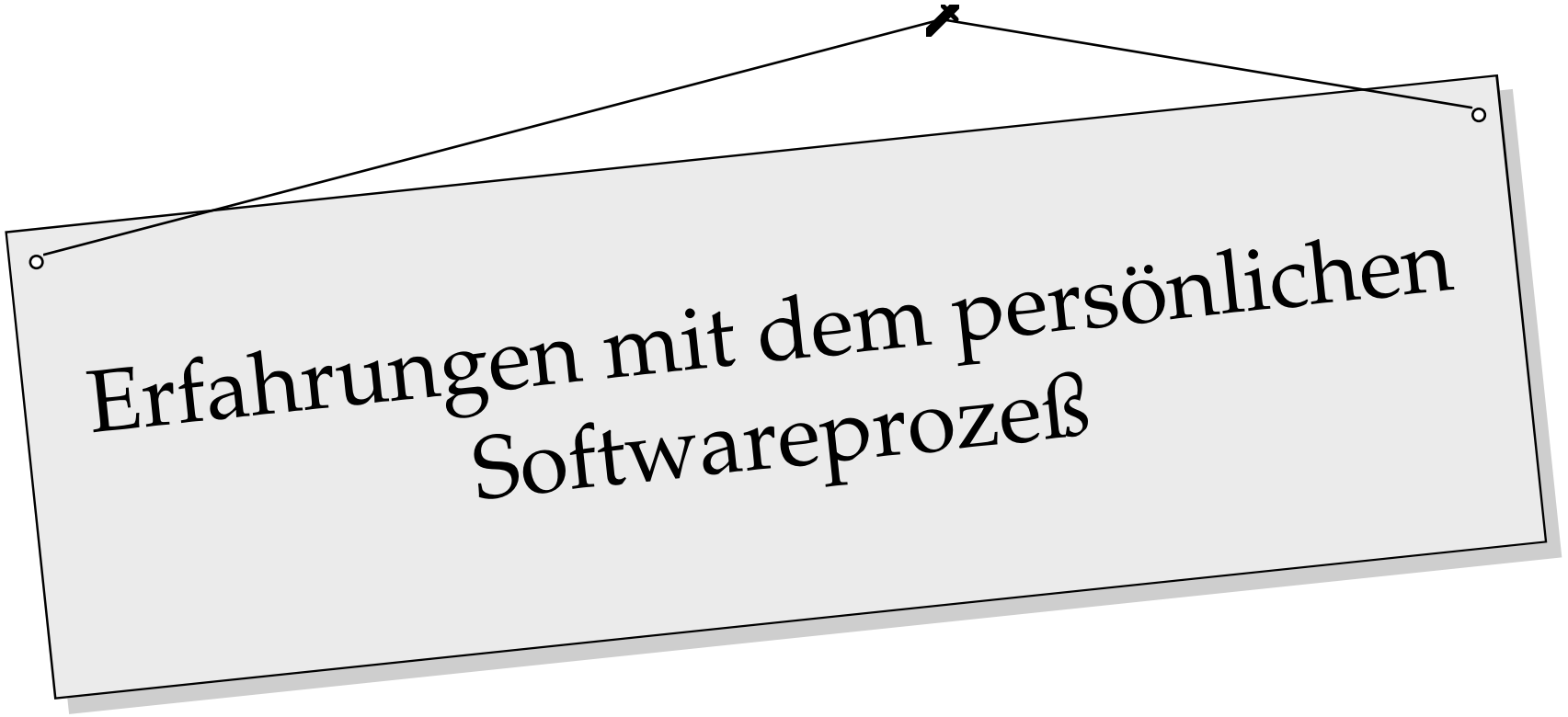


Dr. Lutz Prechelt

Softwarelabor

Fakultät für Informatik

Universität Karlsruhe

Dr. Lutz Prechelt

Softwarelabor

Fakultät für Informatik

Universität Karlsruhe

Warum ?

Um Probleme in der täglichen Arbeit abzumildern.

z.B.

(stets auf der Ebene Ihrer Arbeit als Einzelperson):

- (Planung:)
  - Können Sie genau und zuverlässig Terminzusagen machen?
  - Verlassen Ihre Kollegen sich auf Ihre Terminzusagen?
  - Kennen Sie ihre Produktivität?

Warum ? (2)

- (Definition und Verfolgung:)
  - Könnte ein Kollege Ihre angefangene Arbeit direkt zuendeführen?
  - Wenn Sie verklagt würden, könnten Sie belegen, was Sie getan haben?
  - Wenn ein altes Problem wieder auftaucht, kennen Sie noch die Lösung?

Warum ? (3)

- (Qualität:)
  - Können Ihre Kollegen sich auf die Korrektheit Ihrer Software verlassen?
  - Machen Sie dokumentierte Entwürfe, bevor Sie zu implementieren beginnen?
  - Machen Sie Entwurfsdurchsichten? Codedurchsichten?
  - Benutzen Sie systematische Entwurfsnotationen? Systematische Prüfungsmethoden?
  - Wissen Sie, unter welchen Bedingungen Sie bevorzugt Fehler machen?

Warum ? (4)

- Kennen Sie ihre Stärken und Schwächen?
  Quantitativ?
- Kennen Sie ihre “Lieblingsfehler”?
  Merken Sie, wenn diese drohen?
- Kennen Sie Ihre typischen Fehlerklassen?
  Deren Häufigkeit?
  Kosten?
  Verteilung auf Arbeitsphasen?
- Laufen Ihre Programme manchmal auf Anhieb ganz korrekt?

Darum!

Typische Probleme in der SW-Erstellung:

- geringe Planungssicherheit
- viele Fehler
- unkontrolliertes Arbeiten
- keine systematische Verbesserung der Arbeitsweise

Hohes Verbesserungspotential!

Ziele des persönl. SW-Prozesses (PSP)

- hohe Planungsgenauigkeit
- **hohe Qualität, geringe Fehlerdichte**
- gleiche oder bessere Produktivität
- Selbstbeobachtung und -einschätzung
- **kontinuierliche Leistungsverbesserung**
- auf andere Zwecke als SW-Produktion übertragbar

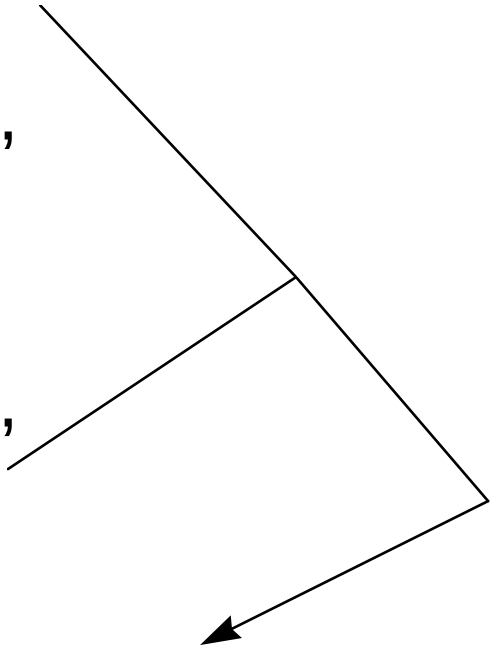
Ansatz

- Selbstdefinierter, dokumentierter SW-Prozeß (**PSP**, *persönlicher* Softwareprozeß)
- Vorgehensvorschläge als Ausgangspunkt
- in Stufen eingeführt
- ständig kontrolliert und verbessert
- Steuerung, Erfolgsnachweis und Motivation durch konkrete Messungen:

Daten, Daten, Daten!

Elemente des PSP

- Ist-Erfassung:
  Zeit (netto, brutto), Programmgröße, Fehleranzahl
- Soll-Erfassung:
  Zeit (netto, brutto), Programmgröße, Fehleranzahl
- Planung:
  Nutzung der Soll/Ist-Daten, kontinuierliches Verbessern
- Systematisierung von Routineaufgaben:
  z.B. Formulare, Entwurfsschablonen



Elemente des PSP (2)

- Begutachtung:
  Frühes Fehlerfinden in Entwurfs- und Codedurchsichten
- Qualitätsmanagement:
  Analyse der Fehlerdaten zur besseren Vorbeugung,
  systemat. Auswahl u. Einsatz neuer Techniken
- stetige Prozessverbesserung:
  Auswahl günstiger Entwicklungsmethoden, Vermeidung von Hindernissen, Einsatz geeigneter Werkzeuge etc.

PSP-Stufen

- **PSP 0**: Ausgangsprozeß, Zeit- und Fehlererfassung, Größenmessen
- **PSP 1**: Größenschätzen, Zeitschätzen, Zeitplanung
- **PSP 2**: Begutachtungen, Qualitätsmanagement, Entwurfsschablonen
- **PSP 3**: inkrementelle Entwicklung

PSP 0

- Detaillierte Zeiterfassung (Entwicklungsphasen)
- Komplette Fehlererfassung
- Fehlertypstandard

Zeit	Nr	Typ	Einfüg	Entfern	Dauer	Beschreibg
14^{20}	1	30	Dsgn	Comp	10	. . . .

PSP 0.1

- Größenmessen (Zählstandard, Zählwerkzeug)
- Prozeßverbesserungsvorschlag
  - registriere Prozeßprobleme, sobald sie auftreten
  - ergänze später Analyse und Verbesserungsvorschläge
  - notiere gelernte Lektionen aus jedem Programm

PSP 1

- Sammlung historischer Größen- und Zeitdaten
- systematisches Größenschätzen aus Grobentwurf
- datenbasiertes Größenschätzen
  - Größenklassen, Objekttypen
  - lineare Regression
- datenbasiertes Zeitschätzen (netto)
  - lineare Regression

PSP 1.1

- Planung von Aufgabenzerlegungen und Arbeitswerten
- Erstellung von Zeitplänen (brutto)
  - tatsächliche erwartete Arbeitszeit pro Tag
  - Setzen und Kontrollieren von Meilensteinen

PSP 2

- Entwurfs- und Codebegutachtung
- Fehlerfindungs-Management
  - Checklisten für Durchsichten
  - Verbesserung über Analyse der Fehlerdaten
  - Steuerung der Begutachtungsgeschwindigkeit
- Fehlervermeidungs-Management
  - Prozessverbesserung aufgrund der Fehlerdaten

PSP 2.1

- Entwurfsschablonen
  - Schnittstellenbeschreibung
  - Zustandsautomat
  - Anwendungsszenarien
- Vollständigkeitsprüfungen
- Konsistenzprüfungen

PSP 3

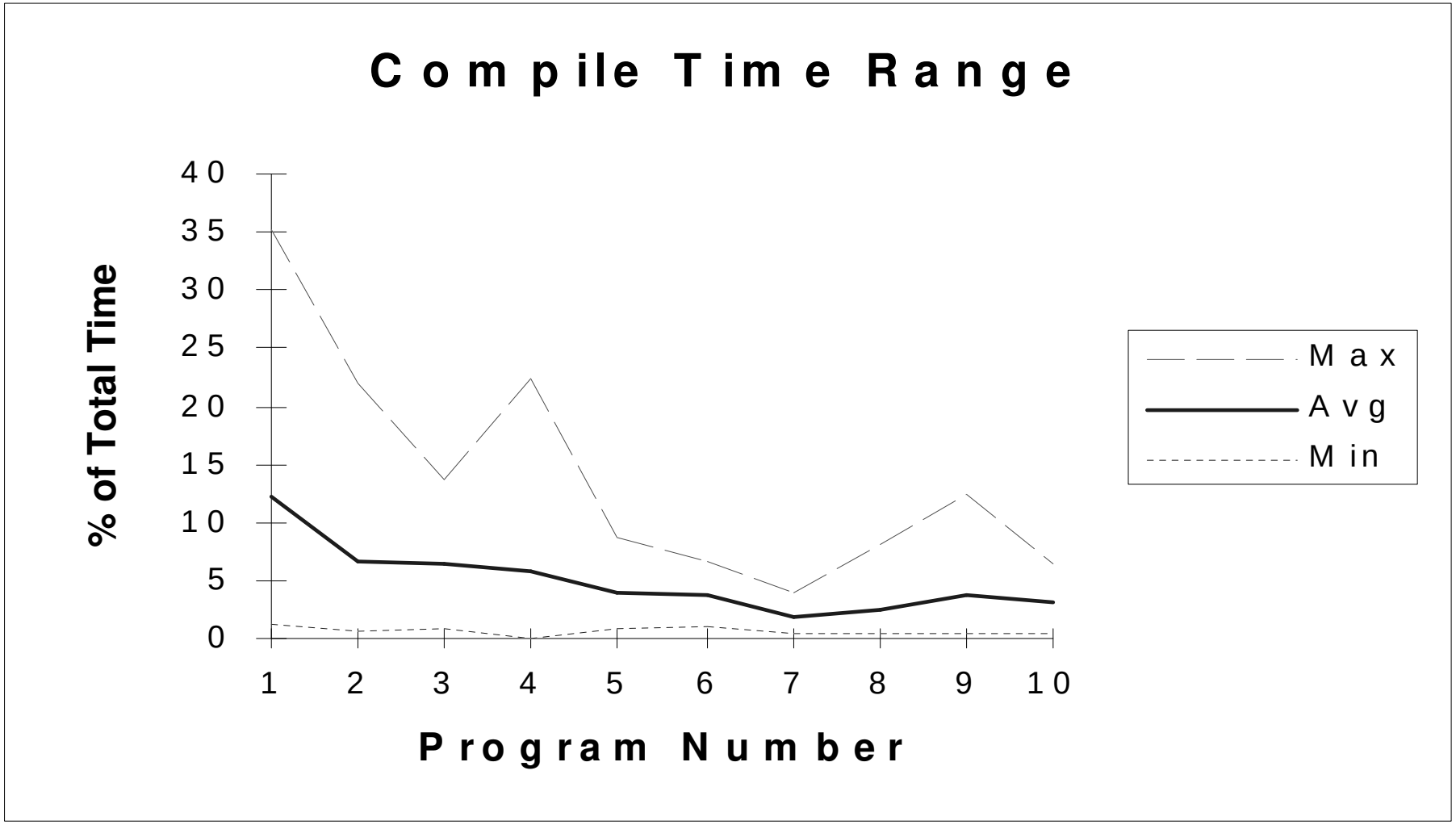
- Zyklische Anwendung von PSP 2.1
  - inkrementelle Entwicklung
  - Qualitätssicherung für jedes Inkrement
  - für große Programme geeignet (aber Einzelperson!)
  - lineare Skalierung, d.h. konstante Produktivität

Wie lernt man das?

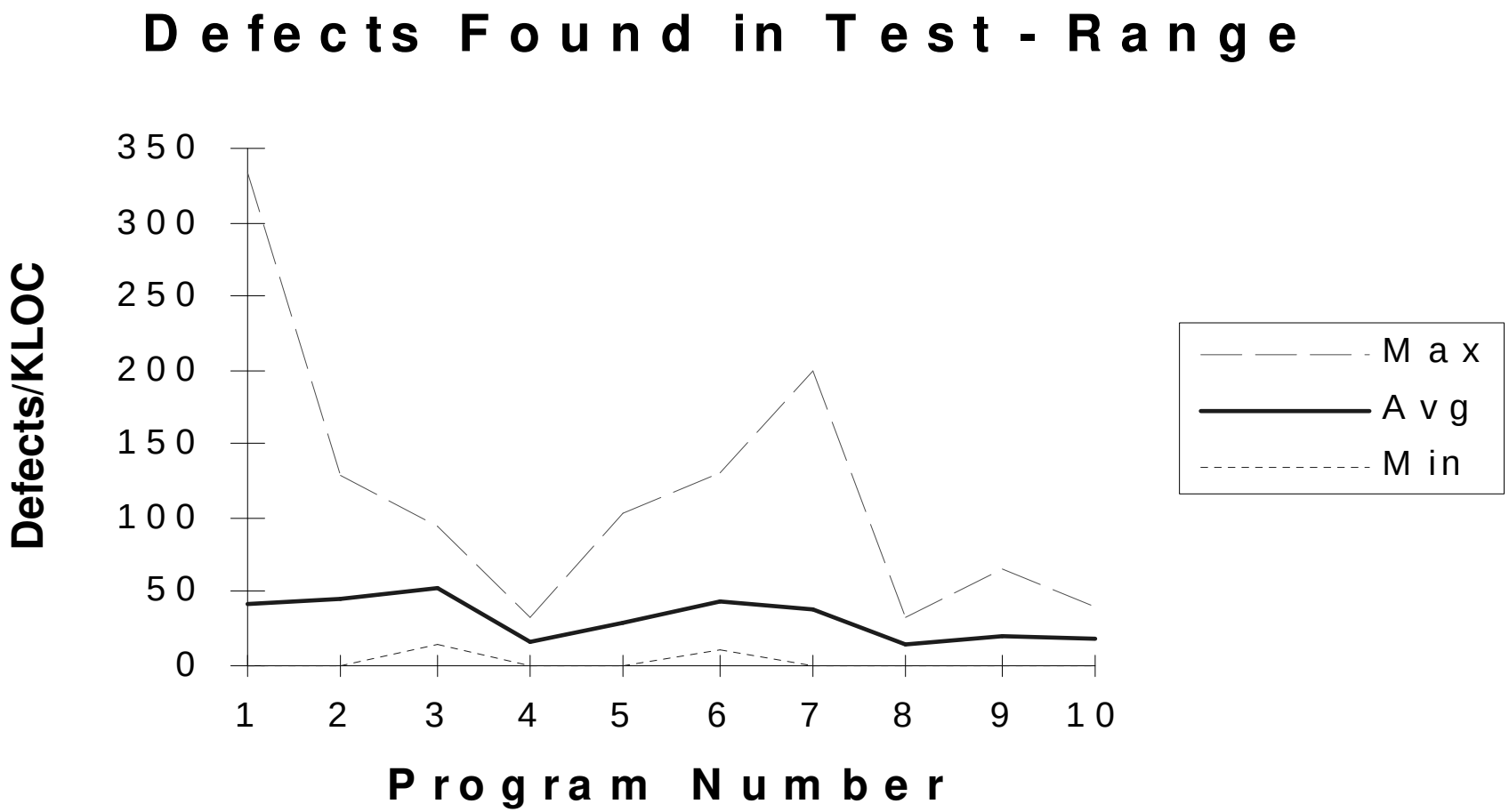
- Kursus über 15 Wochen
- etwa ein ganzer Tag pro Woche
- 10 Übungsprogramme
- 5 Prozessübungen
- beliebige, gutbeherrschte Programmiersprache

- Infrastruktur:
  - Programmierumgebung
  - Tabellenkalkulation
  - genügend Zeit

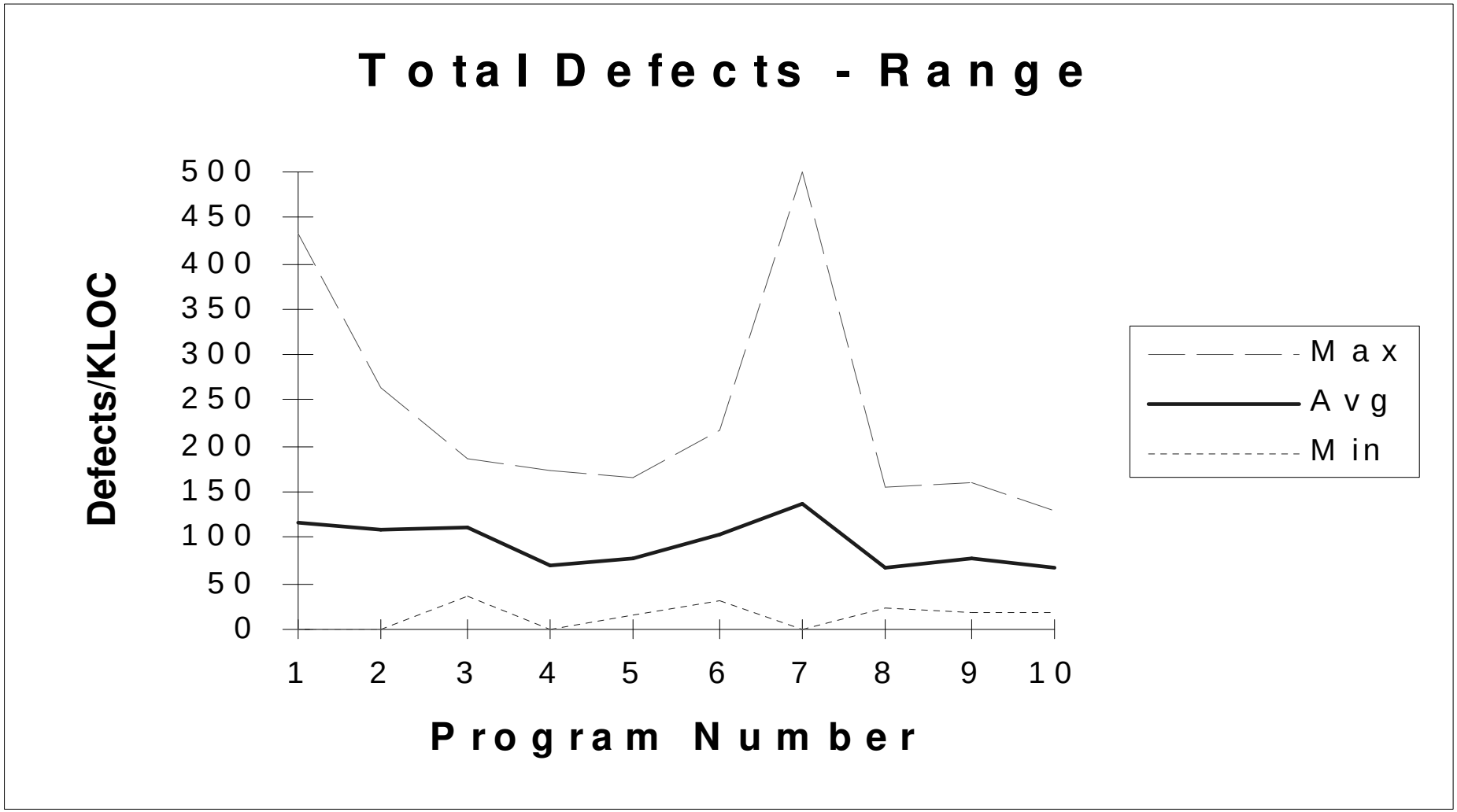
Resultate (F 96): Übersetzungszeit



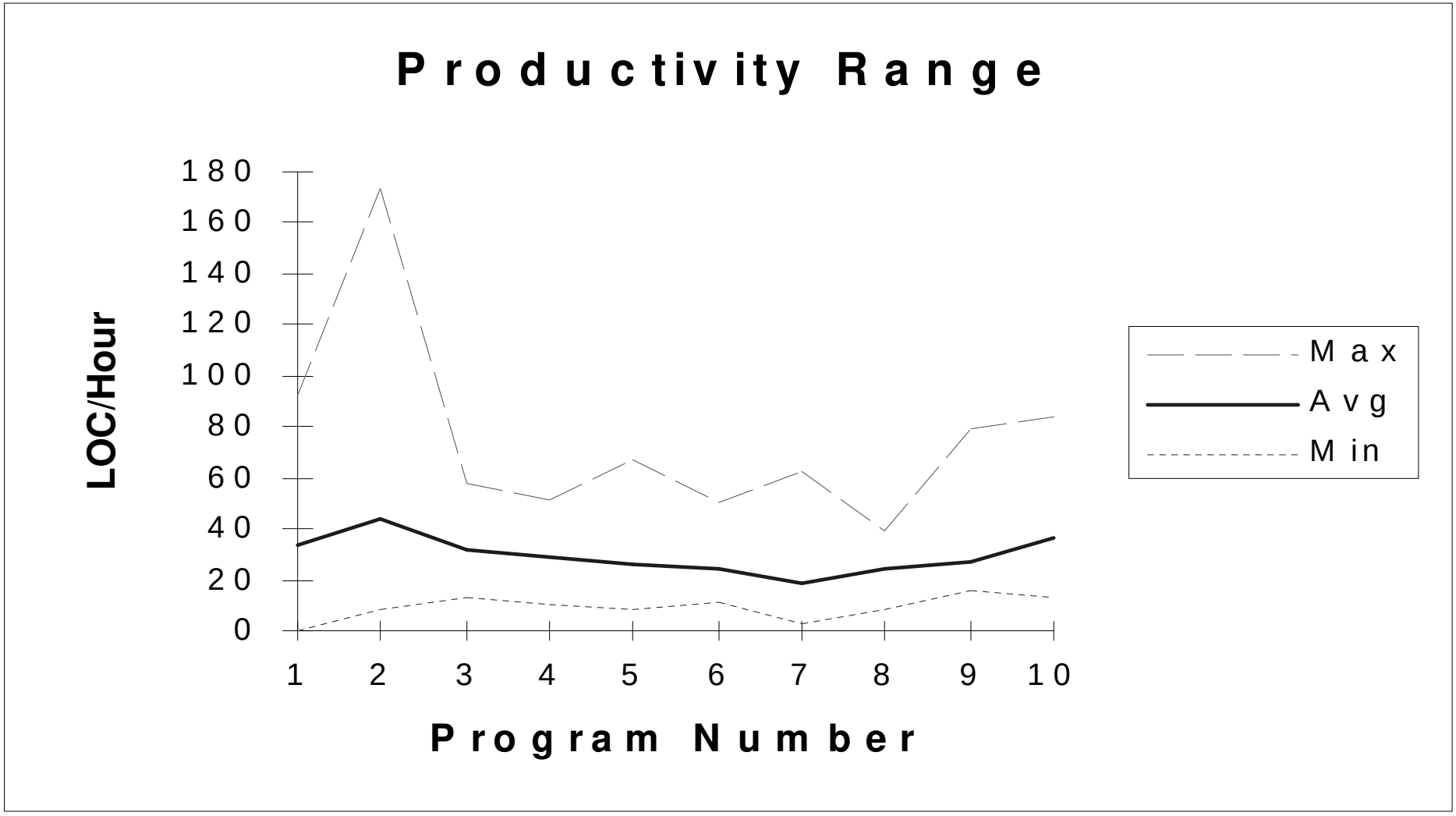
Resultate (F 96): Fehler in Testphase



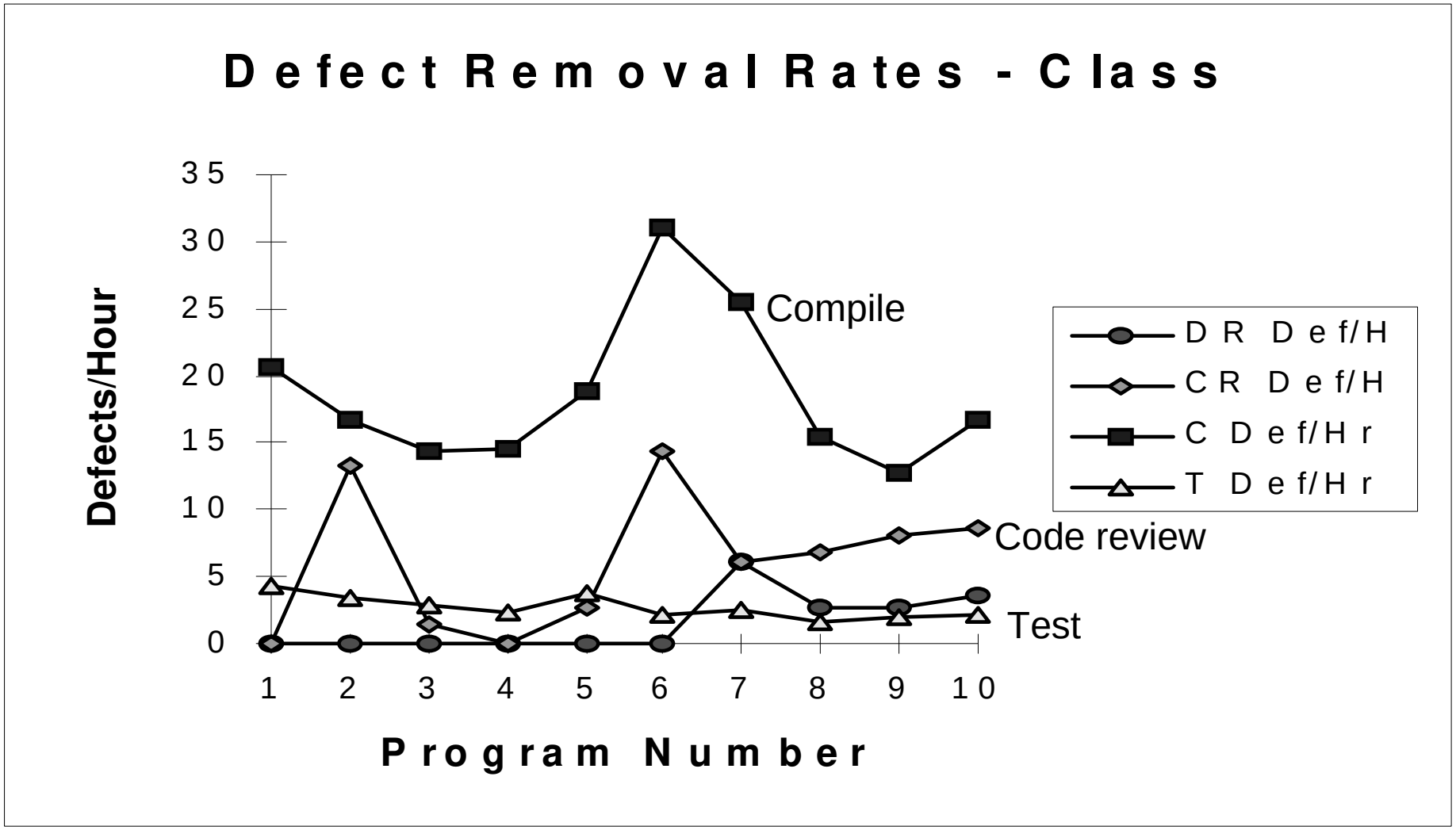
Resultate (F 96): Gesamtfehlerrate



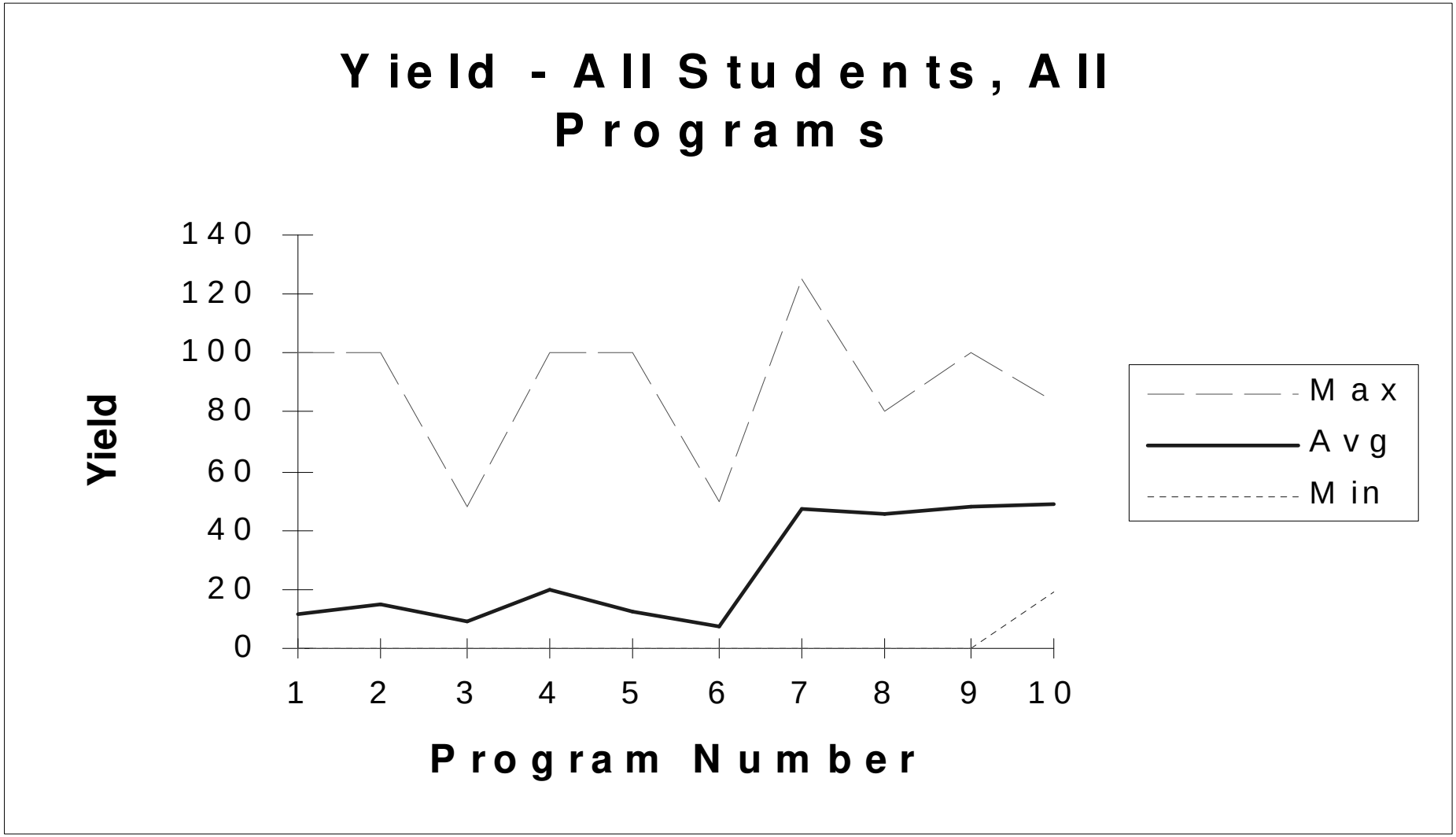
Resultate (F 96): Gesamtproduktivität



Resultate (F 96): Fehlerfindungsraten



Res. (F 96): Ausbeute der Durchsichten



Nebeneffekte

- Bessere Kontrolle über SW-Prozeß, dadurch höhere Arbeitszufriedenheit
- Mehr Vertrauen in die Produkte
- Keine Frustrationserlebnisse durch katastrophale Terminüberschreitung mehr
- Motivation für firmenweite Prozeßverbesserung
- ständiges Weiterlernen
- Auf andere Bereiche übertragbar, z.B.:
  - Schreiben von Dokumenten aller Art
  - Organisationsaufgaben

Unsere Erfahrungen

- **Es funktioniert.**
- Keine Wunderwaffe
- *“Wat dem een sin Uhl”...*
  Nicht jeder hält durch.
  Gegenseitige Motivation wichtig.
- Motivation umso besser, je mehr negative Vorerfahrungen vorliegen.
- Wichtig im Kurs: Konzentration auf das Grundprinzip, nicht auf die Details.
- Selbststudium funktioniert nicht.

Zusammenfassung

- PSP verbessert Planungsgenauigkeit und erhöht Softwarequalität
- Effekt durch Messungen nachweisbar
- PSP verbessert Nachvollziehbarkeit, beeinträchtigt dennoch Produktivität nicht
- PSP steigert Motivation und Arbeitszufriedenheit

➡ ***PSP mildert Probleme bei SW-Herstellung***