Experiment über den Nutzen von Kursen über den „Persönlichen Softwareprozess“ (PSP)

Jan Sebastian Siwy

30. März 2004

Inhaltsverzeichnis

1 Über PSP

Der *Persönliche Softwareprozess* (PSP) ist ein Verfahren, um die eigenen Fähigkeiten zu verbessern, die Dauer für ein Softwareprojekt besser einschätzen zu können und Fehler zu vermeiden.

Bei der Softwareentwicklung werden Termine häufig nicht eingehalten. Dies liegt daran, dass übliche Schätzung viel zu optimistisch sind. Auch mit viel Erfahrung sind solche Schätzungen nur bedingt zutreffend. Die menschliche Psyche möchte nicht wahrhaben, dass bei der Entwicklung Fehler passieren werden. Obwohl frühere Zeitschätzungen nicht aufgegangen sind, werden vergangene Fehler meist als einmalige „Ausrutscher“ eingestuft und nicht hinreichend bei künfiger Terminplanung berücksichtigt. Zudem wird häufig vergessen, dass noch eine Dokumentation zu schreiben ist, so dass dies nicht in die Schätzung mit einbezogen wird. Auch Faustregeln wie „Schätzen und dann verdoppeln“ helfen nur bedingt weiter, da die auf diese Weise geschätzte Zeit viel zu hoch erscheint und diese anschließend wieder kräftig nach unten hin korrigiert wird.

Ein weiterer Aspekt ist, dass Fehler häufig erst sehr spät erkannt werden, unter Umständen erst bei der Auslieferung. Dabei ist hinreichend bekannt, dass Fehler erheblich einfacher zu korrigieren sind, je früher diese erkannt werden. Aus diesem Grund wäre es für jeden Einzelnen gut zu wissen, an welchen Stellen von ihm häufig Fehler einbaut werden. In Kombination mit einer schlechten Zeitschätzung verschlimmert sich zudem noch die Fehlerhäufigkeit, denn unter Zeitdruck sinkt die Sorgfalt.

Es wäre nützlich ein „Rezept“ zu haben, um den Zeitbedarf für künftige Projekte besser einschätzen zu können, da offenbar intuitiv zu wenige Informationen dafür gesammelt werden. Außerdem müssten ein persönliches Fehlerprofil erstellt werden, um aus früheren Fehlern für die Zukunft Schlussfolgerungen ziehen zu können.

1.1 Erstellen eines Logbuchs

Zuerst sammelt man mit etwas Disziplin Daten über seinen persönlichen Arbeitsstil, die dann regelmäßig ausgewertet und einen Pool mit *historischen Daten* für künftige Projekte bilden. Während der Software-Entwicklung führt man ein *Logbuch*, in dem man minutengenau aufschreibt, was man wann gemacht hat. Das heißt, man schreibt die Arbeitsphase (z.B. Entwurf, Codierung, Debugging, Test) auf, in der man sich gerade befindet; der Beginn und das Ende einer Programmeinheit (z.B. Klasse oder Modul) und deren Größe als Anzahl der Quellcode-Zeilen (LOC); nebenbei ausgeführte Tätigkeiten; wann ein Fehler entdeckt und beseitigt worden ist. Dies ist nur ein sehr grobes Muster, welche Informationen für eine Auswertung nützlich sein könnten. Was genau protokolliert werden sollte, muss jeder für sich selbst entscheiden.

Zur Erleichterung sollte man für das Logbuch einen Editor mit einer Zeitstempelfunktionalität benutzen (z.B. UltraEdit oder Emacs).

1.2 Auswerten des Logbuchs

Bei der Auswertung des Logbuchs geht man folgendermaßen vor. Zuerst kategorisiert man die Programmeinheiten nach Typ (z.B. Ein-/Ausgabe oder Datenverwaltung). Danach bestimmt man die benötigte Dauer für jede Programmeinheit. Damit erhält man eine Liste wie viel Zeit eine Programmeinheit einer bestimmten Größe und eines bestimmen Typs erfordert hat. Dies sind dann die historischen Daten.

Wenn man nun die Dauer eines künftigen Projektes schätzen möchte, schätzt man zuerst die Größe des Projektes und teilt es grob ein. Anschließend schätzt man – wiederum nur sehr grob – die Größe der einzelnen Programmteile. Dies macht man nun nicht in LOC, sondern z.B. mit einer vorher festgelegten fünfstufigen Skala von „sehr klein“ bis „sehr groß“. Anhand der historischen Daten kann man nun für die einzelnen Teile die benötigte Zeit folgern.

Fehler werden bei künftigen Projekten nun auch vermieden, da man aus den Einträgen im Logbuch zu Einsichten gelangt, welche Fehler man macht, so dass man durch intensivere Reflexion der eigenen Tätigkeit weniger Fehler macht. Außerdem kann man mit Hilfe des Logbuch bessere Checklisten erstellen, um künftige Codedurchsichten effektiver zu gestalten.

2 Experiment

Es wäre nun interessant zu erfahren, ob die Erkenntnisse aus Kursen über PSP in der Praxis auch tatsächlich umgesetzt werden und ob diese Erkenntnisse zu einem besseren Softwareentwicklungsprozess führen. Das heißt, es ist zu klären, ob die Arbeitweise von Versuchspersonen in Hinblick auf Zuverlässigkeit der Software und Schätzung der benötigten Zeit für ein Softwareprojekt sich bei Teilnehmern eines Kurses über PSP positiv auswirkt. Am Rande wäre zudem interssant, ob vielleicht zudem die Produktivität von Personen, die an einem PSP-Kurs teilgenommen haben, höher ist. In bezug auf den letzten Punkt ist jedoch zu betonen, dass dies keine Kernintention von PSP ist.

2.1 Beschreibung

Zu diesem Zweck haben im Zeitraum Februar 1997 und Oktober 1998 Lutz Prechelt und Barbara Unger ein kontrolliertes Experiment durchgeführt. Bei diesem Experiment sollten die Versuchspersonen ein Programm entwickeln. Die Versuchspersonen waren in zwei Gruppen, eine Versuchs- und eine Kontrollgruppe,

aufgeteilt. Es handelte sich um ein Extra-Subjekt-Design, das heißt, dass jeder Teilnehmer nur eine Aufgabe zu lösen hatte.

Die Aufgabe bestand darin, ein Programm namens „phoneword“ zu entwickeln. Die Funktionalität erinnert an das bei heutigen Handys verbreitete T9. Es soll einer Telefonnummer mit Hilfe eines Wöterbuches und der folgenden Tabelle eine Menge von Wortfolgen zugeordnet werden.

E	J N Q	R W X	D S Y	F T	A M	C I V	B K U	L O P	G H Z
0	1	2	3	4	5	6	7	8	9

Dabei war die Eingabe keine einfache Zahl, sondern eine beliebige Folge der Zeichen /, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Die Zeichen / und - sollten ignoriert werden. Eine Ziffer kann auch für sich selber stehen und muss nicht durch einen Buchstaben ersetzt werden, sofern es im Wörterbuch kein Wort gibt, welches an dieser Stelle der Folge anfangen könnte. Eine solche Telefonnummer konnte maximal aus 50 Zeichen bestehen; das Wörterbuch konnte aus maximal 75.000 Einträgen bestehen. Während der Testphase stand den Versuchspersonen ein Wörterbuch mit 73.113 Einträgen als Textdatei zur Verfügung. Für die Eingabe `3586-75` sollte beispielsweise folgende Ausgabe erzeugt werden:

```
3586-75: Dali um
3586-75: Sao 6 um
3586-75: da Pik 5
```

Es konnte für die Lösung eine beliebige Programmiersprache benutzt werden. Das Hauptkriterium für die Lösung der Aufgabe war die Zuverlässigkeit; die Geschwindigkeit des Programms spielte keine entscheidende Rolle.

Die Versuchspersonen waren 48 männliche Informatik-Studenten. Im Durchschnitt hatte jeder rund 600 Stunden Programmiererfahrung und hatte bisher außerhalb der Uni rund 20.000 Zeilen Quellcode geschrieben.

Die Versuchsgruppe bestand zu Beginn aus 29 Personen, die als Vorbereitung zuvor an einem 15-wöchigen PSP-Kurs teilgenommen haben. In der Versuchsgruppe haben 5 Personen die Teilnahme am Experiments abgebrochen. Die Kontrollgruppe bestand zu Beginn aus 19 Personen, die als Vorbereitung an einem 6-wöchigen kompakten KOJAK-Kurs (Kurs über komponentenbasierte Software, Swing, Beans und RMI) teilgenommen haben. In der Kontrollgruppe haben 3 Personen die Teilnahme am Experiment abgebrochen.

Beide Gruppen wurden während des Experiments genau gleich behandelt. Das heißt insbesondere, dass die PSP-Gruppe keine spezielle Aufforderung erhielt, PSP zu nutzen.

Das Experiment wurde folgendermaßen durchgeführt: Die Versuchspersonen erhielten zuerst einen persönlichen Fragebogen (u.A. zur Ermittlung der oben genannten Programmiererfahrung) sowie die Arbeitsbeschreibung in Papierform.

Anschließend wurden sie um eine Aufwandschätzung gebeten. Erst dann begannen sie die Aufgaben an speziellen UNIX-Rechnern mit präparierter, aber modifizierbarer Arbeitsumgebung zu lösen. An diesen Rechner wurden die Versuchspersonen „aussponiert“. Das heißt, dass alle kompilierten Versionen mit Zeitstempel sowie die Gesamtarbeitszeit protokolliert worden sind.

Wenn eine Versuchsperson der Meinung war, dass ihr Programm korrekt funktioniert, konnte sie einen Akzeptanztest beantragen. Bei diesem Akzeptanztest wurden die Ergebnisse einer zufällige Auswahl von 500 Telefonnummern mit denen einer Referenzimplementierung verglichen. Dabei wurde ein anderes Wörterbuch benutzt als das, welches den Versuchspersonen während des Programmentwurfs zur Verfügung stand. Bei dem Akzeptanztest wurde die Zuverlässigkeit des Programms ermittelt, die folgermaßen definiert war:

$$r = \frac{\text{Anzahl der richtigen Lösungen}}{\text{Anzahl aller Lösungen (richtige und falsche)}}$$

Als „richtige Lösungen“ gelten dabei die Elemente der Schnittmenge der Lösungsmengen des zu testenden Programms und die der Referenzimplementierung; als „alle Lösungen“ gelten dabei die Elemente der Vereinigung der Lösungsmengen. Dies ist deswegen möglich, weil man annehmen kann, dass die Referenzimplementierung korrekt ist. Sie wurde von Prechelt und Unger durch stufenweise Verfeinerung mit halbformaler Verifikation entwickelt. Es wurde noch nie ein Fehler in diesem Programm gefunden.

Wenn bei einem Akzeptanztest eine Zuverlässigkeit $r \geq 95\%$ erreicht worden ist, galt der Test als bestaden und die Versuchsperson erhielt 50 DM als Belohnung. Wurde der Akzeptanztest nicht bestanden, so konnte er wiederholt werden. Jedoch wurden für jeden nicht bestanden Akzeptanztest 10 DM von der Belohnung abgezogen.

2.2 Gültigkeit

2.2.1 Innere Gültigkeit

Die innere Gültigkeit wird bei diesem Experiment durch folgende Störvariablen bedroht:

- Die Einteilung in die Versuchs- und die Kontrollgruppe erfolgte nicht zufällig. Die Zugehörigkeit zu einer Gruppe ergab sich naturgemäß aufgrund der Tatsache, ob die Person zuvor an einem PSP- oder einem KOJAK-Kurs teilgenommen hat. Damit haben sich die Teilnehmer indirekt selber eingeteilt. Prechelt und Unger sind jedoch der Auffassung, dass sich daraus kein bedeutender Unterschied zwischen den beiden Gruppe herleiten lässt.

- Desweiteren könnte die freie Wahl der Programmiersprache die innere Gültigkeit bedrohen. Prechelt und Unger halten diese Störvariable jedoch auch

nicht für entscheidend, weil keine programmiersprachenspezifischen Eigenschaften wie Objektorientierung oder effektive Speicherverwaltung für die Aufgaben von besonderer Bedeutung waren.

2.2.2 Äußere Gültigkeit

Die äußere Gültigkeit (Verallgemeinerbarkeit) wird in diesem Experiment durch folgende Störvariablen bedroht:

- Die Arbeitsbedingungen während des Experiments entsprachen keiner in der Praxis typischen Arbeitsumgebung.
- Die Zeitspanne zwischen dem PSP-Kurs und dem Experiment war sehr gering. Es wäre interessanter zu erfahren, wie die längerfristigen Effekte von PSP sind, da die Vorteile von PSP sich erst durch eine größere Menge von historischen Daten ergeben.
- Die Aufgabenstellung in diesem Experiment war recht „verschult“. Das heißt, es handelte sich um ein recht kleines Programm, die Aufgabenstellung war im Vergleich zur Praxis sehr präzise gestellt, und es gab einen Akzeptanztest, so dass die Versuchspersonen ihre Resultate überprüfen lassen konnten.
- Die Teilnehmer hatten im Vergleich zu „Profis“ weniger Erfahrung.

2.3 Auswertung

2.3.1 Hypothese

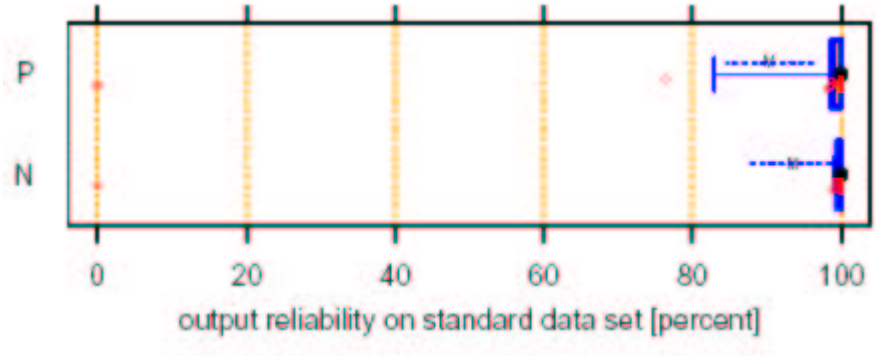
Es wird angenommen, dass die Teilnehmer, die den PSP-Kurs besucht haben,

- zuverlässigere Programme erstellen (siehe 2.3.2),
- die benötigte Arbeitszeit realistischer einschätzen (siehe 2.3.3) und
- auch produktiver arbeiten (siehe 2.3.4).

2.3.2 Zuverlässigkeit

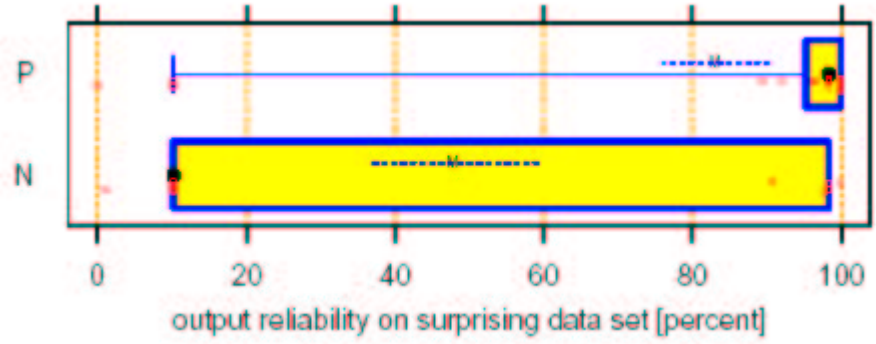
Die Programme wurden im Nachhinein auf ihre Zuverlässigkeit hin untersucht. Diese Messung war strenger als der Akzeptanztest. Es wurden erneut Mengen von Telefonnummern erstellt und zu Gruppen von 100, 1.000, 10.000 und 100.000 Nummern zusammengefasst. Dabei waren die krischen Längen der Nummern (1 und 50) gleichwahrscheinlich, während diese bei dem Akzeptanztest künstlich herausgefiltert worden waren. Die Mengen mit 10.000 und 100.000 Nummern konnten nicht mit allen Programmen getestet werden, weil einige Programme

sehr langsam waren, da die Geschwindigkeit auch kein entscheidendes Kriterium für den Akzeptanztest war. Damit beschränken sich die weiteren Untersuchungen auf Mengen mit 1.000 Nummern.



Wie aus der Grafik ersichtlich wird, hat sich die Zuverlässigkeit in beiden Gruppen nicht besonders unterschieden.

Neben der Menge mit normalen Nummern wurde auch eine „überraschende“ für die Messung benutzt. Diese enthielt keine einzige Ziffer, sondern ausschließlich / und -. Dies war laut Aufgabenstellung eindeutig eine gültige Eingabe. Beim Akzeptanztest wurden solche Spezialfälle bewusst herausgefiltert.



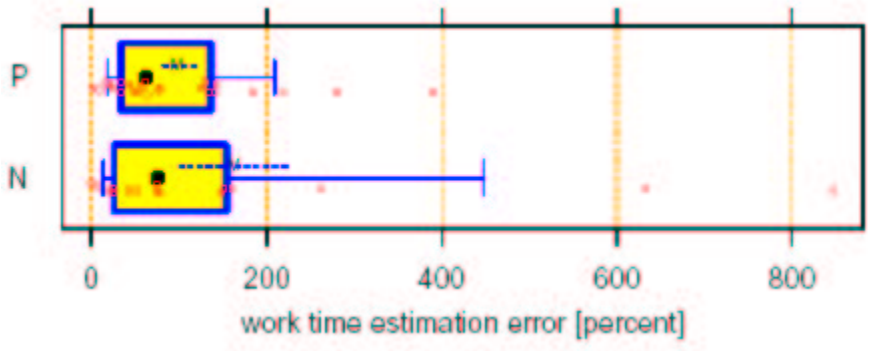
An dieser Stelle wird deutlich, dass die Zuverlässigkeit bei der Gruppe ohne PSP-Kenntnisse erheblich eingebrochen ist.

Prechelt und Unger kommen zu der Schlussfolgerung, dass die Zuverlässigkeit der entwickelten Programme im Allgemeinen nicht bedeutend höher ist. Bei den Spezialfällen ist jedoch ein deutlicher Unterschied zu erkennen, der für die PSP-Gruppe spricht. Damit werde die Hypothese unterstützt, wenngleich nicht bewiesen.

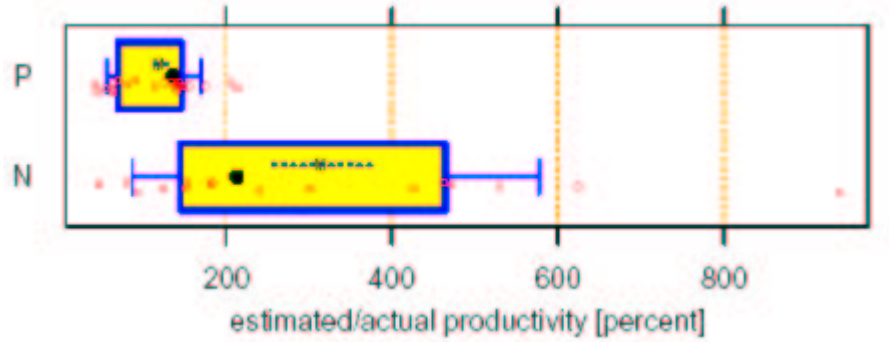
2.3.3 Zeitschätzung

Zur Messung der Qualität der Zeitschätzung wird die Abweichung vom Quotienten aus tatsächlicher und geschätzter Arbeitszeit bestimmt. Dabei wurde festgestellt, dass der Median bei beiden Gruppen nahezu identisch ist. Jedoch gibt

es bei der Gruppe ohne PSP ziemlich deutliche „Abweichler“, wenngleich die Standardabweichung trotzdem keinen großen Unterschied ergab.



Auch wenn bei sich bei der Zeitschätzung keine deutlichen Vorteile bei der PSP-Gruppe gezeigt haben, so hat die PSP-Gruppe zumindest die Größe des Programms besser geschätzt. Dass die PSP-Gruppe trotzdessen nicht zu einer besseren Zeitschätzung gekommen ist, scheint daran zu liegen, dass die Personen bisher noch zu wenige Informationen für ihre Arbeitsgeschwindigkeit gesammelt haben (siehe 2.2.2).



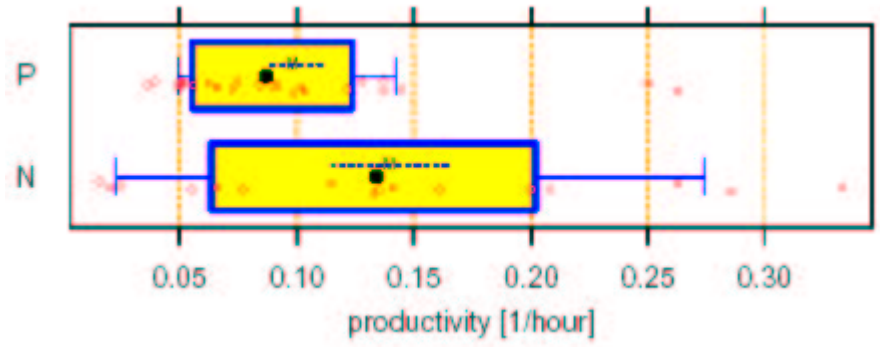
Prechelt und Unger kommen damit zu der Schlussfolgerung, dass die Zeitschätzung bei der PSP-Gruppe nicht präziser ist. Die Hypothese werde damit nicht unterstützt.

2.3.4 Produktivität

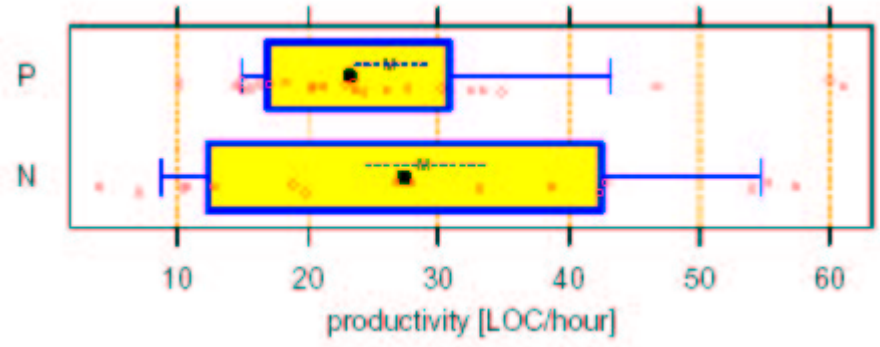
Für die Betrachtung der Produktivität musste erst definiert werden, was unter diesem Begriff zu verstehen ist. Es wurden mehrere Ansätze durchgeführt:

- Die Produktivität wird definiert als die Anzahl der gelösten Aufgaben pro Zeiteinheit. Dabei wurde gemessen, dass diese Größe bei der PSP-Gruppe etwas geringer ist. Dies geht damit einher, dass die PSP-Gruppe im Allgemeinen auch längere Programme schreibt, insbesondere deswegen, weil sie

sich intensiver mit Fehlerbehandlung beschäftigt hat. So sind z.B. bei den Java-Programmierern der PSP-Gruppe deutlich seltener leere `catch`-Blöcke zu beobachten.



- Die Produktivität wird definiert als LOC pro Zeiteinheit. Bei dieser Messgröße sind die Unterschiede zwischen beiden Gruppe bereits deutlich geringer.



Prechelt und Unger kommen damit zu der Schlussfolgerung, dass die Produktivität bei der PSP-Gruppe nicht höher ist.

2.3.5 Varianz

Als zusätzliches Ergebnis war zu beobachten, dass die alle Messungen bei der PSP-Gruppe „näher beieinander“ liegen, das heißt, dass die Varianz der Messwerte geringer ist. Dies kann für Teamwork von Vorteil sein, da alle Team-Mitglieder besser zusammenarbeiten können.

2.3.6 Nutzung von PSP

Es konnte nur bei 6 Versuchspersonen der PSP-Gruppe ein Beweis gefunden werden, dass sie PSP auch tatsächlich genutzt haben. Dieser Beweis bestand darin,

dass ein Logbuch oder PSP-Formular für die Zeitschätzung auf dem Rechner, auf welchem die Aufgabe gelöst wurde, gefunden worden ist. Überraschend ist zudem, dass die Quote der „Abbrecher“ bei denjenigen, die PSP bewiesenermaßen genutzt haben, sehr hoch ist. Es wird vermutet, dass die schwächeren Teilnehmer wahrscheinlich PSP benutzen, weil sie der Meinung sind, dass PSP ihnen dabei hilft, diesen Unterschied auszugleichen.

2.3.7 Weitere Ergebnisse

Desweiteren konnte noch beobachtet werden, dass die Versuchspersonen in der PSP-Gruppe weniger Zeit benötigt haben, um die Fehler zu beseitigen; sie haben weniger triviale Fehler gemacht (Compilierfehler). Außerdem war zu beobachten, dass in der PSP-Gruppe mehr kommentiert worden ist.

2.3.8 Schlussfolgerung

Die Resultate sind eher enttäuschend. Die Gründe dafür, dass die Versuchspersonen, die an dem PSP-Kurs teilgenommen haben, die Erwartungen nicht besonders gut erfüllt haben, könnten darin bestehen, dass sich die Erkenntnisse aus den PSP-Kursen erst längerfristig auswirken. Außerdem deuten die wenigen Beweise für PSP-Nutung darauf hin, dass von vielen PSP überhaupt nicht benutzt worden ist.

Daher stellt sich die Frage, wieso PSP so selten zum Einsatz kam. Es werden drei mögliche Gründe vorgestellt:

- Aufgrund des unterschiedlichen Charakters oder aufgrund kultereller Unterschiede wurde von einigen PSP sofort aufgenommen, während andere nicht die notwendige Selbstdisziplin aufgebracht haben, um die historischen Daten zu sammeln.

- Das Projekt erschien den Versuchspersonen möglicherweise zu klein, um PSP zu nutzen. Bei der Befragung nach dem PSP-Kurs haben viele angegeben, dass sie der Meinung sind, dass PSP nur für große Projekte von Nutzen ist.

- Es gab keine explizite Aufforderung PSP zu nutzen. Nach Aussagen von Watts Humphrey ist ein wesentlicher Bestandteil von PSP, dass man die Teilnehmer auffordert PSP zu nutzen. Dies hätte an dieser Stelle jedoch die innere Gültigkeit gestört.

Nichtsdestotrotz halten Prechelt und Unger PSP für erstrebenswert, was von dem Experiment unterstützt werde. Es müsste jedoch noch besser verstanden werden, wie man es schafft, dass die Teilnehmer der PSP-Kurse die darin erlernten Erkenntnisse auch tatsächlich benutzen.

3 Bewertung

3.1 Experiment

3.1.1 Innere Gültigkeit

In bezug auf die innere Gültigkeit sind Prechelt und Unger der Meinung, dass die Störvariable, dass die Teilnehmer nicht zufällig auf die Gruppen eingeteilt worden sind, nicht entscheidend ist. Dies wird damit begründet, dass viele der Teilnehmer später auch jeweils den anderen Kurs besucht haben. Ich bin jedoch der Meinung, dass die Störvariable durchaus entscheidend sein kann, wenngleich nicht klar ist, wie sie sich auswirken würde. Die Wahl eines Kurs kann durchaus auf bestimmte Interessen, Fähigkeiten oder gerade den Mangel bestimtmen Fähigkeiten hindeuten. Auch wenn viele der Teilnehmer im späteren Verlauf jeweils den anderen Kurs auch besucht haben, so gibt die erste Wahl zumindest eine gewisse Präferenz an.

Desweiteren bin ich der Meinung, dass es nicht sinnvoll war, den Versuchspersonen zu ermöglichen, unterschiedliche Programmiersprachen zu nutzen. Die Ergebnisse sind damit schlechter vergleichbar. Ich halte den Einfluss aufgrund unterschiedlicher Programmiersprachen für größer, als wenn ein Teilnehmer eine Programmiersprache benutzen muss, die ihm weniger vertraut ist.

3.1.2 Äußere Gültigkeit

Die Randbedingungen für das Experiment halte ich für recht ungeschickt gewählt. Die PSP-Gruppe hatte nämlich nur sehr wenig Erfahrung mit PSP. Zum Einen lag der Kurs nicht besonder weit zurück, zum Anderen haben die Teilnehmer des Kurses den Nutzen erst bei großen Projekten für sinnvoll gehalten. Damit sind keine sinnvollen Bedingungen gegeben, welche eine Hypothese zulassen, dass PSP hier zu deutlich besseren Ergebnissen führen würde.

Die Frage, ob es sinnvoll gewesen wäre, die Versuchspersonen in der PSP-Gruppe aufzufordern PSP zu nutzen, möchte ich auf jeden Fall bejahen. Es würde bei einem sonst gleichen Experimententwurf sicherlich die innere Gültigkeit stören. Dies ließe sich jedoch durch eine Anpassung der Fragestellung kompensieren. Man könnte fragen, wie die Ergebnisse in Hinblick auf Zeitschätzung und Zuverlässigkeit bei PSP-Nutzern und Nicht-PSP-Nutzern aussehen.

Es wurden nur wenige Beweise für die Nutzung von PSP gefunden. Hier bin ich der Meinung, dass sich aufgrund dessen eine Vermutung treffen lässt, ob die Versuchsgruppe PSP nutzt oder nicht. Die Erstellung eines Logbuchs während des Experiments hätte die Ergebnisse wahrscheinlich nicht beeinflusst, weil sich ein ausgewertetes Logbuch nur für künftige Projekte nutzen lässt. Um von PSP zu profitieren, hätte die Versuchsgruppe bereits vor dem Experiment genügend historische Daten sammeln müssen.

3.1.3 Glaubwürdigkeit

Das Ergebnis war bei diesem Experiment leider etwas enttäuschend. So erscheint mir die Messung der Zuverlässigkeit mit der „überraschenden“ Menge von Telefonnummern recht konstruiert. Sicherlich gehört es zu einem zuverlässigen Programm, dass es auch mit ungewöhnlichen Eingaben zurecht kommen muss, jedoch waren diese sehr speziellen Eingaben das ausschlaggebene Unterscheidungsmerkmal in Hinblick auf die Zuverlässigkeit. Dies halte ich für übertrieben.

3.2 Vorschlag

Die Zielsetzung des Experiments halte ich für recht verschwommen. Es ist – meiner Meinung nach – nicht sinnvoll nach der Auswirkung eines Kurses über PSP zu fragen. Die direkte Auswirkung ist, dass die Teilnehmer PSP nutzen oder nicht (oder ein bisschen). Um dies festzustellen würde eine einfache Umfrage ausreichen.

Außerdem könnte man untersuchen, ob aufgrund der Nutzung von PSP die Qualität der Zeitschätzung und die Zuverlässigkeit (und vielleicht die Produktivität) steigt. Dazu könnte man ein Experiment mit Personen durchführen, die explizit PSP nutzen, und welchen, die es definitiv nicht nutzen. Bei diesem Experiment würde man sicherlich ein noch größeres Problem mit der Einteilung der Teilnehmer in eine Versuchs- und Kontrollgruppe bekommen.

Das in diesem Aufsatz behandelte Experiment versucht beides zu verbinden, was zur Folge hat, dass keine bedeutenden Unterschiede zwischen den Gruppen entstehen, wenngleich ein Vorteil von PSP vermutet werden kann. Aufgrunddessen wurde das Ziel des Experiments verfehlt.

4 Literaturverzeichnis

- Karlsruhe Empirical Informatics Research Group (EIR): *PSPe: on validating the Personal Software Process (PSP) methodology.* `http://www.ipd.uka.de/EIR/pspe/`
- Lutz Prechelt: *Der persönliche Softwareprozeß.* c't 1998, Heft 19, S.174-175
- Lutz Prechelt, Barbara Unger: *A controlled experiment measuring the effects of Personal Software Process (PSP).* IEEE Trans. on Software Engineering. Vol. 27, No. 5, May 2000, pages 465-472