

CS 497
Software Engineering I

Fall Semester 2005
Chris Lüer

1. What is this Class about?

- You know:
 - How to code
 - on your own
 - small programs
 - At least one programming language
 - Algorithms and data structures
- You will learn in this class:
 - How to develop software
 - that is large and complex
 - in a team
 - Tools and methods used in industry
 - “Software Engineering”
 - UML, Design Patterns, processes

© 2004. 497-1-2

Software Engineering

- Practical problems of software development
 - especially in large, complex projects
- Related to management and psychology
- Focus on activities other than implementation
- Subareas:
 - Requirements Engineering
 - Software Architecture, Design
 - Quality Assurance, Testing
 - Configuration Management
 - Formal Methods
 - Software Engineering Environments
 - Human-Computer Interaction
 - Project Management

© 2004. 497-1-3

Good Software

- Software Engineering is about making good software
- Maintainability
 - changing user needs
 - new technologies
- Dependability
 - software in hospitals, airplanes
 - year 2000
- Efficiency
 - distributed systems
- Usability
 - graphical user interfaces
 - easy to learn for new users

© 2004. 497-1-4

Challenges of Software

Engineering

- Legacy software
 - old software
 - maintaining it, interfacing with it
- Heterogeneity
 - Internet
 - wireless networks
 - small devices: PDAs, embedded computers
- Timeliness and Cost
 - getting what you need right away
 - and cheap, too!

© 2004. 497-1-5

What is this Class about? –

Methods

- Processes
- Requirements Capture
- Analysis
- Design
- Testing

© 2004. 497-1-6

What is this class about? —

Notations

- UML
 - Class diagrams
 - Sequence diagrams
 - Collaboration diagrams
 - Use case diagrams
 - State diagrams
 - Activity diagrams
- Others
 - Data flow diagrams
 - ER diagrams

© 2004. 497-1-7

What is this class about? —

Tools

- Rational Rose
 - UML
 - Commercial
 - Reverse engineering
- Argo/UML
 - UML
 - Open source
- CVS
- Development Environments
- 498:
 - testing tools
 - scripting languages

© 2004. 497-1-8

What is this class about? —

Project

- Customer
 - gives requirements
 - agrees on analysis document at the end of 497
 - agrees on implementation and manual at the end of 498
- Team work
 - coordination
 - communication
- Management
 - time planning
 - decision making

© 2004. 497-1-9

Prerequisites

- CS 336/436 Databases (or corequisite)
- CS 232 Data Structures

© 2004. 497-1-10

Related Classes

- Good if you took them, good if you are taking them now
- CS 337 Internet Programming
- CS 345 Graphical User Interfaces
- CS 397 Web Technology
- CS 438 Computer Graphics

© 2004. 497-1-11

Overview

- Requirements Capture
 - Scenarios
 - Mockups
 - Use Cases
- Process Models
 - Unified Process
 - Extreme Programming
 - Open Source Programming
- Project Management
 - Metrics
 - Risk Management
 - Quality Management
 - Scheduling
- Analysis and Specification
 - Object and Class Diagrams
 - Interaction Diagrams
 - State Diagrams
 - Assertions
- Design
 - Software Architecture
 - Design Patterns
- In 498:
 - More on design
 - Quality assurance / testing

© 2004. 497-1-12

1.2 Administrative Stuff

- Instructor
 - Chris Lürer (clueer@bsu.edu)
 - Office hour: Tu, Th 1:30-3:30
- GA
 - Jerry Jeffers
- Check the Web page frequently!

© 2004. 497-1-13

Textbooks

- Pressman
 - broad
 - background
 - all of software engineering
- Fowler
 - UML
 - Technical, in-depth view
- Handouts
 - essays
 - project management

© 2004. 497-1-14

Be Involved

- Class Web site
- Student Web sites
 - Team Web sites
- Mailing list: CS497-L@bsu.edu

- Be active
 - Ask questions
 - Participate in class discussions
 - Cooperate with team members
 - Go to office hours

© 2004. 497-1-15

Cheating

- Letter to Dean of Students
- Course grade lowered, possibly to F

- No team work on homeworks

- Things that are copied from books or Web pages need to be quoted and the source must be given

© 2004. 497-1-16

Grading

- Final Exam 25%
- Midterm Exam 10%
- Project 40%
- Homeworks 20%
- Quizzes 5%

- Peer evaluations for project

© 2004. 497-1-17

Who Are You?



© 2004. 497-1-18

1.3 Notations, Tools, and Methods

Things Software Engineers Use

Tools

Methods

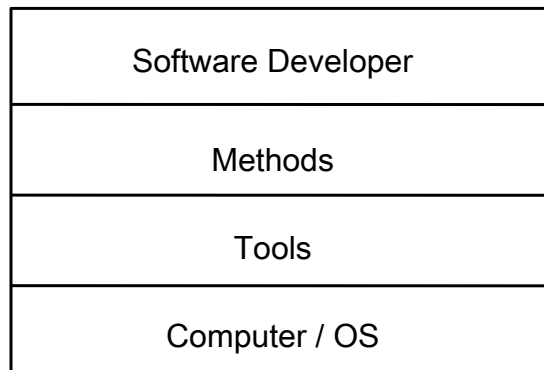
Notations

- Tools: machines, executable programs
- Methods: things people do
- Notations: languages used by tools and methods



© 2004. 497-1-19

Tools and Methods



On top of: uses

© 2004. 497-1-20

Tools

- For example:
 - IDE
 - Compiler
 - Debugger
 - Diagram tool
- Properties:
 - GUI / command-line
 - Which phases of the process are supported?
 - Single-user / multi-user
 - Dependent on specific methods or notations?
 - For example: Programming language
 - Integrated / non-integrated
 - Free / expensive
 - Open-source / closed-source

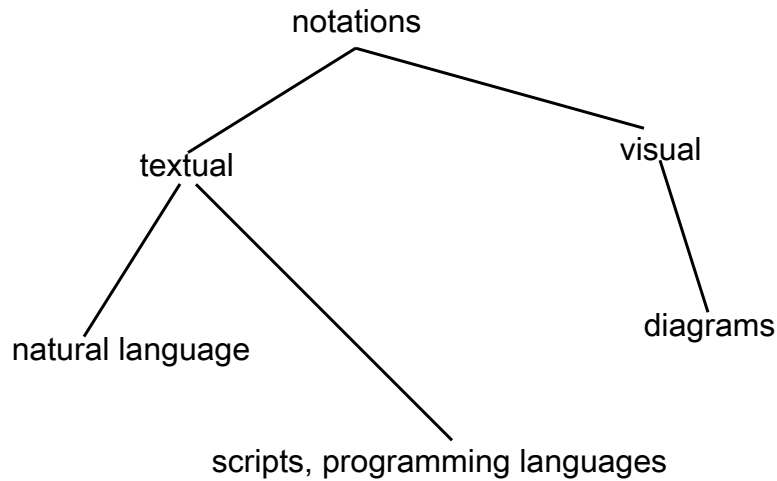
© 2004. 497-1-21

Notations

- For example:
 - Programming languages
 - Design notations
 - Natural language
- Properties:
 - Syntax: formal / informal
 - Semantics: formal / informal
 - For which process phases?
 - Are there tools and methods to support it?
 - Visual / textual
 - Usability
 - Expressiveness

© 2004. 497-1-22

Kinds of Notations



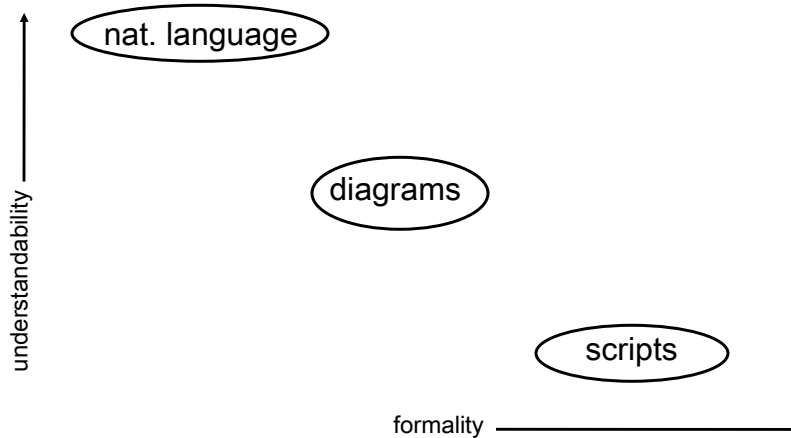
© 2004. 497-1-23

What are Notations Good for?

- Purpose: Communication
 - between humans
 - between humans and tools
- Notations are used to create models
 - A model is a simplified representation of a thing
- Natural language
 - Pro: understandable, maintainable
 - Contra: verbose, imprecise, not machine-readable
- Scripts and programming languages
 - Pro: formal and precise, need little space
 - Con: require learning
- Diagrams
 - Pro: understandable, show big picture, can be vague
 - Con: require GUI, take up space

© 2004. 497-1-24

Notations: Trade-Offs



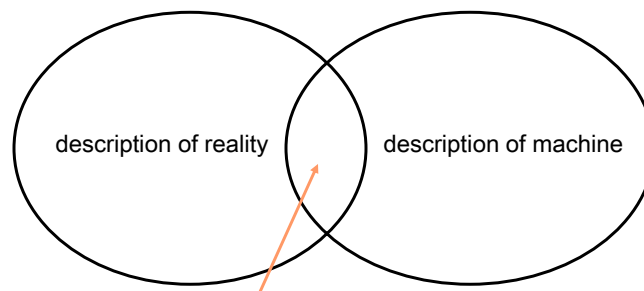
© 2004. 497-1-25

Principles of Modeling

- Models are for communication
 - for example: to communicate requirements or design
- Choice of the model influences the product
- Model should be realistic
- Model should be incomplete
 - abstraction: leave away unimportant things
- One model shows one viewpoint
 - you need several models to show several viewpoints
 - for example:
 - specification view versus design view
 - run time view versus compile time view
 - static view versus dynamic view
- Models in other disciplines: buildings, circuits...

© 2004. 497-1-26

Models and Reality



description of a program that runs on the machine and represents reality

source: Jackson

© 2004. 497-1-27

Levels of Modeling

0. Real Thing
1. Object
2. Class
3. Meta-Class

- Elements in each level model elements in the level below
 - Objects represent Real Things in computer memory
 - Classes represent objects in programs
 - Meta-classes represent classes in interpreters, compilers, meta-languages
- Levels of modeling outside of computers
 - Person – Driver's License – Driver's License Number

© 2004. 497-1-28

Structure of Models

- Designations
 - take things from the real world and import them into the model
 - are necessarily vague and imprecise
 - constraints about designations are called axioms
- Definitions
 - create new terms out of designated and defined terms
 - can be mathematically precise
- Example:
 - Designations:
Person, Birthday(Person), Today
 - Definition:
Age(Person) := Today – Birthday(Person)

© 2004. 497-1-29

Methods

- For example:
 - Process model
 - Iterative development
 - Software architecture
 - Testing
- Properties
 - Can be applied to which process phases?
 - Supported by tools or notations?
 - For large organizations / small organizations
 - Hard rules or soft recommendations?
 - Planability
 - Repeatability
 - Learning effort
 - Heavy-weight / light-weight (amount of paperwork)

© 2004. 497-1-30