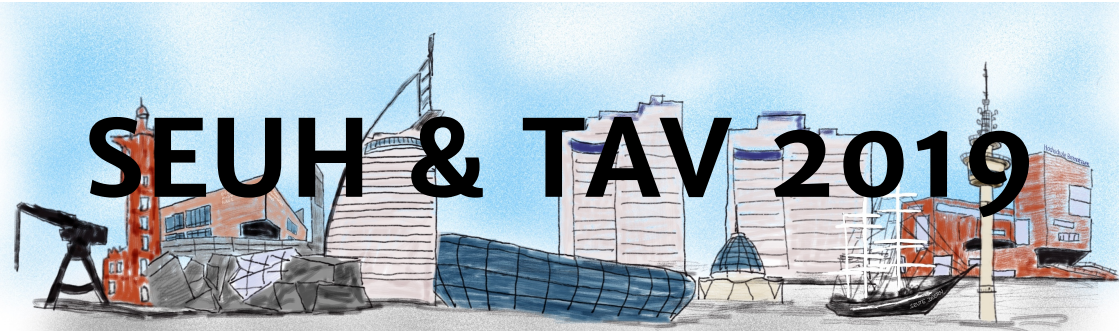
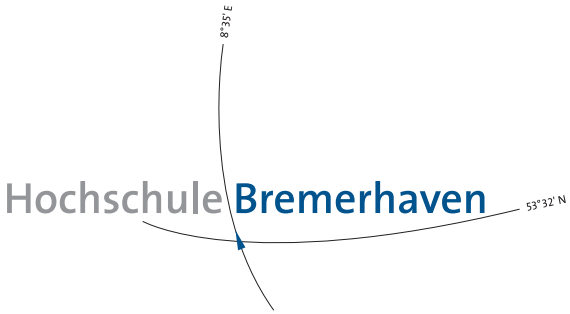
Programmheft



Software Engineering im Unterricht der Hochschulen & GI-Fachgruppentreffen *Test, Analyse, Verifikation*

21. und 22. Februar 2019
Hochschule Bremerhaven



Donnerstag, 21. Februar 2019

9:30 Anmeldung – SEUH und TAV

10:00 Begrüßung – SEUH und TAV - Haus T Raum 002

Karin Vosseberg, Hochschule Bremerhaven
Veronika Thurner, Hochschule München
Andrej Pietschker, Sprecher TAV

Der 16. Workshop der SEUH – Software Engineering im Unterricht der Hochschulen wird gemeinsam mit der Fachtagung der GI-Fachgruppe TAV – Test, Analyse und Verifikation von Software durchgeführt. Von dieser Zusammenarbeit erhoffen wir uns spannende Einblicke in Ausbildung, Forschung und Praxis im Kontext der Qualität von Software und der Qualität von Ausbildung.

Wir freuen uns auf zwei intensive Tage mit vielen interessanten Diskussionen und Ideen, die wir mit in unseren Alltag nehmen können.

10:15 Keynote – SEUH und TAV - Haus T Raum 002

Software-Tests automatisch erzeugen - Frische Ansätze für Forschung, Praxis und Lehre
Andreas Zeller, CISPA Helmholtz-Zentrum für Informationssicherheit, Saarbrücken

Automatisch erzeugte Softwaretests können mit wenig menschlichem Aufwand viele Fehler finden. In diesem Vortrag stelle ich aktuelle Techniken zur Testerzeugung vor, die für ein gegebenes Programm vollautomatisch dessen Eingabesprache ableiten und aus den so entstehenden Grammatiken große Mengen gültiger Testeingaben ableiten.

Unser grammatikbasierter LangFuzz-Testgenerator hat so in den JavaScript-Interpretern von Firefox, Chrome und Edge tausende Fehler gefunden. Die Techniken sind in dem interaktiven Buch "Generating Software Tests" zusammengefasst. In einer Mischung von Text und Programmcode können Leser im Browser direkt mit den Programmen experimentieren und ihren Code live ergänzen und erweitern.

11:15 Kaffeepause

11:30 Session 1 – SEUH und TAV- Haus T Raum 002

UML in der Hochschullehre: Eine kritische Reflexion

Jürgen Anke und Stefan Bente, HFT Leipzig & TH Köln

Trotz ihrer hohen Bekanntheit und Verbreitung ist die Unified Modelling Language (UML) 20 Jahre nach der Vorstellung ihrer ersten Version in der Softwaretechnik nicht unumstritten. Ein wesentlicher Grund ist ihre unklare Nutzung in der Praxis die u.a. durch Ausdifferenzierung von Softwareproduktarten, agilen Entwicklungsansätzen sowie Microservices als Architekturstil mit reduzierter Komplexität beeinflusst wird. Diese Trends befördern eine Abkehr von klassischen Top-Down-Ansätzen mit sequenziellem Vorgehen, das umfangreiche Spezifikationen der Anforderungen und des Entwurfs erfordert. Für die Lehre (insbesondere auf Bachelorniveau) stellt sich daher die Frage, ob wir UML überhaupt noch brauchen und wenn ja, wofür und in welchem Umfang. Ziel dieses Beitrags ist es, empirische Belege zur Relevanz der UML zusammenzutragen und daraus Schlussfolgerungen für mögliche Kompetenzziele in der Softwaretechnikausbildung zu ziehen. Unser Ziel ist es, eine aktuelle Sicht auf die praktische Relevanz der UML zu liefern und damit die Diskussion über die Ausgestaltung der Lehre im Fach Softwaretechnik anzuregen

---

Formale Methoden in der Softwaretechnik-Vorlesung

Bernd Westphal, Uni Freiburg

Dieser Artikel stellt die Ergänzung einer Einführung in die Softwaretechnik um Formale Methoden im Bereich Requirements Engineering, SoftwareModellierung und Qualitätssicherung vor. Wir beschreiben die Konstruktion der Veranstaltung und berichten empirische Daten und Erfahrungen aus 4 Jahren der Durchführung. Ein Schwerpunkt ist die Beschreibung der verwendeten didaktischen Methoden zur Vermittlung der neuen Inhalte.

Stager: Simplifying the Manual Assessment of Programming Exercises

Christopher Laß, Stephan Krusche, Nadine von Frankenberg und Bernd Bruegge, TU München

Assessing programming exercises requires time and effort from instructors, especially in large courses with many students. Automated assessment systems reduce the effort, but impose a certain solution through test cases. This can limit the creativity of students and lead to a reduced learning experience. To verify code quality or evaluate creative programming tasks, the manual review of code submissions is necessary. However, the process of downloading the students' code, identifying their contributions, and assessing their solution can require many repetitive manual steps.

In this paper, we present Stager, a tool designed to support code reviewers by reducing the time to prepare and conduct manual assessments. Stager downloads multiple submissions and adds the student's name to the corresponding folder and project, so that reviewers can better distinguish between different submissions. It filters out late submissions and applies coding style standards to prevent white space related issues. Stager combines all changes of one student

into a single commit, so that reviewers can identify the student's solution more quickly.

Stager is an open source, programming language agnostic tool with an automated build pipeline for cross-platform executables. It can be used for a variety of computer science courses. We used Stager in a software engineering undergraduate course with 1600 students and 45 teaching assistants in three separate programming exercises. We found that Stager improves the code correction experience and reduces the overall assessment effort.

13:00 Mittagspause

14:00 Keynote – SEUH und TAV- Haus T Raum 002

Meine Lehren aus 10 Jahren Forschung und Lehre als Säulen unseres Firmenwachstums

Elmar Jürgens, CQSE GmbH, Garching b. München

Die Firma, die ich mitgegründet habe, ist vor 10 Jahren als Spin-Off unserer Forschungsarbeiten zu Software-Qualität an der TU München entstanden. Seitdem sind wir auf 35 Mitarbeiter gewachsen, von denen die Hälfte ihre Promotion im Bereich SoftwareEngineering gemacht hat. Aus unseren Forschungsprototypen ist in dieser Zeit ein kommerzielles Analysewerkzeug entstanden, das täglich von Entwicklern und Testern auf der ganzen Welt eingesetzt wird.

Forschung und Lehre waren dabei nicht nur der Auslöser unserer Gründung, sondern spielen auch weiterhin eine zentrale Rolle für Innovationen und Wachstum. In dieser Keynote möchte ich meine zentralen Erfahrungen und Überraschungen aus 10 Jahren Forschung, Lehre und Firmenaufbau

---

15:00 Kaffepause

15:15 Session 2 – SEUH- Haus T Raum 002

Software-Projektmanagement lernen ohne Projekt

Mario Winter, TH Köln

Das Modul "Projektmanagement" (PM) wird seit dem Studienjahr 2003 an der Fakultät Informatik und Ingenieurwissenschaften der TH Köln (bis 09/2015 FH Köln) für alle Informatik-Studiengänge angeboten. Besondere Herausforderungen sind die seit 2012 zunehmend hohen Studierendenzahlen und die inhaltlichen und strukturellen Unterschiede der Curricula der vier beteiligten Bachelor-Studiengänge. Zudem lassen es unterschiedliche Gründe nicht zu, das PM unmittelbar auf ein reales (studentisches Lehr-) Software-Entwicklungsprojekt angewendet wird.

Dieser Beitrag beschreibt die Entwicklung des grundsätzlichen Lehr-/ Lern-Arrangements von PM hin zu einer kompetenzorientierten "TeamTeaching"-

Veranstaltung für alle Studiengänge, den didaktischen Hintergrund und die Lehr-Erfahrungen sowie -Aufwände in der PM-Lehre mit über 350 Studierenden.

Future Skills: Algorithmisches Denken in allen Rollen von Softwareprojekten

Gudrun Socher, Sarah Ottinger, Veronika Thurner und Ralph Berchtenbreiter, HS München

Mit der zunehmenden Digitalisierung unserer Gesellschaft durchdringen Softwaresysteme nahezu alle Bereiche des privaten und beruflichen Lebens. Damit diese Softwaresysteme bedarfsgerecht konzipiert und erfolgreich umgesetzt werden bedarf es einer intensiven Zusammenarbeit der Schlüsselrollen in Softwareprojekten, insbesondere von den Rollen Product Owner, User Experience Designer und Software Engineer. Diese Zusammenarbeit von meist unterschiedlich IT-affinen Personen erfordert entsprechende Kompetenzen der Beteiligten, die auch als Future Skills bezeichnet werden. Für alle Beteiligten in Softwareprojekten gleichermaßen wichtig ist insbesondere die Stärkung des algorithmischen Denkens.

Wir beschreiben in diesem Artikel ein interdisziplinäres Lehr-Lernprojekt, bei dem wir in fachlich gemischten Studierendenteams sprachgestützte virtuelle Assistenten (Voice Apps z.B. für Amazon Alexa) bauen, um projekt-basiert algorithmisches Denken und Kollaborationsfähigkeit zu trainieren. Ein erster Kompetenztest überprüft die Wirksamkeit unseres Ansatzes.

Informatik ist nicht nur Programmieren – aber ohne Programmieren ist nichts Informatik

Oliver Radfelder, Karin Vosseberg, Ulrike Erb und Henrik Lipskoch, HS Bremerhaven

Ziel der Studieneingangsphase in den Bachelorstudiengängen Informatik und Wirtschaftsinformatik der Hochschule Bremerhaven ist, die Studierenden in eine Fachkultur einzuführen, in der Informatik nicht nur Programmieren ist – aber ohne Programmieren nichts Informatik ist. In kleinen Schritten werden die Studierenden an ihr grundlegendes Handwerkszeug und eine einfache Linux-basierte Infrastruktur für die Automatisierung von Prozessen herangeführt sowie das Arbeiten in Teams in einem ersten Projekt erprobt. Mit regelmäßigen kleinen Übungsaufgaben werden die verschiedenen Grundlagenfächer miteinander verzahnt. Über den Projektkontext wird ein Anwendungsbezug hergestellt. In dem vorliegenden Beitrag wird das Konzept der überarbeiteten Studieneingangsphase vorgestellt und anhand von kleinen Beispielen die Verzahnung von Modulen demonstriert. Im weiteren wird ein Beispiel für eine Lerneinheit aus den Workshops der Studieneingangsphase beschrieben.

Session 2 – TAV- Haus S Raum 414

Towards a Taxonomy for Applying Behavior-Driven Development

David Faragó, Mario Friske and Dehla Sokenou

Behavior-driven development (BDD) is a topic currently much talked about, especially in the agile community. Small scale examples of BDD suggest an intuitive and easy use, but experience shows that in practice, especially large projects, its application becomes elaborate and challenging. This paints an inconsistent picture about BDD. So, what are the requirements for a successful application of BDD?

We have identified, discussed, and classified the essential aspects of applying BDD. Depending on the application context, an aspect can speak for or against the use of BDD. As main contribution, we list these aspects and their pro and contra arguments.

Everyone can use these aspects to decide whether and how to use BDD in their individual project context.

GUI Testautomatisierung – 50% Erstellungsaufwände sparen, wenn man Anforderungen analysiert

Joerg Sievers

Wenn man heute Tests für Weboberflächen automatisieren möchte, ist die Antwort meistens Selenium. Es ist lizenzkostenfrei und weit verbreitet. Das bekannte Vorgehen, ein technisches Mittel anhand der Anforderungen zu wählen, wird einfach weggewischt, da der Lizenzkostenfaktor in den Vordergrund der Überlegungen gestellt wird. In einem Projekt haben wir verschiedene GUI Test-Automatisierungstools geprüft und ermittelt welches zu dem Entwicklungsvorgehen, den handelnden Personen und deren gestellten Aufgaben am besten passen könnte und kamen zu einem anderen Entschluss, der dem Projekt ca. 50% der Aufwände gegenüber Selenium spart und dazu geführt hat, das das Tool nun unternehmensweit Anklang gefunden hat.

19:00 Abendessen – SEUH und TAV

Apollo, Georgstr. 73, 27570 Bremerhaven

Um 18:15 gehen wir begleitet mit dem SmartCityWalk zum Apollo. Das Apollo war legendär in Bremerhaven und über Jahrzehnte Inbegriff für großes Kino. Seit 2018 ist das Apollo ein multimediales Kultur- und Veranstaltungszentrum im Herzen von Bremerhaven Geestemünde, in dessen Räumlichkeiten wir den Abend ausklingen lassen.

Der Fußweg dauert ca 25 Minuten. Alternativ können die Buslinien 505 bis zum Konrad-Adenauer-Platz oder die 502 bis zur Schillerstraße genommen werden. Die Buslinien fahren über den Hauptbahnhof.

Freitag, 22. Februar 2019

9:00 Keynote – SEUH und TAV - Haus T Raum 002

Test Architects at Siemens

Peter Zimmerer, Siemens AG, München

At Siemens we have invented and defined a new key role Test Architect to meet the diverse challenges of shorter time-to-market, increasing complexity and more agility while keeping quality and other key system properties high. In the real world our test systems increase in size, volume, flexibility, velocity, complexity and unpredictability: think about testing of autonomous systems or testing of AI systems (artificial intelligence). Additionally, digitalization (virtualization, cloud, mobile, big data, data analytics, internet of things, continuous delivery, DevOps) requires more than just a face lift in testing.

This talk shares our motivations, decisions, achievements, and experiences on our journey since 2016 to establish this new key role Test Architect on eye level with the software architects within the company.

10:00 Kaffeepause

10:15 Session 3 – SEUH - Haus T Raum 002

Using Mini-Projects to Teach Empirical Software Engineering

Michael Felderer und Marco Kuhrmann, Uni Insbruck & TU Clausthal

Empirical studies have become a central element of software engineering research and practice. Yet, teaching the instruments of empirical software engineering is challenging, since students need to understand the theory of the scientific method and also have to develop an understanding of the application of those instruments and their benefits. In this paper, we present and evaluate an approach to teach empirical software engineering with course-integrated mini-projects. In mini-projects, students conduct small empirical studies, e.g., surveys, literature reviews, controlled experiments, and data mining studies in collaborating teams. We present the approach through two implementations at two universities as a self-contained course on empirical software engineering and as part of an advanced software engineering course; with 101 graduate students in total. Our evaluation shows a positive learning experience and an improved understanding of the concepts taught. More than a half of the students consider empirical studies helpful for their later careers. Finally, a qualitative coding and a statistical analysis showed the proposed approach beneficial, but also revealed challenges of the scientific work process, e.g., data collection activities that were underestimated by the students.

Kurzbeiträge:

Experience Report on An Inquiry-Based Course on Model Checking

Sebastian Krings, Philipp Körner und Joshua Schmidt, Uni Düsseldorf & HS Niederrhein

The development and improvement of model checkers for the validation of hard- and software is an ongoing research topic in computer science. Model checking research connects theoretical and practical aspects; new algorithms are often implemented inside well-known model checkers which have been in development for many years. This is seldom taken into account by university courses on the topic, which often remain on the theoretical level. Hence, they do not provide practical access to the topic for reasons such as lack of time or inaccessibility of tools and their source code. In this article, we present a recent revision of our course on model checking, shifting from a classical lecture-based format to inquiry and research-based teaching. We document course development, present some didactic methods used and evaluate the course based on peer review and student feedback. Furthermore, we try to assert student engagement empirically.

Evaluation des Arbeitsprozess-integrierten Projektstudiums

Frank Fuchs-Kittowski, Juliane Siegeris und Jörn Freiheit, HTW Berlin

Für das Arbeitsprozess-integrierte Projektstudium im berufsbegleitenden Masterstudiengang „Professional IT-Business" der Hochschule für Technik und Wirtschaft Berlin wurde eine Evaluation durchgeführt. Mithilfe einer Umfrage unter den Studierenden sollte die Wirksamkeit des Konzepts des Arbeitsprozess-integrierten Projektstudiums überprüft werden. Es wurden acht Hypothesen aufgestellt. Die Überprüfung dieser Hypothesen sollte dazu dienen, mögliche Fehlannahmen und Brüche im Konzept zu erkennen. An der Umfrage haben 20 der 31 Studierenden teilgenommen, die das Projektstudium bisher absolviert haben. Die statistische Relevanz ist somit begrenzt, jedoch erlaubt die deskriptive Analyse der Umfrageergebnisse Rückschlüsse auf Änderungs- und Verbesserungspotenziale des dem Projektstudium zugrundeliegenden didaktischen Konzepts. Neben der Beschreibung der Methodik und der erzielten Ergebnisse werden Schlussfolgerungen aus der Auswertung gezogen sowie Ausblicke auf zukünftige Anwendungen des Arbeitsprozess-integrierten Projektstudiums gegeben.

Lernzieldefinitionen für Software Engineering - The Good, the Bad and the Ugly

Axel Böttcher, Veronika Thurner and Olga Nikoliai, HS München

Spätestens seit der Bologna-Reform sind wir als Lehrende immer wieder dazu angehalten, Modulbeschreibungen zu verfassen oder zu überarbeiten und in diesen kompetenzorientierte Lernziele zu definieren.

Um die Qualitätssicherung von Lernzieldefinitionen zu unterstützen hatten wir zunächst die Absicht, ein Softwarewerkzeug zur automatisierten Qualitätsprüfung von Lernzieldefinitionen zu realisieren. Dabei haben wir schnell festgestellt,

dass vor einer Automatisierung noch viele offene Fragen auf konzeptueller Ebene zu klären sind, hinsichtlich Kriterien und Indikatoren für die Qualität von Lernzieldefinitionen. Mit dieser Arbeit wollen wir eine Diskussion der Erkenntnisse anstoßen, die wir bei der Untersuchung dieser Fragen gewonnenen haben.

Ausgehend von der Beobachtung, dass unsere Curricula aus den verschiedenen Perspektiven von Lehrenden, Lernenden, Akkreditierungsagenturen sowie Arbeitgebern betrachtet werden, formulieren wir zunächst Qualitätskriterien für kompetenzorientierte Lernzielbeschreibungen.

Auf Basis dieser Kriterien haben wir systematisch Beschreibungen für Softwarebezogene Module analysiert. Anhand von diesen empirischen Befunden identifizieren wir Indikatoren für positive (good) und weniger positive (ugly bis bad) Lernzielbeschreibungen. Diese abstrahieren wir anschließend zu allgemeineren Kriterien, benennen das jeweils zugrundeliegende Prinzip und gewichten diese Kriterien sodann nach ihrer Bedeutung für die Qualität einer Lernziedefinition.

Gute Lernzieldefinitionen beschreiben das gewünschte Lernergebnis kompetenzorientiert, d. h. als eine Kombination aus Inhalt und Handlungskomponente, die in Form von beobachtbarem Verhalten in Studierenden-zentrierter Form formuliert ist. Sie sind dabei in ihrer Bedeutung verständlich, eindeutig und auf den Punkt.

Eine erste empirische Analyse von bestehenden Modulbeschreibungen zeigt, dass diesbezüglich noch vielfältiges Verbesserungspotenzial vorhanden ist.

Session 3 – TAV - Haus S Raum 414

Modellbasierte Testdatenspezifikation und -generierung mittels Äquivalenzklassen und SQL

Mario Friske und Dierk Ehmke

Für automatisierte Softwaretests werden komplexe Testdaten benötigt. Deren Erstellung ist aufwendig, deshalb werden Verfahren zur effizienten Testdatenspezifikation und -generierung benötigt. In diesem Beitrag stellen wir zwei entsprechende Ansätze vor und diskutieren, wie diese gewinnbringend kombiniert werden können.

Zunächst stellen wir ein modellbasiertes Vorgehen vor, welches es ermöglicht, automatisch generierte logische Testfälle on-the-fly mit kompatiblen Testdatensätzen zu befüllen. Der skizzierte Ansatz basiert auf einer interaktiven Zuordnung existierender Datensätze als typische Repräsentanten der jeweiligen Äquivalenzklasse auf Modellebene.

Anschließend präsentieren wir einen zweiten Ansatz, der auf Modellen in SQL-Notation aufsetzt. Wesentliche Teile von SQL sind deklarativ. Basierend auf diesem Modell werden mit Angaben aus abstrakten Testfällen konkrete automatisierte Testfälle inklusive Testdaten und Testorakel erzeugt. Im Zentrum steht der Tester als technisch versierter Experte, der das Tester-Modell formuliert und kommuniziert.

Automatisierte Erfassung von Nutzungsdaten mobiler Apps zur Verbesserung der App-Qualität – Ein Erfahrungsbericht
Frank Elberzhager, Simon Andre Scherr, Britta Karn and Thomas Immich

"Was haben wir denn jetzt verbockt?" Vor dieser Frage stehen App-Entwickler immer wieder. Nutzer löschen Apps sofort, wenn sie nicht so funktionieren wie erwartet. Typisch für Apps sind kurze Time-to-Market, ständige Updates und geringe Budgets.

Was nun? Die gute Nachricht ist: Nutzer geben Feedback und daraus kann man wertvolle Schlüsse ziehen! Mit Opti4Apps haben wir einen Prozess zur ganzheitlichen Integration von Feedback in die Entwicklung geschaffen. In diesem Beitrag möchten wir das Gesamtkonzept vorstellen sowie Erkenntnisse aus der Studie zur automatisierten Erhebung von Nutzungsfeedback darstellen und zeigen, wie Verbesserungen dem Feedback abgeleitet werden konnte.

Kurzberichte und Treffen der TAV-Arbeitskreise:

Modell-basiertes Testen - Haus S Raum 412
Testmanagement - Haus S Raum 413
Berufsbilder / Ausbildung im QS-Bereich - Haus S Raum 415
Testen & KI - Haus S Raum 414

12:15 Mittagspause

13:00 Keynote – SEUH - Haus T Raum 002

Agile is real - Alltag in einem XP Office
Eleonora Scherl, Volkswagen AG

In einer sich schnell wandelnden Welt werden agile Methoden im Software Engineering allein schon aus wirtschaftlichen Gründen immer interessanter und nehmen auch in den Lehrplänen mehr Platz ein. In dem Vortrag erhalten Sie Einblick in ein Software Product Office, welches vollumfänglich mit schlanken (Lean Startup) und agilen (XP) Methoden arbeitet. Erfahren Sie wie, was und warum wir denken, dass dies das 'State of the Artx' Vorgehen für modernes Software Engineering ist.

Arbeitskreise – TAV - Haus S Räume 412-415

Fortführung der Arbeitskreise

14:00 Session 4 – SEUH - Haus T Raum 002

Teaching Rationale Management in Agile Project Courses
Anja Kleebaum, Jan Ole Johanssen, Barbara Paech und Bernd Bruegge, Uni Heidelberg & TU München

Rationale management is beneficial since it supports decision-making and prevents knowledge vaporization. To apply rationale management, developers need to know how to systematically capture rationale and how to exploit the documentation. We believe that teaching these skills to students further integrates rationale management into the daily work of developers and has positive effects on both the software development process and on the quality of the software. In this paper, we report on a lecture on teaching rationale management to students. In this lecture, students are introduced to a rationale model, to capture and exploitation methods, and tool support for rationale management. The goal is to motivate them to apply rationale management. We present the students' results as well as their attitude and feedback towards the applied methods. Further, we sketch how rationale management will be applied during the semester.

A Syllabus for Usability Engineering in Multi-Project Courses
Jan Ole Johanssen, Dominic Henze and Bernd Bruegge, TU München

Usability engineering is an important activity in today's application development, which raises the need to focus on its teaching efforts. However, in contrast to theoretical concepts, learning usability engineering is most successful in a hands-on environment. Furthermore, a challenge exists in efficiently applying usability assessment techniques to carry out usability engineering over time. We developed the UE4MP syllabus, a four meeting-based teaching approach for agile multi-project courses using cross-functional teams. The syllabus enables students to gain usability engineering knowledge and experience through hands-on learning and to make use of a platform for usability engineering. The overall goal is to relate usability concepts closely to applications that are developed by the students themselves. We applied the UE4MP syllabus over the course of one semester and report on our observations and challenges.

Arbeitskreise – TAV - Haus S Räume 412-415

Fortführung der Arbeitskreise bis 15:30 Uhr

15:00 Verabschiedung – SEUH - Haus T Raum 002

15:30 Plenum – TAV - Haus S Raume 414

Kurzberichte der Arbeitskreise

16:00 Verabschiedung – TAV - Haus S Raume 414

Programm- und Organisationskommitee der SEUH

Ruth Breu, Uni Innsbruck
Bernd Brügge, TU München
Anna Hauptmann, HS Dresden
Maritta Heisel, Uni Duisburg-Essen
Marco Kuhrmann, TU Clausthal
Dieter Landes, HS Coburg
Barbara Paech, Uni Heidelberg
Oliver Radfelder, HS Bremerhaven (Org)
Kurt Schneider, Uni Hannover
Juliane Siegeris, HTW Berlin
Veronika Thurner, HS München (Vorsitz)
Karin Vosseberg, HS Bremerhaven (Org)

Programmzusammenstellung der TAV

Andrej Pietschker, Fachgruppenleiter TAV

Danksagung

Ein herzlicher Dank gebührt den Sponsoren, ohne deren Unterstützung die Durchführung der SEUH und der TAV in dieser From nicht möglich wäre:

German Testing Board e.V.
Merentis GmbH
dpunkt Verlag
ASQF e.V.
GI e.V.
Hochschule Bremerhaven
Hochschule München