Software Architecture

Bertrand Meyer, Carlo Furia, Martin Nordio

ETH Zurich, February-May 2011

Lecture 6.3: PSP

CMMI background

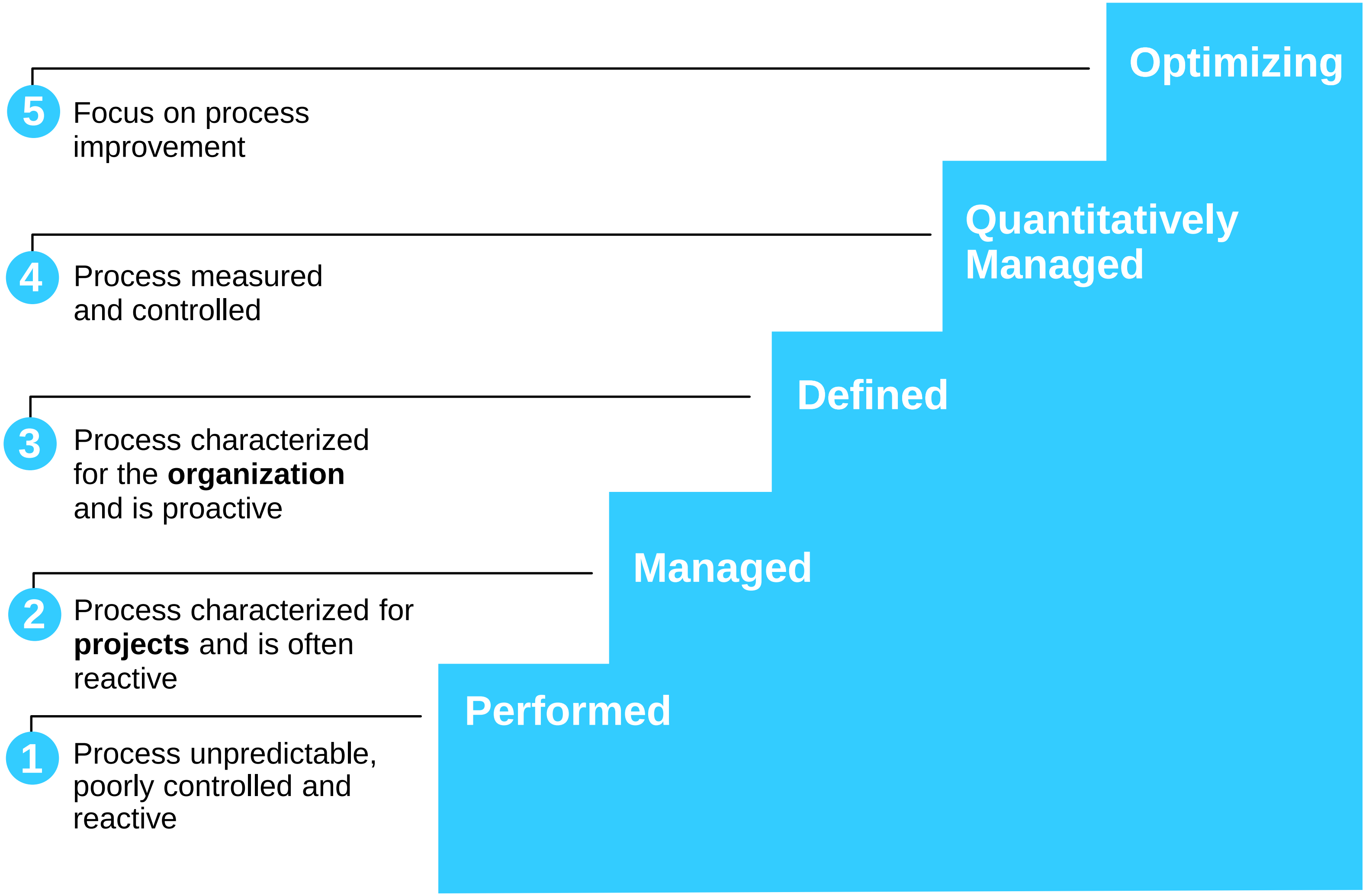
Initially: Capability Maturity Model (CMM), developed by Software Engineering Institute (at Carnegie-Mellon University, Pittsburgh) for the US Department of Defense, 1987-1997; meant for software

Widely adopted by Indian outsourcing companies

Generalized into CMMI (version 1.1 in 2002)

SEI itself offers assessments: SCAMPI (Standard CMMI Appraisal Method for Process Improvement)

The maturity levels



Optimizing

5 Focus on process improvement

Quantitatively Managed

4 Process measured and controlled

Defined

3 Process characterized for the **organization** and is proactive

Managed

2 Process characterized for **projects** and is often reactive

Performed

1 Process unpredictable, poorly controlled and reactive

TSP, PSP

PSP: Personal Software Process

TSP: Team Software Process

Transposition of CMMI-like ideas to work of individual teams and developers

Management support

The initial TSP objective is to convince management to let the team be self-directed, meaning that it:

- Sets its own goals
- Establishes its own roles
- Decides on its development strategy
- Defines its processes
- Develops its plans
- Measures, manages, and controls its work

Management support

Management will support you as long as you:

- Strive to meet their needs
- Provide regular reports on your work
- Convince them that your plans are sound
- Do quality work
- Respond to changing needs
- Come to them for help when you have problems

Management support

Management will agree to your managing your own work as long as they believe that you are doing a superior job.

To convince them of this, you must:

- Maintain and publish precise, accurate plans
- Measure and track your work
- Regularly show that you are doing superior work

The PSP helps you do this

PSP essential practices

- Measure, track, and analyze your work
- Learn from your performance variations
- Incorporate lessons learned into your personal practices

What does a PSP provide?

A stable, mature PSP allows you to

- Estimate and plan your work
- Meet your commitments
- Resist unreasonable commitment pressures

You will also

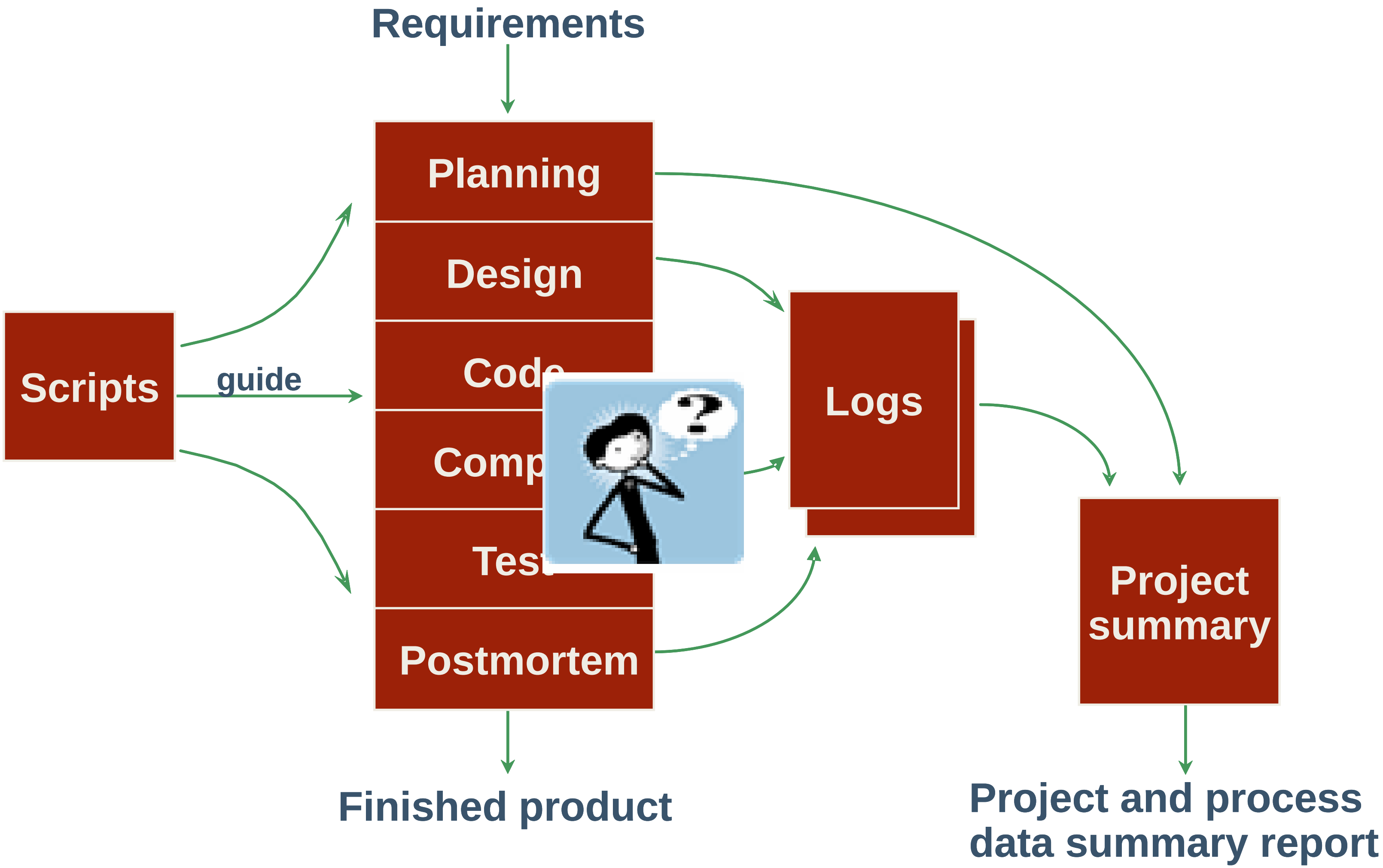
- Understand your current performance
- Improve your expertise as a professional

PSP fundamentals

As a personal process, PSP includes:

- Defined steps
- Forms
- Standards
- A measurement and analysis framework for characterizing and managing your personal work
- A defined procedure to help improve your personal performance

The PSP process flow



Requirements

Planning

Design

Code

Comp

Test

Postmortem

Scripts

guide

Logs

Project summary

Finished product

Project and process data summary report

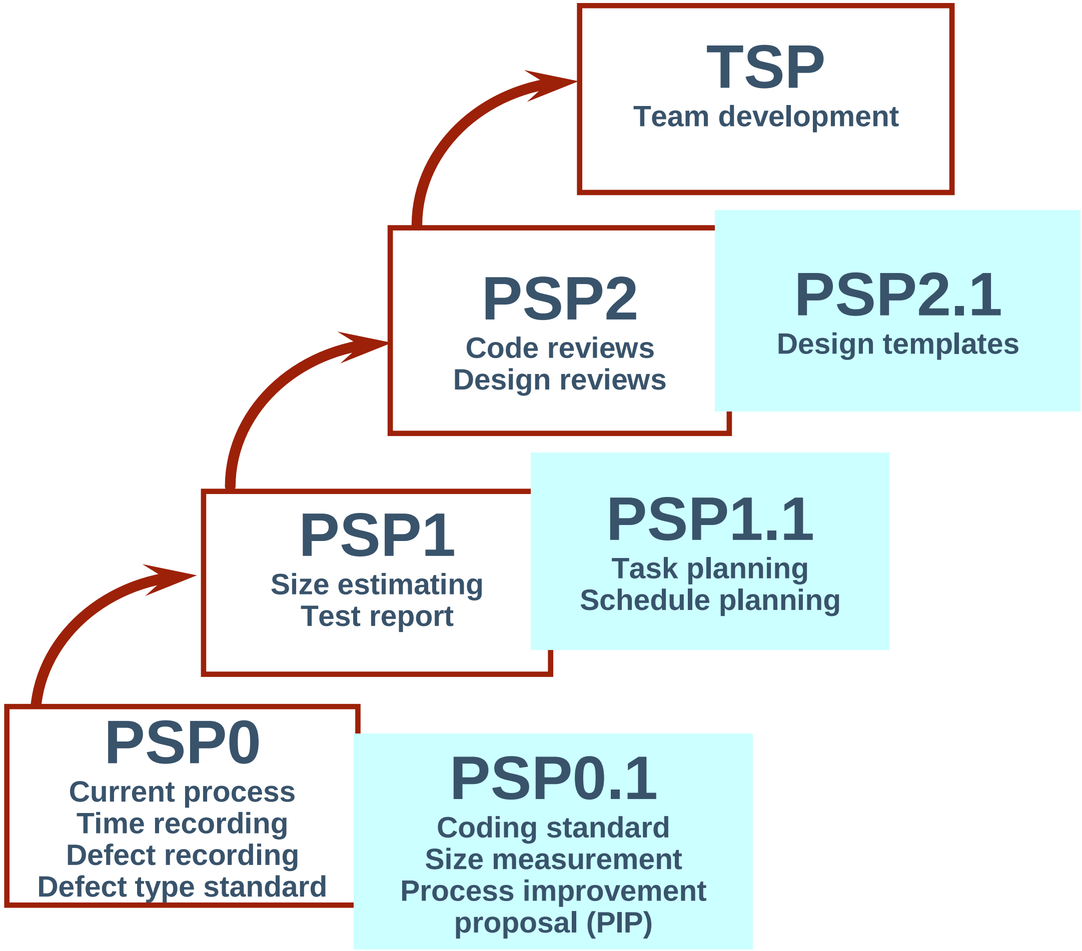
A progressive approach

PSP is introduced in six upward-compatible steps

At each step:

- Write one or more modules
- Gather and analyze data on your work
- Use results to improve your personal performance

The steps



TSP
Team development

PSP2
Code reviews
Design reviews

PSP2.1
Design templates

PSP1
Size estimating
Test report

PSP1.1
Task planning
Schedule planning

PSP0
Current process
Time recording
Defect recording
Defect type standard

PSP0.1
Coding standard
Size measurement
Process improvement
proposal (PIP)

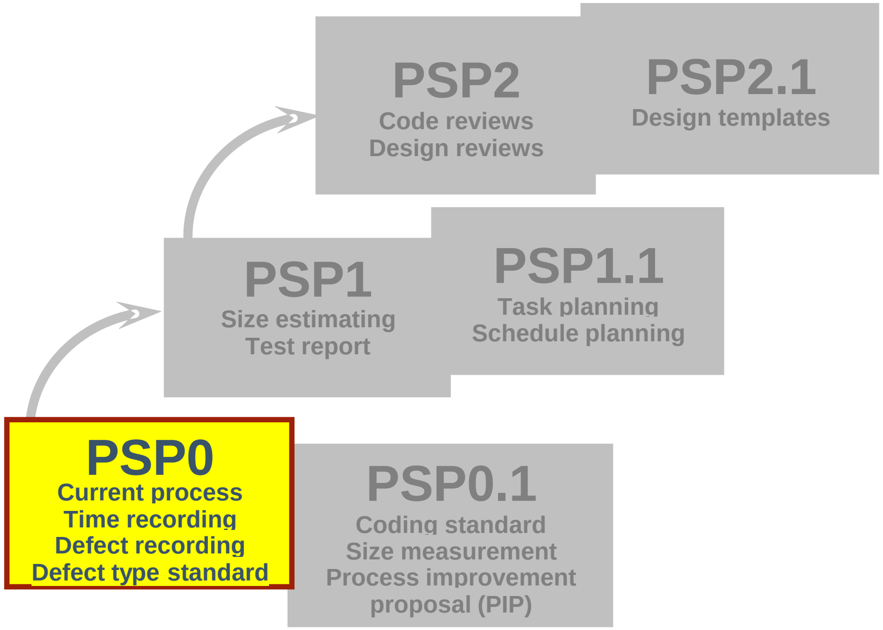
Goals at each level

PSP0: Establish a measured performance baseline

PSP1: Make size, resource, and schedule plans

PSP2: Practice defect and yield management

PSP0



Objective:

- Demonstrate use of defined process for small programs
- Incorporate basic measurements in process
- Minimize changes to your personal practices

PSP0 setup

PSP0 is a simple, defined, personal process:

- Make a plan
- Use your current design and development methods to produce a small program
- Gather time and defect data on your work
- Prepare a summary report

The six phases of PSP0

Produce plan for developing program from requirements

1 **Plan**

Produce design specification for the program.

2 **Design**

Turn design into executable code

(In Eiffel, 2 & 3 are seamless)

3 **Code**

4 **Compile**

Translate into executable code

5 **Test**

Verify that code satisfies requirements

6 **Postmortem**

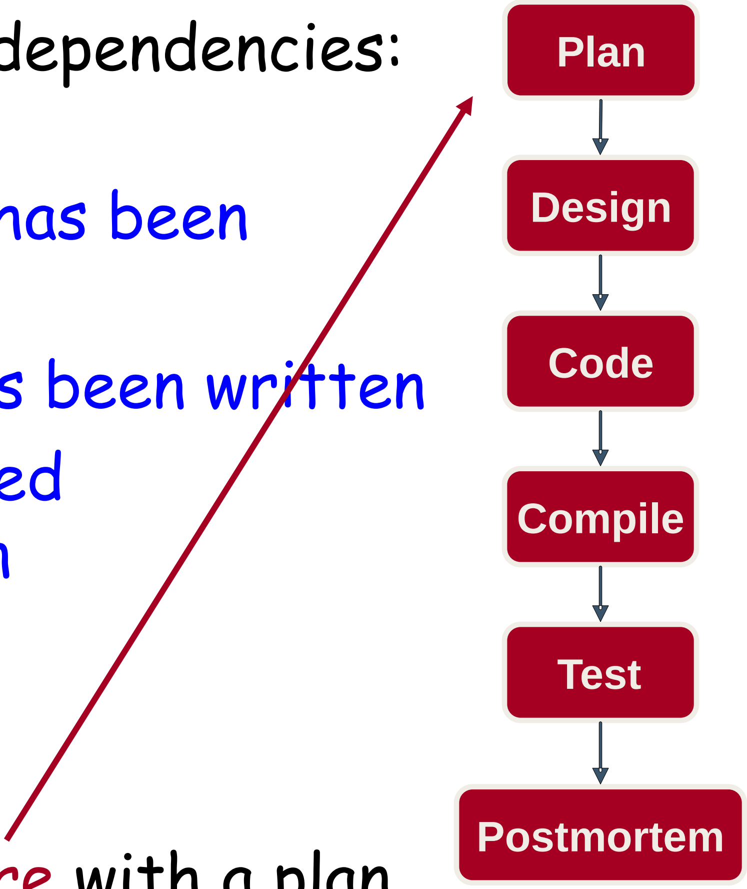
Summarize & analyze project data

Phase order

PSP looks like waterfall but is not

Phase order is determined by dependencies:

- Cannot test code before it has been compiled
- Cannot compile before it has been written
- Cannot use design if produced after code has been written
- No reason to make a plan after you're done



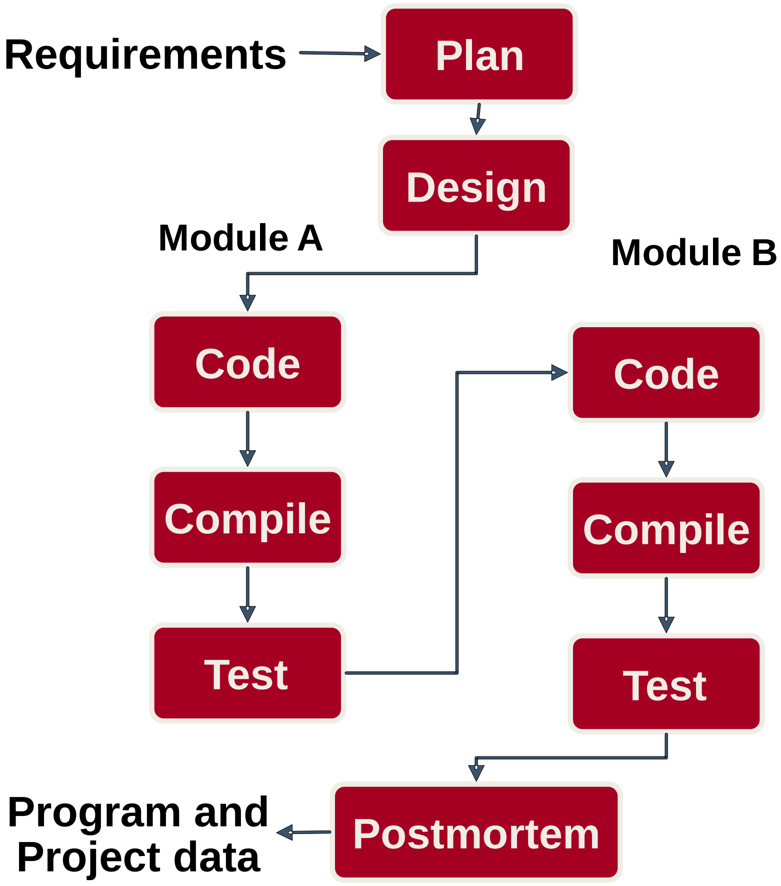
Conclusion: start here with a plan

Cyclic process flow

Programs that are large programs or not well understood may require an iterative approach

In this example, each module is separately coded, compiled, and tested

The example uses PSP0 phases and 2 code-compile-test cycles



Cyclic process flow

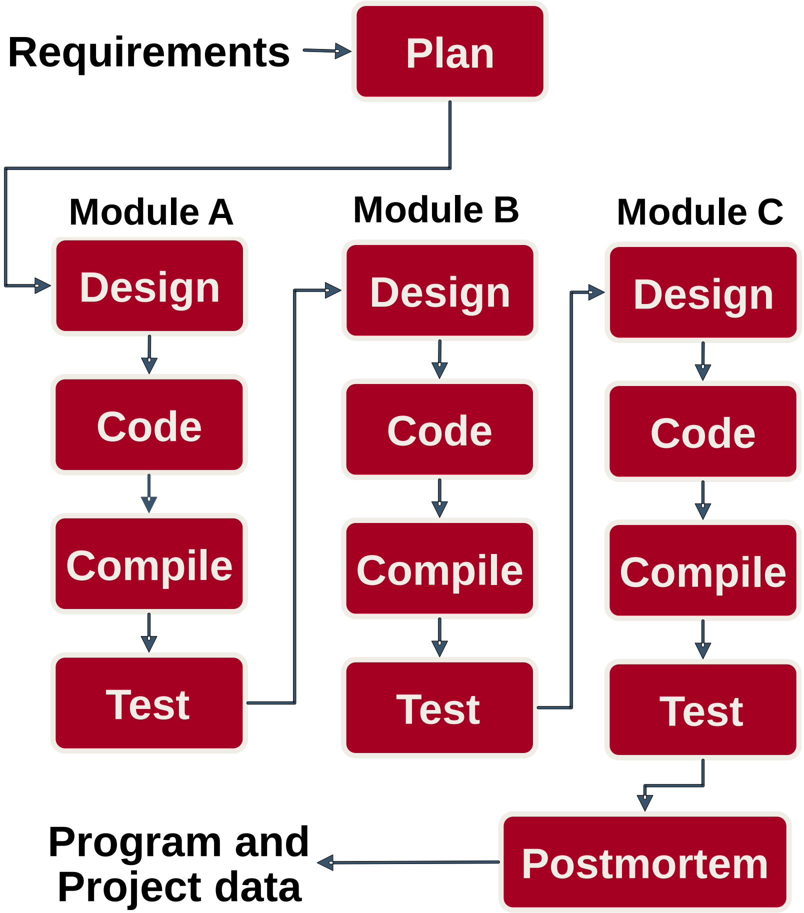
There can be more than 2 cycles

Part size is key factor for determining cycles:

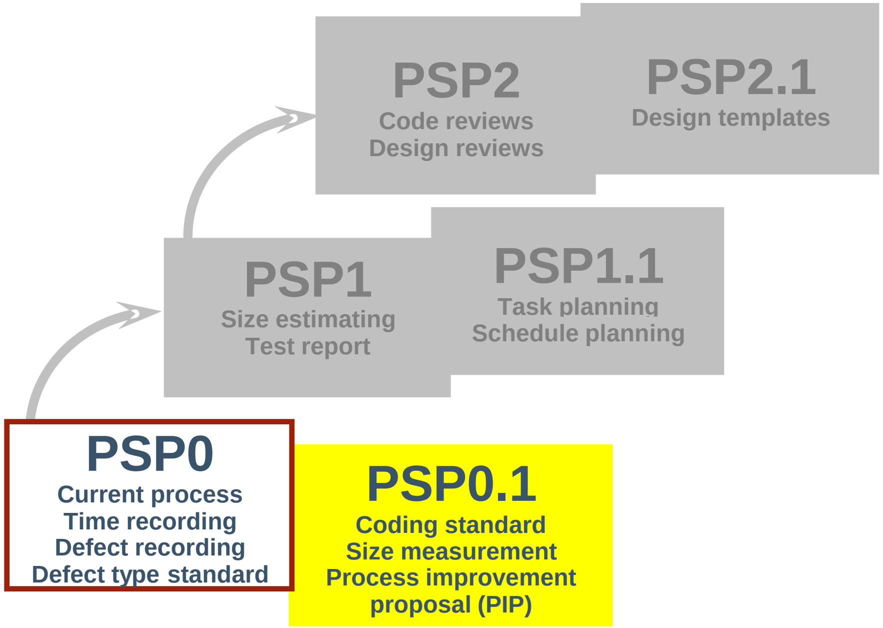
- Line of code: too small
- Program: usually too large

Typical: one or more classes or features

Determine what works for you



PSP0.1



PSP2
Code reviews
Design reviews

PSP2.1
Design templates

PSP1
Size estimating
Test report

PSP1.1
Task planning
Schedule planning

PSP0
Current process
Time recording
Defect recording
Defect type standard

PSP0.1
Coding standard
Size measurement
Process improvement proposal (PIP)

Objective: help you to

- Measure size of programs that you produce
- Perform size accounting for these programs
- Make accurate and precise size measurements

Process measurement

To be useful, measurements should be

- Gathered for a specific purpose
- Explicitly defined
- Properly managed
- Properly used

We measure to

- Understand and manage change
- Predict or plan
- Compare one product, process, or organization with another
- Determine adherence to standards
- Provide a basis for control

Measurement objectives

Measurements only produce numbers

To be useful, they must

- Relate to business objectives
- Be properly interpreted
- Lead to appropriate action

If the business purposes for the measurements are not understood

- The wrong data may be gathered
- Data may not be properly used

PSP measurements

Basic PSP data:

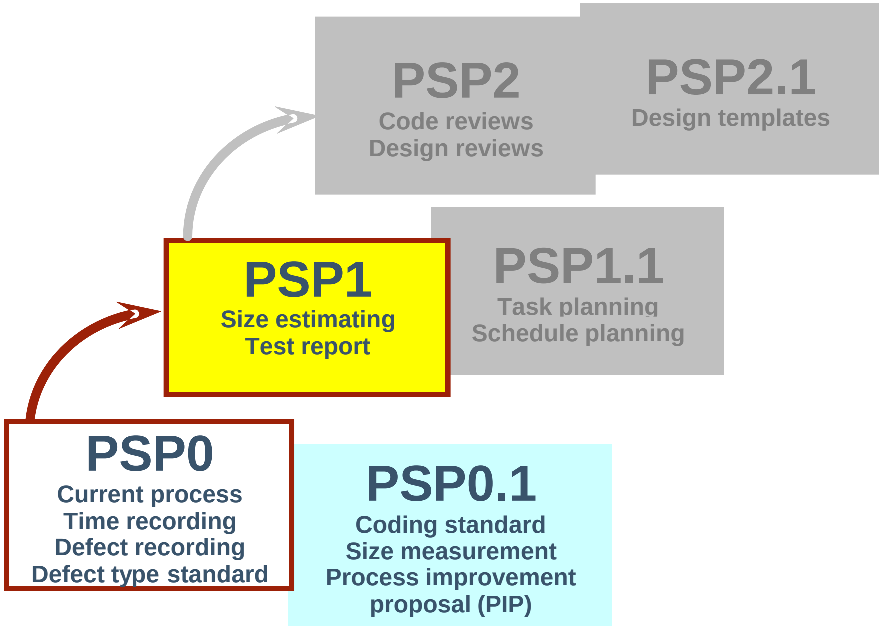
- Program size
- Time spent by phase
- Defects found and injected by phase

On every item, gather both actual and estimated data

Measures derived from these data:

- Support planning
- Characterize process quality

PSP1



Objective:

Establish orderly & repeatable procedure for size estimation

New process elements:

- PROBE size estimating method & template
- Test report template

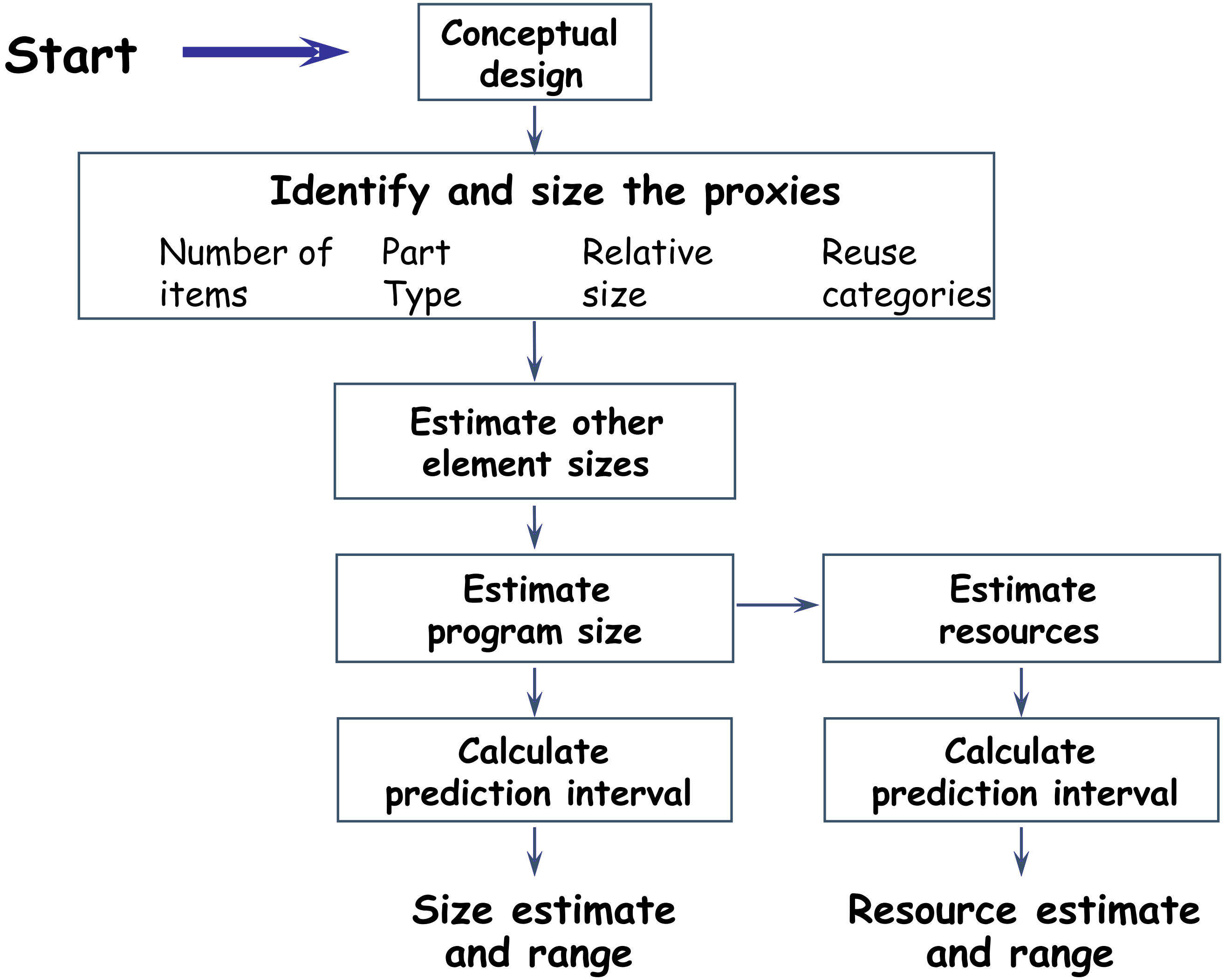
Estimating with PROBE

Stands for PROxy Based Estimating

Uses proxies to estimate program size and development time

A good proxy helps make accurate estimates

The PROBE estimating method



Start

Conceptual design

Identify and size the proxies

Number of items

Part Type

Relative size

Reuse categories

Estimate other element sizes

Estimate program size

Estimate resources

Calculate prediction interval

Calculate prediction interval

Size estimate and range

Resource estimate and range

Conceptual design

Conceptual design relates the requirements to the parts needed to produce the program

Parts categories:

- Reused: Can be used as-is
- Base: Exists, requires modifications
- Added: needs to be developed

Sizing parts

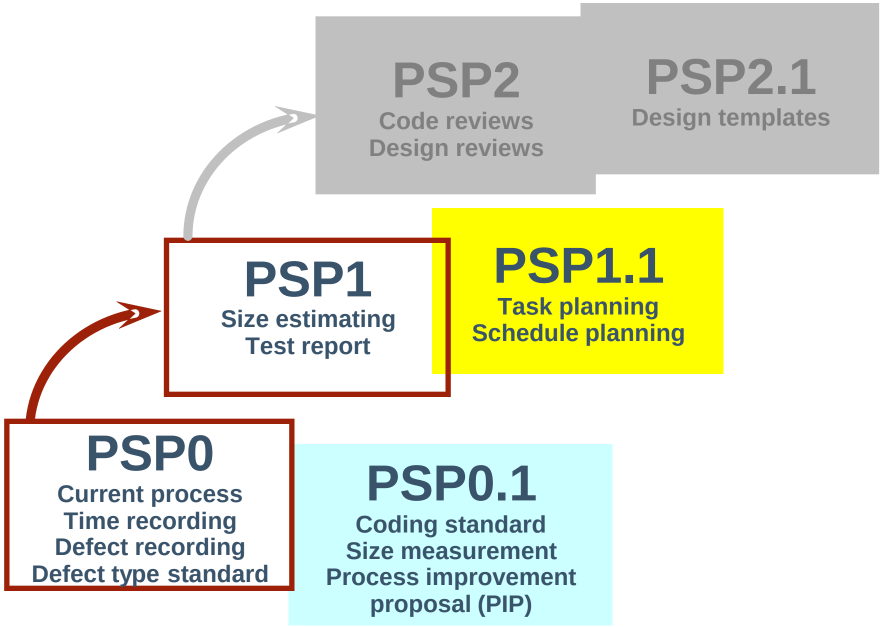
Reused part: Use actual size

Added part: define proxy

- Identify part type, e.g. parsing, GUI, network…
- Estimate number of items, e.g. routines
- Estimate relative size, i.e. very small, small, medium, large, or very large
- Find size of an item of this part type and relative size in the relative size table
- Estimated size = item size * number of items

Base part: start from actual size; estimate additions, deletions, modifications

PSP1.1


PSP2.1
Design templates

PSP2
Code reviews
Design reviews

PSP1.1
Task planning
Schedule planning

PSP1
Size estimating
Test report

PSP0.1
Coding standard
Size measurement
Process improvement proposal (PIP)

PSP0
Current process
Time recording
Defect recording
Defect type standard

Objective: introduce & practice methods for

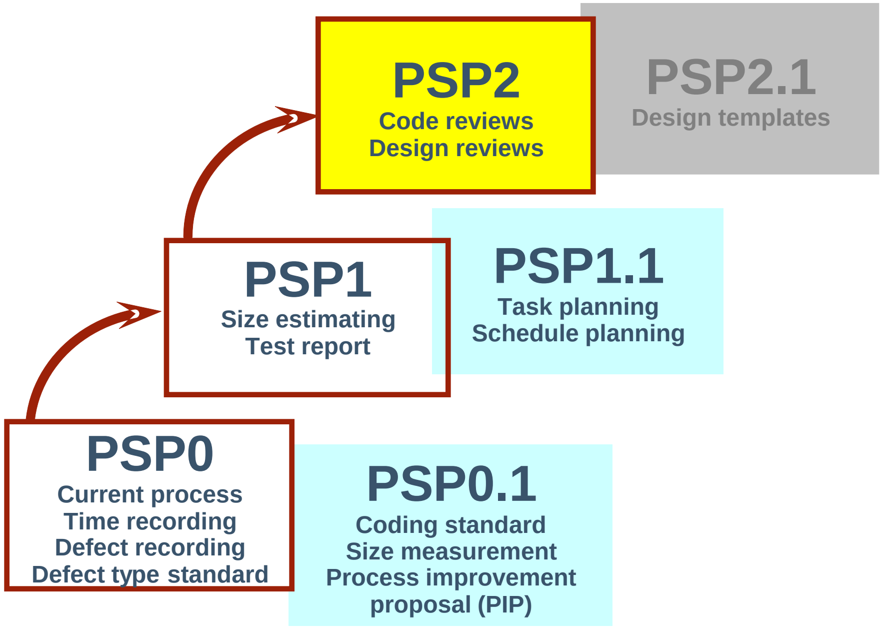
- Making resource & schedule plans
- Tracking your performance against them
- Judging likely project completion dates

Two new process elements:

- Task planning template
- Schedule planning template

Typically used for projects that take several days or weeks

PSP2



Objective: introduce

- Design & code reviews
- Methods for evaluating & improving quality of reviews

Two key capabilities added at this level:

- Design and code reviews
- Quality planning

Two new process elements, separate:

- Design review checklist
- Code review checklist

Quality planning

PSP2 introduces quality planning. This involves estimating:

- Total number of defects that will be injected
- Number of defects injected & removed in each process phase
- Amount of time for design and code reviews

& adjusting these parameters to ensure high-quality result

Arguments for reviews over tests

In testing, you must

- Detect unusual behavior
- Figure out what the test program was doing
- Find where the problem is in the program
- Figure out which defect could cause such behavior

This can take a lot of time

With reviews you

- Follow your own logic
- Know where you are when you find a defect
- Know what the program should do, but did not
- Know why this is a defect
- Are in a better position to devise a correct fix

PSP review process principles

Defined review process: guidelines, checklists, standards.

Goal is to find every defect before first compile/test

To meet it, you must:

- Review before compiling or testing
- Use coding standards
- Use design completeness criteria
- Measure and improve your review process
- Use a customized personal checklist

Code reviews

General principles (not specifically from PSP):

- Uncoupled from evaluation process
- Meeting must have chair, secretary
- Chair is not supervisor
- Purpose is to identify faults
- Purpose is not to correct them
- Purpose is not to evaluate developer; keep focus technical
- Strict time limit (e.g. 2 hours)
- Announced sufficiently long in advance
- Participant number: 5 to 10
- Code available in advance, as well as any other documents
- Meeting must be conducted professionally and speedily; chair keeps it focused

Code review checklist

Reviews are most effective with personal checklist customized to your own defect experience:

- Use your own data to select the checklist items
- Gather and analyze data on the reviews
- Adjust the checklist with experience

Do the reviews on a printed listing, not on screen

The checklist defines steps and suggests their order:

- Review for one checklist item at a time
- Check off each item as you complete it

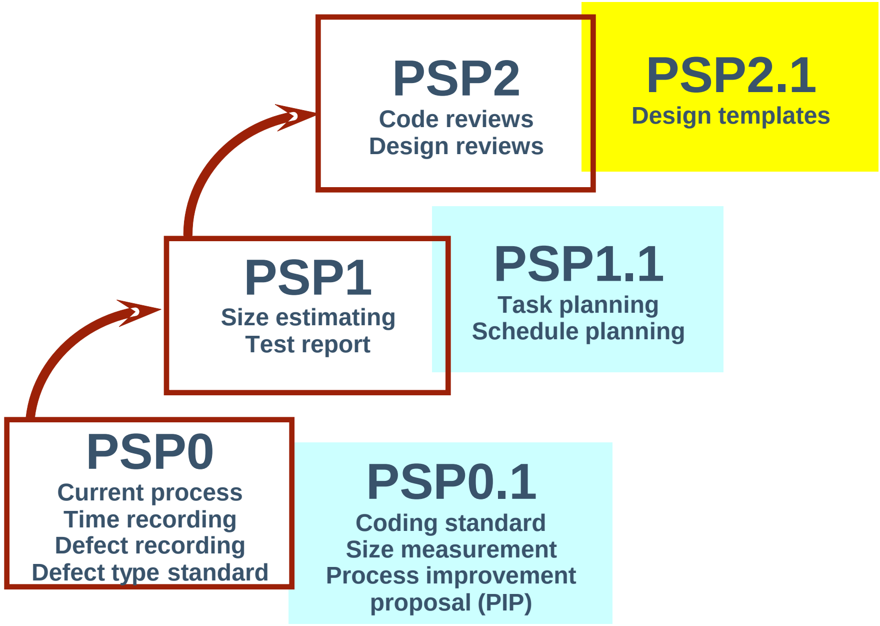
Design review principles

In addition to reviewing code, you should also review your designs

Requires that you

- Produce designs that can be reviewed
- Follow an explicit review strategy
- Review design in stages
- Verify that logic correctly implements requirements

PSP2.1



Objective: introduce

- Additional measures for managing process quality
- Design templates that provide an orderly framework and format for recording designs

New process elements:

- Design review script
- Design review checklist
- Operational specification template
- Functional specification template
- State specification template
- Logic specification template

PSP: an assessment

Ignore technology assumptions (strict design-code-compile-test cycle) which is not in line with today's best practices.

Retain emphasis on professional engineer's approach:

- Plan
- Record what you do both qualitatively and quantitatively:
  - Program size
  - Time spent on parts and activities
  - Defects
- Think about your personal process
- Improve your personal process

Tool support, integrated in IDE, is essential