TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Fakultät Informatik, Institut SMT, Professur Softwaretechnologie

Dr. Sebastian Götz

Einführung in das Softwarepraktikum WS 2021/22

Dresden, 11.10.2021

Folien von Dr. Birgit Demuth

Agenda

- Wozu gibt es das Softwarepraktikum?
- Was wird von Ihnen konkret erwartet?
- Wie sollen Sie das Team organisieren (Rollen im Team, Tutor, Arbeitsteilung)?
- Wie wird Ihr implementierter Code analysiert?
- Welche Erfahrungen gibt es mit Freundschaften in Teams?
- Welche Hilfen stehen zur Verfügung?
- Welche Projektphasen und Meilensteine gibt es?
- Warum ein disziplinierter Softwareentwicklungsprozess?
- Was sind die Bewertungskriterien im Softwarepraktikum?

Wozu gibt es das Softwarepraktikum?

Ziele der Lehrveranstaltung
- Erlernen von Professionalität in der Softwareentwicklung
- Vorbereitung auf das weitere Studium und das Berufsleben

Praxisnähe in der Softwareentwicklung
- (Simulation von) echte(n) Kunden und echte(n) Anwendungen
- Kundengespräche
- Kundenorientiertes Denken
- Große Software
- Professionelle Dokumentation
- Harte Termine
- Professioneller Werkzeugeinsatz
- Kampf mit unvorhergesehenen Problemen (technische, Kundenwünsche)
- Auseinandersetzung mit Teamproblemen

Soziale Kompetenzen und Fähigkeiten

Erwartungen der Wirtschaft an Hochschulabsolventen

(aus der Umfrage der IHK Dresden)

- Einsatzbereitschaft
- Verantwortungsbewußtsein
- Teamfähigkeit und Kooperationsfähigkeit
- Kommunikationsfähigkeit
- Konfliktfähigkeit
- Kritikfähigkeit
- Führungskompetenz
- Interkulturelle Kompetenz

Was wird von Ihnen konkret erwartet?

- Professionelle Softwareentwicklung mit
  - CRC-Karten-Methode
  - Modellierung mit UML in OOA und OOD (mit einem UML-Modellierungstool)
  - Prototyping
  - Arbeit mit Junit mit testgetriebener Entwicklung
  - Wiederverwendung (SalesPoint, weitere Frameworks)
  - Versionsmanagementsystem (Git)
  - GitHub als Plattform für das gesamte Softwareprojekt
  - Projektmanagement
- JEDES Teammitglied muss implementieren (einschl. eigener Prototypen)!
- Zwischen-/Abschlusspräsentation
- Effektive Teamarbeit
  - 6 Mitglieder organisieren sich nach Scrum-Prinzipien
  - Erfolg des Praktikums ist abhängig von der Motivation und der aktiven Beteiligung ALLER Teammitglieder

Warum ein disziplinierter Softwareentwicklungsprozess?

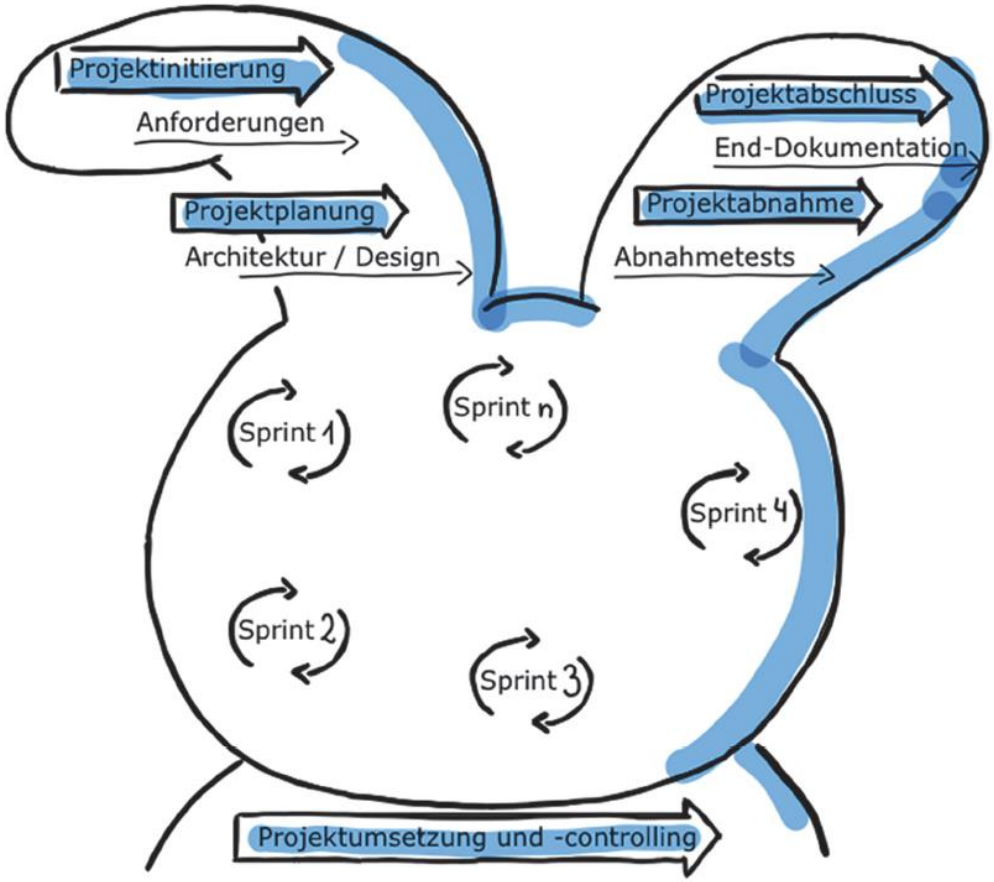
- Erste Erfahrungen mit (professionellen) SE-Prozessen
- Erfüllung von Meilensteinen (Meilenstein nach jeder Phase)
- **Studentensyndrom** (Erfahrung im Projektmanagement)
  - entspricht der Tendenz einer Person, sich erst dann richtig auf eine Aufgabe zu konzentrieren, wenn der Liefertermin in Gefahr ist (mit allen negativen Konsequenzen ☹).

- **Hybride Softwareentwicklung („Angsthasenmodell“)**
  - → Kombination der Vorzüge von

–agilen Methoden (Scrum) und

–schwergewichtigen Methoden (Wasserfall)

„Angsthasenmodell“ bzw. „Water-Scrum-Modell“



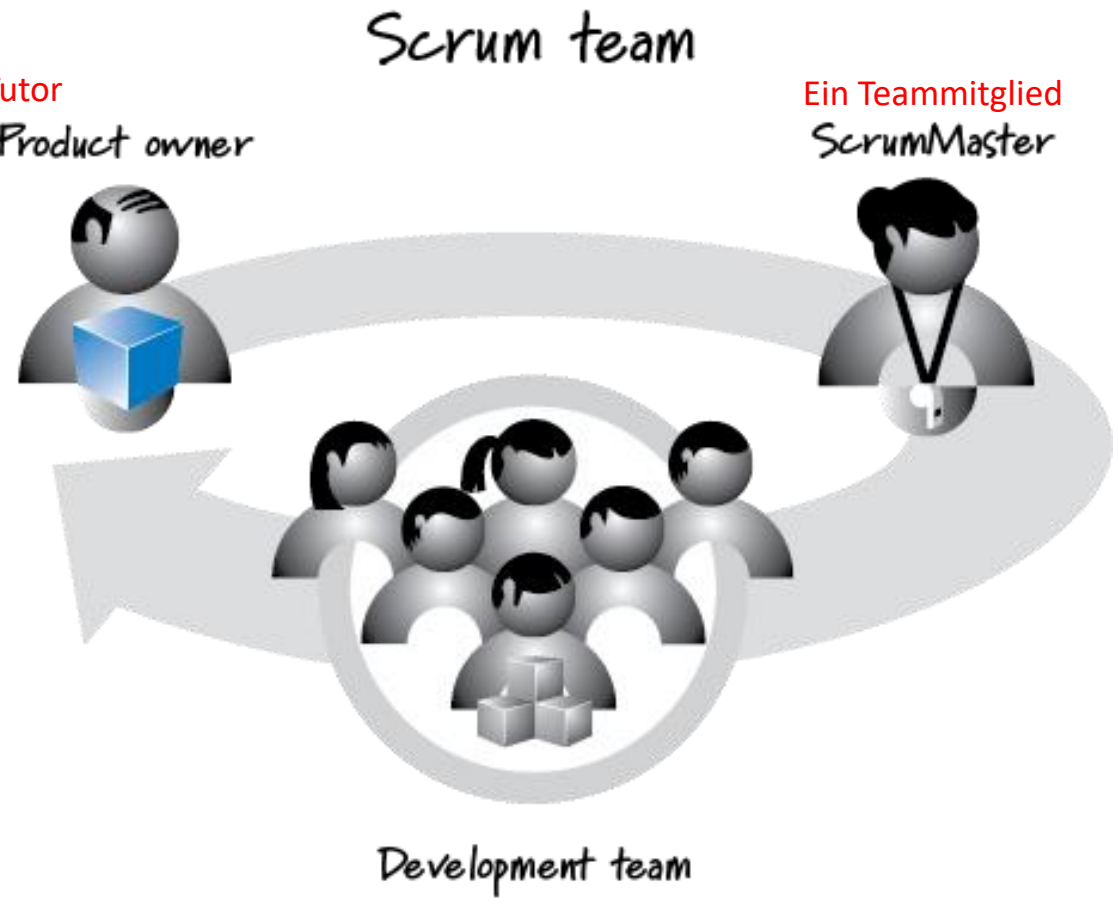
Agiles Vorgehen eingebettet in klassische Projektphasen

Was ist Scrum?

- Leichtgewichtiges Vorgehensmodell im Rahmen der agilen Softwareentwicklung
- SCRUM ist sehr beliebt meist aber angepasst auf die konkreten Projektbedingungen (hybrides Modell)
- Iteratives Vorgehen mit ständiger Kontrolle
- Sprint Planning Meetings und Daily Scrum Meetings
- Wenig Rollen
- Teams organisieren ihren Tagesablauf selbst
- Produkteigenschaften werden im **Product Backlog** festgeschrieben
- Das Team hält seine Aufgaben in einem **Backlog Tasks** fest
- Eigenschaften/Anforderungen können neupriorisiert werden
- **Wir passen den Scrum-Ansatz an die Rahmenbedingungen des Softwarepraktikums an und folgen durch Festlegung von Meilensteinen einem hybriden Ansatz in der Softwareentwicklung**

Rollen im Team

Stakeholder sind externe Kunden und außerhalb des Scrum Teams



Tutor/in

- hat zwei Rollen: Kunde (nur internes Praktikum) und Product Owner
- diskutiert und erstellt mit dem Team die Anforderungen an das Produkt
- priorisiert und erläutert die zu entwickelnden Produkteigenschaften
- beurteilt, welche Eigenschaften am Ende eines Sprints fertiggestellt wurden
- verwendet das **Product Backlog** (bei uns das Pflichtenheft)
- während des Entwicklungsprozesses ist er auch für das **Product Backlog Refinement** verantwortlich, in dem er ggfs. Verfeinerungen im Product Backlog fordert.

Für externe Projekte (Gruppen 44-46) gilt zusätzlich:

- hält zusammen mit dem Entwicklungsteam regelmäßig Rücksprache mit den Stakeholdern (externe Kunden), um deren Bedürfnisse und Wünsche zu verstehen

Scrum Master

- ist dafür verantwortlich, dass die Teamarbeit gelingt
- arbeitet mit dem Entwicklungsteam zusammen
- ist in unserem Praktikum selbst Mitglied des Entwicklungsteams
- moderiert interne Treffen (außerhalb der Pflichtkonsultation)
- ist verantwortlich für die Erstellung des Protokolls für die Pflichtkonsultation (siehe *Template*)
- kümmert sich um die Behebung von Störungen
- kann Teammitglieder disziplinarisch nicht belangen
- dient als Ansprechpartner für sein Team gegenüber den Lehrbeauftragten

- Die Rolle des Scrum Masters kann während des Softwarepraktikums ggfs. einem anderen Teammitglied zugeordnet werden.

Development Team

- ist für die Lieferung der Produktfunktionalitäten in der vom Product Owner (Tutor) gewünschten Reihenfolge verantwortlich
- trägt die Verantwortung für die Einhaltung der vereinbarten Qualitätsstandards
- organisiert sich selbst

- Das ideale **Teammitglied** ist sowohl Spezialist als auch Generalist, damit es Teamkollegen beim Erreichen des gemeinsamen Ziels helfen kann.

Tutor/in als Coach

- dient als Coach für die gesamte Softwareentwicklung
- moderiert die Pflichtkonsultationen
- hilft bei der Behebung von Störungen und Hindernissen in der Softwareentwicklung
- gibt dem Entwicklungsteam regelmäßig Feedback zum SE-Prozess
- bewertet die Ergebnisse bzw. Artefakte der Teammitglieder vor und kann Verwarnungen aussprechen

Welche Projektphasen gibt es?

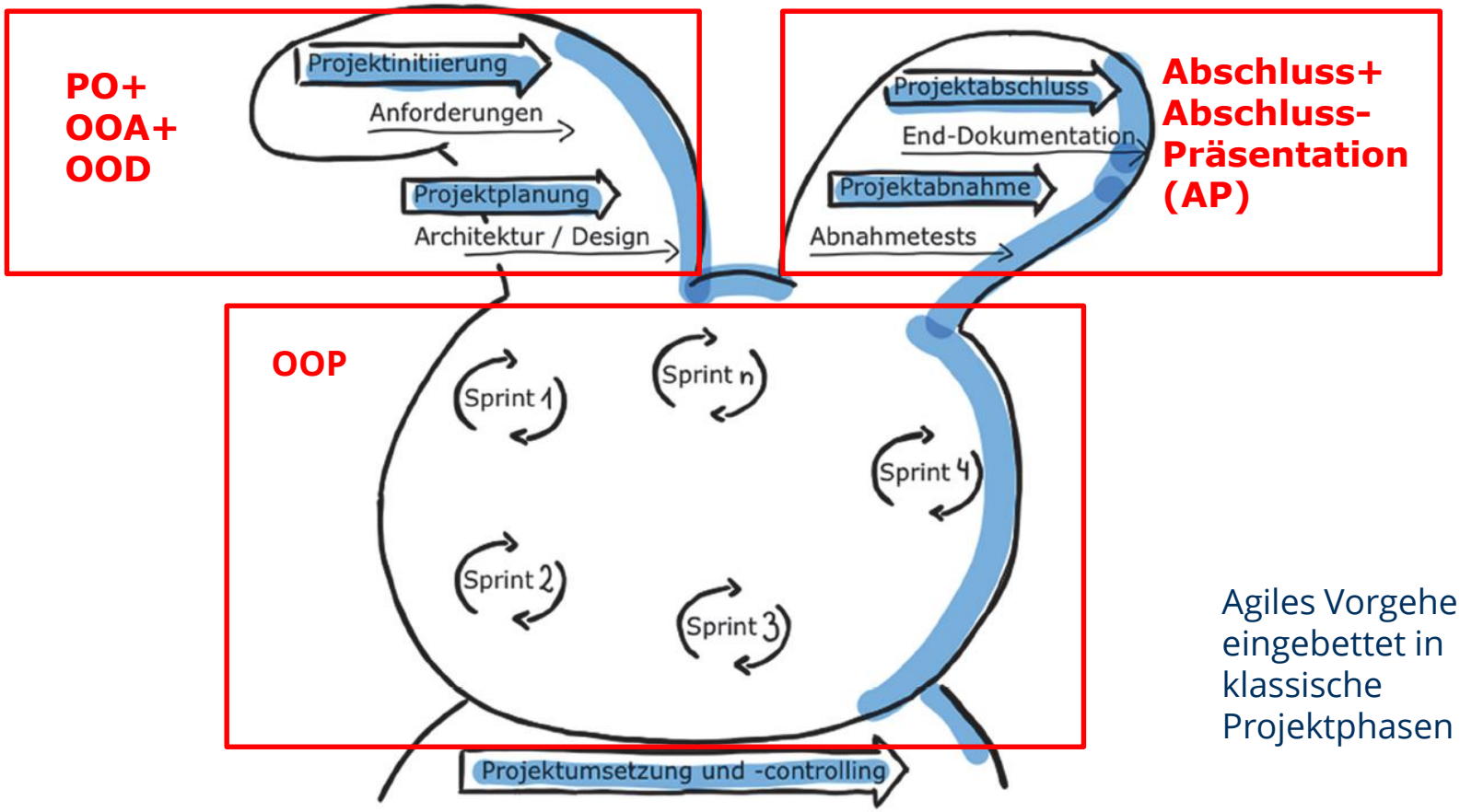
Projektlaufzeit insgesamt 12 Wochen (einschließlich dieser Woche)

Wöchentliche Sprints
- insgesamt 9 Sprints (2 für Prototypen, 7 für eigentliche Anwendung)
- Bewertung ist Gegenstand der wöchentlichen Pflichtkonsultationen

Sechs Meilensteine (OOA, OOD, OOP_I, OOP_II, OOP_III, OOP_IV)
- Die Meilensteine erwarten u.a. jeweils eine getestete und lauffähige Anwendung bzw. einen lauffähigen Prototypen.
- Die Anwendungen werden in den OOP-Phasen durch Continuous Integration und Sonarqube einer automatischen Qualitätskontrolle unterzogen.
- **Jeder Meilenstein muss erfolgreich absolviert werden, um am weiteren Softwareentwicklungsprozess und damit am Praktikum teilnehmen zu können.**

➢**Fertigstellung des Projektes am Sonntag, den 23. Januar 2022 (harte Deadline!)**

„Angsthasenmodell“ bzw. „Water-Scrum-Modell“



Agiles Vorgehen eingebettet in klassische Projektphasen

Arbeitsteilung im Team

- Einarbeitung in Spring und SalesPoint: JEDER implementiert kleine Prototypen
- **Analyse GEMEINSAM** im Team
- **Entwurf GEMEINSAM** im Team
  - **JEDER implementiert** einen Prototypen für eine Anwendungskomponente
  - **Experimentelles oder vertikales Prototyping** (Experimente mit SalesPoint bzw. anderen Frameworks; es ist nicht gedacht, den Prototypen weiter zu verwenden)
- Implementierung und Test in ARBEITSTEILUNG
  - **Vertikale** Arbeitsteilung („Durchstich“ im System, Teilfunktion des Systems) empfohlen!
  - Regelmäßiges Einchecken des Codes ins GitHub Repository
  - JEDER Programmierer schreibt für „seine Klassen“ zuerst die (junit-)Tests und implementiert dann die zugehörige Klasse

Arten von Prototyping (Exkurs)

Experimentelles Prototyping
- Ziel: Sammeln von Erfahrungen mit dem Prototyp
- Ergebnis: ein erster experimenteller Prototyp

Vertikales Prototyping (Durchstich)
- Ziel: Die Entwicklung eines funktionalen Ausschnitts eines Programmes
- Ergebnis: Ein Teil des Systems durch alle Ebenen hindurch implementiert

Horizontales Prototyping (z. B. GUI)
- Ziel: Eine funktionierende Ebene, an der sich andere Ebenen orientieren
- Ergebnis: Eine ausgewählte Ebene des Gesamtsystems ist fertiggestellt.

Evolutionäres Prototyping
- Ziel: Akzeptanz beim Nutzer zu überprüfen
- Ergebnis: Ein Programm mit den Grundfunktionalitäten

Exploratives Prototyping
- Ziel: nachweisen, dass Spezifikationen oder Ideen tauglich sind
- Ergebnis: Anforderungsspezifikation

Projektorganisation und Einarbeitung (PO)

Woche (1)	Aktivitäten	Meilenstein
11.10.-17.10.	– Teamarbeit organisieren – Einarbeitung in GitHub-zentrierte SW-Entwicklung ▪ Java-Tooling (Wdhlg.) ▪ Git und GitHub (Wdhlg.) – **Guestbook Erweiterung**	

Analyse (OOA)

Woche (2)	Aktivitäten	Meilenstein

18.10.-24.10.	▪ Analysemeeting im Team mit
	  ▪ Anforderungen erfassen/diskutieren
	  ▪ Analysemodell erstellen (Kontextdiagramm, Top-Level-Architektur, Anwendungsfall-, Klassen-, Sequenzdiagramme)
	  ▪ GUI-Entwurf
	  ▪ Akzeptanztestfälle
	▪ Einarbeitung in das SalesPoint-Framework
	▪ **Videoshop-Erweiterung** beginnen!
	▪ Siehe https://github.com/st-tu-dresden/videoshop	

Analyse (OOA)

Woche (3)	Aktivitäten	Meilenstein
25.10-31.10.	▪ Zusammenstellung der Modelle im Pflichtenheft ▪ Konsolidierung des Pflichtenheftes ▪ Finalisierung der Videoshop-Erweiterung	**OOA (31.10.)** I. Pflichtenheft (→ ***Template***) II. Erweiterung des Videoshops (pro Teammitglied)

Pflichtenheft des Videoshops unter
https://github.com/st-tu-dresden/videoshop/blob/master/src/main/asciidoc/Pflichtenheft.adoc

Entwurf und Prototyping (OOD)

Woche (4)	Aktivitäten	Meilenstein
01.11.-07.11.	▪ Anpassung des Analysemodells an das SalesPoint-Framework (DDD) ▪ Grobentwurf (Architektur, Persistenz, GUI) ▪ Verteilung der Komponenten (Packages) an die Teammitglieder ▪ Jedes Teammitglied beginnt für „seine“ Komponente, einen Anwendungsprototypen zu implementieren	

Entwurf und Prototyping (OOD)

Woche (5)	Aktivitäten	Meilenstein
08.11.-14.11.	▪ Fertigstellung der Anwendungsprototypen ▪ Ggfs. Überarbeitung des Grobentwurfs ▪ Entwurfsentscheidungen ▪ Verfeinerung der Entwurfsmodelle ▪ Zusammenstellung der Entwurfsmodelle in der Entwicklerdokumentation ▪ Konsolidierung der Entwicklerdokumentation	**OOD (13.11.)** I. Anwendungsproto-typ II. Testplan (verfeinerte Akzeptanztestfälle, → ***Template***) III. Entwicklerdoku-mentation v1(→ ***Template***)

Entwicklerdokumentation des Videoshops unter
https://github.com/st-tu-dresden/videoshop/
blob/master/src/main/asciidoc/developer_documentation.adoc

Implementierung und Test (OOP)

Woche (6 bis 10)	Aktivitäten	Meilenstein
(6) 15.11.-21.11.	• wöchentliche Implementierung entsprechend Protokoll und Backlog • Verteilung der Issues/Packages/ Klassen an die Teammitglieder • fortlaufende junit-Tests • fortlaufende Javadoc-Dokumentation • Cross-Testing in der Woche 10 → ***Template***	
(7) 22.11.-28.11.		**OOP_I (28.11.)** Basisfunktionalität
(8) 29.11.-05.12.		
(9) 06.12.-12.12.		**OOP_II (12.12.)** Muss-Kriterien (als Basis für das Cross-Testing)
(10) 13.12.-19.12.		**OOP_III (19.12.)** Ergebnisse des Cross-Testings

Implementierung und Test (OOP)

Woche (11 bis 12)	Aktivitäten	Meilenstein
(11) 10.01.-16.01.	• Kann-Kriterien • Realisierung weiterer Kundenwünsche • Stabilisierung der Anwendung → Bearbeitung des Cross Testing Feedbacks • Konsolidierung der Dokumentationen	
(12) 17.01.-23.01.		**OOP_IV (23.01.2021)** I. Fertige Anwendung II. Dokumentation III. Auswertung des Praktikums (persönlich durch jedes Teammitglied) → ***Template***
(13) 31.01.-04.02.	ABSCHLUSSPRÄSENTATIONEN (AP) Online-Fragebogen	

Auswertung und Bewertung des Praktikums (1)

- Online-Fragebogen
  - Wird am Ende des Praktikums zusammen mit dem Tutor ausgefüllt
  - Qualitative und quantitative Fragen

Ganz wichtig: von Anfang an **Arbeitsaufwände** jedes einzelnen Teammitgliedes genau protokollieren!

- Pro Student gesamte Stundenzahl (gemeinsam + individuell) pro Woche
- *Template* für Erfassung der Zeitaufwände wird bereitgestellt
- Für Gesamtauswertung durchschnittliche Gesamtstundenzahl pro Student im Team (am Ende)

- Bewertung jedes Teams kontinuierlich im Praktikum
  - durch den Tutor i.S. eines Feedbacks für das Team
  - unterstützt durch automatisierte Qualitätskontrolle durch Continuous Integration und Sonarqube

Hinweise zur Modellierung (OOA und OOD)

- Auf alle Fälle ein UML-Modellierungstool (statt nur ein Zeichentool) verwenden, zum Beispiel
  - Visual Paradigm
  - Papyrus UML (for Eclipse environment)
  - Astah UML
  - StarUML
  - [PlantUML (textual modeling tool)]

- OOA: genau EIN Modell
- OOD: genau EIN Modell
- EIN Modell bedeutet EIN Modellierungsprojekt, besteht typischerweise aus mehreren Diagrammen, keine „Tapeten“ erstellen!
- Und noch einmal: OOA- und OOD-Modellierung GEMEINSAM im Team!

Java-Regelüberprüfungen (Sonarqube)

vgl. CODING RULES FOR THE SOFTWARE PROJECT COURSE

- Blocker Rules
- Critical Rules
- Major Rules

WWW: OPAL Kurs -> Material

Welche Hilfen stehen zur Verfügung? (1)

- **WWW**-Seiten zum Softwarepraktikum
- LV Softwaretechnologie SS 21
- **Skripte zur Einarbeitung**
  - Web application development with Java and Spring: **OPAL**
- Technische **Infrastruktur** der TU Dresden
  - Eclipse unter Windows
  - Mailinglisten
  - Videokonferenzsysteme (insbesondere BBB)

Welche Hilfen stehen zur Verfügung? (2)

- Vorbereitetes privates **GitHub-Gruppenrepository** (basierend auf **Kickstart**)
  - Damit entfällt viel Einarbeitungs- und Setup-Aufwand

- SalesPoint v7.4
  - https://st.inf.tu-dresden.de/SalesPoint/

- SalesPoint/Webseite/Beispielanwendungen auf GitHub:
  - https://github.com/st-tu-dresden/salespoint
  - https://github.com/st-tu-dresden/guestbook
  - https://github.com/st-tu-dresden/videoshop

Welche Hilfen stehen zur Verfügung? (3)

Ihre Ansprechpartner

- Ihr(e) Praktikumsbetreuer(in) (Tutorin)
- Praktikumsforum nicht mehr im Auditorium sondern im OPAL Kurs
- Lehrstuhl Softwaretechnologie

▪Dr. Sebastian Götz (Lehrbeauftragter)

▪Martin Morgenstern (Stellvertreter und IT-Administrator)

Bewertungskriterien (1) – SE-Prozess

- Pflichtenheft
- Qualität der Modelle
  - In der Analysephase (OOA)
  - In der Entwurfsphase (OOD)
  - Konsistenz/Aktualität der Modelle
- Anwendung der CRC-Karten-Methode
- Benutzerschnittstellenentwurf
- Prototyping
- Wiederverwendung von Klassenbibliotheken/Frameworks
- Begründung von Entwurfsentscheidungen
- Testen (TDD, Test-Coverage)
- Forward Engineering
- Versionsmanagement mit Git und Arbeit mit GitHub

Hinweis:
Für jedes Bewertungskriterium gibt es die Schulnoten 1 bis 5

Bewertungskriterien (2) - Anwendung/Endprodukt

- Erfüllung des Pflichtenheftes
  - Musskriterien
  - Kannkriterien
- Funktionsumfang
- Zuverlässigkeit (Robustheit)
- Benutzbarkeit/Ästhetik/Verständlichkeit
- Wartbarkeit (Erfüllung zusätzlicher Kundenwunsch)
- Codequalität (Clean Code)
  - gemessen durch statische Codeanalyse (Sonarqube)
- Qualität javadoc
- Anwenderdokumentation

Bewertungskriterien (3) –Projektmanagement

- Planmäßigkeit der Entwicklung/Termintreue
- Protokolle der Treffen und Kontrolle der Einhaltung der Festlegungen
- Protokollierung der Arbeitsaufwände

Bewertungskriterien (4) – Teamarbeit

- Auftreten als Team nach außen
- Auftreten der Teammitglieder im Team
- klare/gerechte Aufgabenteilung
- Kommunikation mit dem Tutor
- Selbstkritische Einschätzung durch das Team
- Umgang mit Problemen und Konflikten

Bewertungskriterien (5) – Abschlusspräsentation

- Zeiteinhaltung
- Qualität des Vortrages
- Qualität der Vorführung der Anwendung
- Diskussion/Reaktion auf Fragen

Was gibt es konkret diese Woche zu tun (1)?

- Kennenlernen des Teams, der Tutorin und des Kunden
- Absprache des **wöchentlichen Termins** für die Pflichtkonsultation
- Jeder muss sich bei GitHub registrieren (sofern noch nicht erfolgt) und seinen GitHub-Namen dem Tutor mitteilen
- Festlegen der Rollen / (vorläufiger) Scrum Master
- Überlegen, wie die Arbeit organisiert werden soll
- Protokoll über heutiges erstes Gruppentreffen erstellen
  - unter Nutzung des Templates (protocol_template.adoc) in GitHub
- Praktikumsaufgabe gründlich lesen und im Groben verstehen
- Skripte auf static.olivergierke.de/lectures/ wiederholen bzw. neu durcharbeiten
- Einarbeitung in die Arbeit auf der GitHub Plattform
- Entwicklungsumgebung einrichten (OpenJDK 11, Git, Spring Tool Suite)

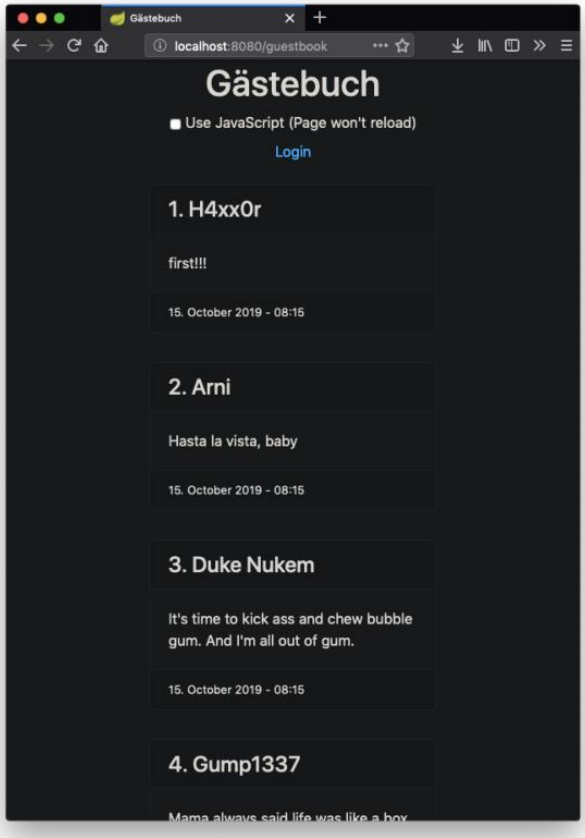
Was gibt es konkret diese Woche zu tun (2)? [gilt nur für alle Gruppen außer 44-46]

Guestbook Erweiterung

https://github.com/st-tu-dresden/guestbook

Jedes Teammitglied macht dies für sich:

- Guestbook Anwendung klonen
- In der eigenen Entwicklungsumgebung (IDE) das Guestbook Projekt aufsetzen
- Kleine Erweiterung schreiben
  - Empfehlung (einfach): Email-Adresse soll als zusätzliches Feld bei einem Gästebucheintrag abgefragt, validiert und angezeigt werden
  - Anspruchsvoller: Admin kann Beiträge bearbeiten
  - Eigene Vorschläge sind erwünscht, aber es müssen Änderungen sowohl im Frontend- als auch im Backend erfolgen



Teamzusammenstellung im WS 2021/22

▪**Zufällige/Selbstständige** Zusammenstellung innerhalb der folgenden Gruppen von Studierenden:

— Informationssystemtechnik / Nebenfach / (Medien)Informatik

— Zusammenstellung **unabhängig** von Vorleistungen

Aus der Auswertung der ikoso-Studie an der TU Dresden (SS 2004):

▪Empirische Forschung: Zumindest kurzfristig zeigen „Freundschaft-Teams“ bessere Leistungen als zufällig zusammengestellte Teams (Jehn & Shah, 1997)

▪Es hat sich aber gezeigt, dass die Leistungen sich über die Zeit angleichen.

▪In der Arbeitswelt ist es üblich, mit Personen zusammen zu arbeiten, die man zuvor nicht kennt (Praxisnähe).

Los geht‘s

Coming together is a beginning.

Keeping together is progress.

Working together is success.

Henry Ford (1863 – 1947)