Design, Durchführung und Präsentation von Experimenten in der Softwaretechnik

Darstellung grundlegender Prinzipien und Fehler bei Aufbau, Durchführung und Präsentation von Experimenten anhand der Analyse einer Diskussion über den empirischen Nutzennachweis des Einsatzes formaler Methoden in der Softwareentwicklung

Jan Klauck

Arbeit im Rahmen des Seminars „Empirische Forschungsmethoden in der Softwaretechnik“ im WS 2003/2004

Freie Universität Berlin
Institut für Mathematik und Informatik

Inhaltsverzeichnis

1. Vorbemerkungen

1.1 Vorbemerkungen und Einordnung der Arbeit

Diese Arbeit entstand im Rahmen des Seminars „Empirische Forschungsmethoden in der Softwaretechnik“ (WS 2003/2004). Grundlage ist die Diskussion dreier Papers im „IEEE Transactions in Software Engineering“ [1], [2], [3].

Ich hoffte zunächst, eine Ausarbeitung über den empirischen Nutzen formaler Methoden in der Softwareentwicklung bearbeiten zu können. Um es gleich zu sagen: Es stellte sich heraus, daß sich das behandelte Paper dazu als ungeeignet erwies und meine Konzentration so auf einen anderen Aspekt gelenkt wurde, nämlich auf den Aufbau, die Durchführung und Präsentation empirischer Experimente.

Die Relevanz dieser drei Punkte ist nicht zu unterschätzen, und ich werde im Anschluß an die Zusammenfassung der Papers die Zusammenhänge darstellen.

Wenn es um die Beurteilung von Methoden in der Softwaretechnik geht, dann sind besonders die Punkte zu klären, ob der behauptete Nutzen mit dem tatsächlichen Nutzen übereinstimmt, ob es Bereiche gibt, in denen keine oder gar nachteilige Wirkungen erzielt werden, ob andere Methoden ähnliche oder bessere Resultate produzieren und wenn ja, mit welchem Aufwand diese erreicht werden.

Für den realen Nutzen formaler Methoden im Vergleich zu anderen Verfahren stehen die Belege noch aus.

Es gibt genug Studien, die zeigen, daß ihr Einsatz zu sicherer Software führt – meistens in Verbindung mit anderen Faktoren wie einer Programmiersprache, die fehlervermeidende Codierung fördert (bspw. Ada 95) oder der Anwendung einheitlicher Coding Styles (verbesserte Wartbarkeit).
Und so ist der Einsatz formaler Methoden zu einem unverzichtbaren Bestandteil zur Produktion missions- und sicherheitskritischer Software (bspw. Luft- und Raumfahrt, Banken, Behörden) geworden.

Die Frage ist hierbei nicht, *ob* man durch formale Methoden sichere Software erstellt, sondern ob sich der dazu erforderliche höhere Aufwand an Zeit und Komplexität positiv gegen weniger umfangreiche bzw. andere Verfahren abhebt.

Das zuerst behandelte Paper beansprucht den Nachweis geführt zu haben, daß der Einsatz formaler Methoden zu besserer Software führt. So interessant das Experiment auch ist, es zeigt aber gerade *das* nicht.
Die Umstände, die zu meiner Aussage führen, werden in der vorliegenden Arbeit behandelt.

Dem interessierten Leser gebe ich zunächst eine kurze Darstellung über formale Methoden, anschließend fasse ich die Papers zusammen und folge mit meiner Beurteilung. Zuletzt füge ich Richtlinien für Aufbau, Durchführung und Präsentation von Experimenten an, die sich besonders aus der Diskussion der Papers herausbilden liessen.

1.2 Formale Methoden

Formale Methoden sind ein Ansatz, mit mathematischen Mitteln der Spezifikation und Verifikation von Softwaresystemen bzw. Systemen allgemein zu gewährleisten, daß gewünschte Funktionen zum gewünschten Zeitpunkt vorhanden sind und eintreten und unerwünschte (Neben-) Effekte ausgeschlossen werden.

Dazu gibt es eine umfangreiche Anzahl verschiedener Verfahren, die in den meisten Fällen auf der (naiven) Mengenlehre und der Prädikatenlogik beruhen.

Grundlegend – und deswegen gerne in der akademischen Ausbildung eingesetzt - ist der hoarsche Kalkül.

„Er erlaubt die Spezifikation und Verifikation von Programmen in imperativen Programmiersprachen des ALGOL-Typs (ALGOL, Pascal, MODULA-2, ADA). Dazu wird die Semantik von Konstrukten dieser Sprachen axiomatisch durch Verifikationsregeln definiert. Die Verifikation von Programmen [...] erfolgt im Hoare-Kalkül durch Korrektheitsbeweise [...] (und) verbindet so mathematische Logik und Programmiersprachen [...].“ [8]

Auf dem Hoare-Kalkül bauen Verifikationssysteme wie „Z“ (sprich Set) und „B“ auf, die zu Industriestandards geworden sind.
Verschiedene Problemfelder (Programme sind sequentiell, parallel, real-time, etc.) werden von konkurrierenden Ansätzen abgedeckt, die teilweise Überschneidungen aufweisen. Da real-world Programme häufig eine Mischung aus diesen Problemfeldern sind, fallen eine Auswahl der geeigneten Methoden und eine empirische Vergleichbarkeit selten leicht, oft ist eine Mischung von Methoden erforderlich.

Entscheidend für eine Beurteilung und solide Anwendung ist aber die Kenntnis der grundlegenden Elemente: Mengenlehre und Prädikatenlogik. Diese waren auch Teil der Methoden, die im in [1] beschriebenen Experiment eingesetzt wurden.

Eine gute Übersicht über das Feld der formalen Methoden bieten die Linklisten unter [4] und [5].

2. Zusammenfassungen der Papers

2.1 Formal Methods Application: An Empirical Tale of Software Development

Als Teil einer Studie über die Integration formaler Methoden in die Lehrpläne für Softwaretechnik wurden die Fähigkeiten von Studenten gemessen und verglichen, inwieweit der Einsatz formaler Analyse deren 'Lösungsfähigkeit komplexer Probleme' (complex problem solving skills) verbessert.

Unterschieden werden hierbei eine 'control group' und eine 'formal methods group', deren Mitglieder gemeinsam eine 'Objekt-orientierte Design (OOD) Vorlesung' besuchten, in der auch das Experiment stattfand. Der Unterschied zwischen den Mitgliedern der beiden Gruppen ist, daß die 'formal methods group'-Studenten zusätzlich zum regulären Lehrplan über zwei Semester Veranstaltungen zu Programmspezifikation und -Ableitung und axiomatischer Semantik abstrakter Datentypen absolvierten.

Als ein Projekt innerhalb der OOD Veranstaltung wurde beiden Gruppen die Entwicklung eines Fahrstuhlsystems zugewiesen (das war eine Aufgabe eines IEEE Workshops von 1987 zu Softwareentwicklung und -Design).

Zusammengefasst die wesentlichen Daten zum Experiment-Aufbau:

1. 20 Gruppen mit durchschnittlich zwei Personen, davon sechs 'formal methods group' Teams und 13 'control group' Teams.

2. Die Mitglieder der 'control group' waren zufällig ausgewählte Studenten mit Informatik-Schwerpunkt im dritten Studienjahr.

3. Die Mitglieder der 'formal methods group' waren ebenfalls Studenten mit Informatik-Schwerpunkt im dritten Studienjahr, die freiwillig (self-selected) den zusätzlichen Lehrplan für formale Methoden absolvierten.
   Nach Aussage der Autoren stellt die Selbstauswahl keinen Nachteil dar sondern war eine Notwendigkeit, da die zufällige Erstellung einer „kleinen“ Gruppe keine Ausgewogenheiten bzgl. einer zufälligen „großen“ Gruppe bildet.
   [Meine Anmerkung: Ein Beispiel zum besseren Verständnis: Eine Gruppe A mit 100 zufälligen Integers und eine Gruppe B mit 5 zufälligen Integers produzieren keine Ausgewogenheit (Gleichverteilung) im Hinblick auf bspw. die Teilbarkeit durch 5.]

4. Beide Gruppen besuchten ansonsten die selben Veranstaltungen

5. Nach Aussage der Autoren waren beide Gruppen gleich in Bezug auf Fähigkeit und Wissensstand (mit Ausnahme der formalen Methoden). Dies wurde durch Tests verglichen.

6. Abgegeben werden sollten Source Code und eine lauffähige Executable.

7. Optionale Abgabe eines UML-Diagramms zum Design der jew. Lösung.

8. Kriterien zur Güte der entwickelten Software waren:
   a) Executable erfüllt sechs Testfälle (funktionale Korrektheit)
   b) Knappheit des Codes
   c) Komplexität des Codes (Schleifen, Fallunterscheidungen, Verschachtelungen)

9. Die 'formal methods group' hatte eine formale Spezifikation anzufertigen.

Zusammengefasst die wesentlichen Ergebnisse:

1. Das Design der 'control group' konnte nur teilweise aus dem Source Code erfasst werden, da keines der 13 Teams ein UML-Diagramm abgab.

2. Das Design der 'formal methods group' teilt sich in zwei Bereiche:
   a) Drei der sechs Teams gaben ein UML-Diagramm ab
   b) Alle sechs Teams gaben Spezifikation ab

3. Die Güte der abgegebenen Spezifikation variiert zwischen umfassender Beachtung vieler Methoden bis hinunter zu alleiniger Behandlung grundlegender Abschnitte, wobei auch ersichtlich wird, daß der Gebrauch der formalen Methoden oft ungeübt und wenig optimal ist.

4. Ein Vergleich zeigt, daß die meisten Teams beider Gruppen schlechtes Design wählten (etwa Koppelung von Funktion und GUI), die Lösungen uneinheitlich sind und kaum auf Effizienz geachtet wurde.

5. Funktionale Korrektheit erfüllten von der 'control group' fünf der elf abgegebenen Executables (45,5%) und sechs von sechs der 'formal methods group' (100%).

6. Ein Vergleich des Source Codes zeigte, daß die meisten Teams beider Gruppen in Bezug auf Knappheit und Komplexität schlechten bis mittelmäßigen Code schrieben, wobei der Code von Zweidrittel der 'formal methods group'-Teams im Schnitt etwas besser war.
   Dokumentation lag bei der 'control group' kaum vorhanden, teilweise wurde der Code durch naive Algorithmen verkompliziert.

7. An Bereichen, in denen der Einsatz formaler Methoden nicht verlangt war (etwa GUI), schrieben die 'formal methods group'-Studenten ähnlich schlechten Code wie die Mitglieder der Kontrollgruppe.

Die Autoren halten Faktoren wie höhere Intelligenz oder mehr Programmiererfahrung bei der 'formal methods group' für unwahrscheinlich, da dies aus standardisierten Tests, denen die Studenten unterzogen wurden, nicht hervorgehe. Sie schließen, daß allein der Einsatz formaler Methoden eine erhöhte Fähigkeit im Lösen komplexer Probleme gab.

Im Anschluß daran kamen vier Studenten der 'formal methods group' zusammen und erstellten eine vollständige formale Lösung inkl. einer Verifikation der relevanten Algorithmen (verification team). Sie schrieben einen formalen Zwischencode und leiteten daraus direkt die Implementierung ab. Das Design war durch rechtzeitige Entdeckung von Fehlern durchdacht, der Codestil hervorragend, die Implementierung korrekt.

Es wurden sogar Lücken in der verbalen Anforderung für das Fahrstuhlsystem entdeckt. Der direkt abgeleitete Code war knapper als in den Lösungen der anderen beiden Gruppen.

Die Autoren fassen zusammen:

1. Nur das 'verification team' entdeckte Lücken in der verbalen Anforderungsbeschreibung und zeigt damit einen Vorteil der (vollständig angewandten) formalen Methoden auf.

2. 100% der 'formal method group'-Implementierungen waren korrekt, dagegen nur 45,5% der 'control group'-Implementierungen.

3. Die Komplexität des Codes war bei der 'formal method group' insignifikant geringer als bei der 'control group'.

4. Die Knappheit des Codes war bei beiden Gruppen gleich. Signifikant knapperen Code erstellte das 'verification team' durch direkte Ableitung aus formalem Zwischencode.

Die Schlußfolgerungen:

1. Die Hypothese, daß durch den Einsatz formaler Methoden die 'formal method group' besseren Code produziere als die 'control group', sehen die Autoren als bestätigt an.

2. Die demonstriert erhöhte Fähigkeit zur Softwareentwicklung stützt die Vermutung, daß der Einsatz formaler Methoden die Lösungsfähigkeit komplexer Probleme verstärkt hat.

3. Das bewahrheitet die allgemeine Vermutung von Vertretern formaler Methoden, daß der Einsatz formaler Analyse während der Entwicklung bessere Programme produziert.

2.2 Comments on “Formal Methods Application: An Empirical Tale of Software Development“

Interessiert an der empirischen Überprüfung des Nutzens formaler Methoden führen die Autoren dieses Papers sechs Kritikpunkte an Aufbau, Durchführung und Bewertung von [1] an.

Sie kritisieren u.a., daß der Leser auf eventuelle Schwächen nicht hingewiesen wurde:

Zitat: "Unfortunately, the paper contains several subtle problems. The reade unfamiliar with the basic principles of experimental psychology may easily miss them and interpret the results incorrectly."

Die Autoren zeigen zudem auf, daß ein weiteres Paper zu [1] (Inroad-Paper [6]) besteht, das genauere Angaben zum Experiment macht (u.a. zu den Vergleichstests, die die Gleichheit der Gruppenmitglieder in Bezug auf Fähigkeit und Kenntnisse dokumentieren sollen) und in [1] in den Referenzen vorkommt, aber nicht entsprechend hervorgehoben wurde. So führen sie Hintergrundinformationen an, die aus [1] nicht ersichtlich sind.

Zusammengefasst die angeführten Schwächen im Experiment:

(1) Design des Experiments

Die hauptsächliche Schwachstelle sehen die Autoren in der Tatsache, daß die Studenten sich freiwillig einer Gruppe zuordnen konnten. Dadurch, daß die 'formal methods group' nicht aus zufällig ausgewählten Studenten bestand, können zur angeführten Hypothese konkurrierende Hypothesen nicht ausgeschlossen werden. Zumindest hätte verdeutlicht werden sollen, daß es sich bei [1] um ein Quasi-Experiment handelt (das ist ein Experiment, das nicht die Anforderungen an ein ordnungsgemäßes Experiment erfüllt und daraus gezogene Schlüsse unbrauchbar macht solange konkurrierende Hypothesen nicht ausgschlossen werden können), damit der Leser entsprechend gewarnt ist vor übermäßiger Gewichtung eventueller Schlüsse.

1. Unterschiedliche Motivation: Die Studenten der 'formal methods group' waren recht sicher höher motiviert und an formalen Methoden interessiert, immerhin haben sie über zwei Semester zusätzliche Kurse absolviert. Wie aus dem Inroad-Paper [6] hervorgeht, wurde der Zeitrahmen öfter überschritten und die hohe Motivation der Studenten betont.

2. Unterschiedliche Konfrontration der Studenten mit Anleitungen und Übungen: Die Studenten der 'formal methods group' hatten einen umfangreicheren Lehrplan und mehr Praxis und vermutlich eine intensivere Beschäftigung mit programmier-bezogenen Themen.

3. Unterschiedliche Lernstile: In [1] wurden die Studenten der 'formal methods group' einem Lernstil-Test unterzogen und als kooperativ und ergebnisorientiert klassifiziert. Weil die Studenten der 'control group' nicht einem ebensolchen Test unterzogen wurden, ist die konkurrierende Hypothese nicht auszuschließen, daß die 'formal methods group'-Studenten bessere Lerner sind oder härter arbeiten.

4. Unterschiedliche Begabungen: Die Studenten konnten freiwillig zwischen den (anspruchsvolleren) formalen oder informalen Kursen wählen. Trotz der in [6] angeführten Vergleichstests ist nicht auszuschließen, daß die Mitglieder der 'formal methods group' talentierter als die 'control group'-Mitglieder sind.

Ein in [6] angeführter Test scheint sogar ein Hinweis auf die höhere Talentiertheit zu bieten: Nach dem Projekt wurden den Studenten zufällige Analytik-Probleme eines Standardtests vorgelegt, den die 'formal methods group'-Studenten gegenüber der anderen Gruppe 36-prozentig korrekter abschlossen, was aber vermutlich nicht auf die Beschäftigung mit formalen Methoden (allein) zurückzuführen sein kann.

Da nicht auszuschließen ist, daß der Erfolg der 'formal methods group'-Studenten auf eigenen Qualitäten (Fähigkeit, Interesse) und nicht zwingend der Anwendung formaler Methoden allein basiert, schlagen die Autoren vor, um die Ausgewogenheit der Gruppen zu gewährleisten, alle für das Projekt heranzuziehenden Studenten in formalen Methoden zu unterrichten und dann zufällig den Gruppen zuzuordnen, die entweder eine formale oder eine informelle Lösung erstellen.

(2) Hawthorne Effekt / Novelty Effekt

Versuchspersonen, die wissen oder ahnen, daß sie an einem Experiment teilnehmen, verhalten sich anders als Personen, die nicht von der Beobachtung wissen. Der Hawthorne Effekt beschreibt, daß Versuchspersonen in diesem Fall meist versuchen, den Versuchsleiter auf ein erwartetes (oder geschätzes) Resultat hin zufriedenzustellen. Es ist also nicht auszuschließen, daß die 'formal method group'-Studenten sich besondere Mühe gegeben haben, um ein möglichst gutes Ergebnis zu erzielen.

Eine Möglichkeit, diesen Effekt abzuschwächen oder gar effektiv zu verhindern besteht darin, ausgewogene Gruppen zu haben, deren Mitglieder zufällig der einen oder anderen zugewiesen wurden. Um Interaktion mit der Versuchsleiter zu unterbinden könnte der Auswahlprozeß automatisiert werden. Um die Versuchspersonen über die Versuchsziele im Unklaren zu lassen kann man ihnen sogar absichtlich falsche Ziele vorgeben (auf die sie sich dann konzentrieren).

Eine Befragung der Versuchspersonen nach dem Test könnte Aufschluß über deren Erwartungen geben und somit einen Hinweis liefern, inwieweit gewisse Vorstellungen ihre Handlungen bestimmt haben.

Der Novelty Effekt beschreibt, daß sich Versuchspersonen anders verhalten, wenn sie mit einer neuen Herausforderung konfrontiert werden. Die Autoren führen diesen Punkt ohne weitere Bearbeitung an.

(3) Mangelnde Kontrolle

Kontrollgruppen dienen der Vergleichbarkeit. Nun führen die Autoren an, daß wie in [1] beschrieben von der 'control group' keine Aufzeichnungen über das Design verfügbar waren (keine UML Diagramme). Daraus entstehen zwei konkurrierende Hypothesen:

1. Die 'control group'-Mitglieder haben wenig bzw. keine Problemanalyse durchgeführt sondern sofort codiert, was den auffällig schlechteren Code miterklären könnte. Möglicherweise waren sie weniger motiviert als die 'formal methods group'-Studenten und haben sich um Fehler nicht allzu sehr gekümmert bzw. auffällige Designfehler mit einem workaround versehen.

2. Alternativ bestünde die Hypothese, daß die 'control group'-Mitglieder tatsächlich formale, informelle oder keine Analyse oder eine Mischung aus alledem gemacht haben.

Zusammengenommen ist über das Verhalten der 'control group' zu wenig bekannt, als daß sie für Vergleiche tauglich ist: Eine Gruppe, die formale Methoden benutzt, wird mit einer Gruppe verglichen, deren Problemlösungsstrategie gänzlich unbekannt ist.

In kontrollierten Experimenten muß das Verhalten aller Gruppen sorgsam überwacht werden. Im vorliegenden Experiment wäre es hilfreich gewesen, Design-Dokumentationen vor der Codierung einzureichen. Desweiteren würde der Einsatz einer einheitlichen Programmierumgebung mit Aufzeichnungsmöglichkeiten über Umfang und Dauer der Arbeit eine bessere Kontrolle und somit Vergleichbarkeit ermöglichen.

(4) Analyse der Resultate

Die Vorbereitung und Durchführung eines Experiments ist stark abhängig von den aufgestellten Hypothesen, also den zu untersuchenden Fragestellungen. Üblicherweise werden dazu überprüfbare Null-Hypothesen aufgestellt, die im Laufe der Arbeit entweder verworfen oder nicht verworfen werden. Die Autoren entwerfen zwei beispielhafte Null-Hypothesen, die jeweils eine unterschiedliche Herangehensweise erforderlich machten. Einen Abschnitt über auf verschiedene Probleme angemessene Metriken und Signifikanztests führe ich aus in Kapitel 3 zu nennenden Gründen hier nicht weiter aus.

(5) Ähnliche Arbeiten

Die Autoren äußern ihre Verwunderung, daß themenähnliche Arbeiten nicht genannt werden und weisen auf eine Fallstudie (kein Experiment) aus der industriellen Praxis hin, in der die Anwendung formaler Methoden in einer umfangreichen sicherheitskritischen Anwendung mit 200.000 LOC untersucht wird und die zu einem gegenteiligen Ergebnis von [1] kommt. Zumindest eine Erklärung oder Stellungnahme zu dieser Diskrepanz halten sie für wünschenswert.

(6) Externe Validierung

Die Überprüfung und Bewertung durch Dritte halten die Autoren für eine unablässige Pflicht. In [1] ist eine Diskussion der Ergebnisse nicht zu erkennen, das Inroad-Paper [6] hingegen zeigt einigermaßen auf, was tatsächlich geschah. Und daraus ersichtliche Versäumnisse im Experiment hätten erörtert werden müssen. Es ist ein Fehler, Ergebnisse schönzureden in der Hoffnung, dadurch eine bessere Akzeptanz der Arbeit zu erhalten - das Gegenteil ist der Fall. Denn kundige Leser wissen, daß Experimente nie im idealen Raume stattfinden und deshalb nicht perfekt sind. Werden mögliche Einflüsse auf die Ergebnisse ausgelassen, so wirft das ein schlechtes Bild auf die Arbeit und verhindert mögliche wichtige Auseinandersetzungen damit.

Schlußfolgerungen

Die vielen genannten Probleme im Experiment in [1] erfordern eine komplette Neustrukturierung, wobei die Anerkennung für die Durchführung eines Quasi-Experimentes nicht versagt werden soll.

2.3 Response to “Comments on 'Formal Methods Application: An Empirical Tale of Software Development'“

Achtung: Dieses ist keine Zusammenfassung. Hier nehme ich schon eine Beurteilung vor.

In diesem Paper verteidigen die Autoren ihre Darstellung von [1] und ich werde leider den Eindruck nicht los, daß man sich „ertappt“ fühlt: Es mag sein, daß sie - wie sie schreiben - etwas anderes meinten als von den Autoren von [2] interpretiert, aber in der Tat sind die Argumente in [2] stichhaltig und nicht mit verbalen Seitenhieben und markiger Sprache zu verdecken.

So kann bspw. der Versuch, den Hawthorne Effekt als widerlegt zu präsentieren, obwohl selbst die zitierte Quelle dazu nur vor einer allgemeingültigen Auslegung warnt, nicht überzeugen. So wirft im Gegenteil eine Recherche via Internet ein differenzierteres Licht auf den Streit um den Hawthorne Effekt: Ursprünglich aus der Industriepsychologie kommend sind die Vermutungen nicht allgemeingültig, aber durchaus anwendbar auf Bereiche, in denen ein enges Verhältnis zwischen Versuchsleitung und -Personen besteht [7]. So ist das gerade auch in der Erziehung/Ausbildung der Fall und genau dort hat das Experiment stattgefunden.

Der Hawthorne Effekt ist also etwas, das potentiell auftreten kann und nicht etwas, das sichere Schlüsse auf ein bestimmtes Verhalten ermöglicht. Da die Gefahr latent bei einem Versuchsaufbau vorhanden ist, aber nicht zwingend auftreten muß, ist darauf zu achten, den Hawthorne Effekt zu vermeiden.

Eine Zusammenfassung dieses Papers würde keine brauchbaren Informationen liefern, zumal das Mißtrauen in die Aufbereitung des Experiments gerechtfertig ist und die Autoren mit zweifelhaften Mitteln die Kritik als überzogen und ungerechtfertigt zurückzuweisen versuchen.

3. Beurteilung der Papers und Schlußfolgerungen

3.1 Anmerkungen zu wissenschaftlichen Standards

Ein grundlegendes Kritierium wissenschaftlicher Methoden ist die Vertrauenswürdigkeit der durchgeführten und präsentierten Arbeit. Das Prinzip des Aufeinanderaufbauens verlangt als Mindestmaß „Ehrlichkeit".
Neben anderen Gründen sollte dieser sehr eingängig ein:

„[...] it is impossible for every scientist to independently do every experiment to confirm every theory. Because life is short, scientists have to trust other scientists. So a scientist who claims to have done an experiment and obtained certain results will usually be believed, and most people will not bother to repeat the experiment." [9]

Um die Vertrauenswürdigkeit nicht mit Beliebigkeiten zu verwässern, existieren Standards, auf die sich die scientific community einigt und die solange eingehalten werden, wie sie brauchbar sind.
Dazu zählen gewisse Grundsätze (etwa der wissenschaftliche Skeptizismus oder auch Ehrlichkeit) als auch Techniken (wie in unserem Fall konkrete Methoden für empirische Experimente).

Gegen Techniken (aus Unwissenheit) zu verstossen mag problematisch sein, solange man ehrlich damit umgeht steht die Reputation (ein wichtiges Merkmal der Vertrauenswürdigkeit) aber nicht in Gefahr. Natürlich nur, solange man kein „Dauertäter" ist.

Verschliesst man sich der Kritik und reagiert trotzig und unsachlich, kann das zu einem Problem werden, denn es lässt möglicherweise Zweifel an der Gründlichkeit auch bei anderen Arbeiten aufkommen.

3.2 Beurteilung des Experiments

Das (Quasi-) Experiment als solches ist sicher interessant und es gibt Hinweise, in welche Richtung man weiter forschen kann.

Hätten sich die Autoren von [1] nur auf das Ausgangsthema konzentriert, nämlich um die Integration formaler Methoden in die Lehrpläne für Softwaretechnik, so wäre der Schluß legitim, daß formale Methoden sicher eine Bereicherung darstellen und möglicherweise auch die Fähigkeit der Studenten erhöhen, mit komplexen Problemen zu arbeiten.

Leider aber sprangen die Autoren zu weit vor und behaupteten, den Beweis für die Überlegenheit der formalen Methoden gegenüber, nun, Nichts, geliefert zu haben.
Da sie keine valide Vergleichsgruppe bieten konnten, haben sie faktisch gegen nichts verglichen und ihr Schluß ist somit unzulässig und unbrauchbar.

Grundlegende Fragen hätten u.a. mit dem Experiment beantwortet werden müssen, etwa, ob eine (in Motivation, Fähigkeiten etc. gleichwertige) Vergleichsgruppe in der gleichen Zeit und mit dem gleichen Aufwand ein gleiches, besseres oder schlechteres Ergebnis erzielt hätte.

Letztlich zeigt es aber nur, daß eine Gruppe motivierter Programmierer mit Anwendung formaler Methoden gute Programme schreiben kann. Und das ist der Stand des Wissens, der schon vor dem Experiment bestanden hat.

Ich kann nicht sagen, welches die Motivation der Autoren war, derartige Schlüsse zu ziehen und zu veröffentlichen. Eventuell war eine gewisse Erwartungshaltung, ein Wunschdenken oder gar eine Überbewertung ihrer Arbeit vorhanden.

Die in [2] geäußerte Warnung, daß in Design von Experimenten ungeübte Leser falsche Schlüsse ziehen könnten, muß ich zumindest in meinem Fall größtenteils bestätigen. Zwar störte ich mich an der geringen Zahl von Testfällen und betrachtete deshalb die Signifikanzmessungen daran als unbrauchbar, hätte aber durchaus gewisse Schlüsse der Autoren akzeptiert.

Zusammenfassend kann ich behaupten, daß die drei Papers einen wichtigen Lehreffekt hatten, zumal ich zunächst etwas anderes erwartete und dann das Experiment unter einem neuen Lichte betrachten mußte.

4. Richtlinien zur Durchführung empirischer Experimente

4.1 Richtlinien für das Design von Experimenten

1. Aufstellung klarer (Null-) Hypothesen
   Der Aufbau und somit der weitere Verlauf des Experiments richten sich nach der Fragestellung, also: Worauf wird getestet? Diese Hypothesen sollten dann am Ende auch nur behandelt werden.

2. Valide Versuchsgruppen erstellen
   Bei der Zusammenstellung von Versuchsgruppen ist es wichtig, auf eine Ausgewogenheit in den relevanten Merkmalen zu achten.
   Auswahl aus einer in den Kriterien homogenen Gruppe und Zuordnung in verschiedene Kontrollgruppen sollte eine Gleichverteilung ermöglichen.

3. Vermeidung des Hawthorne Effekts
   Darauf achten, daß die Versuchspersonen das Ziel des Experiments nicht kennen oder ahnen. Zwei wirksame Möglichkeiten sind die Trennung (keine Interaktion) von Versuchsleitung und -Personen und die „offensichtliche“ Konzentration auf andere Aspekte.

4.2 Richtlinien für die Durchführung von Experimenten

1. Umfassende Überwachung der Versuchsgruppen
   Alle Gruppen müssen in den selben Punkten gleichwertig überwacht werden. Erst das ergibt eine Vergleichbarkeit und somit die Bestimmbarkeit des Wertes von Daten.

4.3 Richtlinien für die Präsentation von Experimenten

1. Einbeziehung / Beachtung bestehender Arbeiten
   Themennahe bzw. -gleiche Ergebnisse, die von anderer Seite produziert wurden, sind – wo gegeben – zu beachten und mit den eigenen Ergebnissen in Einklang zu bringen bzw. mögliche Unterschiede anzusprechen.

2. Diskussion durch Dritte
   Soweit möglich sollten die eigenen Ergebnisse von Dritten bestätigt bzw. diskutiert werden. Das Aufdecken eventueller Versäumnisse und der ehrliche Umgang damit ist wichtig - auch im Hinblick auf Akzeptanz in der scientific community.

3. Korrekte und ausführliche Aufbereitung
   Die Aufbereitung der Experimente sollte umfassend und klar sein. Selbst korrekt durchgeführte Experimente werden unbrauchbar in dem Moment, wo sie von Dritten nicht angemessen beurteilt werden können und somit eher Ablehnung als Interesse erzeugen.

5) Literaturverzeichnis

5.1 Literaturverzeichnis

[1] Ann E. Kelley Sobel, Michael R. Clarkson: Formal Methods Application: An Empirical Tale of Software Development. IEEE Trans. Softw. Eng., volume 28, number 3, 2002, pages 308--320

[2] Daniel M. Berry, Walter F. Tichy: Comments on "Formal Methods Application: An Empirical Tale of Software Development, IEEE Trans. Softw. Eng., volume 29, number 6, 2003, pages 567--571

[3] Ann E. Kelley Sobel, Michael R. Clarkson: Response to "Comments on 'Formal Methods Application: An Empirical Tale of Software Development', IEEE Trans. Softw. Eng., volume 29, number 6, 2003, pages 572--575

[4] Formal Methods Europe
http://www.fmeurope.org/

[5] The World Wide Web Virtual Library: Formal Methods [Linkliste]
http://www.afm.sbu.ac.uk/

[6] Ann E. Kelly Sobel, "Empirical Results of a Software Engineering Curriculum Incorporating Formal Methods," ACM Inroads, vol 32, no. 1, pp. 157-161, Mar. 2000

[7] The Hawthorne effect: a note
http://www.psy.gla.ac.uk/~steve/hawth.html

[8] Bernhard Hohlfeld / Werner Struckmann, „Einführung in die Programmverifikation“, BI-Wissenschaftsverlag, 1992

[9] sci.skeptic FAQ
http://www.faqs.org/faqs/skeptic-faq/