

# Reflective Software Engineering

## Module 01: Basics

(1)



## Module Objectives

### Understand:

- Basic principles of reflective software engineering
- Basic use of Leap toolkit
- Installation procedure for Leap toolkit

### Related materials:

- Reflective Software Engineering with the Leap Toolkit, Chapter 1.

(2)



## **"Internet Time"**

Software development on "internet time" is characterized by:

- time-to-market important
- short release cycles
- eat your own dogfood
- customer-based testing
- non-critical quality constraints
- under-capitalized, autonomous teams
- crunch mode

How can we become more effective and efficient in these environments?

(3)



## **Reflective Software Engineering**

People learn best from their own experience.

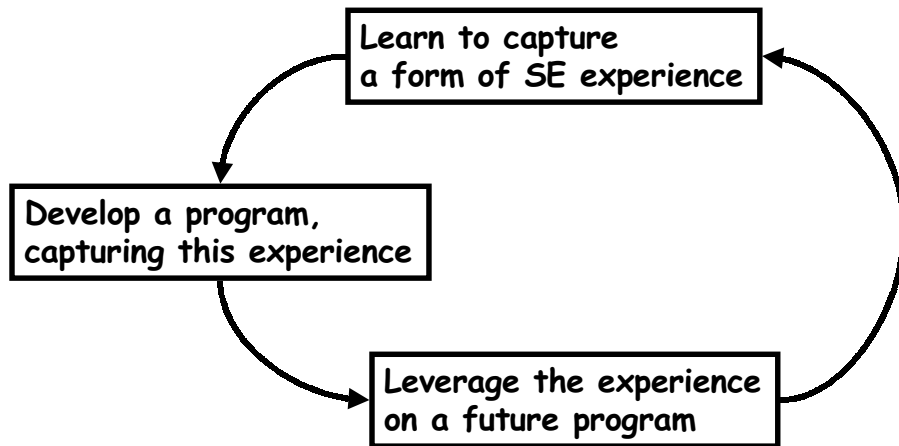
Reflective software engineering is intended to help you:

- Maximize the value of your experiences.
- Learn despite the generally "learning-hostile" nature of an internet time organization.
- Acquire skills and techniques to make you more marketable and valuable as a developer.

(4)



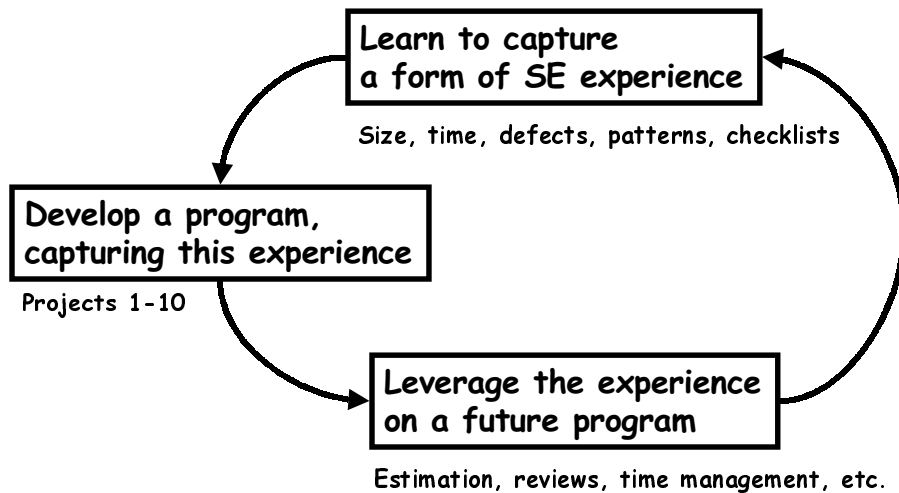
# Experience-based Learning



(5)



# Experience-based Learning



(6)



## The curriculum modules

01: Prerequisites	13: Testing
02: Basics	14: Project 6
03: Time measurement	15: UML design
04: Project 1	16: Project 7
05: Size measurement	17: Patterns
06: Project 2	18: Project 8
07: Personal defect man.	19: Special Topic
08: Project 3	20: Project 9
09: Group defect man.	21: Custom project def.
10: Project 4	22: Project 10
11: Project data man.	23: Custom project 01
12: Project 5	24: Project 11

(7)



## The importance of overhead

For experience-based learning via reflective software engineering to occur in an Internet-time environment,

- The amount of overhead must be kept to an absolute minimum
- The return-on-investment in recording/analysis/reflection must come back quickly and exceed the investment in value.

This places high demands on the computational infrastructure (i.e. tool support)!

(8)



## Reflective Software Engineering and Traditional SE

RSE complements traditional SE.

Traditional SE focuses on “best practices” for requirements, specification, planning, etc.

Reflective software engineering helps you gain insight into when, where, and whether to adopt a best practice and what the impact of adoption actually is.

(9)



## Reflective Software Engineering and the PSP

Reflective software engineering might be viewed as a “PSP for the differently disciplined”...

Similarities:

- Collection and analysis of individual SE data.

Differences:

- Tool support
- Types of data collected and analyzed
- “Process” constraints
- Focus on internet time organizations

(10)



## **You can stop reading now if...**

**You are not committed to becoming more expert in your field of work.**

**You do not create work products as a primary job activity.**

**Your work products bear no interesting "similarity" to those you've created in the past.**

(11)



## **If you're still here...**

**Go on to the "Guided Tour of Leap" module.**

**Do the evaluation quiz on WebCT.**

(12)

