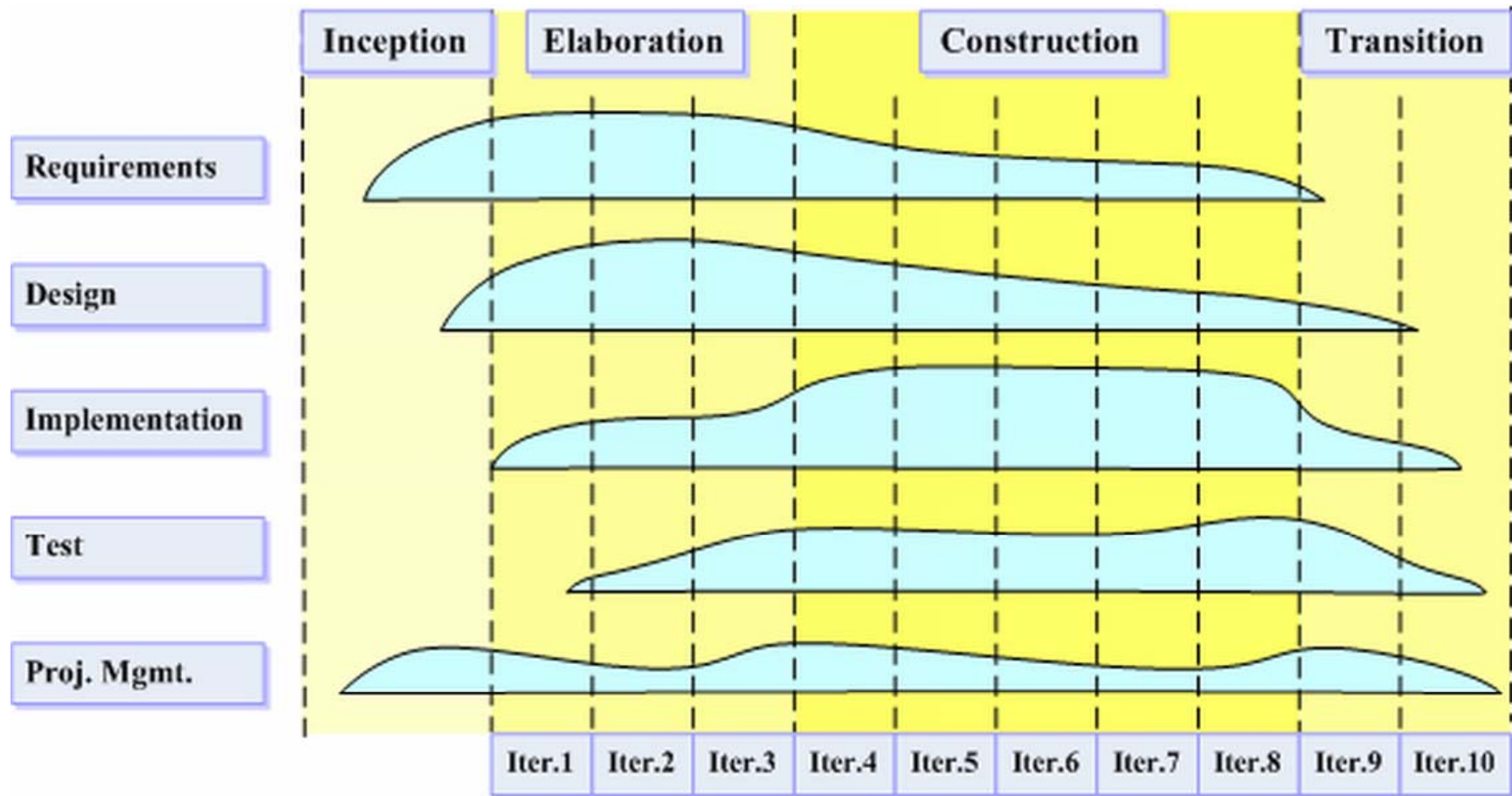
## 2. Elaboration Phase

- Requirements, risk factors
- System architecture (Executable Architecture Baseline)
- Construction plan (including cost and schedule estimates)

## 3. Construction Phase: building the rest of the system (longest)

## 4. Transition Phase: deployment, feedback, user training

# RUP Lifecycle



2009  
YEAR

COMPSCI 732

The University of Auckland | New Zealand



2006 Giles Lewis

# RUP Criticism

- "High ceremony methodology"
- Bureaucratic: process for everything
- Slow: must follow process to comply
- Excessive overhead: rationale, justification, documentation, reporting, meetings, permission
- Very customizable: can be everything and nothing

But:

- RUP can be used in traditional waterfall style or in agile manner
- Example: dX process
  - Fully compliant instance of RUP
  - Identical to XP

# Summary

# Summary

- **Adaptive vs. predictive Processes**
- **eXtreme Programming (XP)**
  - **Agile** process focused on programming as a team
  - **Short iterations**, as much feed back as possible
  - Best practices include collective code ownership, refactoring, pair programming
- **Rational Unified Process (RUP)**
  - **Heavyweight** process framework
  - Phases divided into iterations, several disciplines happening simultaneously
  - Best practices include risk & change management, use of tools, models & components

# Quiz

1. Describe three differences between adaptive and predictive processes.
2. Name five of the XP best practices.
3. What are the characteristics of the RUP lifecycle?