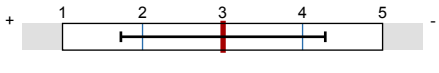
Compilerbau (INMW-CB)
Erfasste Fragebögen = 12

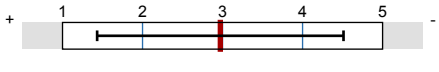
Globalwerte

Globalindikator



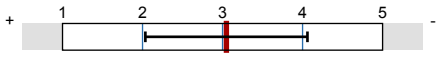
mw=3
s=1.3

Lehrerfolg



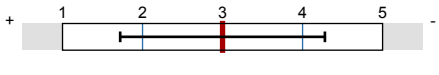
mw=3
s=1.5

Dozentin/Dozent



mw=3
s=1

Gesamtbewertung

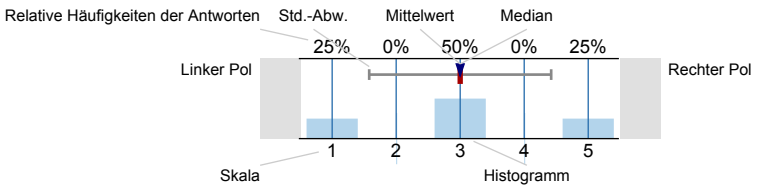


mw=3
s=1.3

Auswertungsteil der geschlossenen Fragen

Legende

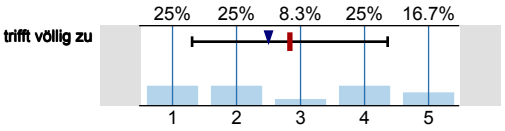
Fragetext



n=Anzahl
mw=Mittelwert
md=Median
s=Std.-Abw.
E.=Enthaltung

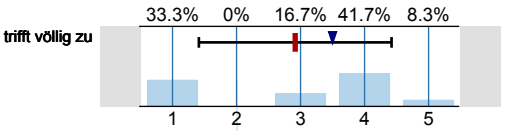
Lehrerfolg

1. Die Ziele der LV sind klar erkennbar.



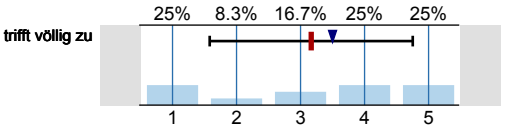
n=12
mw=2.8
md=2.5
s=1.5

2. Die LV fördert mein Interesse an dem Thema.



n=12
mw=2.9
md=3.5
s=1.5

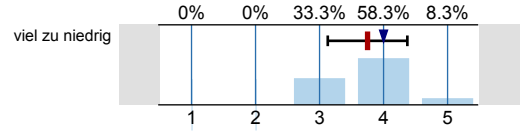
3. Ich habe in der LV ein tiefes Verständnis für den Stoff gewonnen.



n=12
mw=3.2
md=3.5
s=1.6

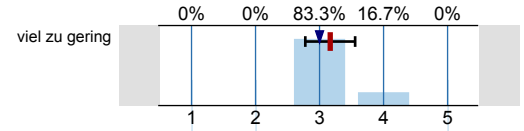
Angemessenheit von Schwierigkeit und Umfang

4. Der Schwierigkeitsgrad des behandelten Stoffes ist mir ...



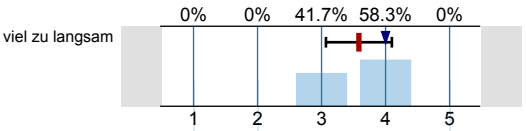
n=12
mw=3.8
md=4
s=0.6

5. Die Stoffmenge der LV ist mir ...



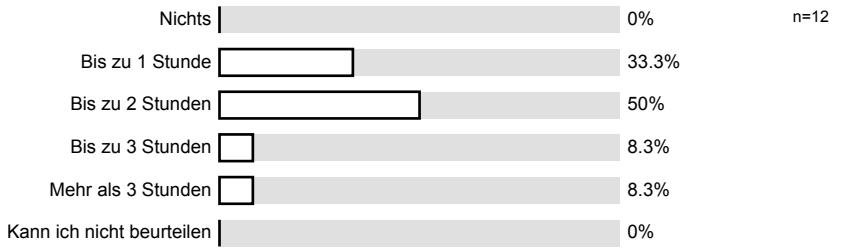
n=12
mw=3.2
md=3
s=0.4

6. Der/die Dozent/in präsentiert den Stoff für mich ...

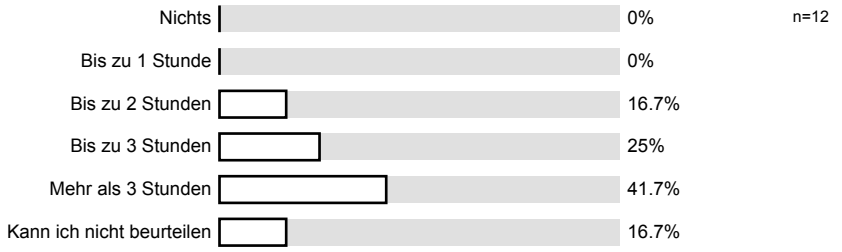


n=12
mw=3.6
md=4
s=0.5

7. Versuchen Sie abzuschätzen, wie viele Stunden pro Woche an Vor- und Nachbereitung von Ihnen für diese LV aufzuwenden waren:
Wie viele Stunden haben Sie im Schnitt investiert?

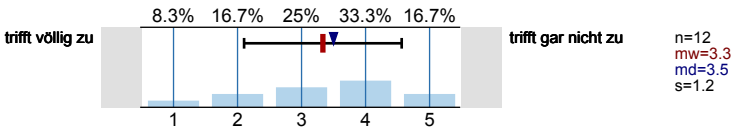


8. Und wie viele Stunden wären eigentlich notwendig?

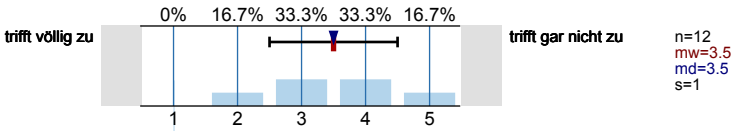


Dozentin/Dozent

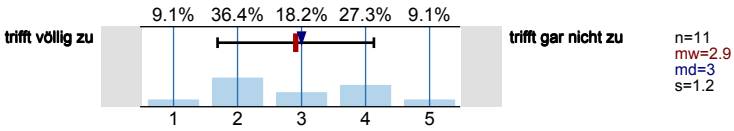
9. Der inhaltliche Aufbau der LV ist stets nachvollziehbar.



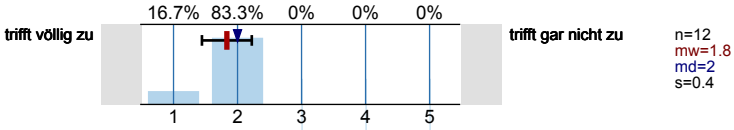
10. Der/die Dozent/in erklärt auch komplexe Sachverhalte verständlich.



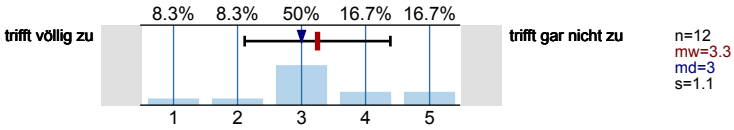
11. Der/die Dozent/in stellt den Anwendungsbezug her.



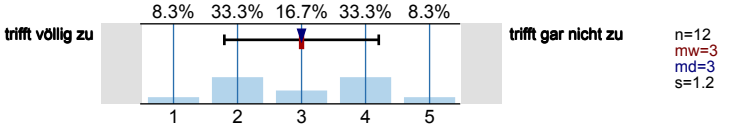
12. Der/die Dozent/in geht auf Fragen und Anregungen der Studierenden ausreichend ein.



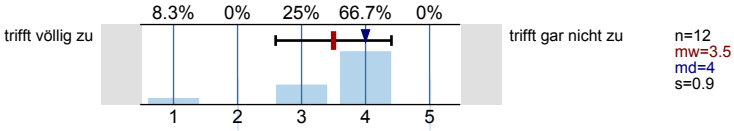
13. Die vom Dozenten eingesetzten Medien/ Hilfsmittel (Folien, Tafelbilder u.ä.) sind eine große Unterstützung beim Verstehen des Stoffes.



14. Der/die Dozent/in stellt in ausreichendem Maße Unterlagen der LV (Folien, Skripte, Übungsaufgaben u.ä.) im Intranet/Internet zur Verfügung.

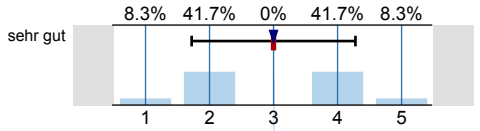


15. Dieses Online-Angebot ist eine große Unterstützung beim Verstehen des Stoffes.



Gesamtbewertung

18. Insgesamt bewerte ich diese LV mit der Note



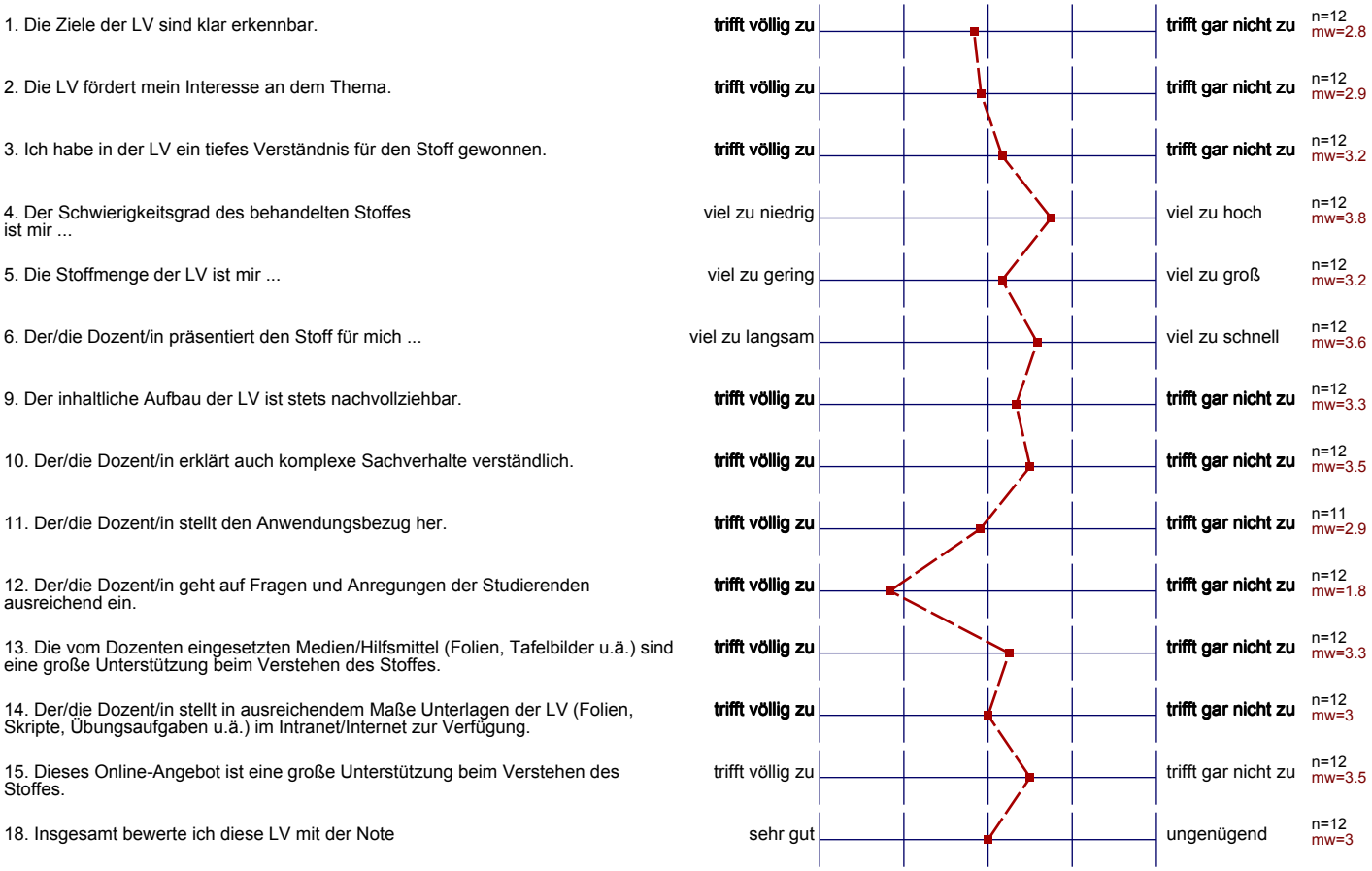
n=12
mw=3
md=3
s=1.3

19. In welchem Studiengang studieren Sie?

n=12

Profillinie

Teilbereich: Informatik, Mathematik und Naturwissenschaften
Name der/des Lehrenden: Prof. Johannes Waldmann
Titel der Lehrveranstaltung: (Name der Umfrage) Compilerbau



Auswertungsteil der offenen Fragen

Gesamtbewertung

16. Was finden Sie an dieser LV besonders gut?

- - (2 Nennungen)
- Das Haskell nebenbei noch gelehrt wird. Livebau vom Interpreter (auch in der Vorlesung) und dadurch den praktischen Beug.
- Es wird versucht, ein großes Projekt von anfang an innerhalb der Seminare aufzubauen und über die Seminare immer weiter zu entwickeln.
- Es wird versucht, ein komplexes Beispiel durch die gesamte Vorlesung zu ziehen. Dadurch hat man immer einen Orientierungspunkt.

  Die Vorlesung behandelt interessante und zum Teil auch für mich bisher unbekannte Dinge, was im Master-Studium nicht immer der Fall ist.
- Haskell macht alles kaputt. Haskell ist war schön für die Lehre und dem Umgang mit Lambda-Funktionen - praxtisch findet Haskell keine Verwendung.
- Mathematischer Bezug, Haskell
- Ständiger Praxisbezug.

17. Was finden Sie an dieser LV besonders schlecht?

- -
- 1. Zu Beginn der Seminare wurde das zu bearbeitende Projekt niemals in einen Kontext gesetzt. Es wurde einfach gesagt, dass der ghci zu verwenden ist und der Quelltext, den der Professor aufschreibt, einfach abzuschreiben ist. Dabei wird ein globaler Kontext des Projektes vermisst, genauso wie die einzelnen Teilschritte in den nicht vorhandenen globalen Kontext zu setzten sind. Erst in den späteren Seminaren wurde etwas konkret erklärt, was die nächste Entwicklungsstufe des Projektes tun soll.

  2. Eine fundierte theoretische Betrachtung von Compilern und den dazu gehörigen Konzepten hat nie statt gefunden. Jeglicher Begriff und jegliche Technik wird ausschließlich durch einen entsprechenden Haskell-kontext begründet, jedoch nie in einen notwendigen Compiler-kontext gesetzt.

  3. Der Einsatz der Sprache Haskell ist in diesem Kontext sicherlich nicht unnötig, jedoch sehr unglücklich. Die Fachkompetenz der Studenten bzgl. Haskell ist in den meisten Fällen unzureichend bzw. nicht existent. Dieser Sachverhalt erhöht den Schwierigkeitsgrad der LV enorm, da Haskell in einem solch fortgeschrittenem Syntax nicht intuitiv zu begreifen oder zu erlernen ist.

  4. Von einem Professor, der uns einige Semester zuvor noch über die Thematik der Code Smells unterrichtet hat, erwartet man eigentlich eine stärkere Beachtung der damals vorgestellten Konzepte. Vor allem die Benennung der Variablen ist stark verbesserungswürdig (uncommunicative name) und erschweren das Verständnis des Seminarprojektes erheblich.

  5. Der aktuelle Stand des Seminares ist nur unter größeren Umwegen zu erhalten. Hat man aus verschiedenen Gründen keine Möglichkeit, ein Seminar zu besuchen, erhält man so quasi keine Chance mehr, wieder in den aktuellen Seminarstand hineinzufinden. Gerade durch die starken Refaktorisierungsmaßnahmen der letzten Wochen, ist das Verständnis des Projektes wieder stark eingeschränkt, da man die Änderungen nur schwer nachvollziehen kann. Leider ist der gesamte Quelltext auch nicht kommentiert.

  Leider gelingt es dem Professor nicht, einen direkten Compiler-bezug zu den behandelten Thematiken herzustellen. Dadurch wird das Ziel der Lehrveranstaltung, nämlich die Vermittlung von den Techniken eines Compilers, absehbar leider nicht erreicht. Die Lehrveranstaltung sollte eher in "Funktionale Programmiersprachen (Aufbaukurs)" umbenannt werden, um dem Studenten kein falsches Bild über den Inhalt der LV zu vermitteln.
- Das Skript hat eine sehr hohe Informationsdichte. Die fehlenden Rückbezüge, Erklärungen, Beispiele, Bedeutung für den Compilerbau erschweren die Nachbereitung und das nachhaltige Verstehen des Stoffinhaltes. Eine Vorbereitung ist ebenfalls schwierig, da das Skript zur entsprechenden Vorlesung erst nach der Vorlesung verfügbar ist.
- Man hat nicht das Gefühl, dass Leute wirklich so Compiler bauen.

  Außerdem sind die genauen Ziele der Vorlesung stellenweise völlig unklar, was die Motivation negativ beeinflusst. Es wird zwar in der Regel erklärt, warum man einen bestimmten Mechanismus implementiert, dennoch wird das Gesamtkonzept erst nach und nach sichtbar.

  Ein weiteres Problem ist, dass man regelmäßig in der Übung auf einem anderen Stand als der Professor ist und das Beziehen des aktuellen Standes umständlich ist.

  Außerdem muss ich leider sagen, dass die auf der Seite des Professors verfügbaren Folien keine nachvollziehbare Grundlage zum Verständnis des behandelten Stoffs bilden, d.h. dass man sich nicht nach der Vorlesung ohne Weiteres mit den Folien nochmals beschäftigen kann, um die Konzepte zu verstehen. Fehler und Lücken machen das oft unmöglich und man muss sich den Rest durch Fragen in der Übung oder Beratung mit Kommilitonen zusammenreimen.

  Die Syntax von Haskell nervt.
- Neue Programmiersprache, die das Verständnis für die Problematik nicht immer erleichtert.

- Sehr kryptisch und anspruchsvoll. Über große Strecken reiner ASCII-Schnee ohne unterstützende Grafiken oder (nicht-verbale) Erläuterungen.
- Skript ist sehr knapp gehalten (Erklärungen/Zusammenhänge fehlen); starke Bindung an Haskell;
- Variablennamen könnten "sprechender" in den Beispielen benannt werden. Statt (k,v) wäre später (key, value) besser verständlich.
- Vorlesung ist eher eine "Wie baue ich ein Interpreter in Haskell" als wie alles andere. Klassische Compilerbau mit einer imperativen Sprache wäre hier angebrachter. Auch wenn dem Dozenten die Sprache Haskell gefällt, so gut wie alle Studenten würden sich lieber für eine andere Sprache entscheiden, wo bereits das Verständnis vorhanden ist. Die meiste Zeit wird über was muss man so und so in Haskell umsetzen gespochen, als wie über Compiler als solche - Scanner / Parser (LL, LR, LF,...).

  Möglichkeiten:
  Vorher bereits Wissen zu Haskell schaffen, damit dies nicht Aufgabe der Compilerbau-LV ist
  oder
  Haskell über Bord werfen, dafür eine imperative Sprache nutzen. Möglichst mit C-Syntax-ähnlichen Stukturen, da diese den Studenten bereits vertraut sind und eine Einarbeitung erfolgen kann.