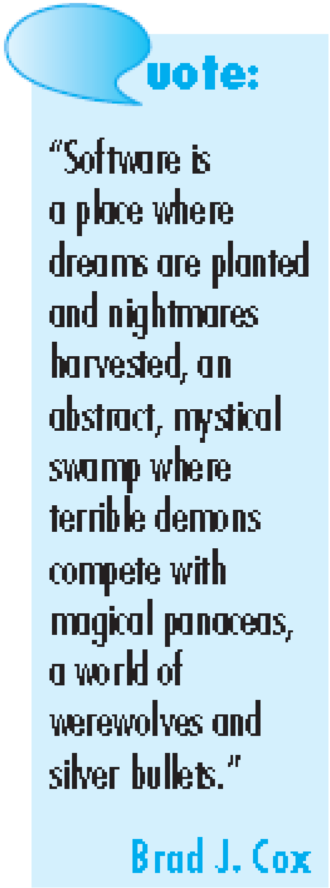
Software Engineering Unit-1

Part 1 Software Process

These slides are designed to accompany *Software Engineering: A Practitioner's Approach, 7/e* (McGraw-Hill 2009). Slides copyright 2009 by Roger Pressman.

What is Software?

- Software is:
  - (1) *instructions* (computer programs) that when executed provide desired features, function, and performance;
  - (2) *data structures* that enable the programs to adequately manipulate information and
  - (3) *documentation* that describes the operation and use of the programs.



uote:

"Software is a place where dreams are planted and nightmares harvested, an abstract, mystical swamp where terrible demons compete with magical panaceas, a world of werewolves and silver bullets."

Brad J. Cox

Questions about Software?

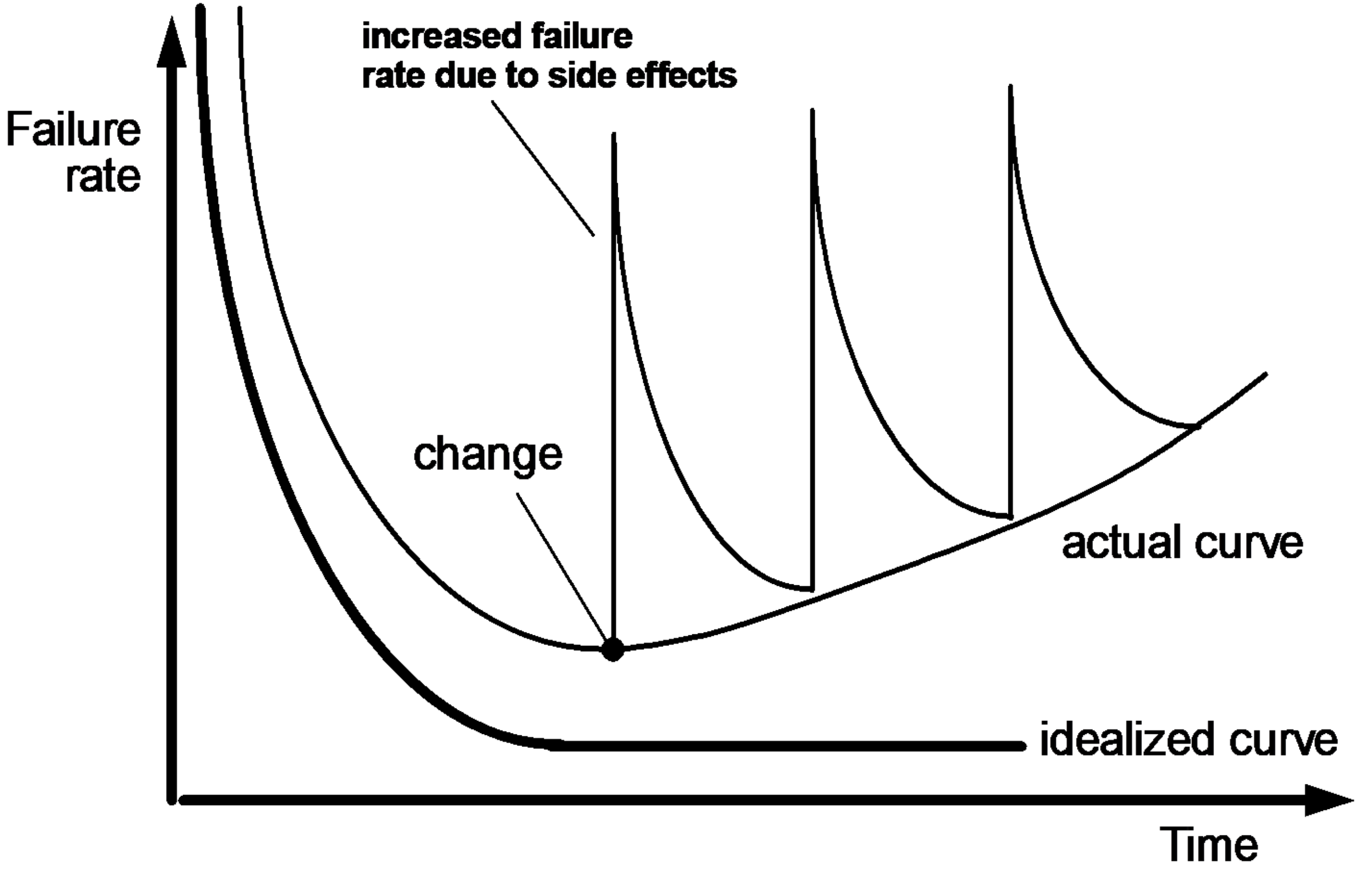
- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can’t we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

Does Software has expiration?

Does Software wear out?

- ***Software is developed or engineered, it is not manufactured in the classical sense.***
- ***Software doesn't "wear out."***
- ***Although the industry is moving toward component-based construction, most software continues to be custom-built.***

Wear vs. Deterioration



Software Applications

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- WebApps (Web applications)
- AI software

Software Applications

- System software
- **System software— a collection of programs written to service other** programs.
  - e.g., compilers, editors, and file management utilities. (Determinate)
  - operating system components, drivers, networking software, telecommunications processors (Indeterminate)

Software Applications

- **Application software** —stand-alone programs that solve a specific business need.
- **Engineering/scientific software** —a broad array of “number-crunching programs
  - Astronomy
  - Automotive
  - Orbital dynamics,
  - Computer-aided design
  - Molecular biology
  - Genetic analysis
  - Meteorology.

Software Applications

- **Embedded software—** resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.
- Limited and esoteric functions
- Provide significant function and control capability
- e.g., digital functions in an automobile such as fuel control, dashboard displays, and braking systems

Software Applications

- **Product-line software** —designed to provide a specific capability for use by many different customers.
- Address mass consumer
- e.g., inventory control products
- **Web/Mobile applications**
- **Artificial intelligence software**
  - Applications within this area include robotics, expert systems, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing.

Legacy Software

- These older programs—often referred to as *legacy software —have been the* focus of continuous attention and concern since the 1960s.

Legacy software systems . . . *were developed decades ago and have been continually* modified to meet changes in business requirements and computing platforms. The proliferation of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve.

Dayani-Fard

Legacy Software

- The software must be adapted to meet the needs of new computing environments or technology.
- The software must be enhanced to implement new business requirements.
- The software must be extended to make it interoperable with other more modern systems or databases.

The software must be re-architected to make it viable within a evolving computing environment.

? What do I do if I encounter a legacy system that exhibits poor quality?

? What types of changes are made to legacy systems?

THE CHANGING NATURE OF SOFTWARE

WebApps

Mobile Applications

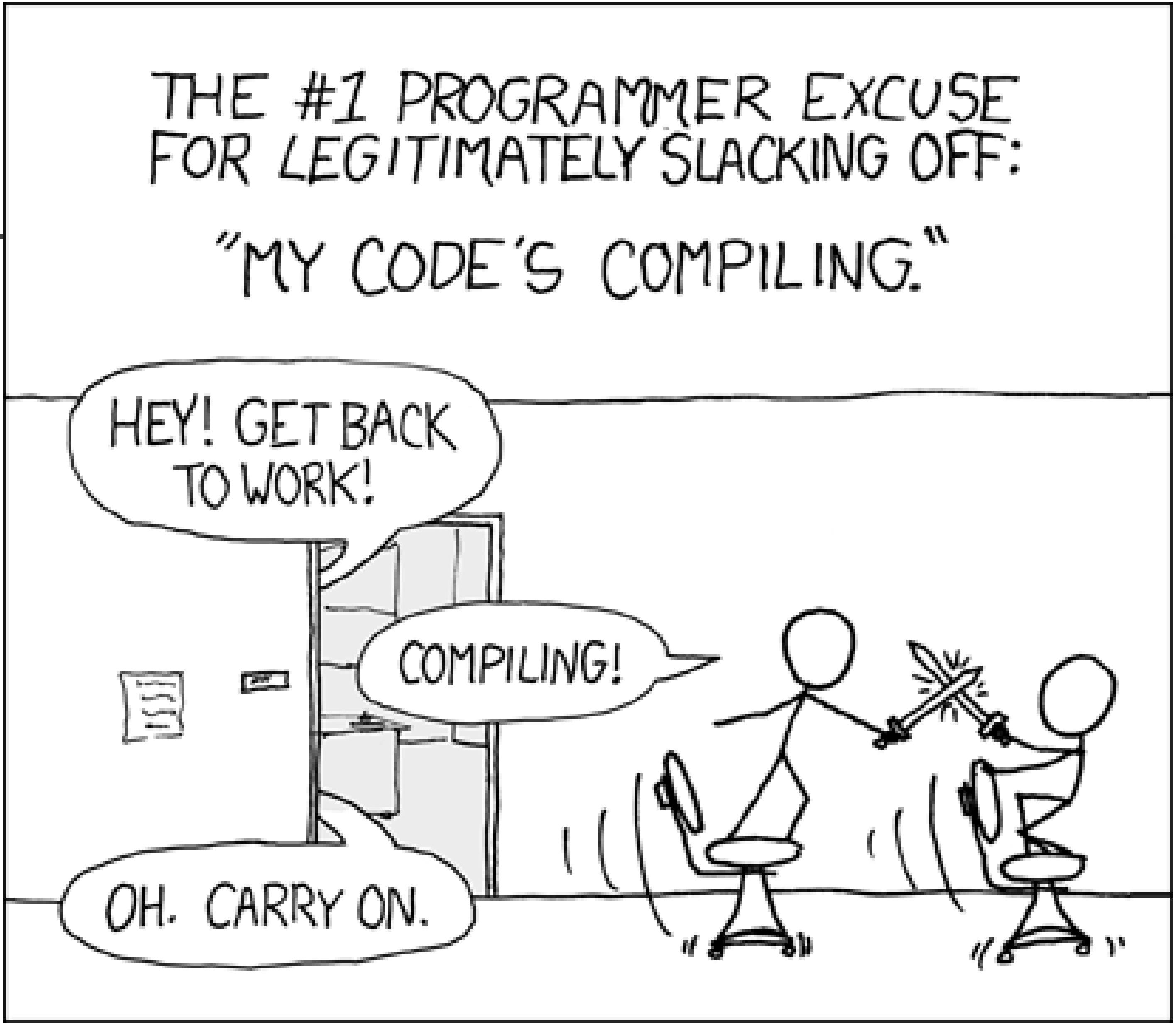
Cloud Computing

Provide examples (both positive and negative) that indicate the impact of software on our society

Software Engineering

- The IEEE definition:
  - *Software Engineering:*
  - *(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
  - *(2) The study of approaches as in (1).*

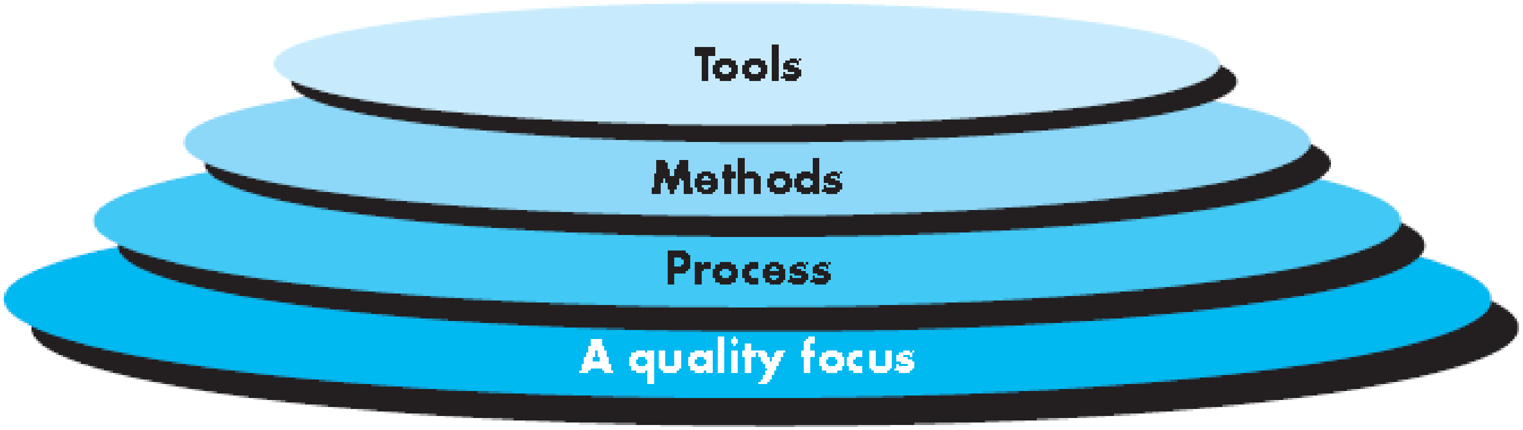
How do you want to develop a Software? Where do you generally start?



THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."
HEY! GET BACK TO WORK!
COMPILING!
OH. CARRY ON.

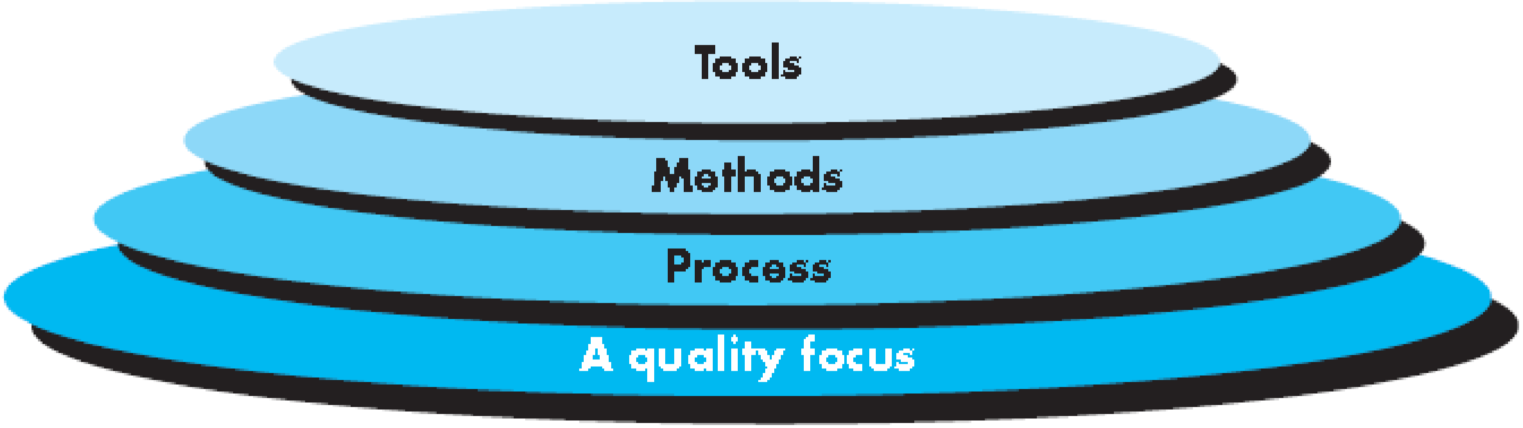
A Layered Technology

- Any engineering approach (including software engineering) must rest on an organizational commitment to quality.



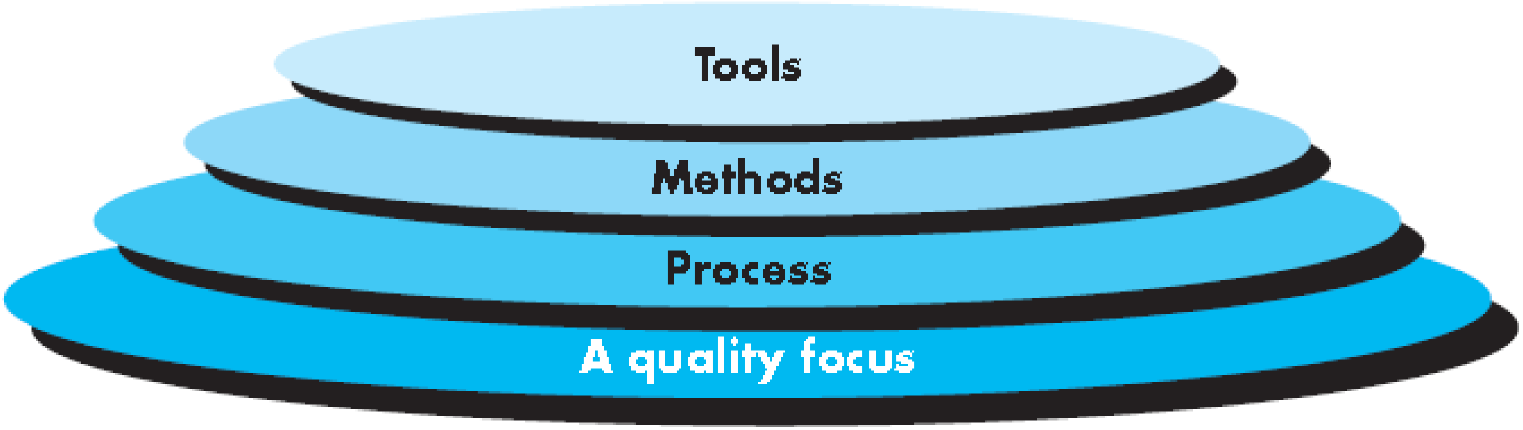
A Layered Technology

- The software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.
- *Process* defines a framework that must be established for effective delivery of software engineering technology.



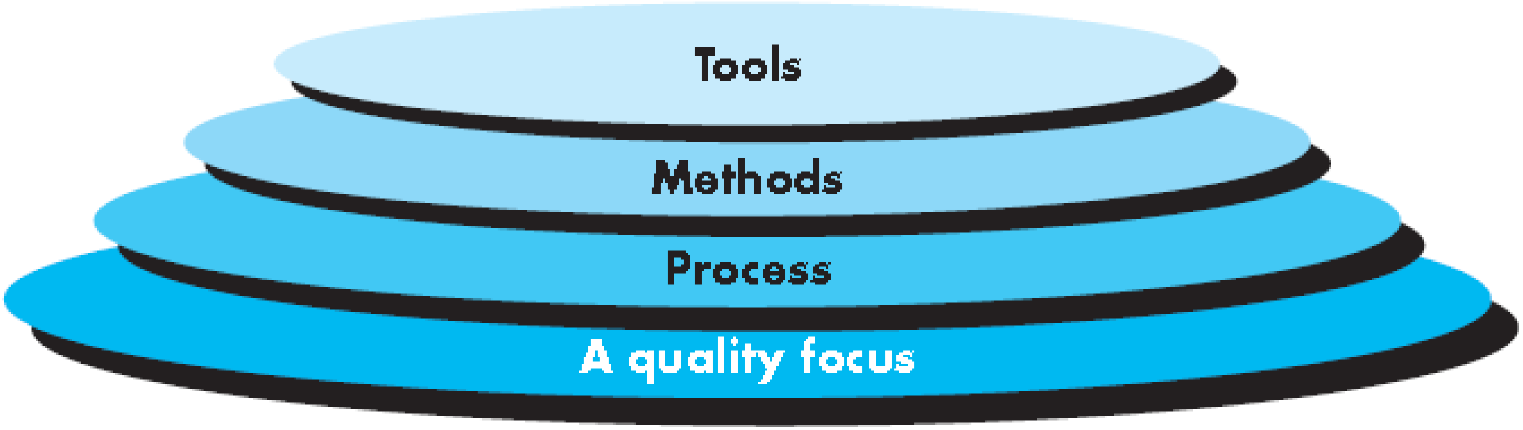
A Layered Technology

- Software engineering ***methods*** provide the technical how-to's for building software.
- Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing, and support.



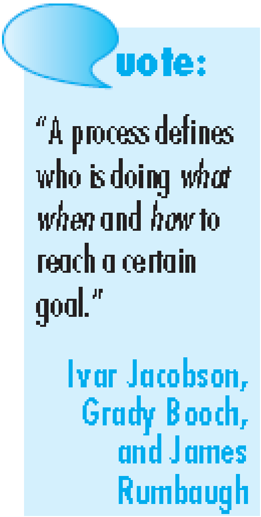
A Layered Technology

- Software engineering ***tools*** provide automated or semi-automated support for the process and the methods.



THE SOFTWARE PROCESS

- A *process is a collection of activities, actions, and tasks that are performed when* some work product is to be created.
- An *activity strives to achieve a broad objective* (e.g., communication with stakeholders).
- An *action (e.g., architectural* design) encompasses a set of tasks that produce a major work product (e.g., an architectural model).
- A *task focuses on a small, but well-defined objective (e.g.,* conducting a unit test) that produces a tangible outcome.



Quote:

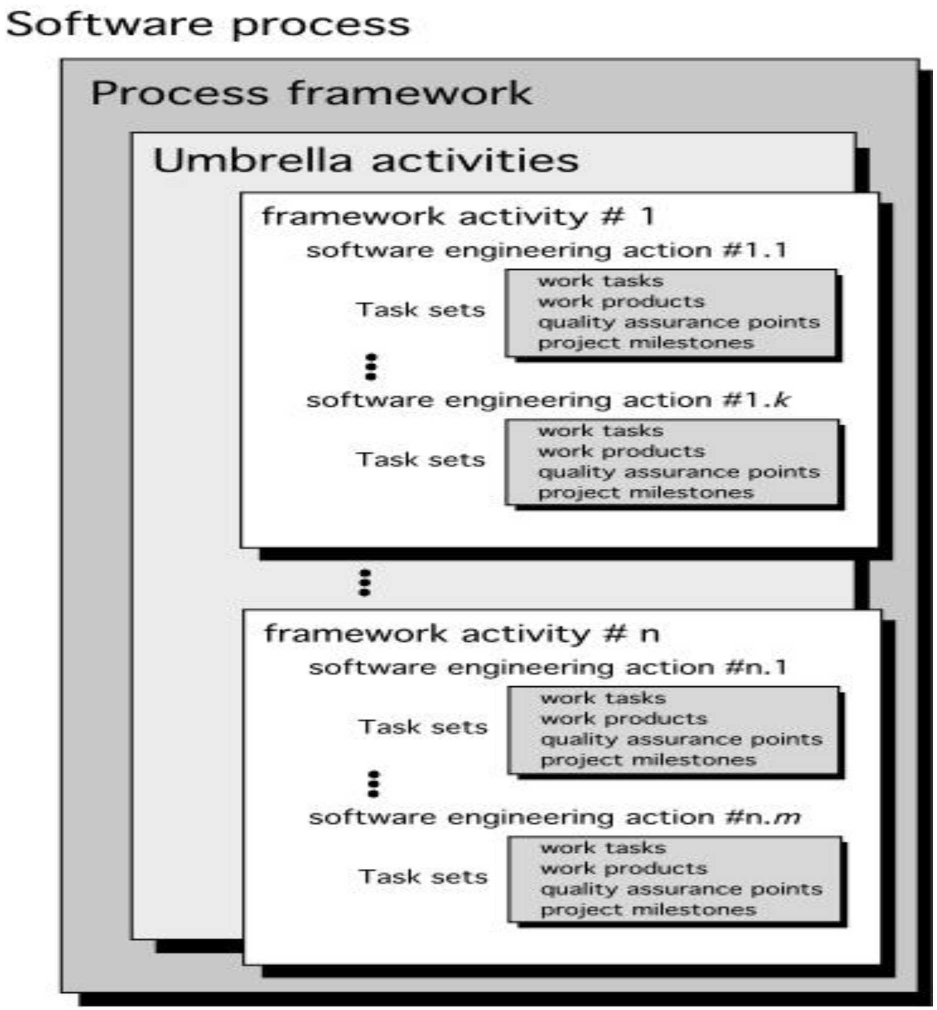
"A process defines who is doing *what* when and *how* to reach a certain goal."

Ivar Jacobson, Grady Booch, and James Rumbaugh

The Process Framework

- A ***process framework*** establishes the foundation by identifying a small number of *framework activities* that are applicable to all software projects.
- The process framework encompasses a set of *umbrella activities that are* applicable across the entire software process.

The Process Framework



Software process

Process framework

Umbrella activities

framework activity # 1

software engineering action #1.1

Task sets

work tasks
work products
quality assurance points
project milestones

⋮

software engineering action #1.*k*

Task sets

work tasks
work products
quality assurance points
project milestones

⋮

framework activity # n

software engineering action #n.1

Task sets

work tasks
work products
quality assurance points
project milestones

⋮

software engineering action #n.*m*

Task sets

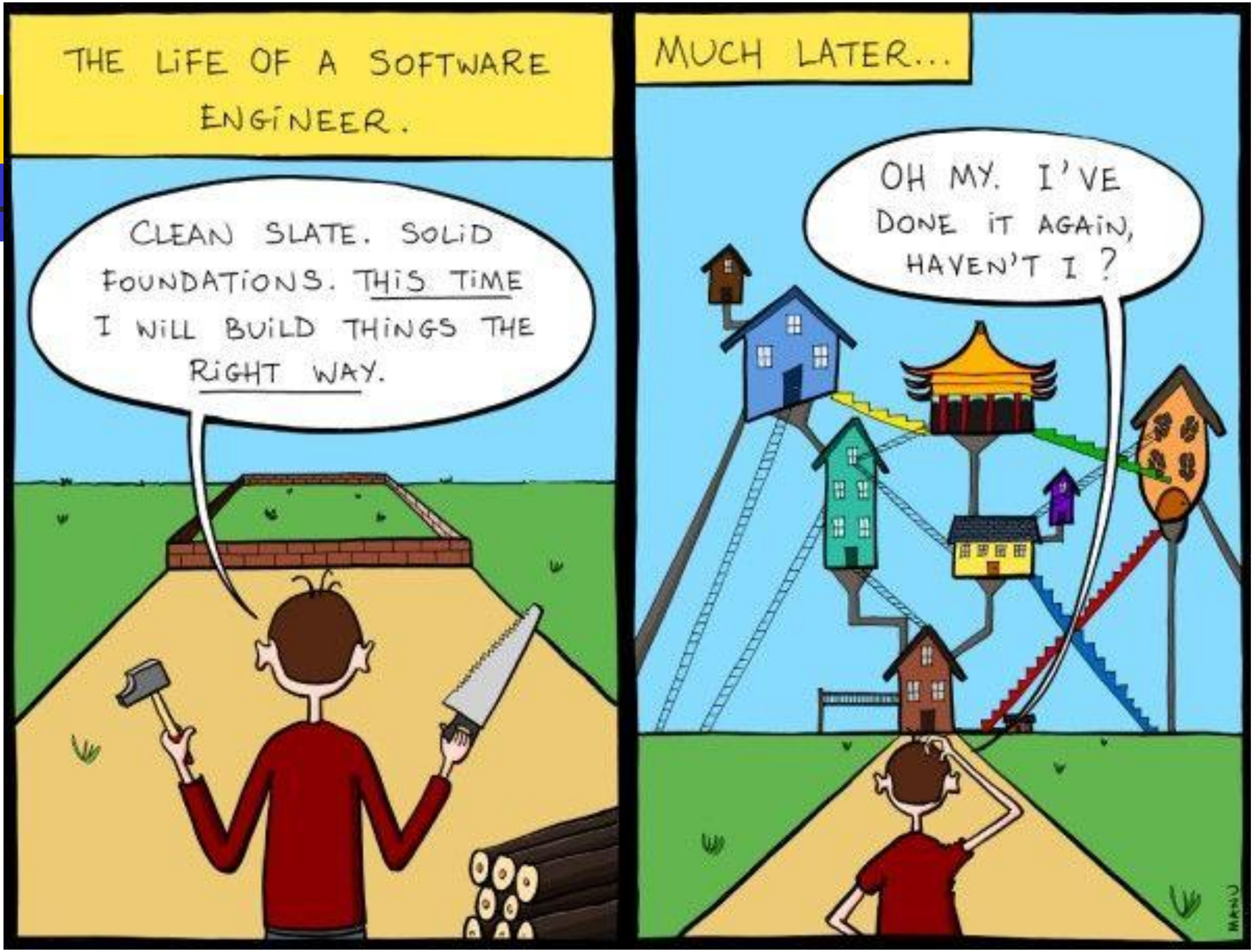
work tasks
work products
quality assurance points
project milestones

The Process Framework

- A generic process framework for software engineering encompasses five activities.
- **Communication.**
  - Communicate and collaborate with the customer (and other stakeholders).
  - The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions.

The Process Framework

- A generic process framework for software engineering encompasses five activities.
- **Planning.**
  - Defines the software engineering work by describing the technical tasks to be conducted.
  - Risks that are likely, the resources that will be required, the work products to be produced, and a work schedule


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

The Process Framework

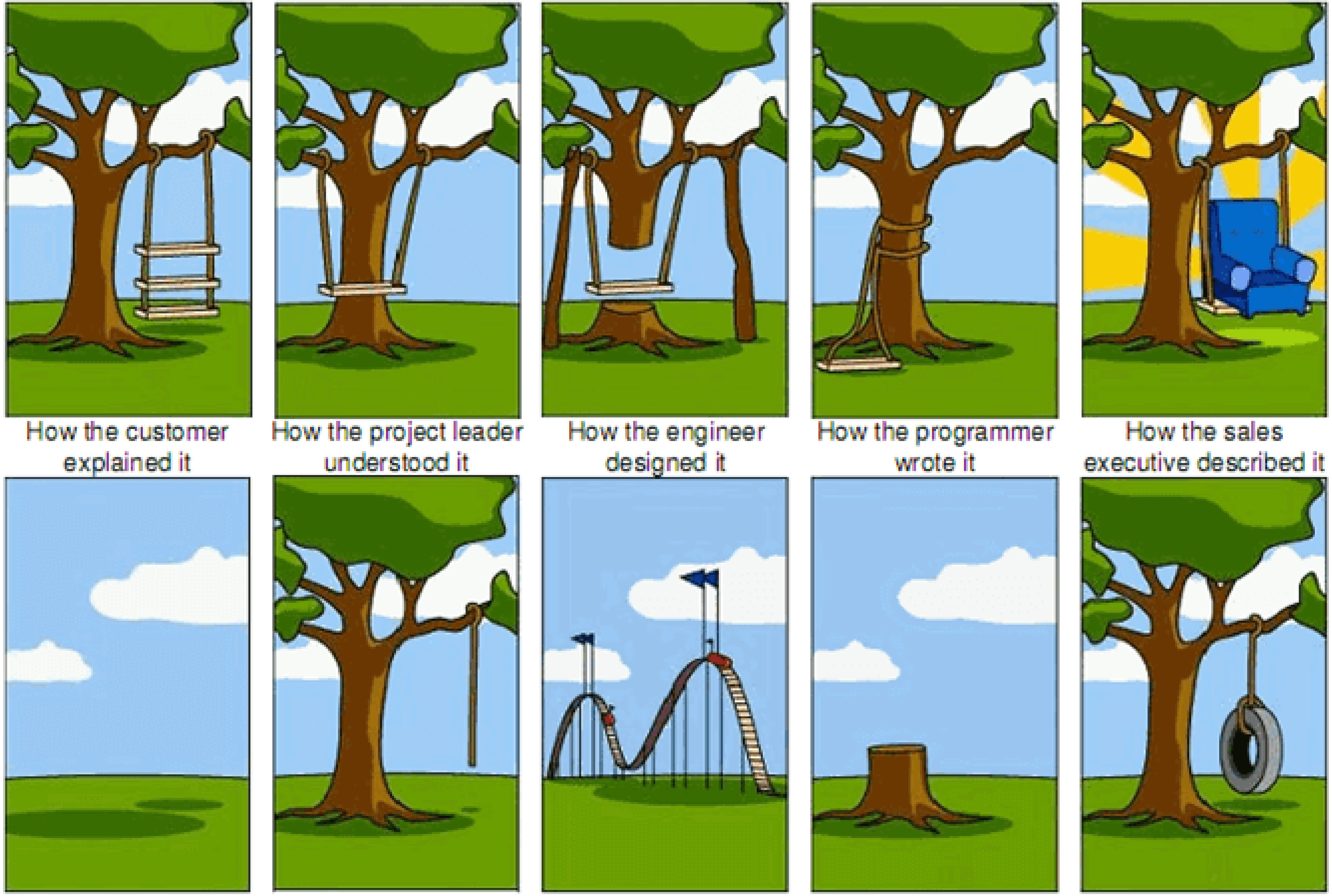
- A generic process framework for software engineering encompasses five activities.
- **Modeling.**
  - Better understand software requirements and the design that will achieve those requirements.
  - Waterfall
  - V-model
  - Prototype
  - Agile

The Process Framework

- A generic process framework for software engineering encompasses five activities.
- **Construction.**
  - Code generation (either manual or automated) and the testing that is required to uncover errors in the code.

The Process Framework

- A generic process framework for software engineering encompasses five activities.
- **Deployment.**
  - The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.



How the customer explained it

How the project leader understood it

How the engineer designed it

How the programmer wrote it

How the sales executive described it

How the project was documented

What operations installed

How the customer was billed

How the helpdesk supported it

What the customer really needed

Umbrella Activities

- **Software project tracking and control**
  - allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.
- **Risk management**
  - assesses risks that may affect the outcome of the project or the quality of the product.
- **Software quality assurance**
  - Defines and conducts the activities required to ensure software quality.

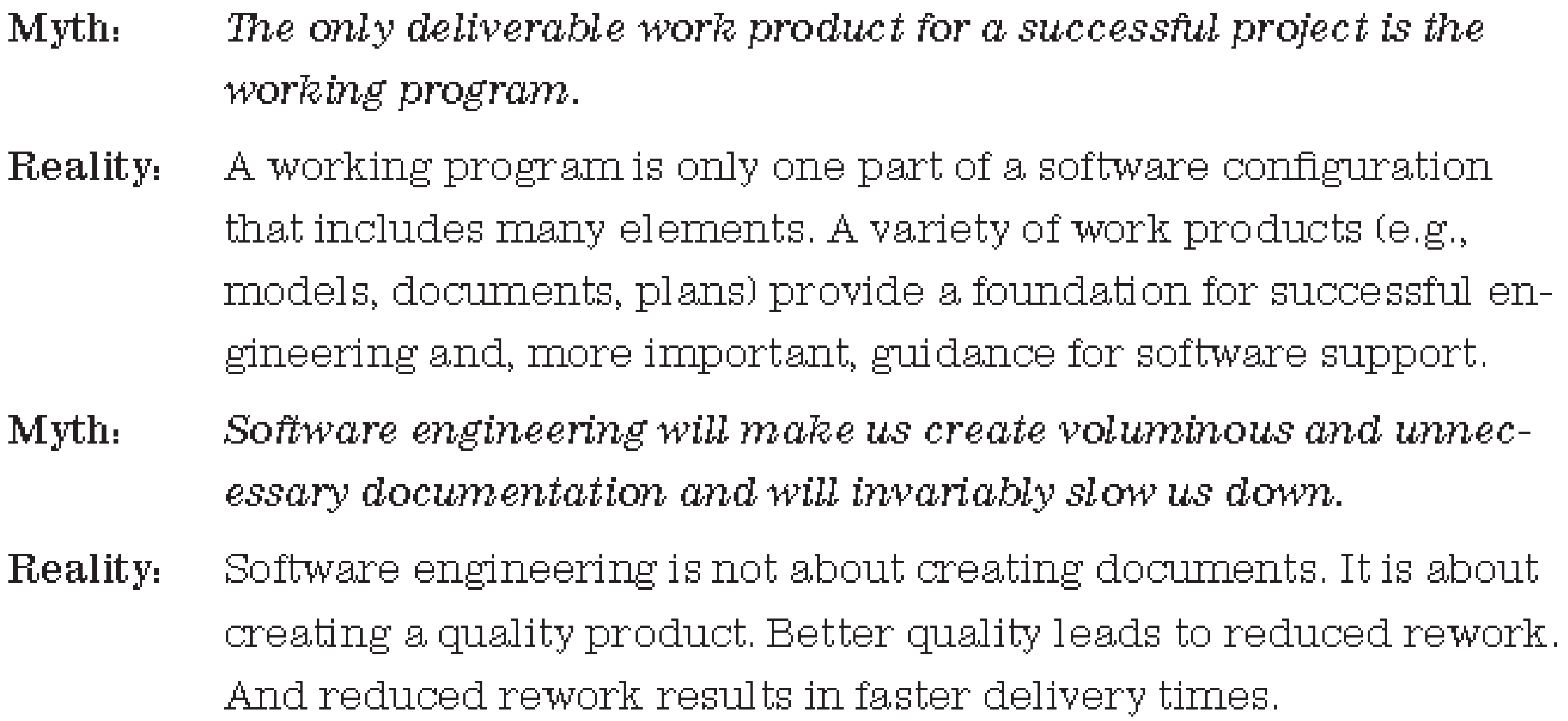
Umbrella Activities

- **Technical reviews**
  - assess software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.
- **Measurement**
  - Defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs.
- **Software configuration management**
  - Manages the effects of change throughout the software process.
- **Reusability management**
  - Defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.
- **Work product preparation and production**
  - encompass the activities required to create work products such as models, documents, logs, forms, and lists.

Process Adaptation

- A process adopted for one project might be significantly different than a process adopted for another project.
- Differences are
  - **Overall flow of activities**, actions, and tasks and the interdependencies among them.
  - Degree to which **actions and tasks** are defined within each framework activity.
  - Degree to which **work products** are identified and required.
  - Manner in which **quality assurance activities** are applied.
  - Manner in which **project tracking and control activities** are applied.
  - Overall degree of **detail and rigor** with which the process is described.
  - Degree to which the **customer and other stakeholders** are involved with the project.
  - Level of **autonomy** given to the software team.
  - Degree to which **team organization and roles** are prescribed.

Process Adaptation



Myth: *The only deliverable work product for a successful project is the working program.*

Reality: A working program is only one part of a software configuration that includes many elements. A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software support.

Myth: *Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.*

Reality: Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

Summary

- Software engineering encompasses process, methods, and tools that enable complex computer-based systems to be built in a timely manner with quality.
- The software process incorporates five framework activities—communication, planning, modeling, construction, and deployment—that are applicable to all software projects.
- Software engineering practice is a problem-solving activity that follows a set of core principles.

Recap

- What is Software Engineering?
- What is a Software Process?
- What are the activities in a Software Process?
- What are the umbrella activities in a Software Process?

Part -2 Process Models

Process model

- A process model provides a specific roadmap for software engineering work.
- It defines the flow of all activities, actions and tasks, the degree of iteration, the work products, and the organization of the work that must be done.

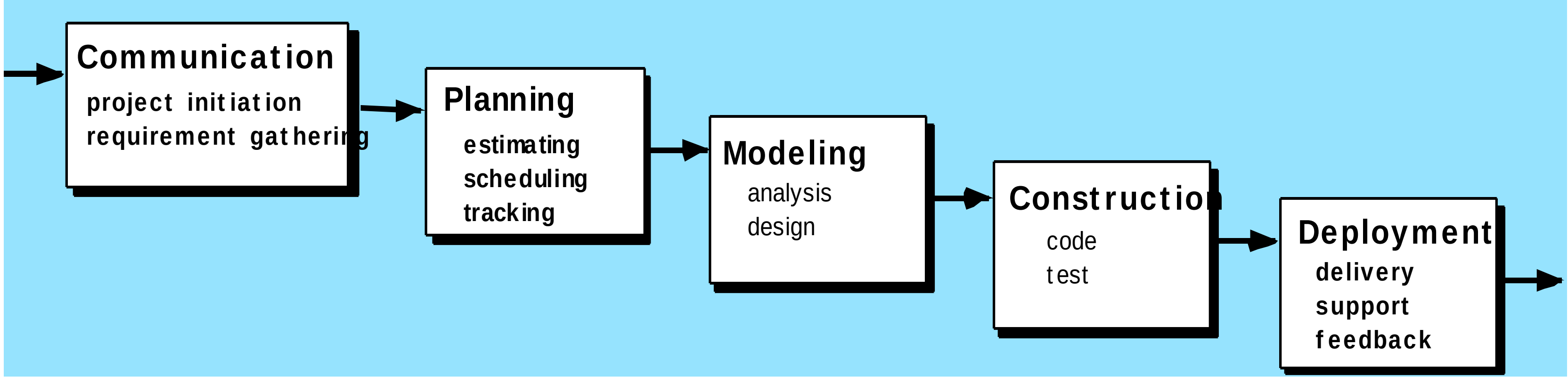
Prescriptive Models

- A *prescriptive process model* strives for structure and order in software development.

That leads to a few questions …

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

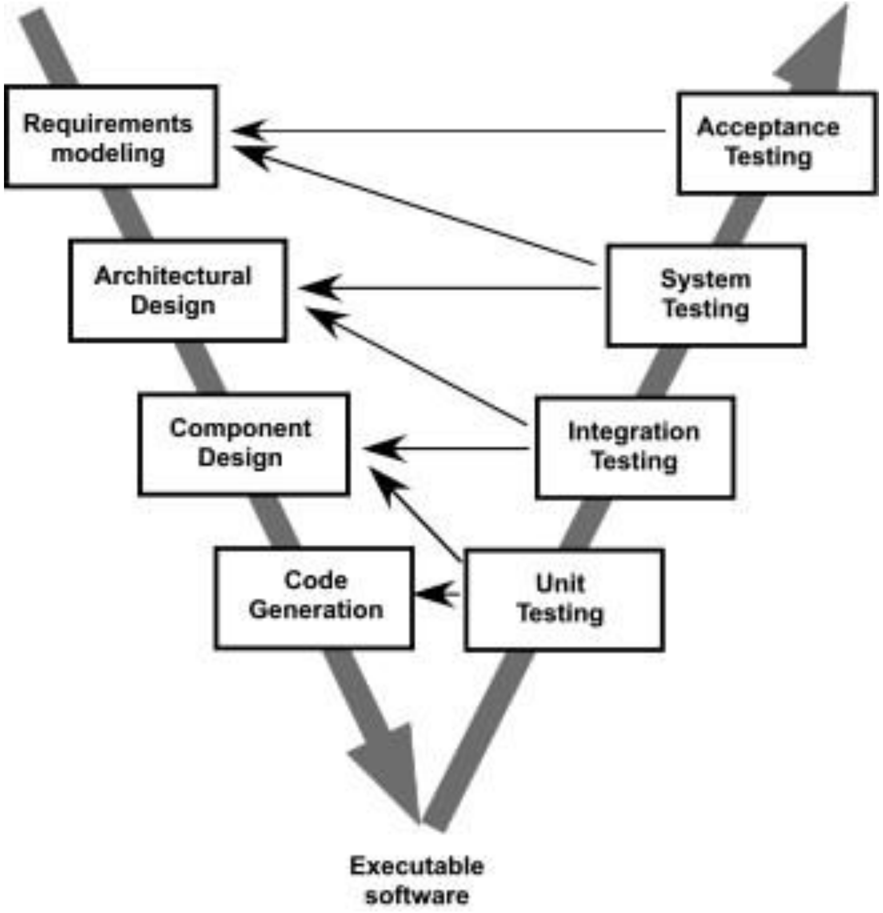
The Waterfall Model



- Classic life cycle
- Systematic, sequential approach

The V-Model

- Relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities.
- It provides a way of visualizing how verification and validation actions are applied to earlier engineering work.



Problems

- Real projects rarely follow the sequential flow.
  - Changes can cause confusion as the project team proceeds.
- Difficult for the customer to state all requirements explicitly.
- A working version of the program(s) will not be available until late in the project time span.

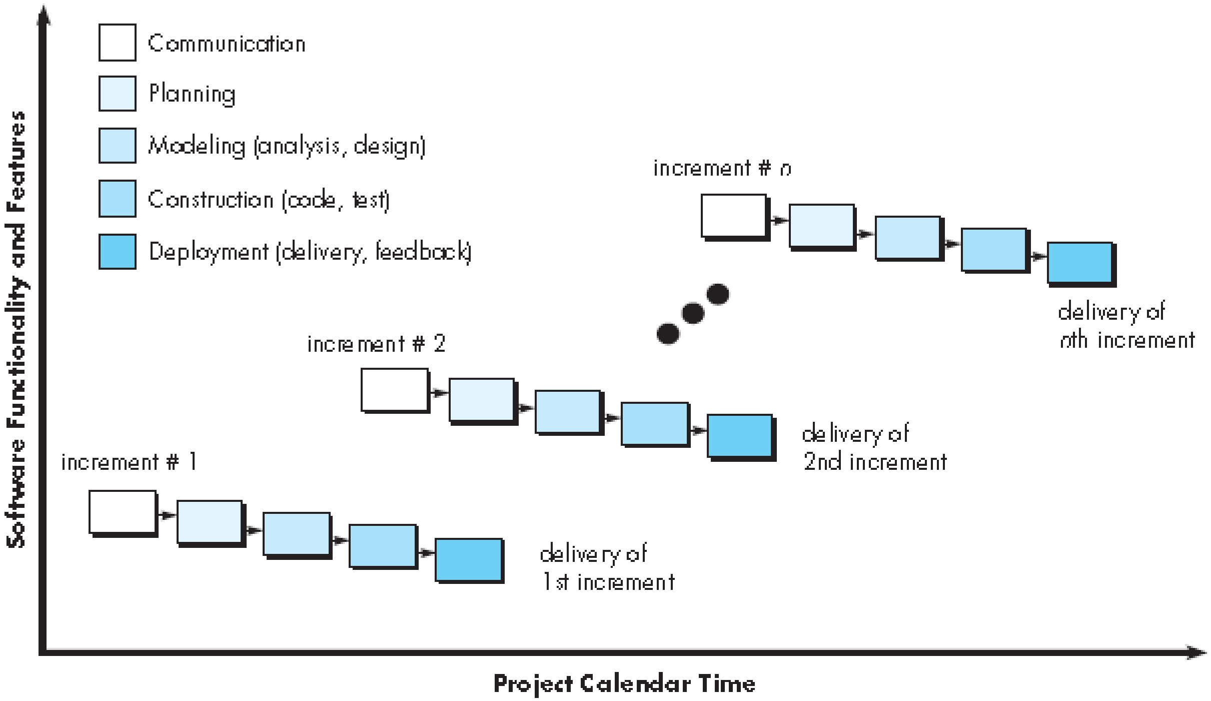
Waterfall model in Practice

- DOD-STD-2167A
- Defense Systems Software Development, February 29, 1988.
- This document established "uniform requirements for the software development that are applicable throughout the system life cycle.
- The standard was that it was biased toward the Waterfall Model.

Basically, due to a new spam reduction policy enacted by Twitter, Buffer had to remove automated retweets from their product within 2 weeks. If late, they risked six million users losing access to Twitter. In this case, they didn’t have time for iteration — they set final requirements and got to work on delivering.

https://en.wikipedia.org/wiki/DOD-STD-2167A

Incremental Process Models



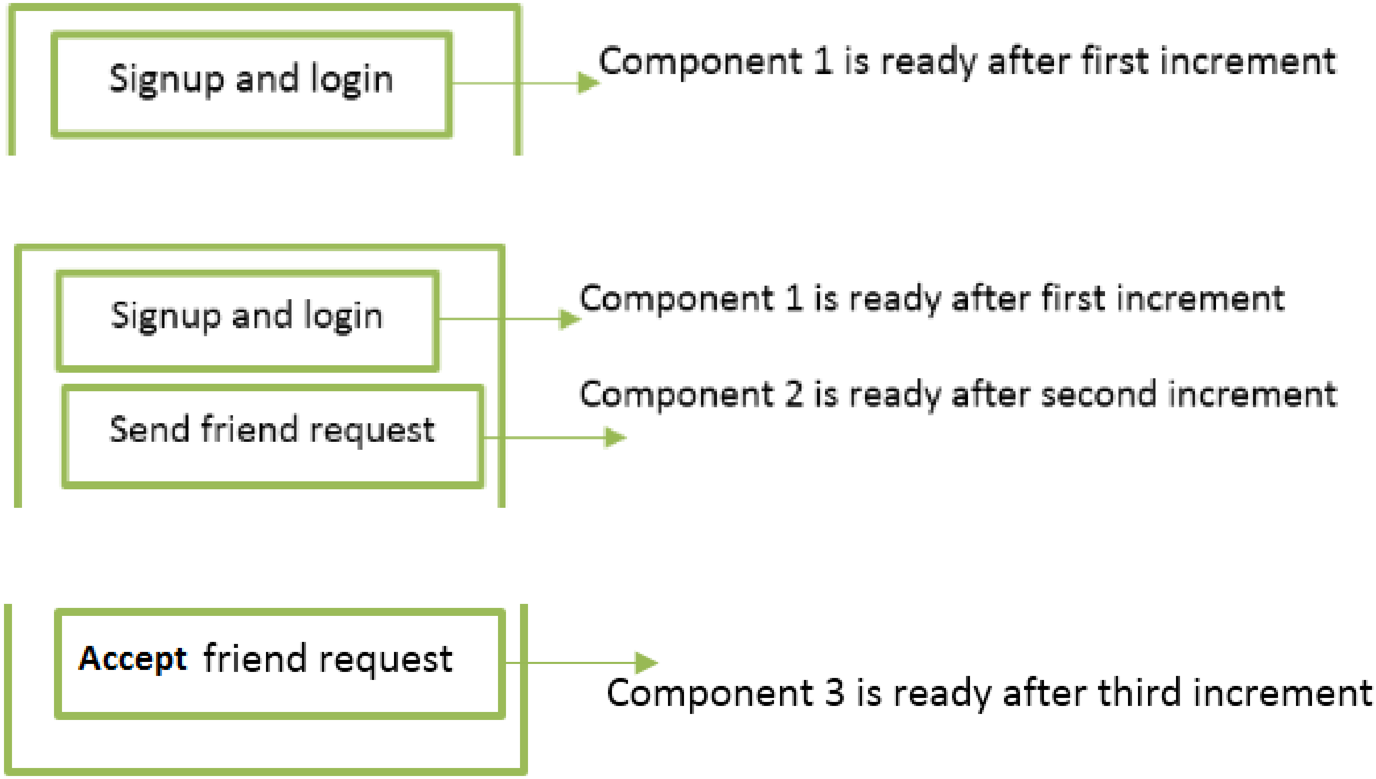
- Process model is designed to produce the software in increments

Incremental Process Models

- The first increment is often a core product.
- For example, word-processing software
  - basic file management, editing..
  - spelling and grammar..
  - advanced page layout capability..

Incremental Process Models Example

- Web-based social network



Signup and login

Component 1 is ready after first increment

Signup and login

Send friend request

Component 1 is ready after first increment

Component 2 is ready after second increment

Accept friend request

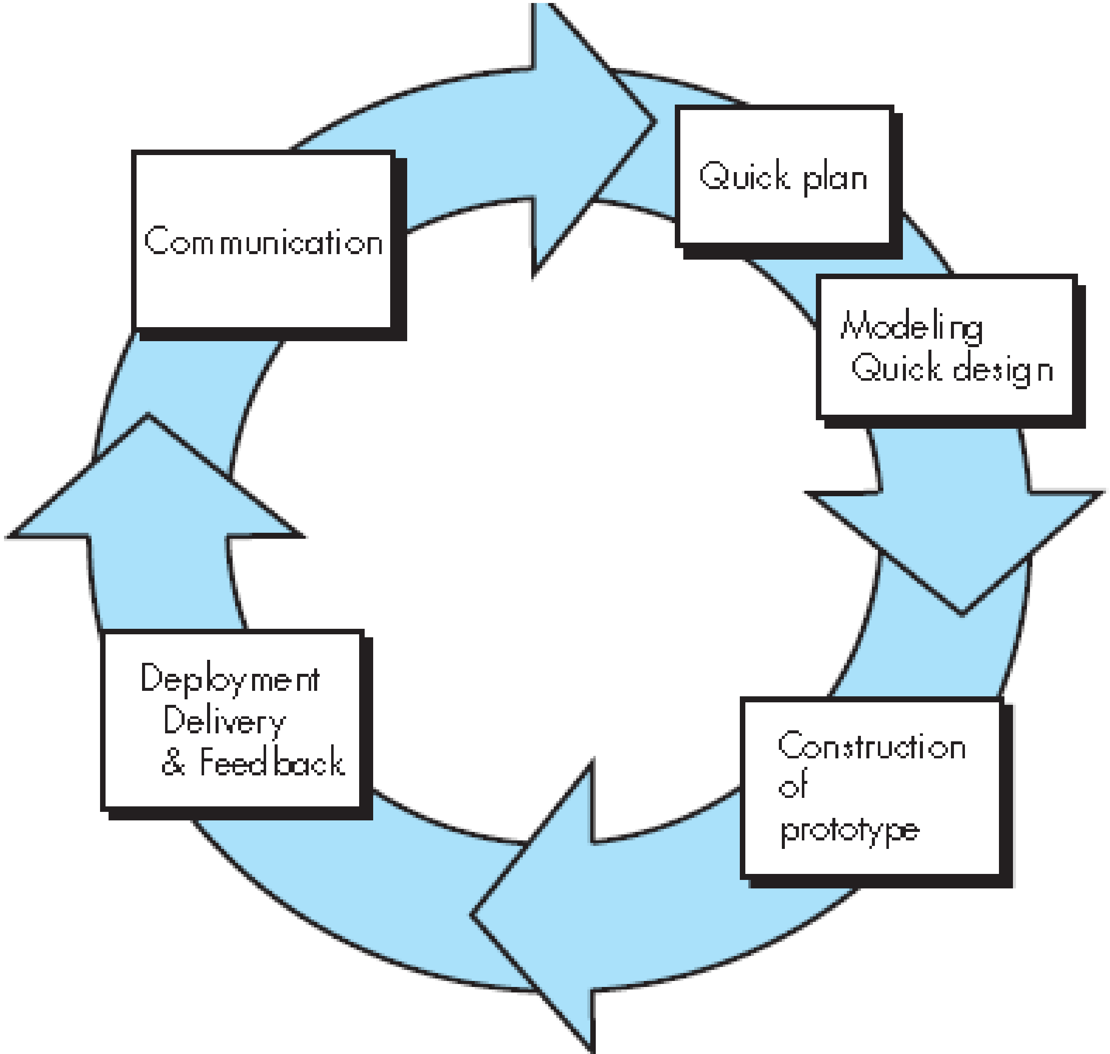
Component 3 is ready after third increment

Evolutionary Process Models

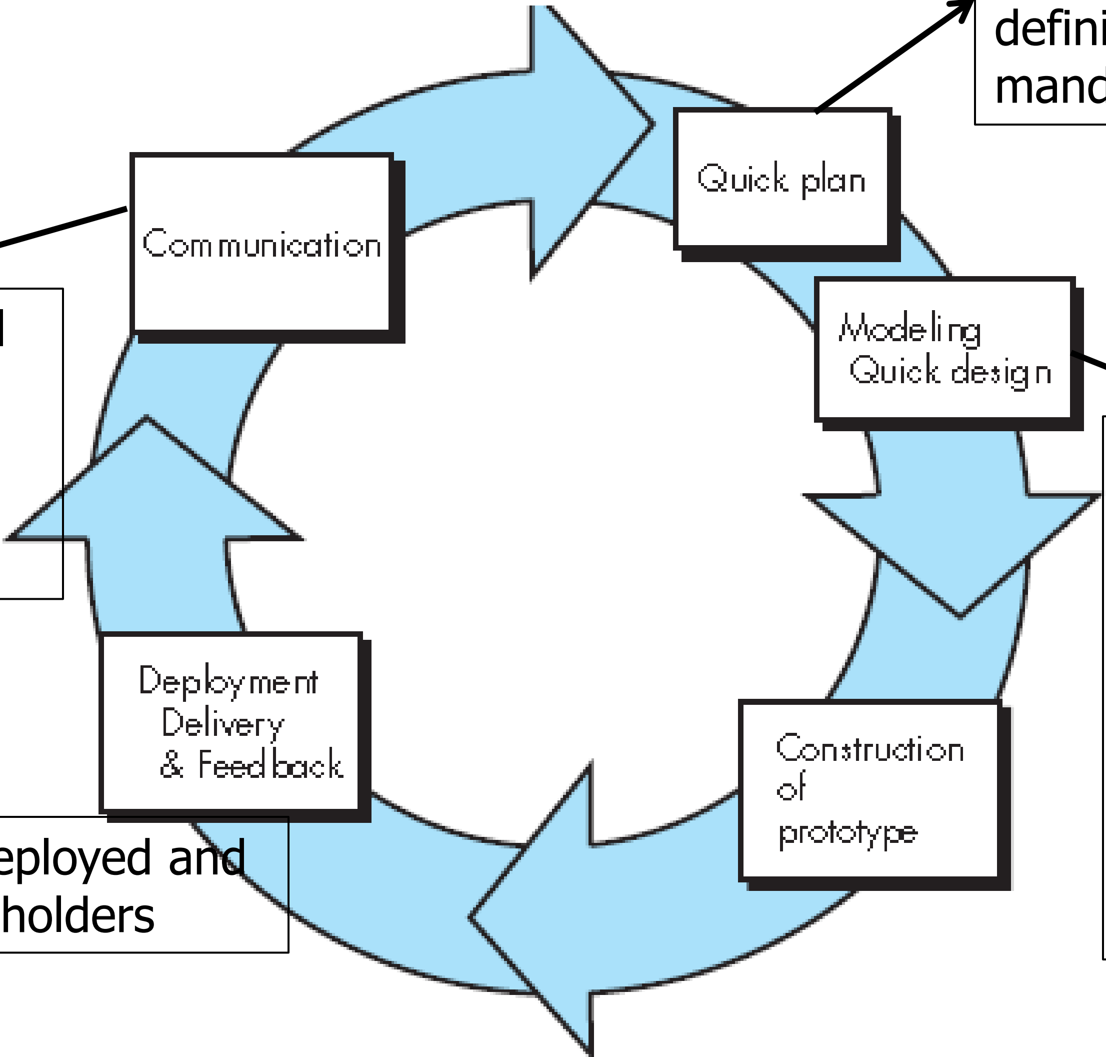
- Software, like all complex systems, evolves over a period of time.
- Business and product requirements often change.
  - Tight market deadlines that limit the completion.
- Evolutionary models are iterative.
- They are characterized in a manner that enables you to develop increasingly more complete versions of the software.
  - Prototyping
  - Spiral Model

Prototyping

- Customer
  - General Objectives but not detailed requirements.
- Developer
  - Unsure about the efficiency of algorithm.
  - Human- machine interaction.



Prototyping



Outline areas where further definition is mandatory

Define the overall objectives.

Identify requirements

Representation of those aspects of the software that will be visible to end users

e.g. output display formats

The prototype is deployed and
Evaluated by stakeholders

Problems in Prototyping

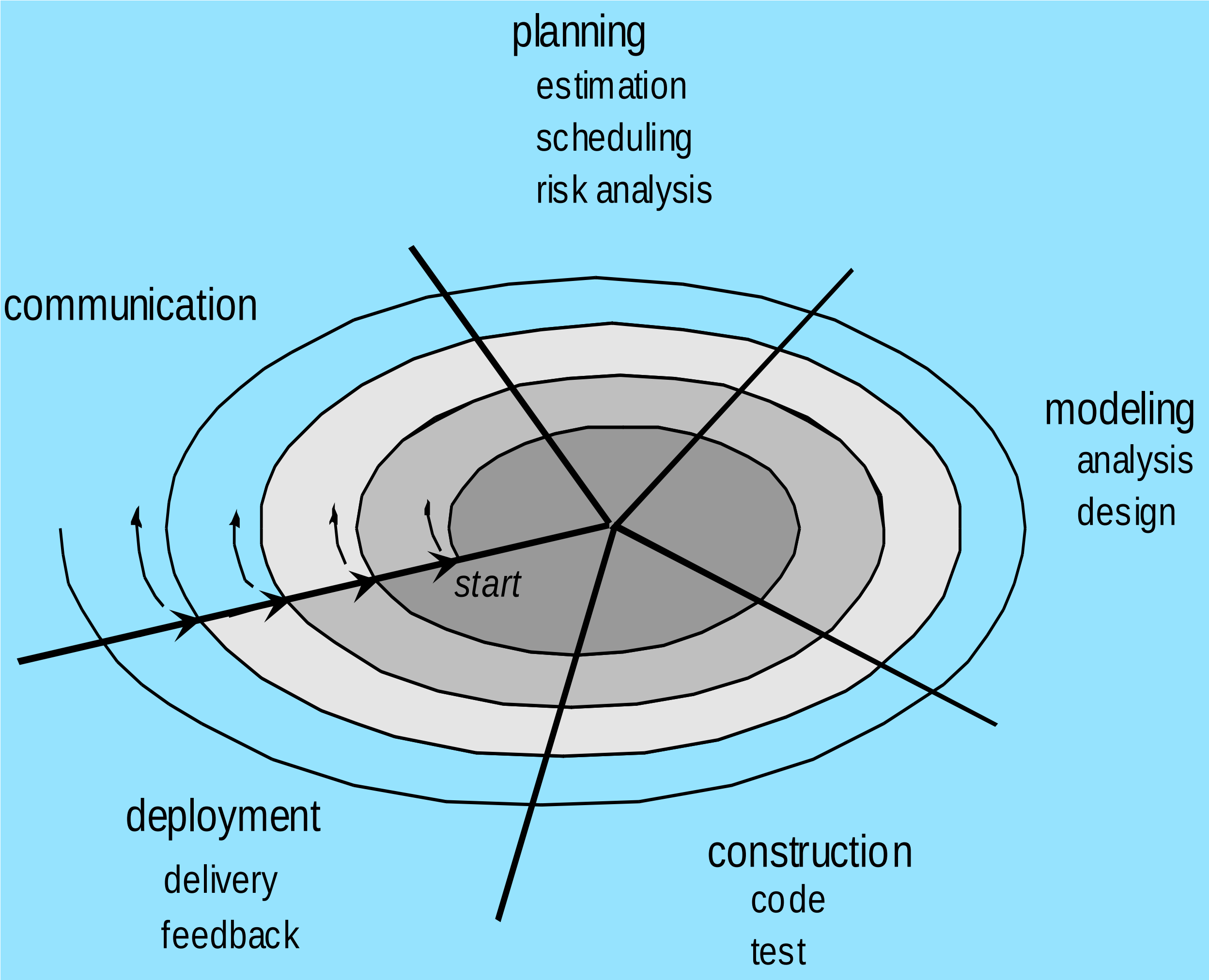
- Stakeholders see what appears to be a working version of the software, unaware that the prototype is held together haphazardly.
- Implementation compromises in order to get a prototype working quickly.
  - Inappropriate operating system or programming language may be used.

Spiral Model

- Proposed by Barry Boehm.
- *Spiral model* is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
- Boehm describes the model in the following manner:

The spiral development model is a *risk -driven process model generator that is used* to guide multi-stakeholder concurrent engineering of software intensive systems. It has two main distinguishing features. One is a *cyclic approach for incrementally* growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of *anchor point milestones for ensuring stakeholder* commitment to feasible and mutually satisfactory system solutions.

Spiral Model



planning

estimation

scheduling

risk analysis

communication

modeling

analysis

design

start

deployment

delivery

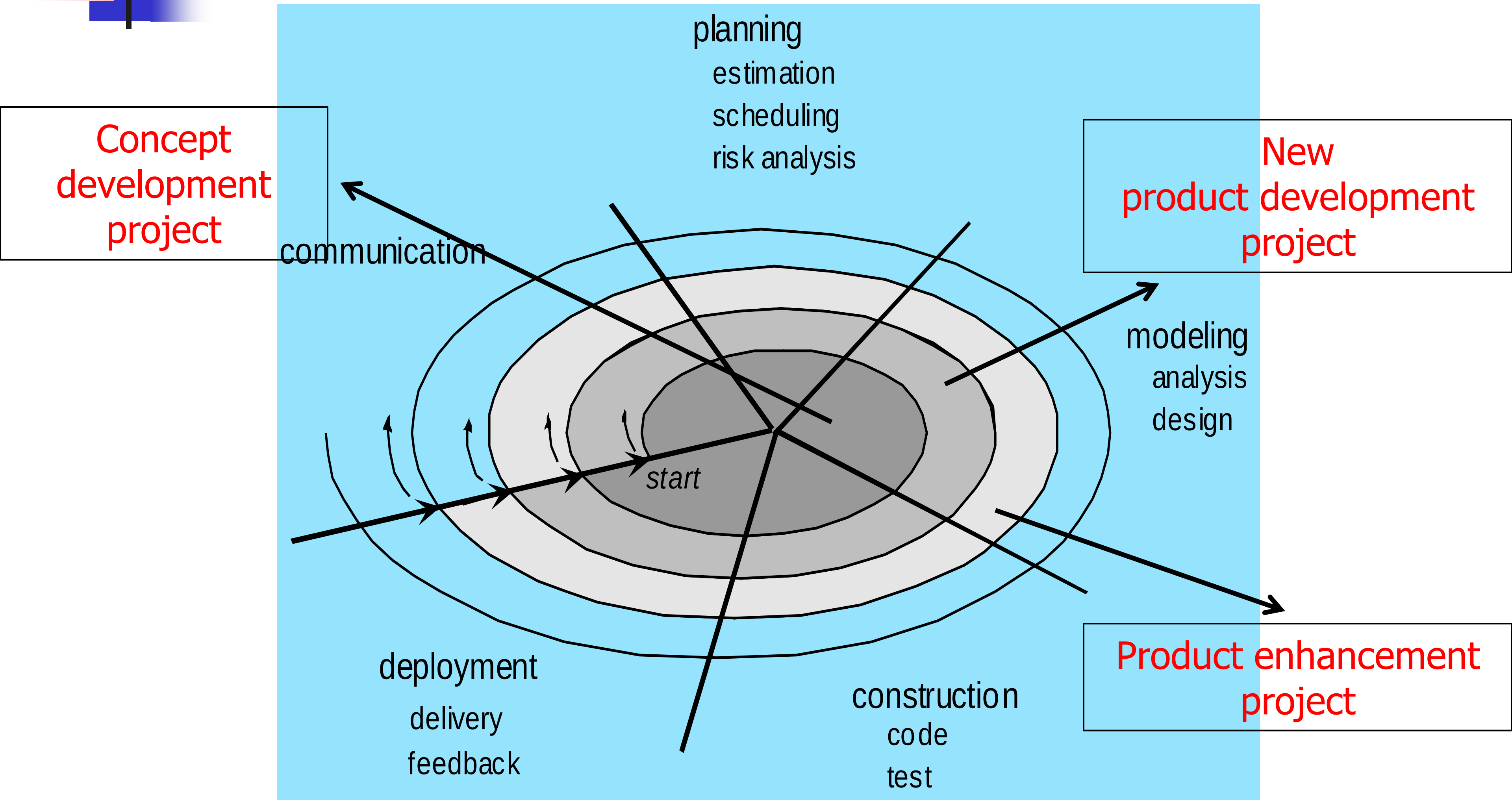
feedback

construction

code

test

Spiral Model



planning
estimation
scheduling
risk analysis

Concept development project

communication

New product development project

modeling
analysis
design

start

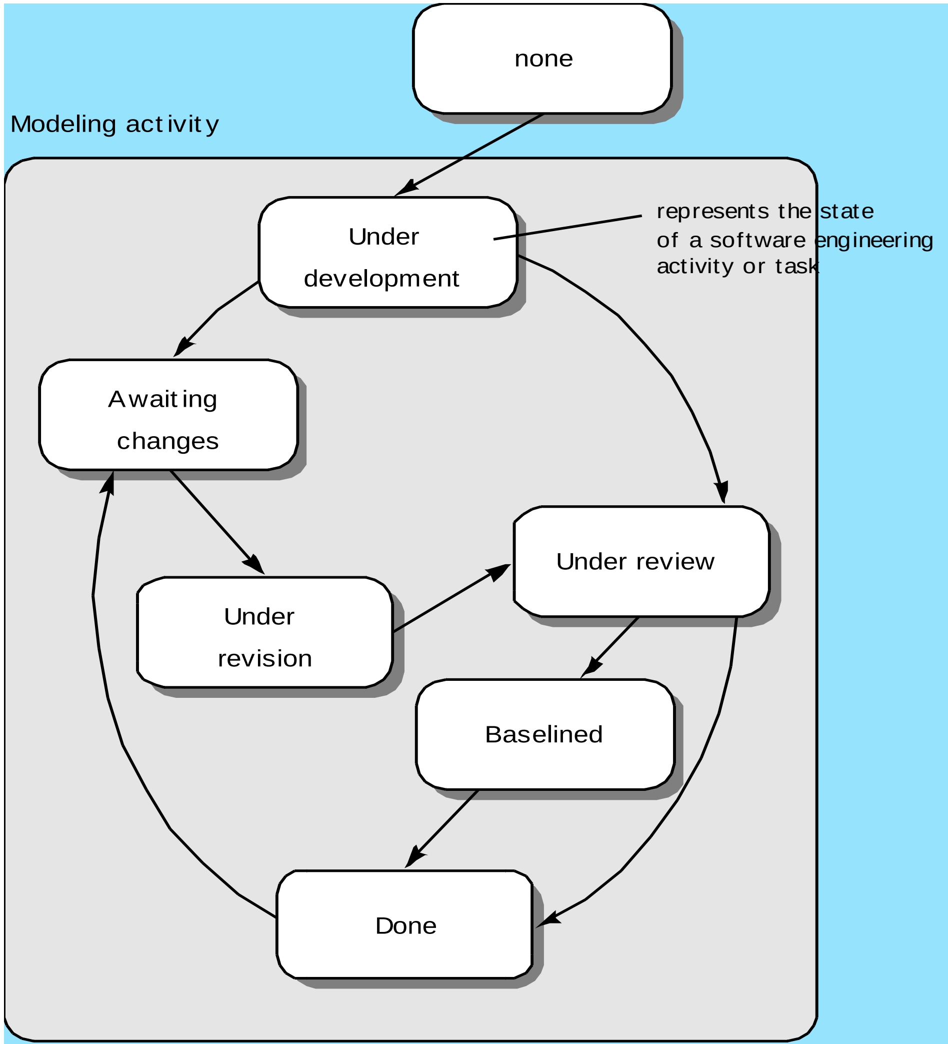
deployment
delivery
feedback

construction
code
test

Product enhancement project

Concurrent Models

- Allows a software team to represent iterative and concurrent elements of any of the process model.
- An activity— **modeling —may be in any one of the states noted at any given time.**
- Similarly, other activities, actions, or tasks (e.g., **communication or construction**) can be represented in an analogous manner.



Problems

- https://www.ics.uci.edu/~emilyo/SimSE/downloads.html

- Provide three examples of software projects that would be amenable to the prototyping model. Be specific.

Specialized Process Models

- Specialized process models take on many of the characteristics of one or more of the traditional models.
  - **Component-Based Development**
  - **The Formal Methods Model**
  - **Aspect-Oriented Software Development**

Component-Based Development

- The component-based development model incorporates many of the characteristics of the spiral model.
  - Evolutionary in nature
  - Iterative approach
  - Applications from prepackaged software components
- Commercial off-the-shelf (COTS) software components.
  - Provide targeted functionality with well-defined interfaces that enable the component to be integrated into the software.

Component-Based Development

- Component-based development model incorporates the following steps.
  - Available component-based products are researched and evaluated for the application domain in question.
  - Component integration issues are considered.
  - A software architecture is designed to accommodate the components.
  - Components are integrated into the architecture.
  - Comprehensive testing is conducted to ensure proper functionality.

Component-Based Development

- Advantages
  - Software reuse
  - Reduction in development cycle time
  - Reduction in project cost

Examples of COTS

- Microsoft Office is a COTS product that is a packaged software solution for businesses.
- Anti-virus programs
- Database managers
- Education/training software
- SAP
- Inventory management
- Payroll tax processing

The Formal Methods Model

- It encompasses a set of activities that leads to *formal mathematical specification* of computer software.
- A variation on this approach, is called *Cleanroom software engineering*.
- It enables to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.

The Formal Methods Model

- Advantages:
  - Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily.
  - Formal methods used during design, serve as a basis for program verification.
- Shortfalls:
  - Time consuming and expensive.
  - Extensive training is required.
  - Difficult to use the models as a communication mechanism

Safety Critical Systems

- Formal methods model are widely used in safety critical systems.
- Nuclear reactors, chemical plants, avionics, medical care, space missions, railway signaling, ...

TOO MANY DIGITS FOR ARIANE 5

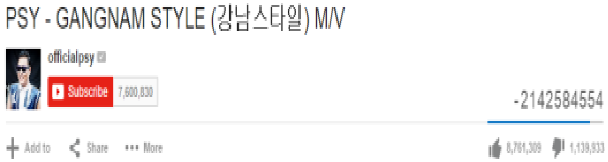
a 64-bit variable with decimals was transformed into a 16-bit variable without decimals.

A **16 bits** variable can have a value of −32.768 to 32.767.

On the other hand, a **64 bits** variable can have a value of −9.223.372.036.854.775.808 to 9.223.372.036.854.775.807 (that's almost an infinity of options).

Gangnam Style Broke Youtube

The maximum value for a 32-bit signed integer is 2, 147, 483, 647 and when a song Gangnam Style came to the views on this hit song by Korean pop star exceeded the maximum value



PSY - GANGNAM STYLE (강남스타일) M/V

officialpsy

-2142584554

Aspect-Oriented Software Development

- When developing a software, localized software characteristics are modeled as components (e.g., object-oriented classes) and then integrated into the whole system.
- Concerns:
  - Security, fault tolerance
  - Application of business rules
  - Task synchronization, memory management

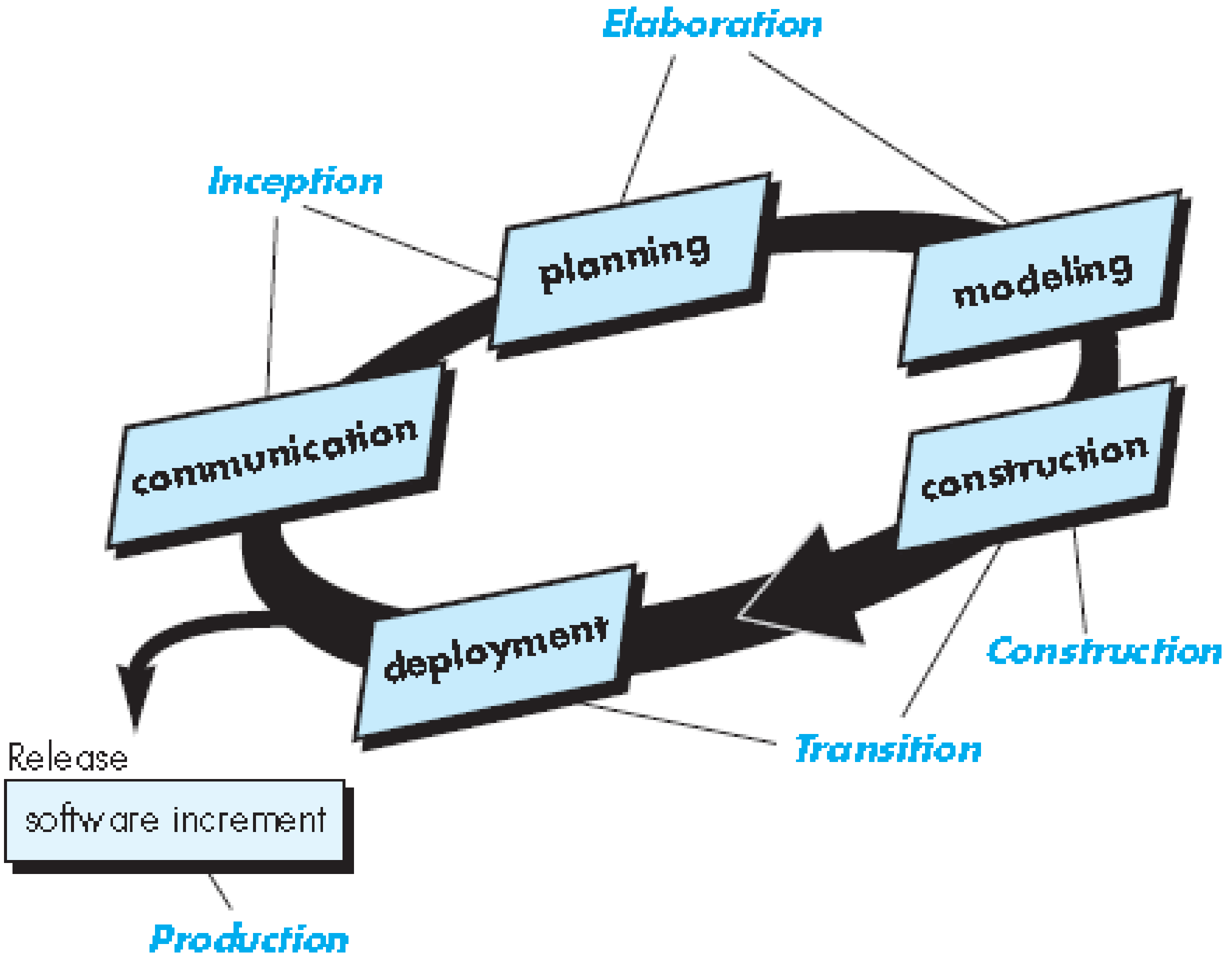
Aspect-Oriented Software Development

- *Aspectual requirements* define crosscutting concerns that have an impact across the software architecture.
- *Aspects* —"*mechanisms beyond subroutines and inheritance for* localizing the expression of a crosscutting concern"
- *Aspect-oriented software development* provides a process and methodological approach for defining, specifying, designing, and constructing *aspects.*

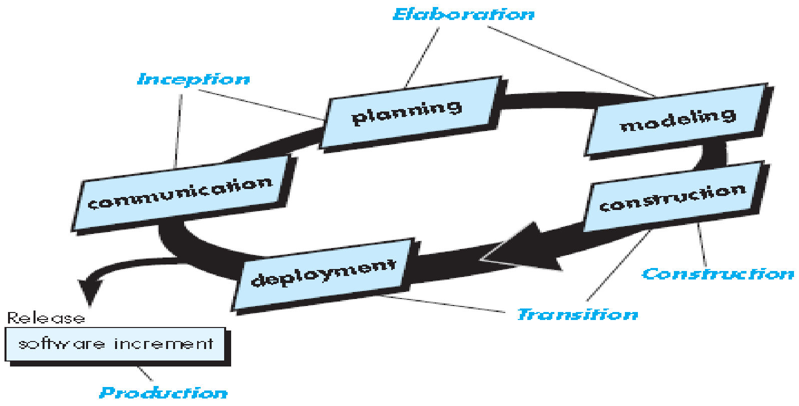
The Unified Process

- Unified Process is an attempt to draw on the best features and characteristics of traditional software process models.
- Streamlined methods for describing the customer's view of a system (the use case).
  - A use case is written by the user and serves as a basis for the creation of a more comprehensive analysis model.
- UML—a unified modeling language that contains a robust notation for the modeling and development of object- oriented systems.

The Unified Process

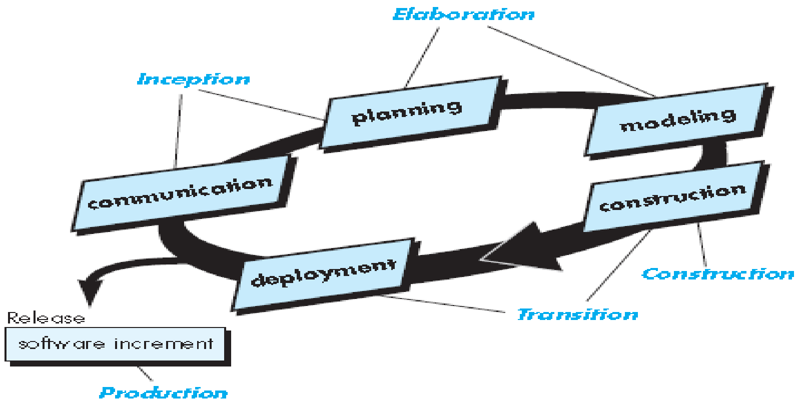


The Unified Process



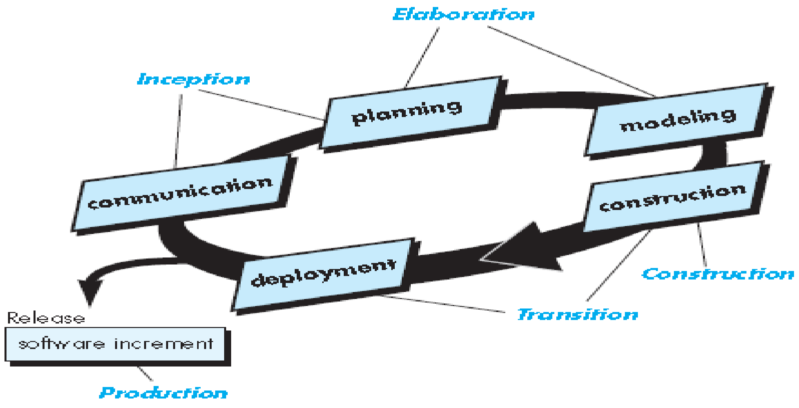
- Inception Phase:
  - Customer communication and planning activities.
  - A rough architecture for the system is proposed.
  - A set of preliminary use cases that describe which features and functions each major class of users desires.
  - Planning identifies resources, assesses major risks, defines a schedule, and establishes a basis for the phases that are to be applied as the software increment is developed.

The Unified Process



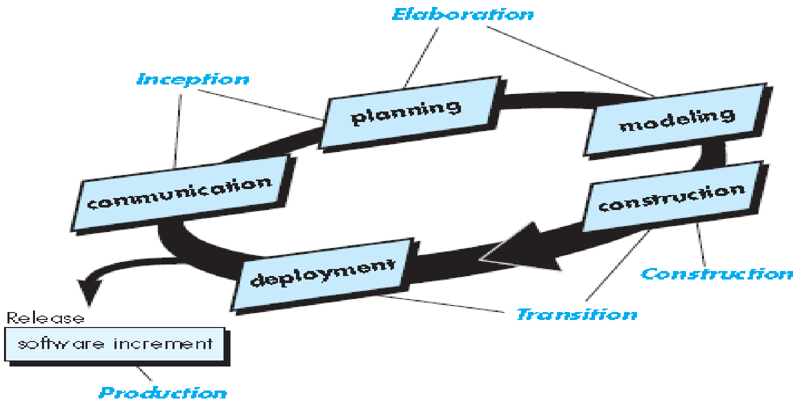
- Elaboration phase:
- It refines and expands the preliminary use cases.
- Expands the architectural representation to include five different views of the software—the use case model, the analysis model, the design model, the implementation model, and the deployment model.

The Unified Process



- Construction phase:
- Uses the architectural model as input
- In this phase, the software components are developed or acquired that will make each use case operational for end users.
- Components are implemented, unit tests are designed and executed for each.

The Unified Process

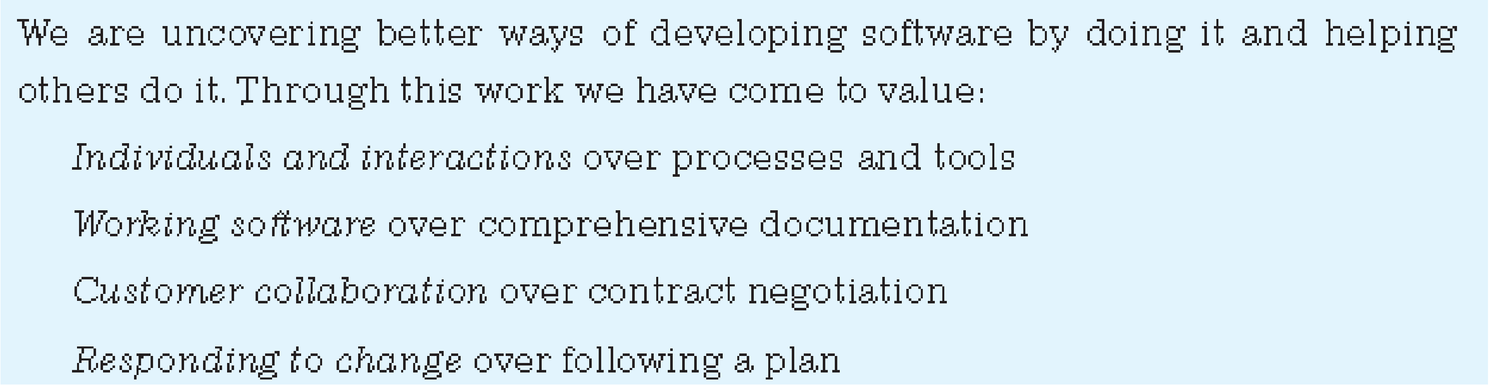


- Transition phase:
- Delivery and feedback activities.
- Software is given to end users for beta testing, and user feedback reports both defects and necessary changes.
- Necessary support information
  - e.g., user manuals, troubleshooting guides, installation procedures
- Production phase:
  - The ongoing use of the software is monitored, support for the operating environment (infrastructure) is provided, and defect reports and requests for changes are submitted and evaluated.

AGILE DEVELOPMENT

What Is Agility ?

- An agile team is a nimble team able to appropriately respond to changes.
- The pervasiveness of change is the primary driver for agility.
- Agile Manifesto



We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

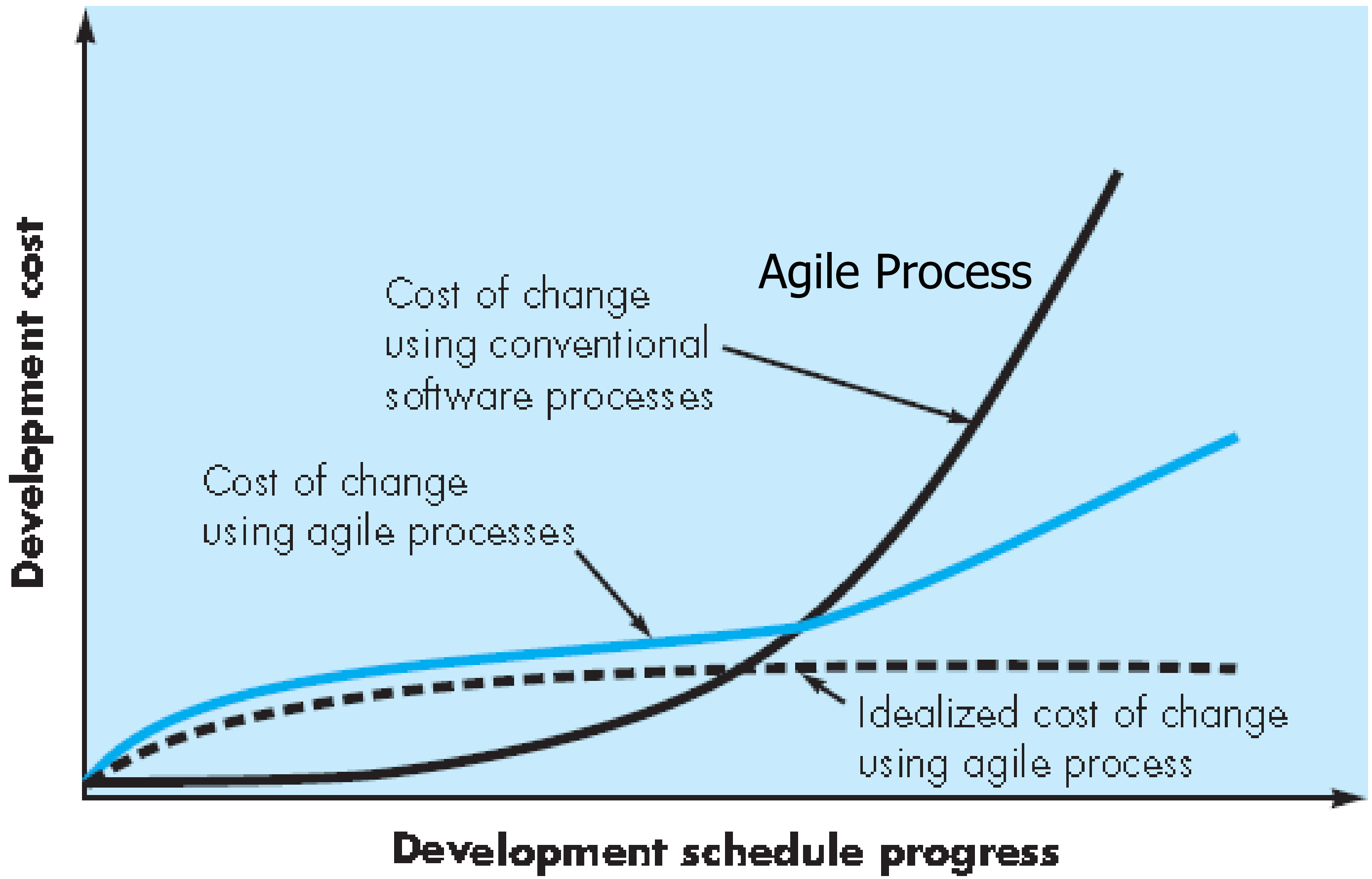
Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Agility and the Cost of Change



KEY POINT

An agile process reduces the cost of change because software is released in increments and change can be better controlled within an increment.

Agile Process

- Basic Assumptions of a Software Project
  - It is difficult to predict in advance which software requirements will persist and which will change.
  - It is difficult to predict how much design is necessary before construction is used to prove the design.
- How do we create a process that is agile and can manage *unpredictability ?*

Agile Process

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Delivers multiple ‘software increments’
- Adapts as changes occur

Agility Principles

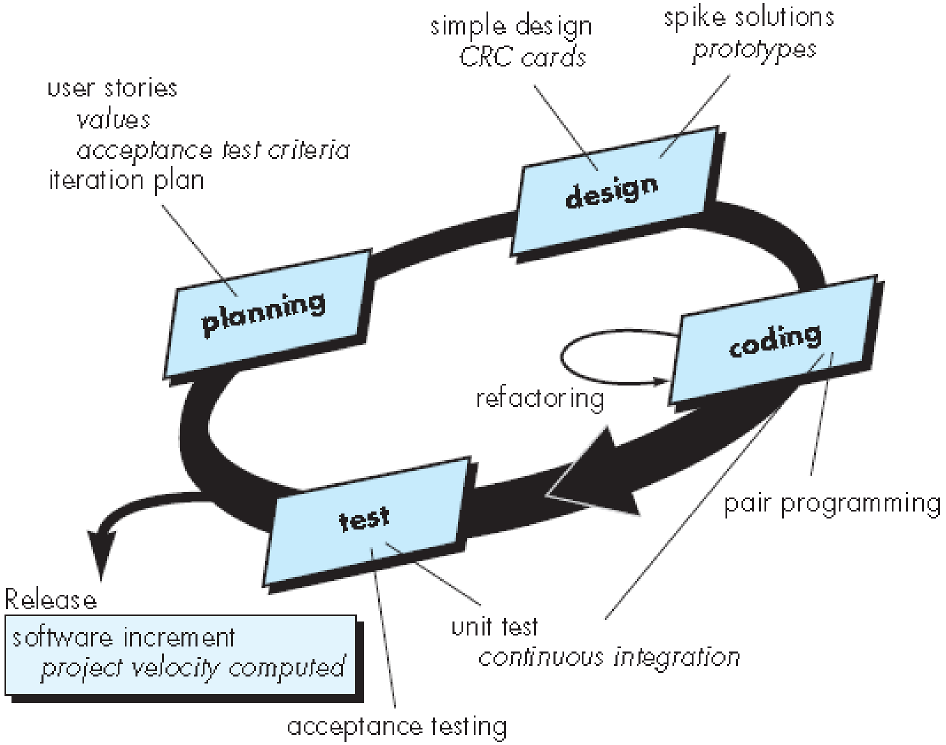
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face–to–face conversation.

Agility Principles

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self–organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Extreme Programming

- *Extreme Programming (XP), the most widely used approach to* agile software development.
- **XP Process**



XP Process

- XP Planning
  - Begins with the creation of "user stories"
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment "project velocity" is used to help define subsequent delivery dates for other increments

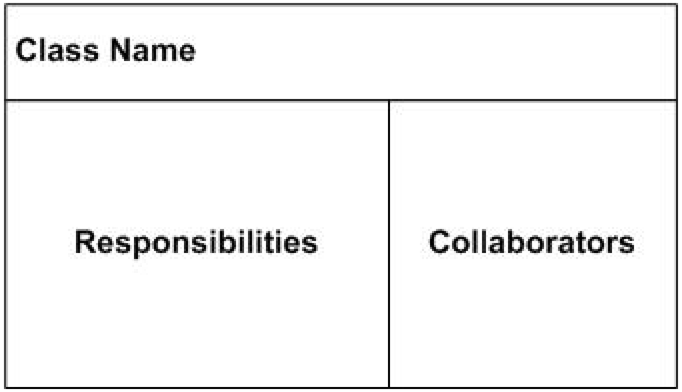
User story Example

- User Story - Registration Screen - Username and Password creation
  - User Story:
    As a customer, I want to create login credentials so I can securely access my self-service online account
  - Acceptance Criteria:
    Input data fields to enter - 1) Username 2) Password 3) Re-enter Password 4) Security question 5) Security answer
  - Username data field specifications - TBD for specifications
  - Password data field specifications - At least 7 characters, 1 number, 1 uppercase letter, 1 lowercase letter, one special character
  - Security questions drop-down - TBD for content
  - Security Answer data field specifications - at least 4 characters

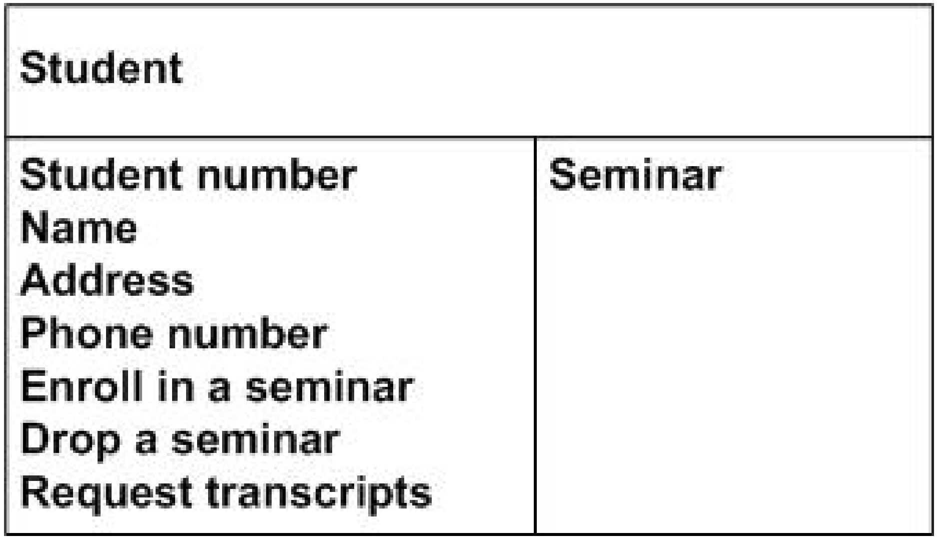
XP Process

- XP Design
  - Encourage the use of CRC (Class-responsibility-collaboration) cards
  - For difficult design problems, suggests the creation of “spike solutions”—a design prototype
  - Encourages “refactoring”—an iterative refinement of the internal program design

CRC Card Example



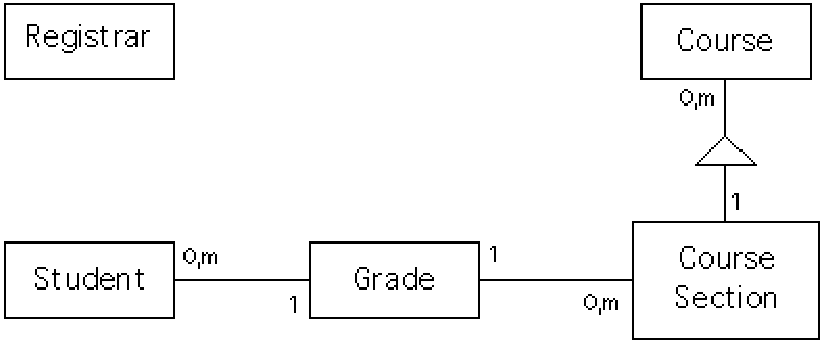
Class Name	
Responsibilities	Collaborators



Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

CRC Card Example

- Student Information System



class name: Course

class type: (e.g., device property, role, event...) Structure

class characteristics (e.g., tangible, atomic, concurrent...)
Abstract; sequential; persistent; guarded

responsibilities:	collaborators:
Display Courses Menu	
Hide Courses Menu	
Enable Courses Menu	
Disable Courses Menu	
Create New Course	
Update Course Information	
List All Courses	
Display Course Information	
Delete Course	Course Section
Select Existing Course	
Get Course	

XP Process

- XP Coding
  - Recommends the construction of a unit test for a store *before* coding commences
  - Encourages "pair programming"

We already used that variable name for a different variable.

XP Process

- XP Testing
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionality

User story

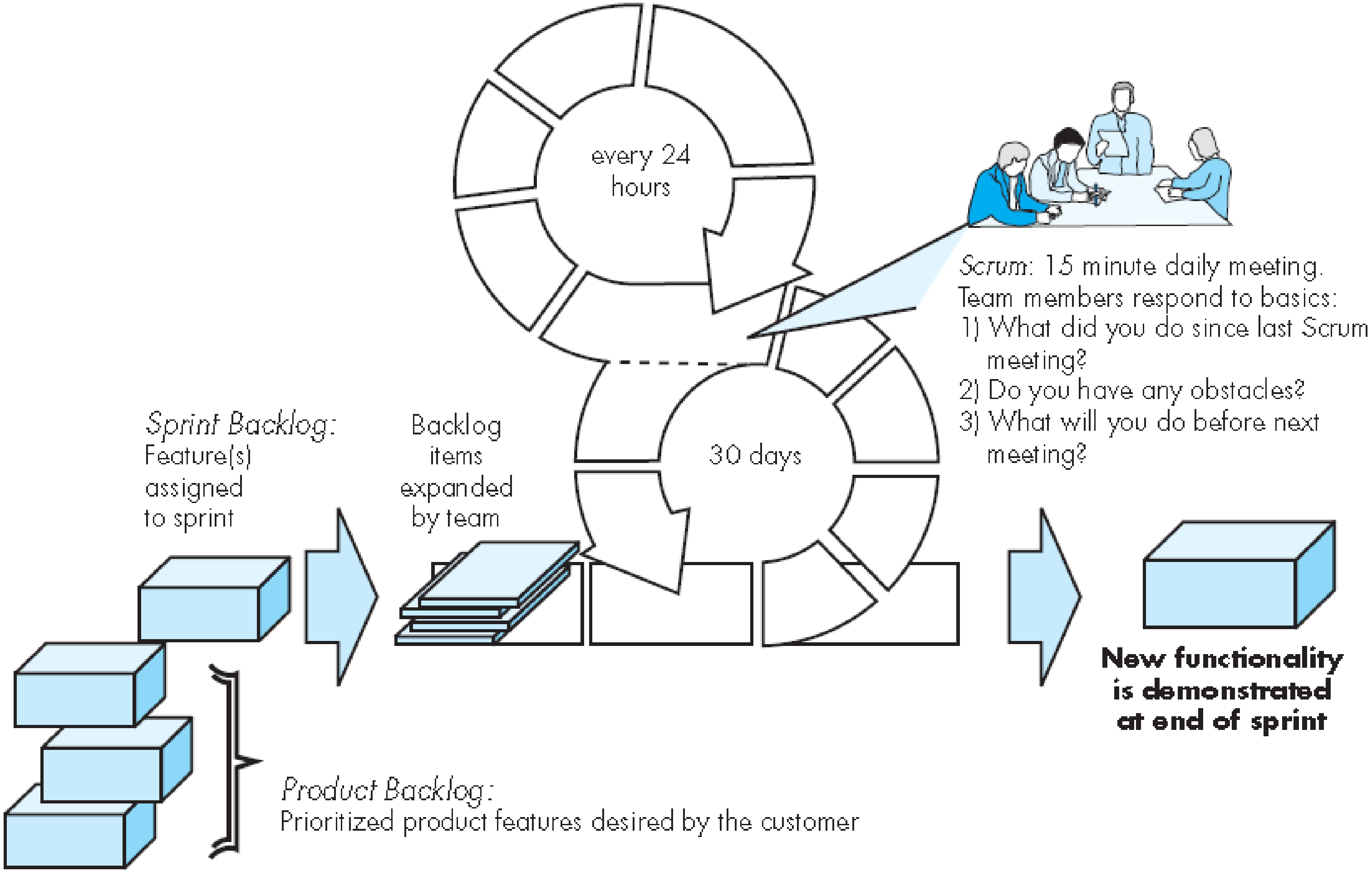
- Write an XP user story that describes the “favorite places” or “favorites” feature available on most Web browsers.
- Create a CRC card for the same.

Other Agile Process Models

Scrum

- Scrum principles are consistent with the agile manifesto
- It is used to guide development activities within a process that incorporates the following framework activities: requirements, analysis, design, evolution, and delivery.
- Work tasks occur within a process pattern called a *sprint*.

Scrum Process


every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Scrum Process

- *Backlog —a prioritized list of project requirements or features that provide* business value for the customer.
- The product manager assesses the backlog and updates priorities as required.

Scrum Process

- *Sprints —consist of work units that are required to achieve a requirement* defined in the backlog that must be fit into a predefined time-box 10 (typically 30 days).
- Changes (e.g., backlog work items) are not introduced during the sprint.

Scrum Process

- *Scrum meetings —are short (typically 15-minute) meetings held daily by the*
- Scrum team. Three key questions are asked and answered by all team members
  - What did you do since the last team meeting?
  - What obstacles are you encountering?
  - What do you plan to accomplish by the next team meeting?
- A team leader, called a *Scrum master,* leads the meeting and assesses the responses from each person.

Scrum Process

- *Demos —deliver the software increment to the customer so that functionality* that has been implemented can be demonstrated and evaluated by the customer.

Dynamic Systems Development Method (DSDM)

- It "provides a framework for building and maintaining systems which meet tight time constraints through the use of incremental prototyping in a controlled project environment".
- DSDM is an iterative software process in which each iteration follows the 80 percent rule.
- That is, only enough work is required for each increment to facilitate movement to the next increment. The remaining detail can be completed later.

Dynamic Systems Development Method (DSDM)

- DSDM life cycle
- *Feasibility study* that establishes basic business requirements and constraints and is followed by a *business study* that identifies functional and information requirements.
- *Functional model iteration*— produces a set of incremental prototypes that demonstrate functionality for the customer.

Dynamic Systems Development Method (DSDM)

- DSDM life cycle
- *Design and build iteration*— revisits prototypes built during the functional model iteration to ensure that each has been engineered in a manner that will enable it to provide operational business value for end users.
- *Implementation*— places the latest software increment (an "operationalized" prototype) into the operational environment.

Agile Modeling

- Business-critical systems
- (1) all constituencies can better understand what needs to be accomplished.
- (2) the problem can be partitioned effectively among the people who must solve it.
- (3) quality can be assessed as the system is being engineered and built.

Agile Modeling principles

- **Model with a purpose**
- A developer who uses AM should have a specific goal
  - (e.g., to communicate information to the customer or to help better understand some aspect of the software) in mind before creating the model.
- **Use multiple models**
  - AM suggests that to provide needed insight, each model should present a different aspect of the system and only those models that provide value to their intended audience should be used.

Agile Modeling principles

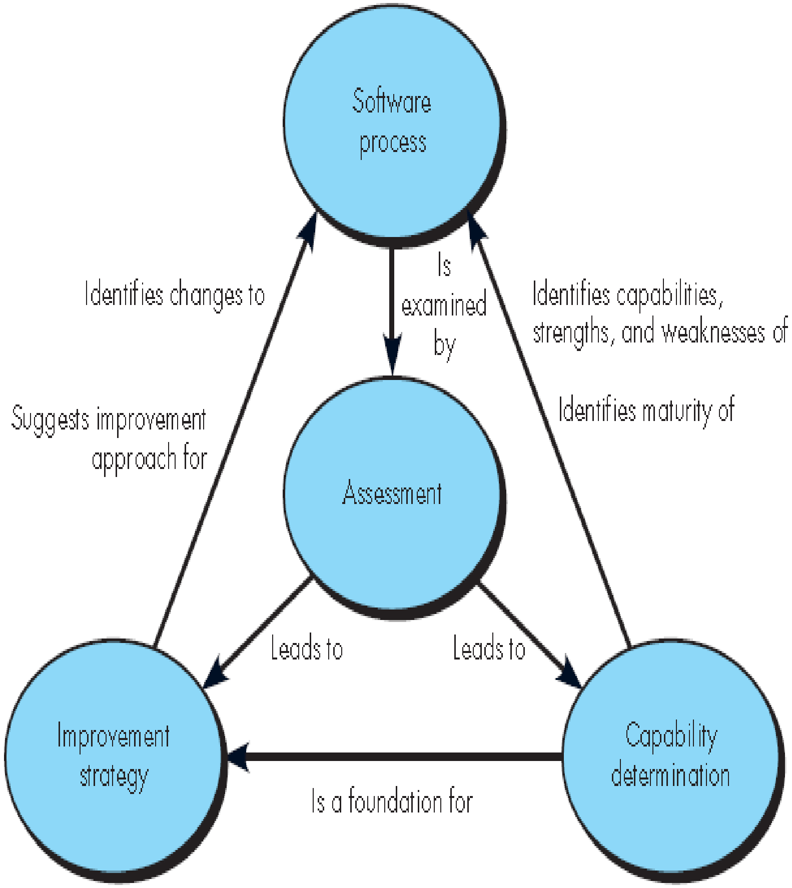
- **Travel light.**
  - As software engineering work proceeds, keep only those models that will provide long-term value and jettison the rest.
- **Content is more important than representation.**
  - **Modeling should impart information** to its intended audience.
- **Know the models and the tools you use to create them.**
  - **Understand the** strengths and weaknesses of each model and the tools that are used to create it.
- **Adapt locally.**
  - **The modeling approach should be adapted to the needs of the** agile team.

SPI (Software Process Improvement)

- It implies that elements of an effective software process can be defined in an effective manner.
- An existing organizational approach to software development can be assessed against those elements.
- A meaningful strategy for improvement can be defined.

SPI (Software Process Improvement)

- An *SPI framework defi*nes
- (1) a set of characteristics that must be present if an effective software process is to be achieved,
- (2) a method for assessing whether those characteristics are present,
- (3) a mechanism for summarizing the results of any assessment
- (4) a strategy for assisting a software organization in implementing those process characteristics that have been found to be weak or missing.



The SPI Process

- **Assessment and Gap Analysis**
  - It examines a wide range of actions and tasks that will lead to a high-quality process.
- Is the objective of the activity clearly defined?
- Are work products required as input and produced as output identified and described?
- Are the work tasks to be performed clearly described?
- Are the people who must perform the activity identified by role?
- Have entry and exit criteria been established?
- Have metrics for the activity been established?
- Are tools available to support the activity?
- Is there an explicit training program that addresses the activity?
- Is the activity performed uniformly for all projects?

The SPI Process

- **Education and Training**
- It follows that a key element of any SPI strategy is education and training for practitioners, technical managers, and more senior managers who have direct contact with the software organization.
- Three types of education and training should be conducted:
  - generic software engineering
  - concepts and methods,
  - specific technology and tools, and communication and
  - quality-oriented topics.

The SPI Process

- **Selection and Justification**
- Process characteristics and specific software engineering methods and tools are chosen to populate the software process.
- Choose the process model
- Decide the set of framework activities
- Major work products that will be produced

The SPI Process

- **Installation/Migration**
  - *Installation is the first point at which a software organization feels the effects of* changes implemented as a consequence of the SPI road map.
  - An incremental *migration from one process (that doesn't work as* well as desired) to another process is a more effective strategy.

The SPI Process

- **Evaluation**
  - Assesses the degree to which changes have been instantiated and adopted, the degree to which such changes result in better software quality.
- **Risk Management for SPI**
  - Experience with quality programs, level of success
  - Patience with change; ability to spend time socializing
  - Tools orientation—expectation that tools can solve the problems

Capability Maturity Model(CMMI)

- A comprehensive process meta-model that is predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity.
- Capability levels:
  - **Level 0:** ***Incomplete***
  - **Level 1:** ***Performed***
  - **Level 2:** ***Managed***
  - **Level 3:** ***Defined***
  - **Level 4:** ***Quantitatively managed***
  - **Level 5:** ***Optimized***

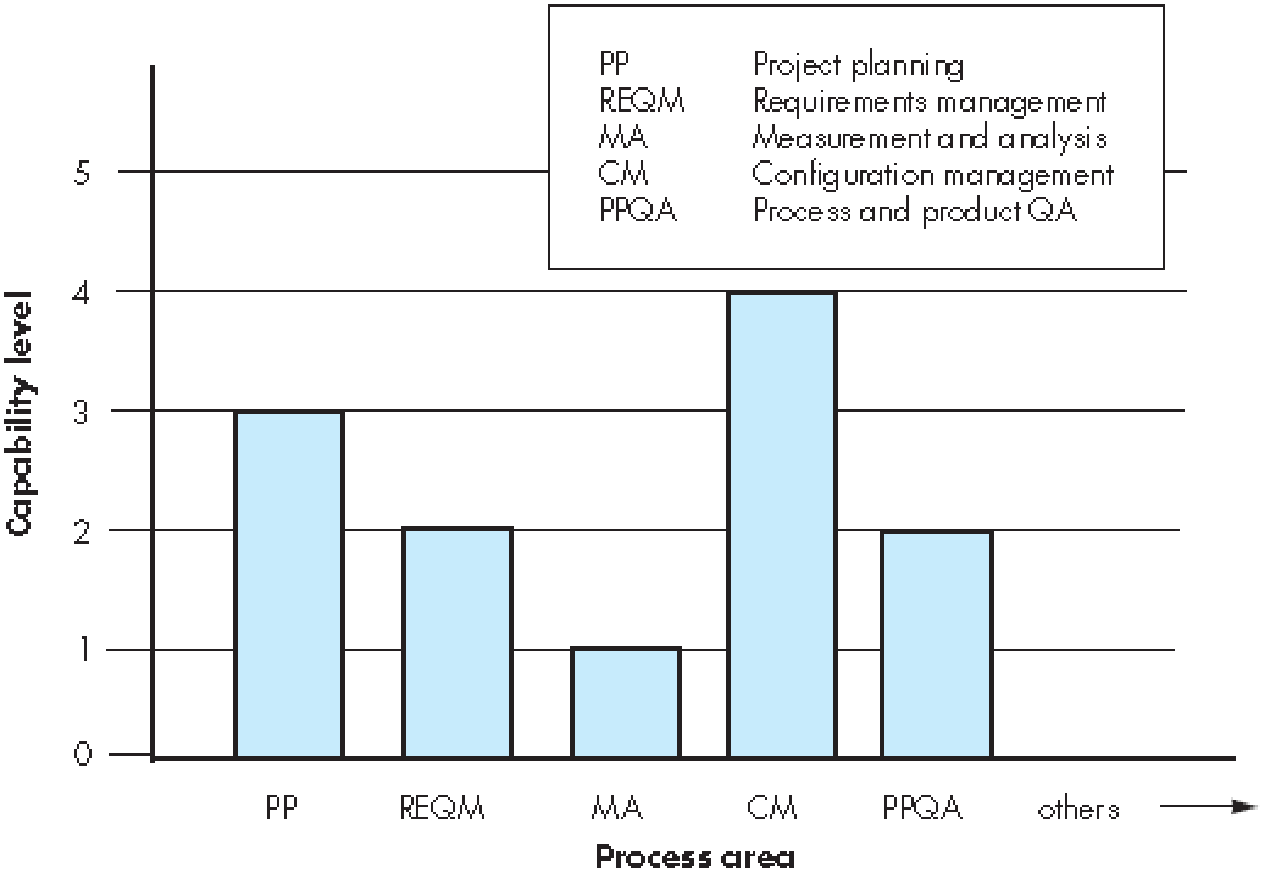
Capability Maturity Model(CMMI)

- **Level 0: *Incomplete* —**
  - The process area (e.g., requirements management) is either not performed or does not achieve all goals and objectives defined.
- ***Level 1: Performed* —**
  - All of the specific goals of the process area (as defined by the CMMI) have been satisfied.
- ***Level 2: Managed* —**
  - All capability level 1 criteria have been satisfied. In addition, all work associated with the process area conforms to an organizationally defined policy.

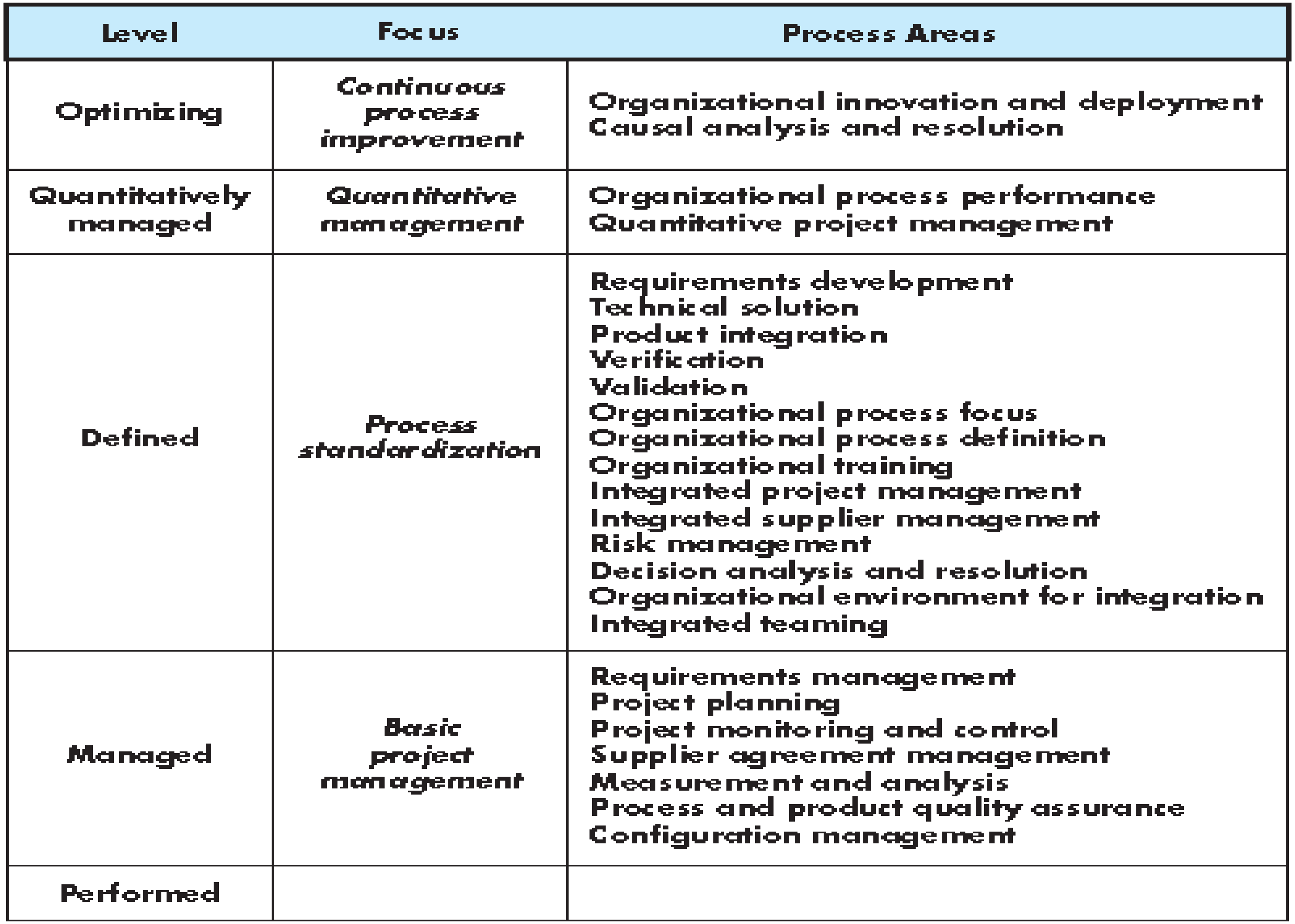
Capability Maturity Model(CMMI)

- **Level 3: *Defined* —**
  - All capability level 2 criteria have been achieved. In addition, the process is “tailored from the organization’s set of standard processes according to the organization’s tailoring guidelines, and contributes work products.
- **Level 4: Quantitatively managed**
  - All capability level 3 criteria have been achieved.
- **Level 5: Optimized —**
  - All capability level 4 criteria have been achieved.
  - In addition, the process area is adapted and optimized using quantitative (statistical) means to meet changing customer needs

Capability Maturity Model(CMMI)



CMMI and Process Areas



Level	Focus	Process Areas
Optimizing	*Continuous process improvement*	Organizational innovation and deployment Causal analysis and resolution
Quantitatively managed	*Quantitative management*	Organizational process performance Quantitative project management
Defined	*Process standardization*	Requirements development Technical solution Product integration Verification Validation Organizational process focus Organizational process definition Organizational training Integrated project management Integrated supplier management Risk management Decision analysis and resolution Organizational environment for integration Integrated teaming
Managed	*Basic project management*	Requirements management Project planning Project monitoring and control Supplier agreement management Measurement and analysis Process and product quality assurance Configuration management
Performed		

Review Questions

- What is the simplest model of software development paradigm?
- **a.**Spiral model
- **b.**Big Bang model
- **c.**V-model
- **d.**Waterfall model

Review Questions

- Programs, documents, and data produced are __________ .

a. Product

b. Work product

c. Tools

d. Methods

Review Questions

- Software is developed or engineered (by following systematic, disciplined and quantifiable approach), it is not manufactured.
- True
- False

Review Questions

- Spiral model is suitable for student projects.
- True
- False

Review Questions

- Prototyping model and spiral model follows generalize approach of ____________________

a. Waterfall Model (Linear sequential model)

b. Incremental Model

Review Questions

- ________________ model follows "quick design" approach.

a. Linear Sequential Model

b. Prototype Model

c. Spiral Model

d. Incremental model

Questions

- You work for a very small software organization—only 11 people are involved in developing software. Is SPI for you? Explain your answer.