Experimentbeispiele

An Experiment about Static and Dynamic Type Systems

Doubts about the Positive Impact of Static Type Systems on Development Time

Objective

- Evaluierung des Einflusses von statischen und dynamischen Typsystemen auf Entwicklungszeit
- Verschiedene Argumente für und gegen statische Typsysteme
- Keine Forschungshypothese, sondern Forschungsfrage (nicht explizit genannt)

Variablen (1)

- Unabhängige Variablen:
  - Typsystem
    - 2 Stufen (statisch und dynamisch)
    - 2 gleiche Programmiersprachen mit unterschiedlichem Typsystem
  - Aufgabe
    - 2 Stufen (Scanner und Parser)
- Abhängige Variablen:
  - Entwicklungszeit

Variablen (2)

- Tool-erfahrung

Variable	Kontroll-Technik		Gemessen/Gesichert	
	Wie	Warum	Wie	Warum
Erfahrung	Matching	Erfahrung spielt große Rolle bei Entwicklungszeit	Interview	Um möglichst genaue Einschätzung von Erfahrung
Programmier-sprache	Konstant	Spielt große Rolle bei Entwicklungszeit	Neue Sprache	Garantiert gleiche Erfahrung aller Probanden
Tool-erfahrung	Konstant	Spielt große Rolle bei Entwicklungszeit	Neues Tool	Garantiert gleiche Erfahrung aller Probanden

Material

- Purity, ein paar Bibliotheken
- PurityIDE
- -> beides kontrolliert am besten die Einflüsse von Sprache und IDE

- Spezifikation eines Parsers in kontext-freier Grammatik
- Geheimhaltungserklärung

Aufgabe

1. Scanner:
   - Scannt Wort und entfernt Sonderzeichen
   - Characters als Tokens
2. Parser:
   - Bekommt Wort als Input
   - Gibt wahr oder falsch aus, abhängig davon, ob Wort Teil der Grammatik ist

Design

- Latin Square
- Within-Subjects nicht sinnvoll, da man in einem Typsystem denkt
- Zeitaufwand zu lang

Probanden

- 49 Bachelor-Studenten
- Erfahrung mit formalen Sprachen und Java
- Unerfahren mit Parser-Implementierung

Ausführung

- Erst Interview zur Erstellung der Gruppen
- 16 Stunden Training (dynamisch)
- 18 Stunden Training (statisch -> braucht mehr Erklärung)
- 27 Stunden, frei verteilt auf 4 Arbeitstage
- Probanden durften kein Material mit nach Hause nehmen

Deviation

- There are no deviations to report

Threats to Internal Validity

- Lernen einer neuen Sprache
  - Kontrolliert durch:
    - Sprache ist Java-ähnlich und einfach gehalten
    - Intensives Training (16/18 ausreichend?)
    - Vergleich zwischen zwei Gruppen, die beide neue Sprache lernen mussten (Effekt müsste in beiden Gruppen gleich groß sein)
  - auch Gefahr für externe Validität

Threats to Internal Validity (cont.)

- Gruppenbildung basierend auf Interviews
  - Nur diskutiert, nicht kontrolliert
- Unklar, wann welche Aufgabe bearbeitet wurde

Threats to Construct Validity

- Aufgabe evtl. ungeeignet
  - Generell: Tradeoff zwischen kleiner, kontrollierbarer Aufgabe und Realismus
  - Hier: eine große Aufgabe über 27h, um Forschungsfrage beantworten zu können

Threats to External Validity

- Studenten als Probanden
  - Studenten können unter bestimmten Umständen als Experten dienen
  - Unbekannte Sprache, die auch Experten hätten lernen müssen
- Sprache
- Tool support ("künstliche" IDE)

Interpretation (Scanner)

- Nochmal Begründung, warum statisches Typsystem besser sein sollte (u.a. statisches Typsystem verhindert Fehler)
- Weitere Analyse der Daten: Debugzeiten von Logs rekonstruiert und zwischen Gruppen verglichen -> kein Unterschied, also ist Aufwand zum Debuggen gleich groß
- Letztendlich keine Begründung/Deutung der Ergebnisse

Interpretation (Parser)

- Wieder keine Unterschiede
- Wieder Debugzeiten verglichen: Dynamisches Typsystem schneller beim Debuggen
- Allerdings keine weitere Erklärung

Bewertung der Interpretation

- Darstellung Daten und Interpretation der Daten nicht deutlich getrennt
- Interpretation bricht mittendrin ab
- Vermischung von Conclusion und Interpretation

Using Students as Subjects

An Empirical Evaluation

Objective

- Studenten werden oft herangezogen in Experimenten, und dann Aussagen über Experten getroffen
- Eignung von Studenten, um Aussagen über Experten treffen zu können
- To what extent are students capable of imagining how industry professionals work in a complex requirements engineering decision process?

Variablen

- Unabhängig:
  - Status (Student [2 Sichten], Experte)
  - Daten von Experten aus anderem Artikel
- Abhängig:
  - Auswahl von Anforderungen
- Störvariablen:
  - Nicht klar erkennbar

Material/Task

- Fragebogen
  - Studentensicht auf Anforderungen
  - Expertensicht aus Sicht der Studenten auf Anforderungen

- Auswahl von Anforderungen aus zwei Sichten

Design

- Between-Subjects (Studenten vs. Experten)

Probanden

- Studenten aus Requirements-Engineering-Kurs
- Durchschnittsalter: 26
- Verschiedene Kulturen
- Praktische Erfahrung: 0-2 Jahre, 2 Studenten mit 7 Jahren

Ausführung/Deviation

- Anforderungsauswahl über 4 Wochen
- Auswahl aller Studenten wurde besprochen
- Jede Woche konnten Anforderungen angepasst werden

- There are no deviations to report

Threats to Validity

- Erfahrung mit Requirements
  - Studenten aus Requirements-Kurs, dadurch grundelegende Erfahrung existent
- Kulturelle Unterschiede
  - Verschiedene Kulturen (schlecht für interne Validität, aber besser für externe Validität)
- Stichprobengröße?

Interpretation

- Studenten können sich in Lage von Experten versetzen
- Studenten lassen sich nicht unbedingt von Versuchsleitern beeinflussen

- Bewertung:
  - Kurz und knapp auf die wichtigsten Dinge konzentriert
  - Gut für ein Short Paper

An Empirical Study of the Effects of Personality in Pair Programming using the Five-Factor Model

Objective

- Effektivität von Pair-Programming in Ausbildung verbessern
- Fokus auf Persönlichkeitstypen
- Hypothese:

  Unterschiede in Persönlichkeit beeinflussen Effektivität von Studenten, die Pair-Programming angewendet haben

Variablen

- Unabhängig:
  - Persönlichkeitskombinationen (IPIP-NEO)
  - 2 Stufen: gleiche/verschiedene Persönlichkeit
- Abhängig:
  - Effektivität bei Pair-Programming (Benotete Leistung bei Test und 3 Aufgaben)
  - Zufriedenheit d. Studenten (Fragebogen)
- Störvariablen:
  - Motivation
  - Programmiererfahrung

Material/Aufgabe

- Persönlichkeitsfragebogen
- Demografischer Fragebogen
- Programmiererfahrungsfragebogen
- Pair-Programming-Aufgaben (nicht weiter spezifiert)

Probanden

- Studenten (BSc)
- Einführungskurs Informatik

Ausführung/Deviation

- Persönlichkeitsfragebogen
- Demografischer/Programmiererfahrungs-fragebogen
- Aufgaben über das ganze Semester

Threats to Internal Validity

- Programmiererfahrung nicht kontrolliert
- Partnerwechsel; nur diskutiert, nicht kontrolliert (gehört eigentlich in Abschnitt "Deviation")

Threats to External Validity

- Stichprobengröße
- Nur eine Dimension des Five-Factor-Models

Interpretation

- Diskussion von Geschlecht und Fähigkeit als weitere Faktoren, inkl. Begründung durch Quellen
- Bezug zu anderen Forschungsarbeiten

- Bewertung:
  - Guter Bezug zu anderen Forschungsarbeiten
  - Zu wenig Bezug zu eigenen Daten

Empirical Studies of Programming Knowledge

Objective

- Warum sind Experten besser als Anfänger?
- Do expert programmers possess programming plans and discourse rules?
- Programming plans:
  - Programmfragmente, die stereotypische Handlungssequenzen repräsentieren, z.B., Itemsuche in Liste
- Discourse rules
  - Coding conventions, z.B., Variablennamen

Variables

- Unabhängig:
  - Pläne; 2 Stufen: konform oder nicht-konform
  - Aufgabe; 4 Stufen
  - Erfahrung; 2 Stufen
- Abhängig:
  - Korrektheit von Antworten
  - Antwortzeit
- Störvariable:
  - Motivation (5$ für Teilnahme)
  - Reihenfolge (Randomisierung)
  - Zeitdruck (kein Zeitlimit)

Material/Aufgabe

- 4 verschiedene Programme in jeweils 2 Versionen in Pascal
- Auf Papier
- Fill-in-the-blank: Probanden sollten fehlende Codezeile ersetzen

Probanden

- Studenten (1. Semester, oder mindestens 3 Programmierkurse bzw. Master)

Ausführung/Deviation

- Probanden bearbeiteten Aufgaben (keine Erwähnung von Einführung, Fragebögen,…)
- Keine Abweichungen

Threats to Internal Validity

- Sind die gewählten Programme, die Pläne verletzen, wirklich repräsentative Beispiele?

Threats to Construct Validity

- Messung von Programmverständnis durch Fill-in-the-blank und recall -> wirklich geeignet dafür?

Threats to External Validity

- Nur einige discourse rules
- Abhängig von Programmiersprache (Pascal)
- Nur kurze Programme

Interpretation

- Anwendung von Plänen oder nicht (top-down vs. bottom-up comprehension)
- Bei Programmen, die Pläne verletzen, verhielten sich Probanden so, als würden die Pläne eingehalten werden
- Pläne sollten eingehalten werden, sonst werden Experten so schlecht wie Anfänger

Bewertung der Interpretation

- Darstellung Daten und Interpretation der Daten nicht getrennt
- Interpretation könnte tiefer gehen (allerdings: Artikel ist von 1984; da galten vmtl. Andere Konventionen)
- Vermischung von Conclusion und Interpretation

Understanding Exception Handling: Viewpoints of Novices and Experts

Objective

- Herausfinden, wie Anfänger und Experten mit Exceptions umgehen

Variablen

- Unabhängig: Erfahrungslevel
  - Anfänger
  - Experte
- Abhängig:
  - Exception-Handling-Strategien
- Störvariablen:
  - Erfahrung mit Java

Material-Anfänger

- Beispiel-Fragen aus dem Interview-Leitfaden:
  - What approach do you follow to understand exception-flow information in a program?
  - When working with code (e.g., coding, testing, reviewing, and understanding) how often do you pay attention to the functionality associated with exception-handling?
  - Are there scenarios in which you avoid/ignore using exception-handling in your programs? (Yes/No) If yes, when do you do so? Why do you do this?
  - Are you satisfied with the way you approach understanding the program with respect to exceptions? (Yes/No/Maybe)

Material-Experten

- Beispiel-Fragen aus dem Interview-Leitfaden:
  - How do you perceive error/exception-handling when you are coding/designing? Why?
  - When do you typically start thinking about exceptional conditions? Has your strategy changed over time? How? What factors caused these changes?
  - Have you worked with junior/novice developers? Have you observed any typical patterns in the way they handle the exceptions?
  - How did you learn about exception-handling? (Educated in the work place/college/learned on your own.)

Probanden/Design

- Anfänger:
  - 8 Praktikanten, graduate students
  - Im Durchschnitt 2 Jahre Programmiererfahrung
- Experten
  - 6 Experten von 5 verschiedenen Firmen
  - Min. 5 Jahre professionelle Programmiererfahrung
- Jeder Proband wurde einmal befragt

Ausführung

- Vorher Aufklärung, Erlaubnis über Audioaufzeichung und e-Mail-Kontakt
- Ca. 1 Stunde pro Proband

Auswertung-Anfänger

- Zusammenfassung der Antworten der Probanden in 3 Kategorien
  - Approaching Exception Handling (ignore)
  - Using Exception Handling (debugging)
  - Perceiving Exception Handling (forced)

Auswertung-Experten

- Zusammenfassung der Antworten der Probanden in 3 Kategorien
  - Approaching Exception Handling (handle right away)
  - Using Exception Handling (failing gracefully in unexpected situations)
  - Perceiving Exception Handling (forced)
- Strategie von Anfängern:
  - Ignorieren
  - Verallgemeinern
  - Fehlendes Logging/unvollständiges weiterreichen

Threats to Internal Validity

- Interviews mit Experten nicht face-to-face
  - Kontrolliert durch Aufzeichnung der Daten und email-Kontakt bei Nachfragen
- Bei Anfänger Fokus auf Java, bei Experten mehrere Sprachen
  - Durch Ergebnisse von Anfängern Experten-Befragung erweitert
  - Beide Gruppen haben von sich aus immer andere Sprachen mit Java verglichen

Threats to Construct Validity

- Interview-Leitfaden könnte Probanden beeinflusst haben oder unvollständig sein
  - Wurde in Pilotstudie getestet und angepasst

Threats to External Validity

- Stichprobengröße (aber qualitative Studie erlaubt kleinere Größe)
- Anfänger aus einer Firma, Experten aus verschiedenen
  - Könnte zu kulturell-organisatorischem Bias führen
  - Aber Studenten haben gerade angefangen, hatten also genau wie Experten unterschiedlichen Hintergrund

Interpretation

- Finden von Kategorien in Antworten
- Beispiel-Aussagen von Probanden

Literatur

- Shah, Görg, and Harrold. Understanding Exception Handling: Viewpoints of Novices and Experts. TSE, 36(2), pp. 150-161, 2010.
- Nachfolgeexperimente:
  - Shah and Harrold. Exception Handling Negligence Due To Intra-Individual Goal Conflicts. CHASE, pp. 80-83, 2009.
  - H. Shah, Görg, and Harrold. Visualization of exception handling constructs to support program understanding. In *Proceedings of the 4th ACM Symposium on Software Visuallization,* pages 19–28, Sep 2008.
  - Shah, Görg, and Harrold. Why do developers neglect exception handling? In *Proceedings of the 4th International Workshop on Exception Handling, pages 62–68,* Nov 2008.

Allgemein: Threats to Validity

- Threat beschreiben
- Mögliche Auswirkung erklären
- Darlegen, wie und warum Threat minimiert wurde, bzw. warum Minimierung nicht möglich war
- Dabei Einteilung:
  - Internal validity
  - External validity
  - (Construct validity)
  - (Statistical conclusion validity)

Allgemein: Interpretation

- Bestätigung der Hypothese
  - Nochmal kurz Erklärung/Begründung zusammenfassen, die zur Aufstellung der Forschungshypothese führte
  - Evtl. zusätzliche/besondere Ergebnisse erwähnen
- Ablehnen der Hypothese
  - Begründen, warum Hypothese nicht bestätigt werden konnte

  -> educated guesses, weitere Datenanalyse (z.B. Kommentare der Probanden), Heranziehen von Quellen mit ähnlichen Ergebnissen