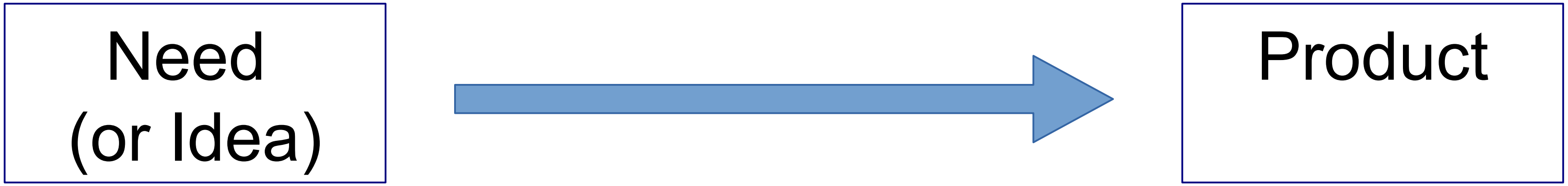
Intro to Software Processes

and the Software Development Life Cycle

Goal of Software Development

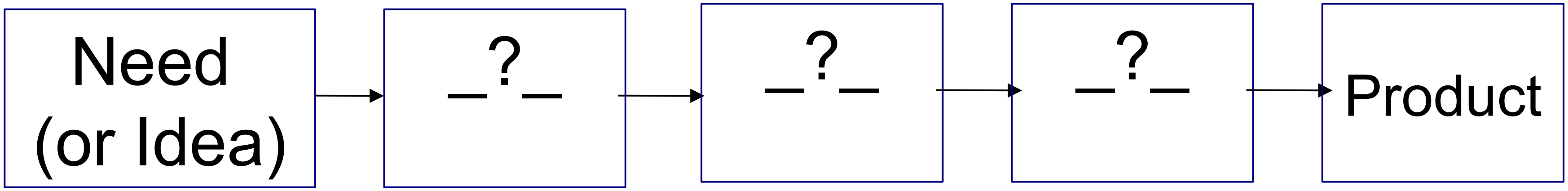


Need
(or Idea)

Product

Produce a software product that fulfills a need or realizes an idea.

What are the Steps?



Need (or Idea)

_?_

_?_

_?_

Product

What are the major steps or **activities** you would need to do?

List major **activities** that would apply to almost any software project.

Activities in Software Development

Not necessarily in the order they are performed.

1. ____________________

2. ____________________

3. ____________________

4. ____________________

Activities

Creating software involves

- elicit requirements
- analysis & specification
- design
- construction & testing
- validation
- documentation
- maintenance
- enhancement

Managing the project involves

- planning
- obtaining resources
- tracking progress
- resolving problems
- analyzing results
- closing the project

Process

Process -

a [systematic] series of actions to achieve a particular result

Software process - a method for producing software

Software Process according to experts

A software process is a sequence of activities that leads to production of a software product.

-- Ian Summerville, *Software Engineering*, 9 Ed.

...a collection of activities, actions, and tasks that are performed to create [software].

-- Roger Pressman,
Software Engineering: A Practitioner's Approach, 7 Ed.

Do You Have a Software Process?

What is your software process?

(discussion)

What did you do to create:

- Programming 2 project?
- Exceed Camp project?

Do You Have a Software Process?

Yes!

Everyone who creates software uses a process.

Do You Have a Software Process?

"***I never thought about it***"

,,, the process is *implicit* or *informal*

"***It's different for each project***"

... *ad hoc* process

Why Define a Software Process?

Why not *just do it*? *(like Nike)*

Realities of Software

Software projects are plagued by defects, over-budget, schedule overrun, and complete failure.

Why?

1. Change can occur almost any time in a project.

2. Software is complex.

3. Software must *evolve* over time (more change)

4. Communication problems

    - between developers and customer

    - within development team

    - implicit assumptions are often not true

Common Project Outcomes (failures)

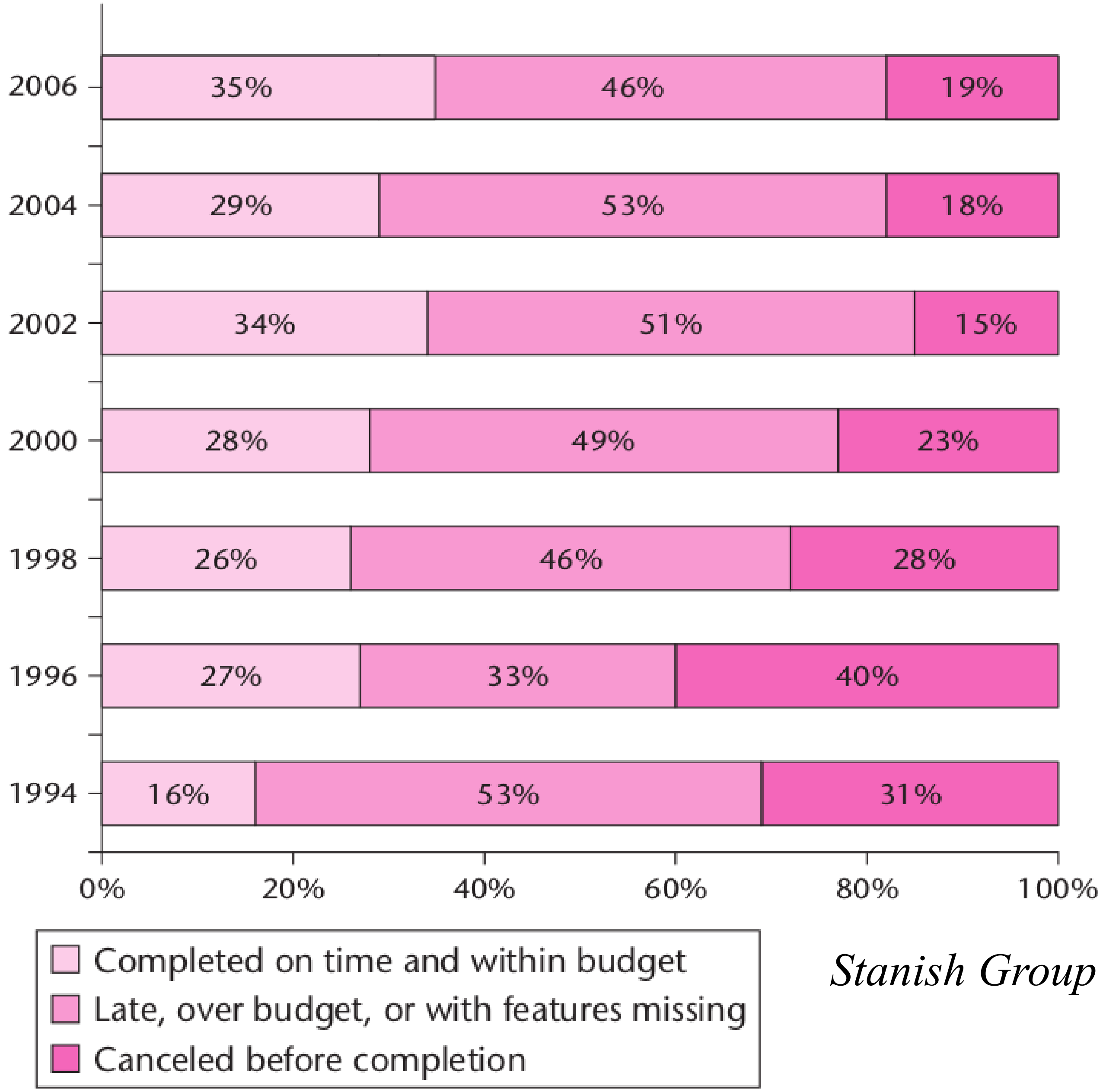
1. Project is late and over-budget.

2. Software does not do what customer wants.

3. Excessive defects.

4. Project is canceled.

Software Project Failure over Time



Stanish Group annual CHAOS report

Britain Abandons NHS IT Project

After 10 years and 11 Billion pounds (450,000,000,000 Baht), the British government abandoned a huge IT project for the National Health System (NHS) in 2011.

Some components continue to be developed, but they are all late and over-budget.

Why? What Happened?

https://www.henricodolfing.com/2019/01/case-study-10-billion-it-disaster.html

https://www.computerweekly.com/opinion/Six-reasons-why-the-NHS-National-Programme-for-IT-failed

Microsoft Windows Critical Flaws

In 2020, Microsoft set new records for the number of critical vulnerabilities disclosed & patched *each month*.

Microsoft programmers have been working on Windows code for almost 20 years.

...but Windows still contains **thousands** of (unknown) critical vulnerabilities.

Why?

** Assuming Windows 7 as the start for current code base*

Causes of Project Failure

1. Poor communication.
2. Unclear requirements.
3. Excessive change in requirements.
4. Unwillingness to accept change.
5. Not monitoring *actual* progress regularly.
6. Unrealistic schedule or budget.
7. Forced deadlines.
8. Insufficient developer skills.

process
related

Benefits of a Defined Process

- **Save Time** - don't rediscover how to perform each project

- **Enable Planning** and **Tracking**

- **Basis for Estimation** - you collect data for each activity and task from previous projects and learn

- **Repeatable** results

- **Process Improvement** - it must be defined before you can examine and improve it

4 Factors in Development Speed

1. **People**

   ability, knowledge, skills, motivation

2. **Process**

   promotes effective work or hinders it

   helps team stay on track? quality focus?

3. **Product**

   Size and characteristics, nature of requirements

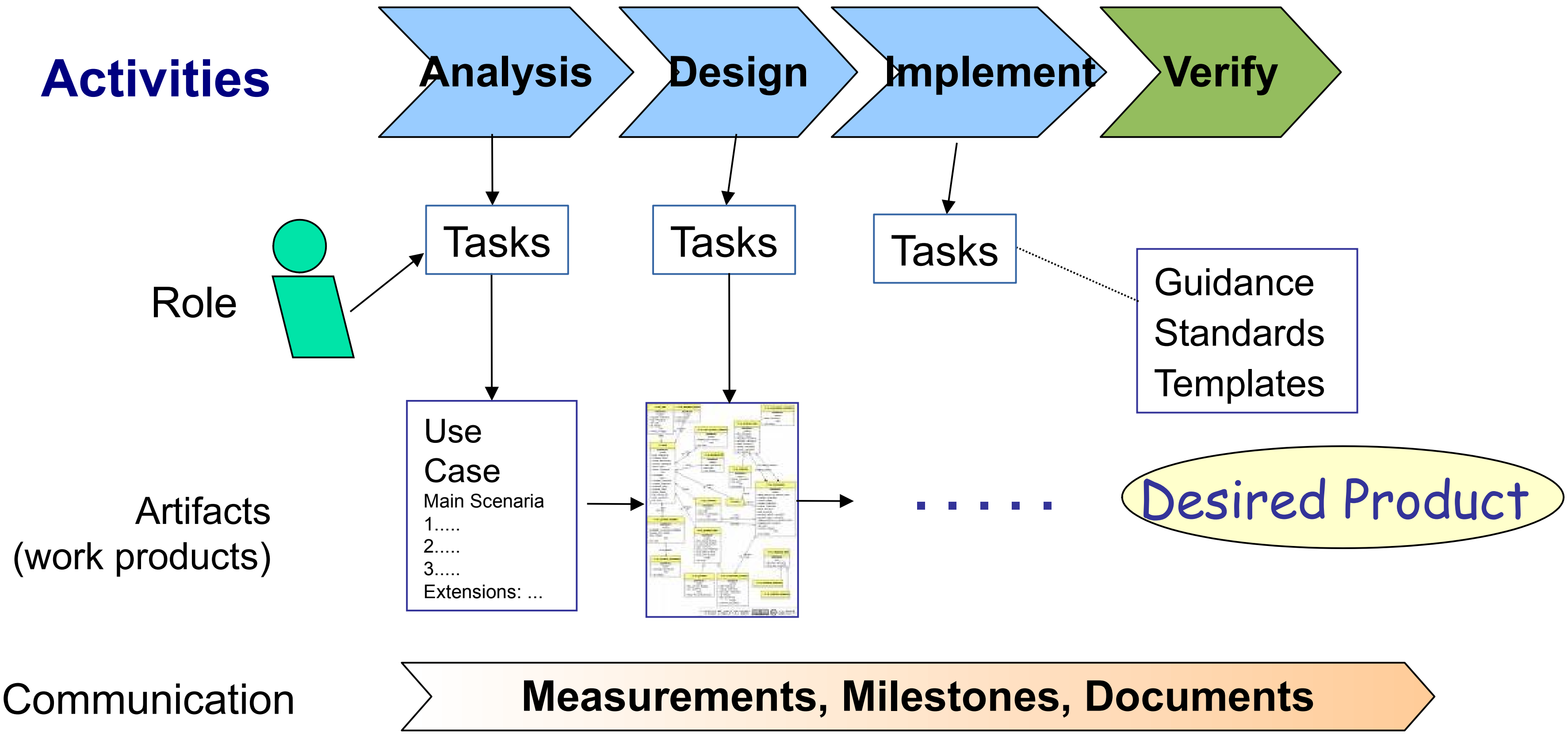
4. **Technology**

   Language and software frameworks

   Tools

Software Process **Model**

Process consists of activities



Activities

Analysis **Design** **Implement** **Verify**

Role

Tasks

Tasks

Tasks

Guidance
Standards
Templates

Artifacts
(work products)

Use
Case
Main Scenaria
1.....
2.....
3.....
Extensions: ...

. . . . .

Desired Product

Communication

Measurements, Milestones, Documents

Activities

Activities are large(r) scopes of work. They may be general things that occur repeatedly.

Major activities:

- requirements specification
- modeling & design
- construction
- validation
- deployment

[Major activities listed by Summerville & Pressman.]

Tasks

Activities are large and general.

Activity is broken down into concrete **tasks**.

Some tasks during Construction:

- iteration planning
- backlog selection & estimation
- detail design
- coding
- unit testing
- integration testing

Activity May Subdivide into 2 Levels

In Pressman, an activity consists of **actions** divided into **tasks**.

Activity: Construction

Action: *iteration (or sprint) planning meeting*

Tasks:

- review & prioritize items in product backlog
- select items for this iteration (sprint)
- estimate time for each item
- define a "done" criterion (acceptance test) for each
- software design to implement the items

How to do it? What to produce?

"Activities", "actions", and "tasks" should make *progress toward finishing* the project.

What to do?

A task has a description & guidance

What is the result?

Every task should have an output -- a work product

Is the work correct?

Define how to evaluate the work product

Example Task

Title: `Add Item to Cart` **Priority**: `High` **Est**: `8 hr`

Description:

When a visitor navigates to item detail page, there is an "*Add to Cart*" button on the page. When visitor clicks "Add to Cart", a unit of the item is added to his shopping cart.

Acceptance Criteria

Given that user is viewing an in-stock item

When he clicks "Add to Cart"

Then the item is added to his shopping cart.

When navigate to "My Cart" page, the item, with quantity and price, are shown.

Common Process Models

Code and Fix

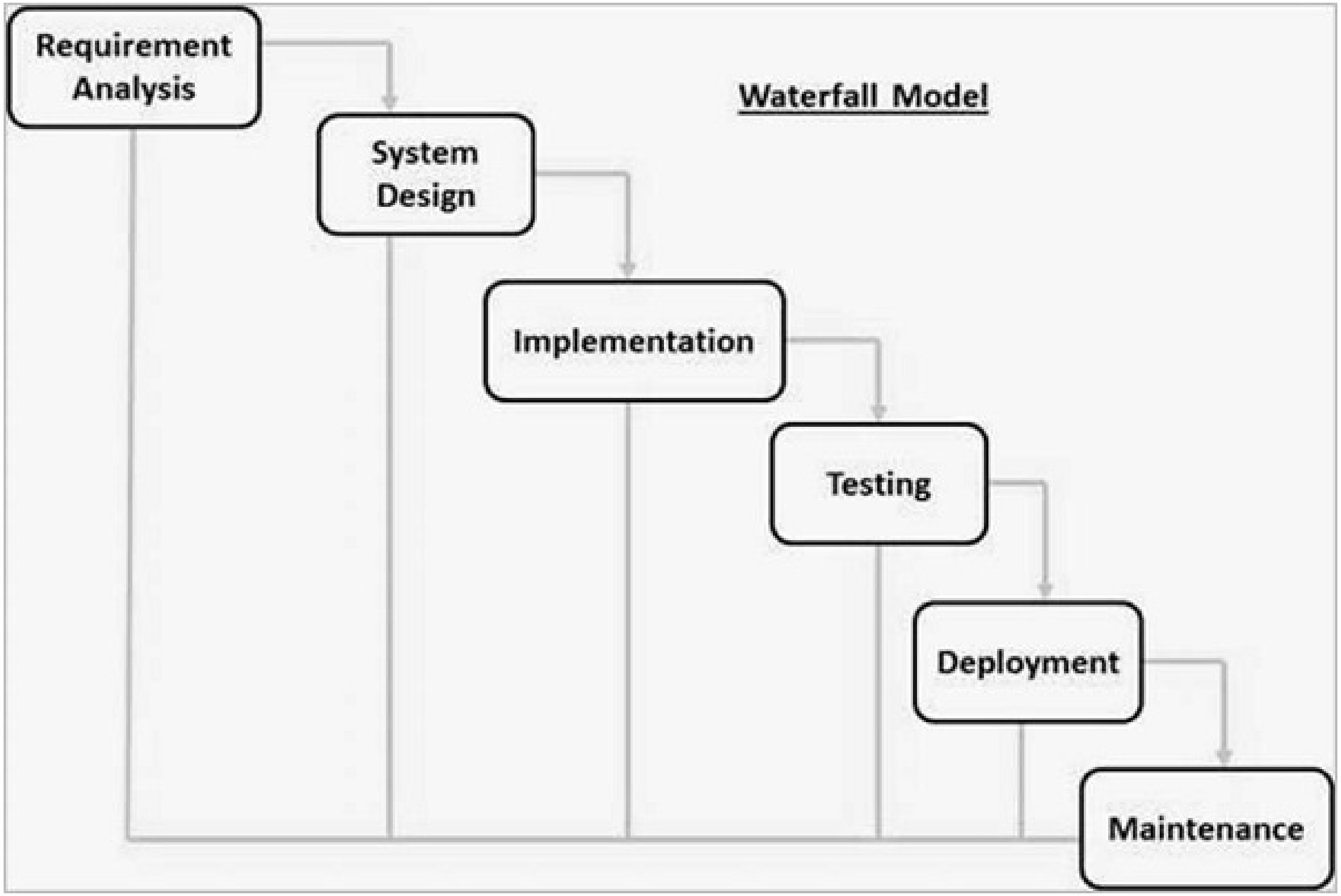
- The most common software development process
- Little or no planning and design.

1. think about the problem, write ideas on paper
2. start coding
3. run it. fix the code.
4. add another feature. As code grows I need to rewrite some parts to support each new feature.
    - modify the code for new feature
    - goto step 2.

My software process since high school

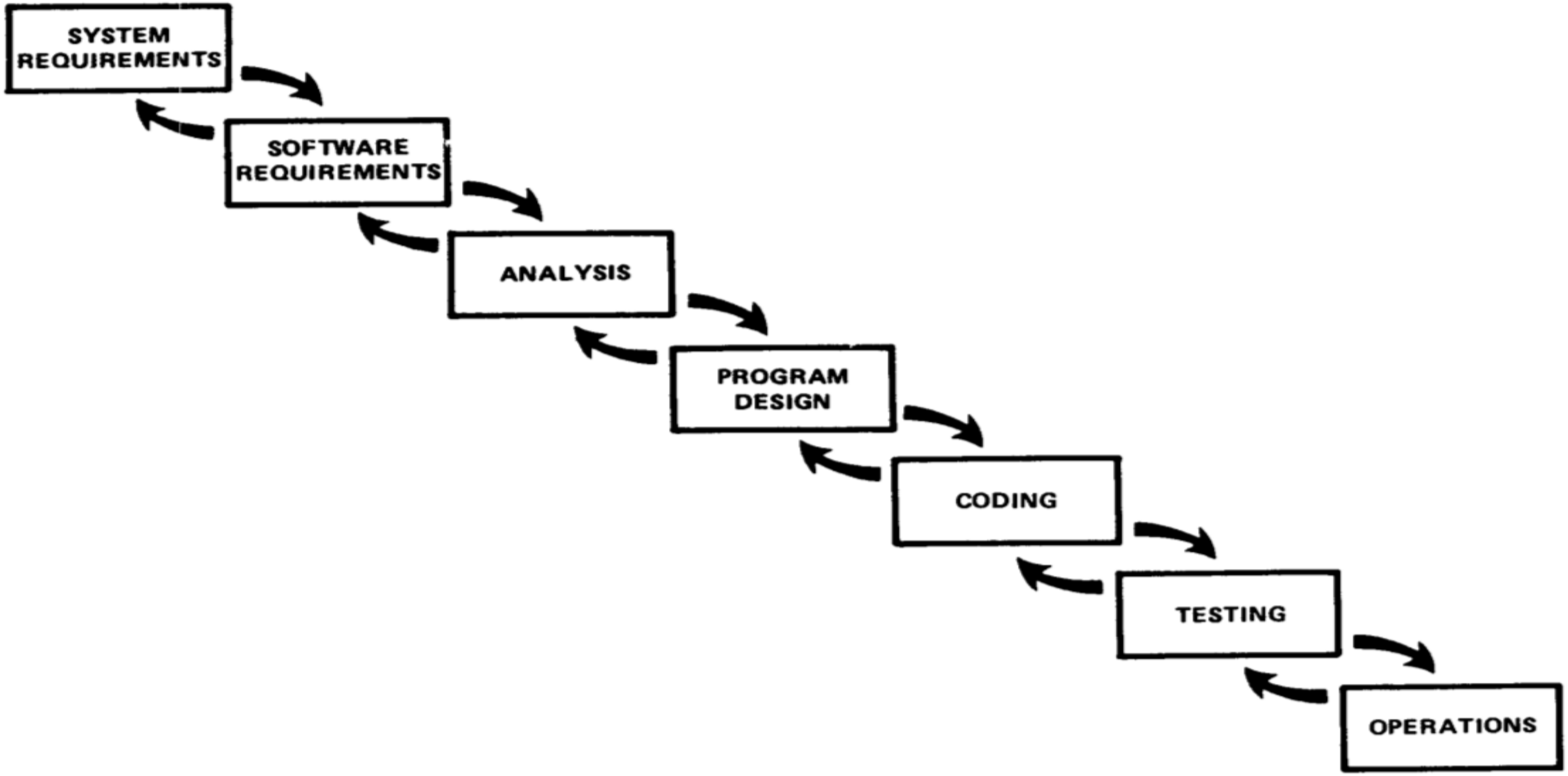
Do the activities in order

Similar to a civil engineering project.



(This is a typical diagram of the waterfall model.)

The Original Waterfall Model



Winston Royce, *Managing the Development of Large Software Systems* (1970)

Waterfall is still widely used.

What Could Go Wrong?

Problems with Waterfall

What would be the effect on project if ...

1. You miss some requirement(s).

2. You misunderstand a requirement, so the design is not what the customer wants.

3. The solution you design can't meet the requirements.

4. Coding takes a lot longer than expected.

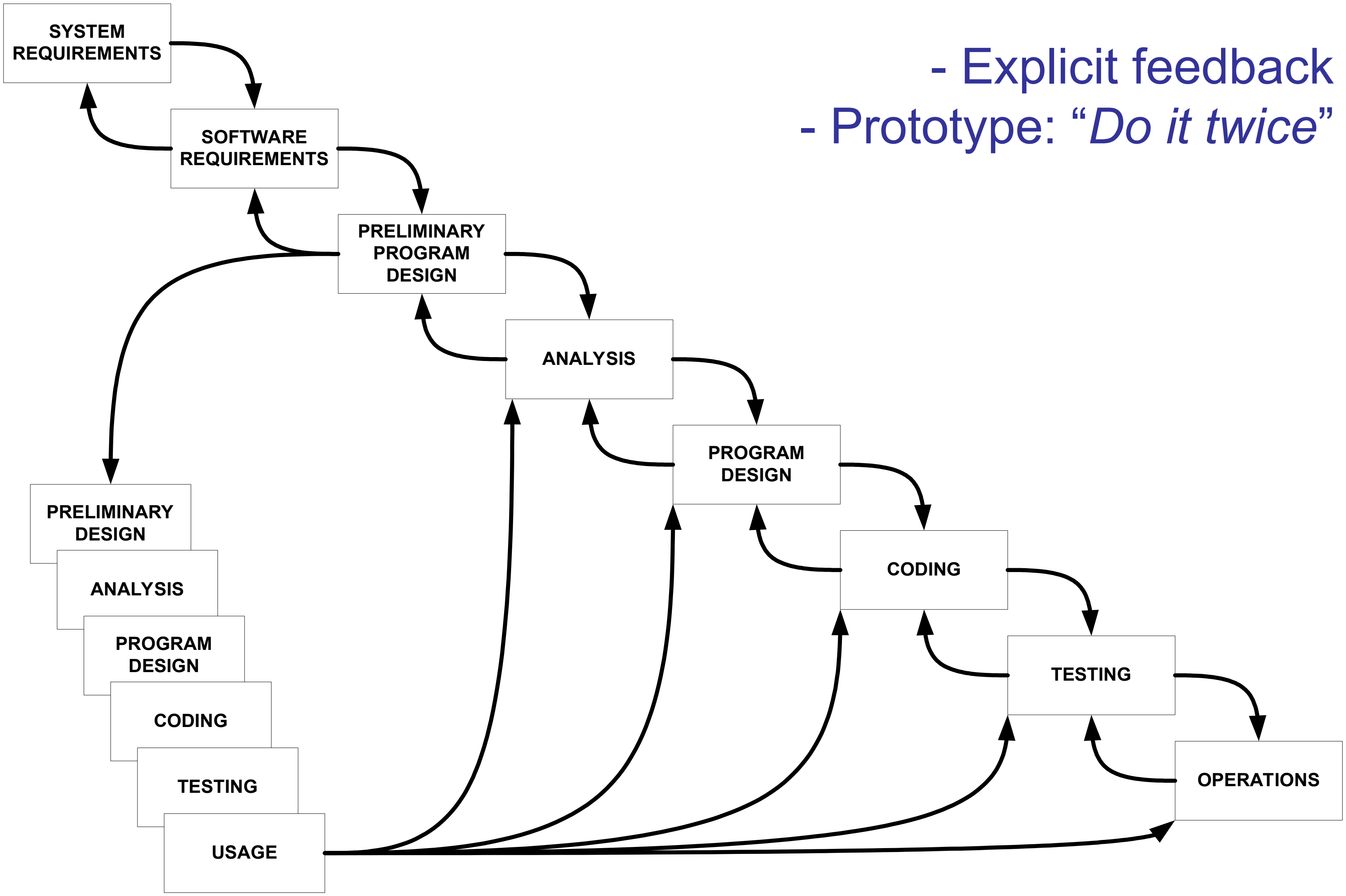
5. Testing discovers a lot of defects in the code.

How to Avoid These Problems?

- *Early* Feedback
- *Early* Testing
- *Continuously review* actual versus planned progress
- *Involve customer* at key points during project
- *Incremental delivery* of functionality -- get feedback.
- *Analyze* results and take corrective action

Royce Waterfall Model with Prototype

- Explicit feedback
- Prototype: “*Do it twice*”



Project Phase = Process Activity

In Waterfall, major activities are *phases* of the project...

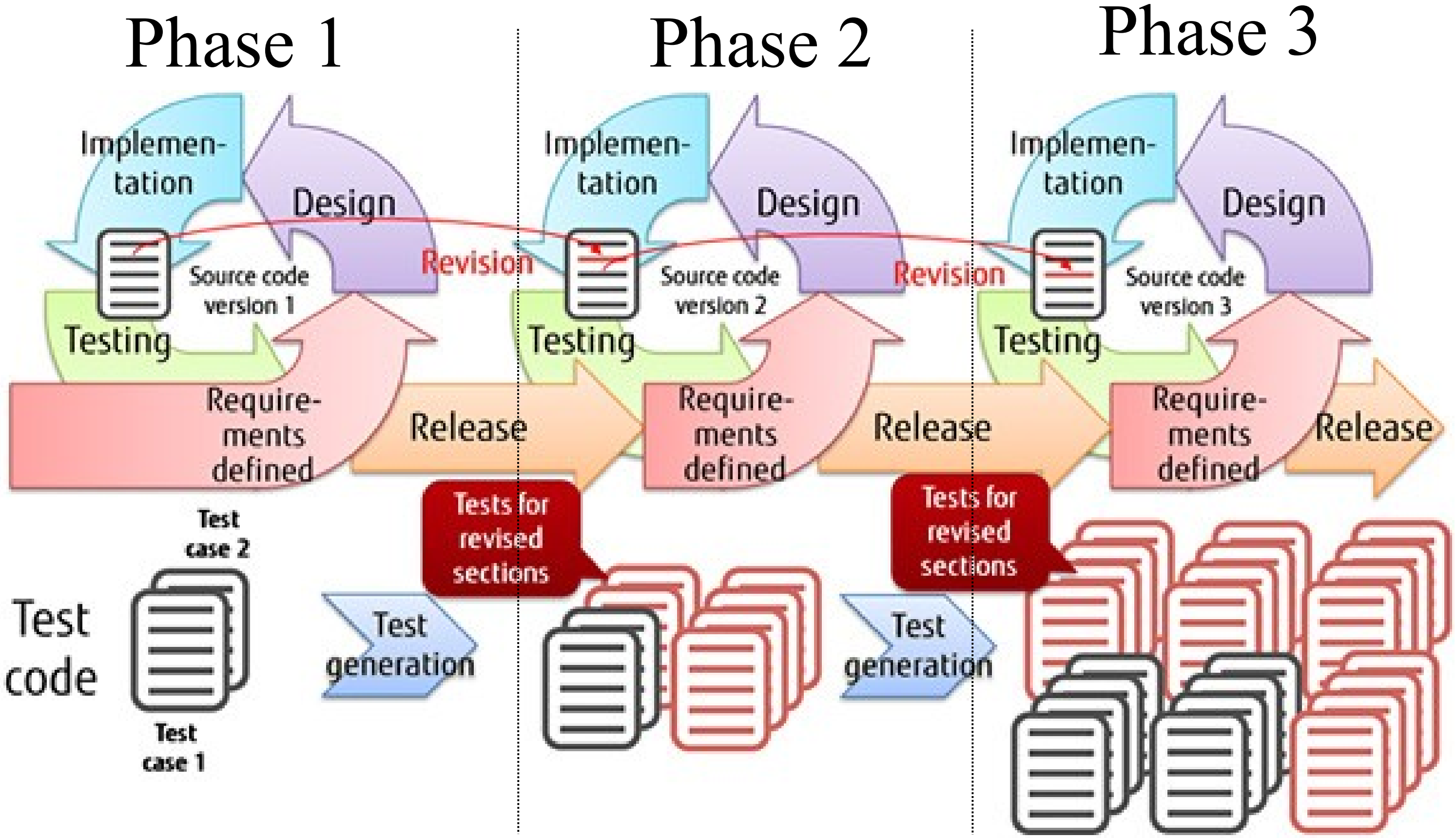
- Requirements phase
- Analysis phase
- Design phase
- Construction phase

...

Iterative and Incremental

Let's not build the whole product at once.
Build a useful part (subset), evaluate it, and repeat.

Activities ≠ Phases



Phase 1 Phase 2 Phase 3

Iterative and Incremental

Incremental - product divided into increments. Each increment adds **new features** and produces a **usable product**.

Iterative - iterate over the (almost) same activities for each product increment.

You may have many iterations to produce one increment.

Benefits

1. Rapid delivery of value to customer - he can try the features you have implemented.

What are other benefits of iterative & incremental?

Consider:

feedback

detecting problems in design or implementation

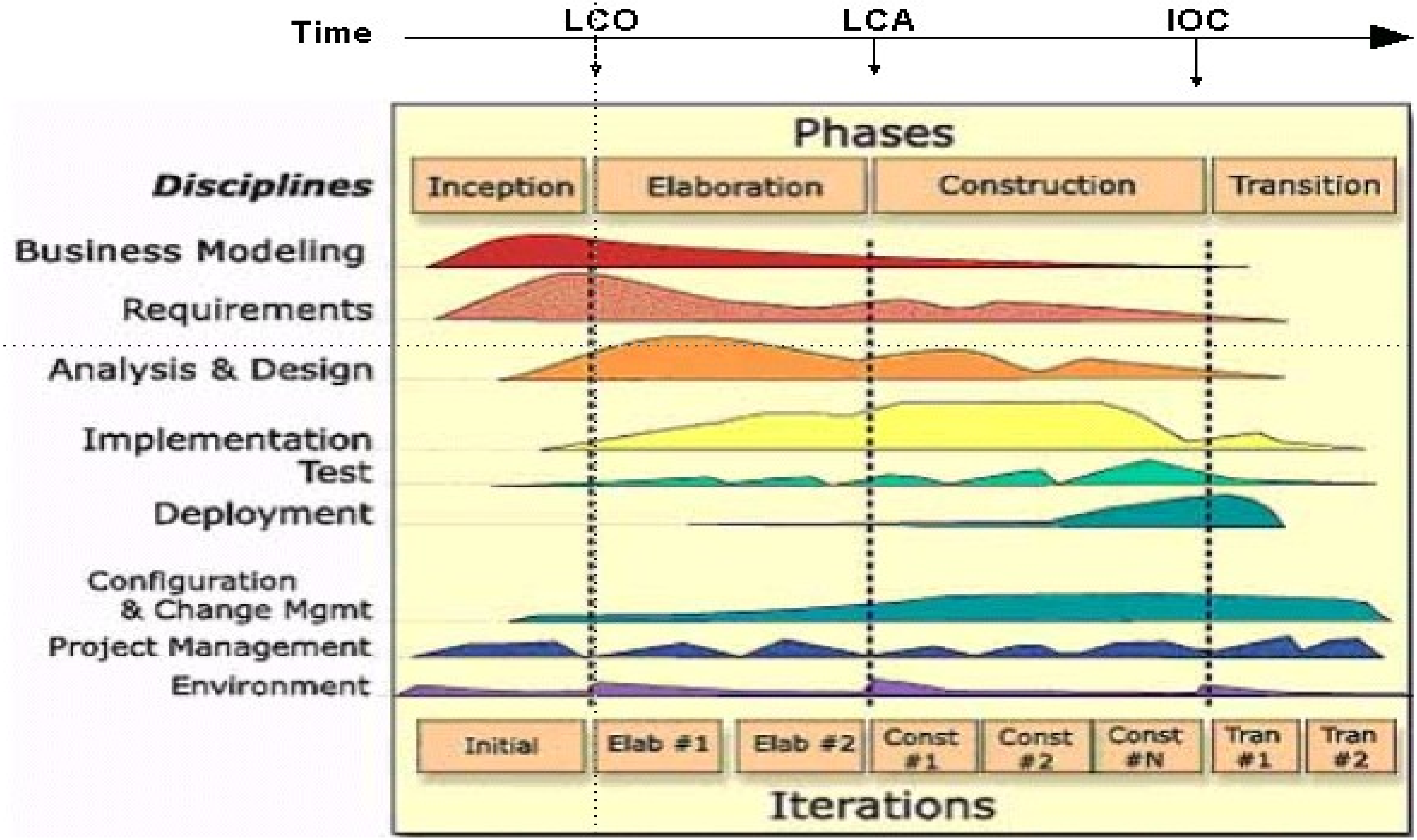
monitoring progress & deviation from schedule

effect of change

Unified Software Dev't Process (U.P.)

Workflows (disciplines) for different kinds of activities.

Phases: major divisions of project. Each has iterations.



UP is an Iterative Process **Model**

The diagram *conveys a lot* about UP...

- workflows (activities) are done in parallel
- "phases" for major evolutions of the project
- iterations within each phase, as needed

Characteristics of UP

- **Time-boxed** iterations
- **Plan based**, but adapts to change
- "Architecture centric"
- Identify **risks** & address them **early**
- Order requirements based on business value, architecture, & risk
  - handle risky requirements early
  - implement requirements that have big impact on the architecture
- UP is a "framework" for a process -- tailor it to your project

UP is covered in SKE *Software Spec and Design* course.

Agile

Agile is not a software process

Agile is a mindset, collection of values, and practices that reflect those values.

Agile & Scrum are covered later

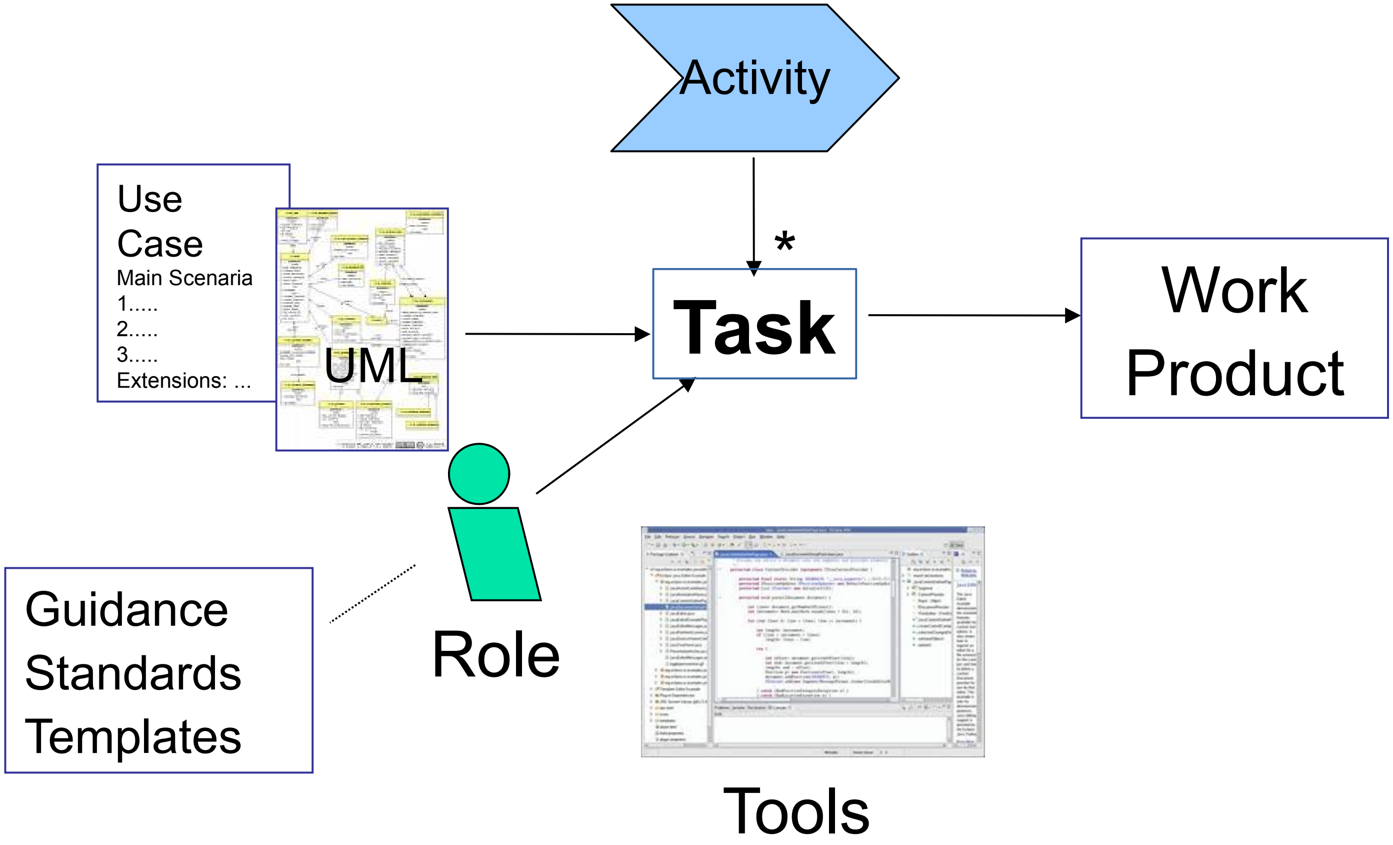
What About Individual Process?

This is a course about individual process.

What is that?

The Individual

People apply a process, use tools, technology, & guidance, to create the work products.



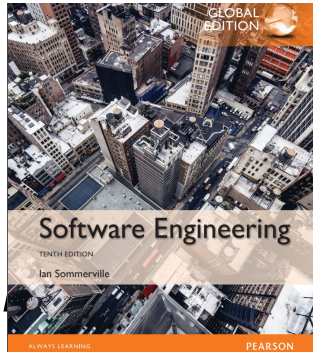
Problem of Teaching Software Process

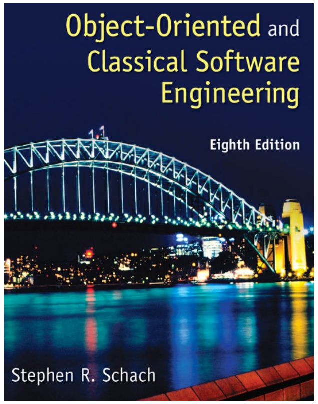
1. We learn on *small, one-semester* projects.
2. Projects often succeed based on heroic effort or super-programmers.
3. Programs aren't deployed or supported.
4. We are still learning, so process seems awkward.
5. We have many courses -- different environment from full-time developers
6. Outcome is a grade, not a paycheck or bonus

Reading

These are highly regarded books about *Software Engineering*. Each has a chapter or two on software process.

- Ian Summerville, *Software Engineering,* 10th Ed.
- Stephen Schach, *Object-oriented & Classical Engineering,* 8th Ed.
- Roger Pressman, *Software Engineering: A Practitioner's Approach*.

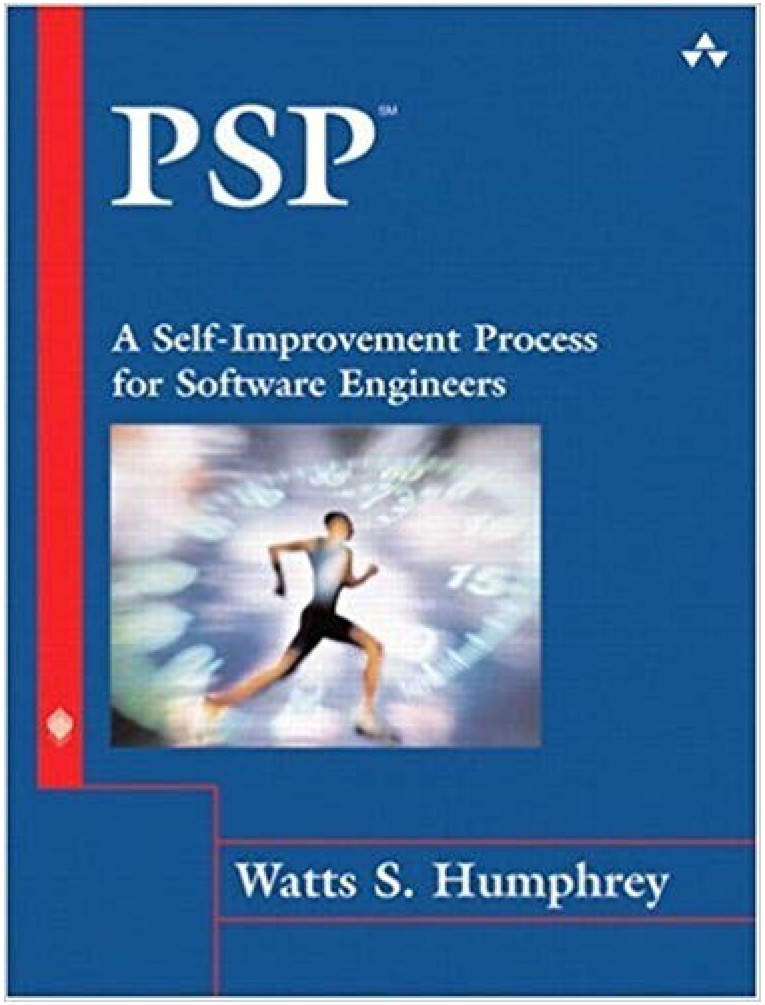




Historical Material

...for the curious

Original Syllabus: Personal Software Process



Step-by-step course to build a personal process for:

planning

defect tracking

estimation

measuring quality & efficiency

evaluation

process improvement

Goals of PSP

Objective: provide a disciplined process for SEs to manage their own work

- improve estimation and planning skills
- reduce defects in their products
- manage their own schedule & work quality
- improve their own software process

PSP progress through levels

PSP0: [baseline] measure time you spend on planning, design, coding, test, and *post mortem* (retrospective)

PSP0.1: measure output LOC. Add a coding standard and process improvement proposal (PIP).

PSP 1.0: Estimate program size using level 0 data. Make a test plan.

PSP 1.1: Add planning. Estimate time from program size.

PSP 2.0: Add design & code review. Emphasis on defect removal and prevention.

PSP 2.1: Add design specification.

PSP 3: Apply an iterative process to PSP2.1.

PSP Tools and Support

PSP emphasizes use of scripts, forms, and checklists to guide the user. These are included in course.

A useful tool is Process Dashboard (Sourceforge).

- performs time tracking. Automates some reporting.
- includes the PSP scripts and forms, and generates reports
- *can be used for other processes*!