COMP 3111
SOFTWARE ENGINEERING

SOFTWARE DEVELOPMENT

LEARNING OBJECTIVES

1. Understand how software is developed in practice.

2. Understand and critique different software development processes.

3. Understand the importance of using an iterative and incremental software development process.

4. Understand the importance of identifying and mitigating risks in software development.

5. Know what are the components of a good software development process.

SOFTWARE DEVELOPMENT OUTLINE

Overview of Software Development

- Nature and Types of Software
- Types of Software Development Projects
- Software Development Life Cycle
- The Four P's in Software Development

Survey of Software Development Processes

- Code and Fix
- Prototyping
- Waterfall
- Spiral
- Phased-release
- Agile
- Unified Process (UP)

THE NATURE OF SOFTWARE

- **Largely _____**
  - Hard to: visualize; asses quality; appreciate development effort.
- **Easy and cheap to _____**
  - Cost is mainly in its development, not its manufacture.
- **Development is _____**
  - Hard to automate the design and programming process.
- **Easy to physically _____ by anyone**
  - May not be well designed; may be hard to find defects or modify correctly!
- **Does not _____ with use**
  - Code and design deteriorates due to defects added when modified.

TYPES OF SOFTWARE

A.

	Copies in use	Development effort	Requirements source
generic	*medium*	*medium*	*market research*
custom	*low*	*high*	*client needs*
embedded	*high*	*low*	*client/hardware needs*

B. **data processing** → organizes and stores _____

real-time processing → controls devices/processes in _____

C. **technical systems** → do not include knowledge of work procedures and processes.

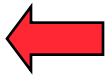
socio-technical systems → include knowledge of work procedures and processes.

Software engineering focuses mainly on the development of technical systems, BUT …

TYPES OF SOFTWARE DEVELOPMENT PROJECTS

1. Green field projects → new development

2. Evolutionary projects → maintenance



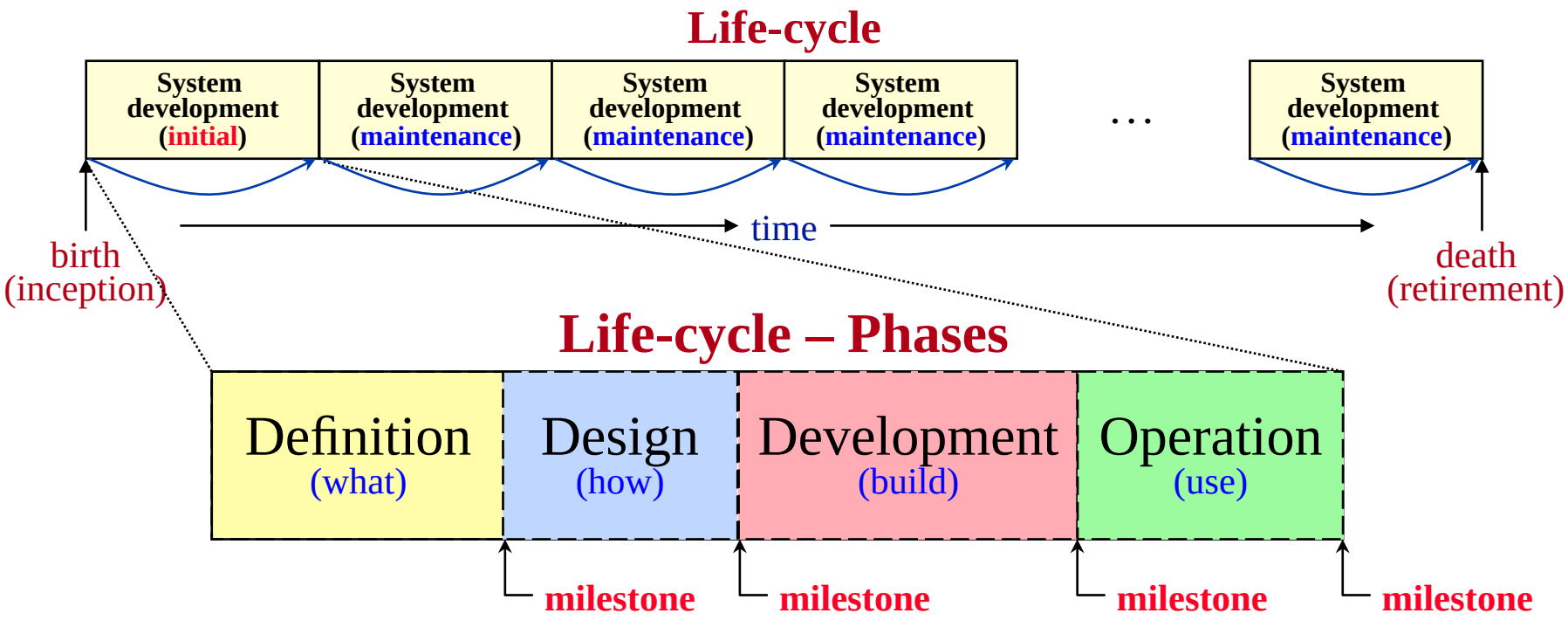
Most common type.

   - corrective – fix defects
   - adaptive – adapt to new technology, new laws, etc.
   - enhancement – add new features
   - perfective (re-engineering) – make more maintainable

3. Framework/component projects → reuse
   - Use an existing framework or plug together several existing components.

     A framework is a software system designed specifically to be reused in different projects or a product line, but needing to be adapted to handle specific requirements.

SOFTWARE DEVELOPMENT LIFE-CYCLE



- The life-cycle concept _____ software development.
- The phases provide a basis for _____
  e.g., milestones, deliverables, etc.

SOFTWARE DEVELOPMENT LIFE-CYCLE (cont'd)

A software system is created by a dynamic process that moves through a series of phases called the **software development life-cycle**.

1. **Definition phase (WHAT)** - The problem is identified, alternate system solutions are studied and recommendations are made about committing the resources required to design the system.

2. **Design phase (HOW)** - The detailed design of the system, selected in the definition phase, is carried out including the allocation of resources to equipment tasks, personnel tasks and programming tasks. The technical specifications are prepared for the performance of all allocated tasks.

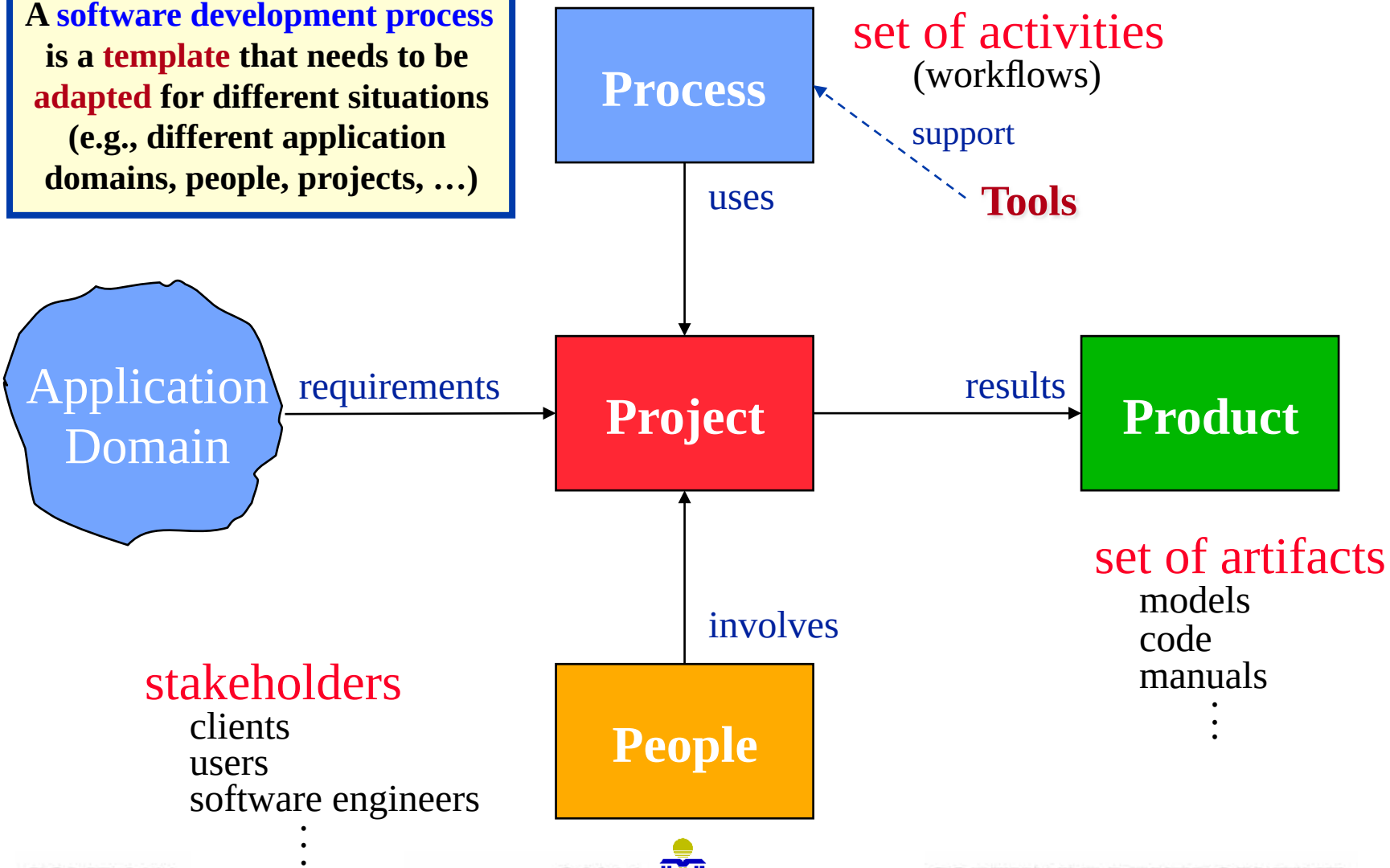
3. **Development phase (BUILD)** - The system is constructed from the specifications prepared in the design phase; the complete system is tested; the equipment is acquired; the manuals are completed; staff is trained.

4. **Operation phase (USE)** - The new system is installed or there is a changeover from the old system to the new system; the system is in use.

milestone: A management decision point that determines whether to proceed to the next phase, cancel the project or redo parts of the previous phase.

THE FOUR P'S IN SOFTWARE DEVELOPMENT

A software development process is a template that needs to be adapted for different situations (e.g., different application domains, people, projects, …)



THE FOUR P'S IN SOFTWARE DEVELOPMENT

Project An instance of a process, carried out by people, that results in the release of a product and that requires management and control.

People Clients, users, and software engineers are involved in a project.

- Software engineers play various roles (also called workers) who are responsible for a set of activities (e.g., analyst, designer, programmer, etc.).

Product All the artifacts produced by a project.

- An artifact is any kind of information created, produced, changed or used in developing the system.
- management artifact: budget, schedules, etc. → usually a short life
- engineering artifact: models, code, manuals, etc. → usually a long life

THE FOUR P'S IN SOFTWARE DEVELOPMENT (cont'd)

Process The complete set of activities (also called workflows) and their sequencing that is needed to transform user requirements into a consistent set of artifacts (software product).

- A process prescribes all of the major activities.
- A process has a set of guiding principles that explain the goals of each activity.
- A process uses resources, subject to a set of constraints (such as a schedule) and produces intermediate and final products.
- A process may be composed of subprocesses that are linked in some way (e.g., a hierarchy of processes)
- Each process activity has an entry and an exit criteria, so that we know when the activity begins and ends.
- The process activities are organized in a sequence, so that it is clear when one activity is performed relative to the other activities.
- Constraints or controls may apply to a process activity, resource or product (e.g., the budget may constrain the time spent in an activity or a tool may limit the way in which a resource may be used).

WHY PROCESS IS IMPORTANT

- Eases project _____
- Allows division of _____
- Promotes teamwork / individual work / _____
- Allows _____ reuse / reassignment
- Eases training
- Promotes productivity / better development

A *process* imposes consistency and structure on the software development activities.

SOFTWARE DEVELOPMENT PROCESSES STAGES

- Most software development processes share the following stages:
  - gathering the system requirements
  - analyzing and designing the system
  - implementing the system
  - testing the system

- They mainly differ in how these stages are:

1. combined
2. emphasized
3. carried out

We want to understand the strengths and weaknesses of different software development processes.

SOFTWARE DEVELOPMENT PROCESSES STAGES

Points to keep in mind as we survey different processes:

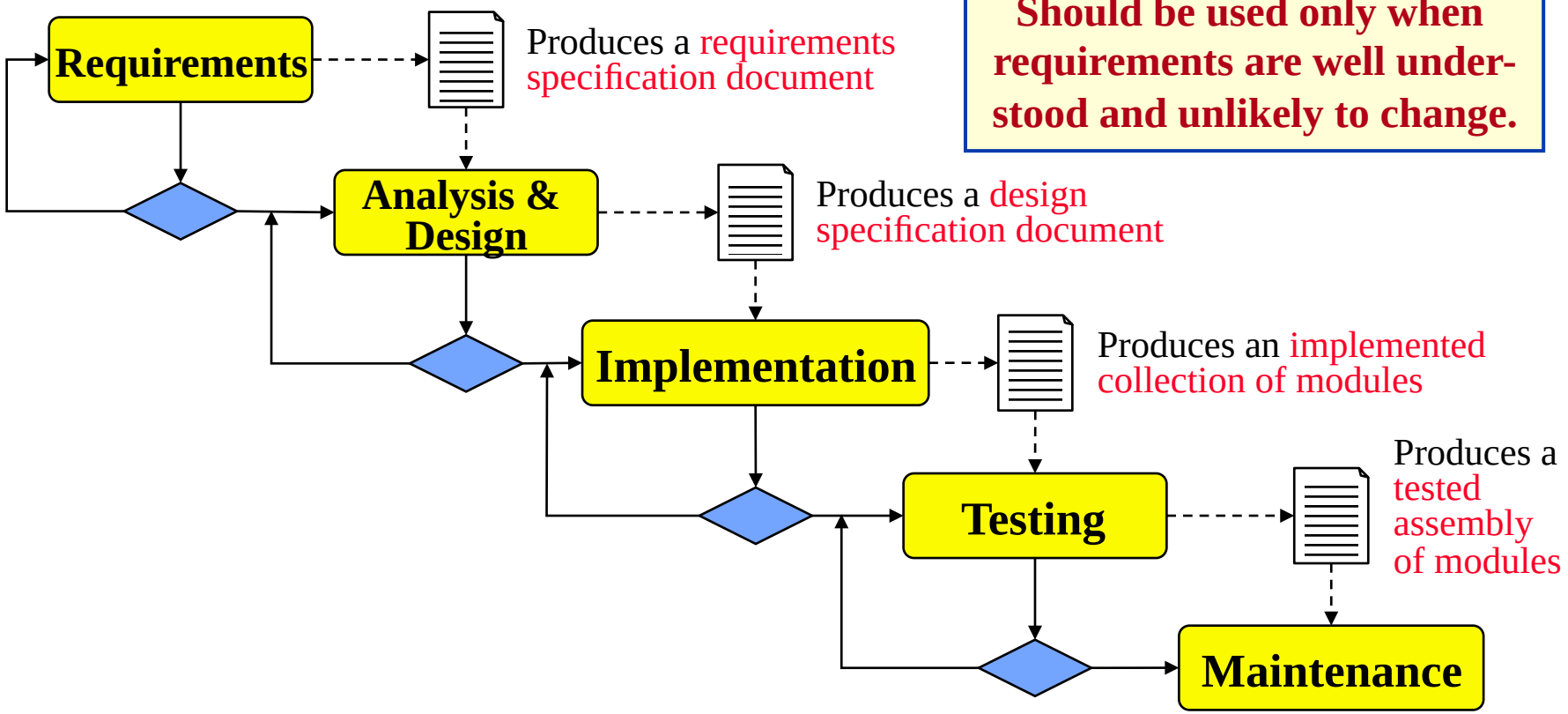
- A software development process is an abstract representation of the life-cycle phases and engineering activities and thus we see the framework of the process, but not the fine details of specific phases or activities.

- Each software development process represents the life-cycle phases and engineering activities from a particular perspective, and thus provides only partial information about the overall software development process.

- The different software development processes are not mutually exclusive and are often used together in practice, especially for large systems development.

- Our purpose here is to understand the strengths (pros) and weaknesses (cons) of different software development processes.

WATERFALL PROCESS



Should be used only when requirements are well under-stood and unlikely to change.

Plus: reviews (for correctness, standards), deliverables (documentation, code, training material, …), …

WATERFALL PROCESS: PROS & CONS

Pros

- Imposes needed discipline (rigor and formality).
- Keeps development predictable and easy to monitor.
- Enforces documentation standards and approval of documents before proceeding.
- Fits well with other engineering process models (e.g., hardware development).

Cons

- Assumes linear, sequential development is possible.
- Rigid assuming results of each phase can be frozen before proceeding to the next phase.
- Different languages/notations often used in each phase.
- Makes little provision or opportunity for user feedback, which is a source of high risk.

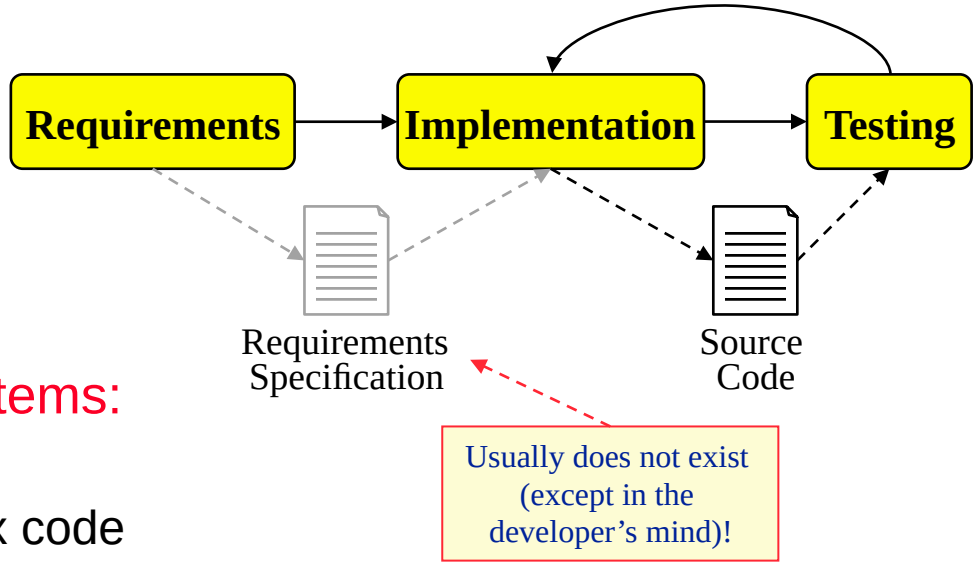
SOFTWARE DEVELOPMENT OUTLINE

- ✓ Overview of Software Development
  - Nature and Types of Software
  - Types of Software Development Projects
  - Software Development Life Cycle
  - The Four P's in Software Development

- ➔ **Survey of Software Development Processes**
  - Monolithic
    - Waterfall
  - ➔ **Iterative and Incremental**
    - **Code and Fix**
    - **Prototyping**
    - **Spiral**
    - **Phased-release**
    - **Agile**
    - **Unified Process (UP)**

CODE-AND-FIX PROCESS

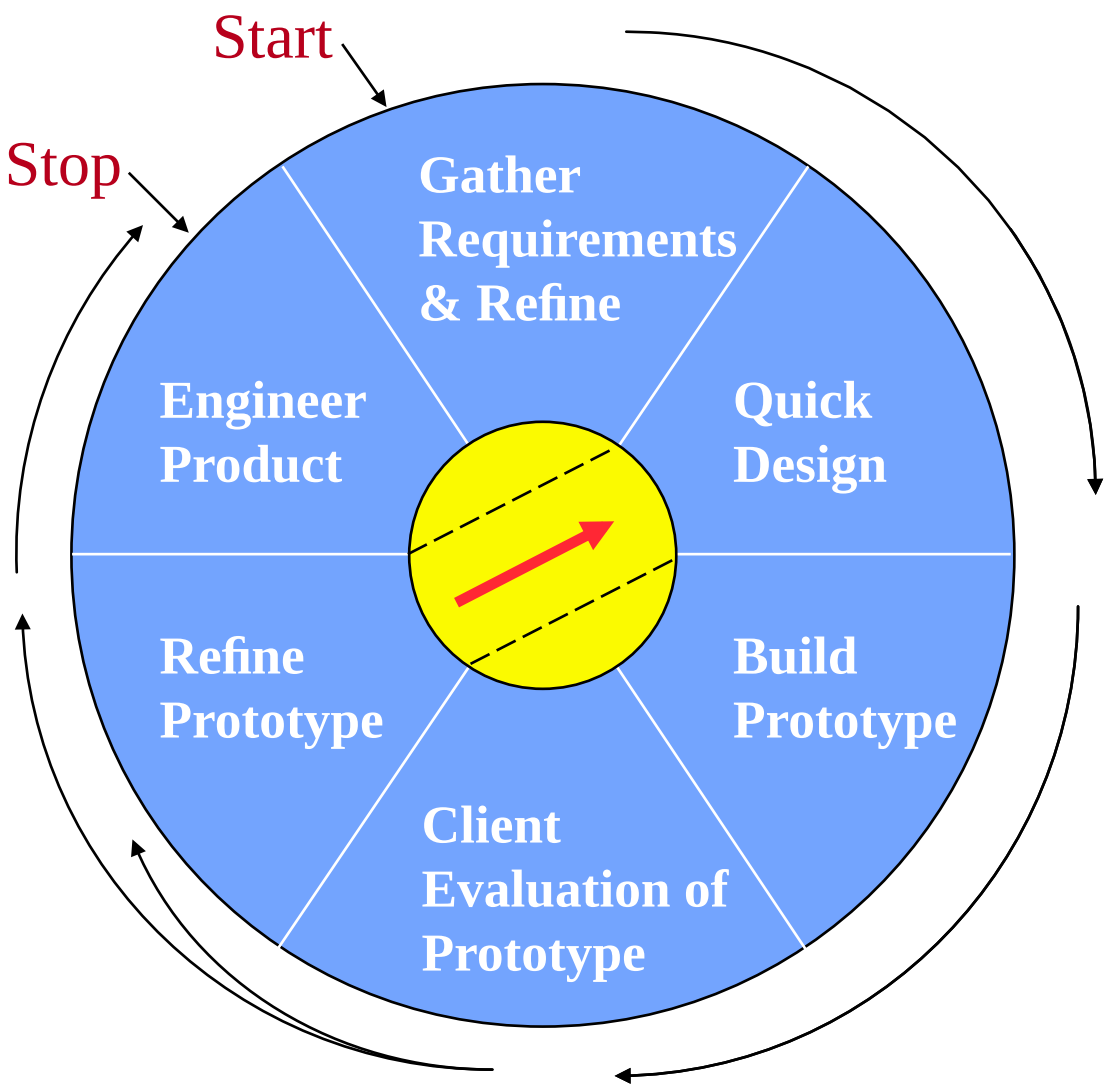
- Many changes
  - ☞ code structure often becomes messy



- Unsuitable for large systems:
  - turnover of personnel
  - difficult to understand/fix code
  - requirements can easily be unmatched

- The software development process becomes:
  - unpredictable and uncontrollable
  - over schedule, over budget and fails to meet expectations

PROTOTYPING PROCESS



- Basically a code-and-fix process, **BUT** includes client evaluation and enforces some discipline.
- Useful when requirements are vague or unknown as it allows exploration of
  - functionality needed
  - user interface

What to do with the final prototype?

PROTOTYPING PROCESS: PROS & CONS

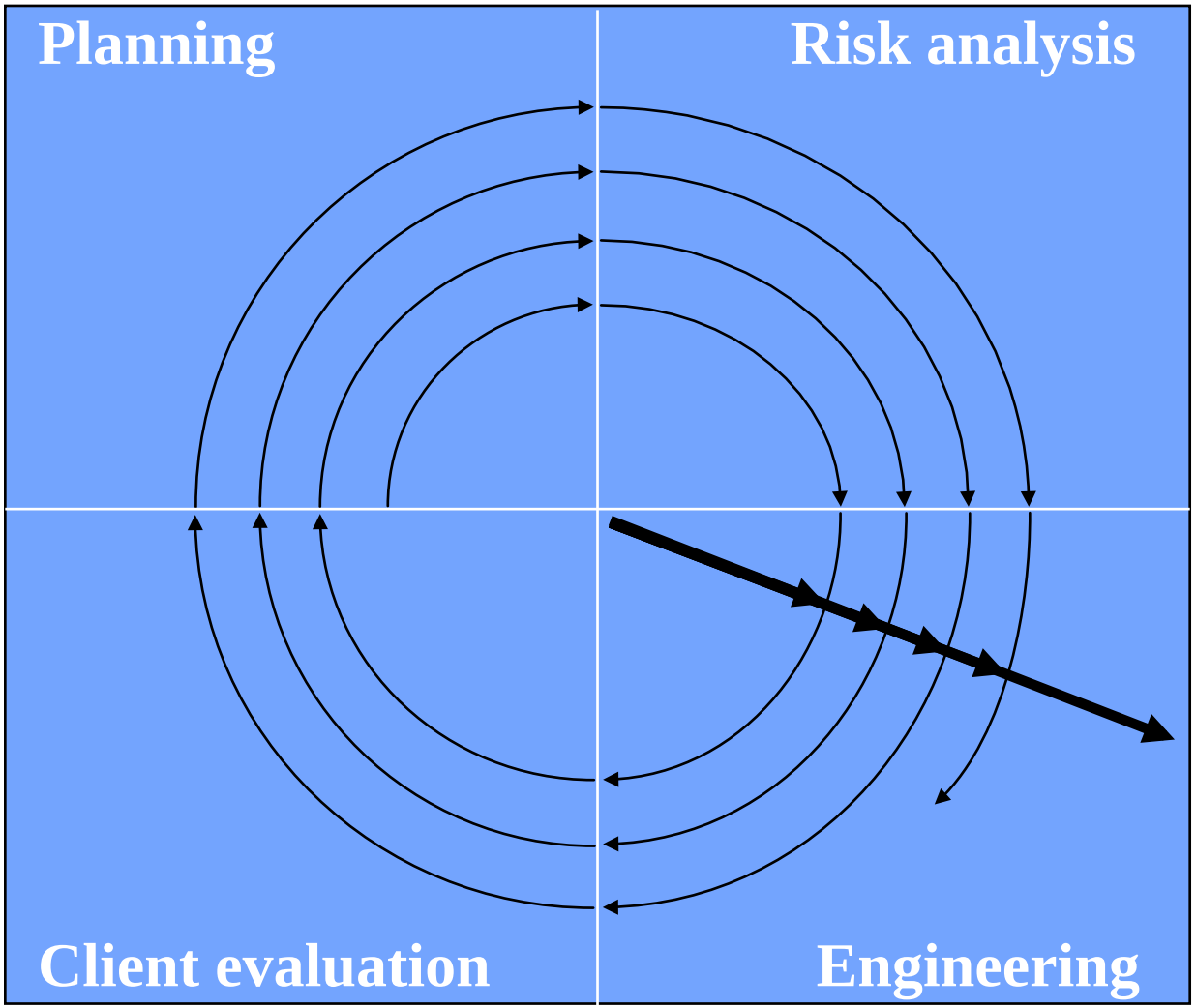
Pros

- Allows requirements to be quickly explored.
- Allows user feedback and approval to be obtained.
- Allows different solutions to be explored.

Cons

- It is not really a complete software development process.
- The process is not visible making progress hard to measure.
- Documentation is often sparse or completely absent.
- The final “product” is not a complete system.

SPIRAL PROCESS



Go, no-go decision

Toward a completed system

SPIRAL PROCESS: RISKS

RISK

Anything that can go wrong (endanger success) in a project.

Technical risks

- building the right system
- system architecture
- new technologies
- performance

Non-technical risks

- right expertise
- needed training
- tight schedule
- timely approvals

Dealing with risks

- avoid (re-plan or change requirements)
- confine (restrict the scope of its effect)
- mitigate (devise tests to see if it occurs)
- monitor (constantly be on the lookout for it)

GOAL: Prioritize and deal with biggest risks as early as possible.

SPIRAL PROCESS: PROS & CONS

Pros

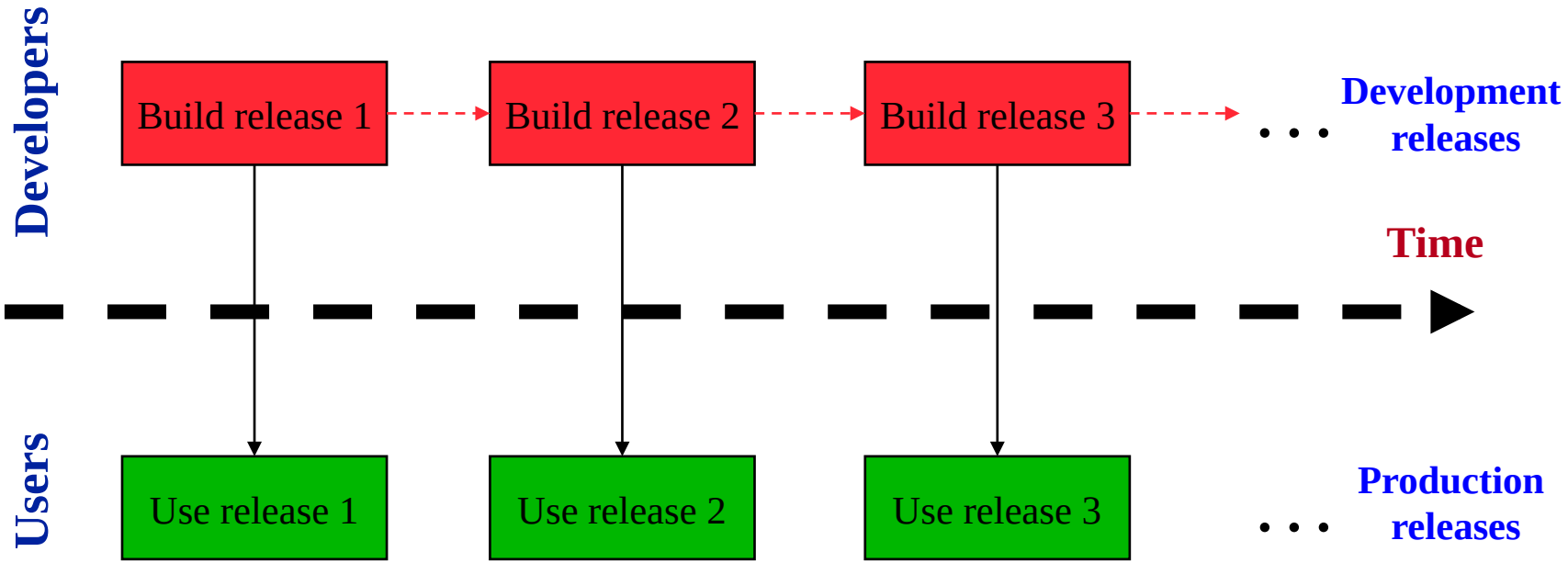
- Risk evaluation can help reduce development problems.
- Planning and client evaluation phases help the product better meet client expectations.
- Iterative and incremental planning, engineering and evaluation facilitates project management.

Cons

- Relies on expertise in risk assessment.
- Needs more elaboration of the phases (i.e., specific activities that should be performed).
- More appropriate for internal development than contract development.

PHASED-RELEASE PROCESS

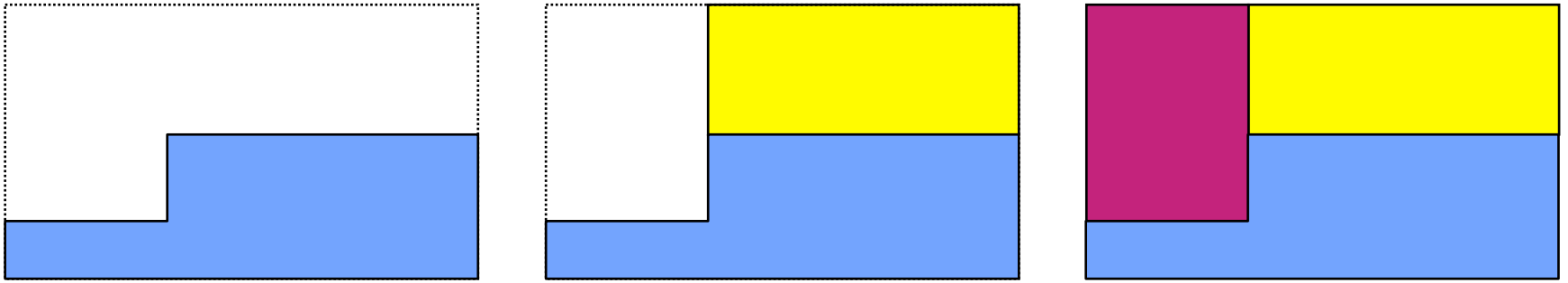
Premise: Change is inevitable, so plan for it!



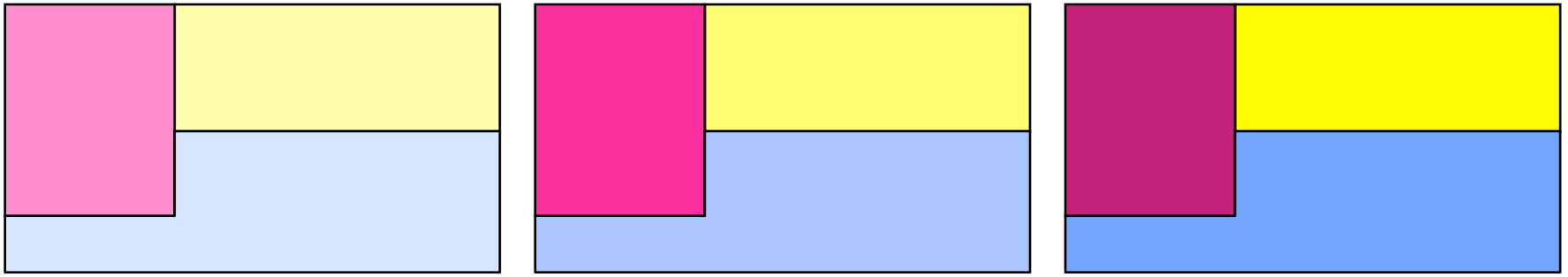
Releases are developed and used in parallel.

PHASED-RELEASE PROCESS (cont'd)

incremental development → partial system; full functionality



iterative development → full system; partial functionality



Many organizations use a combination of iterative and incremental.

PHASED-RELEASE PROCESS: PROS & CONS

Pros

- Reduces the risk of project failure.
- Promotes system modularity.
- Allows frequent releases.
- Allows appropriate expertise to be applied.
- Allows early training and feedback.

Cons

- The system pieces need to be relatively small.
- It may be hard to identify common facilities needed by all pieces.

AGILE PROCESS

- Any phased (incremental) approach where the emphasis is more towards the items on the left.

← more important	less important →
individuals and interactions	*processes and tools*
working software	*comprehensive documentation*
client involvement/collaboration	*contract negotiation*
responsiveness to change	*following a plan*

☞ **This does not imply that there is no value in the less important items!**

AGILE PROCESS (cont'd)

- **Methods**
  - Extreme Programming (XP)
  - Scrum

- **Practices**
  - Planning poker → used to estimate time required to implement a feature (see http://en.wikipedia.org/wiki/Planning_poker)
  - Pair programming → used to write code for a feature
  - Test Driven Development (TDD) → used to test the code

AGILE PROCESS: EXTREME PROGRAMMING (XP)

- Requirements and analysis:
  - *developer* determines features needed
    estimates time and cost for each feature
  - *client* selects features to be included in each iteration

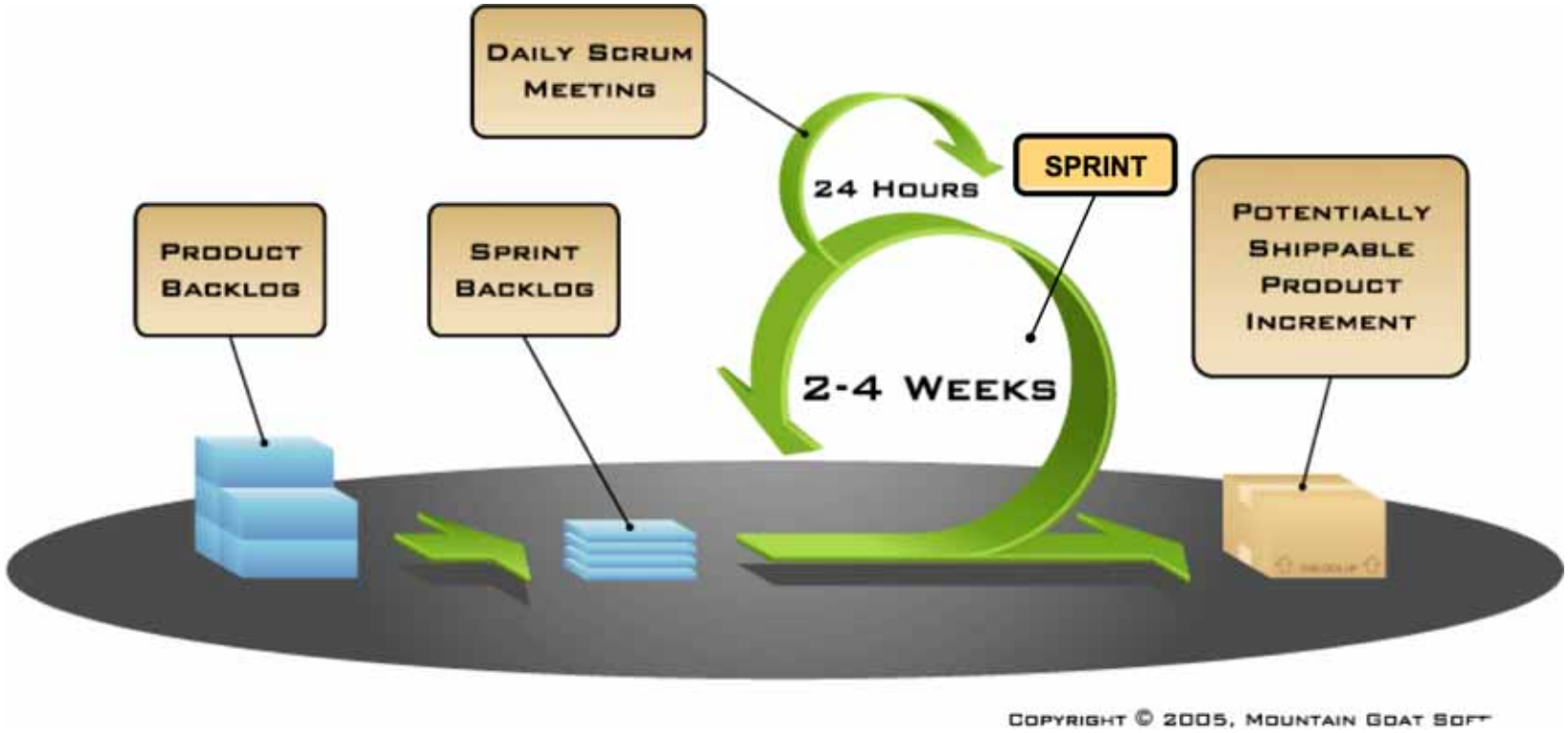
- Implementation (by iterations/sprints):
  - the *developer* breaks each iteration into tasks
  - for each task (where tasks can be carried out in parallel) the *developer*
    - designs test cases (*test-driven development*)
    - implements the task using *pair programming*
    - integrates the task into the current product

The major emphasis is here.

AGILE PROCESS: SCRUM

- Scrum is an agile software development process that mainly specifies what you should do to develop a software product.
- No specific software engineering practices are prescribed for developing the product; the team needs to decide how to do it.
- The requirements are captured as items in a "product backlog"; the product owner (client) sets the priorities for the items.
- The software product is developed in a series of iterations called "sprints".
- Teams self-organize to determine the best way to deliver the product.

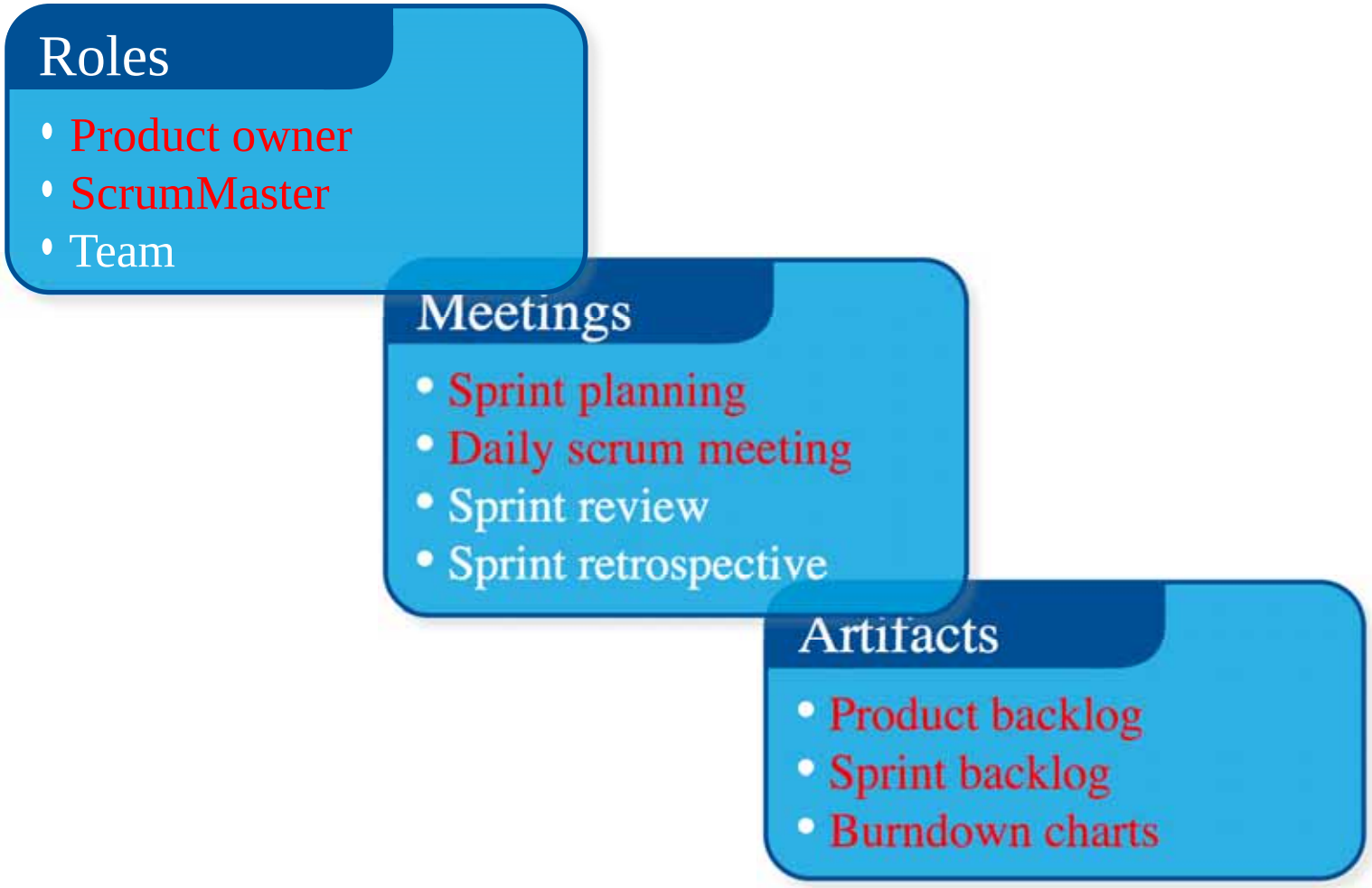
SCRUM: SPRINT WORKFLOW



- The software product is designed, coded and tested during the sprints.

- The requirements are not allowed to change during a sprint.

SCRUM: FRAMEWORK



SCRUM: ROLES

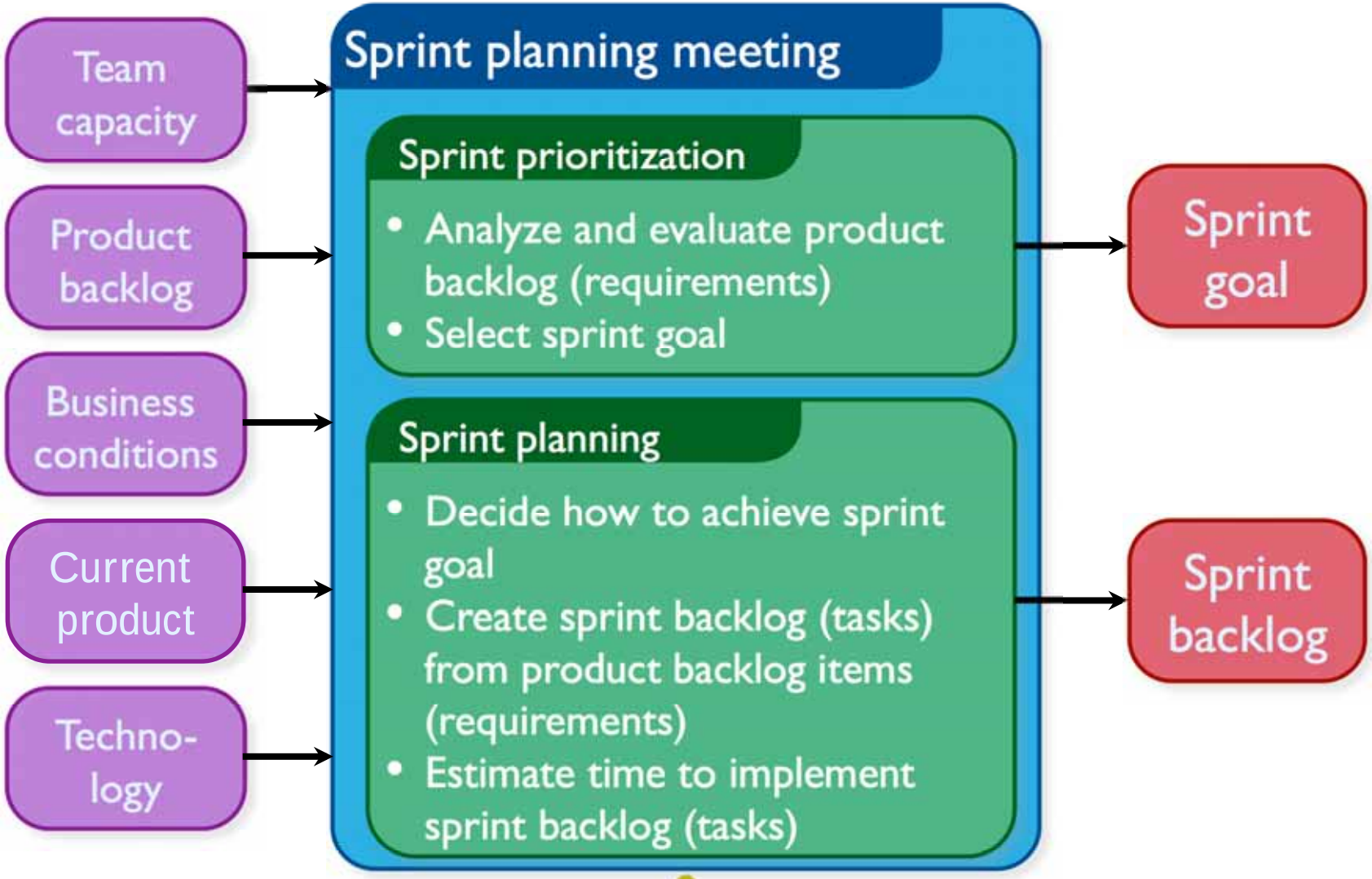
Product Owner (aka Client)

- Is the key stakeholder (represents users, client)
- Defines and prioritizes the requirements of the product.
- Adjusts requirements and priority every iteration, as needed.
- Decides on the release date and content.
- Accepts or rejects work results.

ScrumMaster (aka Project Manager / Team Leader)

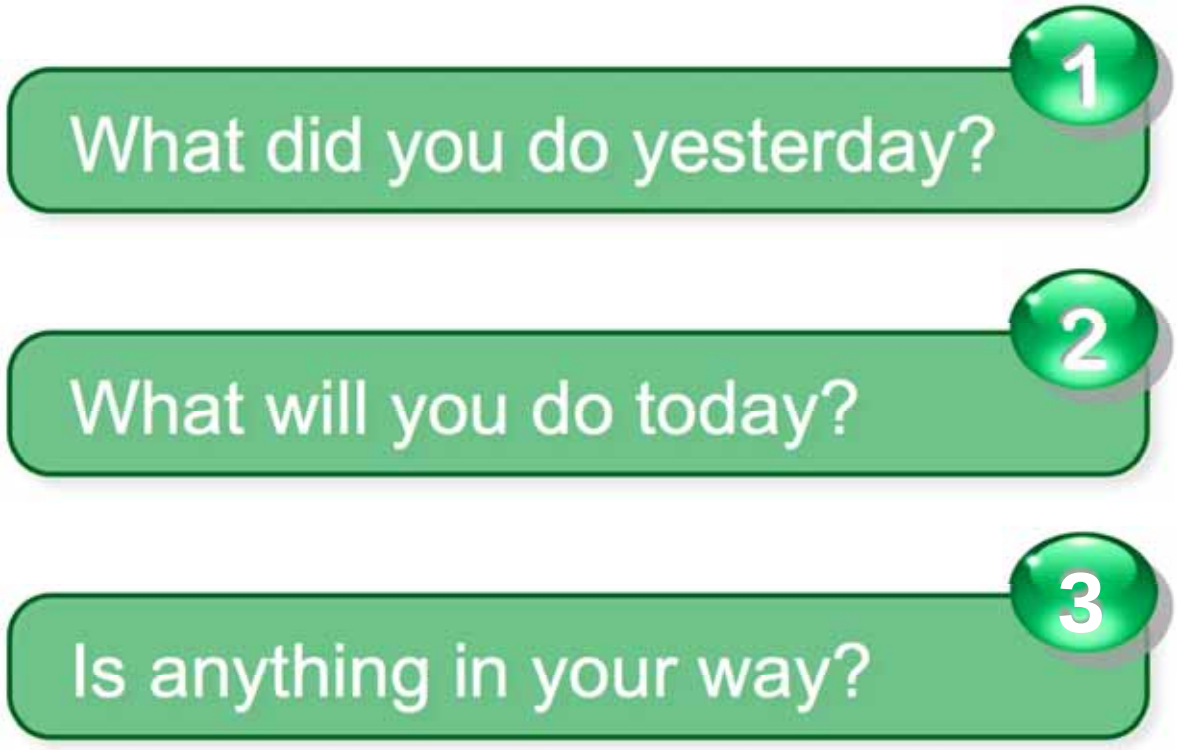
- Responsible for enacting Scrum values and practices.
- Ensures that the team is fully functional and productive.
- Enables close cooperation across all roles and functions.
- Removes impediments to progress and shields the team from external interferences.

SCRUM: SPRINT PLANNING MEETING



SCRUM: DAILY SCRUM MEETING

- A team meeting in which everyone answers three questions:



SCRUM: ARTIFACTS

Product Backlog

- Represents the requirements of the system (i.e., a list of all desired functionality of the system).
- Ideally expressed such that each item has value to the users or customers of the product.
- Items in the backlog are prioritized by the product owner (client) and reprioritized at the start of each sprint.

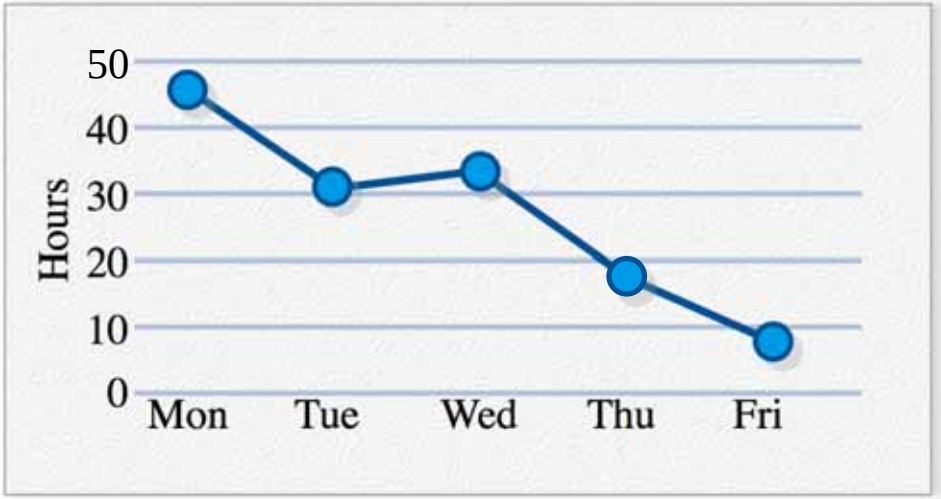
Sprint Backlog

- Contains items selected from the product backlog based on item priority and on how much the team thinks they can do in a sprint.
- A product backlog item may become several sprint backlog tasks.
- Team members select sprint backlog items to work on during the sprint.

SCRUM: ARTIFACTS (CONT'D)

Burndown Chart

Tasks	Mon	Tue	Wed	Thu	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				



The burndown chart graphically shows the *total hours remaining* each day to complete the sprint.

AGILE PROCESS: PROS & CONS

Pros

- The development is adaptable to changing requirements (flexible).
- Immediate feedback is provided by the client/users.
- Results in faster speed-to-market.
- There are fewer defects in the final product.

Cons

- Active user involvement and close collaboration are required.
- There is often a lack of documentation.
- There can be scope creep as the client/users add requirements.
- Daily stand-up meetings can take a toll.

EXAMPLE: A BUDGET CONTROL SYSTEM

Problem: Build a budget control system for a small (~30 person), high-tech software consulting and development company that will monitor whether the financial transactions involved in their various software projects are proceeding according to the original budgets.

☞ **Want to take corrective action early if not on track.**

Scenario: In general, the budget control activity often needs to be customized to the particular activities of an organization. While budget control is related to other administrative activities, (e.g., payroll processing, income and expense monitoring, etc.), unlike these, budget control is based both on objective data, such as actual time and costs expended, and on subjective data, such as estimates of the value of the "work in progress". As staff may be involved in several projects at the same time, and no log is kept about their contribution to each project, it is hard to estimate “work in progress” costs for each project.

EXAMPLE: A BUDGET CONTROL SYSTEM

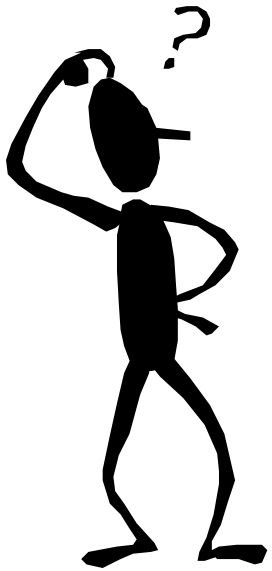
Initial findings

- The current administrative system is not suitable for providing all of the data required for budget control since the required data is either missing or in the wrong format.
- The real problem is not budget control *per se*, but understanding what it is in this company.
- Difficulties of developing a budget control system are related to:
  - unusual nature of the activities of the company (not standard).
  - lack of standard production rules (e.g., for estimating value of "work-in-progress").
  - the need to often re-schedule and re-budget most activities.
  - personnel turnover.

A precise statement of requirements is not possible.

EXAMPLE: A BUDGET CONTROL SYSTEM

1. Which of the software development processes, or combinations of processes, listed below would you use to develop the Budget Control System?
2. What are the specific characteristics and/or problems of this software development project that cause you to make this choice and how does your choice address these characteristics/ problems?

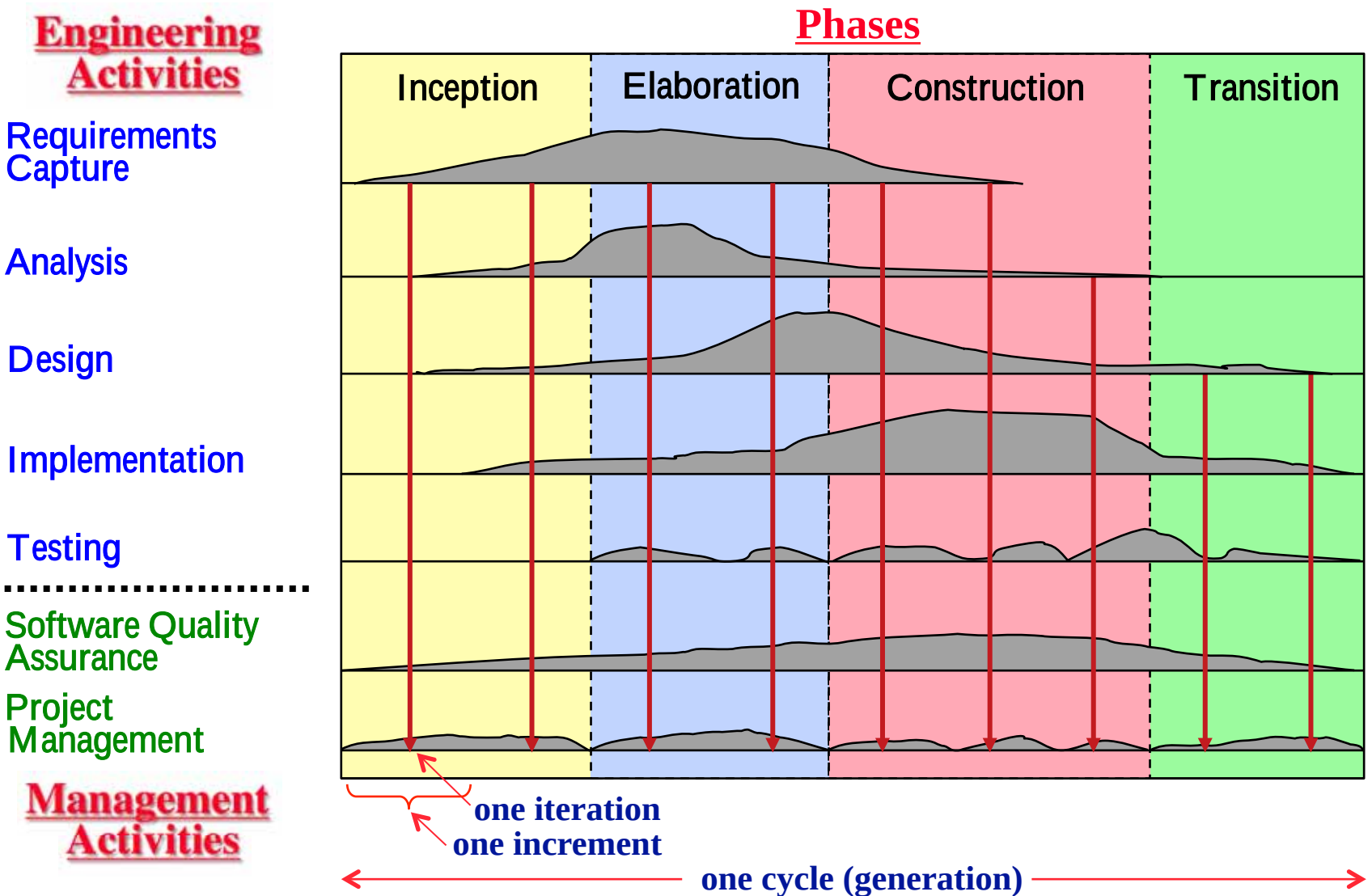


- **Waterfall**
- **Code-and-Fix**
- **Prototyping**
- **Spiral**
- **Phased-release**
- **Agile**

SOFTWARE DEVELOPMENT PROCESS: PRINCIPLES

- **rigor and formality** (Waterfall; Spiral)
- **separation of concerns and modularity** (Waterfall; Spiral; Phased-release)
- **abstraction and generality** (Waterfall; Spiral)
- **anticipation of change** (Spiral; Phased-release; Agile)
- **incremental development** (Prototyping; Spiral; Phased-release; Agile)
- **risk assessment** (Spiral)

UNIFIED PROCESS (UP): LIFE CYCLE



UNIFIED PROCESS (UP):
LIFE CYCLE PHASES AND GOALS

Inception – determines the system's *life cycle objectives*

- establish feasibility → validate: technical decisions, business requirements
- create a business case → show a quantifiable business benefit
- capture the essential requirements to set the system scope
- identify critical (most important) risks

Elaboration – determines the system's *life cycle architecture*

- create an executable architectural baseline for the system
- refine the risk assessment (deal with second most important risks)
- define the system's quality attributes → design goals
- capture ~80% of the functional requirements
- create a detailed construction phase plan
- estimate needed: resources, time, equipment, staff, cost

Construction – builds the *initial operational system*

- complete all requirements, analysis and design
- evolve the architectural baseline into the final system

Transition – *releases the product* to the client

- correct defects
- prepare the user site for the new software and install it
- tailor the software to operate at the user site and/or modify it if needed
- create user manuals and other documentation
- provide user consultancy/training
- conduct a post project review

UNIFIED PROCESS (UP): MAIN FEATURES

The UP selects from the best practices of previous processes to:

- **provide a generic process framework**
  - ☞ It needs to be instantiated/specialized for specific application areas, organizations, competence levels, project sizes, etc.
- **define a set of activities (workflows)**
  - ☞ The workflows transform users' requirements into a software system.
- **define a set of models (artifacts)**
  - ☞ Models range from abstract (user-level) to concrete (code).
  - ☞ Models are transformed by the workflows into other models.

Each iteration results in a working product.
Each increment establishes a system baseline.

SOFTWARE DEVELOPMENT: SUMMARY

- A software development process needs to consider both management and engineering issues.

- A software development process needs to consider the characteristics of the:
  - organization → size; access to users/client; need for formality.
  - project → small/large; vague/well-defined; novel/well-known.
  - people → availability of expertise; skill of developers.

- The *Unified Process* incorporates best practices of previous software development processes.

The Unified Process provides a *generic framework* to discuss software development activities.