TSP Symposium 2010

How to Teach Programming:

Introducing PSP into a University Curriculum

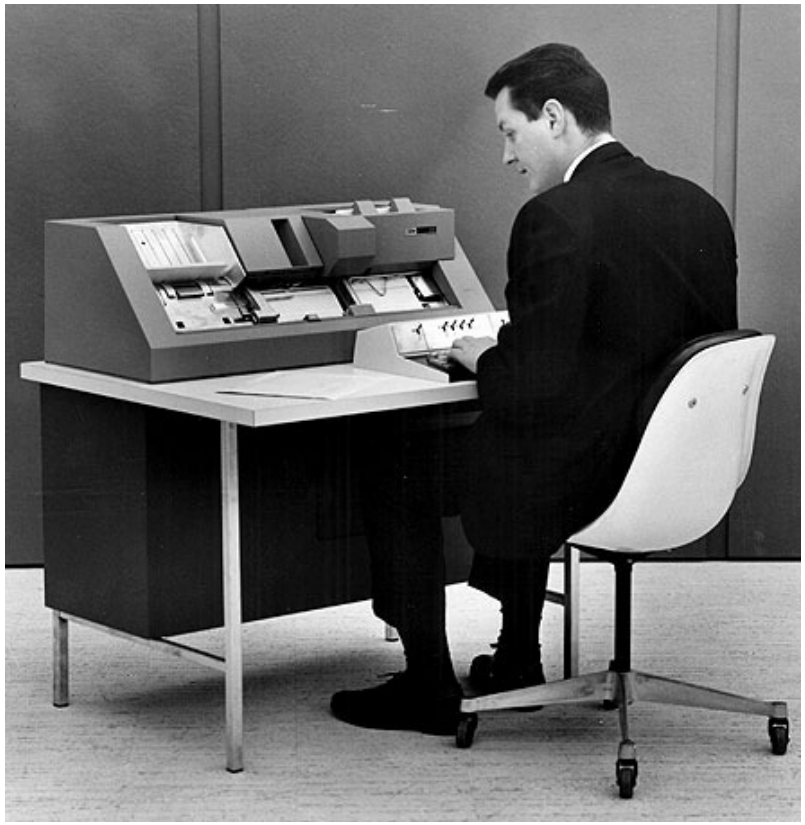
Prof Barry Dwolatzky,
Joburg Centre for Software Engineering (JCSE)
at Wits University, South Africa

Introducing:
Dwolatzky the Programmer

Wrote his first program in 1970 – Fortran IV on IBM 360.
A few hundred lines-of-code.

Introducing:
Dwolatzky the Programmer

Wrote his first program in 1970 – Fortran IV on IBM 360.
A few hundred lines-of-code.

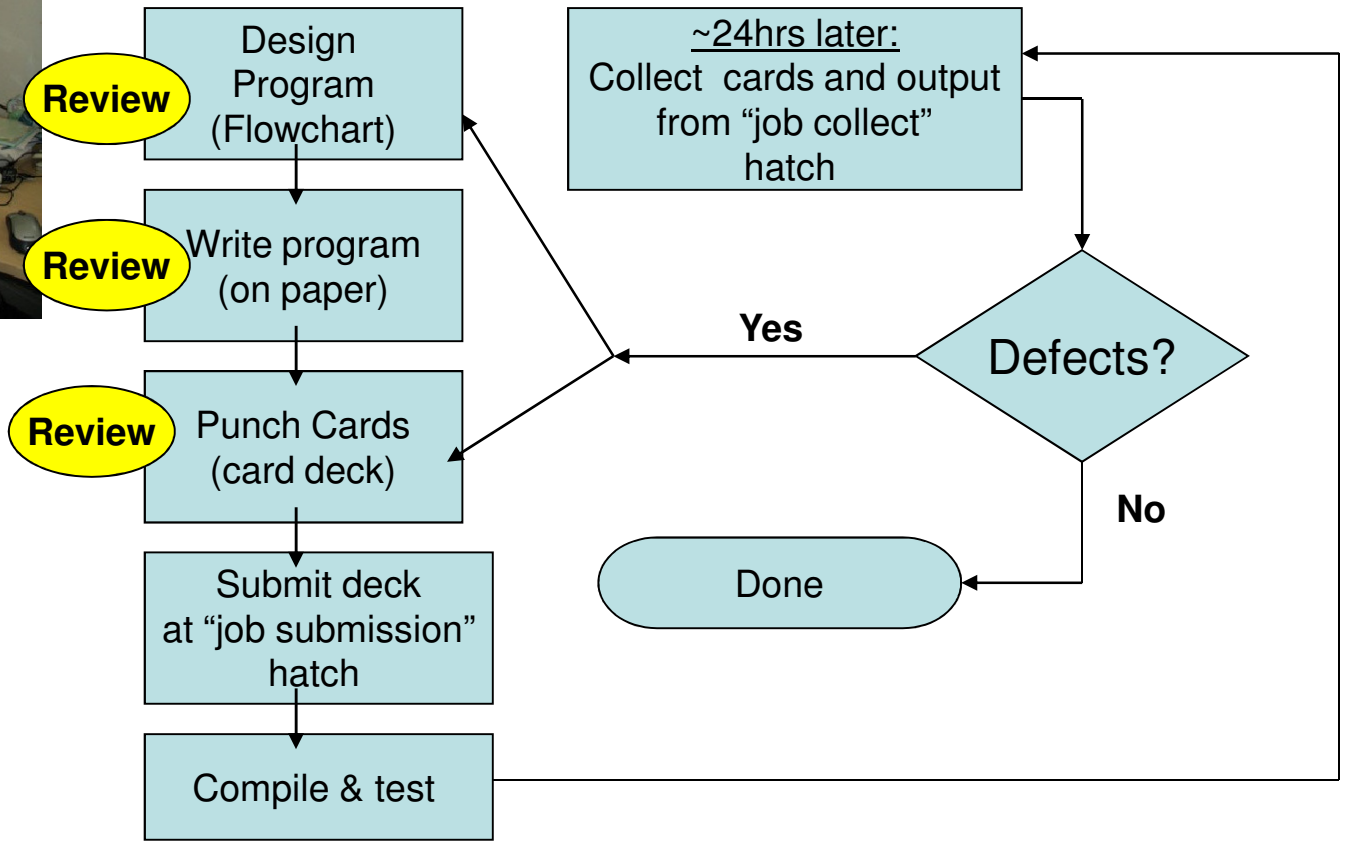


Introducing:
Dwolatzky the Programmer

Wrote his first program in 1970 – Fortran IV on IBM 360.
A few hundred lines-of-code.

Introducing:
Dwolatzky the Programmer

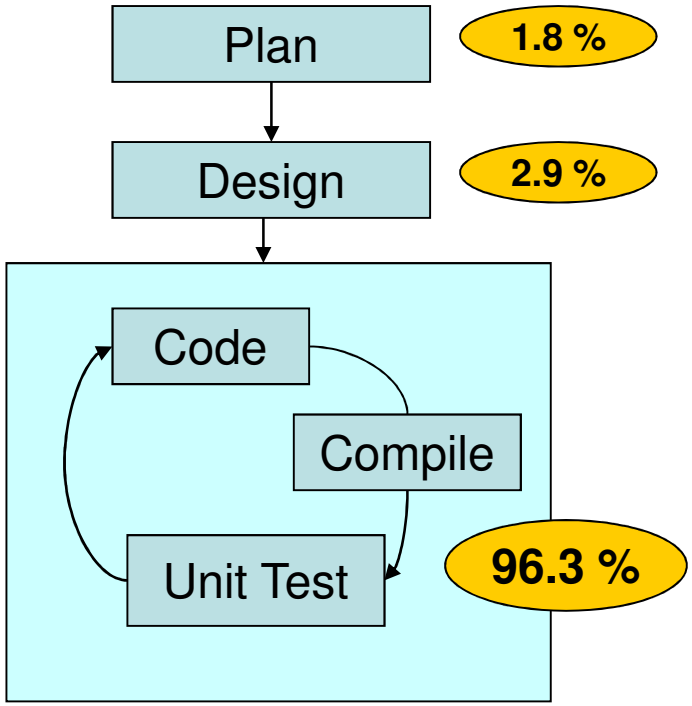
My Personal Software Process in 1970



Introducing:
Dwolatzky the Programmer

My Personal Software Process in October 2009

Plan	1.8 %
Design	2.9 %
Code	53.1 %
Compile	14.8 %
Unit Test	26.8 %



Introducing:
Dwolatzky the Programmer

My Personal Software Process in October 2009

Estimate of size for “Program 2”	135 LOC
Estimate of development time for “Program 2”	240 minutes

Actual size of “Program 2”	361 LOC
Actual development time for “Program 2”	530 minutes

Introducing:
Dwolatzky the Programmer

My Personal Software Process in December 2009

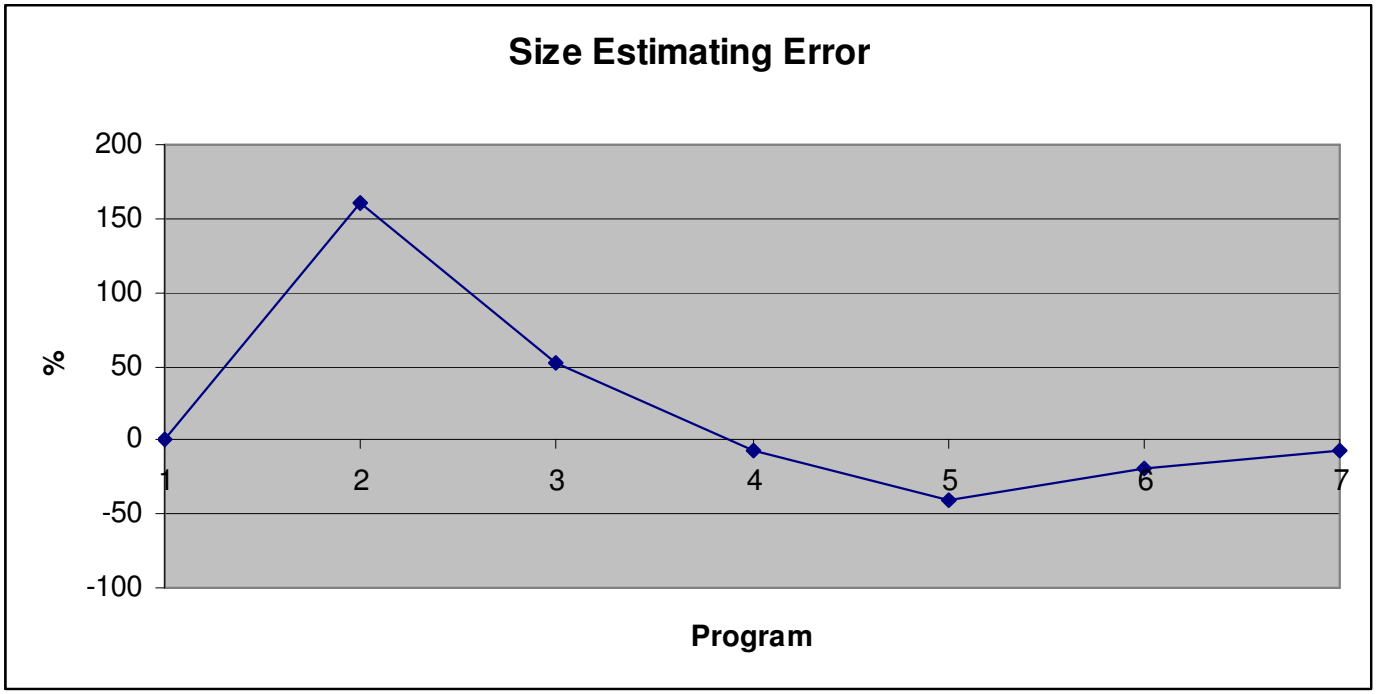
Plan	4.1 %
Design	18.6 %
Design Review	8.7 %
Code	32.6 %
Code Review	8.7 %
Compile	9.1 %
Unit Test	13.2 %
Post Mortem	5.0 %

Introducing:
Dwolatzky the Programmer

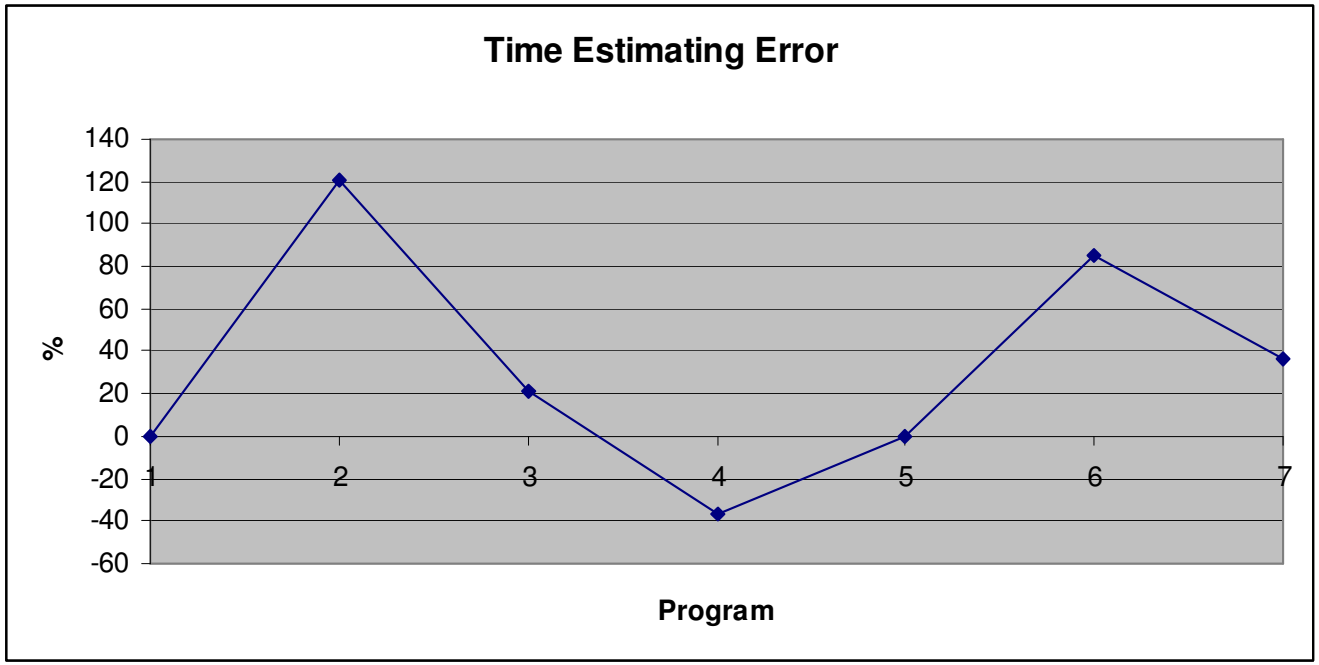
My Personal Software Process in December 2009

Program 7	Estimate	Actual	Error [%]
Size [LOC]	126	118	-6.4 %
Development Time [minutes]	176	242	37 %

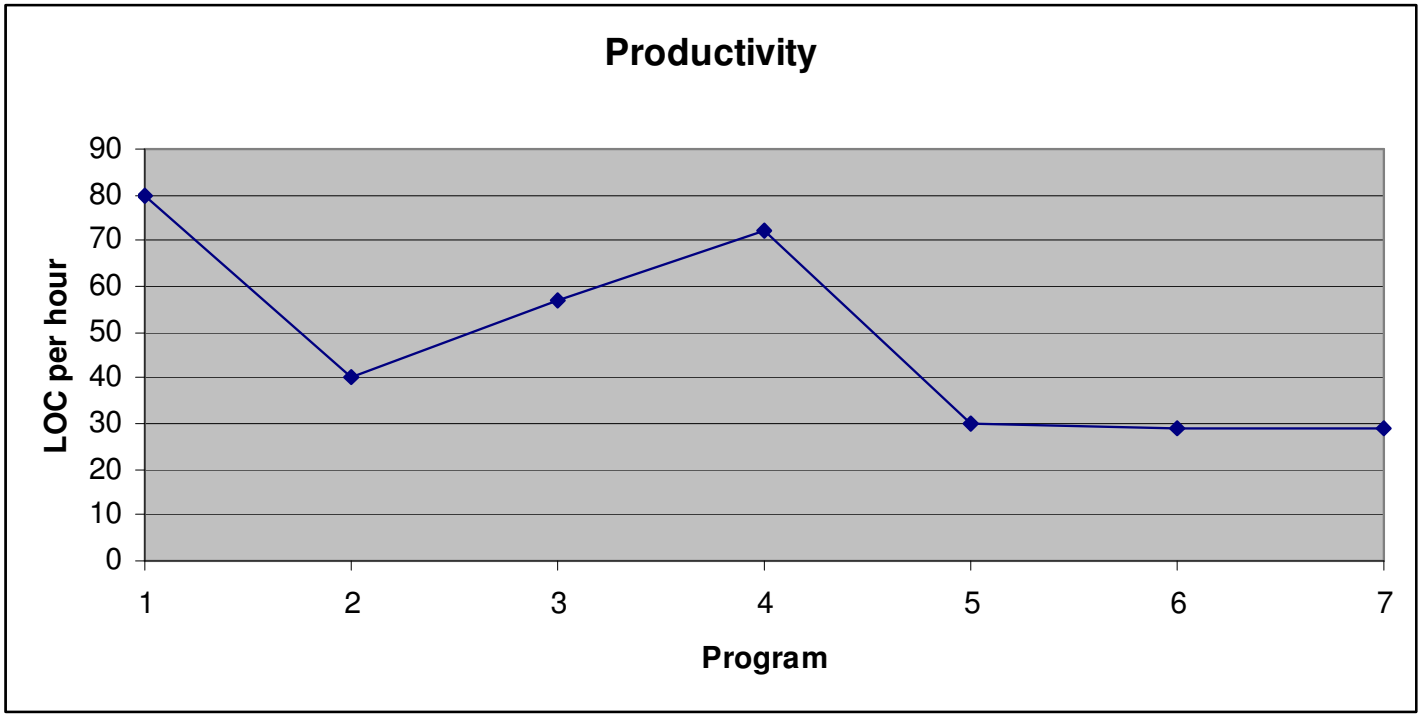
PSP: Barry Dwolatzky



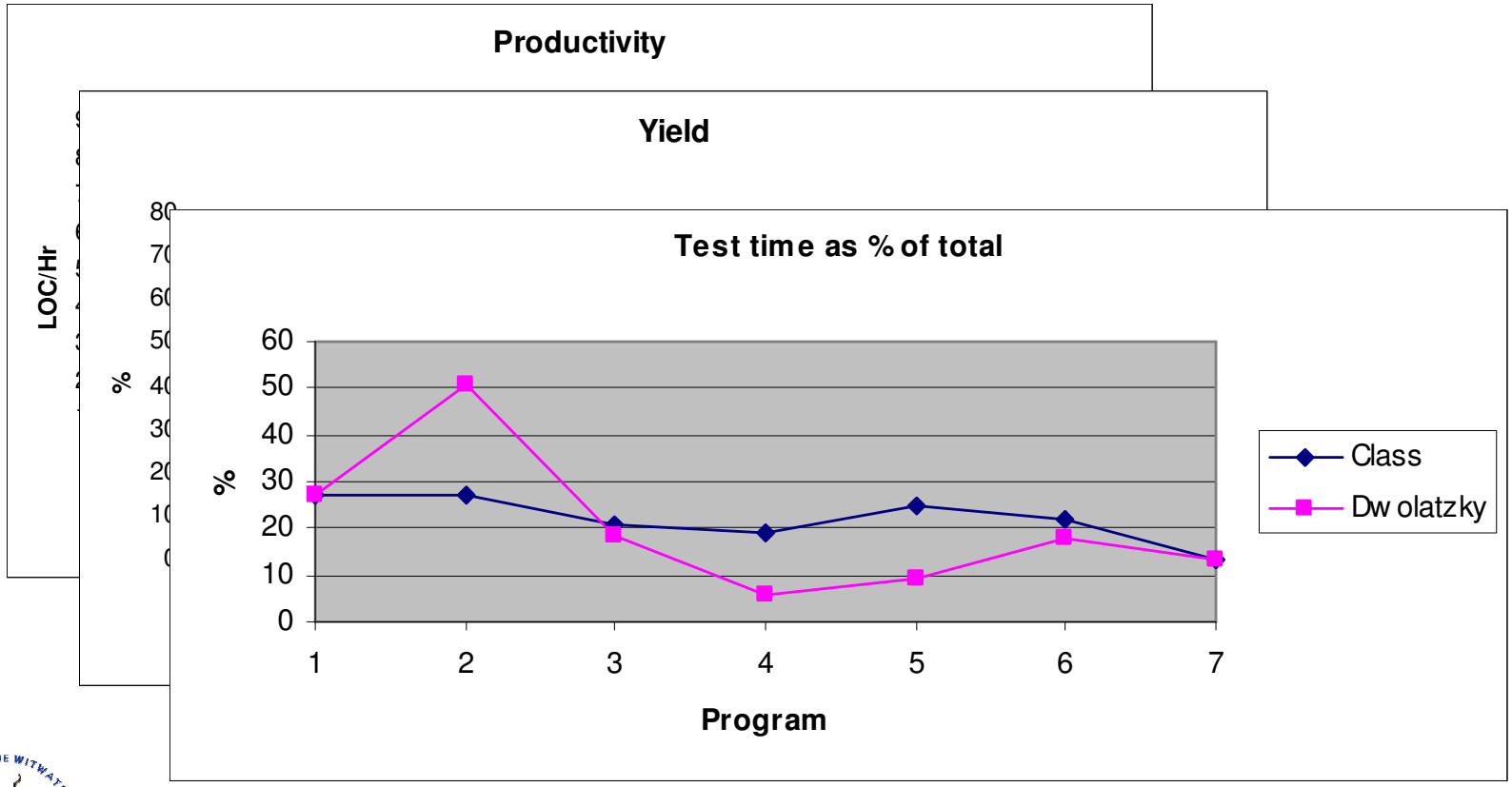
PSP: Barry Dwolatzky



PSP: Barry Dwolatzky



PSP: My whole class

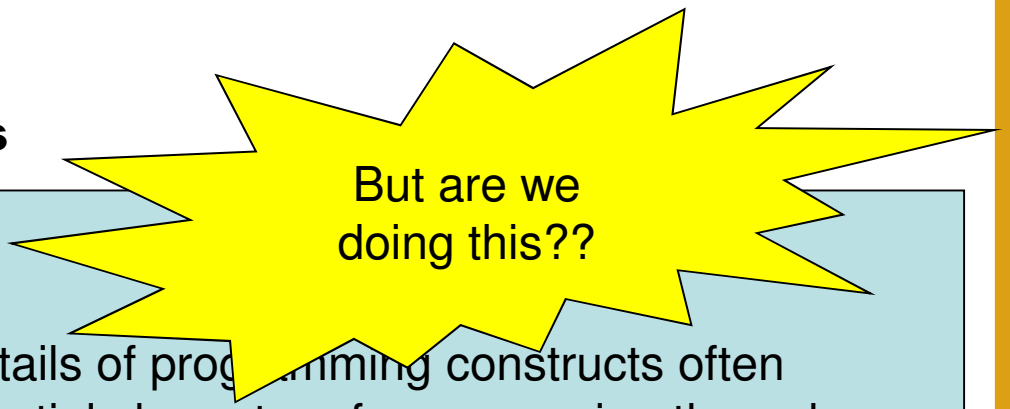


Quality: The secret of PSP

- A programmer always introduces defects
- By removing them in the phase in which they are introduced you remove sources of uncertainty
- Locating defects in unit testing is a slow, unpredictable, frustrating and wasteful activity – **I knew this 40 years ago!**
- **The major lesson from PSP training:**
  - Early and efficient defect removal (high process yield) is not only the key to good quality – it also leads to good estimation of effort.
- **How do we teach this lesson to students learning to program?**

How to teach programming

- **Common approach is to focus on:**
  - **Language syntax**
  - **Algorithmic skills**
  - **Abstraction mechanisms**



IEEE/ACM Joint Task Group 2001

"concentrating on the mechanistic details of programming constructs often leaves students to figure out the essential character of programming through an *ad hoc* process of trial and error.

Such courses thus risk leaving students who are at the very beginning of their academic careers to flounder on their own with respect to the complex activity of programming."

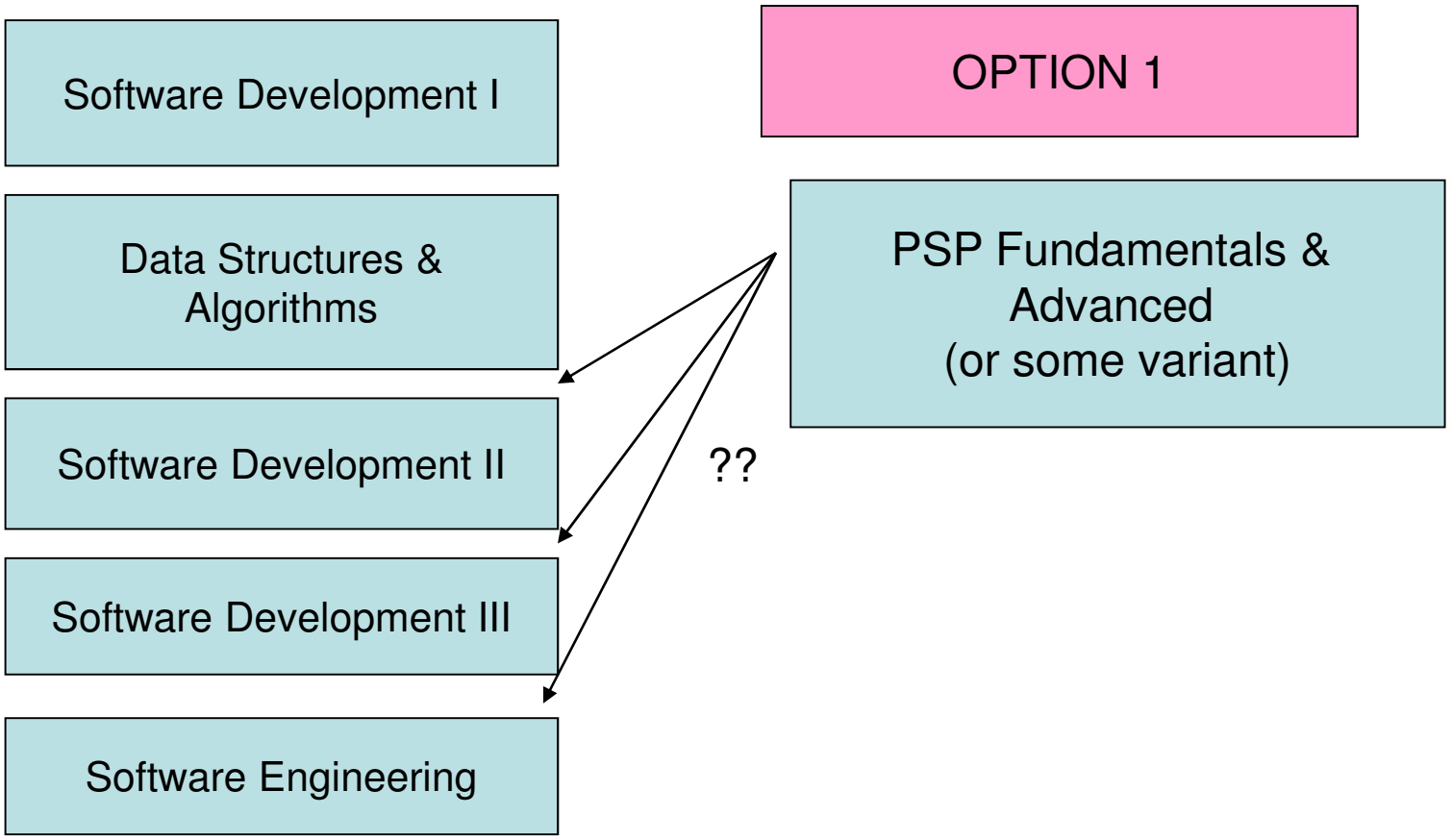
Introducing: Dwolatzky the Professor

- **Guilty as charged!**
- **But … how to introduce lessons from PSP training into a programming course, or sequence of courses?**

Current Software Stream: 4-year BSc(Engineering)

Software Development I	Language syntax; Abstraction mechanisms
Data Structures & Algorithms	Algorithmic skills; Abstraction mechanisms
Software Development II	Software Design; Advanced language syntax; Advanced abstraction mechanisms Testing
Software Development III	Architecture; advanced concepts
Software Engineering	SDLC's ; CMMI; Agile; Project Management Design; No Programming required

Introducing PSP into the curriculum



Introducing PSP into the curriculum

OPTION 2:
More radical

- **When I learnt to program 40 years ago I paid a huge price – in time and grades – for defect found in compile and test.**
- **In the past 40 years we have been seduced by the apparent ease of an interactive environment.**
- **How do we re-discover the lessons learnt with batch processing and punch cards?**
- **The only currency that students respect: GRADES**
- **Teach Programming in 4 stages**

Introducing PSP into the curriculum

OPTION 2:
More radical

- **Stage 1: Getting started**
  - **Enough language syntax, algorithmic skills and abstraction mechanism to write simple programs**
  - **Provide standard checklist for code reviews. This should be modified by student**
  - **Award bonus marks if first compile has fewer than a specified number of defects**

Introducing PSP into the curriculum

OPTION 2:
More radical

- **Stage 2: Advanced programming + process and data collection**
  - **Major focus on advanced programming skills, but …**
  - **Introduce the PSP process steps – particularly design and design review**
  - **Collect time and defect data**
  - **Marks awarded should reward adherence to process and accurate collection of data**
  - **In assessing work do not examine the content of data**
  - **Use defect data to refine checklists**
  - **Award marks for high process yield**

Introducing PSP into the curriculum

OPTION 2:
More radical

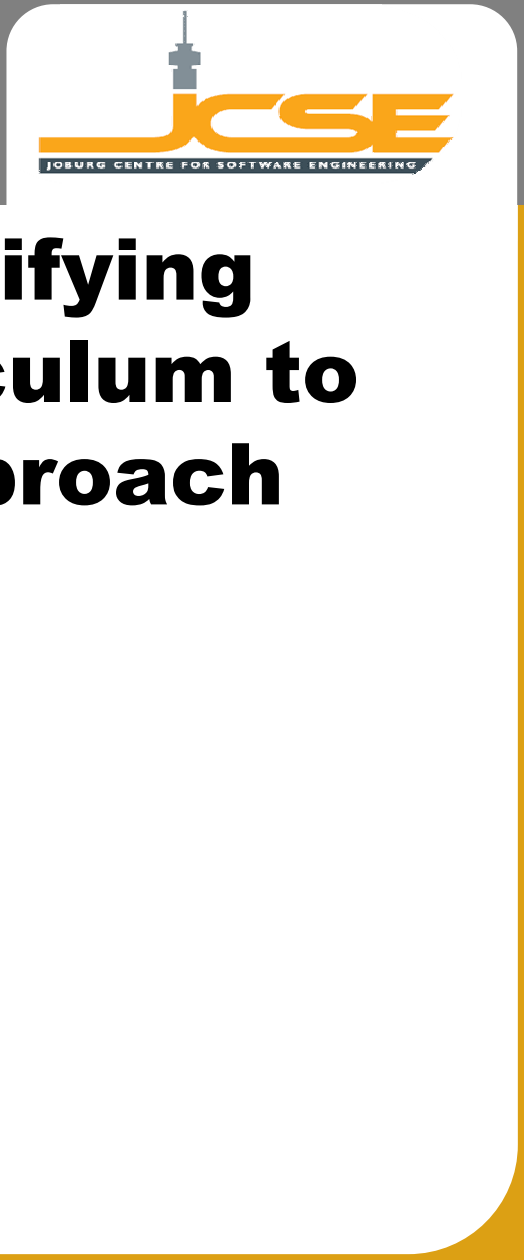
- **Stage 3: Formal PSP Training**
  - **Full 7 or 8 program course**
  - **Emphasis on**
    - ***estimation based on personal data,***
    - ***design and design review, and***
    - ***developing and improving one's own personal process.***

Introducing PSP into the curriculum

OPTION 2:
More radical

- **Stage 4: A capstone project using TSP**
  - **Emphasis on Team Work**
  - **Refining PSP skills within a team environment**

Conclusion



- **Currently working on modifying modules in existing curriculum to introduce this 4 stage approach**