Vererbungstiefe in der Wartung

Ausarbeitung im Rahmen des Seminars
"Empirische Forschungsmethoden in der Softwaretechnik"

FU Berlin

Wintersemester 03/04

Georg Sisow 20.03.2004

Seminarbetreuer: Stephan Salinger, Sebastian Jekutsch

Inhaltsverzeichnis

1.Einleitung

Nicht alle Themen in der Softwaretechnik lassen sich überhaupt mit Experimenten erforschen. Zu dehnen für die es diese Methode jedoch besonders gut eignet zählen solche die an dem Programmverstehen forschen. In diesem Dokument soll die Anwendung von kontrollierten Experimenten auf die Frage nach dem Einfluss der Vererbungstiefe auf die Wartung von Software betrachtet werden. Es werden hauptsächlich zwei umfassende Experimente zu diesem Thema behandelt, die zu unterschiedlichen Erkenntnissen gelangt sind und anschließend die dazu ermittelten Erklärungen diskutiert. Die Erwähnung zwei weiterer eher kleinerer Versuche wird die Betrachtung ergänzen.

Mit dem Aufkommen der objektorientierter Programmierung verfestigten sich auch endliche positive Vorurteile. Diese, zwar teilweise auf sehr schlüssigen logischen Überlegungen begründet, wurden aber kaum mit aus Beobachtungen stammenden Daten untermauert.

1.1 Subjektive Bewertungen

Objekt orientierte Software bietet mit Vererbung leichte und sicherere Wiederverwendung von Code, weniger Redundanz, reduziertes Defektpotenzial, leichte Erweiterbarkeit in Zusammenhang mit Polymorphie, einfaches klares und flexibles Design und die natürlich begleitende Eleganz. Alles Gründe für Vorteile. Dabei ist schon allein die unsystematische Wiederverwendung ein oft unangenehme Folgen nach sich ziehende Programmierpraktik die mit Vererbung sauber gelöst wird.

Die genauso herleitbare Nachteile die mit der Vererbung kommen sind zusätzliche Komplexität (man muss nicht nur mit den technischen Mitteln vertraut sein sondern auch alle Möglichkeiten steht's im Bewusstsein halten), Wechselwirkungen (es ist nicht mehr alles so gradlinig wie bei konventioneller Programmierung), Abhängigkeiten (die Verflechtung von Codeteilen nimmt zu) und Zerstreuung (es liegt in der Natur der Sache das übernommene Funktionalität sich dann nicht an einem Ort befindet). Sind also auch Aspekte negativer Einflusse vorhanden, wird es interessant zu erforschen ob vielleicht ein Übergewicht auf eine oder andere Seite irgendwann eintritt.

1.2 Was diese Frage interessant macht

Viele Softwaresysteme verlassen heutzutage unausgereift und voll von Fehlern die Entwicklungsabteilung. Die Optimierung und eventuelle nötige Fehlerbeseitigung bleibt dann der Wartung überlassen. Dazu kommen die ständigen Änderungen um die Software an die wandelnden Umgebungsbedingungen anzupassen und sie nicht veraltern zu lassen. Sehr oft werden diese Arbeiten nicht von den ursprünglichen Programmierern des Produkts durchgeführt was zum Teil auch an der langen Lebensdauer des Systems liegt. Die Leute, die die Wartung an dem Produkt durchführen brauchen dann vor allem Verständnis des Softwareinneren. Es gilt als erwiesen, dass das Verstehen eines Programms ein wichtiger Aspekt für eine erfolgreiche Änderung oder Anpassung eines Programms gilt. Die verschiedenen Arten der Wartung sind Korrektur, Optimierung, Änderung und Erweiterung wovon die letzteren beiden am häufigsten durchgeführt werden. Ein ganz Praktischer Vorteil der sich von den Erkenntnissen versprochen werden könnte ist eventuelles Wissen zur Verringerung der Wartungszeit und damit Kostenreduzierung.

Gestartet wurde die Erforschung mit der Befragung unter Softwareentwicklern, die ergab dass etwa die Hälfte die Vererbungstiefe für einen Faktor halten wenn es um das Verständnis der zu wartenden Software geht. Auch wurde unter den Befragten ausgemacht das der Beginn der Schwierigkeiten in der Vererbungstiefe von 4 bis 6 erwartet wurde.

2.Experiment von Daly, Brooks, Miller, Roper und Wood

Ausgegangen wurde in diesem Experiment von dem Glauben das die Vererbungstiefe einen zumindest nicht unerheblichen Einfluss auf die Wartbarkeit von Software hat ohne jedoch sich vorher festzulegen ob es ein positiver oder negativer Effekt sein wird. Die Wartbarkeit wurde in diesem und dem nächsten Experiment in der für Änderungen benötigten Zeit gemessen und die Unterschiede in der Wartbarkeit in der Zeitdifferenz. Das Experiment stellt das Suchen und Erweitern in den Vordergrund.

2.1 Experimentaufbau

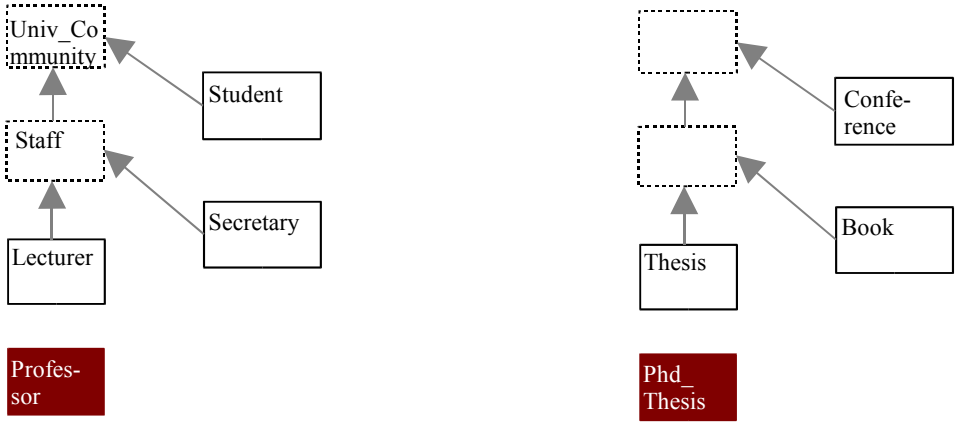
Da es bis dahin keine große Erforschung dieses Gebiets gab, musste man mit einer unteren Stufe der Empirischen Forschung vorgehen. Die bis hierhin angestellten Überlegungen wurden in Hypothesen umgewandelt um sie anschließend einem Hypothesentest zu unterziehen. Es wurden für den Vergleich von „flacher" mit vererbungsstrukturierter Software die Vererbungstiefen 3 und 5 ausgewählt. Die Vererbungstiefe 5 soll die falls vorhandenen Schwierigkeiten in der Region 4 bis 6 entdecken, während die Vererbungstiefe 3 ein Bindeglied zum Vergleich zu der „flachen" Software darstellt. Folgende Hypothesen wurden dabei aufgestellt:

> Die Verwendung von Vererbung der Tiefe 3 **beeinflusst nicht** die Wartbarkeit objektorientierter Software.
>
> Die Verwendung von Vererbung der Tiefe 3 **beeinflusst** die Wartbarkeit objektorientierter Software.

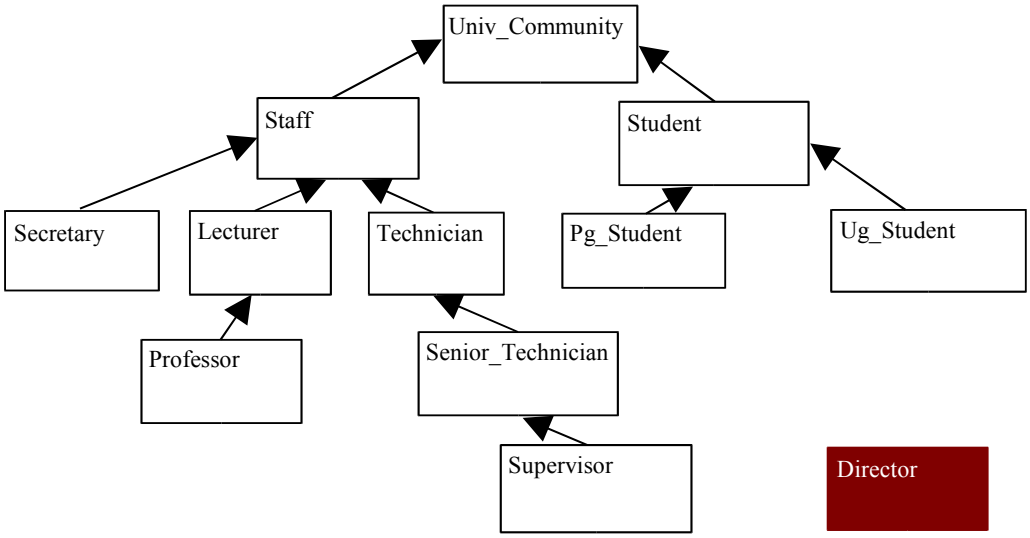
Die Verwendung von Vererbung der Tiefe 5 **beeinflusst nicht** die Wartbarkeit objektorientierter Software.

Die Benutzung einer Hierarchie von 5 Stufen Vererbungstiefe **beeinflusst** die Wartbarkeit der objektorientierten Software – Teilnehmer an der Vererbungsversion werden länger brauchen als die an der „flachen".

Die Richtung in der letzten Hypothese ist nur deshalb vorhanden weil die Umfragen eindeutig auf Probleme hinweisen. Das Experiment soll von der unabhängigen Variable Programmstruktur („flach" oder mit Vererbungsstruktur) reguliert werden. Die Auswirkungen auf die unabhängige Variable Zeit sollen dabei studiert werden. Um sicherzugehen das die Veränderungen nur von der gewünschten Variation der unabhängigen Variable herrühren mussten andere Variablen konstant gehalten werden. Die konstanten Variablen waren Aufgabe und Programm.
Als Programme kamen zwei Datenbanken zum Einsatz. Eine die verschiedene Mitarbeiter einer Universität (Universitätspersonal) erfassen kann und eine die das gleiche mit schriftlichen Arbeiten (Schriftliche Arbeiten) tun kann. Beide objektorintiert mit Vererbungstiefe 3 entworfen und in C++ implementiert.



Für die Vererbungstiefe 5 wurde eine kompliziertere und größere Version der Universitätspersonal Datenbank verwendet.



Die entsprechenden „flachen" Versionen der Programme wurden durch Codeeinfügen und Entfernen der Oberklassen erstellt. Die Größe der Programme betrug in Teil 1 360 bis 390 LOC, in

Teil 2 waren es ca. 800 bis 1100 LOC. Die Ähnlichkeit der Programme in Teil 1 trägt wesentlich zu Vergleichbarkeit bei so dass man beide Programme als eins sehen kann und dennoch nicht zwei Mal die gleiche Aufgabe ausgeführt wurde.
Die Aufgaben verlangten eine Erweiterung, eine Verbreiterung der Gegenstände die die Datenbank aufnehmen kann. Für Universitätspersonal Datenbank war ein Klasse Professor hinzuzufügen und für die Schriftliche Arbeiten Datenbank die Klasse Phd_Thesis.
Eine computerunterstützte Datenaufzeichnung lieferte neben der Hauptmessgröße Zeit auch die Backups der Zwischenlösungen, ein Skript über die Vorgehensweise und die letztendliche Lösung.

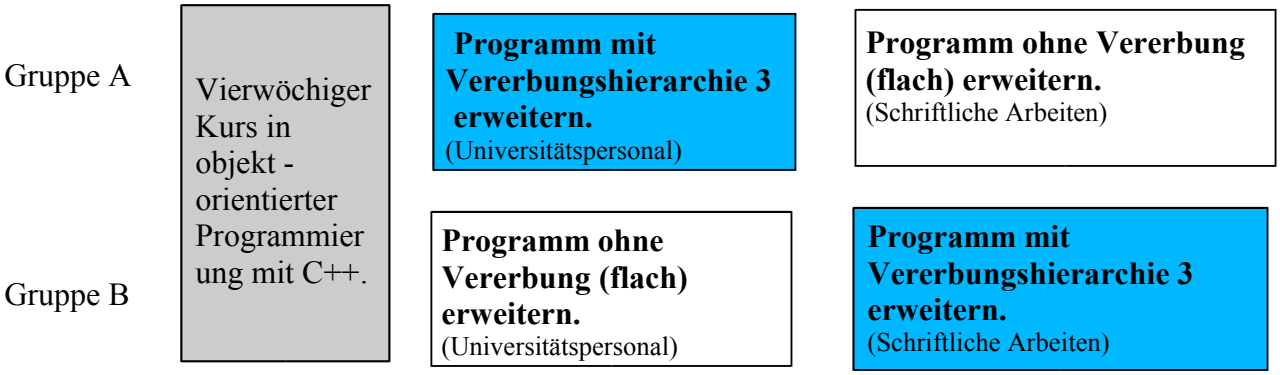
2.2 Teilnehmer

Da die Betrachtung der Leistung jeden einzelnen Teilnehmers wenig Sinn macht (individuelle Variation) wird die mittlere Leistung von Gruppen verwendet. Um die Ausbalancierung der Gruppen zu erhalten wurden alle Teilnehmer in Paare gruppiert so dass sich jedes Paar an Leistung gleicht. Damit wird ermöglicht eine Person mit sich selbst zu vergleichen, was die wertvollste Information liefert. Vergleiche zwischen unterschiedlichen Personen haben steht's den Nachteil dass der Unterschied in ihrer Leistung immer auch mit ihrer Unterschiedlichkeit zusammenhängt was schwer zu messen ist. Von einer derartigen Gruppierung ausgehend wurden von jeder Zweiergruppe jeweils eine Person in eine Gruppe A und eine Gruppe B eingeteilt.
Der erste Teil wurde von einem IT Umschulungskurs durchgeführt. Insgesamt kamen um die 30 Teilnehmer zu Teil 1. Für die Gruppierung dieser Teilnehmer wurden die Ergebnisse von zwei Tests verwendet, die das Wissen in objektorintierter Programmierung der Personen testeten.
Für den zweiten Teil dieses Experiments waren erfahrenere Teilnehmer angeworben worden. Diese setzten sich aus Absolventen und denen im Abschluss befindenden Studierenden zusammen. Auch diese ca. 30 Leute starke Gruppe wurden gruppiert, hier aber nach ihren Examensnoten.

2.3 Durchführung

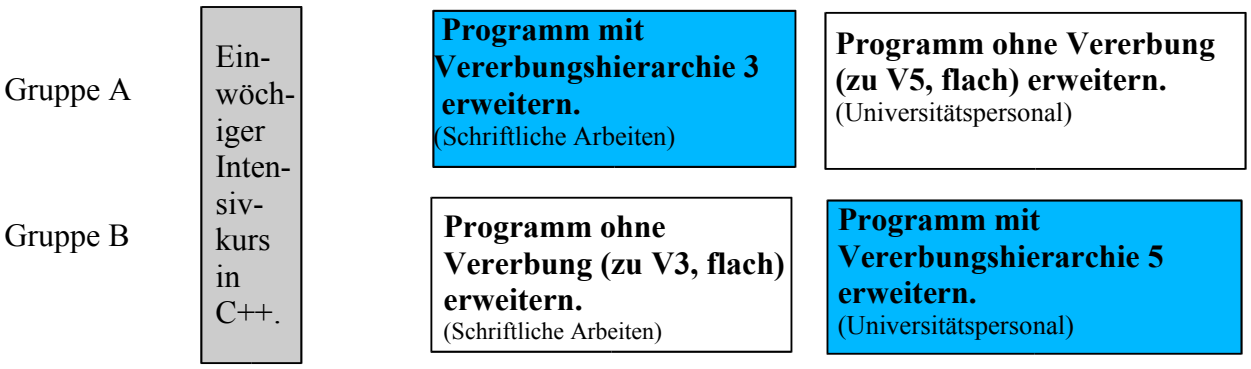
Um eventuell durch Aufgaben eingeführte Voreingenommenheit auszugleichen und auch möglichen durch individuelle Fähigkeiten eingeschleusten Effekten vorzubeugen, wurde ein gegenbalanzierter Entwurf verwendet. Dabei führte Gruppe A die Wartungsaufgaben an einer Software mit Vererbungsstruktur und Gruppe B die Wartungsaufgaben an einer „flachen" Software durch. Anschließend wurde mit vertauschten Strukturen und neuen Programmen noch mal eine Wartung durchgeführt. Die Aufgaben waren dabei von der Aufgabenmuster gesehen steht's die gleichen. Ein positiver Effekt dieses Entwurfs ist der Erhalt von mehr Datenpunkte aus der zur Verfügung stehenden Menge von Versuchspersonen. Der negative Nebeneffekt ist das dadurch eine zusätzliche unabhängige Variable dazukommt – die Reihenfolge.

Teil 1:

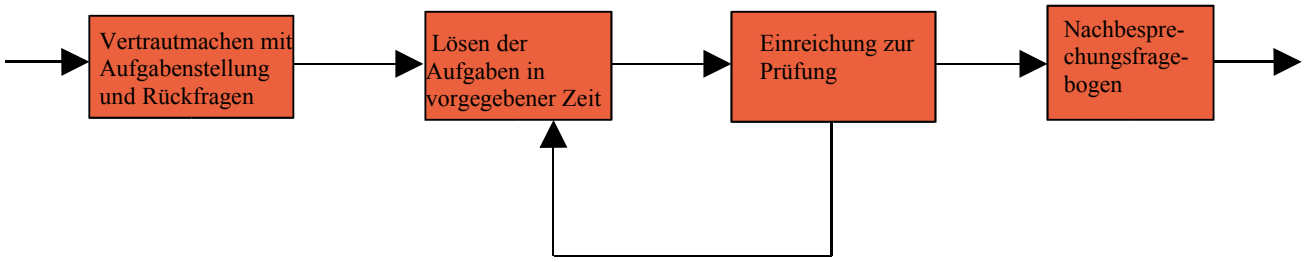


Dieser kleine Nebeneffekt wurde in diesem Fall zu einer Störvariable – dem Lerneffekt und sollte bei der Auswertung der Resultate noch mal zur Sprache kommen.

Teil 2:



Die Durchführung von Teil 2 fand vor dem Bekannt werden der Ergebnisse von Teil 1 statt um die innere Gültigkeit nicht durch eventuelle Effekte aus der Historie zu beeinflussen. Die Ausführung der Aufgaben folgte dem folgenden Muster:



Lösungen die nicht in der vorgeschriebenen Zeit erledigt werden konnten wurden bei der Zeitbetrachtung nicht berücksichtigt, die Bemühungen aber aufgenommen.

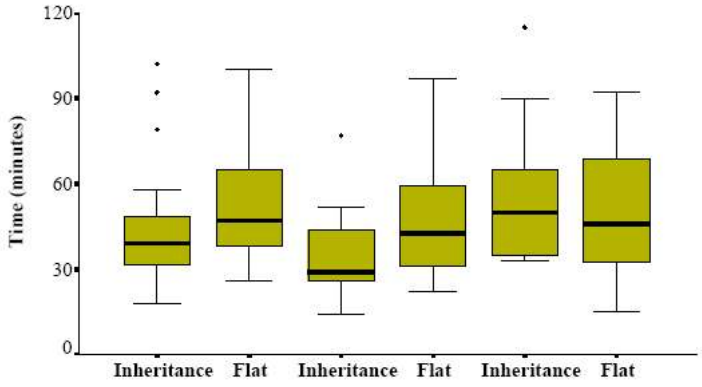
2.4 Analyse

Bedrohungen für die innere Gültigkeit kommen steht's durch unbeobachtete Variablen die die Resultate beeinflussen. *Auswahleffekte* wurden durch ausgeglichene Gruppen soweit wie möglich ausgeschlossen. *Lerneffekte* wurden in einem geringen Maße festgestellt. *Instrumentation* die von den Unterschieden in der verwendeten Software und Aufgaben kommen könnte wurde aufgrund der Ähnlichkeit beider kaum ermöglicht und auch in den Zeiten wurde kein Hinweis gefunden.

Tatsachen die die Verallgemeinerung behindern gab es in diesem Experiment recht viele so dass die äußere Gültigkeit nicht sehr groß ausfiel. Die *Teilnehmer* sind nicht repräsentativ. Die aus Teil 1 sind es erst recht nicht und die aus Teil 2 sind keine Professionellen. Allerdings kommen diese den wirklich eingesetzten Programmierern sehr nahe und was die Teilnehmer von Teil 1 rechtfertigen könnte ist das es nicht auf die absolute Leistung ankommt, sondern auf die Richtung wohin die Resultate tendieren. Die *Programme* waren nicht sehr groß was man als ein echtes Problem betrachten kann. Obwohl die Vererbungstiefe als repräsentativ betrachtet werden kann wurden hier einige Sachen wie Polymorphie nicht verwendet, was etwas an der Verallgemeinerbarkeit nagt. Es spricht aber auch etwas dafür, dass bei Komplexen Systemen der Effekt der Vererbungstiefe auf die Wartbarkeit sonst kaum herauszulesen wäre. Die Untersuchung der Wartbarkeit ist am besten bewertet mit dem Blick auf den ganzen Prozess und nicht nur auf die *Implementierungsphase*. Es könnte sein das die Vererbungstiefe Einflusse auf Entscheidungen in der Planung, was und wie verändert wird, ausübt. Aber so was ist schwer praktizierbar.

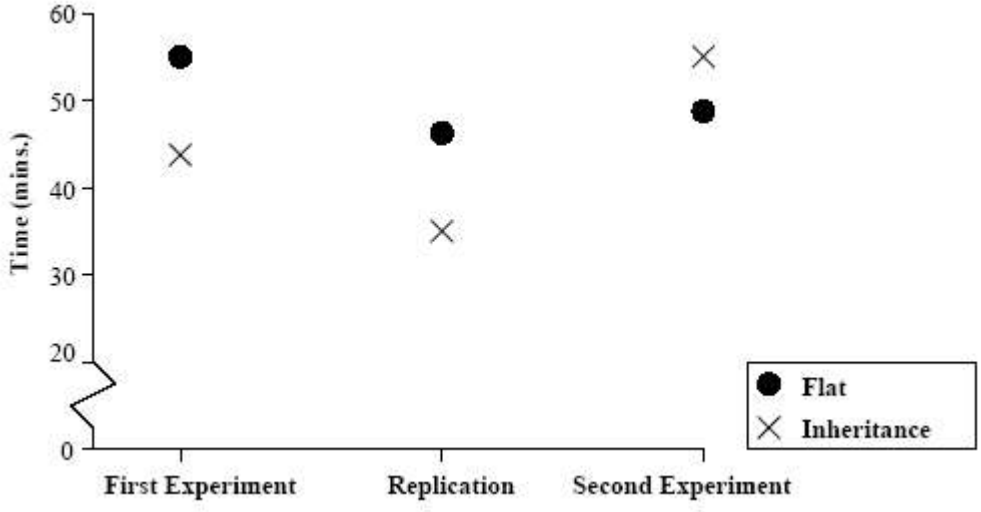
Zu Teil 1:
Nicht alle teilnehmenden Personen erledigten beide Aufgaben. Für die 20 Teilnehmer die es schafften beide Aufgaben erfolgreich zu meistern, lieferten statistische Tests ein signifikantes Resultat. Da 13 Teilnehmer von ihnen in der Softwareversion mit Vererbungsstruktur besser waren als in der flachen, 6 in der „flachen“ als in der mit Vererbungsstruktur und 1 in beiden gleichschnell wurde die Hypothese „Die Verwendung von Vererbung der Tiefe 3 **beeinflusst** die Wartbarkeit objektorientierter Software.“ angenommen und die gegenteilige verworfen. Das statistisch signifikante Resultat basierte auf paarweiser Beobachtung, so dass Teilnehmer mit nur einer erledigten Aufgabe nicht berücksichtigt wurden. Der hier festgestellte kleine Lernefekt war das im ersten Durchgang 11 (von 15) „flache“ Lösungen abgegeben wurden im zweiten Durchgang waren es 7 (von 16) „flache“ Lösungen. Die handvoll Personen die im ersten Durchgang ihre Aufgaben mit Vererbung gelöst haben sind somit etwas geübter in den Durchgang zwei zu Vererbungsversion gegangen. Dieser Lernefekt trat bei der erfahreneren Gruppe im Teil 2 nicht auf da eine so kleine Erfahrung nur im Anfängerstadium signifikant ist.

Zu Teil 2:
Die Resultate der Wiederholung waren in der Richtung wie die von Teil 1 auch wenn sie von der durchschnittlichen Zeit wegen der erfahreneren Programmierer schneller waren. Die Resultate der Vererbungstiefe 5 Version passten zu den in der Hypothese „Die Benutzung einer Hierarchie von 5 Stufen Vererbungstiefe **beeinflusst** die Wartbarkeit der objektorientierten Software – Teilnehmer an der Vererbungsversion werden länger brauchen als die an der flachen.“ genannten Richtung. Die Vergleiche zu der Wiederholung zeigten das während die Zeiten der „flachen“ Version nur minimal stiegen (obwohl das Programm sowohl viel größer als auch etwas komplexer war), stiegen die Zeiten der Version mit Vererbungsstruktur substantiell stark an. Ein Signifikanztest bestätigte aber die Unterschiede als nicht signifikant. Die Hypothese „Die Verwendung von Vererbung der Tiefe 5 **beeinflusst nicht** die Wartbarkeit objektorientierter Software.“ konnte deshalb nicht abgewiesen werden. Der Untersuchung des Vererbungstiefe 5 Programms im Vergleich zu seiner „flachen“ Version in Teil 2 mangelte es an Statistischer Kraft. Dennoch war ein drehen in der Richtung der mittleren Zeiten (jetzt „flach“ vor Vererbung) erkannt worden.



Das Boxplot Diagramm zeigt besonders gut die Ähnlichkeit in der Verteilung der Resultate von Teil 1 (ersten beiden Boxplots) und der Wiederholung (mittleren zwei Boxplots). Auch wird es erkennbar wie sich die Differenz zwischen „flach“ und Vererbung mit steigender Vererbungstiefe entwickelt. Die Möglichkeit zum Vergleich von den beiden Teilen ermöglicht die Vernachlässigung der Programmgröße. Es ist nicht von Bedeutung wie groß das Programm ist wenn für die Erledigung der Aufgabe nur ein gewisser Teil betrachtet werden muss. Es gibt auch nur einen sehr geringen Anstieg der Zeiten der „flachen“ Version der umfassenderen Universitätspersonal Datenbank (letzter Boxplot) im vergleich zu der „flachen“ Version von der Wiederholung wo ein 3-

mal kleineres Programm verwendet wurde. Ein Indiz dafür das während in der „flachen“ Version immer noch alle Codeteile an einem Platz stehen in der Vererbungsversion diese den Programmierer zu weiten Suchwegen zwangen.



Eine mögliche Interpretation dafür wurde in der Erklärung gefunden das Teil 1 und Wiederholung den Vorteil der Vererbung in der Änderbarkeit offen legte, während er in Teil 2 mit dem Vererbungstiefe 5 Programm durch das belastende Suchproblem verschüttet wurde welches durch die größere Anzahl der Klassen in der Vererbungshierarchie verursacht wurde.
Antworten aus den Nachbesprechungsfragebogen liefern weitere Anhaltspunkte die dafür sprechen. Einige beachtenswerte Angaben daraus waren zum einen das fast gleich viele Personen sowohl aus der „flachen“ als auch aus der Vererbungsstruktur Version der Software angaben eine optimale Lösung gewählt zu haben und zum anderen die Wahl einer Schablone bzw. Oberklasse bei der „flachen“ Version häufig unter 5 min während in der Vererbungsstruktur Version am häufigsten zwischen 5 und 10 min dauerte. Auch gab es in der Vererbungsstruktur Version sogar 3 Personen die an der falschen Klasse ansetzten was nicht gerade für die Leichtigkeit der Suche steht. Im Versuch mit der Vererbungstiefe 5 Version gab es keine so klar ersichtliche Ideallösung so dass es 2 Personen für die Klasse Lecturer, 9 Personen für die Klasse Staff und 4 Personen für die Klasse Supervisor als Oberklasse waren. Auch das wurde als Beleg für das anspruchsvolle Suchproblem angesehen. Andere Schwierigkeiten die in der Vererbungsstruktur Version essenziell häufiger auftraten als in der „flachen“ waren das Aufspüren der vererbten Funktionen. Die von der Vererbung geförderte Zerstreuung von zusammengehörenden Teilen führt dazu das Ausdrücke in den unteren Hierarchieebenen nur verstanden werden können wenn man den Vererbungsbaum auf und ab marschiert um zu sehen wo und wie die eigentliche Arbeit getan wird. Eine kleine Vererbungstiefe mag nicht viel ins Gewicht fallen, eine mehr und mehr steigende dagegen schon. Auch ist erwähnenswert das derartige Probleme wie Wahl der Oberklasse/Schablone oder Auffinden von Funktionsimplementierungen nur aus Teil 1 und Wiederholung berichtetet wurden. Das verschärfte Suchproblem der Oberklasse und das Auffinden von Implementierungen für die zu verstehenden Funktionen werden für die Verschlechterung der Wartungszeit ausgemacht.

Die Resultate der Studie suggerieren das wenn es unerheblich ist von welcher Klasse man erbt (was wohl kaum realistisch ist) und wenn der Aufwand für Auffinden der Implementierungen unwesentlich ist, die Vererbungsstruktur Vorteile in der Veränderung bietet. Andererseits wenn mit steigender Vererbungstiefe die Oberklassen in ihren Konzepten nicht beständig erweitert oder spezialisiert wurden (verstreute Implementierungen) und er Aufwand für Auffinden der Implementierungen nicht unwesentlich ist verliert sich der Vorteil der objektorientierten Software.

3.Experiment von Prechelt, Unger, Philippsen und Tichy

Die Autoren dieses Experiments vertrauten nicht so richtig auf die Ergebnisse des Daly u.a. . Es wurde beschlossen eine Replikation der Ergebnisse durchzuführen um nicht die als nachteilig angesehenen Eigenschaften des Daly Experiments zu wiederholen aber dennoch am gleichen Problem zu experimentieren. Erst durch eine Wiederholung werden üblicherweise in einem Experiment erlangten Erkenntnisse als gesichert angesehen zumal es zeigt das das Thema auch von anderen Forschern aufgenommen und weiter erforscht wird. Außerdem wird dadurch auch eine Verbreiterung der Erkenntnisse erreicht was auch hier der Fall ist. Das Experiment stellt das Verstehen in den Vordergrund, weniger die Änderung.

3.1 Experimentaufbau

Da es sich um eine Replikation des Experiments von Daly u.a. handelt, auch wenn's nur eine Replikation der Ergebnisse ist, ist der grundsätzliche Aufbau gleich dem des genannten Experiments. Das Wesentliche hier ist das es in an bestimmten Stellen verändert ist um einige beargwöhnte Eigenschaften zu beseitigen und die äußere Gültigkeit zu verbessern.
Was an dem Experiment von Daly u.a. zu beanstanden war, ist das den Teilnehmern kein Vererbungsdiagramm oder etwas anderes das die konzeptionelle Architektur aufzeigte übergeben wurde. Auch die verwendeten Programme verdienten Kritik für ihre Einfachheit in Größe und Struktur. Folgende Unterschiede kennzeichnen deshalb den Aufbau dieses Experiments:

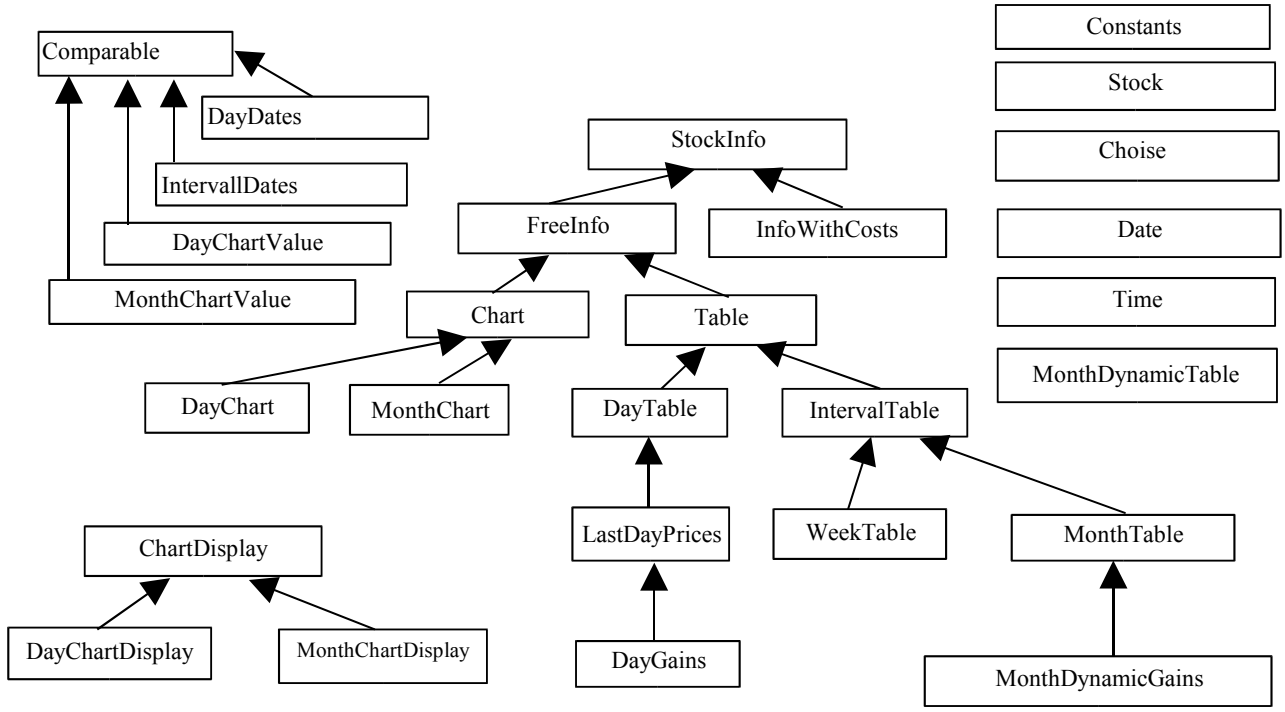
- Es wird ein einziges Programm für alle Durchgänge und Gruppen verwendet. Begründung: Mit nur einem Programm wird die Vergleichbarkeit viel besser und schärfer als in dem Daly Experiment da es jetzt auch der direkte Vergleich zwischen der Vererbungsstruktur 3 und 5 ermöglicht wird.
- Ein Klassendiagramm wird zur Verfügung gestellt wodurch das Betrachten der Vererbungsstruktur insgesamt erleichtert wird. Begründung: Auch die Dokumentation mit der die Architektur des Softwareprodukts sichtbar wird ist in jeder Software-Organisation vorhanden.
- Die Komplexität der verwendeten Programme stieg im Vergleich erheblich. Es gab neben der Größe vor allem mehr Beziehungen und nicht offensichtlicher Funktionalität. Begründung: Durch Verwendung von größerer und komplexerer Programme wurden zwar nicht die gleichen Bedingungen wie im realer Software-Organisation erreicht, da durften die Programme noch zig Fach größer sein, aber ein guter Schritt zu mehr äußerer Gültigkeit ist das schon denn die Programme im Experiment von Daly waren sehr einfach.
- Verwendete Aufgaben waren komplexer und vielseitiger. Vor allem wurde nicht ausschließlich auf Erweiterung gesetzt. Auch eine Aufgabe die die Änderung des Vorhandenen Codes ohne Erweiterung war dabei. Begründung: Die Aufgaben wurden der Realität angenähert denn in Wirklichkeit kommen verschiedenste Arten von Aufgaben vor. Anmerkung: Viele sind allerdings auch nicht sehr schwierig
- Über die Fertigstellung der Aufgabe entschied jeder Teilnehmer nach eigenem Ermessen. Es wurde keine Kontrollinstanz eingesetzt. Begründung: In der Praxis gibt es auch keinen allwissenden Überwacher. Einwand: In vielen Software-Organisationen gibt es aber eine Qualitätssicherung die diese Aufgabe übernimmt.

Die Begründung für all das ist die Annäherung an die wirkliche Softwarewartung in der sowohl größere als auch kompliziertere Programme vorkommen die nebenbei auch Polymorphie und anderes enthalten. Folgende Hypothesen wurden aufgestellt:

A: Programme mit größerer Vererbungstiefe brauchen mehr Zeit in der Wartung.

B: Programme mit größerer Vererbungstiefe sind Ursache schlechterer Qualität in der Wartung.

Das verwendete Programm Börse wurde in Java in der Ursprungsversion in einer Struktur mit Vererbungstiefe 5 geschrieben und dann wie oben erwähnt in andere Versionen umgewandelt.



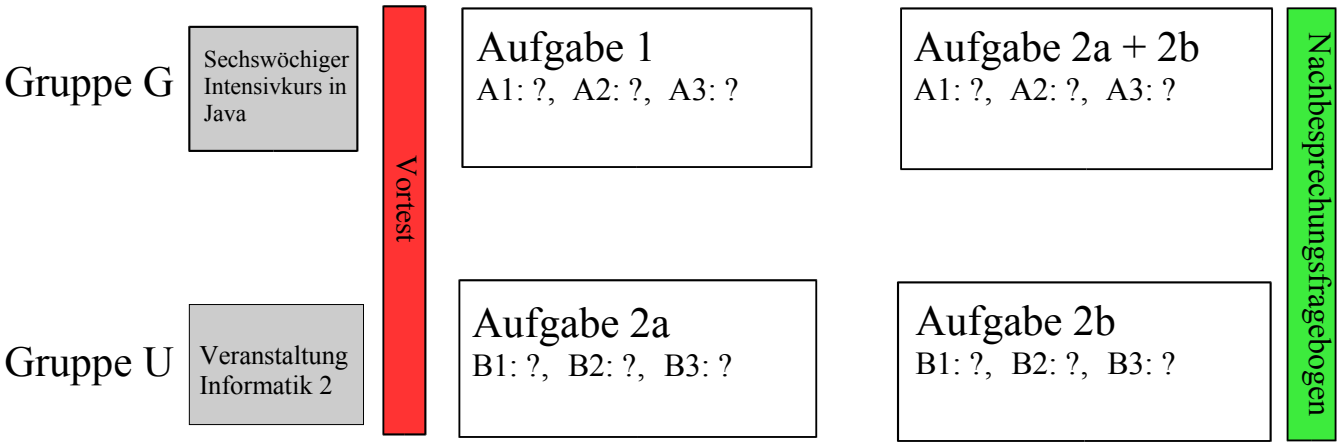
Die abhängige und unabhängige Variable blieben die gleichen wie im Daly Experiment.

3.2Teilnehmer

57 Absolventen der Informatik, die vorher an einem sechswöchigen Intensivkurs in Java teilnahmen und die Klausur erfolgreich bestritten stellten die Gruppe G. 58 Studenten selben Fachs am Ende des ersten Jahres, die davor die Veranstaltung Informatik 2 erfolgreich besucht haben stellten die Gruppe U. Nach eigenen Angaben hatte die Absolventengruppe im Durchschnitt ca. 8 Jahre und die Studentengruppe ca. 6 Jahre Programmiererfahrung.

3.3Durchführung

Das Experiment wurde zwei Mal durchgeführt, jeweils ein mal mit jeder Gruppe. Die Teilnehmer wurden über die unabhängige Variable in dem ihnen vorliegenden Teil nicht informiert. Aus jeder Personengruppe wurden drei kleinere Gruppen nach einem matched-between-subjects Design zusammengesetzt. Dafür wurden alle Personen, im Grunde wie im Experiment von Daly nur mit drei Gruppen, in einer Personengruppe erst nach zu erwartender Leistung gruppiert und dann so in die Gruppen aufgeteilt das es keine signifikanten Leistungsunterschiede zwischen den Gruppen gab. Für die Leistungsbewertung verwendete man bei Absolventen die Leistungen im vorangegangenen Javakurs, für die Studenten die Angaben über das größte bis dahin geschriebene Programm. Die Einordnung der Studenten anhand diese Kriteriums erscheint nicht sehr vertrauenerweckend wurde jedoch in dem Vortest bestätigt.
Es gab die Aufgabe 1 – Behebe das Jahr 2000, Aufgabe 2a – Erweitern um eine neue Art von Anzeige und Aufgabe 2b – Erweitern um eine andere neue Art von Anzeige. Die Gruppen, die aus der Studentengruppe hervorgingen bearbeiteten die Aufgaben 2a und 2b, die Absolventen alle 3.

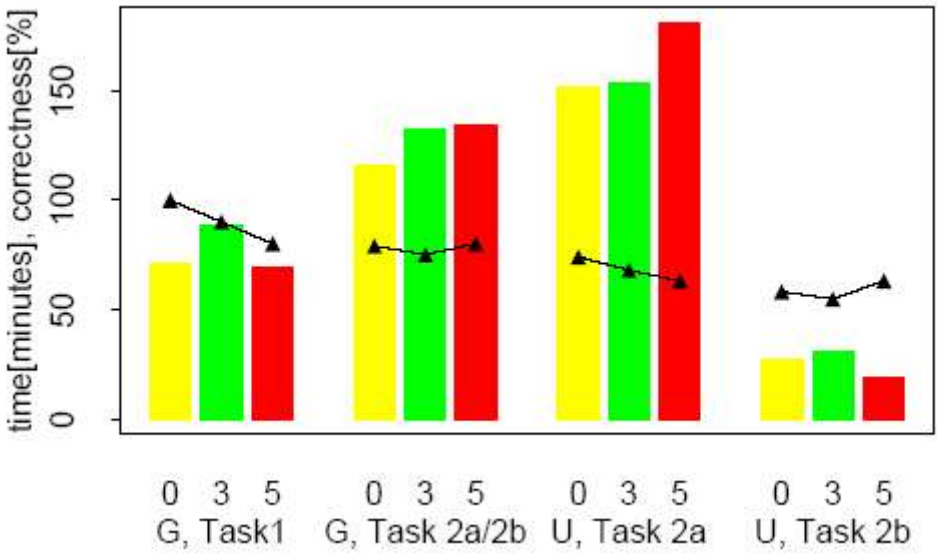


Das interessanteste an den Aufgaben ist das es bei Aufgabe 1 in der Vererbungstiefe 5 Version nur halb so viele Änderungen erforderlich waren wie in den anderen beiden, bei Aufgabe 2a die Funktionalität die für die Lösung nötig war in Vererbungstiefe 5 Version am heftigsten verteilt und in der Aufgabe 2b die Lösung ganz leicht durch verwenden der Lösung der Aufgabe 2a mit nur kleinen Änderungen möglich war aber nur die Vererbungstiefe 5 Version profitierte davon am meisten.
Neben der Zeitmessung zwischen der Aushändigung der Aufgaben und der Abgabe wurde jede Compelierung und jede Programmausführung samt Zeit mit aufgezeichnet. Jede fertige Lösung wurde nach dem Grad bewertet inwieweit sie die Aufgabe gelöst hat.

3.4 Analyse

Bei der inneren Gültigkeit wurde befürchtet es könnten *Unterschiede in Programmen* die nichts mit Vererbung zu tun haben (Störvariablen) und während der Umwandlung eingeschleppt wurden für Unvergleichbarkeit sorgen. *Auswahleffekte* aus durch Zufall unausbalancierten Gruppen wurden in betracht gezogen. Die im Vortest abgefragten Antworten wurden auf das Verhältnis von richtigen und falschen Antworten untersucht aber es gab keine signifikanten Unterschiede.
In der äußeren Gültigkeit wurden zwei Unterschiede zur wirklichen Wartung angegeben. Zum einen *erfahrenere Programmierer* und zum anderen *Struktur und Komplexität von Programm und Aufgabe*. Die Verwendung von Studenten von denen man nicht so gut verallgemeinern kann trifft hier wohl nur auf die Gruppe A zu, dafür aber so richtig. Bei den Absolventen ist es aber umso besser da diese so gut wie die tatsächlichen Programmierer sind. Für Wartung werden in der Realität auch nicht typischerweise die erfahrensten Programmierer eingesetzt. Die Verwendung der Vererbung war auch ziemlich geradlinig so dass nicht erwartet wurde das die Studenten mit kurzer Erfahrung in Java deshalb die Resultate beeinflussten. Bei Struktur und Komplexität könnte konkret die Qualität vorhandener Dokumentation eine Rolle spielen. Im allgemeinen steht diese Vermutung aber für all das was schwer in einem Experiment gemessen werden kann da man sonst in einem daraus entstehenden faktoriellen Entwurf (mehr als ein sich ändernder Faktor) die Wirkung nicht sauber getrennt werden kann. Eine Andere Spur zur Äußeren Gültigkeit ist aber mit Sicherheit auch die *Einarbeitung*, ein realer Programmierer führt nicht eine Aufgabe pro Programm aus und je mehr man an einem Programm rumbastelt um so mehr lernt man es kennen. Insgesamt ist die Art der Wartungsaufgabe ein wichtiger Faktor der noch Erforschungsbedarf hat.

Leider wurde keine Wiederholung für die Version mit dem Jahr 2000 Problem gemacht es fehlt somit ein Vergleich für diese Ergebnisse.



Zu Gruppe G:
Die Resultate der Aufgabe 1 unterstützen die Hypothese A nur sehr gering sind aber mit der Hypothese B mehr übereinstimmend. Überraschenderweise waren in der Vererbungstiefe 5 Version viel Unachtsamkeitsfehler trotz geringem Änderungsbedarf. Die Resultate der Aufgabe 2a/2b unterstützen Hypothese A moderat und sind mit der Hypothese B unvereinbar.

Zu Gruppe U:
Die Resultate der Aufgabe 2a unterstützen beide Hypothesen, signifikant aber nur beim Zeitunterschied zu Vererbungstiefe 5 Version. Die Resultate der Aufgabe 2b sind davon beeinflusst das mansche während diese Aufgabe ausschieden. Am meisten war davon die Vererbungstiefe 5 Gruppe betroffen. Da es wie erwartet die von unterem Leistunkspektrum waren, verzerrte es erwartungsgemäß die Ergebnisse.

Die Antworten auf den Nachbesprechungsfragebogen haben eine klare Tendenz zu einfacherer und fehlerfreieren Wartbarkeit der „flachen“ Version. Die Vererbungsstruktur Version erhielt am meisten Zustimmung von den Absolventen.

Insgesamt werden beide Hypothesen von den Resultaten unterstützt. Die Aufgabe 2b die die Vererbungsversion bevorteiligte führte zu gegenteiligen Resultaten ohne die alle anderen Resultate die Erwartungen unterstützten. Ganz ausschließen sollte man die Aufgabe 2b nicht, ist sie doch neben Aufgabe 1 ein gutes Beispiel für die Variabilität der Aufgaben in Wirklichkeit und ein Einspruch zu Gunsten der Vererbung. Sicher wird hier beanstandet das es eine vorteilhaftere Lösung für die Vererbungsversion gibt, aber das ist die Errungenschaft der Vererbung von dessen Vorteil sie lebt und der nicht als ungerecht angesehen werden darf.

Das Resultat dieses Experiments ist damit: Programme mit geringerer Vererbungstiefe sind leichter zu warten.

4.Andere Experimente

Cartwright machte eine Replikation des Vergleichs eines Vererbungstiefe 3 Programms mit seinen „flachen“ Äquivalenten. Das aus 5 studentischen Teilnehmern bestehende Experiment fand die „flache“ Version 40% schneller zu arbeiten als die mit der Vererbungsstruktur. Die Schlussfolgerung war das nicht einmal die Vererbungstiefe 3 Vorteile in der Wartung biete.

Harrison u.a. führten ein etwas seltsam abgewandeltes Experiment das aber im Grunde an gleichartigen Programmen aber anderen Aufgaben arbeitete. Die ebenfalls studentische Teilnehmer arbeiteten unter begrenzter Zeit nur auf Papier und mussten aus den vorgelegten Programmen einige Ausgaben rausfinden, das Vererbungsdiagramm rekonstruieren und zum Schluss die Stellen rausfinden wo Veränderungen angebracht werden müssen um eine bestimmte Verbesserung zu erhalten. Gearbeitet wurde an allen drei („flacher“, Vererbungstiefe 3 und Vererbungstiefe 5) Strukturversionen. Anschließend wurde nur die Korrektheit der Lösungen verglichen. Es wurde festgestellt das sowohl die Vererbungstiefe 3 als auch die Vererbungstiefe 5 Version dazu tendierten mehr Fehler zu provozieren als die „flache“ Version.

5.Metaanalyse

Wenn man nach all dem nach einer idealen Vererbungstiefe fragen würde, wäre keine klare Antwort darauf möglich. Beide hier ausführlich betrachteten Experimente kommen zu einem gegenteiligen Schluss was aber nicht daran liegt das eines der Experimente grob falsch oder ungültig war. Es wird vermutet das diese Antwort viel mehr von Programm selbst und der Änderung die daran ausgeführt werden muss abhängt.
Anhand der Experimente von Daly, Preschelt und Cartwright wurde im Nachhinein nach anderen Erklärungen gesucht. Dazu wurden Modelle zur Vorhersage des Wartungsaufwandes erstellt die zu einer Endeckung eines viel versprechenderen Faktors führten. Die Modelle sollten die Zeit für eine Erweiterungsaufgabe vorhersagen. Die Vorhersagen beziehen sich auf die durchschnittliche Gruppenarbeitszeit und den absoluten Zeitaufwand. Die interessantesten von allen erstellten waren die folgenden:
Für den geradlinigsten Ansatz gemäß den Hypothesen könnte mit einem Modell das lediglich die

$$t = f(\text{Vererbungstiefe})$$

Vererbungstiefe als Eingabe bekam nur 10 Prozent der Experimentergebnisse erklärt werden. Dazu kam auch ein relativ großer Zufallswert ($p = 0.23$) was für die Vorhersagen dieses Modells bedeutet das sie mit einer Wahrscheinlichkeit von 1/5 durch Zufall die Realität trafen. Ein ähnlich angelegtes Erklärungsmodell, das von den Hierrarchiesprüngen die während der Bearbeitung der Aufgabe zum Verständnis gemacht werden müssen eine Vorhersage zu erzielen

$$t = f(\text{Hierrarchiesprünge})$$

versuchte, gelang zu einem Modell das 55 Prozent der Experimentergebnisse erklärte. Eine viel bessere Vorhersage gelang aber als man allein die Anzahl der Methoden die verstanden werden müssen um eine Aufgabe zu lösen als Variable heranzog.

$$t = f(\text{Methoden zu verstehen})$$

Dieses Modell erklärte schon 84 Prozent der Experimentergebnisse. Eine Kombination der letzteren Variable mit Vererbungstiefe lieferte ein Modell das fast keine Verbesserung (85 Prozent)

$$t = f(\text{Vererbungstiefe, Methoden zu Verstehen})$$

der Vorhersage erzielte dafür aber einen sehr hohen Zufallswert ($p = 0.33$) aufwies. Das ist ein klares Indiz das die Vererbungstiefe praktisch keine Rolle beim Wartungsaufwand einnimmt. Ein sehr wichtiger Aspekt in der Programmierung, die Erfahrung, kann nicht allein als Variable für Vorhersagen eingesetzt werden. Zu stark ist die Arbeitszeit von Programm und Aufgabe abhängig. In Verbindung mit „Methoden zu verstehen“ lieferte die binäre Variable Erfahrung (die

Unterschiede der Gruppen waren steht's Anfänger oder Profi)

$$t = f(\text{Methoden zu verstehen, Erfahrung})$$

gute 94 Prozent Überdeckung der Experimentergebnisse. Auch hier bewies die Zunahme der Vererbungstiefe als dritte Variable kaum Verbesserung (96 Prozent).

Der neu entdeckte Faktor war damit die „Anzahl Methoden zu verstehen".

6.Fazit

In beiden Experimenten steht das Verstehen im Vordergrund, im zweiten sogar noch mehr als im ersten. Es ist deshalb eine recht passende Tatsache das die Resultate am meisten mit der Anzahl der zu verstehenden Methoden erklärt werden konnten. Es ist aber gewiss nicht immer, aber bestimmt oft, der entscheidende Aspekt in der Wartung. Was es nicht mit einbezieht sind Wartungsvorgänge in denen es nicht mehr um Verständnis gilt. Dies kann z.B. sein wenn der Programmierer das Programm schon sehr gut kennt und es um strategische Entscheidungen geht oder solche wo die Machbarkeit erst durch Vererbung ermöglicht wird und eine hohe Komplexität beim Verzicht auf Vererbung du Codevervielfältigung und Konsistenzbedarf führt.
Einen wertvollen Hinweis den die Erkenntnis dieses Experiments gibt ist das es sich weniger um Vererbungstiefe handelt wenn es darum geht einen Einflussfaktor für den Wartungsaufwand zu finden sondern mehr um begleitende Eigenschaften des Wartungsvorgangs die gewiss auch mit der Vererbungstiefe zusammenhängen können.
An der in Metaanalyse entdeckte Vorhersagengröße „Anzahl Methoden zu verstehen" ist in der Praxis kaum praktizierbar. Sie ist einfach schwer zu ermitteln denn ihre Bestimmung würde die Lösung der Aufgabe erfordern. Einen Nutzen für die Praxis kann man dem ganzen aber dennoch abgewinnen wenn man bedenkt das bei dem Entwurf eines Softwaresystems im voraus für häufige Wartungsaufgaben ein leichterer Weg vorgezeichnet werden kann.
Die Hypothesen waren in beiden Experimenten irreführend. Dies kann man im Nachhinein als relativ gesichert ansehen. Die Vererbungstiefe ist nicht der wichtige Faktor für den man es anfangs hielt. Die Versuche brachten zu tage das es noch sehr viele andere Aspekte die vor allem im Programm selbst und in der Wartungsaufgabe stecken, mehr die Wartbarkeit beeinflussen. Auch die hier nicht betrachtete Einarbeitung, also das Wissen das durch lösen vorangegangener Aufgaben erlangt wurde, ist von größerer Bedeutung. Dieser kleiner Forschungszyklus, welcher mit einer Umfrage startete und zu einer Erkenntnis gelang, brachte keinen großen Sprung in dem Sinne das es nur eine Ablehnung eines von vielen möglichen wartungsrelevanten Faktoren ergab aber dennoch ein Beitrag für das Thema Wartung war und Hinweise für weitere Forschung lieferte.

7.Literatur

A) An Emprical Study Evaluating Depth of Inheritance on the Maintainability of Object-Oriented Software von Daly, Brooks, Miller, Roper und Wood
B) A Controlled Experiment on Inheritance Depth as a Cost Factor for Code Maintenance von Preschelt, Unger, Philippsen und Tichy
C) Kontrollierte Experimente in der Softwaretechnik, Lutz Preschelt, Springer Verlag