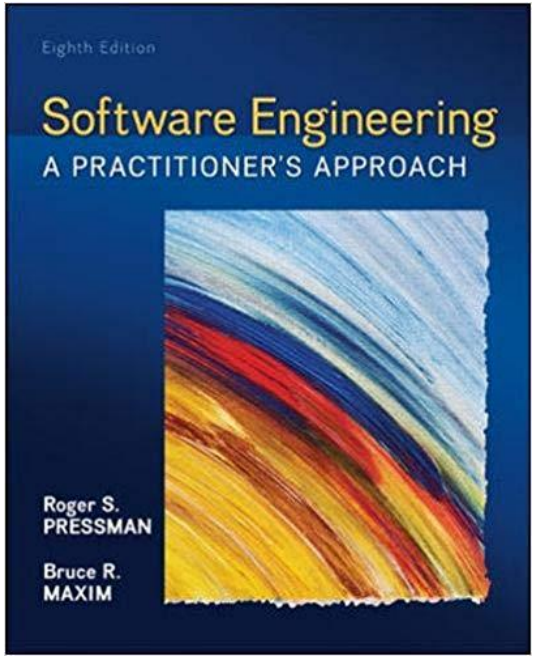


Introduction to Software Engineering

CS46 Software Engineering

Dr. Shilpa chaudhari

Department of Computer Science and Engineering
RIT, Bangalore

CS46 Syllabus

	Software Engineering **Course Code: CS46 Credits: 4:0:0:0**	Reference Chapter
Unit I	**Software Process**: The Changing Nature of Software - WebApps, Mobile Applications, Cloud Computing, Product Line Software;	1.2
	Software Process - The Process Framework, Umbrella Activities, Process Adaptation;	2.2
	Process Models: Prescriptive Process Models - The Waterfall Model, Incremental Process Models, Evolutionary Process Models, Concurrent Models; Specialized Process Models - Component-Based Development, The Formal Methods Model, Aspect-Oriented Software Development; The Unified Process;	4.1, 4.2, 4.3
	Agile Development - What Is Agility? Agility and the Cost of Change, What Is an Agile Process?, Extreme Programming, Other Agile Process Models – Scrum, Dynamic Systems Development Method, Agile Modeling, Agile Unified Process; A Tool Set for the Agile Process;	Chapter 5
	SPI Process, CMMI;	37.2, 37.3
	Case study on current problem statement of software development.	
Unit II	**Understanding Requirements**: Requirements Engineering, Eliciting Requirements, Developing Use Cases, Building the Analysis Model, Negotiating Requirements, Requirements Monitoring, Validating Requirements, Avoiding Common Mistakes.	8.1, 8.3, 8.4, 8.5 8.6, 8.9
	Scenario-Based Requirements Modeling: Requirements Analysis, Scenario-Based Modeling, UML Models That Supplement the Use Case.	9.1, 9.2, 9.3
	Requirements Modeling for Web and Mobile Apps	11.5
	Applying requirement engineering on the same case study of Unit-1.	

CS46 Syllabus continued…

Unit III	**Design Concepts**: The Design Process, Design Concepts, Design Model;	12.2, 12.3, 12.4
	Architectural Design: Software Architecture, Architectural Genres, Architectural Styles, Architectural Considerations, Architectural Decisions, Architectural Design;	13.1, 13.2, 13.3, 13.4, 13.5, 13.6
	User Interface: The Golden Rules of User Interface Analysis and Design, WebApp and Mobile Interface Design;	15.1, 15.2,15.5
	Applying design modeling by taking requirement specification from Unit-2	
Unit IV	**Quality Concepts:** Software Quality, Software Quality Dilemma, Achieving Software Quality	19.2, 19.3, 19.4
	Formal Technical Reviews	20.6
	Software Project Estimation - Observations on Estimation, The Project Planning Process Software Project Estimation, Decomposition Techniques, Empirical Estimation Models, Estimation for Object-Oriented Projects	33.1, 33.2, 33.5, 33.6, 33.7, 33.8
	Project Scheduling –Scheduling.	34.5
	Risk Management: Reactive versus Proactive Risk Strategies, Software Risks, Risk Identification, Risk Projection, Risk Refinement, Risk Mitigation, Monitoring, and Management.	35.1, 35.2, 35.3, 35.4, 35.5, 35.6
	Computation of relevant metrics for the case study on current problem statement of software development considered in Unit-1.	
	Software Maintenance, Software Supportability, Software Reengineering, Reverse Engineering, Restructuring, Forward Engineering.	36.2,36.5, 36.6, 36.7, 36.8

CS46 Syllabus continued…

Unit V	**Software Testing Strategies**: A Strategic Approach to Software Testing, Strategic Issues, Test Strategies for Conventional Software, Test Strategies for Object-Oriented Software, Test Strategies for WebApps, Test Strategies for MobileApp, Validation Testing, System Testing	22.1,22.2, 22.3,22.4, 22.5, 22.6, 22.7, 22.8
	Testing Conventional Applications: Software Testing Fundamentals, Internal and External Views of Testing, White-Box Testing, Basis Path Testing, Control Structure Testing, Black-Box Testing, Model-Based Testing, Testing Documentation and Help Facilities	23.1, 23.2, 23.3, 23.4, 23.5, 23.6, 23.7, 23.8
	Testing OOA and OOD Models, Object-Oriented Testing Strategies, Object-Oriented Testing Methods;	24.3, 24.4
	Designing test cases as per the requirement specification from Unit-2	

Books

- Text Book:

  Roger S Pressman and Bruce R. Maxim, Software Engineering: A Practitioner's Approach, 8/e, New York: McGraw-Hill, 2015

- Reference Books:

  1. Ian Sommerville, Software Engineering, 10th Edition,Pearson, 2016
  2. Emilia Mendes, Nile Mosley: Web Engineering, Springer, 2006
  3. David Gustafson: Software Engineering, Schaum's Outline Series, McGraw Hill, 2002

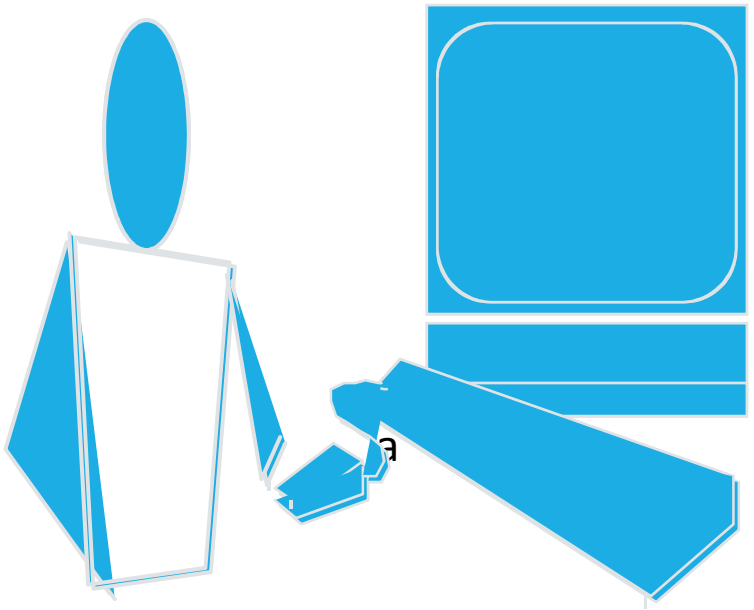
What is Software?

Software is a set of items or objects that form a "configuration" that includes

- programs
- documents
- data ...

Characteristics:

- software is engineered
- software doesn't wear out
- software is complex



6

Software Applications Domains

- system software
- application software
- engineering/scientific software
- embedded software
- product-line software
- WebApps (Web applications)
- AI software

Dominating Software categories

- WebApps (Web applications)
- Mobile Applications
- Cloud Computing
- Product Line Software

Software—New Categories

- Open world computing—pervasive, distributed computing
- Ubiquitous computing—wireless networks
- Netsourcing—the Web as a computing engine
- Open source—”free” source code open to the computing community (a blessing, but also a potential curse!)

Changing nature of software

- Web-based systems and applications
  - Evolved from simple collections of information content to sophisticated systems that present complex functionality and multimedia content
  - Have unique features and requirements
- Mobile applications present new challenges as apps migrate to a wide array of platforms.
- Cloud computing will transform the way in which software is delivered and the environment in which it exists.
- Product line software offers potential efficiencies in the manner in which software is built

Legacy Software – older programs

Why must it change?

- software must be adapted to meet the needs of new computing environments or technology.
- software must be enhanced to implement new business requirements.
- software must be extended to make it interoperable with other more modern systems or databases.
- software must be re-architected to make it viable within a network environment**.**

9

Why Software Project Fail ?

Lack of s/w mind

Insufficient software project management

Lack of appropriate SE skills

10

Software Engineering – Process

Software Engineering – Process

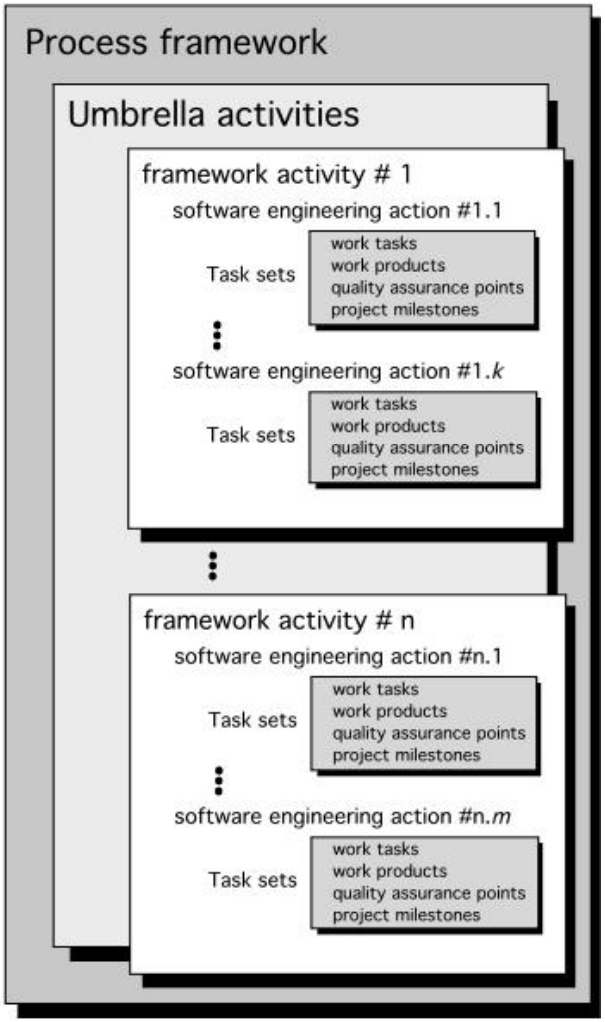
- Software Process : a series of steps involving activities, constraints, and resources that produce an intended output of some kind
  - Process itself - dynamic entity that captures the actions performed
  - Process specification :
    - Products, which are the outcomes of a process activity;
    - Roles, which reflect the responsibilities of the people involved in the process;
    - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced

Software Engineering – Process

- Process includes many types of activities performed by different people in a software project --- process model
  - Activities can be considered as many component processes
    - Activities are related to technical or management issues of software development

Software Engineering – Process

Software process



Framework activities

- Communication
- Planning
- Modeling- Analysis of requirements and Design
- Construction- Code generation and Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement
- Risk management

Process Adaptation

- software engineering process should be agile and adaptable to the problem, to the project, to the team, and to the organizational culture
- 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.
    - work products are identified and required.
    - detail and rigor with which the process is described.
    - the customer and other stakeholders are involved with the project.
    - team organization and roles are prescribed.
  - Level of autonomy given to the software team.
  - Manner in which quality assurance activities are applied.
  - Manner in which project tracking and control activities are applied.

Software Process Model

- Process model –
  - specific roadmap for software engineering work
  - 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
- Varieties of process model
  - ***Prescriptive process model*** - strives for structure and order in software development
  - **Specialized Process Models** – characterized for accomplishing a specific software development goal
  - **The Unified Process** - framework for UML methods and tools
  - **Personal and Team Process Models** - emphasize measurement, planning, and self-direction as key ingredients
  - **Agile Process Models** - define maneuverable, adaptive, lean processes

Software Process Model - Prescriptive Models

Prescriptive Models

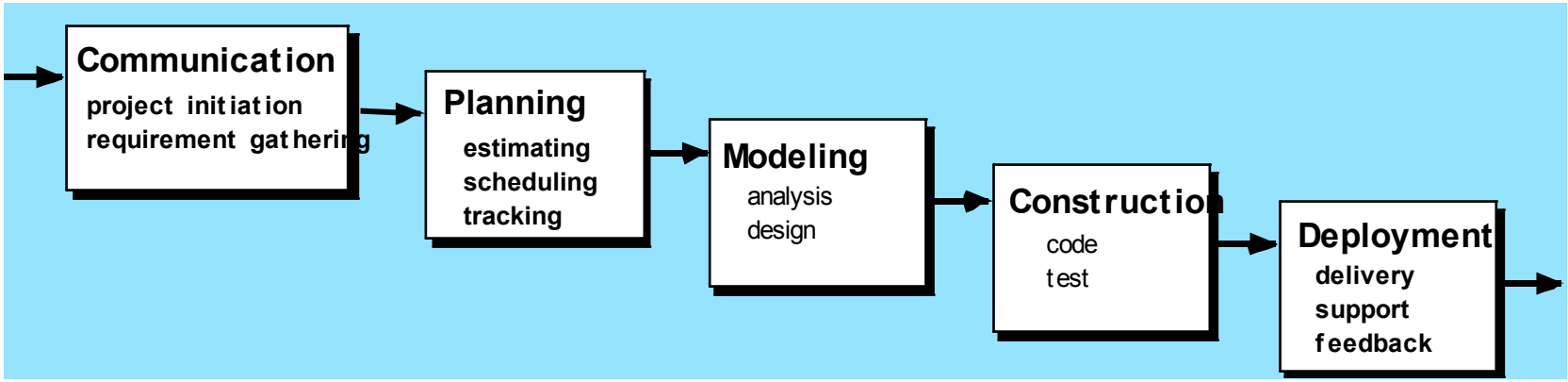
- Prescriptive process models advocate an *orderly* approach to software engineering

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?
- **Prescriptive Process models**
  - Waterfall Model - Sometimes called the classic life-cycl
  - Incremental Process Model
  - Evolutionary Process Model
  - Concurrent model

18

The Waterfall Model (1/2)



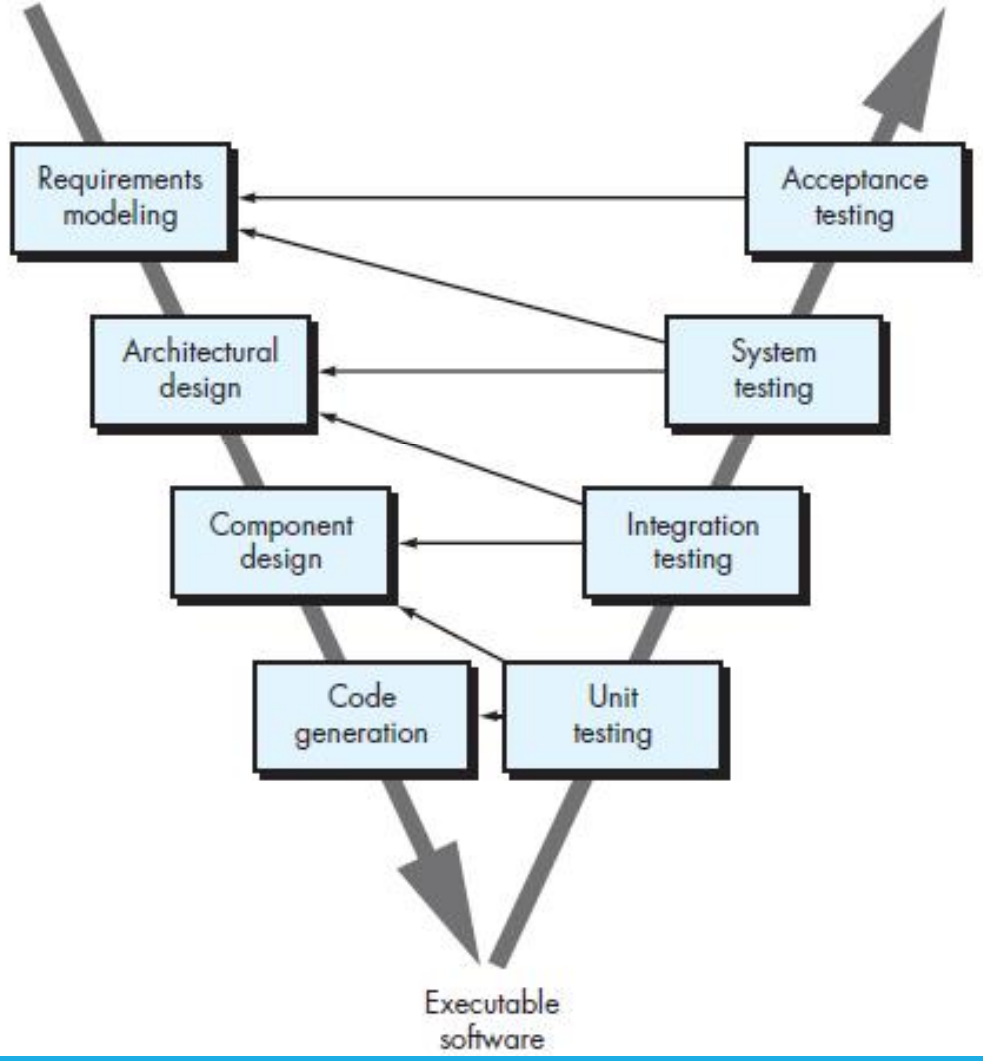
- It is a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modelling, construction and deployment.
- Bradac[BRA94] found that the linear nature of the waterfall model leads to "blocking states"
  - Where some members must wait for other members of the team to complete dependent tasks
  - Especially, at the beginning of the project

19

The Waterfall Model (2/2)

- Still, however, the waterfall model serve as a useful process model where requirements are fixed and work is to proceed to completion in a linear manner
- Which problems does the waterfall model have?
  1. Real projects rarely follow the sequential flow
  2. Difficult to accommodating the uncertainty in requirements
  3. A working version of SW will not be available until late in the project

V-Model

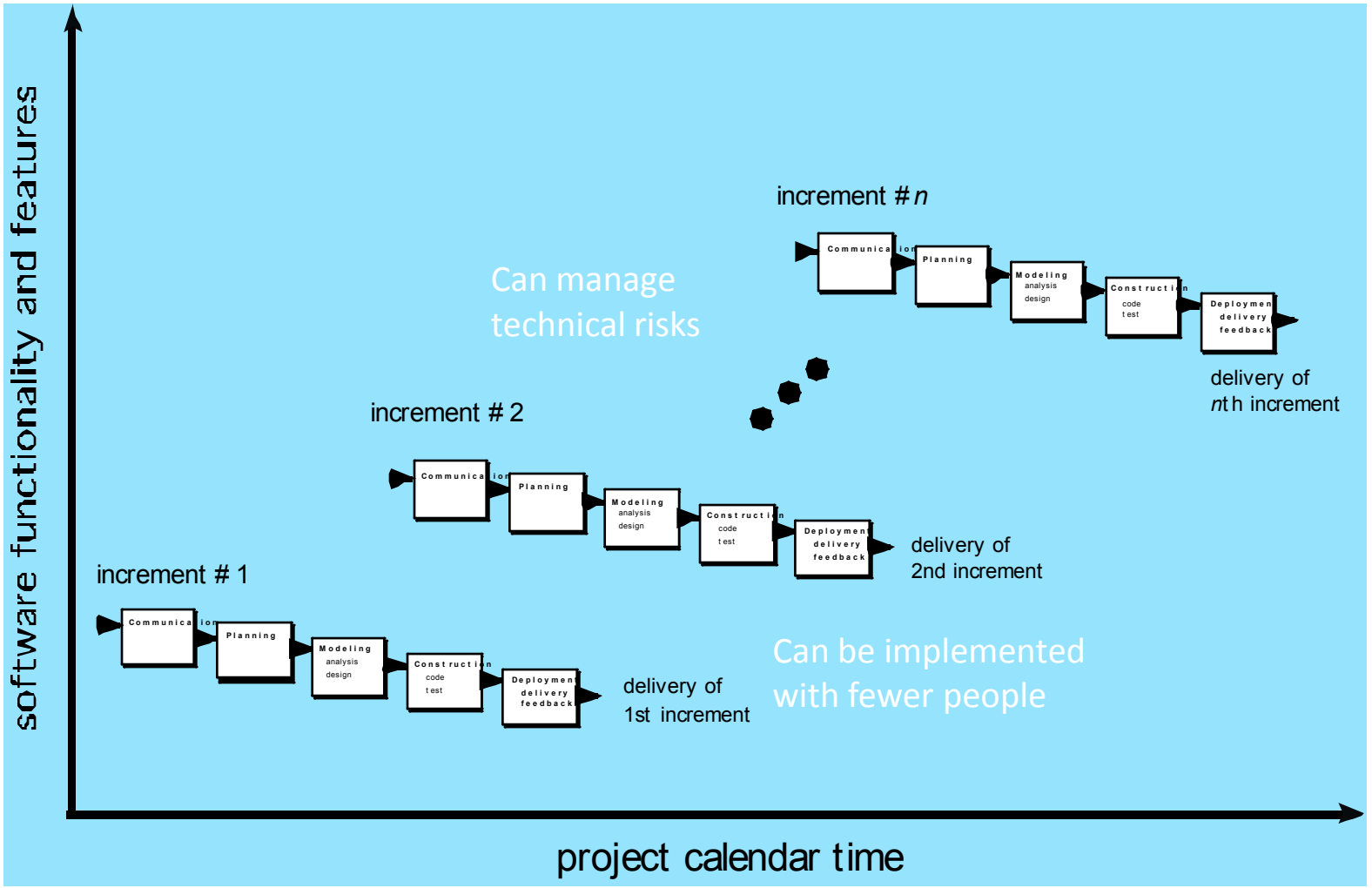


- Describes the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities
- provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

The Incremental Model

- Combines elements of linear and parallel process flows.
- It delivers a series of releases, called increments that provide progressively more functionality for the customer as each is delivered
- The first increment is often a core product.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional feature and functionality.
- It focuses on the delivery of an operational product with each increment.
- It is useful when staffing is unavailable for a complete implementation.
- Increments can be planned to manage technical risks.

The Incremental Model



23

Example using the Incremental Model

- Word-processing software
  - Develop the basic file management, editing, and document production functions in the first increment
  - More sophisticated editing and document production capabilities in the second increment
  - Spelling and grammar checking in the third increment

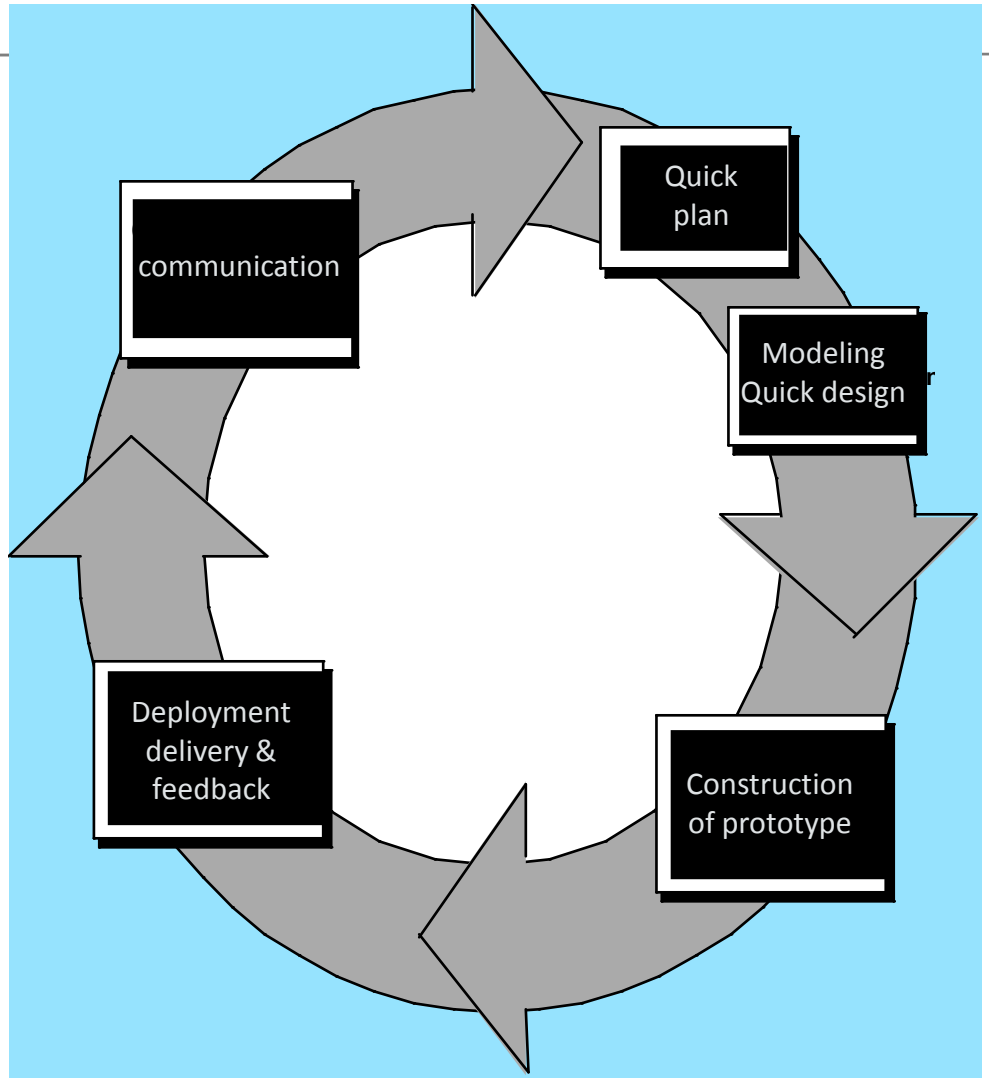
Evolutionary Process Model

- Complex systems evolve over a period of time-
  - Business and product requirements often change as development proceeds.
  - tight market deadlines make completion of a comprehensive software product impossible, but a limited version must be introduced to meet competitive or business pressure
  - a set of core product or system requirements is well understood, but the details of product or system extensions have yet to be defined.
- Evolutionary models are iterative.
- Evolutionary Process Model produce an increasingly more complete version of the software with each iteration.
- Specification, development and validation are interleaved.
- Evolutionary Process Model
  - Prototyping
  - Spiral Model

Evolutionary Models: Prototyping (1/2)

- A prototyping paradigm is the best-fit for the following situations
  - A customer does not identify detailed requirements for SW
  - SW engineers are not sure of the efficiency of an algorithm, usability of SW, and so on.
- The quick design and implementation focuses on a representation of those aspects of the SW that will be visible to the customer
  - Ideally, the prototype serves as a mechanism for identifying SW requirements
- It assists you and stakeholders to better understand what is to built when requirements are fuzzy.
- Prototyping paradigm
  - begins with communication.
  - planned quickly and modeling occurs
  - quick design focuses on a representation of those aspects of the software that will be visible to end users.

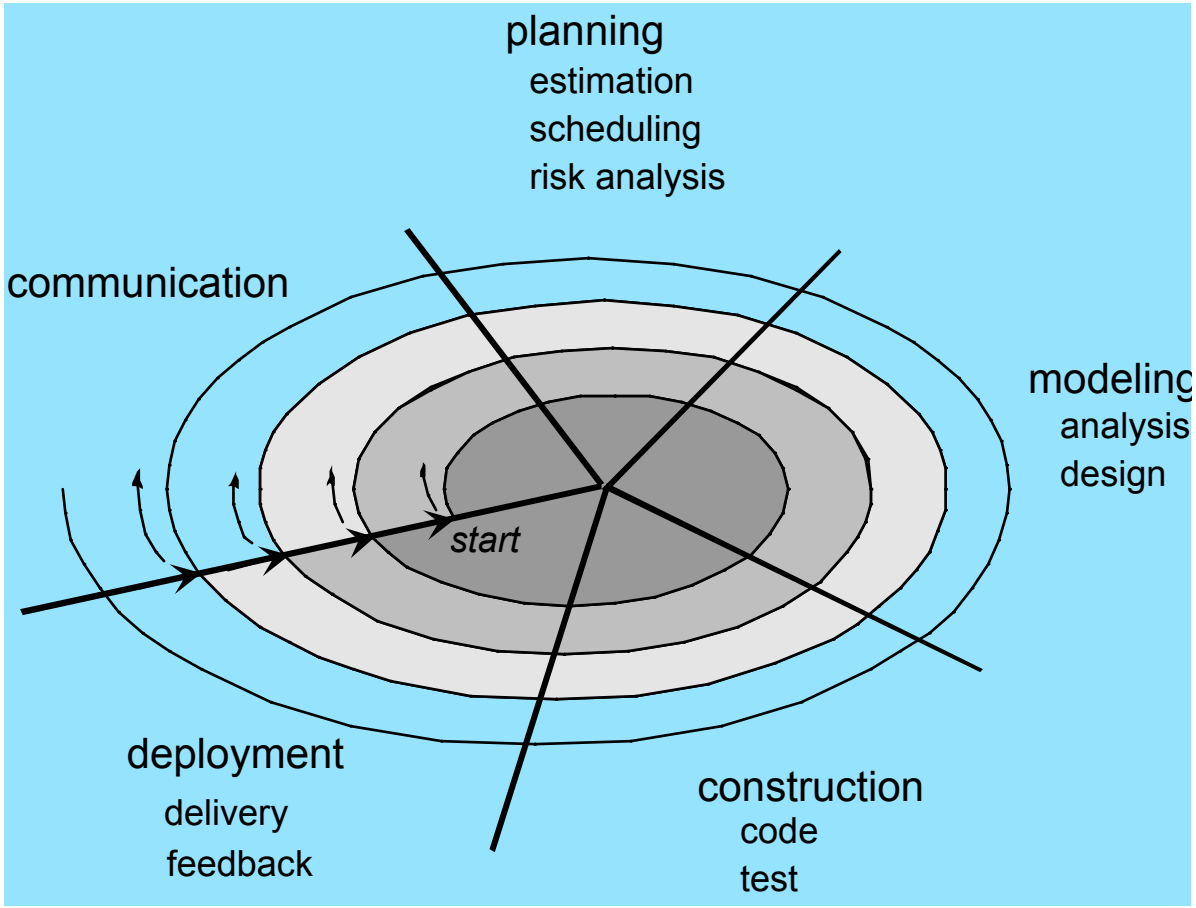
Evolutionary Models: Prototyping (2/2)



- Some problems in prototyping paradigm
  - SW engineers try to modify the prototype to use as a working version
  - Once the customer see the working prototype, he/she expects to get working product soon
  - Overall software quality or long-term maintainability is not considered.
  - As a Software Engineer make implementation compromises in order to get a prototype working quickly.
- Useful
  - The customer get feel for the actual system and developers get to build something immediately
  - Key here is all the stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements.

27

Evolutionary Models: The Spiral



28

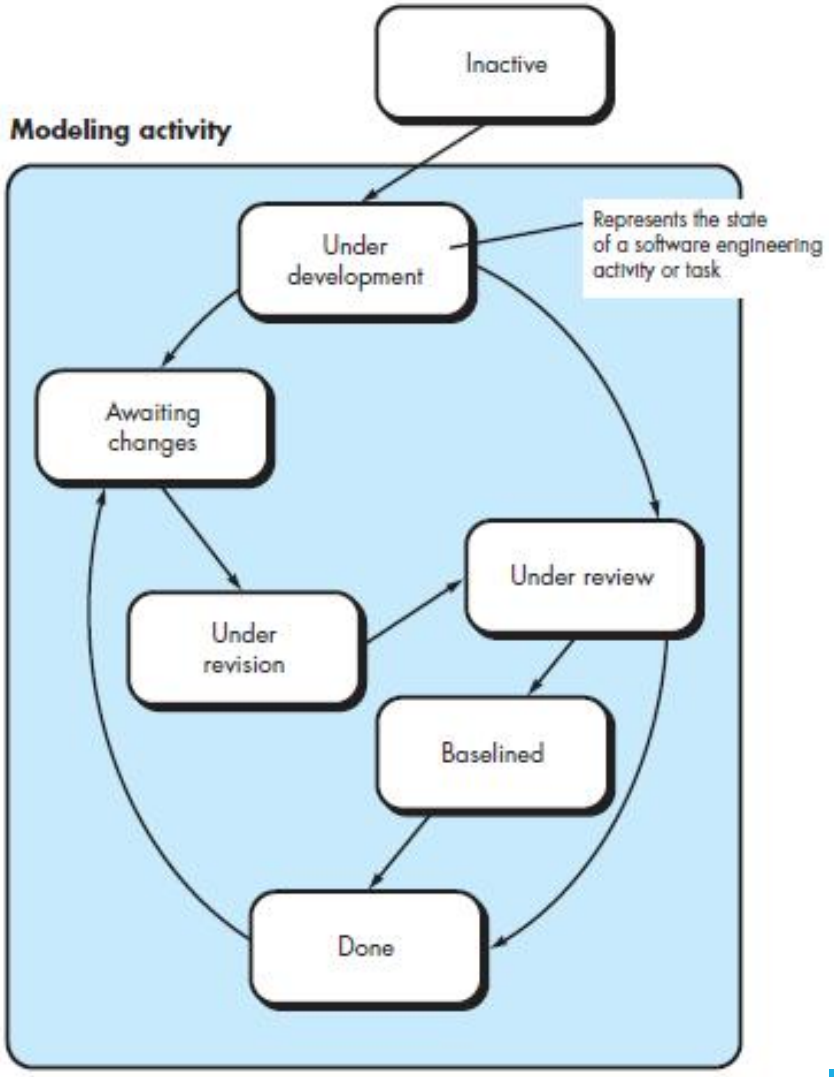
Evolutionary Models: The Spiral

- Characteristics
  - Originally proposed by Barry Boehm.
  - It couples the iterative nature of prototyping with the controlled and systematic aspects of water fall model.
  - Process is represented as a spiral rather than as a sequence of activities with backtracking.
  - Each loop in the spiral represents a phase in the process.
  - No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
  - Risks are explicitly assessed and resolved throughout the process.
  - Uses prototyping as a risk reduction mechanism, but more importantly, enables us to apply the prototyping approach at any stage in the evolution of the product
  - It is a realistic approach to the development of large-scale systems and software.
  - The software evolves as the process progresses, the developer and customer better understand and react at each evolutionary level.
  - It demands considerable risk assessment expertise and realize on this expertise for success.

Evolutionary development

- Problems
  - Do not establish the max. speed of the evolution.
  - Systems are often poorly structured;
  - Special skills (e.g. in languages for rapid prototyping) may be required.
  - Proj. mgmt. and estimation technique do not fit completely
- Applicability
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems.

Concurrent Model



- The concurrent development model sometimes called Concurrent Engineering.
- It allows team to represent iterative and concurrent elements of any of the process models.
- All software engineering activities exist concurrently but reside in different states.
- Concurrent modeling defines a series of events that will trigger transitions from state to state for each of the activities.
- Concurrent modeling is applicable to all types of software development and provide an accurate picture of the current state of a project.

Software Process Model – Specialized Models

Specialized Process Models

- Special Process Models take many of the characteristics of the previous models shown.
- Special Process Models tend to be used when a narrowly defined Software Engineering Approach is chosen.

Component Based Development

- The process to apply when reuse is a development objective
- COTS or Commercial Off-The-Shelf components are becoming more available.
- Most (not all) COTS components have targeted functionality with good interfaces that enable the component to be integrated.
- This approach incorporates many of the aspects of the spiral model.

34

Formal Methods Development

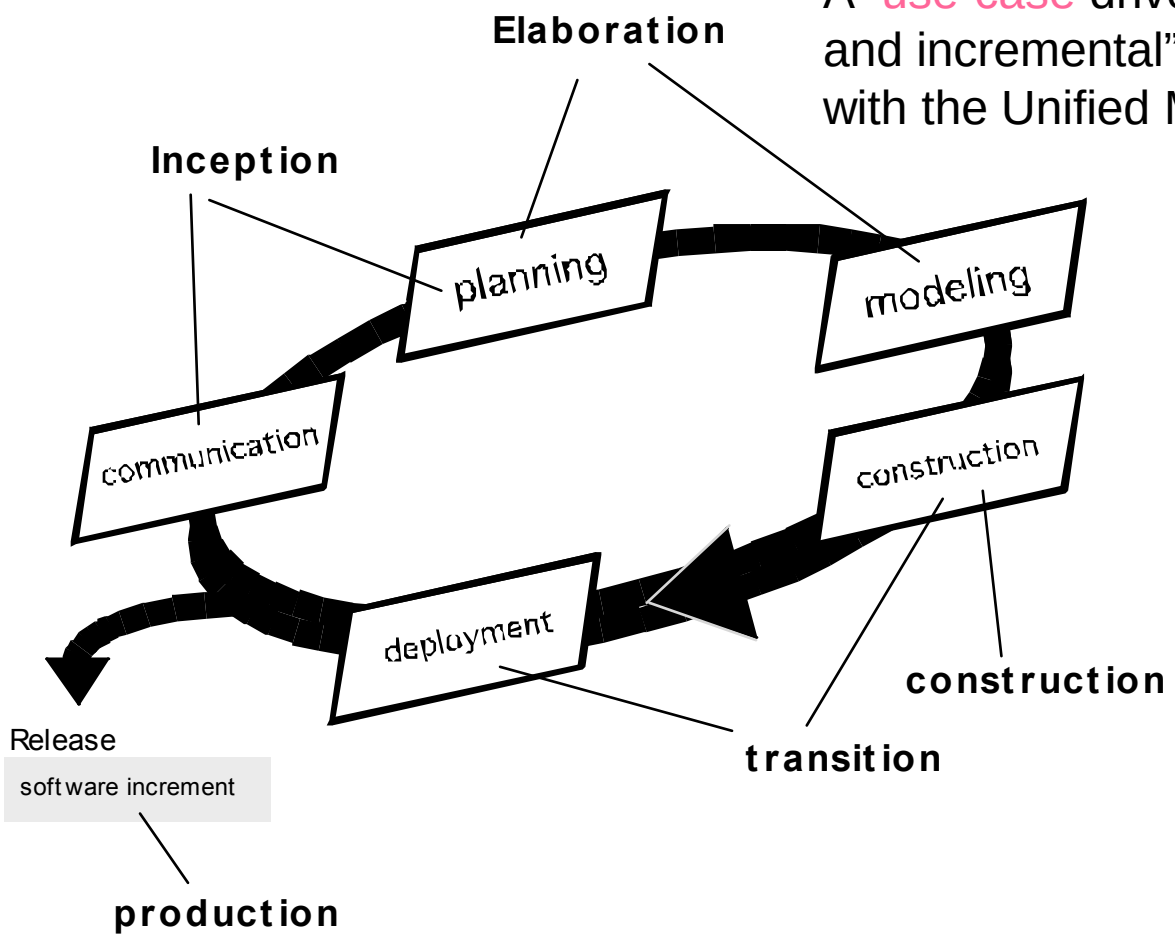
- Model Formal mathematical specification of the software.
- Specify, develop and verify by rigorous math notation.
- Eliminates ambiguity, incompleteness, and inconsistency.
- Used more where safety-critical software is developed, e.g., aircraft avionics, medical devices, etc.

Aspect-Oriented SW Development

- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects* (e.g., security, fault-tolerance, the application of business rules)
- Nearly all SW has localized features, functions and information content.
- User or customer concerns or needs must be included. These can be high-level concerns like security or lower-level such as marketing business rules or systemic such as memory management.
- Aspect-Oriented process is new and still developing.

The Unified Process (UP)

A "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)



37

UP Work Products

Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan,
  - phases and iterations.
- Business model,
  - if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements
  - including non-functional
- Analysis model
- Software architecture
  - Description.
- Executable architectural
  - prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software
  - increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current
    - increment

Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

38

Software Process Improvement (SPI)

Software Process Improvement

- Never stops learning
- How to improve software engineering practices?
  - Address weaknesses in the existing process and work to improve the approach to software work
    - iterative and continuous approach that lead to a better software process and higher quality software delivery in a more timely manner
- Universal five steps: to assess the “maturity” of an organization’s software process and provide a qualitative indication of a maturity level (**Maturity Models)**

  (1) assessment of the current software process

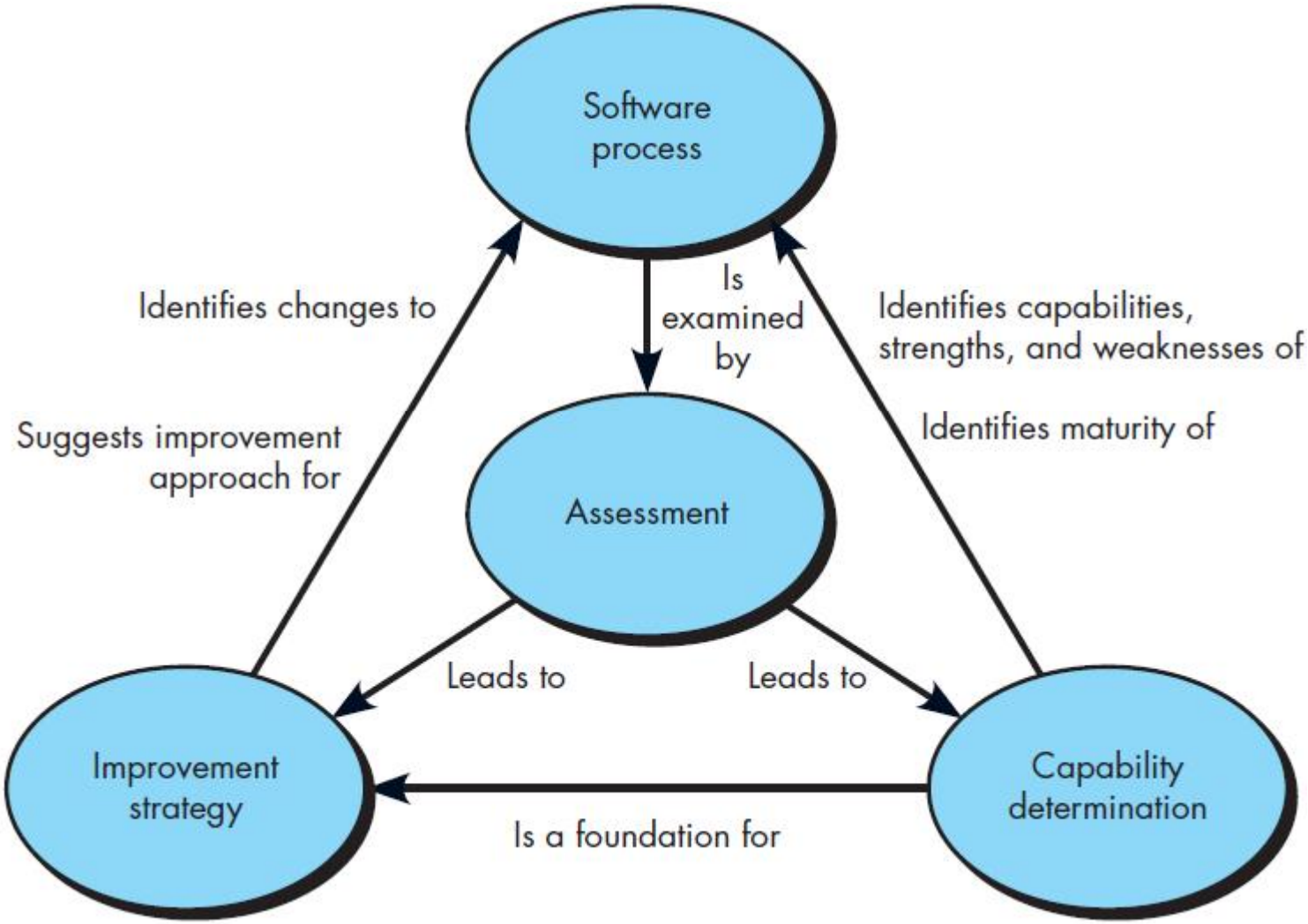
  (2) education and training of practitioners and managers

  (3) selection and justification of process elements, software engineering methods, and tools

  (4) implementation of the SPI plan

  (5) evaluation and tuning based on the results of the plan

Assessment and Improvement



41

SPI Process

- Software Engineering Institute's original *Capability Maturity Model suggests five levels of maturity ranging from initial* (rudimentary software process) to *optimized (a process that leads to best practices.*
- Hard part of SPI - establishing a consensus for initiating SPI and defining an ongoing strategy for implementing it across a software organization

IDEAL road map for SPI	Different road map for SPI
Developed by Software Engineering Institute – defines five distinct SPI activities to guide an organization	five iterative (cyclical) activities are applied in an effort to foster continuous process improvement
1. Initiating	1. look in the mirror - **Assessment and Gap Analysis**
2. Diagnosing	2. get smarter so it can make intelligent choices - **Education and Training**
3. Establishing	3. select the process model (and related technology elements) that best meets its needs - **Selection and Justification**
4. Acting	4. instantiate the model into its operating environment and its culture- **Installation/Migration**
5. Learning	5. evaluate what has been done - **Evaluation**

1. Assessment and Gap Analysis

- Assessment examines a wide range of actions and tasks that will lead to a high quality process.
  - **Consistency**. Are important activities, actions and tasks applied consistently across all software projects and by all software teams?
  - **Sophistication**. Are management and technical actions performed with a level of sophistication that implies a thorough understanding of best practice?
  - **Acceptance**. Is the software process and software engineering practice widely accepted by management and technical staff?
  - **Commitment**. Has management committed the resources required to achieve consistency, sophistication and acceptance?
- Gap analysis—The difference between local application and best practice represents a "gap" that offers opportunities for improvement.

2. Education and Training

- Three types of education and training should be conducted:
  - **Generic concepts and methods**. Directed toward both managers and practitioners, this category stresses both process and practice. The intent is to provide professionals with the intellectual tools they need to apply the software process effectively and to make rational decisions about improvements to the process.
  - **Specific technology and tools**. Directed primarily toward practitioners, this category stresses technologies and tools that have been adopted for local use. For example, if UML has been chosen for analysis and design modeling, a training curriculum for software engineering using UML would be established.
  - **Business communication and quality-related topics.** Directed toward all stakeholders, this category focuses on "soft" topics that help enable better communication among stakeholders and foster a greater quality focus.

3. Selection and Justification

- choose the process model that best fits your organization, its stakeholders, and the software that you build
- decide on the set of framework activities that will be applied, the major work products that will be produced and the quality assurance checkpoints that will enable your team to assess progress
- develop a work breakdown for each framework activity (e.g., modeling), defining the task set that would be applied for a typical project
- Once a choice is made, time and money must be expended to install it within an organization and these resource expenditures should be justified.

4. Installation/Migration

- actually software process redesign (SPR) activities. Scacchi states that “SPR is concerned with identification, application, and refinement of new ways to dramatically improve and transform software processes.”
- three different process models are considered:
  - the existing (“as-is”) process,
  - a transitional (“here-to-there”) process, and
  - the target (“to be”) process.

5. Evaluation

- assesses the degree to which changes have been instantiated and adopted,
- the degree to which such changes result in better software quality or other tangible process benefits, and
- the overall status of the process and the organizational culture as SPI activities proceed
- From a qualitative point of view, past management and practitioner attitudes about the software process can be compared to attitudes polled after installation of process changes.

Risk Management for SPI

- manage risk at three key points in the SPI process:
  - prior to the initiation of the SPI roadmap
  - during the execution of SPI activities (assessment, education, selection, installation), and
  - during the evaluation activity that follows the instantiation of some process characteristic.
- In general, the following categories can be identified for SPI risk factors:
  - budget and cost
  - content and deliverables culture
  - maintenance of SPI deliverables
  - mission and goals
  - organizational management and organizational stability
  - process stakeholders
  - schedule for SPI development
  - SPI development environment and process
  - SPI project management and SPI staff

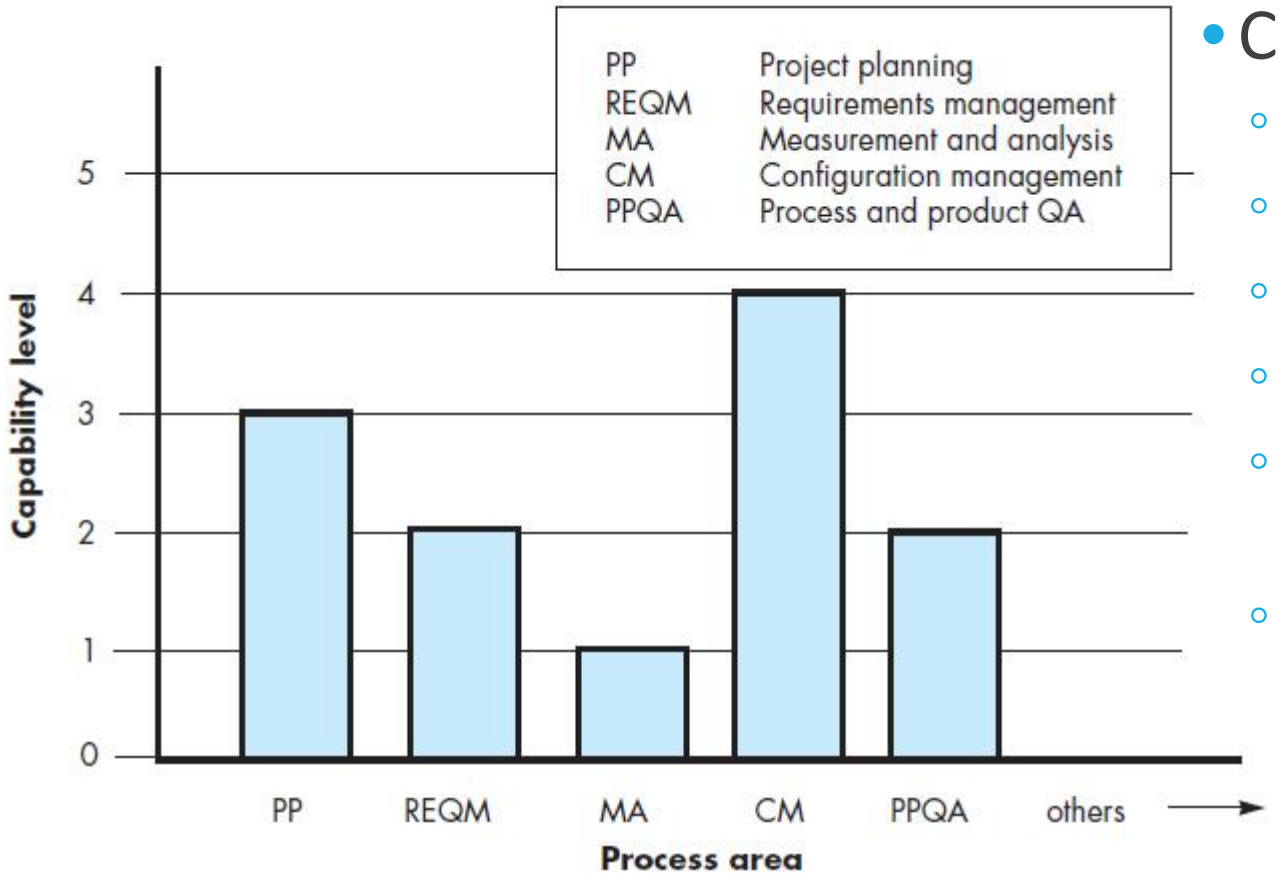
Complete SPI framework

- original CMM -developed and upgraded by the Software Engineering Institute throughout the 1990s
- *Capability Maturity Model Integration (CMMI)* - a comprehensive process meta-model
  - predicated on a set of system and software engineering capabilities that should be present as organizations reach different levels of process capability and maturity
  - Two different ways for representation of process meta-model
    - as a “continuous” model - where a capability rating is computed
    - as a “staged” model- where the model is expressed in terms of capability levels

Continuous CMMI Meta-model

- The CMMI defines each process area in terms of “specific goals” and the “specific practices” required to achieve these goals.
- each process area is rated according to the following capability levels
  - Level 0: Incomplete
  - Level 1: Performed
  - Level 2: Managed
  - Level 3: Defined
  - Level 4: Quantitatively managed
  - Level 5: Optimized
- *Specific goals* establish the characteristics that must exist if the activities implied by a process area are to be effective.
- *Specific practices* refine a goal into a set of process-related activities.

Continuous CMMI Meta-model



- Capability levels
  - Level 0: *Incomplete*
  - Level 1: *Performed*
  - Level 2: *Managed*
  - Level 3: *Defined*
  - Level 4: *Quantitatively managed*
  - Level 5: *Optimized*

Level	Remark on Continuous CMMI Meta-model process area rating level
0:*Incomplete*	The process area is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability for the process area
1: *Performed*	All of the specific goals of the process area have been satisfied. - conducted work products
2: *Managed*	Satisfied all capability level 1 criteria; all work associated with the process area conforms to an organizationally defined policy; all people doing the work have access to adequate resources to get the job done; stakeholders are actively involved in the process area as required; all work tasks and work products are "monitored, controlled, and reviewed; and are evaluated for adherence to the process description"
3: *Defined*	All capability level 2 criteria have been achieved. Process is "tailored from the organization's set of standard processes according to the organization's tailoring guidelines, and contributes work products, measures, and other process-improvement information to the organizational process assets"
4: Q*uantitatively managed*	All capability level 3 criteria have been achieved. Process area is controlled and improved using measurement and quantitative assessment
5: *Optimized*	All capability level 4 criteria have been achieved. Process area is adapted and optimized using quantitative (statistical) means to meet changing customer needs and to continually improve the efficiency of the process area under consideration

Staged CMMI model

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		

- defines five maturity levels, rather than five capability levels
- To achieve a maturity level, the specific goals and practices associated with a set of process areas must be achieved.

Agile Development

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 v.s. processes and tools
- Working software v.s. comprehensive documentation
- Customer collaboration v.s. contract negotiation
- Responding to change v.s. over following a plan

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

55

What is “Agility”?

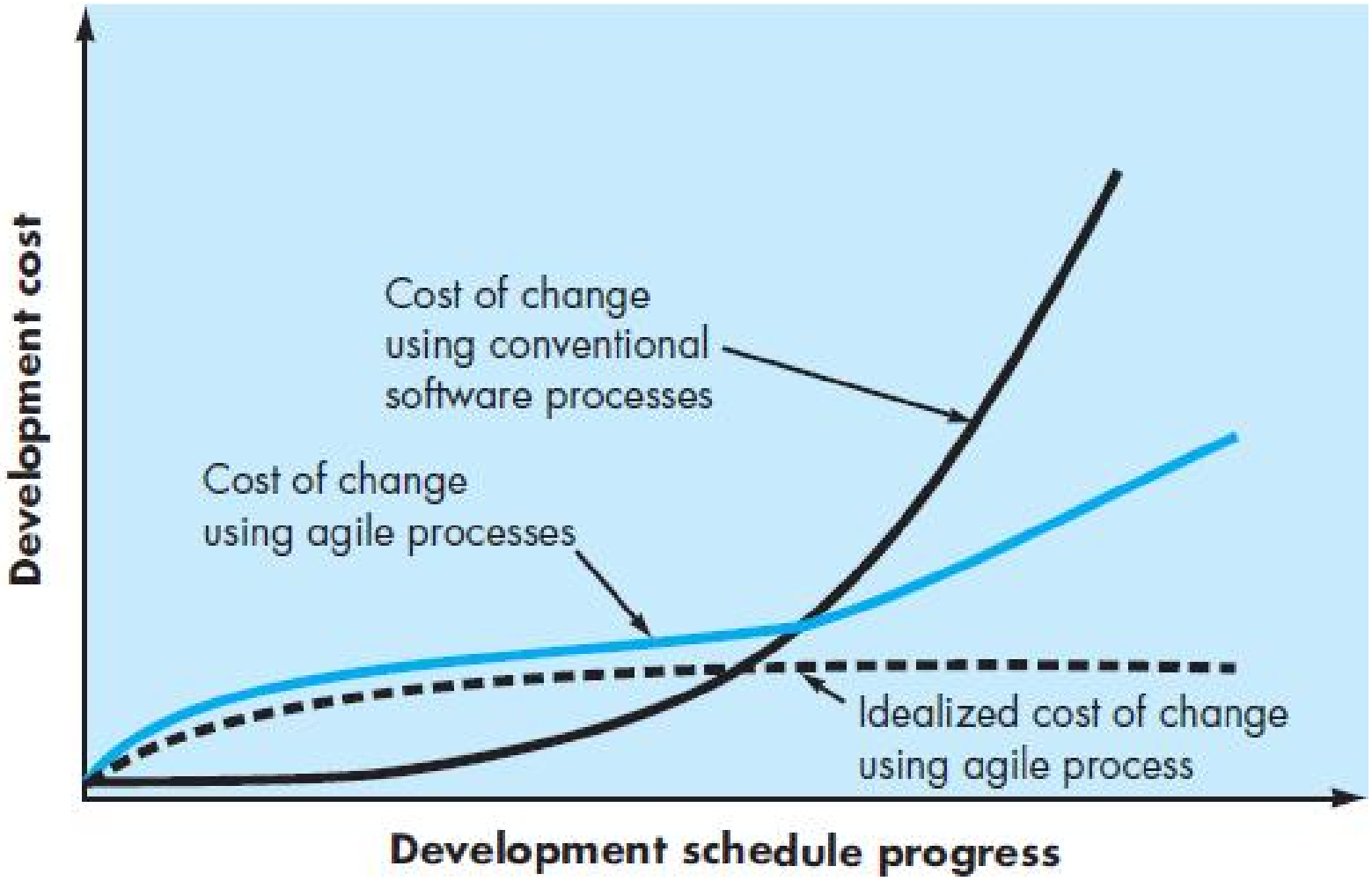
- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders
- Drawing the customer onto the team
- Organizing a team so that it is in control of the work performed

Yielding …

- Rapid, incremental delivery of software

56

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

58

12 Principles of Agile Process

- To satisfy the customer through early and continuous delivery of valuable SW
- Welcome changing requirements, even late in development
- Deliver working SW frequently (from a couple of weeks to a couple of months)
- Business people and developers must work together daily
- Build projects around motivated individuals
- The most effetive method of communication is face-to-face conversation

- Working SW is the primary measure of progress
- Agile processes promote sustainable development
- Continuous attention to technical excellence and good design
- Keep it simple (KIS)
- The best architectures, requirements, and designs emerge from self-organinzing teams
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior

59

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.**

60

The Politics of Agile Development

- There is considerable debate (sometimes strident) about the benefits and applicability of agile software development as opposed to more conventional software engineering processes.
- Like all software technology arguments, this methodology debate risks degenerating into a religious war. If warfare breaks out, rational thought disappears and beliefs rather than facts guide decision making.
- No one is against agility. The real question is: What is the best way to achieve it?
- There are no absolute answers to either of these questions. Even within the agile school itself, there are many proposed process models, each with a subtly different approach to the agility problem.

Extreme Programming

- The most widely used agile process, originally proposed by Kent Beck
- Extreme Programming –
  - is one of the most important software development framework of Agile models.
  - Is used to improve software quality and responsive to customer requirements.
  - model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

Extreme Programming (XP)

- XP Planning
  - Begins with the creation of “user stories” that describe required features and functionality for SW
  - Agile team assesses each story and assigns a cost (in weeks)
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
  - After the first increment “project velocity” (# of user stories implemented during the first release) is used to help define subsequent delivery dates for other increments

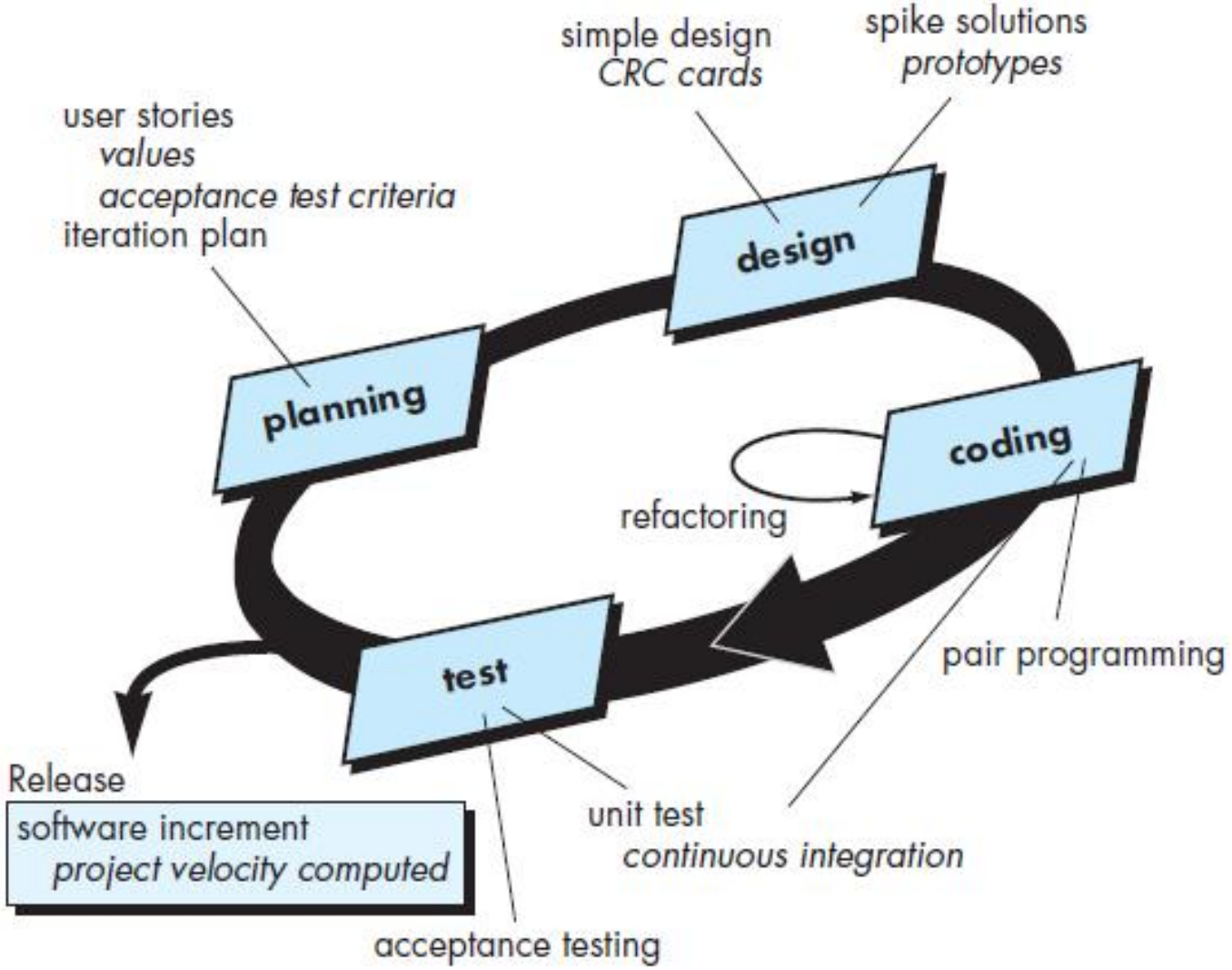
63

Extreme Programming (XP)

- XP Design- Follows the KIS principle
  - Encourage the use of CRC (class-responsibility-collaborator) cards
  - For difficult design problems, suggests the creation of a spike solution (a design prototype)
    - A spike solution is a very simple program to explore potential solutions.
  - 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
    - Test oriented implementation
  - 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

64

Extreme Programming (XP)



65

Industrial Extreme Programming (IXP)

- incorporates six new practices for significant projects within a large organization
  - **Readiness assessment**
  - **Project community**
  - **Project chartering**
  - **Test-driven management**
  - **Retrospectives**
  - **Continuous learning**
- modifies a number of existing XP practices and redefines certain roles and responsibilities to make them more amenable to significant projects for large organizations

Other Agile Process Models

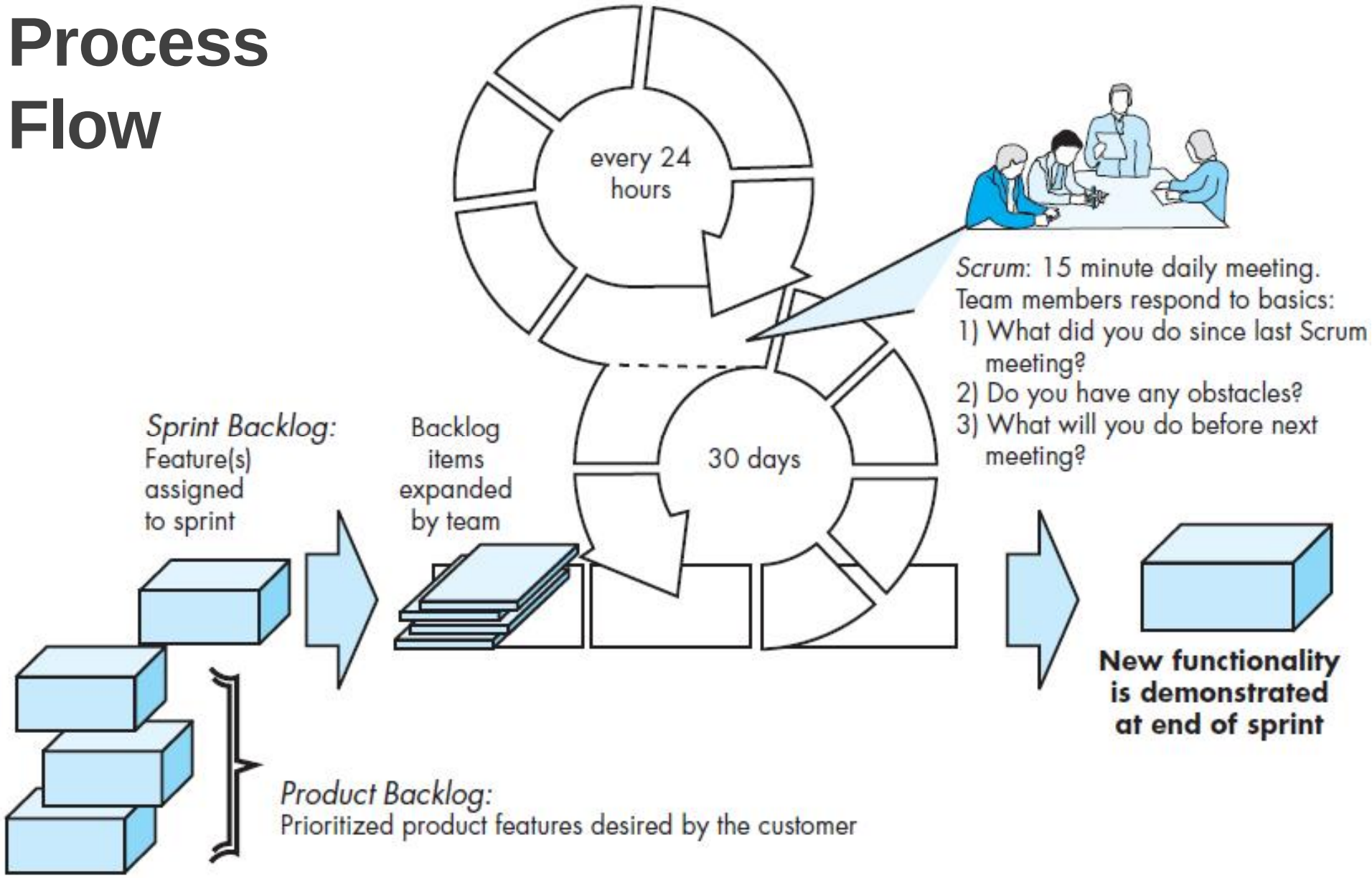
- Many other agile process models have been proposed and are in use across the industry.
- Among the most common are:
  - Adaptive Software Development (ASD)
  - Scrum
  - Dynamic Systems Development Method (DSDM)
  - Crystal
  - Feature Drive Development (FDD)
  - Lean Software Development (LSD)
  - Agile Modeling (AM)
  - Agile Unified Process (AUP)

Scrum

- Originally proposed by Schwaber and Beedle
- Scrum—distinguishing features
  - Testing and documentation are on-going as the product is constructed
  - Work occurs in "sprints" and is derived from a "backlog" of existing requirements
    - Backlog: a prioritized list of project requiremetns or features
    - Sprint: work tasks within a process pattern
  - Daily meetings are very short and sometimes conducted without chairs
  - Demos are delivered to the customer within the time-box allocated

68

Scrum Process Flow



Dynamic Systems Development Method (DSDM)

- Promoted by the DSDM Consortium (www.dsdm.org)
- DSDM—distinguishing features
  - Similar in most respects to XP and/or ASD
  - Nine guiding principles
  - Active user involvement is imperative.
  - DSDM teams must be empowered to make decisions.
  - The focus is on frequent delivery of products.
  - Fitness for business purpose is the essential criterion for acceptance of deliverables.
  - Iterative and incremental development is necessary to converge on an accurate business solution.
  - All changes during development are reversible.
  - Requirements are baselined at a high level
  - Testing is integrated throughout the life-cycle.

70

Agile Modeling

- Originally proposed by Scott Ambler
- practice-based methodology for effective modeling and documentation of software-based systems
- adopts all of the values that are consistent with the agile manifesto
- Suggests a set of agile modeling principles for building large, business critical systems
  - Model with a purpose
  - Use multiple models
  - Travel light - keep only those models that will provide long-term value and jettison the rest
  - Content is more important than representation
  - Know the models and the tools you use to create them
  - Adapt locally

71

Agile Unified Process

- adopts a “serial in the large” and “iterative in the small”
- has historical and technical connections to the Unified Modeling Language
- Activities of Each AUP iteration
  - *Modeling*
  - *Implementation*
  - *Testing*
  - *Deployment*
  - *Configuration and project management*
  - *Environment management*

A Tool Set for Agile Process

- Social- starting even at the hiring stage
- Technological - helping distributed teams simulate being physically present
- Physical - allowing people to manipulate them in workshops
- Collaborative and communication “tools”
  - Active communication is achieved via the team dynamics
  - passive communication is achieved by “information radiators”
- Project management tools - deemphasize the Gantt chart and uses earned value charts or “graphs of tests created versus passed

Tools – support agile approach

- *OnTime, developed by Axosoft (**www.axosoft.com),*** provides agile process management support for various technical activities within the process.
- *Ideogramic UML, developed by Ideogramic (**http://ideogramic-ml.software.informer.com/) is a***UML tool set specifically developed for use within an agile process.
- *Together Tool Set, distributed by Borland (**www.*** **borland.com), provides a tools suite that supports** many technical activities within XP and other agile processes.

Factors	Waterfall	V-Shaped	Evolutionary Prototyping	Spiral	Iterative and Incremental	Agile
Unclear User Requirement	Poor	Poor	Good	Excellent	Good	Excellent
Unfamiliar Technology	Poor	Poor	Excellent	Excellent	Good	Poor
Complex System	Good	Good	Excellent	Excellent	Good	Poor
Reliable system	Good	Good	Poor	Excellent	Good	Good
Short Time Schedule	Poor	Poor	Good	Poor	Excellent	Excellent
Strong Project Management	Excellent	Excellent	Excellent	Excellent	Excellent	Excellent
Cost limitation	Poor	Poor	Poor	Poor	Excellent	Excellent
Visibility of Stakeholders	Good	Good	Excellent	Excellent	Good	Excellent
Skills limitation	Good	Good	Poor	Poor	Good	Poor
Documentation	Excellent	Excellent	Good	Good	Excellent	Poor
Component reusability	Excellent	Excellent	Poor	Poor	Excellent	Poor