V22.0474-001 Software Engineering
Spring 2008

Lecture 2

Clark Barrett, New York University

Review

What is Software Engineering?

“The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.”

Important principles

- Have a plan
- Don’t fall victim to common fallacies and pitfalls
- Many good approaches exist
- Hopefully better approaches are still to come

Outline

Software Processes

- Review and Introduction to Software Processes
- The Waterfall Model
- The Evolutionary Model
- Component-Based Software Engineering
- Choosing a Process
- Fundamental Activities of Software Processes
- Improving Software Processes

Sources for today's lecture:

Necula, George and Aiken, Alex. Software Engineering Lecture notes.

McConnell, Steve. *Code Complete*, Second Edition, ch. 3.

Sommerville, Ian. *Software Engineering*, Eighth Edition, ch. 4, 19, 28.

Sommerville, Ian. Presentation slides for ch. 4 from http://www.cs.st-andrews.ac.uk/∼ifs/Books/SE8/Presentations/.

Why should you care?

Software Processes

Most projects follow recognized stages. The set of steps which lead to a software product is called a *software process*.

Software processes

- organize the activities involved in producing a software product.
- focus on *how* things are to be done, not *what* is to be done.
- represent accumulated wisdom about software development.

Choosing the right process depends on many factors

- Size and scope of the project
- Likelihood of changes in the requirements
- Trade-offs between time-to-market and reliability
- Styles and preferences of development team
- etc.

Software Processes

Although many different software processes have been developed, there are some fundamental activities common to all processes.

Fundamental Activities

- *Software Specification* The functionality of the software and constraints on its operation must be defined.
- *Software Design and Implementation* The software is constructed to meet the specification.
- *Software Validation* The software must be validated to ensure that it is a correct implementation of the specification.
- *Software Evolution* The software must evolve to meet changing customer needs.

The way in which the fundamental activities are organized differs depending on the software process.

Software Process Models

We will look at three broad categories of software processes. Think of these as process paradigms or generic process models. Individual processes are often instances (or combinations) of these models.

The waterfall model

The fundamental activities are arranged sequentially, with each stage providing the input for the next stage.

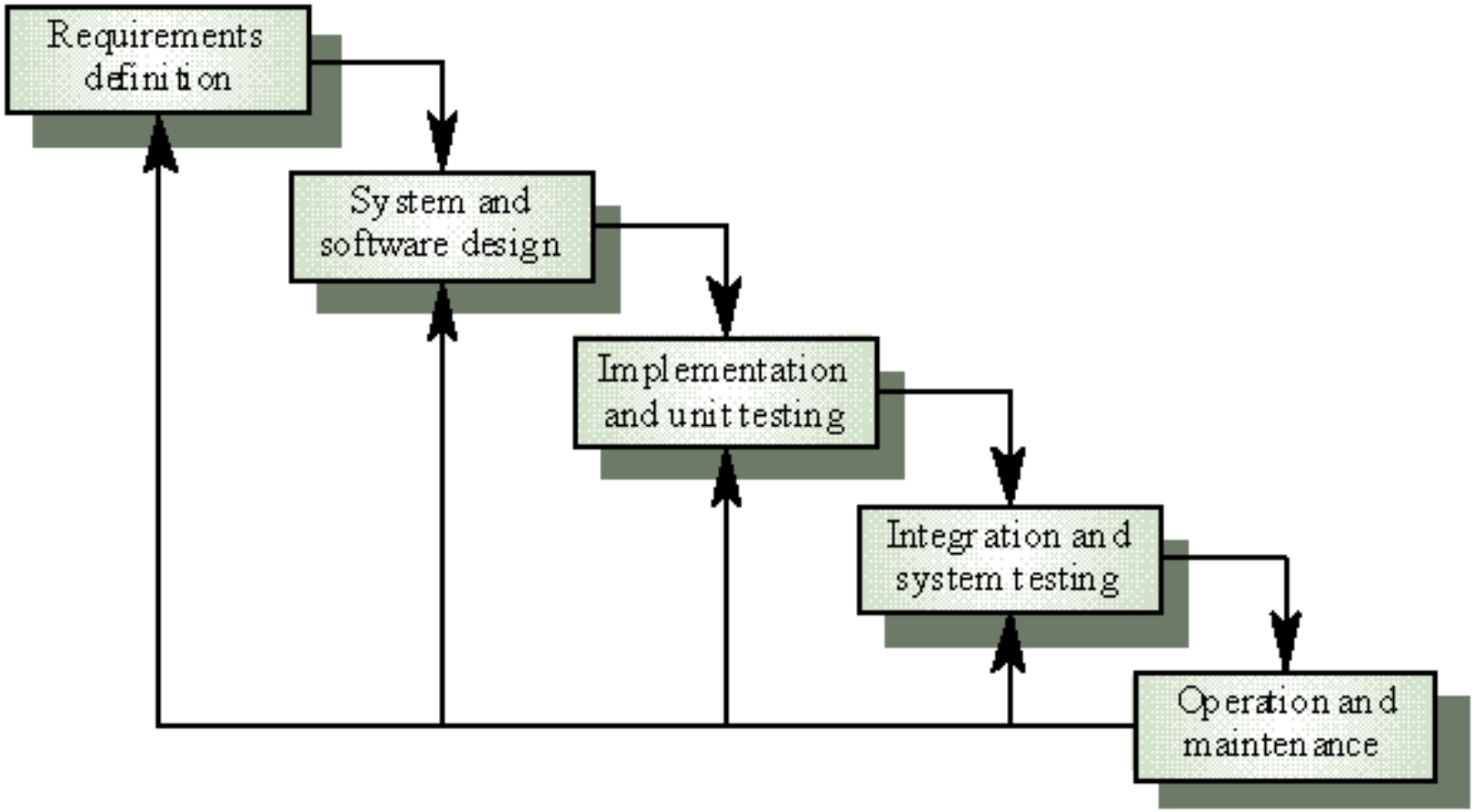
Evolutionary development (iterative model)

The fundamental activities are interleaved. An initial system is developed rapidly and then gradually refined in many small development cycles.

Component-based software engineering (building-block model)

This process focuses on integrating a number of existing components rather than developing them from scratch.

Waterfall model



The Waterfall Model

Sequential Development

- Based on other successful engineering processes (i.e. how to build a bridge)
- In principle, each phase is completed before moving to the next
- In practice, there is some overlap and iteration in the stages.

Top-down Design

- Requirements
- System Design
- Software Design

Bottom-up Implementation

- Unit Implementation and testing
- Module integration and testing
- System operation

The Waterfall Model

Discussion Question:

What are the advantages and disadvantages of the waterfall model?

The Waterfall Model

Discussion Question:

What are the advantages and disadvantages of the waterfall model?

Advantages

- Provides structured plan
- Based on other successful engineering processes
- Emphasis on communication from one stage to the next

The Waterfall Model

Discussion Question:

What are the advantages and disadvantages of the waterfall model?

Advantages

- Provides structured plan
- Based on other successful engineering processes
- Emphasis on communication from one stage to the next

Disadvantages

- Relies heavily on being able to accurately assess requirements at the start
- Little feedback from customers until relatively late
- Problems in requirements and design may not be found until very late
- May take a long time before first working version is ready (especially if requirements change often).

The Waterfall Model

Discussion Question:

Why is the waterfall model especially bad for software?

The Waterfall Model

Discussion Question:

Why is the waterfall model especially bad for software?

Requirements change more often and more rapidly than in other engineering disciplines

- Because there is no physical product, customers have the perception that making changes is easy
- Customers often don’t know what they want until they see the system in operation (lack of familiarity with final product, unlike building a bridge)
- Changing requirements delays the whole process

The Waterfall Model

Discussion Question:

Why is the waterfall model especially bad for software?

Requirements change more often and more rapidly than in other engineering disciplines

- Because there is no physical product, customers have the perception that making changes is easy
- Customers often don’t know what they want until they see the system in operation (lack of familiarity with final product, unlike building a bridge)
- Changing requirements delays the whole process

Time is the enemy of all software projects

The Waterfall Model

Discussion Question:

Why is the waterfall model especially bad for software?

Requirements change more often and more rapidly than in other engineering disciplines

- Because there is no physical product, customers have the perception that making changes is easy
- Customers often don't know what they want until they see the system in operation (lack of familiarity with final product, unlike building a bridge)
- Changing requirements delays the whole process

Time is the enemy of all software projects

- Technology changes rapidly
- Many products are obsolete before they ship
- Fast time-to-market gives a huge competitive advantage
- Software depends on components which are also changing

An (Unfortunately Typical) Example

California DMV Software

- Attempt to merge driver and vehicle registration systems

An (Unfortunately Typical) Example

California DMV Software

- Attempt to merge driver and vehicle registration systems

Initial Estimate in 1987

- 6 years
- $8 million

An (Unfortunately Typical) Example

California DMV Software

- Attempt to merge driver and vehicle registration systems

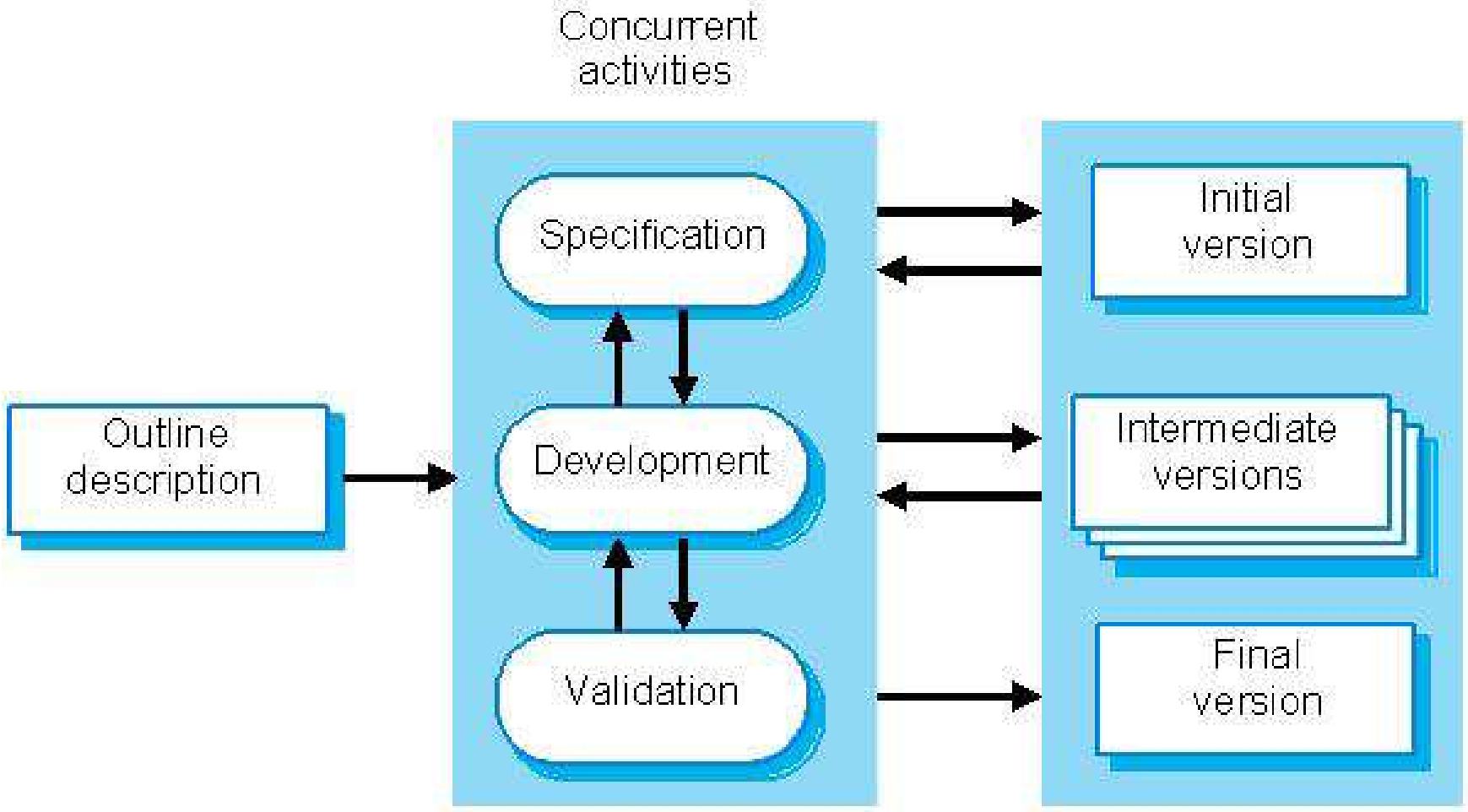
Initial Estimate in 1987

- 6 years
- $8 million

Results

- 7 years later
- $50 million spent
- Expected delivery slipped to 1998 (11 years!)
- Pulled the plug

Evolutionary Development



Evolutionary Development

Plan for change

- Same fundamental activities as waterfall
- Many quick iterations of the whole cycle
- Minimize risk by getting early feedback from customers

Variations

- *Exploratory development* Start with the part of the system that is understood. Work with the customer to add new features in each cycle.
- *Throwaway prototyping* Use a prototype to get feedback from customer and develop requirements. Then build the real product.

Evolutionary Development

Discussion Question:

What are the advantages and disadvantages of evolutionary development?

Evolutionary Development

Discussion Question:

What are the advantages and disadvantages of evolutionary development?

Advantages

- Specification can be developed incrementally
- Smaller cost for changes in requirements
- Progress is more quantifiable (percentage of requirements implemented in current cycle)

Evolutionary Development

Discussion Question:

What are the advantages and disadvantages of evolutionary development?

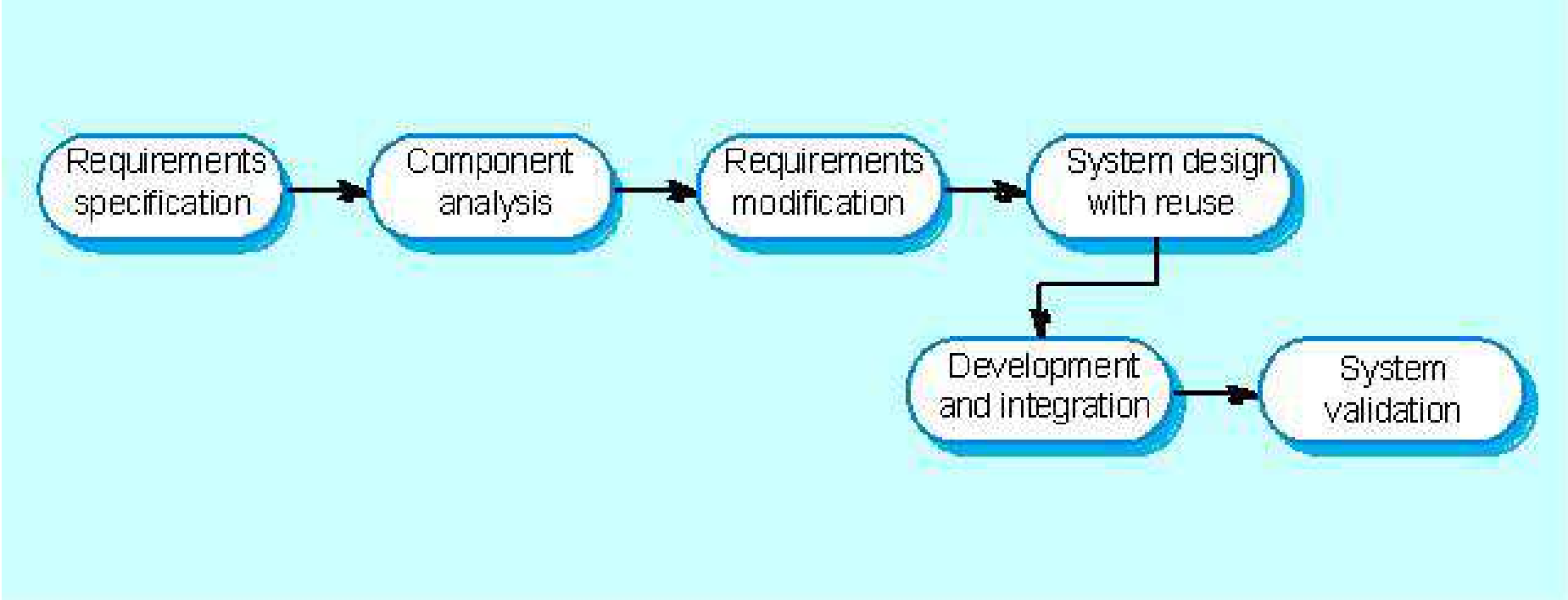
Advantages

- Specification can be developed incrementally
- Smaller cost for changes in requirements
- Progress is more quantifiable (percentage of requirements implemented in current cycle)

Disadvantages

- Less time spent on requirements could lead to mistakes
- Process is less visible to management and systems engineers
- Continual change can lead to poorly structured systems

Component-Based Software Engineering



Component-Based Software Engineering

Reuse-oriented Approach

- Relies on a large base of existing software components
- Framework for integrating components

Intermediate stages of component-based software engineering

- *Component analysis* Search for components which match the requirements.
- *Requirements modification* Requirements are modified to reflect available components.
- *System design with reuse* Components are organized into a system design. If components and requirements cannot be reconciled, some new software may need to be developed.
- *Development and integration* Components and newly developed code are integrated.

Component-Based Software Engineering

Discussion Question:

What are the advantages and disadvantages of component-based software engineering?

Component-Based Software Engineering

Discussion Question:

What are the advantages and disadvantages of component-based software engineering?

Advantages

- Less software to develop
- Less risk
- Faster time-to-market

Component-Based Software Engineering

Discussion Question:

What are the advantages and disadvantages of component-based software engineering?

Advantages

- Less software to develop
- Less risk
- Faster time-to-market

Disadvantages

- Requirements compromises are usually necessary
- Reusable components may not be as easy to change and maintain
- Black-box components can compromise system reliability and security

Choosing a Software Engineering Process

Discussion Question:

How would you decide which software engineering process to use?

Choosing a Software Engineering Process

Discussion Question:

How would you decide which software engineering process to use?

Waterfall

- Requirements not likely to change much
- Large or distributed project with many people involved
- Mission-critical or safety-critical systems

Evolutionary

- Small or medium-sized projects
- Business systems
- Time-to-market is most important consideration

Component-based

- Requires an existing base of components
- Requirements must match capabilities of components
- Must be willing to trust components

Choosing a Software Engineering Process

Often, the best process is one which is tailored to a specific project, incorporating the best of each process model.

Discussion Question:

How might you combine process models?

Choosing a Software Engineering Process

Often, the best process is one which is tailored to a specific project, incorporating the best of each process model.

Discussion Question:
How might you combine process models?

Hybrid processes

- Components used where possible or for initial prototype
- Critical components designed carefully with waterfall-like process
- Less critical components (e.g. user interface) use a more iterative approach
- Many other combinations possible

Fundamental Activities: Software Specification

Types of requirements

- Functional requirements
- Non-functional requirements: time, memory, power, etc.
- Environmental constraints

Phases of requirements engineering

- *Feasibility study* Estimate required resources. Determine whether proposed system is cost-effective.
- *Gathering requirements* Derive the system requirements through interaction with customers or potential users of the system.
- *Requirements specification* Gathered requirements are captured in a document of some kind.
- *Requirements validation* In this important step, requirements are checked for realism, consistency, and completeness.

Fundamental Activities: Design and Implementation

Design is a complex process which typically involves many sub-activities. Some of these may include the following:

- *Architectural design* Sub-systems and their relationships are identified and documented.
- *Abstract specification* Specification for each sub-system.
- *Interface design* Interface between sub-systems is designed and documented.
- *Component design* More detailed design of sub-systems
- *Data structure design* Data structures to be used in the implementation are specified in detail.
- *Algorithm design* Algorithms to be used in the implementation are specified in detail.

Structured methods of design rely on graphical models. One important structured method uses the *Unified Modeling Language (UML)*

We will have a lot more to say about design and implementation.

Fundamental Activities: Software Validation

Validation, verification, and testing all refer to the process of checking whether an implementation conforms to its specification.

Stages of testing

- *Component or unit testing* Individual components are tested, often by the designer of the component.
- *Integration testing* Find errors from unanticipated interactions and component interface problems.
- *System testing* Validate that the system meets its functional and non-functional requirements.
- *Acceptance (Alpha) testing* Test system with actual data from customers.
- *Beta testing* Test system with a limited pool of potential customers who agree to provide feedback.

Fundamental Activities: Software Evolution

The paradox of change

Ironically, the biggest headache for software developers is also one of software's greatest appeals: in many cases, requirements *can* be changed late in the design cycle, or even after the product is complete.

Although the cost is nontrivial, it is much cheaper than changes in hardware or other physical systems.

In the classical process models, evolution typically is equated with software maintenance, a process that takes place after design is complete.

As people move to more iterative process models, it is clear that software evolution is something that can and should be considered at every stage of development.

Improving Software Processes

Process improvement means understanding existing processes and changing them to increase product quality and/or reduce costs and development time.

Three stages of process improvement

- *Process measurement* Attributes of the current project or product are measured.
- *Process analysis* The current process is assessed, and process weaknesses and bottlenecks are identified.
- *Process change* Changes to the process that were identified during analysis are introduced.

Process improvement can range from very informal to very formal.

We will not discuss it further, but it is an important consideration for any development effort that includes more than one project.

Homework

Project Proposals

- Due by midnight on Sunday
- Please follow the instructions carefully

Next Assignment

- Vote on which proposal you like
- Candidates will be posted on Tuesday
- Votes are due by Thursday