

10 Paketdiagramm

Inhalt

10.1 Allgemein	-2
10.2 Darstellung eines Pakets	-4
10.3 Paket als Namensraum	-5
10.4 Regeln zur Sichtbarkeit	-6
10.5 Dependencies - Abhängigkeitsbeziehung zwischen Paketen	-7
10.6 Import von Elementen oder Paketen	-8
10.6.1 Import von Elementen	-8
10.6.2 Import von Paketen	-9
10.6.3 Import-Beziehung	-10
10.6.4 Beispiel für Sichtbarkeit von Paketelementen	-11
10.7 Verschmelzung von Paketen	-13
10.7.1 Verschmelzungsregeln	-14
10.7.2 Resultat der Verschmelzung von Paketen	-15
10.8 Beispiel Online Shopping System	-18
10.9 Sonderform: Paket als Modell	-19
10.10 Sonderform: Paket als Bibliothek	-21
10.11 Weitere besondere Sprachmerkmale	-21
10.12 Verwendung und Organisation von Paketen	-22

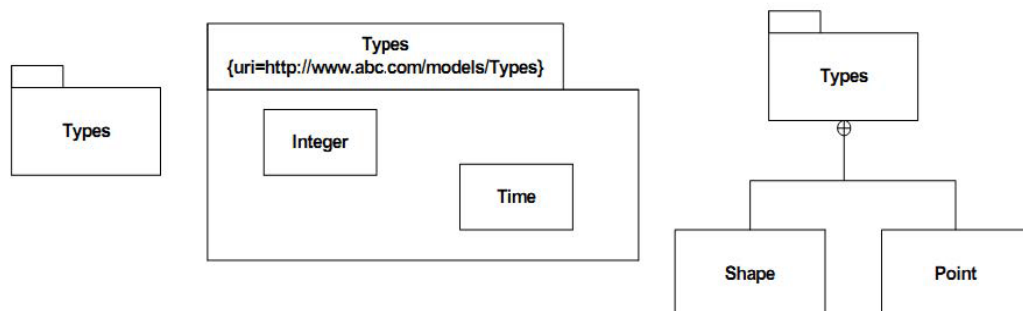
10.1 Allgemein

- ❑ Fokus liegt auf **Strukturierungsmechanismus** in Form von Paketen um Komplexität von Modellen beherrschbar zu halten, Übersicht zu wahren.
 - Android SDK – 69 Pakete vs. 1.231 Klassen
- ❑ Ein Paket (*package*) erlaubt es, eine beliebige Anzahl von paketierbaren UML-Modellelementen (kurz: Elementen) zu **gruppieren**: u.a. Klassen, Datentypen, Aufzähltypen und Komponenten.
- ❑ Pakete sind selbst paketierbare Elemente.
Pakete können daher wieder Pakete enthalten.
- ❑ Wesentliche Konzepte in einem Paketdiagramm (*Packages Diagram*)
 - Paket als **Namensraum** (siehe auch Konzept *Namespace* in UML [OMG 2015])
 - **Sichtbarkeitsregeln** steuern die Nutzung der Teile eines Pakets
 - **Import** von Elementen und/oder Paketen in einen Namensraum
 - **Verschmelzung von Paketen** ermöglicht die Erweiterung/Anpassung von Elementen eines Modells

- ❑ Soll mit der Paketstruktur nicht nur Gruppierung sondern noch zusätzlich die Semantik eines Modells im Sinne einer Beschreibung eines Systems ausgedrückt werden, kann eine im UML-Standard vorgesehene Spezialform von Paketen zum Einsatz kommen, so genannte Modelle.
 - Modelle werden ebenfalls wie Pakete dargestellt, jedoch mit dem Schlüsselwort «model» gekennzeichnet
 - Gesamtsystem, i.d.R. eine Sammlung von Modellen, wird mit «systemModel» gekennzeichnet
- ❑ Analoges gilt für Bibliotheken und Frameworks
 - Bibliotheken und Frameworks werden ebenfalls wie Pakete dargestellt, jedoch mit dem Schlüsselwort «modelLibrary» bzw. «framework» gekennzeichnet
- ❑ Vollständige **Spezifikation der Konzepte siehe [OMG 2015]**; → Literaturangaben!)
- ❑ Hinweis: unterschiedliche Unterstützung von Paketdiagrammen durch Tools.

10.2 Darstellung eines Pakets

- ❑ Darstellung: großes Rechteck mit aufgesetztem kleinen Rechteck (Karteikarte mit Reiter oder File Folder Symbol)
 - Alternative 1: Paketinhalt wird nicht gezeigt, Paketname steht innerhalb des großen Rechtecks.
 - Alternative 2: Im großen Rechteck wird eine Anzahl der vom Paket gruppierten Elemente, d.h. der Paketinhalt, gezeigt. Paketname steht im kl. Rechteck (Reiter).
 - Alternative 3: (einige) Elemente des Paketinhalts werden außerhalb des Pakets notiert und über Linien mit dem Paket verbunden. Linien sind durch ein in einem Kreis eingeschlossenes Plusymbol am Linienende beim Paket ausgezeichnet.



10.3 Paket als Namensraum

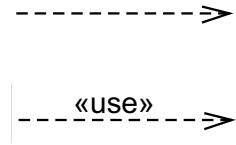
- ❑ Ein Element kann zu höchstens einem Paket gehören.
- ❑ Ein Paket definiert einen Namensraum, in dem alle im Paket befindlichen Elemente enthalten sind (*“owned”* oder *“contained”*)
- ❑ Namen dienen zur Referenz auf Elemente.
Namen der Elemente innerhalb eines Paktes müssen eindeutig sein.
- ❑ Qualifizierte Namen ermöglichen global eindeutige Namen für Elemente:
Namen aller (direkt und indirekt) übergeordneten Pakete (beginnend mit äußerstem Paket) eines Elements, getrennt durch “::”
 - Beispiele: `Types::Time` oder `Types::Point`

10.4 Regeln zur Sichtbarkeit

- ❑ Elemente eines Pakets müssen eine Art der Sichtbarkeit zugeordnet bekommen (auch wenn Sichtbarkeit nicht explizit angezeigt wird, ist Art der Sichtbarkeit definiert)
 - privat - Elementname mit Präfix “-” versehen:
Elemente sind nur innerhalb des Pakets (auch in den darin enthaltenen geschachtelten Paketen) sichtbar, nicht jedoch außerhalb des umschließenden Pakets
 - public - Elementname mit Präfix “+” versehen:
Elemente sind global sichtbar, d.h., auch außerhalb des umschließenden Pakets verwendbar

10.5 Dependencies - Abhängigkeitsbeziehung zwischen Paketen

- ❑ Paket A *hängt ab* (*depends on*) von einem Paket B, wenn Paket A eine Klasse enthält, die von einer Klasse in B abhängt.
- ❑ Abhängigkeitsbeziehung hält die Abhängigkeiten zwischen paketierbaren Elementen aus verschiedenen Paketen fest
- ❑ Es sollten keine Zyklen über die Abhängigkeitsbeziehungen entstehen.
- ❑ Verschiedene Arten von Abhängigkeiten (Dependencies) in UML möglich
 - unspezifisch, keine Benennung des Pfeils
 - sehr häufig benannt mit **<<use>>**, mit obiger Bedeutung der Abhängigkeit
 - andere Arten, z.B. "Implementation", bei mehreren Implementierungen einer Spezifikation



10.6 Import von Elementen oder Paketen

10.6.1 Import von Elementen

- ❑ Statt Elemente (z.B. *K*) aus anderen Paketen mit qualifizierten Namen (z.B. *A::B::...::/::K*) zu referenzieren, können Elemente aus anderen Paketen **importiert** werden.
- ❑ Namen von importierten Elementen werden dem Namensraum hinzugefügt und können innerhalb des Pakets direkt verwendet werden.
- ❑ Voraussetzung: zu importierendes Element muss Sichtbarkeit public (+) zugeordnet haben.
- ❑ Es kann für das importierte Element ein Alias-Namen angegeben werden. Statt des Elementnamens wird der angegebene Alias dem Namensraum hinzugefügt. Referenzen auf den Alias entsprechen Referenzen auf das importierte Element. Der Alias-Namen muss sich von den anderen Namen im importierenden Paket unterscheiden.

□ Regelungen bei Namenskonflikten, z.B.:

- Ein Name, der schon im Paket existiert, oder mehrere gleiche Namen können nicht importiert werden. Diese Namen der importierten Elemente werden nicht dem Namensraum hinzugefügt. Es müssen qualifizierte Namen für diese Elemente verwendet werden.
- Importierte Namen können Namen von Elementen in (direkt oder indirekt) übergeordneten Paketen "verdecken". Elemente mit "verdecktem" Namen können nur über qualifizierte Namen referenziert werden.

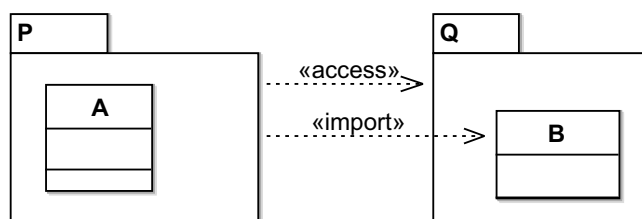
10.6.2 Import von Paketen

- Der Import eines Pakets ist gleichwertig dem Element-Import **eines jeden öffentlich sichtbaren** (public, "+"-Präfix) Elements des Namensraums des Pakets.

10.6.3 Import-Beziehung

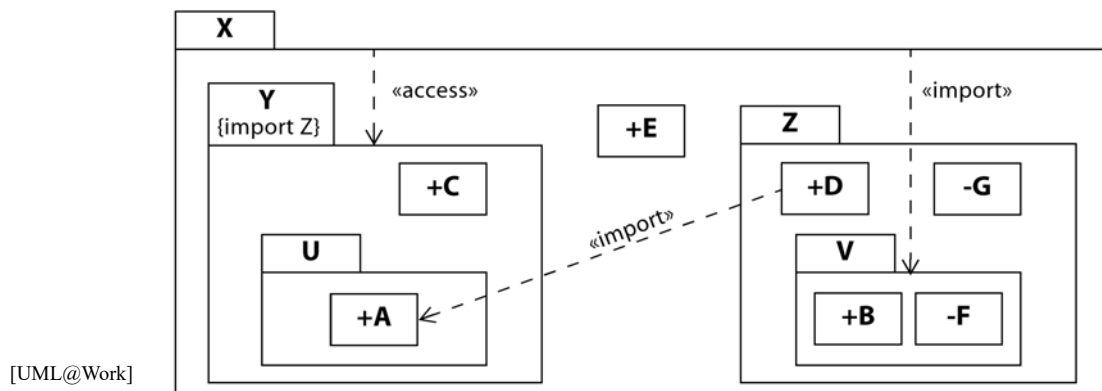
□ Notation der Import-Beziehung

- Strichlierte Linie mit offener Pfeilspitze, die auf das importierte Element (oder Paket) zeigt.
- Sichtbarkeit der importierten Elemente kann neu bestimmt werden.
Allerdings nur von *public* auf *privat* änderbar.
- Schlüsselwort **<<access>>** : Sichtbarkeit der importierten Elemente und Pakete wird auf "*privat*" gesetzt (private Import-Beziehung bezeichnet).
- Schlüsselwort **<<import>>** : Sichtbarkeit bleibt "*public*" (public Import-Beziehung bezeichnet).



- Alternative Notation: Text nach/unter P: {access Q}
{element import B}

10.6.4 Beispiel für Sichtbarkeit von Paketelementen

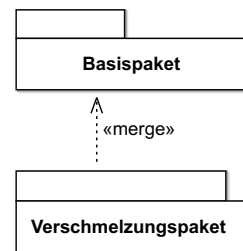


- Klasse A kann Klassen C und E sehen, weil diese in umschließenden Paketen
- Klasse A kann D sehen, da Y Paket Z importiert und Y umschließendes Pakte von U
- Klasse A kann B sehen, weil X Paket V importiert
- Klasse C sieht D, weil Y importiert Z
- C sieht E, da E in umschließendem Paket ist

- C sieht B, weil X das Paket V, das B enthält, importiert
- E sieht B, weil X das Paket V importiert
- E sieht C, weil X das Paket Y importiert
- E sieht D, transitiv über die Import-Beziehungen von X zu Y und von Y zu Z
- E kann G nicht sehen, weil G durch seine private Sichtbarkeit beim Import nach Y nicht berücksichtigt wird
- D sieht E, weil E im umschließenden Paket ist
- D sieht C, weil Y Paket X importiert
- A von keinem anderen Element gesehen außer D, da es sonst keine Import-Beziehung zum Paket U oder zur Klasse A gibt
- G sieht die gleichen Klassen wie D ausgenommen A
- B und F sehen C, D, E und G, da die Klassen D, G und E in den umschließenden Paketen Z und X sind, und da C zum Paket Y gehört, das von X importiert wird
- Die Klasse F ist privat und daher außerhalb von V nicht sichtbar
- B und F sehen sich untereinander, da sie zum selben Paket V gehören
- D und G sehen sich untereinander

10.7 Verschmelzung von Paketen




- ❑ Erweiterung eines Pakets oder Anpassung bei seiner Verwendung macht es oft erforderlich, die enthaltenen Elemente systemisch zu modifizieren, z.B.:
 - einer enthaltenen Klasse ein Attribut hinzuzufügen
 - eine enthaltene Klasse an einer zusätzlichen Assoziation teilnehmen zu lassen
 - oder eine enthaltene Klasse von einer neuen abzuleiten
- ❑ “Manuelle” Lösung: Import-Beziehungen und entsprechenden Generalisierungen und Redefinitionen vornehmen, allerdings i.d.R. oft repetitiv und schematisch
- ❑ Besser: “Automatische” Lösung durch Nutzung des Konzepts der Paketverschmelzung
- ❑ Notation: wie Paket-Import, aber mit Schlüsselwort «merge»
- ❑ Verschmelzungsbeziehung nur zwischen Paketen möglich
- ❑ Inhalt des Basispakets (bei der Pfeilspitze) wird mit Inhalt des Verschmelzungspakets (am anderen Linienende) nach bestimmten Regeln verschmolzen



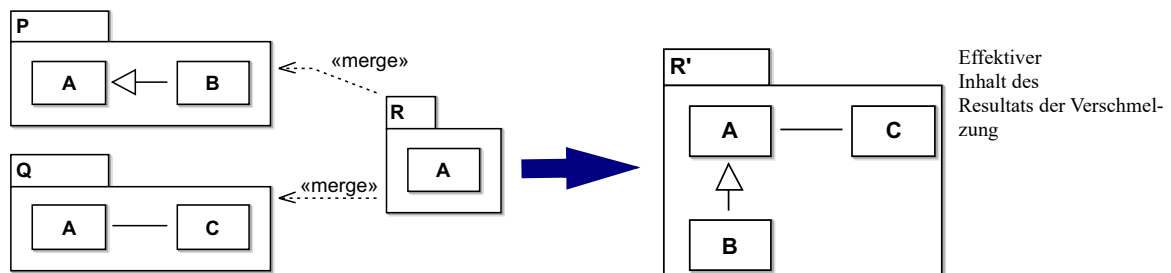
10.7.1 Verschmelzungsregeln

- ❑ Effektiver Inhalt des Verschmelzungspakets (= Resultat der Verschmelzung) ist die Vereinigungsmenge aus dem Inhalt des Basispakets mit dem Inhalt des Verschmelzungspakets - gemäß bestimmter Regeln!
- ❑ Verschmelzungsbeziehung nur möglich wenn:
 - Kein Zyklus von “merge”-Beziehungen/Abhängigkeiten
 - Verschmelzungspaket darf das Basispaket nicht enthalten
 - Basispaket darf nicht das Verschmelzungspaket enthalten
 - Element im Verschmelzungspaket darf kein Element des Basispakets referenzieren
 - zu verschmelzende (übereinstimmende) typisierte Elemente müssen den gleichen Typ (Klasse) haben oder einen gemeinsamen Supertyp (-klasse)
 - Typkonformität bei Klassen und Datentypen durch Generalisierungshierarchie definiert, ansonsten durch Typgleichheit
- ❑ Spezielle Regelungen und Vorbedingungen für Pakete und deren Elemente siehe Abschnitt 3.3.2 in [UML@Work] (→ Literaturangaben!) oder siehe Originalspezifikation in Abschnitt 12.2.3.3 in [OMG 2015]

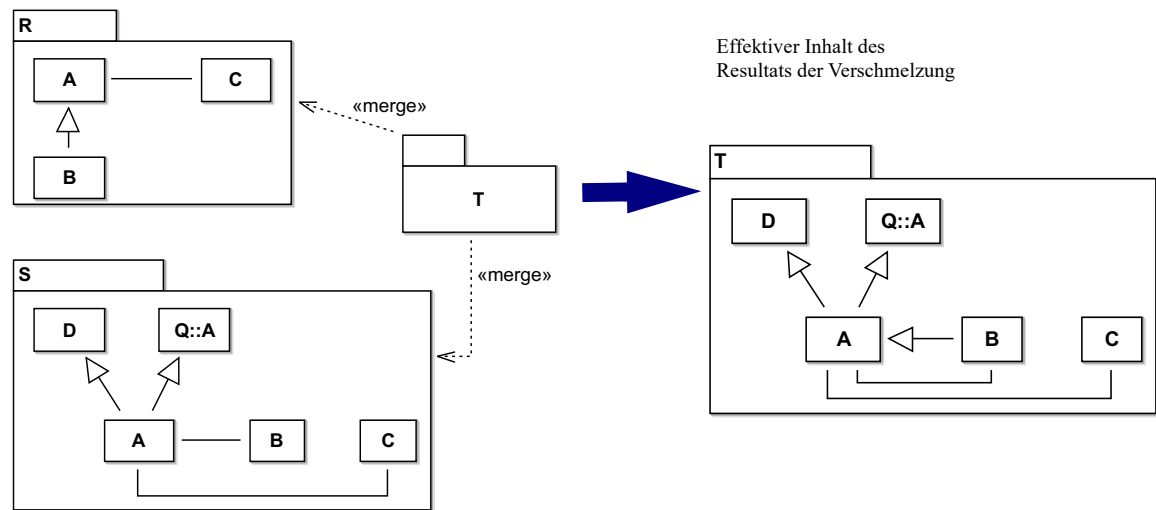
10.7.2 Resultat der Verschmelzung von Paketen

- ❑ Verschmelzungspaket beschreibt nur neu hinzukommende bzw. redefinierende Modellelemente.
- ❑ Konzeptuell enthält es jedoch jene Modellelemente, die aus der Verschmelzung der Modellelemente im Basispaket mit den Modellelementen im Verschmelzungspaket resultieren.
- ❑ Verschmelzung von   $\leftarrow \langle\langle \text{merge} \rangle\rangle$ resultiert in einem effektiven Inhalt  !!

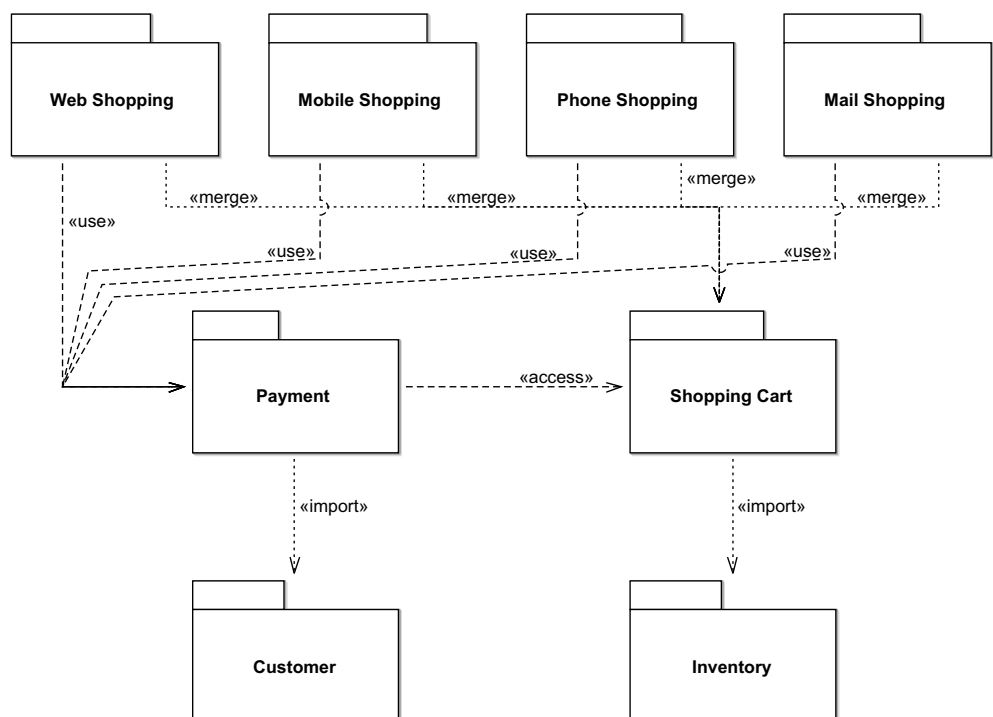
- ❑ Beispiel 1 für Resultat einer Verschmelzung:



□ Beispiel 2 für Resultat einer Verschmelzung:



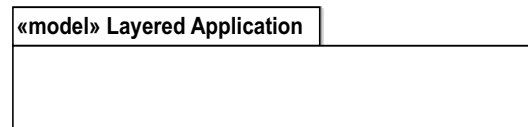
10.8 Beispiel Online Shopping System



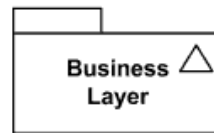
[OMG 2015]

10.9 Sonderform: Paket als Modell

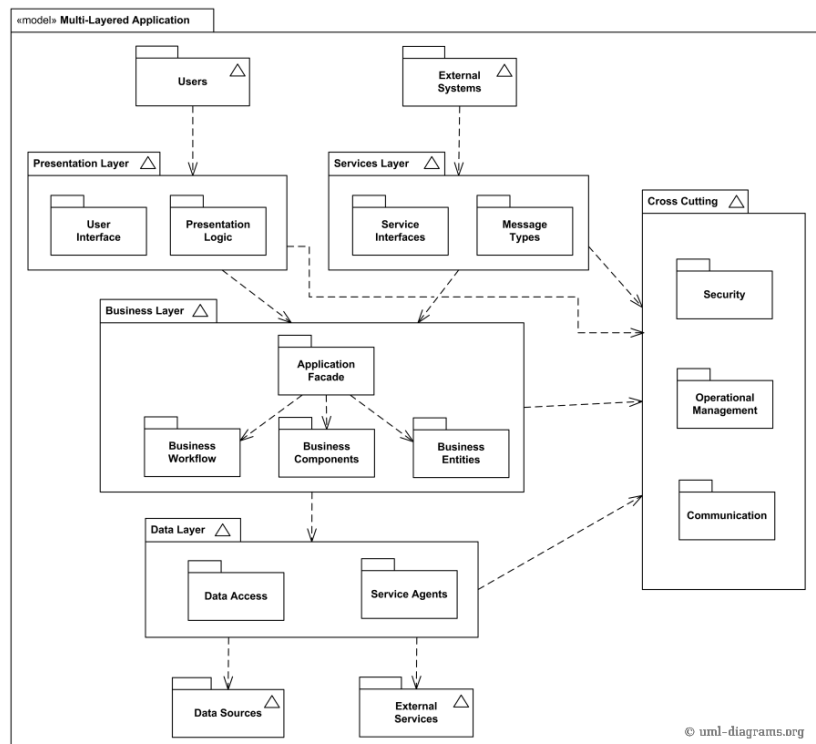
- Modell-Paket erkennbar an «model»



- Alternative Notation mit Dreieck-Symbol:



- Beispiel siehe:
<http://www.uml-diagrams.org/package-diagrams-overview.html#model-diagram>



10.10 Sonderform: Paket als Bibliothek

- ❑ Bibliothek-Paket erkennbar an «modelLibrary»
- ❑ Soll eine Bibliothek von generischen, wiederverwendbaren Modellelemente definiert werden, so wird diese durch ein Paket mit dem Stereotyp «modelLibrary» modelliert.

10.11 Weitere besondere Sprachmerkmale

- ❑ Seit UML 2.4 können Pakete optional ein *URI Attribut* mit einer eindeutigen URI zur Identifizierung eines Pakets besitzen
 - wird i.d.R. unter dem Paketnamen angegeben: {uri=<uri>}
- ❑ Pakete = *Package* / Paketdiagramm = *Packages Diagram* oder *Package Diagram*
- ❑ Allgemeiner Hinweis:
Generell verfügen alle UML-Sprachkonzepte, die einen Namensraum definieren, über die Fähigkeit zu importieren. Dazu zählen neben Paketen u.a. auch Klassen, Interfaces, Datentypen, Komponenten sowie die Verhaltenskonzepte wie z.B. Zustandsautomaten.

10.12 Verwendung und Organisation von Paketen

- ❑ Grundsätzlich: sinnvolle, nützliche Namen für Pakete vergeben
- ❑ Zwei häufige, speziellere Fälle:
 - “Klassen Paket Diagramm” (tritt sehr häufig auf)
 - Klassendiagramme stellen die essentiellen zu gruppierenden Elemente dar
 - “Anwendungsfall Paket Diagramm”
 - Use Case Diagramme stellen die zu gruppierenden Elemente dar
 - bei großen Projekten zur Organisation von Anforderungen sehr sinnvoll
- ❑ Heuristik zur Organisation von Klassendiagrammen in Paketdiagramme
 - Klassen(diagramme) eines Frameworks gehören in das gleiche Paket
 - Klassen in einer Vererbungshierarchie gehören i.d.R. in das gleiche Paket
 - Klassen die mit Aggregations- oder Kompositionsbeziehungen verbunden sind, gehören i.d.R. in das gleiche Paket
 - Klassen, die sehr oft miteinander “kollaborieren”, gehören i.d.R. in das gleiche Paket
- ❑ Heuristik zur Organisation von Use Case Diagrammen in Paketdiagramme
 - Assoziierte Use Cases (d.h. included, extending, inheriting use cases) gehören i.d.R. in das gleiche Paket



Literatur und Quellen:

- ❑ [OMG 2015] OMG Unified Modeling Language™ (OMG UML), Version 2.5
Normative Reference: <http://www.omg.org/spec/UML/2.5>
OMG Document Number formal/2015-03-01
- ❑ [UML@Work]
Kapitel 3.3 in:
Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, Werner Retschitzegger:
UML @ Work - Objektorientierte Modellierung mit UML2.
dpunkt Verlag 2005 / 3. aktualis. u. überarb. Aufl. 2005.
ISBN-13: 9783898642613 ISBN-10: 3898642615
- ❑ Tool zur Vertiefung: BEE-UP Modelling Tool:
<http://austria.omilab.org/psm/content/bee-up/info>
<http://www.omilab.org/web/guest/omilab-in-education/cmmc>