

UNIT - 1

21CSC303J- Software Engineering Project Management (M.Tech Integrated)

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

SOFTWARE ENGINEERING

- Some realities:
 - *to understand the problem before a software solution is developed*
 - *design becomes a essential activity*
 - *software should exhibit high quality*
 - *software should be maintainable*

SOFTWARE ENGINEERING

- The IEEE definition:
 - *Software Engineering: The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software*

THE EVOLVING ROLE OF SOFTWARE

- Software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product
- As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware
- As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments)

Changing Nature of Software

1. System software: Infrastructure software come under this category like compilers, operating systems, editors, drivers, etc. Basically system software is a collection of programs to provide service to other programs.

2. Real time software: These software are used to monitor, control and analyze real world events as they occur. An example may be software required for weather forecasting. Such software will gather and process the status of temperature, humidity and other environmental parameters to forecast the weather.

Changing Nature of Software

3. Embedded software: This type of software is placed in “Read-Only-Memory (ROM)” of the product and control the various functions of the product. The product could be an aircraft, automobile, security system, signalling system, control unit of power plants, etc. The embedded software handles hardware components and is also termed as intelligent software .

4. Business software : This is the largest application area. The software designed to process business applications is called business software. Business software could be payroll, file monitoring system, employee management, account management. It may also be a data warehousing tool which helps us to take decisions based on available data.

Management information system, enterprise resource planning (ERP) and such other software are popular examples of business software

Changing Nature of Software

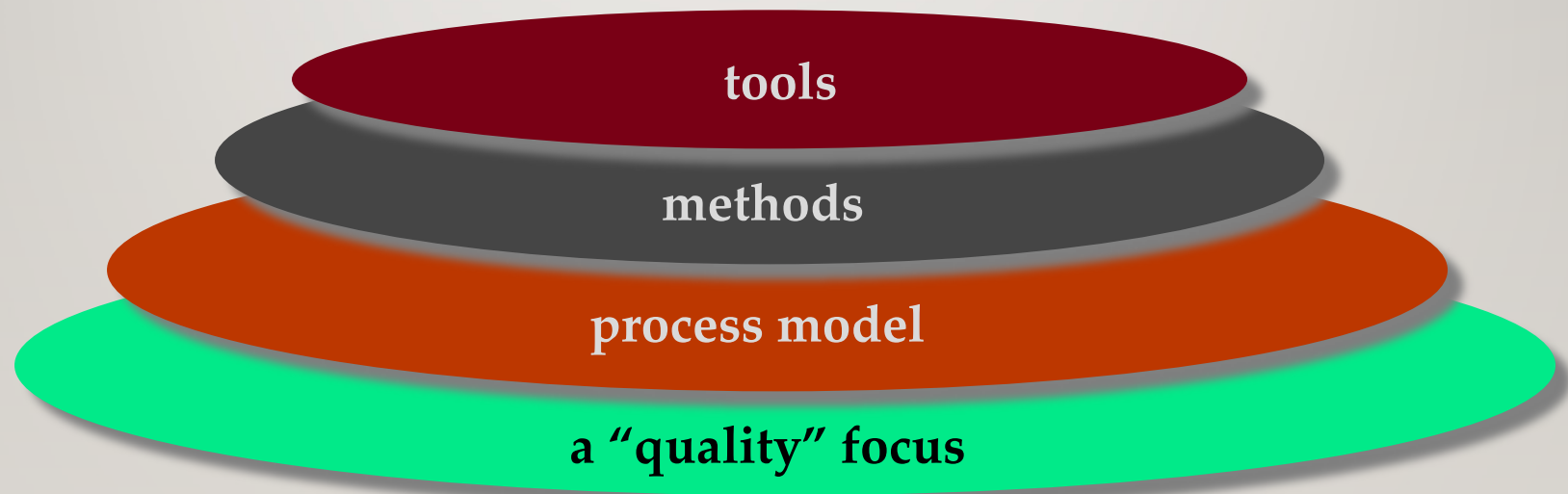
5. Personal computer software :The software used in personal computers are covered in this category. Examples are word processors, computer graphics, multimedia and animating tools, database management, computer games etc. This is a very upcoming area and many big organisations are concentrating their effort here due to large customer base.

6. Artificial intelligence software: Artificial Intelligence software makes use of non numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis. Examples are expert systems, artificial neural network,signal processing software etc.

Changing Nature of Software

8. Web based software: The software related to web applications come under this category. Examples are CGI, HTML, Java, Perl, DHTML etc.

A LAYERED TECHNOLOGY



*Software
Engineering*

-
- Software engineering is a **layered technology**
 - The foundation for software engineering is the *process layer*
 - **Process** defines a **framework** that must be established for effective delivery of software
 - 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.
 - Software engineering tools provide automated or semi automated support for the process and the methods.

THE ESSENCE OF PRACTICE

- Polya suggests:
 1. *Understand the problem* (communication and analysis).
 2. *Plan a solution* (modeling and software design).
 3. *Carry out the plan* (code generation).
 4. *Examine the result for accuracy* (testing and quality assurance).

UNDERSTAND THE PROBLEM

- who are the stakeholders?
- *What are the unknowns?* What data, functions, and features are required to properly solve the problem?
- *Can the problem be compartmentalized?* Is it possible to represent smaller problems that may be easier to understand?
- *Can the problem be represented graphically?* Can an analysis model be created?

PLAN THE SOLUTION

- *Have you seen similar problems before?* Are there patterns that are familiar in a potential solution? Is there existing software that implements the data, functions, and features that are required?
- *Has a similar problem been solved?* If so, are elements of the solution reusable?
- *Can subproblems be defined?* If so, are solutions readily apparent for the subproblems?
- *Can you represent a solution in a manner that leads to effective implementation?* Can a design model be created?

CARRY OUT THE PLAN

- *Does the solution conform to the plan?* Is source code traceable to the design model?
- *Is each component part of the solution provably correct?*
Has the design and code been reviewed, or better?

EXAMINE THE RESULT

- *Is it possible to test each component part of the solution?*
Has a reasonable testing strategy been implemented?
- *Does the solution produce results that conform to the data, functions, and features that are required?* Has the software been validated against all stakeholder requirements?

HOW IT ALL STARTS

- *SafeHome:*
 - Every software project is precipitated by some business need —
 - the need to **correct a defect** in an existing application;
 - the need to the need **to adapt** a 'legacy system' to a changing business environment;
 - the need **to extend** the functions and features of an existing application, or
 - the need **to create** a new product, service, or system.

Process Models

WHAT / WHO / WHY IS PROCESS MODELS?

- **What** : a series of predictable steps--- a road map that helps you create a timely, high-quality results.
- **Who** : Software engineers and their managers, clients also
- **Why** : Provides stability, control, and organization to an activity that can if left uncontrolled, become quite chaotic.
- **What Work products** : Programs, documents, and data

WHAT / WHO / WHY IS PROCESS MODELS?

- **What are the steps** : The process you adopt depends on the software that you are building. One process might be good for aircraft avionic system, while an entirely different process would be used for website creation.
- **How to ensure right:** A number of software process assessment mechanisms can be used to determine the maturity of the software process. However, the quality, timeliness and long-term feasibility of the software are the best indicators of the efficacy of the process you use.

DEFINITION OF 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*
 - An *action* encompasses a set of tasks that produce a major work product
 - A *task* focuses on a small, but well-defined objective that produces a tangible outcome

PROCESS FRAMEWORK(Phases)

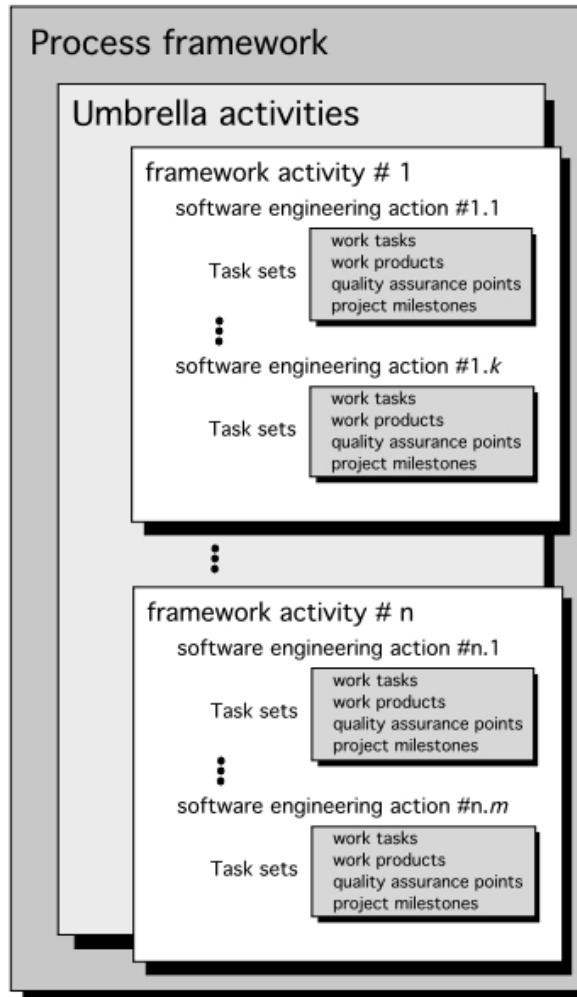
- Communication
- Planning
- Modeling
- Construction
- Deployment

Umbrella activities

e.g. Risk management, Measurement, Reviews, SCM

A GENERIC PROCESS MODEL

Software process



A GENERIC PROCESS MODEL

- a generic process framework for software engineering defines five framework activities-communication, planning, modeling, construction, and deployment.
- **Communication:** to gather requirements that help define software features and functions.
- **Planning:** describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

A GENERIC PROCESS MODEL

- **Modeling** : create a “sketch” of the thing so that you’ll understand the big picture. If required, you refine the sketch into greater and greater detail in an effort to better understand the problem and how you’re going to solve it.
- **Construction**. This activity combines code generation and the testing that is required to uncover errors in the code.

A GENERIC PROCESS MODEL

- **Deployment:** The software is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.
- In addition, a set of umbrella activities- project tracking and control, risk management, quality assurance, configuration management, technical reviews, Measurement , Reusability management and Work product preparation and production are applied throughout the process.

A GENERIC PROCESS MODEL

-
- **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 to monitoring the **software** engineering processes and methods used to ensure quality .

A GENERIC PROCESS MODEL

- **Technical reviews** —assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.
- **Software configuration management** — is the task of tracking and controlling changes in the software throughout the software process
- **Reusability management** —defines criteria for work product reuse and establishes mechanisms to achieve reusable components.

A GENERIC PROCESS MODEL

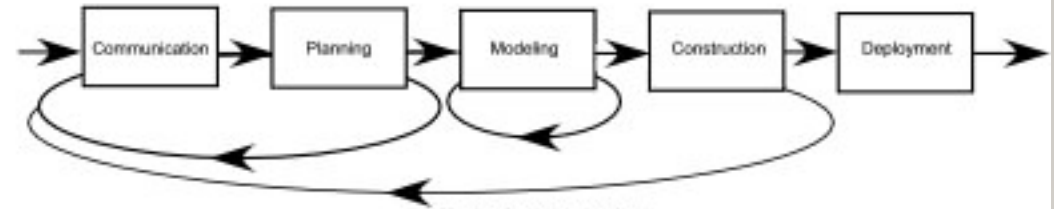
- **Work product preparation and production** —encompasses the activities required to create work products such as models, documents, logs, forms, and lists..
- **Measurement** —defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs

TYPES OF PROCESS FLOW

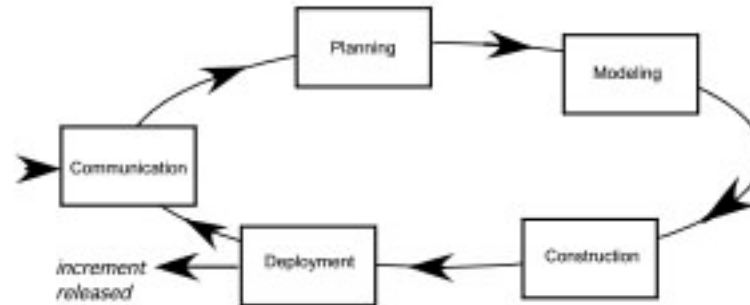
- 1) Linear Process Flow
- 2) Iterative Process Flow
- 3) Evolutionary Process Flow
- 4) Parallel Process Flow



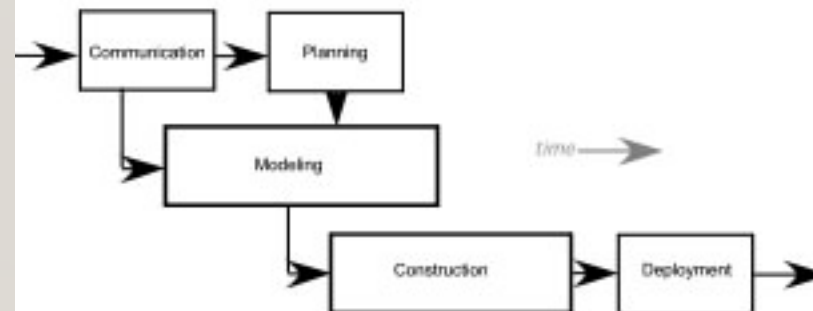
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

PROCESS FLOW_{CONTD...}

- 1) **Linear process flow** executes each of the five activities in sequence.
- 2) **An iterative process flow** repeats one or more of the activities before proceeding to the next.
- 3) **An evolutionary process flow** executes the activities in a circular manner. Each circuit leads to a more complete version of the software.
- 4) **A parallel process flow** executes one or more activities in parallel with other activities

IDENTIFYING A TASK SET

- Before you can proceed with the process model, a key question: what **actions** are appropriate for a framework activity given the nature of the problem, the characteristics of the people and the stakeholders?
- A task set defines the actual work to be done to accomplish the objectives of a software engineering action.

IDENTIFYING A TASK SET

- For example, a small software project requested by one person with simple requirements, the communication activity might encompass little more than a phone call with the stakeholder. Therefore, the only necessary action is phone conversation, the work tasks of this action are:
 1. Make contact with stakeholder via telephone.
 2. Discuss requirements and take notes.
 3. Organize notes into a brief written statement of requirements.
 4. E-mail to stakeholder for review and approval.

EXAMPLE OF A TASK SET FOR ELICITATION

(IS A TECHNIQUE USED TO GATHER INFORMATION)

- The task sets for Requirements gathering action for a **simple** project may include:
 1. Make a list of stakeholders for the project.
 2. Invite all stakeholders to an informal meeting.
 3. Ask each stakeholder to make a list of features and functions required.
 4. Discuss requirements and build a final list.
 5. Prioritize requirements.
 6. Note areas of uncertainty.

The task sets for Requirements gathering action for a **big** project may include:

-
1. Make a list of stakeholders for the project.
 2. Interview each stakeholders separately to determine overall wants and needs.
 3. Build a preliminary list of functions and features based on stakeholder input.
 4. Schedule a series of facilitated application specification meetings.
 5. Conduct meetings.
 6. Produce informal user scenarios as part of each meeting.
 7. Refine user scenarios based on stakeholder feedback.
 8. Build a revised list of stakeholder requirements.
 9. Use quality function deployment techniques to prioritize requirements.
 10. Package requirements so that they can be delivered incrementally.
 11. Note constraints and restrictions that will be placed on the system.
 12. Discuss methods for validating the system.

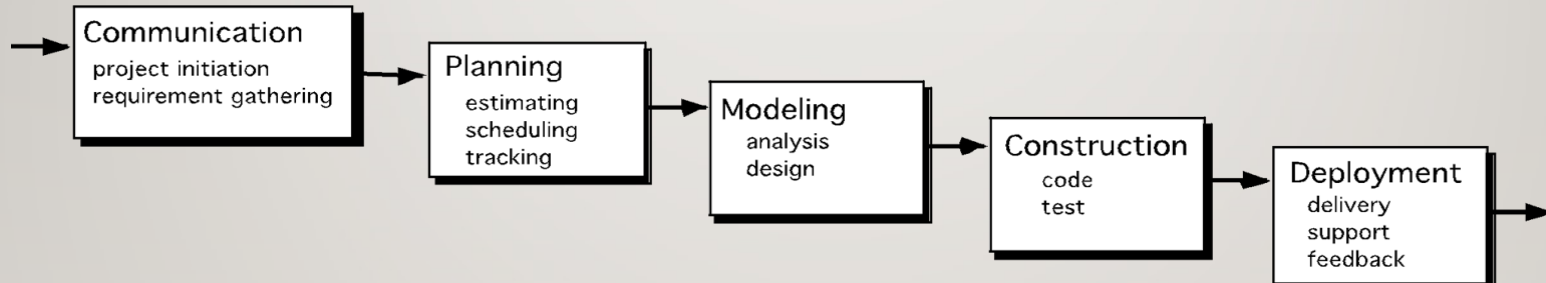
PROCESS MODELS

- **Prescriptive Process Models**
- Specialized Process Models (component based- Not in syllabus)
- **Unified Process**

PRESCRIPTIVE MODELS

- Waterfall/V model
- Incremental
- Evolutionary
 - Prototyping
 - Spiral

THE WATERFALL MODEL



It is the oldest paradigm for SE. When requirements are well defined and reasonably stable, it leads to a linear fashion.

The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software

THE WATERFALL MODEL

- When to select?
- There are times when the requirements for a problem are well understood—when work flows from **communication through deployment in a reasonably linear fashion.**
- (problems : 1. rarely linear, iteration needed. 2. hard to state all requirements explicitly . Blocking state . 3. code will not be released until very late .)

WATERFALL MODEL (CONTD)

When do you choose this model?

- Requirements are well defined
- Well understood
- Well defined enhancements to existing system

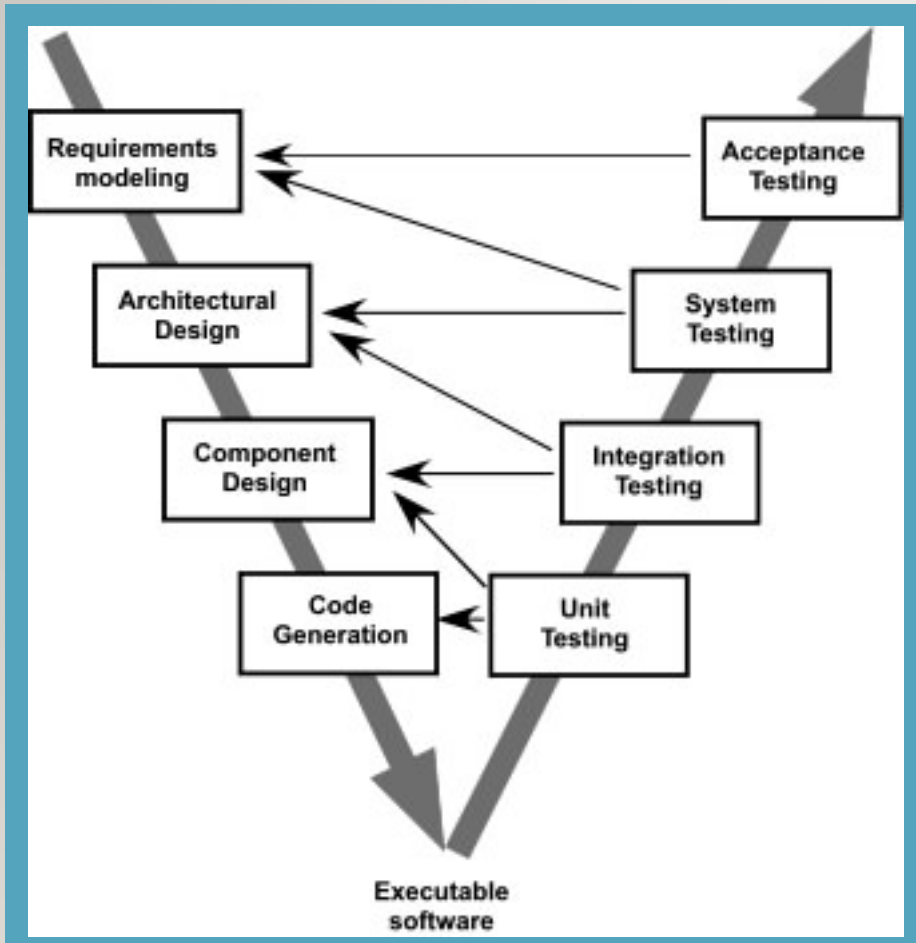
Advantages

1. Simple
2. Easy to understand even for non technical customers
3. Oldest, widely used
4. Base for all other models by including feed back loops, iterations etc.

Disadvantages

- Real projects rarely follow this linear sequence.
- Difficult for customer to state all requirements at one shot
- Customer must have patience.

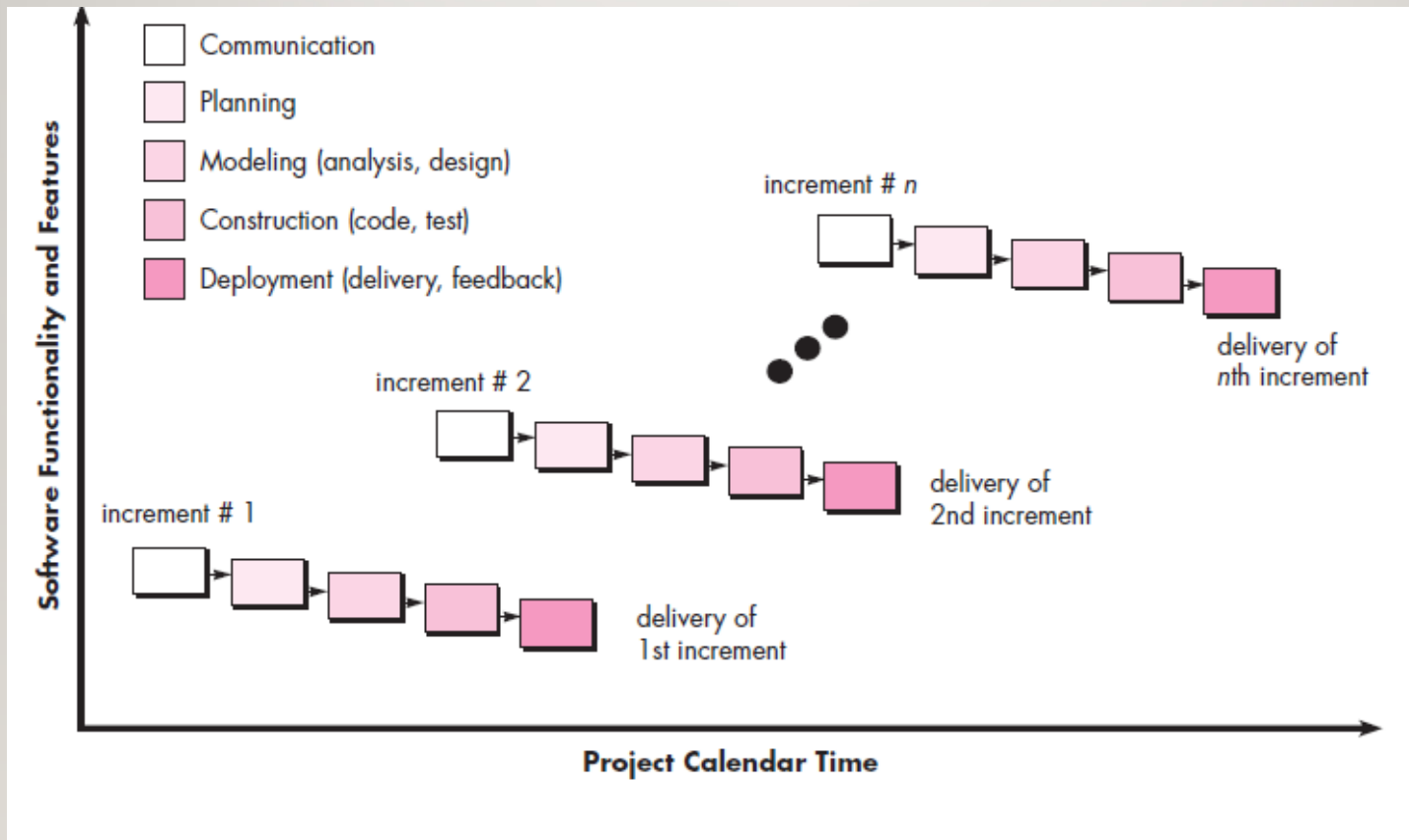
THE V-MODEL



A variation of waterfall model depicts the relationship of quality assurance actions to the actions associated with communication, modeling and early code construction activates.

Team first moves down the left side of the V to refine the problem requirements. Once code is generated, the team moves up the right side of the V, performing a series of tests that validate each of the models created as the team moved down the left side.

THE INCREMENTAL MODEL



INCREMENTAL MODEL (CONTD)

When do we use this model?

- Requirements are reasonably well defined
- But overall scope precludes linear flow
- Difficult deadlines
- Constraints in staffing
- Manage technical risks.
- Compelling need to provide limited set of functionality to users quickly.

Steps

- First increment is a core product
- Core product used/reviewed by the customer
- A plan for the next increment is laid. Modification of the core product for additional functionality
- Process is repeated following the delivery of each increment, until the complete product is produced.

THE INCREMENTAL MODEL

- When initial requirements are reasonably well defined, but the overall scope of the development effort prevent a purely linear process.
- A compelling need to expand a limited set of new functions to a later system release.
- It combines elements of linear and parallel process flows. Each linear sequence produces deliverable increments of the software.
- The first increment is often a core product with many supplementary features. Users use it and evaluate it with more modifications to better meet the needs.

THE INCREMENTAL MODEL

- This process is repeated following the delivery of each increment, until the complete product is produced
- When to use?
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

THE INCREMENTAL MODEL

- Early increments can be implemented with fewer people. If the core product is well received, then additional staff (if required) can be added to implement the next increment.
- In addition, increments can be planned to manage technical risks

INCREMENTAL MODEL (CONTD)

Advantages

1. Early feedback is there on the core product.
2. Reduced risk
3. Effective solution since user evaluates core product at each increment level
4. Complexity is reduced

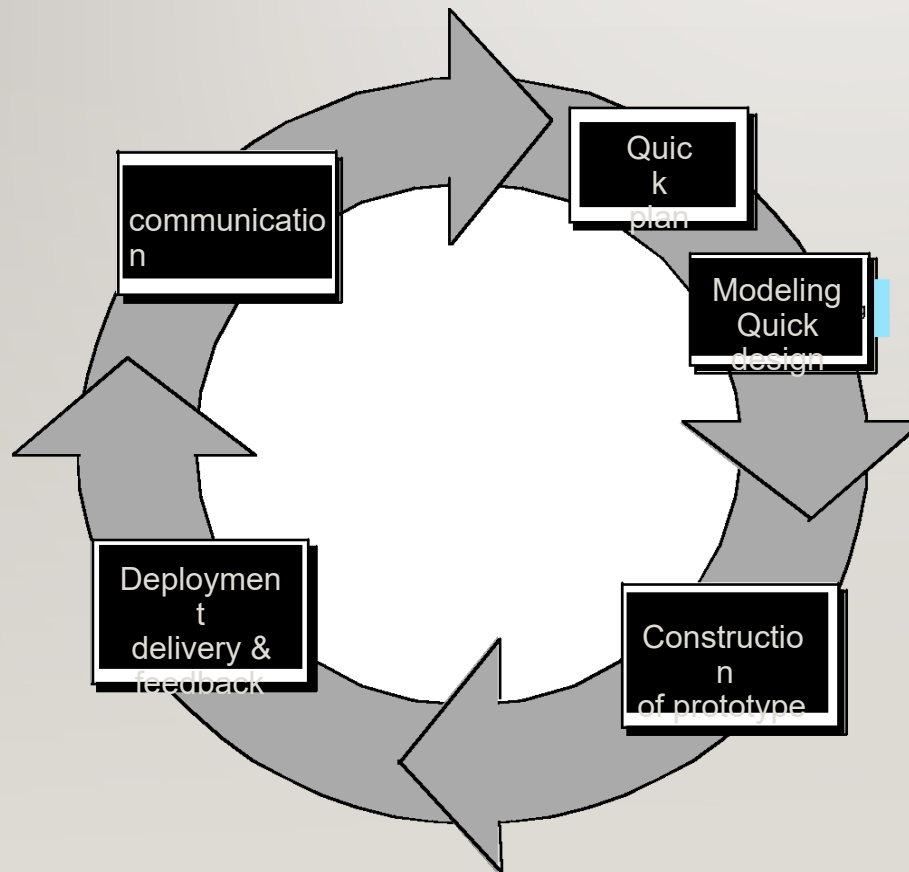
Disadvantages

1. Requires expertise planning both at management and technical level
2. Client dependent, customer should accept phased deliverables

EVOLUTIONARY MODELS

- Software system evolves over time as requirements often change as development proceeds. Thus, a straight line to a complete end product is not possible. However, a limited version must be delivered to meet competitive pressure.
- Usually a set of core product or system requirements is well understood, but the details and extension have yet to be defined.
- You need a process model that has been explicitly designed to accommodate a product that evolved over time.
- It is iterative that enables you to develop increasingly more complete version of the software.
- Two types are introduced, namely **Prototyping and Spiral models**.

EVOLUTIONARY MODELS: PROTOTYPING



Steps

- Begins with communication
- A quick plan for prototyping and modeling occur.
- Quick design focuses on a representation of those aspects the software that will be visible to end users. (interface and output).
- Design leads to the construction of a prototype which will be deployed and evaluated.
- Stakeholder's comments will be used to refine requirements.

EVOLUTIONARY MODELS: PROTOTYPING

- When to use: Customer defines a set of general objectives but does not identify detailed requirements for functions and features. Or Developer may be unsure of the efficiency of an algorithm, the form that human computer interaction should take.
- What step: Begins with communication by meeting with stakeholders to define the objective, identify whatever requirements are known, outline areas where further definition is mandatory. A quick plan for prototyping and modeling (quick design) occur. Quick design focuses on a representation of those aspects the software that will be visible to end users. (interface and output). Design leads to the construction of a prototype which will be deployed and evaluated. Stakeholder's comments will be used to refine requirements.
- Both stakeholders and software engineers like the prototyping paradigm. Users get a feel for the actual system, and developers get to build something immediately. However, engineers may make compromises in order to get a prototype working quickly. The less-than-ideal choice may be adopted forever after you get used to it.

EVOLUTIONARY MODELS: PROTOTYPING

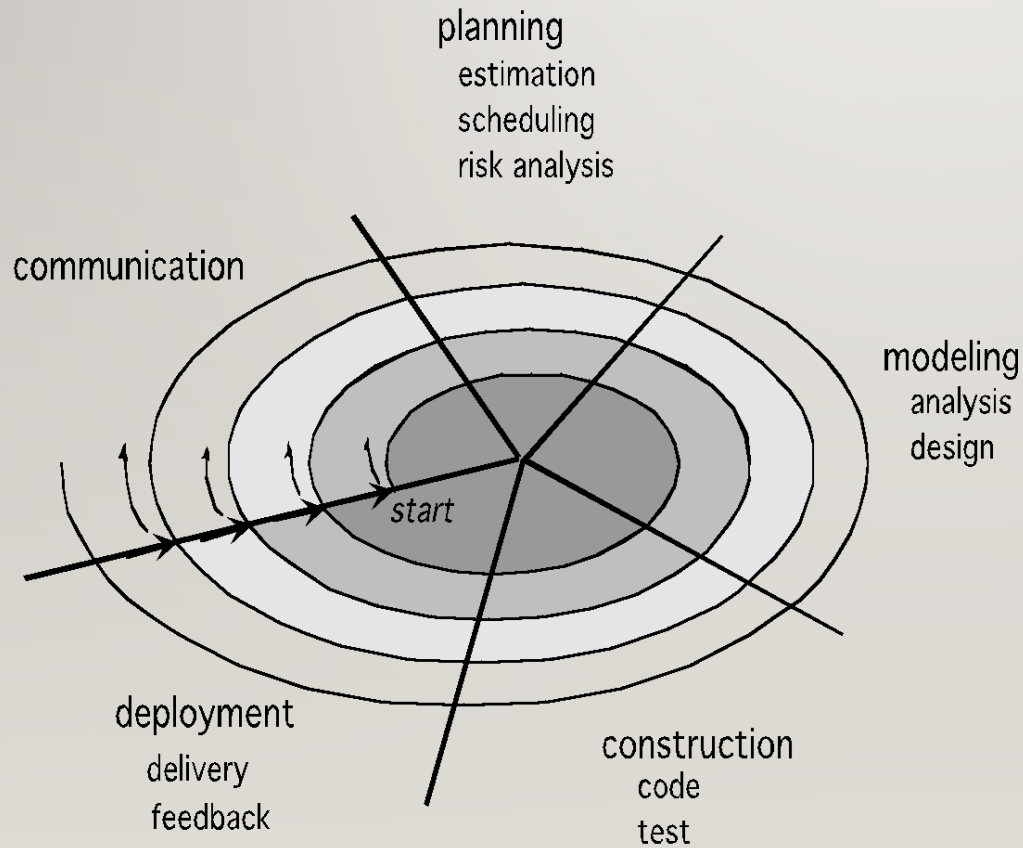
Advantages

- Provides working model.
- Customer is highly satisfied with such a modeling at initial stages
- Developer gains business insight, reducing ambiguity
- Great involvement of users
- Reduce risks

Disadvantages

- Customer - not aware that only interface or appearance is concentrated much and long term quality is at stake
- False expectations from customer that end s/m is finished or will have the same behavior/pace of the prototype
- Inappropriate choices of technology
- Various iterations to a prototype that is to be discarded is expensive

EVOLUTIONARY MODELS: THE SPIRAL



How it works?

- Series of evolutionary releases
 - Earlier releases – prototype, later implementation
 - First circuit - specification, next prototype and sophisticated versions
- Each pass
 - Project plan, Cost, schedule, number of future iterations adjusted
 - Risk is evaluated

EVOLUTIONARY MODELS: THE SPIRAL

- It couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model and is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems.
- Two main distinguishing features: one is **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.
- A series of evolutionary releases are delivered. During the early iterations, the release might be a model or prototype. During later iterations, increasingly more complete version of the engineered system are produced.
- The first circuit in the clockwise direction might result in the product **specification**; subsequent passes around the spiral might be used to develop a **prototype** and then progressively more sophisticated versions of the **software**. Each pass results in adjustments to the project plan. Cost and schedule are adjusted based on feedback. Also, the number of iterations will be adjusted by project manager.
- Good to develop large-scale system as software evolves as the process progresses and risk should be understood and properly reacted to. Prototyping is used to reduce risk.
- However, it may be difficult to convince customers that it is controllable as it demands considerable risk assessment expertise.

EVOLUTIONARY MODELS: THE SPIRAL

Advantages

- Applies throughout lifecycle
 - Concept Development
 - New Prod Development
 - Product Enhancement
- Risk is considered at each pass
- Uses prototyping as risk reduction mechanism
- Customer and developer understand and better react to risks

Disadvantages

- Difficult to convince customers that it is controllable
- Demands considerable risk assessment expertise
- Major risk is not uncovered/managed, problem will occur

THREE CONCERNS ON EVOLUTIONARY PROCESSES

- First concern is that prototyping poses a problem to project planning because of the uncertain number of cycles required to construct the product.
- Second, it does not establish the maximum speed of the evolution. If the evolution occur too fast, without a period of relaxation, it is certain that the process will fall into chaos. On the other hand if the speed is too slow then productivity could be affected.
- Third, software processes should be focused on flexibility and extensibility rather than on high quality. We should prioritize the speed of the development over zero defects. Extending the development in order to reach high quality could result in a late delivery of the product when the opportunity niche has disappeared.

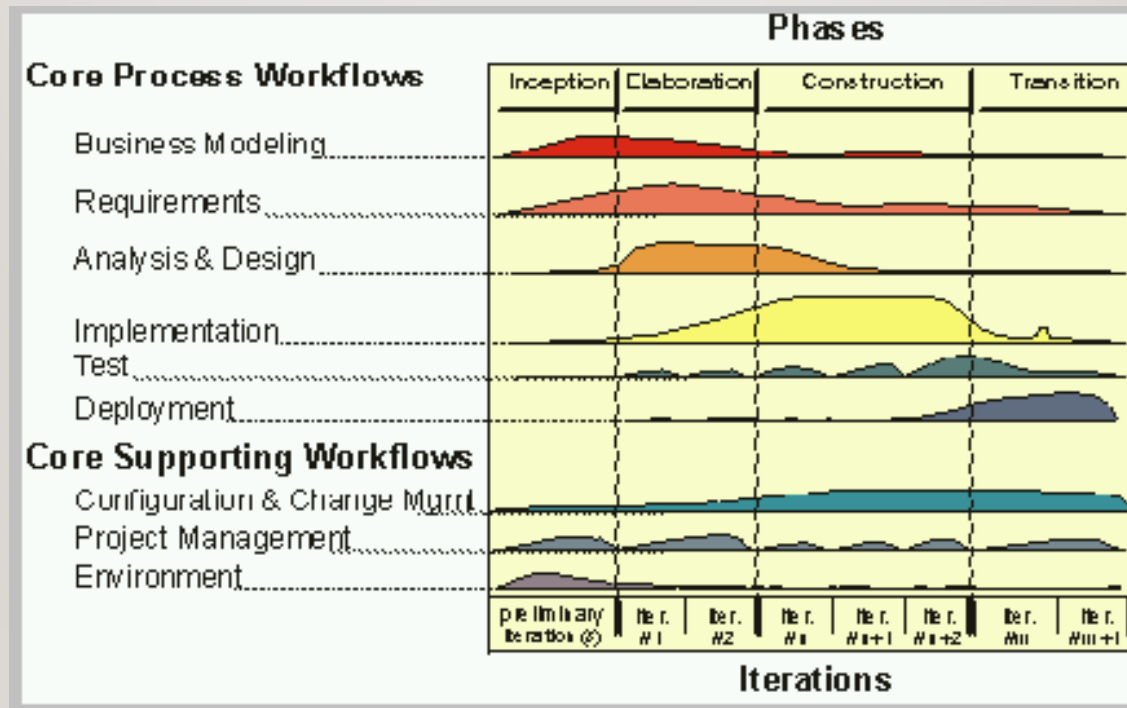
THE RATIONAL UNIFIED PROCESS

- RUP is a method of managing OO Software Development
- It can be viewed as a Software Development Framework which is extensible and features:
 - Iterative Development
 - Requirements Management
 - Component-Based Architectural Vision
 - Visual Modeling of Systems
 - Quality Management
 - Change Control Management

RUP FEATURES

- Online Repository of Process Information and Description in HTML format
- Templates for all major artifacts, including:
 - RequisitePro templates (requirements tracking)
 - Word Templates for Use Cases
 - Project Templates for Project Management
- Process Manuals describing key processes

THE PHASES



AN ITERATIVE DEVELOPMENT PROCESS...

- Recognizes the reality of changing requirements

 - Caspers Jones's research on 8000 projects
 - 40% of final requirements arrived after the analysis phase, after development had already begun
- Promotes early risk mitigation, by breaking down the system into mini-projects and focusing on the riskier elements first
- Allows you to “plan a little, design a little, and code a little”
- Encourages all participants, including testers, integrators, and technical writers to be involved earlier on
- Allows the process itself to modulate with each iteration, allowing you to correct errors sooner and put into practice lessons learned in the prior iteration
- Focuses on component architectures, not final big bang deployments

AN INCREMENTAL DEVELOPMENT PROCESS...

- Allows for software to evolve, not be produced in one huge effort
- Allows software to improve, by giving enough time to the evolutionary process itself
- Forces attention on stability, for only a stable foundation can support multiple additions
- Allows the system (a small subset of it) to actually run much sooner than with other processes
- Allows interim progress to continue through the stubbing of functionality
- Allows for the management of risk, by exposing problems earlier on in the development process

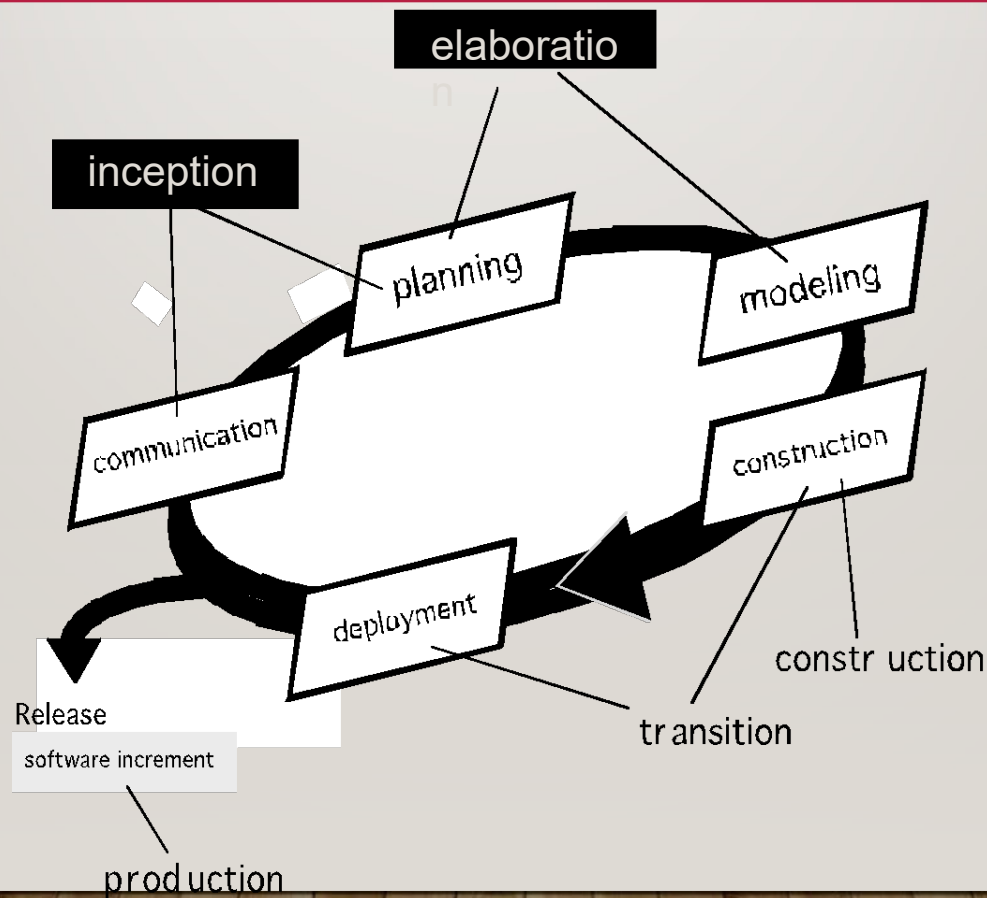
GOALS AND FEATURES OF EACH ITERATION

- The primary goal of each iteration is to slowly chip away at the risk facing the project, namely:
 - performance risks
 - integration risks (different vendors, tools, etc.)
 - conceptual risks (ferret out analysis and design flaws)
- Perform a “miniwaterfall” project that ends with a delivery of something tangible in code, available for scrutiny by the interested parties, which produces validation or correctives
- Each iteration is risk-driven
- The result of a single iteration is an increment--an incremental improvement of the system, yielding an evolutionary approach

THE DEVELOPMENT PHASES

- Inception Phase
- Elaboration Phase
- Construction Phase
- Transition Phase

THE UNIFIED PROCESS (UP)



INCEPTION PHASE

- Overriding goal is obtaining buy-in from all interested parties
- Initial requirements capture
- Cost Benefit Analysis
- Initial Risk Analysis
- Project scope definition
- Defining a candidate architecture
- Development of a disposable prototype
- Initial Use Case Model (10% - 20% complete)
- First pass at a Domain Model

ELABORATION PHASE

- Requirements Analysis and Capture

 - Use Case Analysis
 - Use Case (80% written and reviewed by end of phase)
 - Use Case Model (80% done)
 - Scenarios
 - Sequence and Collaboration Diagrams
 - Class, Activity, Component, State Diagrams
 - Glossary (so users and developers can speak common vocabulary)
 - Domain Model
 - to understand the problem: the system's requirements as they exist within the context of the problem domain
 - Risk Assessment Plan revised
 - Architecture Document

CONSTRUCTION PHASE

- Focus is on implementation of the design:
 - cumulative increase in functionality
 - greater depth of implementation (stubs fleshed out)
 - greater stability begins to appear
 - implement all details, not only those of central architectural value
 - analysis continues, but design and coding predominate

TRANSITION PHASE

- The transition phase consists of the transfer of the system to the user community
- It includes manufacturing, shipping, installation, training, technical support and maintenance
- Development team begins to shrink
- Control is moved to maintenance team
- Alpha, Beta, and final releases
- Software updates
- Integration with existing systems (legacy, existing versions, etc.)

AGILE PROCESS MODEL

THE MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

“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*

That is, while there is value in the items on the right, we value the items on the left more.”

Kent Beck et al

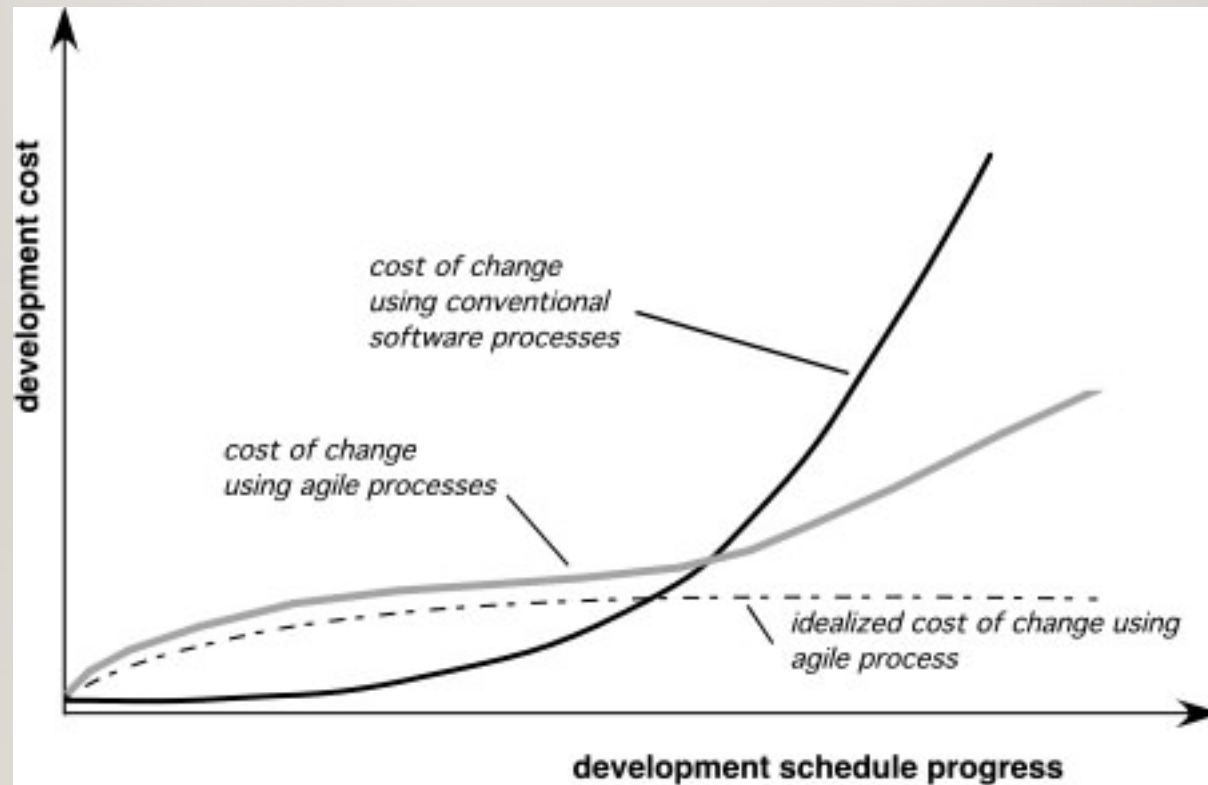
WHAT IS “AGILITY”?

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding ...

- Rapid, incremental delivery of software

AGILITY AND THE COST OF CHANGE



AN AGILE PROCESS

- Is driven by customer descriptions of what is required (scenarios)
- Recognizes that plans are short-lived
- Develops software iteratively with a heavy emphasis on construction activities
- 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.

HUMAN FACTORS

- *the process molds to the needs of the people and team, not the other way around*
- key traits must exist among the people on an agile team and the team itself:
 - **Competence.**
 - **Common focus.**
 - **Collaboration.**
 - **Decision-making ability.**
 - **Fuzzy problem-solving ability.**
 - **Mutual trust and respect.**
 - **Self-organization.**

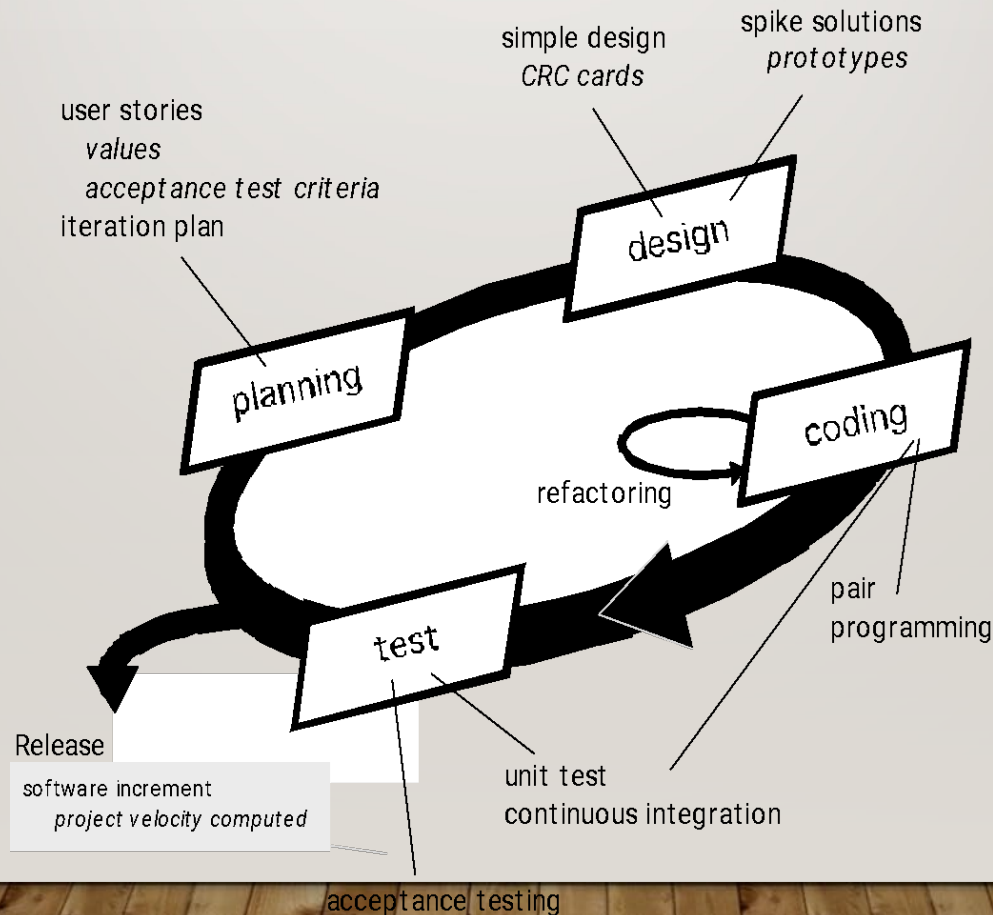
EXTREME PROGRAMMING (XP)

- The most widely used agile process, originally proposed by Kent Beck
- 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

EXTREME PROGRAMMING (XP)

- XP Design
 - Follows the **KIS principle**
 - Encourage the use of **CRC cards** (see Chapter 8)
 - For difficult design problems, suggests the creation of “**spike solutions**”—a design prototype
 - Encourages “**refactoring**”—an iterative refinement of the internal program design
- XP Coding
 - Recommends the **construction of a unit test** for a store *before* coding commences
 - Encourages “**pair programming**”
- XP Testing
 - All **unit tests are executed daily**
 - “**Acceptance tests**” are defined by the customer and executed to assess customer visible functionality

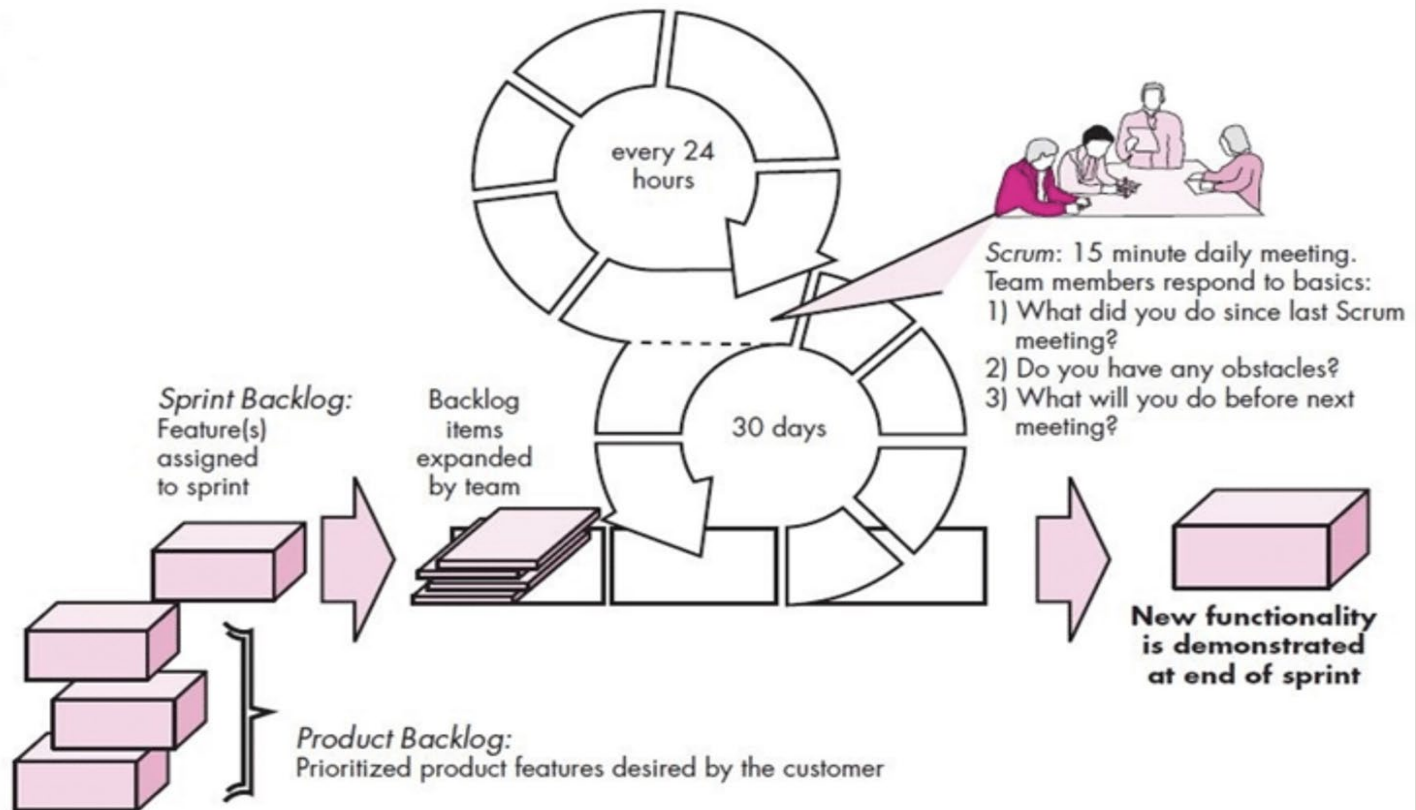
EXTREME PROGRAMMING (XP)



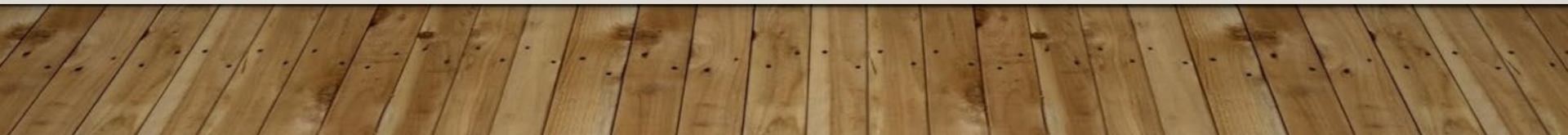
SCRUM

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
 - Development work is partitioned into “packets”
 - Testing and documentation are on-going as the product is constructed
 - Work occurs in “sprints” and is derived from a “backlog” of existing requirements
 - Meetings are very short and sometimes conducted without chairs
 - “demos” are delivered to the customer with the time-box allocated

SCRUM



Project Initiation Management



82 Project Initiation

- Project initiation is the first step in starting a new project. During the project initiation phase, you establish why you're doing the project and what business value it will deliver.
- Project initiation is the first phase of the project management life cycle and in this stage, companies decide if the project is needed and how beneficial it will be for them. The two metrics that are used to judge a proposed project and determine the expectations from it are the business case and feasibility study.

83 Importance of Project Initiation

- Take major decisions that establish the direction and resource requirements, like the project charter and selecting the project stakeholders, are made during this phase. The stakeholders arrive at a clear objective to ensure everyone stays on the same page in terms of how the project should proceed.
- There will be multiple checks during and after project execution to prevent miscommunication and to ensure the project stays on track throughout its course. However, precious time and resources might get wasted which is undesirable.
- Effective project management requires you to maximize benefits and minimize costs while delivering 'value' to the customer. Having a clear project objective helps you achieve all this.

84 Project initiation process 6 steps

1. Creating a business case
2. Conducting a feasibility study
3. Establishing a project charter
4. Identifying stakeholders and making a stakeholder register
5. Assembling the team and establishing a project office
6. Final review

Project Initiation

Follow these six steps to start your project off right



1 Business Case

Explain why the project is necessary and how it will succeed

2 Feasibility Study

Research the reason for the project and determine if it will succeed

3 Project Charter

How will the project be structured and executed?

4 Team

Find the people with the right skills and experience to execute the project

5 Project Office

Where the project manager and support staff are located to assist with projects

6 Review

Review the initiation phase and keep reviewing progress throughout the project

86 Project charter

A project charter demonstrates why your project is important, what it will entail, and who will work on it—all through the following elements:

Why: The project's goals and purpose

What: The scope of the project, including an outline of your project budget

Who: Key stakeholders, project sponsors, and project team members

87 Project Scope

Project scope is a way to set boundaries on your project and define exactly what goals, deadlines, and project deliverables you'll be working towards. By clarifying your project scope, you can ensure you hit your project goals and objectives without delay or overwork.

88 Benefits of defining your project scope

- Ensure all stakeholders have a clear understanding of the boundaries of the project
- Manage stakeholder expectations and get buy-in
- Reduce project risk
- Budget and resource plan appropriately
- Align your project to its main objectives
- Prevent scope creep (when project deliverables exceed the project scope)
- Establish a process for change requests (for complex projects)

89 Steps in defining project scope

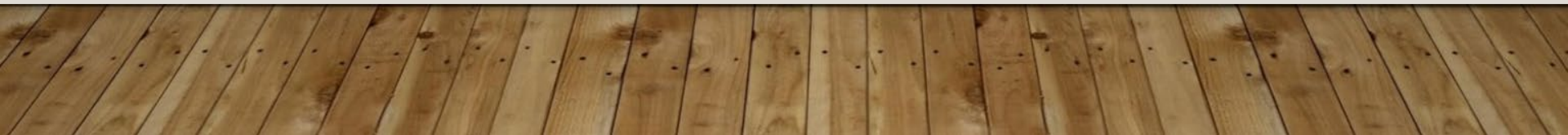
- 1: Define the project goal and objectives
- 2: Identify the potential roadblocks
- 3: Incorporate additional project requirements
- 4: List necessary resources
- 5: Chart a project milestone schedule
- 6: Define the tasks and deliverables
- 7: Have a change management and control system
- 8: prepare and share the project scope statement

90 Project Objectives

Project objectives are what you plan to achieve by the end of your project.

This might include deliverables and assets, or more intangible objectives like increasing productivity or motivation.

Your project objectives should be attainable, time-bound, specific goals you can measure at the end of your project.



91 Steps to follow in writing Project Objectives

1. Set your project objectives at the beginning of your project
2. Involve your project team in the goal-setting process
3. Create brief, but clear, project objective statements
4. Make sure your objectives are things you can control (SMART-Specific, Measurable, Achievable, Realistic, Time-bound)
5. Check in on your project objectives during the project's lifecycle

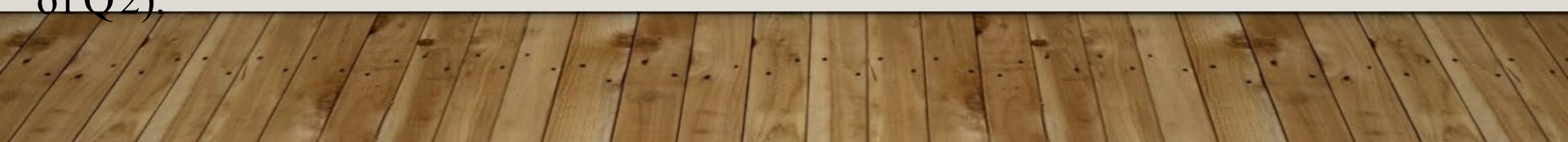
92 Example 1 Business project objective

Bad: Launch new home page.

This project objective is missing many important characteristics. Though this objective is measurable, achievable, and realistic, it's not specific or time-bound. When should the home page be live? What should the redesign focus on?

Good: Create net-new home page assets and copy, focusing on four customer stories and use cases. Launch refreshed, customer-centric home page by the end of Q2.

This project objective is solid. It's specific (create net-new home page assets and copy), measurable (launch refreshed, customer-centric home page), achievable and realistic (focusing on four customer stories and use cases), and time bound (by the end of Q2).



93 Example 2 Non profit Project Objective

Bad: Increase sustainability in our production process by 5%

Though this project objective is more specific than the previous bad example, it's still lacking several important characteristics. This objective is measurable (by 5%), but it's not specific or time-bound, since we don't specify what "sustainability" means or by when the production process should improve. As a result, we don't really know if it's achievable or realistic.

Good: Reduce operational waste by 5% and increase use of recycled products by 20% in the next 12 weeks.

This project objective builds upon the previous one, because we now have a specific objective. This project objective also includes a way to measure the goal (by 5%.. by 20%). The objective is a little ambitious, but the fact that it's time-bound (in the next 12 weeks) makes it

