INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Fall Quarter, 2007
Michele Rousseau
Set 2

(Some slides adapted from Susan E. Sim)

Announcements

- **Labs 1,2 Due today**

Previous Class…

- **Brief Review of S/W Engineering**
- **Introduction to Tools & Methods**
- **Review Questions**
  - T or F – Software Engineering can be defined as the practice of programming a software product.
    - False
  - Why do we need software engineering?
    - Many reasons
      - To build larger systems
      - Reduce costs
      - Have some level of confidence in the quality of the system
  - What is a S/W Lifecycle Model?
    - An Abstract representation of the software process - that defines the process from inception through maintenance

More Review Questions

- What are the 3 elements that are necessary to create a S/W product?
  - People
  - Processes
  - Tools
- Why do we need tools?
  - Scaling problem
  - They support the process/people so that we can build bigger systems
- Why is there a gap between research and practice?
  - Leaning curve
  - Unclear payoffs
  - Focus tends to be more on what and not how

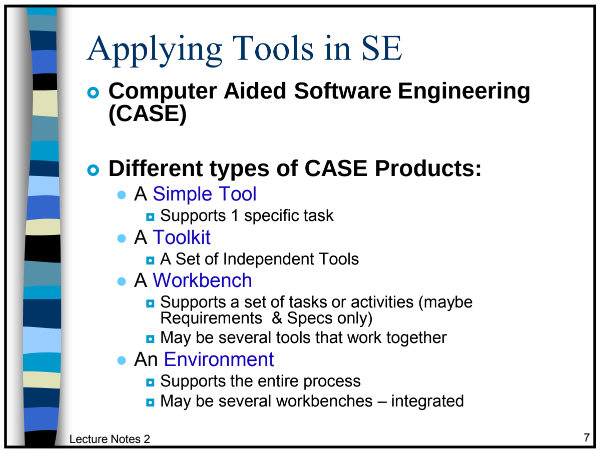
Today's Lecture

- **Software Tools**
- **Methods & Notations**
- **Process Modeling**
  - Agile Process
  - Extreme Programming

Notations, Tools & Methods

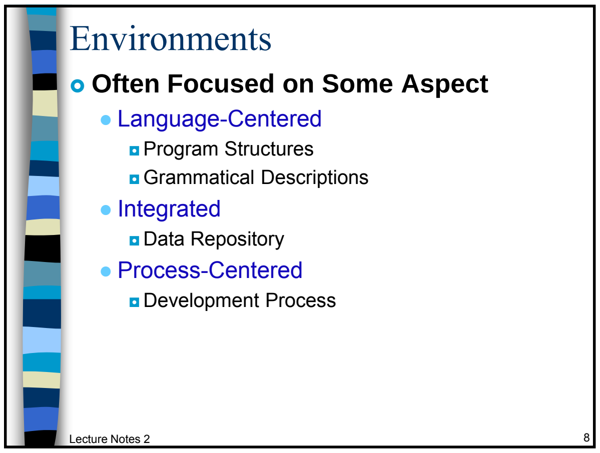
- ***Tools*:**
  - Machines, Executable Programs
- ***Methods*:**
  - Processes, Procedures
- ***Notations*:**
  - Languages Used by Tools and Methods

Remember the Guitar Example …
Tool: Guitar
Method: How I play (strum/pick/style)
Notation: Music



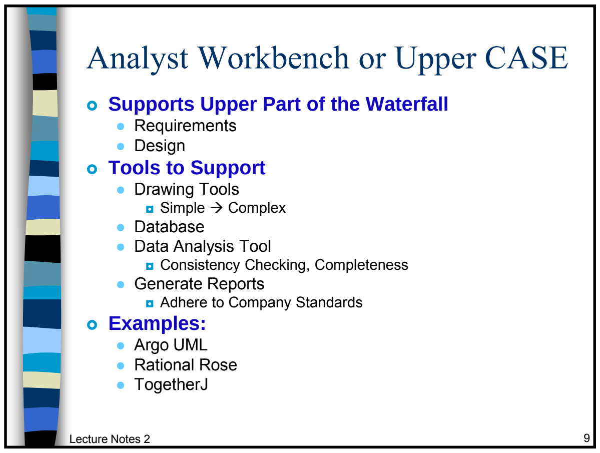
Applying Tools in SE

- **Computer Aided Software Engineering (CASE)**
- **Different types of CASE Products:**
  - A Simple Tool
    - Supports 1 specific task
  - A Toolkit
    - A Set of Independent Tools
  - A Workbench
    - Supports a set of tasks or activities (maybe Requirements & Specs only)
    - May be several tools that work together
  - An Environment
    - Supports the entire process
    - May be several workbenches – integrated



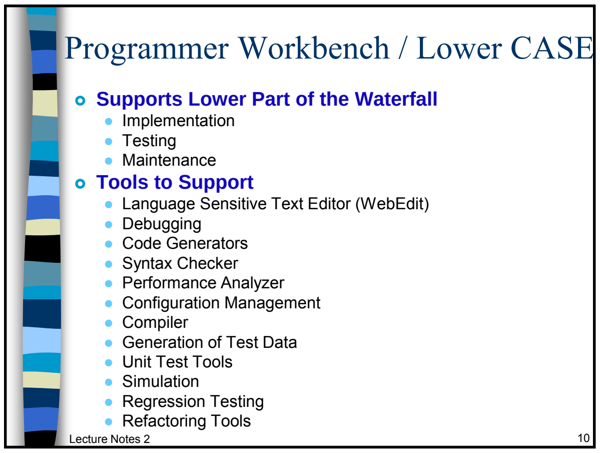
Environments

- **Often Focused on Some Aspect**
  - Language-Centered
    - Program Structures
    - Grammatical Descriptions
  - Integrated
    - Data Repository
  - Process-Centered
    - Development Process



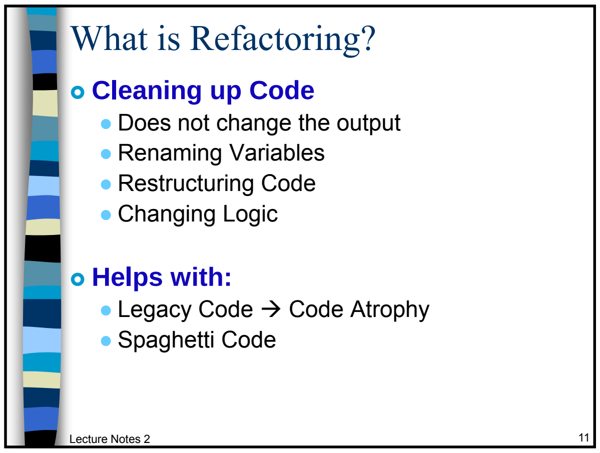
Analyst Workbench or Upper CASE

- **Supports Upper Part of the Waterfall**
  - Requirements
  - Design
- **Tools to Support**
  - Drawing Tools
    - Simple → Complex
  - Database
  - Data Analysis Tool
    - Consistency Checking, Completeness
  - Generate Reports
    - Adhere to Company Standards
- **Examples:**
  - Argo UML
  - Rational Rose
  - TogetherJ



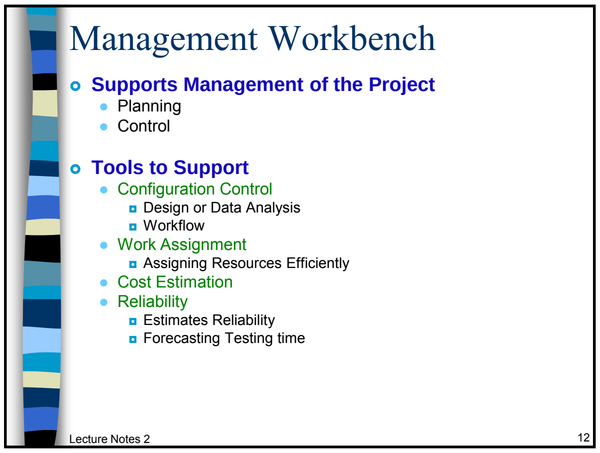
Programmer Workbench / Lower CASE

- **Supports Lower Part of the Waterfall**
  - Implementation
  - Testing
  - Maintenance
- **Tools to Support**
  - Language Sensitive Text Editor (WebEdit)
  - Debugging
  - Code Generators
  - Syntax Checker
  - Performance Analyzer
  - Configuration Management
  - Compiler
  - Generation of Test Data
  - Unit Test Tools
  - Simulation
  - Regression Testing
  - Refactoring Tools



What is Refactoring?

- **Cleaning up Code**
  - Does not change the output
  - Renaming Variables
  - Restructuring Code
  - Changing Logic
- **Helps with:**
  - Legacy Code → Code Atrophy
  - Spaghetti Code



Management Workbench

- **Supports Management of the Project**
  - Planning
  - Control
- **Tools to Support**
  - Configuration Control
    - Design or Data Analysis
    - Workflow
  - Work Assignment
    - Assigning Resources Efficiently
  - Cost Estimation
  - Reliability
    - Estimates Reliability
    - Forecasting Testing time

Take a break!

- **Stretch, Relax**
- **Get some water, Use the restroom**
- **Get to know your classmates…**
- **Etc…..**

When we return…

- **More on Software Tools**
  - Why we need them and what they are
- **Modeling**
  - What they are and how they apply to S/E

Before Break we discussed

- **Review Questions**
  - What are Tools, Methods and Notations?
    - Tools: Machines, Executable Programs
    - Methods: Processes, Procedures
    - Notations: Languages used by tools and Methods
  - What are the different types of CASE products?
    - Simple Tool → Supports 1 task
    - Toolkit → Set of Independent Tools
    - Workbench →Supports a set of tasks or activities
    - Environment → Supports the entire process

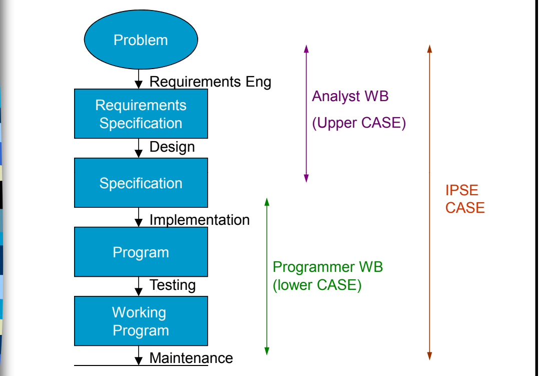
More Review Questions

- What is a Analyst Workbench? (AWB)
  - Supports the upper part of the “Waterfall”
- What is a Programmer Workbench? (PWD)
  - Supports the lower Part of the “Waterfall”
- What is a Management Workbench? (MWB)
  - Supports the Management of the Project
- What is the difference between a WB and an Environment?
  - WB supports part of the process whereas an environment supports the entire process
- What is Refactoring?
  - Cleaning up the code – without changed the output

Integrated Project Support Environments (IPSE)

- **Supports the Entire Project**
  - Analyst Workbench
  - Programmer Workbench
  - Management Workbench
- **Tight Integration vs. Loose Integration**

Integrated Environments / Workbenches

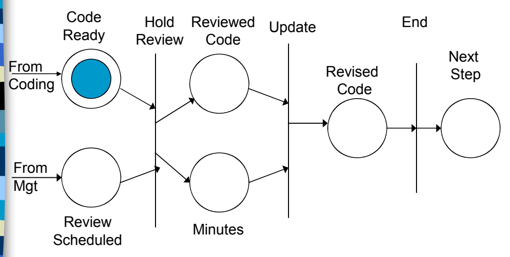


Adapted from Van Vliet

Process-Centered Environment (PSEE)

- **Supports the Development Process**
- **Closely Tied to Process Modeling**
  - Petri-Nets
  - State Transition Diagrams
  - Etc…
- **Tends to support Back-End (Imp. & Testing)**
  - Easier to Formalize

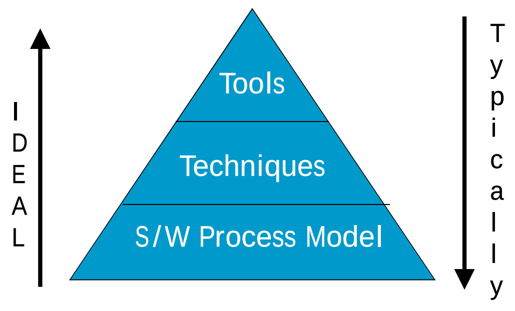
Petri-Net View of PSEE



Some of the Tools/Environments We Will Use

- **Eclipse JDT**
- **JUnit**
- **Eclipse Plugins**
- **Argo UML (Or Rational Rose)**
- **Etc…**

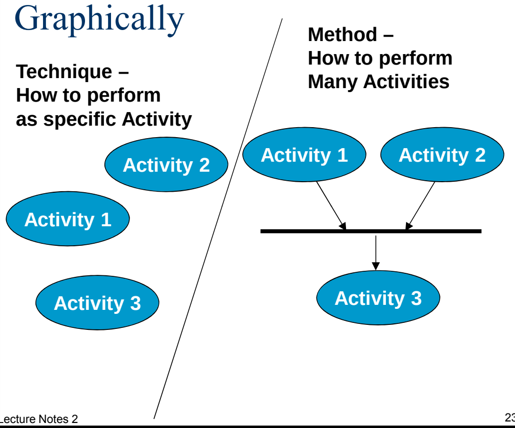
Remember -- Selecting a Tool?



Methods

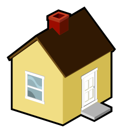
- A ***Method*** is a technical *prescription for how to perform a collection of activities*, focusing on integration of techniques and *guidance* on their use.
  - Prescribe → to lay down a rule
- A ***Technique*** is a *prescription of how to perform a particular activity*
  - May include rules on how to describe a product of that activity in a particular notation
  - Smaller than a Method
  - Example: Unit Testing

Graphically



Tools vs. Methods

- **Construction**
  - Tools
    - Hammer
    - Saw
    - Measuring Tape
  - Methods
    - Rules for Construction



Tools vs. Methods – Take 2

- **I give you a camera**

- **I teach you how to take a picture:**
  - Auto-focus
  - Push the Button
- **I teach you how to shoot a very nice picture**
  - Lighting
  - Aperture
  - Shutter Speed
  - Composition

Method vs. Methodology

- A ***method*** is a description of how we do something
- A ***methodology*** is the study of methods
- Methodology (from Wikipedia)

  The common idea here is the collection, the comparative study, and the critique of the individual methods that are used in the given discipline or field of inquiry

Notations

- A ***notation*** is a representation scheme (or language)
- A ***process model*** is an abstract description of *how to conduct a collection of activities*, focusing on resource usage and **dependencies** between activities
  - **Often expressed using a notation**

Notations, Tools & Methods

- **Tools:**
  - Machines, Executable Programs
- **Methods:**
  - Processes, Procedures
- **Notations:**
  - Languages Used by Tools and Methods

Remember the Guitar Example ...
Tool: Guitar
Method: How I play (strum/pick/style)
Notation: Music

Modeling

Modeling

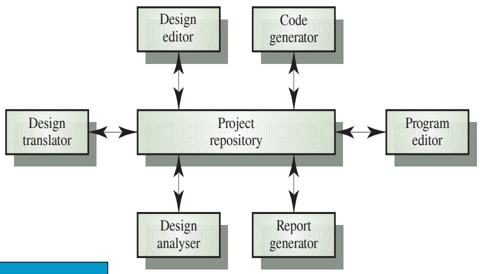
- A ***model*** is an abstract representation of a specification
- Defined by a consistent set of rules
  - Dictate the meaning of the components

  … and interactions
- Some Basic Principles
  - Models are used for breaking down concepts
    - Requirements or Design (Unified Modeling Language)
  - Used for *communicating*
  - Choice of the Model influences the Product
    - Object Models
    - Data Repository Models
    - Pipe and Filter
    - Etc..

For Example

- **Different Modeling Styles Can Influence the Product**
  - Styles restrict the way in which components can be connected
  - Prescribe patterns of interaction
  - Promote fundamental principles
    - Rigor, separation of concerns, anticipation of change, generality, incrementality
    - Low coupling
    - High cohesion
- **Architectural styles are based on success stories**
  - Almost all compilers are build as “pipe-and-filter”
  - Almost all network protocols are build as “layers”

Model Case Toolset

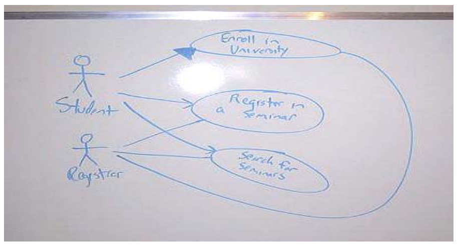


Data Repository

Modeling (2)

- **One Model → One Viewpoint**
  - Specification View vs. Design View
  - Runtime View vs. Compile-Time View
  - Static View vs. Dynamic View
- **Model should be realistic**
- **Model is usually incomplete**
  - Abstraction – doesn’t include details
- **Other Disciplines use Models too…**
  - Architects – Buildings
  - Circuit Board Designers

Another Example



Models Can Be Informal or Formal

Process Modeling

- A ***process model*** is an abstract description of how to conduct a collection of activities, focusing on resource usage and **dependencies** between activities
  - **Often expressed using a notation**

Remember: a ***notation*** is a representation scheme (or language))

Software Process (revisited)

General Software Process Activities

Phase	Purpose	Deliverables
User Requirements	Problem Def	User Req. Spec. Acceptance Test Plan
S/W Requirements	Problem Analysis	S/w Req. Spec. Support Service Brief System Test Plan
Architectural Design	High Level Solution	Architectural Design Support Serv. Design Integration Test Plan
Production	Implementation & Testing	Detailed Design Tested Software Est. Sup. Serv.
Transfer	Installation	Installed Software
Maint. & Support	S/w Operations & Support	Maintained & Supported S/W

We Discussed Traditional S/W Process Models

- Waterfall
- Spiral
- Incremental
- …etc

Criticisms with Traditional Process Models

- Generally **don't handle change** well
- **Implementation is delayed** until uncertainties are completely resolved
- **Too mechanistic** to be used in detail

The Agile Method

- ***Agile*** – "having a quick resourceful and adaptable character" – Merriam-Webster
- Works best for **smaller teams** and **projects**
- **Quick Product Releases**

Four Central Values of Agile Methods

1. Focus on the human role of s/w dev
2. Continuously turn out tested working software
3. Foster the relationship with the client (over nitpicking the contract)
4. The Development Group

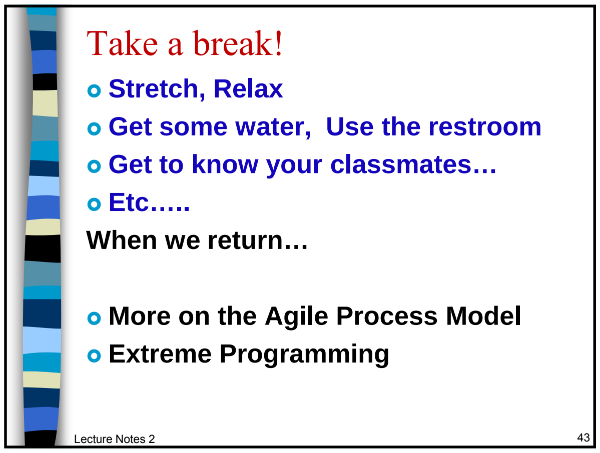
What makes a Method Agile?

- **Incremental**
  - Small software releases with rapid cycles
- **Cooperative**
  - Customers and developers working together constantly - close communication
- **Straightforward**
  - Method is easy to learn, modify and well documented
- **Adaptive**
  - Able to make last moment changes

How is Agile Different

- "What is new about agile methods is not the practices they use but their **recognition of people** as the primary drivers of project success, coupled with an **intense focus on effectiveness and maneuverability**. This yields a new combination of values and principles that define an *agile* world view"

Highsmith anc Cockburn (2001, p 122)

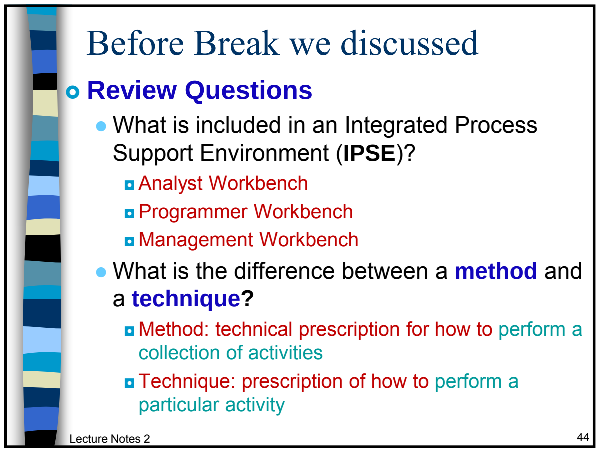


Take a break!

- **Stretch, Relax**
- **Get some water, Use the restroom**
- **Get to know your classmates…**
- **Etc…..**

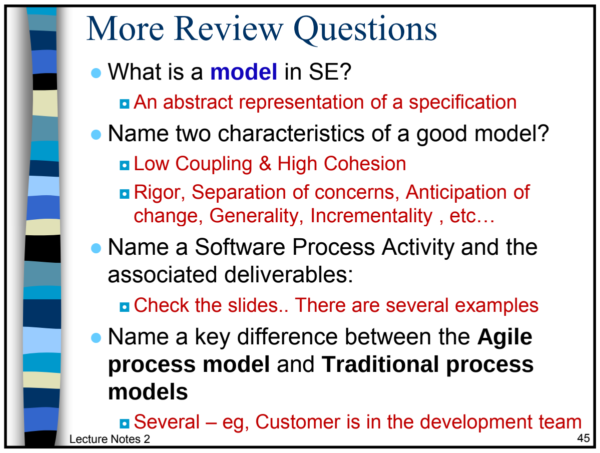
When we return…

- **More on the Agile Process Model**
- **Extreme Programming**



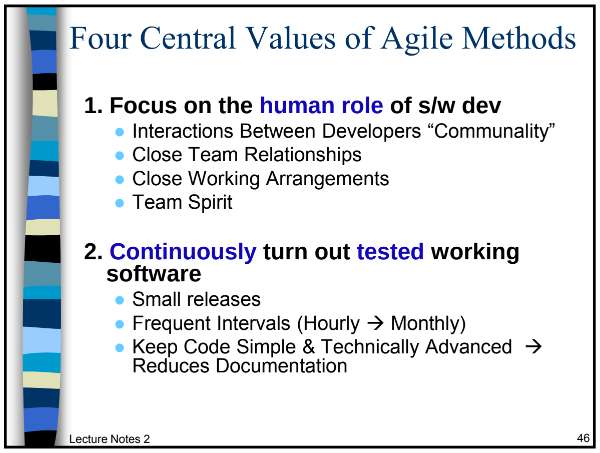
Before Break we discussed

- **Review Questions**
  - What is included in an Integrated Process Support Environment (**IPSE**)?
    - Analyst Workbench
    - Programmer Workbench
    - Management Workbench
  - What is the difference between a **method** and a **technique?**
    - Method: technical prescription for how to perform a collection of activities
    - Technique: prescription of how to perform a particular activity



More Review Questions

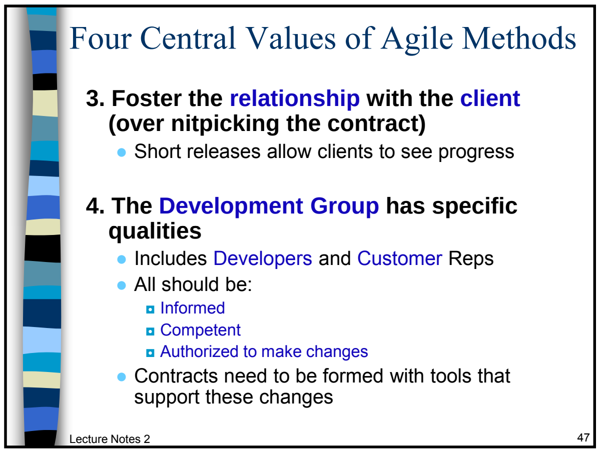
- What is a **model** in SE?
  - An abstract representation of a specification
- Name two characteristics of a good model?
  - Low Coupling & High Cohesion
  - Rigor, Separation of concerns, Anticipation of change, Generality, Incrementality , etc…
- Name a Software Process Activity and the associated deliverables:
  - Check the slides.. There are several examples
- Name a key difference between the **Agile process model** and **Traditional process models**
  - Several – eg, Customer is in the development team



Four Central Values of Agile Methods

1. Focus on the human role of s/w dev
- Interactions Between Developers "Communality"
- Close Team Relationships
- Close Working Arrangements
- Team Spirit

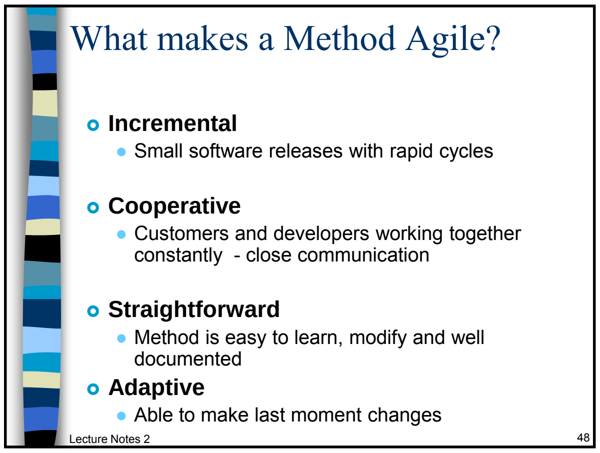
2. Continuously turn out tested working software
- Small releases
- Frequent Intervals (Hourly → Monthly)
- Keep Code Simple & Technically Advanced → Reduces Documentation



Four Central Values of Agile Methods

3. Foster the relationship with the client (over nitpicking the contract)
- Short releases allow clients to see progress

4. The Development Group has specific qualities
- Includes Developers and Customer Reps
- All should be:
  - Informed
  - Competent
  - Authorized to make changes
- Contracts need to be formed with tools that support these changes



What makes a Method Agile?

- **Incremental**
  - Small software releases with rapid cycles
- **Cooperative**
  - Customers and developers working together constantly - close communication
- **Straightforward**
  - Method is easy to learn, modify and well documented
- **Adaptive**
  - Able to make last moment changes

How is Agile Different

- **“What is new about agile methods is not the practices they use but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability. This yields a new combination of values and principles that define an *agile* world view”**

Highsmith anc Cockburn (2001, p 122)

Agile vs. Traditional Plan-driven

Home-Ground Area	Agile Methods	Plan-driven Methods
Developers	Agile, knowledgeable, collocated, & collaborative	Plan-Oriented, adequate skills, access to external knowledge
Customers	Dedicated, knowledgeable, *collocated*, collaborative, representative, & empowered	Access to knowledgeable, collaborative, representative, and empowered customers

Agile vs. Traditional Plan-driven

Home-Ground Area	Agile Methods	Plan-driven Methods
Requirements	Largely emergent; rapid change	Knowable early; largely stable
Architecture	Designed for current requirements	Designed for current and *foreseeable* requirements
Refactoring	Inexpensive	Expensive
Size	Smaller teams and Products	Larger Teams and Products
Primary Objective	Rapid Value	High Assurance

Examples of Agile Methods

- **XP → Extreme Programming**
- **Scrum →**
  - ”Getting out-of play ball back into the game”
- **FDD → Feature Driven Development**
- **RUP → Rational Unified Process**

Extreme Programming (XP)

- **Invented by Kent Beck in 1996**
  - “Seat of the pants” fix to Chrysler project
  - To fix problems caused by long development cycles of traditional process models
- **Beck Published in 1999**
  “Extreme Programming Explained: Embrace Change”
  - Current hot topic in S/W Process
  - Loved and Hated
  - Tries to associate s/w process with eXtreme sports
- **Idea: Take a good programming practice and push it to the extreme**
  - Eg. Testing
  - Testing is good so… do it all the time

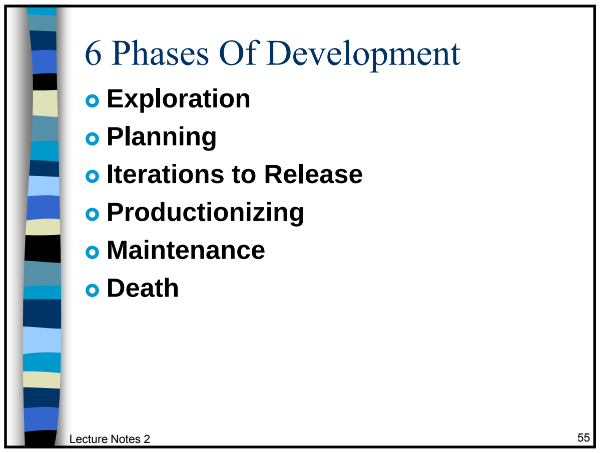
Premise of XP

- **The Four Values**

Communication | **Simplicity** | **Feedback**

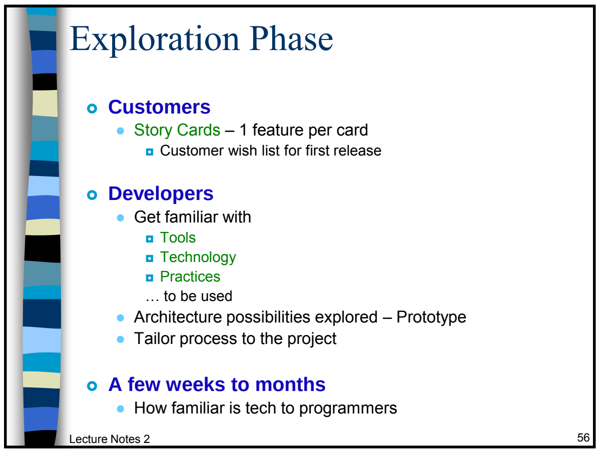
Courage

Hmmm.. But aren’t these standard “Best Practices”? What’s new here?



6 Phases Of Development

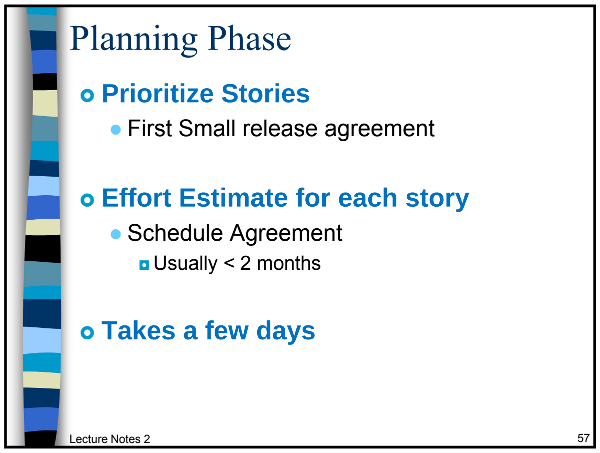
- **Exploration**
- **Planning**
- **Iterations to Release**
- **Productionizing**
- **Maintenance**
- **Death**



Exploration Phase

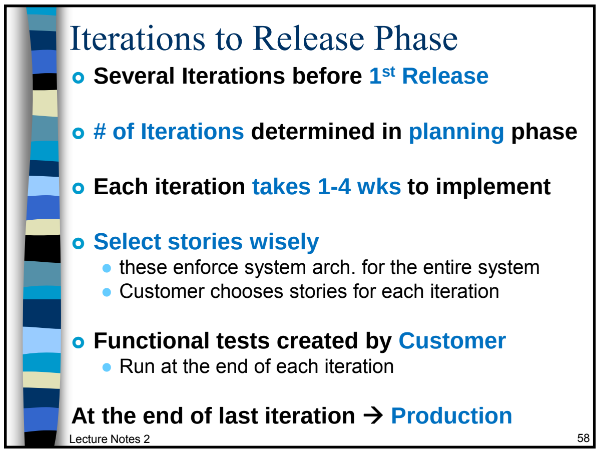
- **Customers**
  - Story Cards – 1 feature per card
    - Customer wish list for first release
- **Developers**
  - Get familiar with
    - Tools
    - Technology
    - Practices

    … to be used
  - Architecture possibilities explored – Prototype
  - Tailor process to the project
- **A few weeks to months**
  - How familiar is tech to programmers



Planning Phase

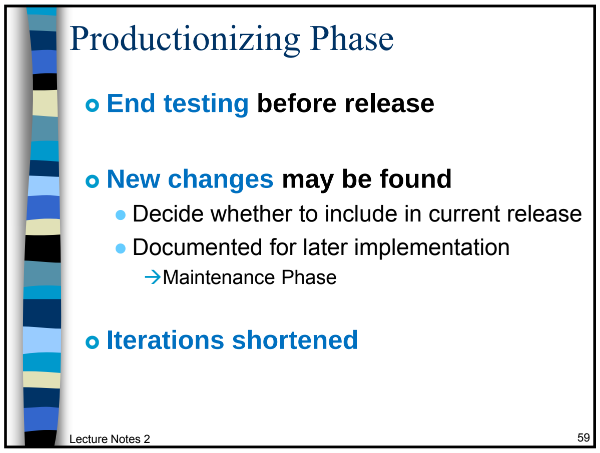
- **Prioritize Stories**
  - First Small release agreement
- **Effort Estimate for each story**
  - Schedule Agreement
    - Usually < 2 months
- **Takes a few days**



Iterations to Release Phase

- **Several Iterations before 1st Release**
- **# of Iterations determined in planning phase**
- **Each iteration takes 1-4 wks to implement**
- **Select stories wisely**
  - these enforce system arch. for the entire system
  - Customer chooses stories for each iteration
- **Functional tests created by Customer**
  - Run at the end of each iteration

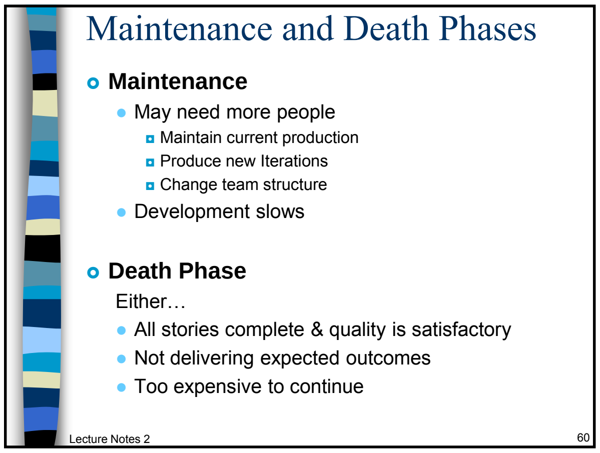
At the end of last iteration → Production



Productionizing Phase

- **End testing before release**
- **New changes may be found**
  - Decide whether to include in current release
  - Documented for later implementation

    →Maintenance Phase
- **Iterations shortened**



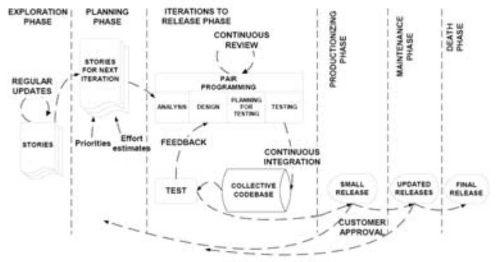
Maintenance and Death Phases

- **Maintenance**
  - May need more people
    - Maintain current production
    - Produce new Iterations
    - Change team structure
  - Development slows
- **Death Phase**

  Either…
  - All stories complete & quality is satisfactory
  - Not delivering expected outcomes
  - Too expensive to continue

XP Lifecycle Model

The life cycle of XP consists of five phases: Exploration, Planning, Iterations to Release, Productionizing, Maintenance and Death



Life cycle of the XP process.