

SWT-1 - Modulzusammenfassung SS2021

Version 0.0.1

uwgmn, ujict, uzytw, udlzq, uiynn
Julian Keck, Julian Leitner, Christian Schliz, Niklas, Möwe

Stand: 28. April 2021

INHALTSVERZEICHNIS


1	Übersicht über die implementierten Layout-Commands	3
1.1	Unterüberschrift	3
1.1.1	UnterUnterüberschrift	3
2	Vorbemerkungen	4
3	Definitionen	4
4	Software	5
4.1	Allgemein	5
4.2	Beispiele	5
4.3	System- vs. Softwareentwicklung	5
4.4	Änderung an Software der letzten Jahren	5
5	UML (Universal Modeling Language)	6
5.1	Vererbung	6
5.2	Liskovsches Substitutionsprinzip	7
5.3	Varianzen von Parametern	7

1 ÜBERSICHT ÜBER DIE IMPLEMENTIERTEN LAYOUT-COMMANDS

1.1 Unterüberschrift

1.1.1 UnterUnterüberschrift

- Bäume sind Pflanzen
- «Ich stehe in Anführungszeichen»
- Ich bin grün, eigentlich aber als Verweis gedacht
- Hier¹
- $\mathbb{N}_+, \mathbb{N}_0, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{K}, \mathbb{F}, 1$
- $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \in \mathbb{R}^{3 \times 3}$
- $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n : v \mapsto Av$
- Sei $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n$
 $v \mapsto Av$

-  Softwaretechnik wird von Walter F. Tichy gelehrt.

¹Beginnt eine Fußnote

2 VORBEMERKUNGEN

3 DEFINITIONEN

■ **Informatik** ist die Wissenschaft von den (natürlichen und künstlichen) Informationsprozessen.

■ **Softwaretechnik** (engl. Software engineering) ist die technologische und organisatorische Disziplin zur systematischen Entwicklung und Pflege von Softwaresystemen, die spezifizierte funktionale und nicht-funktionale Attribute erfüllen.

■ **Softwarekonfigurationsverwaltung** (software configuration management) ist die Disziplin zur Verfolgung und Steuerung der Evolution von Software.

■ **Softwarekonfiguration** ist eine **eindeutig** benannte Menge von Software-Elementen mit den jeweils gültigen Versionsangaben, die zu einem bestimmten Zeitpunkt im Produktlebenszyklus in ihrer Wirkungsweise und ihren Schnittstellen aufeinander abgestimmt sind und gemeinsam eine vorgesehene Aufgabe erfüllen sollen.

■ **Software** (engl., "Weichware", Programmatik, im Gegensatz zur Apparatur) ist eine Sammelbezeichnung für Programme und Daten, die für den Betrieb von Rechensystemen zur Verfügung stehen, einschl. der zugehörigen Dokumentation.

■ **Softwareprodukt** ist ein Produkt für ein in sich abgeschlossenes, für einen Auftraggeber oder einen anonymen Markt bestimmtes Ergebnis eines erfolgreichen durchgeführten Projekts oder Herstellungsprozesses.

■ **Teilprodukt** beschreibt einen abgeschlossenen Teil eines Produkts.

4 SOFTWARE

4.1 Allgemein

Software (engl., "Weichware") ist eine Sammelbezeichnung für Programme und Daten für den Betrieb von Rechensystemen einschließlich deren Dokumentation.

4.2 Beispiele

- Quellprogramme
- Bibliotheken
- Testprogramme
- Initialisierungsdaten
- Dialogtexte
- Anforderungsdokumentation

4.3 System- vs. Softwareentwicklung

■ **Softwareentwicklung** ist ausschließliche Entwicklung von Software.

■ **Systementwicklung** ist die Entwicklung eines Systems, das aus Hard- und Softwarekomponenten besteht. Bei deren Entwicklung auch Randbedingungen berücksichtigt werden müssen.

4.4 Änderung an Software der letzten Jahren

- Steigende Komplexität
- Wachsender Anteil auf Mobilgeräten
- Vernetzung
- Steigende Qualitätsanforderungen
- Zunehmend mehr Ätlastenünd Äußer-HausEntwicklung

Trotz der Tatsachen, dass Software an sich keinem Verschleiß unterliegt, ist sie nicht einfacher zu warten wie Apparaturen ähnlicher Komplexität, da Software vielen Abhängigkeiten unterliegt.

5 UML (UNIVERSAL MODELING LANGUAGE)

■ **Kardinalität** Die Anzahl der Elemente einer Menge (Ganzzahl ≥ 0).

■ **Multiplizität** ein geschlossenes Intervall der zulässigen Kardinalitäten.

- $0..1 \rightarrow 0$ oder 1
- $0.. \rightarrow$ mindestens 0
- $x..* \rightarrow$ mindestens x
- $x \rightarrow$ **genau** x
- $* \rightarrow 0,1$ oder mehrere

Keine Angabe heißt nicht Spezifiziert \rightarrow IMMER GENAU 1

■ **Qualifizierer** (engl. qualifier): Ein(e) Attribut(kombination), die eine Partitionierung auf der Menge der assoziierten Exemplaren definiert.

■ **Qualifizierte Assoziation** (engl. qualified association): Eine Assoziation, bei der die Menge der referenzierten Objekte durch einen Qualifizierer partitioniert ist.

TODO: Static \rightarrow unterstrichen Constructor: «create», name irrelevant. Abstrakte Methode: kursiv oder {abstract} als Suffix «interface» (wird mit einem gestrichelten Pfeil mit weißer Spitze implementiert «realize») A «uses» B : A benutzt etwas aus B

5.1 Vererbung

■ **Vererbung** Es seien A und B Klassen, sowie Ω_A und Ω_B die Menge der Objekte, die die Klassen A und B ausmachen. Dann ist B eine **Unterklasse/Spezifizierung** von A (oder A eine **Oberklasse/Generalisierung** von B), wenn gilt: $\Omega_B \subseteq \Omega_A$.

Man sagt dann auch, dass B von A erbt. Da jedes Exemplar von B auch ein Exemplar von A ist, heißt die Beziehung zwischen A und B die «**ist-ein-Beziehung**» (engl. is-a relation). Wenn A mehrere Unterklassen hat, so sollten diese Unterklassen in der Regel disjunkt sein.

🔗 $\Omega_B \subseteq \Omega_A$ Diese Teilmengenrelation gilt, da Ω_A alle Klassen enthält, die mindestens die angegebenen Attribute beinhalten, allerdings auch beliebige viele andere.

■ **Signaturvererbung** (engl. signature inheritance) Eine in der Oberklasse definiert und evtl. implementierte Methode überträgt nur ihre Signatur auf die Unterklasse.

■ **Implementierungsvererbung** (engl. implementation inheritance) Eine in der Oberklasse definierte und implementierte Methode überträgt ihre **Signatur und ihre Implementierung** auf die Unterklasse.

■ **Überschreiben** (engl. override): Eine geerbte Methode unter Beibehaltung der Signatur wird **neu implementiert**

■ **Abstrakte Methode** (engl. abstract Method): Signaturdefinition ohne Angabe einer Implementierung

■ **Schnittstelle** (engl. interface): Definition einer Menge **abstrakter** Methoden, die von den Klassen, die sie implementieren, angeboten werden müssen.

🔗 Schnittstelle nicht direkt instanzierbar.

💡 Nutzende Klasse darf beliebige implementierende Klasse instanziiieren.

❗ Bei Schnittstellen gibt es keine Vererbung

■ Wenn eine Schnittstelle B eine Schnittstelle A **erweitert**, dann ist die Menge der abstrakten Methoden von A eine Teilmenge der Mengen der abstrakten Methoden von B ($A \subseteq B$).

■ Wenn eine Klasse X eine Schnittstelle A **implementiert**, dann ist die Menge der abstrakten Methoden von A eine Teilmenge der Methodendefinitionen von X, wobei X zusätzlich jeweils eine Implementierung angeben darf.

5.2 Liskovsches Substitutionsprinzip

TODO: Change image

❗ **Liskovsches Substitutionsprinzip** (engl. Liskov substitution principle): In einem Programm, in dem U eine Unterklasse von K ist, kann jedes Exemplar der Klasse K durch ein Exemplar von U ersetzt werden, wobei das Programm weiterhin korrekt funktioniert.)

💡 Das Substitutionsprinzip fordert nur, dass man ein Exemplar der Unterklasse so verwenden kann, als wäre es ein Exemplar der Oberklasse. Es wird nicht gefordert, dass die Signatur gleich bleibt!

💡 Die Unterklasse hat die gleichen oder schwächeren Vorbedingungen als die Oberklasse
Für Methoden heißt das, dass bei einer Substitution der Oberklasse durch eine Unterklasse die Vorbedingungen der Unterklassenmethode ebenfalls erfüllt sind.

💡 Die Unterklasse bietet die gleichen oder stärkeren Nachbedingungen wie die Oberklasse
Für Methoden heißt das, dass bei einer Substitution die Ergebnisse der Unterklassenmethode die Bedingungen erfüllen, die an die Oberklassenmethode gestellt werden.

💡 Unterklassenmethoden dürfen nicht mehr erwarten und weniger liefern.

5.3 Varianzen von Parametern

■ **Varianz** (engl. variance): Modifikation der Typen der Parameter einer überschriebenen Methode.

■ **Kovarianz** (engl. covariance): Verwendung einer Spezialisierung des Parametertyps in der überschreibenden Methode

❗ Bei Eingabeparametern ein Problem

💡 Bei Rückgabewerten unproblematisch

■ **Kontravarianz** (engl. contravariance): Verwendung einer Verallgemeinerung des Parametertyps in der überschreibenden Methode

💡 Bei Eingabeparametern unproblematisch

❗ Bei Rückgabewerten ein Problem

■ **Invarianz** (engl. invariance): keine Modifikation des Typs

💡 Invarianz ist nie ein Problem