TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut SMT, Professur Softwaretechnologie

Einführung in das Softwarepraktikum WS 2018/19

Dr. Birgit Demuth Dresden, 09.10.2018

Gliederung

- Wozu gibt es das aufwendige 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 so ein strenger SE-Prozess?
- Was sind die Bewertungskriterien im Softwarepraktikum?

Wozu gibt es das aufwendige 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
  - Testgetriebene Entwicklung mit Java
  - 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
  - 5-6 Mitglieder organisieren sich nach Scrum-Prinzipien
  - Erfolg des Praktikums ist abhängig von der Motivation und der aktiven Beteiligung ALLER Teammitglieder

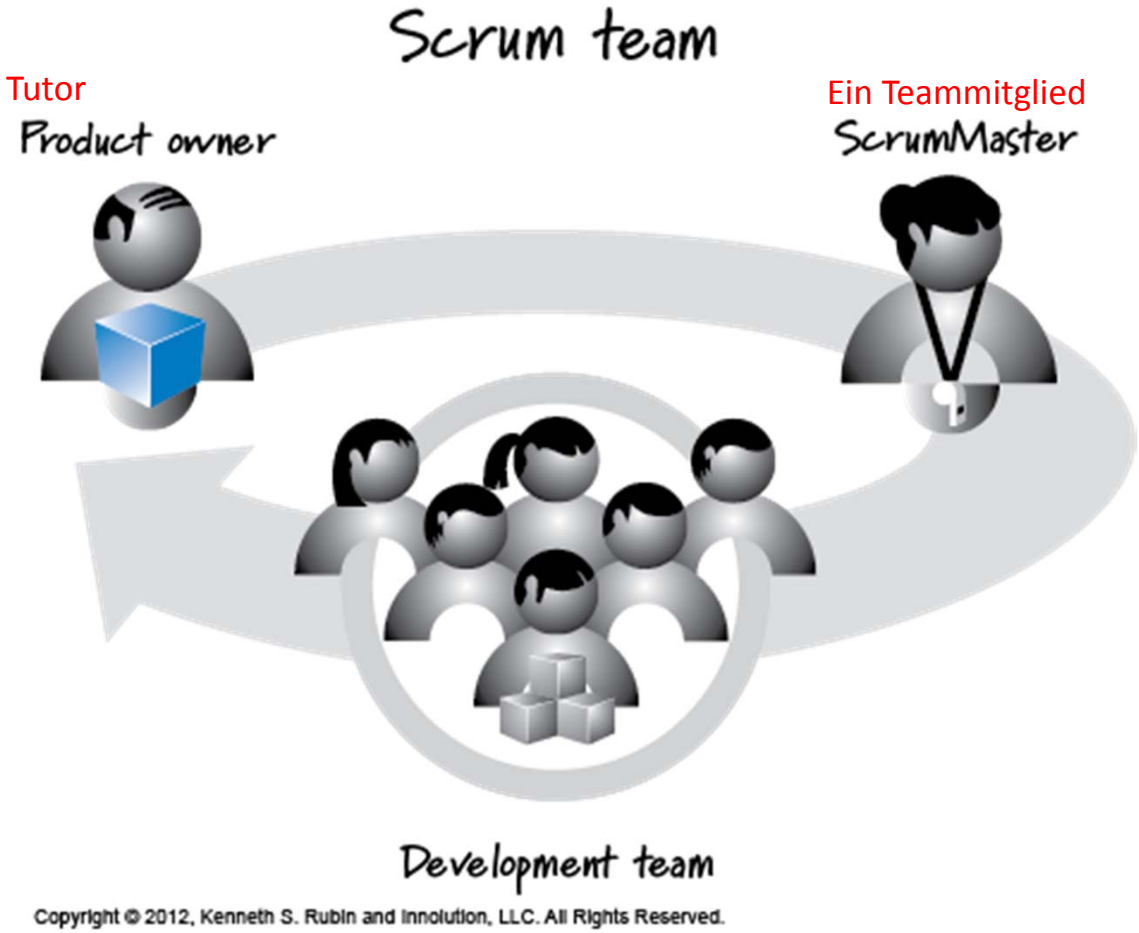
Was ist Scrum?

- Leichtgewichtiges Vorgehensmodell im Rahmen der agilen Softwareentwicklung
- SCRUM ist sehr beliebt, wird schätzungsweise heute in ca. 60% aller Firmen angewendet
- 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

- 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 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 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
- kann Teammitglieder disziplinarisch belangen (Verwarnungen aussprechen)

Arbeitsteilung im Team

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

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)

- Empfehlung: Magic Draw UML als Modellierungstool
- Alternative Tools sind möglich, aber auf alle Fälle ein UML-Modellierungstool verwenden (statt nur Zeichentool), zum Beispiel
  - Visual Paradigm
  - Papyrus UML (for Eclipse environment)
  - Astah UML
  - 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: Softwaretechnologie-Projekt → Ressourcen → Frameworks & Interna → Codierungsregeln für das Softwarepraktikum

Welche Erfahrungen gibt es mit Freundschaften in Teams?

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).

Welche Hilfen stehen zur Verfügung? (1)

- **WWW**-Seiten zum Softwarepraktikum

- **Tutorials** jeweils in der 6.DS
  - Webapplikationen mit Java & Spring **16.10.** (Oliver Gierke), **HSZ/H/004**
  - SalesPoint Framework **23.10.** (Oliver Gierke), **HSZ/02/E**

- **Lernraum** dienstags ab 30.10. 6. DS, E042
- LV Softwaretechnologie SS 17
- Skript zur Vorbereitung auf das Softwarepraktikum (http://static.olivergierke.de/lectures/)

- Technische **Infrastruktur** der TU Dresden
  - Eclipse unter Windows
  - Magic Draw UML (Academic License)
  - Mailinglisten

Welche Hilfen stehen zur Verfügung? (2)

Ihre Ansprechpartner

- Ihr(e) Praktikumsbetreuer(in) (studentische Tutorin)
- Praktikumsforum im Auditorium (Oliver Gierke u.a.)
- Lehrstuhl Softwaretechnologie
  - Dr. Birgit Demuth (Lehrbeauftragte)
  - Martin Morgenstern (Stellvertreter und technischer Ansprechpartner)

Welche Projektphasen gibt es?

- Projektlaufzeit insgesamt 12 Wochen (ab dieser Woche)

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

- Sechs Meilensteine (OOA, OOD, OOP_I, OOP_II_a, OOP_II_b, OOP_III)
  - 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.

➢ **Fertigstellung des Projektes am Freitag, den 11. Januar 2019 (harte Deadline!)**

Projektorganisation und Einarbeitung

Woche	Aktivitäten	Meilenstein

(1)
09.10.-14.10.
	• Teamarbeit organisieren
	• Einarbeitung in GitHub-zentrierte SW-Entwicklung
	  • Java-Tooling (Wdhlg.)
	  • Git und GitHub (Wdhlg.)	

Analyse (OOA)

Woche	Aktivitäten	Meilenstein
(2) 15.10.-21.10.	• CRC-Kartenmethode, Anforderungen, Analysemodell (Kontextdiagramm, Top-Level-Architektur, Anwendungsfall-, Klassen-, Sequenzdiagramme), GUI-Entwurf, Akzeptanztestfälle • Einarbeitung in das SalesPoint-Framework mit **Videoshop-Prototyp**	
(3) 22.10.-28.10.		**OOA** I. Pflichtenheft (→ ***Template***) II. Erweiterung des Videoshops (pro Teammitglied) 28.10. HARTE Deadline!

Entwurf und Prototyping (OOD)

Woche	Aktivitäten	Meilenstein
(4) 29.10.-04.11.	• **Anwendungsprototyp** von jedem Teammitglied • Entwurfsentscheidungen (Architektur, Entwurfs-muster, Persistenz, GUI) • Anpassung des Modells an das SalesPoint-Framework • verfeinerte UML-Modelle	
(5) 05.11.-11.11.		**OOD** I. Anwendungsproto-typ II. Testplan (verfeinerte Akzeptanztestfälle, → ***Template***) III. Entwicklerdoku-mentation v1(→ ***Template***)

Implementierung und Test (OOP)

Woche	Aktivitäten	Meilenstein
(6) 12.11.-18.11.	• wöchentliche Implementierung entsprechend Protokoll und Backlog • Verteilung der Issues/Packages/ Klassen an die Teammitglieder • fortlaufende junit-Tests • fortlaufende Javadoc-Dokumentation • Cross-Testing → ***Template***	
(7) 19.11.-25.11.		**OOP_I** Basisfunktionalität
(8) 26.11.-02.12.		
(9) 03.12.-09.12.		**OOP_II_a** Muss-Kriterien (als Basis für das Cross-Testing)
(10) 10.12.-16.12.		**OOP_II_b** Ergebnisse des Cross-Testings

Implementierung und Test (OOP)

Woche	Aktivitäten	Meilenstein
(11) 17.12.-21.12.	• Kann-Kriterien • Realisierung weiterer Kundenwünsche • Stabilisierung der Anwendung → Bearbeitung des Cross Testing Feedbacks • Abschluss von Entwickler- (und Anwender-) Dokumentation via GitHub Pages	
(12) 07.01.-11.01.		**OOP_III** I. Fertige Anwendung II. Dokumentation III. Auswertung des Praktikums (persönlich durch jedes Teammitglied) → ***Template***
(13) 14.01.-18.01.	ABSCHLUSSPRÄSENTATIONEN Online-Fragebogen	

Warum so ein strenger SE-Prozess?

- Erste Erfahrungen mit (professionellen) SE-Prozessen
- Analyse vs. Entwurf/Prototyping vs. Implementierung/Test
- Erfüllung von Meilensteinen
- **Hybride Softwareentwicklung**
  - → Kombination der Vorzüge von
    - agilen Methoden und
    - schwergewichtigen Methoden
- **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 ☹).

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

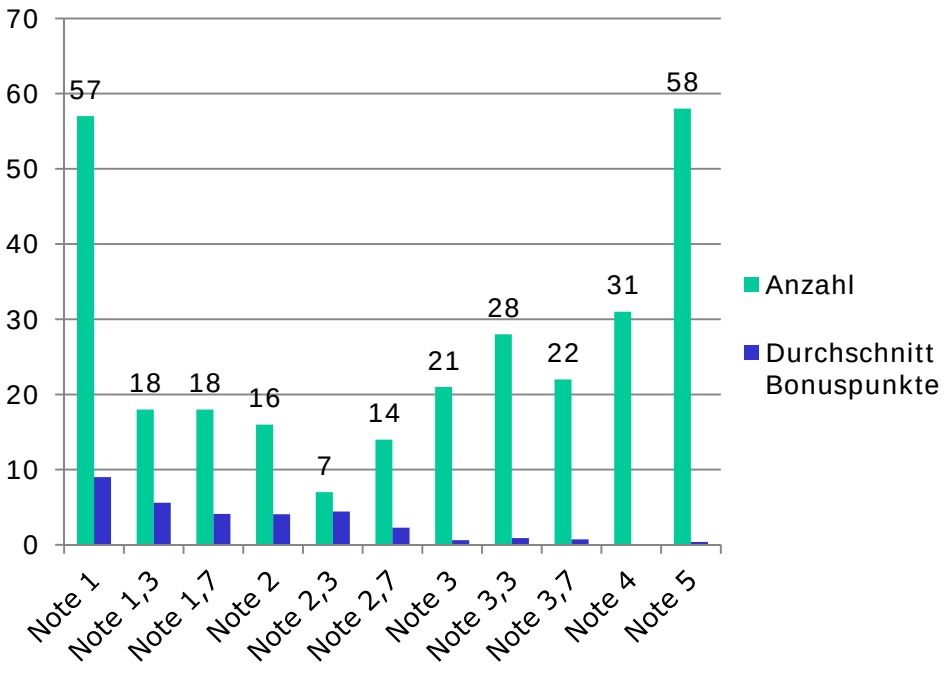
Bewertungskriterien (5) – Abschlusspräsentation

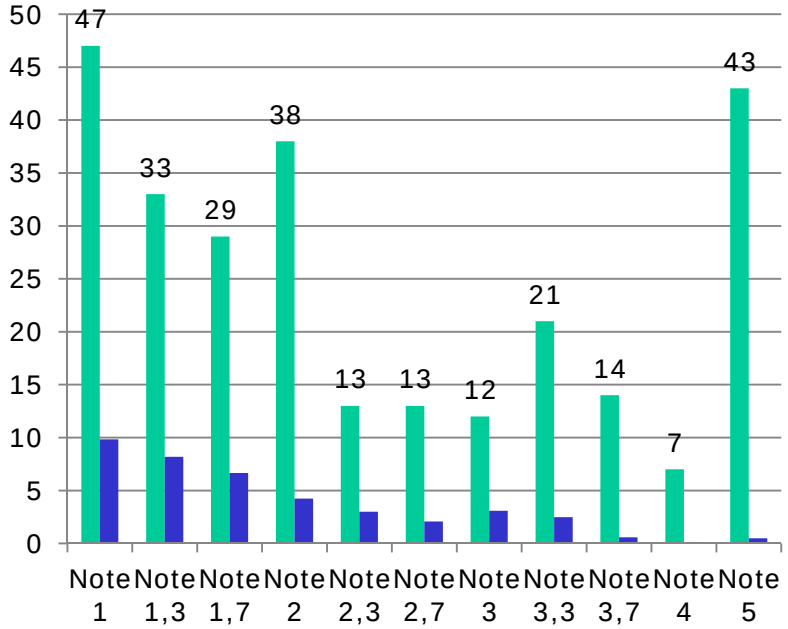
- Zeiteinhaltung
- Qualität des Vortrages
- Qualität der Vorführung der Anwendung
- Diskussion/Reaktion auf Fragen

Was gibt es konkret diese Woche zu tun?

- Kennenlernen des Teams, der Tutorin und des (externen) 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 Treffen 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

Auswertung der Klausuren SS 2017 und SS 2018





Dienstag 30.10., 16:00-17:00 Uhr, APB 2101

KLAUSUREINSICHT SOFTWARETECHNOLOGIE (SS 2018)

Los geht‘s

Coming together is a beginning.
Keeping together is progress.
Working together is success.

Henry Ford (1863 – 1947)