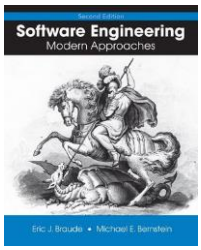
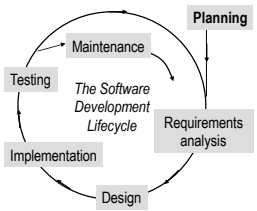
Software Engineering

Modern Approaches



Eric Braude and Michael Bernstein

Chapter 3: Software Process



Phase most relevant to this chapter is shown in bold

Learning goals of this chapter

- What are the main activities of software processes?
- What are the main software process types?
- How would a team go about selecting a process?

Software Process

- Software project composed of activities
  - E.g. planning, design, testing, etc.
- Activities organized into phases
- A *software process:*
  - prescribes the **order** and **frequency** of phases
  - specifies **criteria** for moving from one phase to the next
  - defines the **deliverables** of the project

Umbrella Activities

- Generic activities implemented ***throughout*** the life of a project (umbrella activities)
  - Project management
  - Configuration management
  - Quality management
  - Risk management

Software Process Benefits

- Process **DOES NOT** mean
  - "overhead"
  - "unnecessary paperwork"
  - "longer schedules"
  - etc.
- Software process has **positive** effect if applied correctly
  - Meet schedules
  - Higher quality
  - More maintainable

Software Process Phases

Phases of Software Processes

1. **Inception**
   Software product is conceived and defined
2. **Planning**
   Initial schedule, resources and cost are determined
3. **Requirements Analysis**
   Specify what the application must do; answers **"*what*?"**
4. **Design**
   Specify the parts and how they fit; answers **"*how*?"**
5. **Implementation**
   Write the **code**
6. **Testing**
   Execute the application with input test data
7. **Maintenance**
   Repair defects and add capability

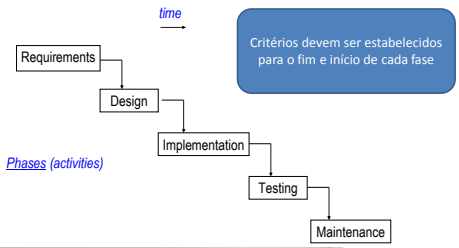
Example – Video Store Application

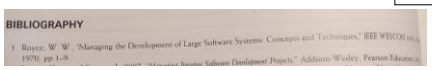
Software Process Phases: Video Store Example

- **Inception**
  " …An application is needed to keep track of video rentals …"
- **Planning** (Software Project Management Plan)
  "…The project will take 12 months, require 10 people and cost $2M…"
- **Requirements Analysis** (Product: Software Requirements Spec.)
  "…The clerk shall enter video title, renter name and date rented. The system shall …"
- **Design** (Software Design Document: Diagrams and text)
  " … classes *DVD*, *VideoStore*, …, related by …"
- **Implementation** (Source and object code)
  … class DVD{ String title; … } …
- **Testing** (Software Test Documentation: test cases and test results)
  "… Ran test case: *Rent "The Matrix" on Oct 3; rent "SeaBiscuit" on Oct 4; return "The Matrix" on Oct 10*…
  Result: *"SeaBiscuit" due Oct 4, 2004 balance of $8. (correct)* …"
- **Maintenance** (Modified requirements, design, code, and text)
  Defect repair: "Application crashes when balance is $10 and attempt is made to rent *"Gone With the Wind"*. …"
  Enhancement: "Allow searching by director."

Waterfall Process

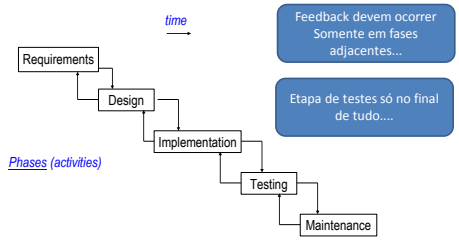
The Waterfall Software Process




BIBLIOGRAPHY
1. Royce, W. W., "Managing the Development of Large Software Systems: Concepts and Techniques," IEEE WESCON, 1970, pp. 1–9.

Waterfall Process – with feedback

The Waterfall Software Process with Feedback



Waterfall Process - Advantages

- Simple and easy to use
- Practiced for many years
- Easy to manage
- Facilitates allocation of resources
- Works well for smaller projects where requirements are very well understood

Waterfall Process - Disadvantages

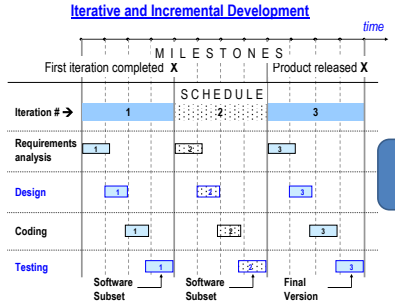
- Requirements must be known up front
- Hard to estimate reliably
- No feedback of system by stakeholders until after testing phase
- Major problems are discovered too late in process
- Lack of parallelism
  - Otherwise, disjoint parts could be completed in parallel…
- Inefficient use of resources (people are also resources)

Iterative and Incremental

- *Iterative*
  - repeated execution of the waterfall phases, in whole or in part, resulting in a refinement of the requirements, design and implementation
- *Incremental*
  - operational code produced at the end of an iteration
  - supports a subset of the final product functionality and features
- **Artifacts** evolve during each phase
- Artifacts considered complete only when software is released

Iterative and Incremental (cont.)

Iterative and Incremental Development



The "relased" is the final product !

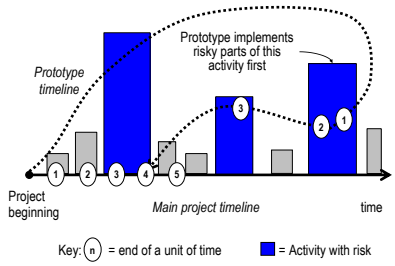
Release Types

- Proof of concept
  - Used to investigate the feasibility of a particular aspect of the software
- Prototype
  - A working version demonstrating a particular capability that is deemed to high risk.
- "Internal" release
- "External" release

Prototyping

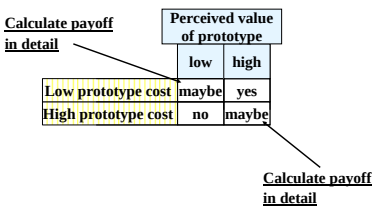
Protótipos podem ser construídos com objetivos diversos: interfaces ou avaliar grandes riscos.

Prototype Rationale



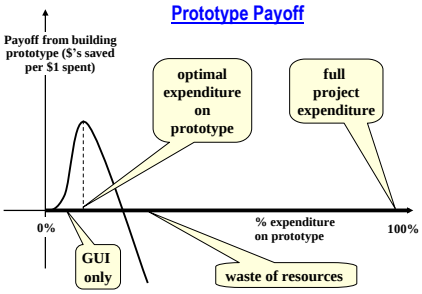
Prototyping (cont.)

Prototype Payoff: First Cut



Prototyping (cont.)

Prototype Payoff



1) Evitar tempo gasto no desenvolvimento de requisitos que o protótipo mostra que são desnecessários;
2) Retrabalho de mudar requisitos só depois que o cliente visualizasse o produto final.

Prototype Payoff Calculations for E-commerce Clothing Application

Valor que será economizado se o protótipo for construido. (ver fatores)

	Esti-mated cost	Gross Benefit excluding code re-use		Percentage of prototype code reused in application	Net Payoff		
		min	max		min	max	average
Prototype feature	*B*	*D*	*E*	*C*	*D-(1-C)B*	*E-(1-C)B*	
1. GUI screenshots	$10,000	$10,000	$80,000	50%	$5,000	$75,000	$40,000
2. Transaction security	$50,000	$10,000	$300,000	80%	$0	??	$145,000
3. Complete transaction	$80,000	$10,000	$400,000	50%	??	$200,000	165,000
4. Customer tries on clothing	$120,000	$20,000	$140,000	30%	-$64,000	$56,000	-$4,000
	260,000	50,000	920,000		- 89,000	781,000	346,000

Errado ! 360.000

Prototyping (cont.)

Prototype Payoff Calculations for E-commerce Clothing Application

	Esti-mated cost	Gross Benefit excluding code re-use		Percentage of prototype code reused in application	Net Payoff		
		min	max		min	max	average
Prototype feature	*B*	*D*	*E*	*C*	*D-(1-C)B*	*E-(1-C)B*	
1. GUI screenshots	$10,000	$10,000	$80,000	50%	$5,000	$75,000	$40,000
2. Transaction security	$50,000	$10,000	$300,000	80%	$0	$290,000	$145,000
3. Complete transaction	$80,000	$10,000	$400,000	50%	-$30,000	$200,000	$85,000
4. Customer tries on clothing	$120,000	$20,000	$140,000	30%	-$64,000	$56,000	-$4,000

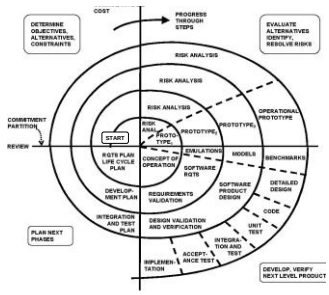
- Qual das features compensa mais e qual compensa menos ? Como julgar ?
- Qual o custo estimado de se implementar os quatro protótipos (com reuso) ?
- Considerando o valor max, quantos porcento o protótipo toma da receita ?

Spiral Model

- Barry Boehm, TRW Defense Systems,1988
- One of the earliest and best known iterative and incremental processes
- Risk-driven process
- Project starts at the center, and each cycle of the spiral represents one iteration
- Goal of each cycle is to increase the degree of system definition and implementation, while decreasing the degree of risk

Spiral Model

The Spiral Model



Spiral Model - Iteration Steps

1. Identify critical objectives and constraints
2. Evaluate project and process alternatives
3. Identify risks
4. Resolve (cost-effectively) a subset of risks using analysis, emulation, benchmarks, models and prototypes
5. Develop project deliverables including requirements, design, implementation and test
6. Plan for next and future cycles – update project plan including schedule, cost and number of remaining iterations
7. Stakeholder review of iteration deliverables and their commitment to proceed based on their objective being met

Spiral Model - Advantages

- ***Risks are managed early and throughout the process*** – risks are reduced before they become problematic
- ***Software evolves as the project progresses*** - errors and unattractive alternatives eliminated early.
- ***Planning is built into the process*** – each cycle includes a planning step to help monitor and keep a project on track.

Spiral Model - Disadvantages

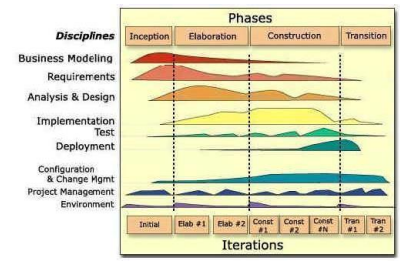
- ***Complicated to use*** - risk analysis requires highly specific expertise. There is inevitably some overlap between iterations.
- ***May be overkill for small projects*** – complication may not be necessary for smaller projects. Does not make sense if the cost of risk analysis is a major part of the overall project cost.

Unified Process

- Developed by Jacobson, Rumbaugh and Booch
- Major elaboration and refinement of Spiral
- Use-case driven
- Commercial product: Rational Unified Process (RUP)

Unified Process

Unified Process



http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

Inception

- Establish feasibility
- Make business case
- Establish product vision and scope
- Estimate cost and schedule, including major milestones
- Assess critical risks
- Build one or more prototypes

Elaboration

- Specify requirements in greater detail
- Architectural baseline
- Iterative implementation of core architecture
- Refine risk assessment and resolve highest risk items
- Define metrics
- Refine project plan, including detailed plan for beginning Construction iterations

Construction

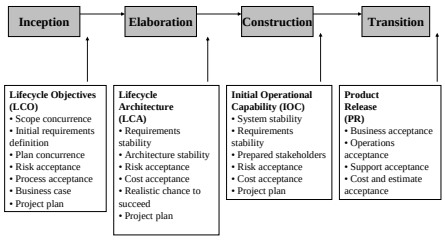
- Complete remaining requirements
- Iterative implementation of remaining design
- Thorough testing and preparation of system for deployment

Transition

- Beta tests
- Correct defects
- Create user manuals
- Deliver the system for production
- Training of end users, customers and support
- Conduct lessons learned

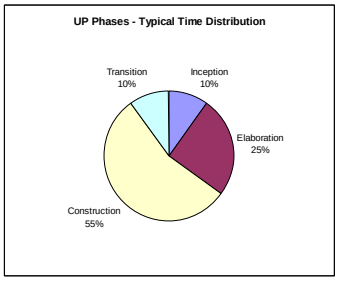
Unified Process Milestones

Unified Process Milestones



Adapted from http://www.ambysoft.com/downloads/managersIntroToRUP.pdf

Phases Times



Adapted from: Ambler, S.W., *A Manager's Introduction to the Rational Unified Process (RUP)*

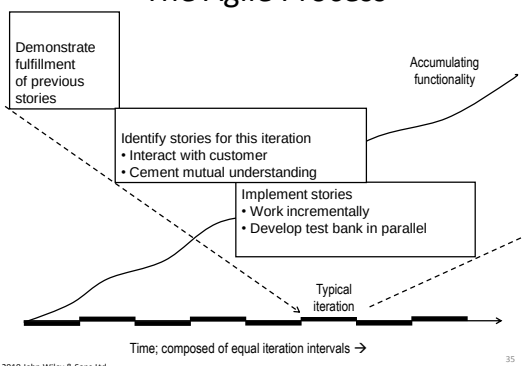
Agile Processes

The Agile Manifesto:

Agile processes 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

The Agile Process



Open Source

Reasons for Making a Project Open Source *1*

☺Leveraging large number of resources

☺Professional satisfaction

☺ To enable tailoring and integration

☺ Academic and research

☺ To gain extensive testing

☺ To maintain more stably

Richard Stallman
Originator of open source development
http://stallman.org/

Open Source (cont.)

Reasons for Making a Project Open Source *2*

☺To damage a competitor's product

☺To gain market knowledge

☺ To support a core business

☺ To support services

Open Source (cont.)

Reasons Against Open Source

☹ Documentation inconsistent or poor

☹No guarantee that developers will appear

☹No management control

☹No control over requirements

☹Visibility to competitors

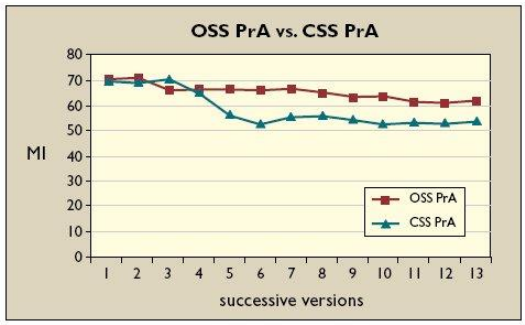
Bill Gates
http://www.microsoft.com/billgates/default.asp

Maintainability Index: OSS vs. CSS Same Application *2*

Project Mnemonic Code	Application Type	Total Code Size (KLOCs)	No. of releases measured	Project Evolution Path
OSSPrA	Operating system application	343	13	OSS project that gave birth to a CSS project while still evolving as OSS
CSSPrA	Operating system application	994	13	CSS project initiated from an OSS project and evolved as a commercial counterpart of OSSPrA

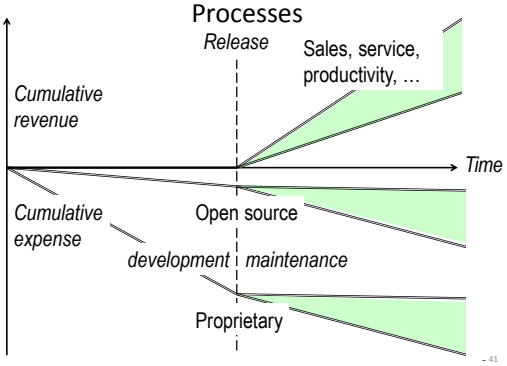
Communications of the ACM v 47, Number 10 (2004), Pp 83-87; Open source software development should strive for even greater code maintainability ; Ioannis Samoladas, Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou

Maintainability Index: OSS vs. CSS Same Application *2*



Communications of the ACM v 47, Number 10 (2004), Pp 83-87; Open source software development should strive for even greater code maintainability ; Ioannis Samoladas, Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou

Hybrid Open Source / Proprietary Processes



Collabnet Enterprise Edition

http://www.collab.net/products/index.html

Initial Student Team Meeting*: General Issues*

1. Set agenda and time limits.
2. Choose the team leader.
3. Get everyone's commitment to required time
   - Define an expected average number of hours per week
   - Gather dates of planned absences
4. Take a realistic census of team skills
   - Common problem: inflated programming skill claims
5. Begin forming a vision of the application
6. Decide how team will communicate.
7. Take meeting minutes with concrete action items

Communication Precepts

1. Listen to all with concentration
   - Don't pre-judge
2. Give all team members a turn
   - See the value in every idea
3. Don't make assumptions
   - Ask questions to clarify
4. When in doubt, communicate

Communication Plan

1. Meetings: Team will meet each Monday from ... to ... in …
   Caveat: do not replace face-to-face meeting with remote meetings unless remote meetings are clearly effective.
2. Meeting alternative: Team members should keep Fridays open from ... to ... in case an additional meeting is required.
3. Standards: Word processor, spreadsheet, compiler, ….
4. E-mail: Post e-mails?; require acknowledgement?
   Caveat: e-mail is poor for intensive collaboration
6. Collaboration: Tools for group collaboration and discussion –
   e.g. Yahoo Groups, Wiki tool, Google tools, …
7. Other tools: Microsoft Project (scheduling), Group calendar, …