Zurich University of Applied Sciences

School of Engineering

Bachelor of Science (BSc) in Informatik

Modul Advanced Software Engineering 1 (ASE1)

Software Architektur

Beschreibung und Kommunikation

Institut für Angewandte Informationstechnologie (InIT)

Walter Eich (eicw) / Matthias Bachmann (bacn)

https://www.zhaw.ch/de/engineering/institute-zentren/init/

Agenda

Zurich University of Applied Sciences

School of Engineering

- 4.3. Sichten und Schablonen
- 4.4. Technische und querschnittliche Konzepte
- 4.5. Architektur und Implementierung
- 4.6. Übliche Dokumententypen zu Softwarearchitekturen
- 4.7. Praxisregel zur Dokumentation
- 4.8. Beispiele weitere Architekturframeworks

4.1 Lernziele

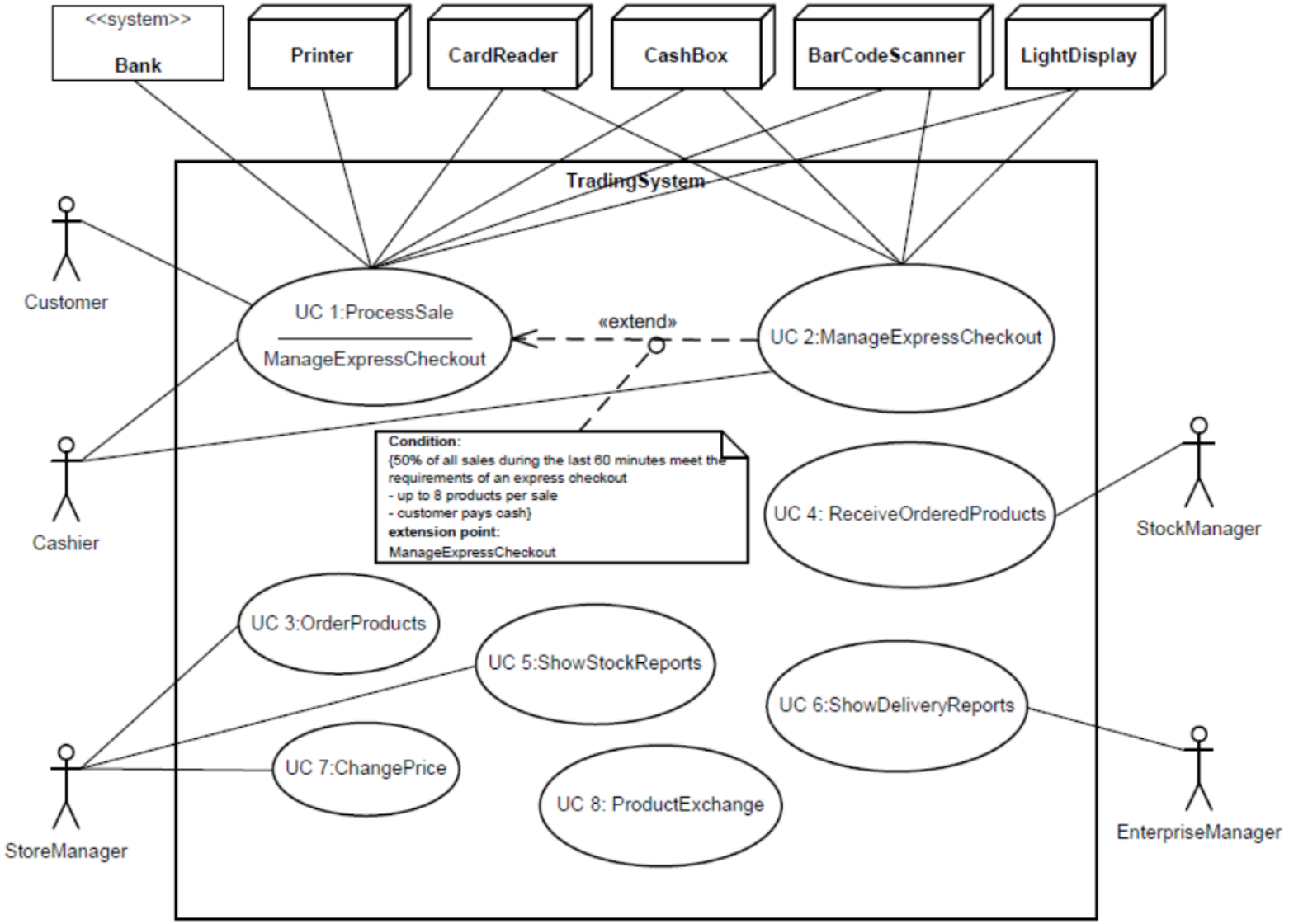
Zurich University of Applied Sciences
zh aw School of Engineering

- Lernziele
  - LZ 3-1: Qualitätsmerkmale technischer Dokumentation erläutern und berücksichtigen
  - LZ 3-2: Softwarearchitekturen stakeholdergerecht beschreiben und kommunizieren
  - LZ 3-3: Notations- / Modellierungsmittel für Beschreibung von Softwarearchitektur erläutern und anwenden
  - LZ 3-4: Architektursichten erläutern und anwenden
  - LZ 3-5: Kontextabgrenzung von Systemen erläutern u. anwenden
  - LZ 3-6: Querschnittliche und technische Architekturkonzepte erläutern und anwenden
  - LZ 3-7: Schnittstellen beschreiben
  - LZ 3-8: Architekturentscheidungen erläutern und dokumentieren
  - LZ 3-9: Dokumentation als schriftliche Kommunikation nutzen
  - LZ 3-10: Weitere Hilfsmittel und Werkzeuge zur Dokumentation kennen

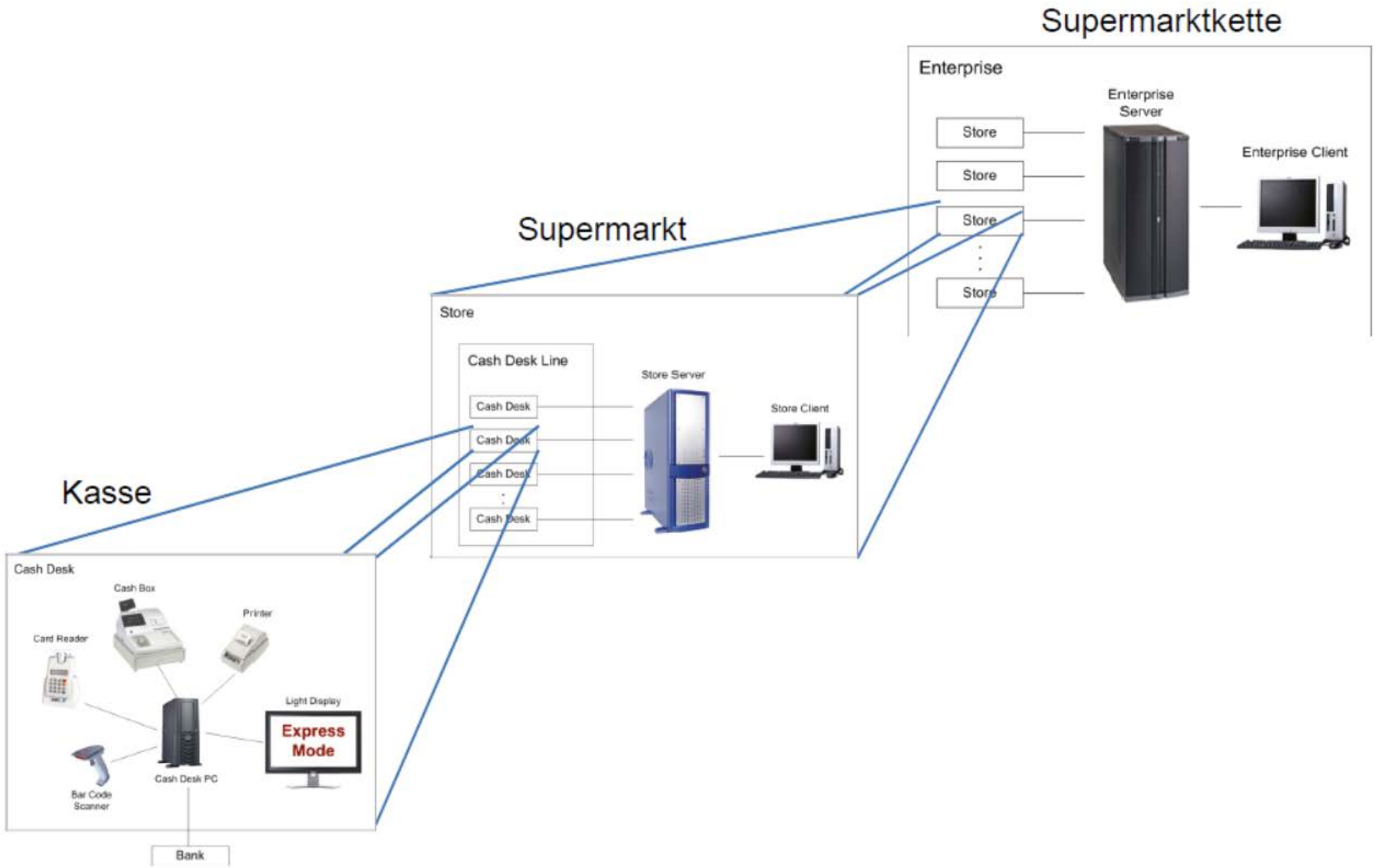
4.2 Einführung Beispiel CoCoME (1) Common Component Modeling Example

- Das Beispiel CoCoME wird verwendet um einzelne Aspekte in den folgenden Kapiteln darzustellen
- Anwendungssystem für eine Supermarktkette
  - Anwendungssystem für eine Supermarktkette
  - [UC1] ProcessSale: Kernfunktionen der Supermarktkasse
  - [UC2] ManageExpressCheckout: Extension zu [UC1]
  - [UC3-UC8] Verwaltung des Warenbestandes
- Weiterführende Info: http://www.cocome.org/

Einführung Beispiel CoCoME (2) Use Cases



Einführung Beispiel CoCoME (3)
Übersicht des Aufbaus

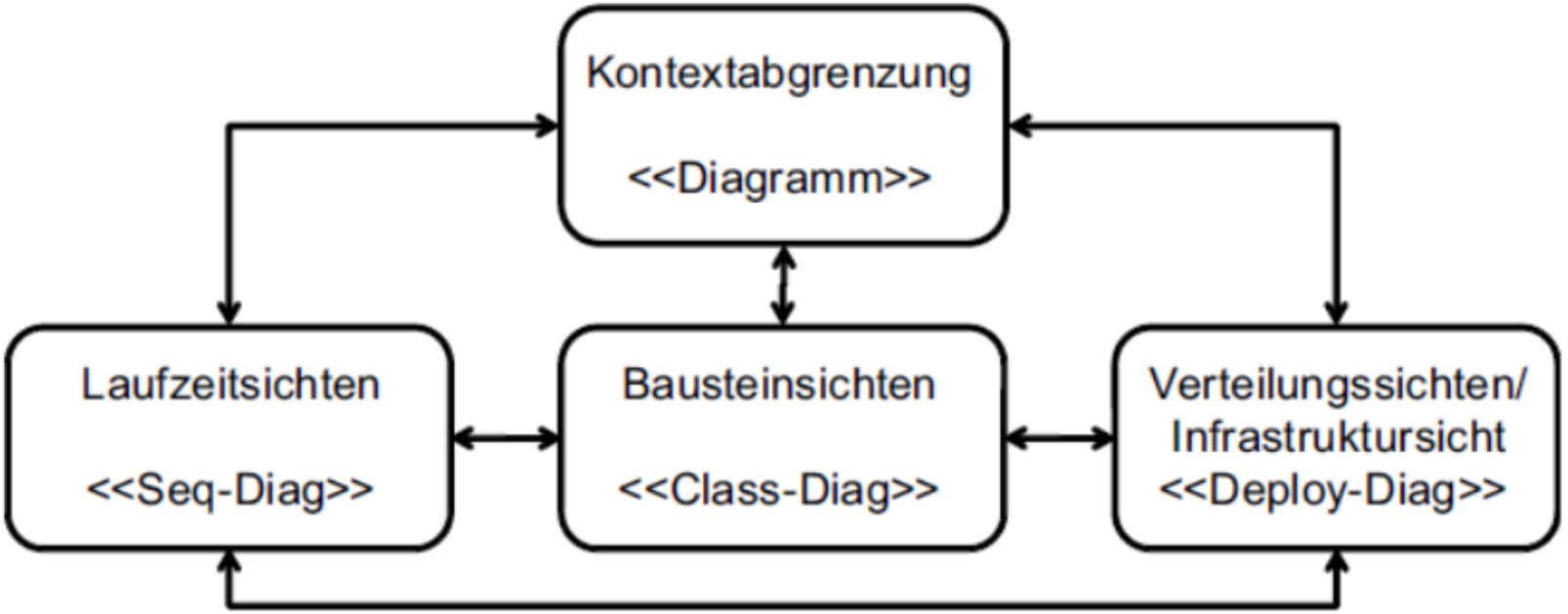


4.3 Bewährte Sichten nach iSAQB (2)

- 1. Kontextsicht oder Kontextabgrenzung
  - Diagramm mit (vorzugsweise) UML-Komponenten, mit »System unter Design« als Blackbox und allen externen Systemen und Nutzern als Akteure bzw. ebenfalls UML-Komponenten.
- 2. Bausteinsicht
  - UML-Komponenten- oder Top-Level-Klassendiagramme der funktionalen und ggf. technischen »Softwarebausteine« des Softwaresystems sowie ihrer Beziehungen untereinander
- 3. Laufzeitsicht
  - Sequenz-, Aktivitäts- oder ähnliche Diagramme zur Illustration wesentlicher bzw. besonders wichtiger Abläufe besonders zwischen den Bausteinen (»innerhalb «) des Softwaresystems
- 4. Verteilungs- bzw. Infrastruktursicht
  - Verteilung von Softwareartefakten des Softwaresystems auf Rechnerknoten, Netzwerke usw., also eine Abbildung der Software auf reale technische Infrastruktur

4.3.1 Bewährte Sichten nach iSAQB (1)

- www.arc42.de



Ergänzende Sichten

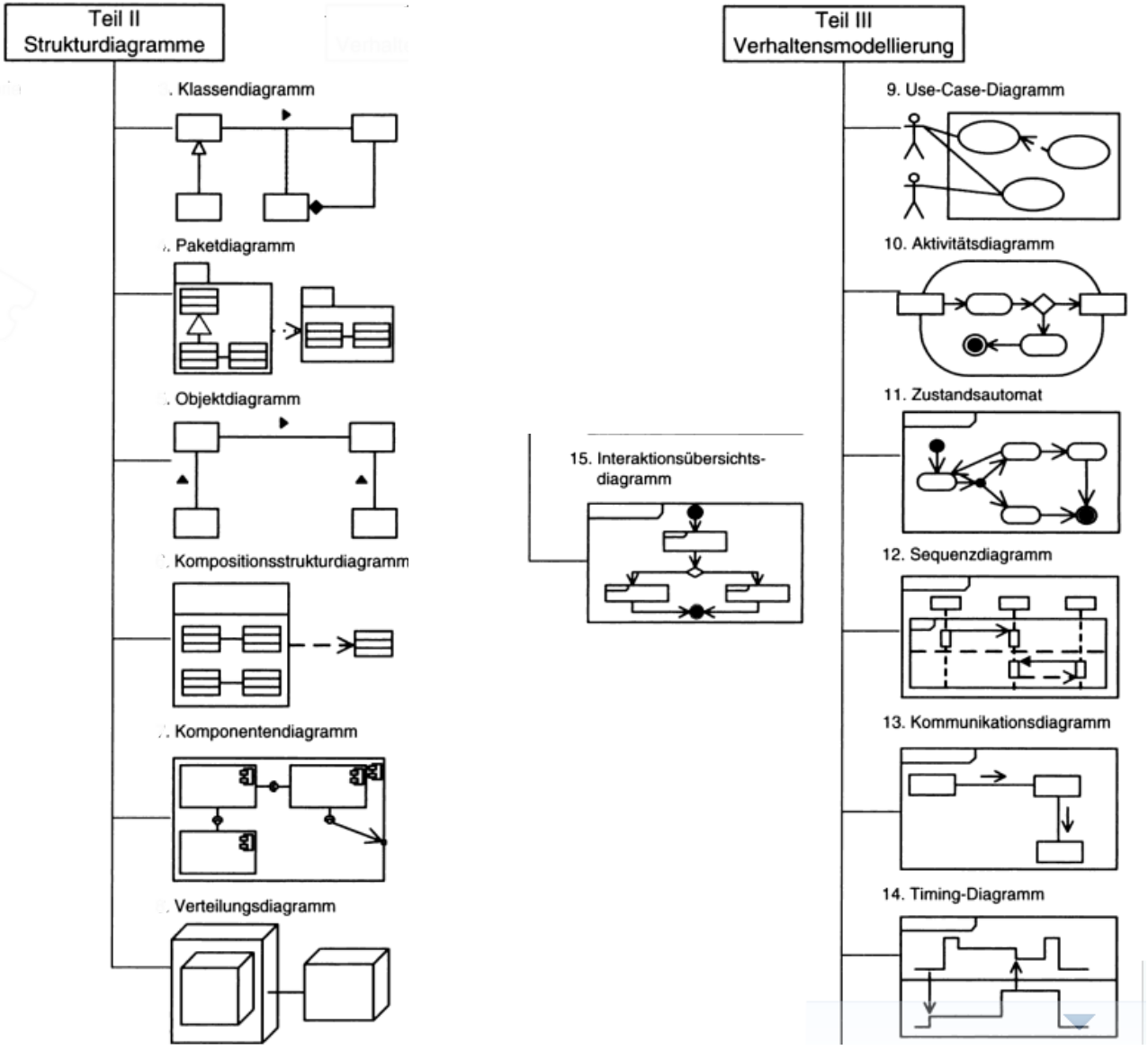
- Datensicht
  - Detaillierte Beschreibung der Datenbankstrukturen eines Softwaresystems z.B. mithilfe eines »Entity Relationship (ER)«-Modells
- »Big Picture«
  - Darstellung der »High Level«-Systemarchitektur zur Kommunikation mit der (die Mittel bewilligenden) Managementebene
- Masken- oder Ablaufsicht
  - Bildschirmmasken- oder Webseiten-Ablaufdiagramme usw.

4.3.2 UML-Diagramme als Notationsmittel (1)

- Die 2010 veröffentlichte Version 2.3 der UML ([UML-1b], [UML-1c]) enthält insgesamt 14 UML-Diagrammarten, die in je sieben Struktur- und Verhaltensdiagramme aufgeteilt sind.

UML 2.3-Strukturdiagramme	**UML 2.3-Verhaltensdiagramme**
UML-Klassendiagramm	**UML-Aktivitätsdiagramm**
UML-Kompositionsstrukturdiagramm	UML-Anwendungsfalldiagramm (auch: Use-Case-Diagramm)
UML-Komponentendiagramm	UML-Interaktionsübersichtsdiagramm
UML-Verteilungsdiagramm (auch: Infrastrukturdiagramm)	**UML-Kommunikationsdiagramm**
UML-Objektdiagramm	**UML-Sequenzdiagramm**
UML-Paketdiagramm	UML-Zeitverlaufsdiagramm
UML-Profildiagramm	UML-Zustandsdiagramm

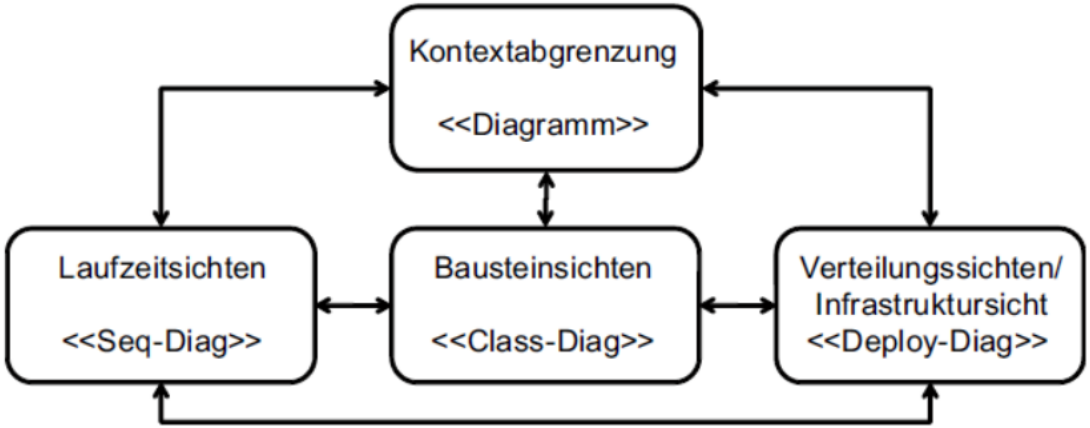
UML-Diagramme als Notationsmittel (1)



4.3.3 Grobaufbau – Schablonenartige Sichtenbeschreibung (1)

- Für Kontext-, Baustein-, Laufzeit- und Verteilungssicht:
  - Es kann eine einheitliche Struktur oder Gliederung zu deren Beschreibung verwendet werden
  - Verwendung von Text und Diagrammen
  - Faustregel : »So wenig Formalismus wie möglich, aber so viel wie nötig.«

- Aufbaumöglichkeit:
  - Kurzbeschreibung
  - Diagramme
  - Elementkatalog
  - Variabliltäten
  - Hintergrundinformationen



Grobaufbau – Schablonenartige Sichtenbeschreibung (2) Aufbau der Beschreibung

Zurich University of Applied Sciences
School of Engineering

- 1. Kurzbeschreibung:
  - Die Kurzbeschreibung der Sicht gibt in einer kurzen textuellen Beschreibung einen Überblick, »worum es im konkreten Fall geht«.
- 2. Diagramme:
  - Diagramme liefern eine grafische Darstellung der Sicht.
- 3. Elementkatalog:
  - Der Elementkatalog enthält eine textuelle oder tabellarische Aufstellung derjenigen Elemente oder Bausteine, die in dieser Sicht vorkommen.
    - Elemente und ihre Eigenschaften
    - Beziehungen und ihre Eigenschaften
    - Schnittstellen von und zwischen Elementen
    - Elementverhalten

Grobaufbau – Schablonenartige Sichtenbeschreibung (3) Aufbau der Beschreibung

Zurich University of Applied Sciences
zh aw School of Engineering

- 4. Variabilitäten:
  - Der Punkt Variabilitäten behandelt in Form einer textuellen Beschreibung die Frage, welche der in dieser Sicht dargestellten Elemente oder Beziehungen variabel sind.
  - Es sind Variabilitäten in Anforderungen, Architektur, Design, beteiligten Fremdsystemen oder Infrastruktur gemeint.
  - Abhängig vom Typ der Sicht können Konfigurations-, Installations- und Betriebsparameter hier ebenfalls erläutert werden.
  - Auch eine Liste aller berücksichtigten Technologiestandards kann hier genannt werden.
  - Innerhalb der Variabilitäten zwischen Änderbarkeit und Flexibilität zu unterscheiden.
    - Änderbarkeit behandelt die absehbaren Anpassungsmöglichkeiten des aktuellen Systems, z.B. die Änderung des eingesetzten JDBC-Datenbanktreibers.
    - Flexibilität betrachtet hingegen die Erweiterungsfähigkeit des Systems, z.B. durch das Vorsehen von Ausbaustufen.

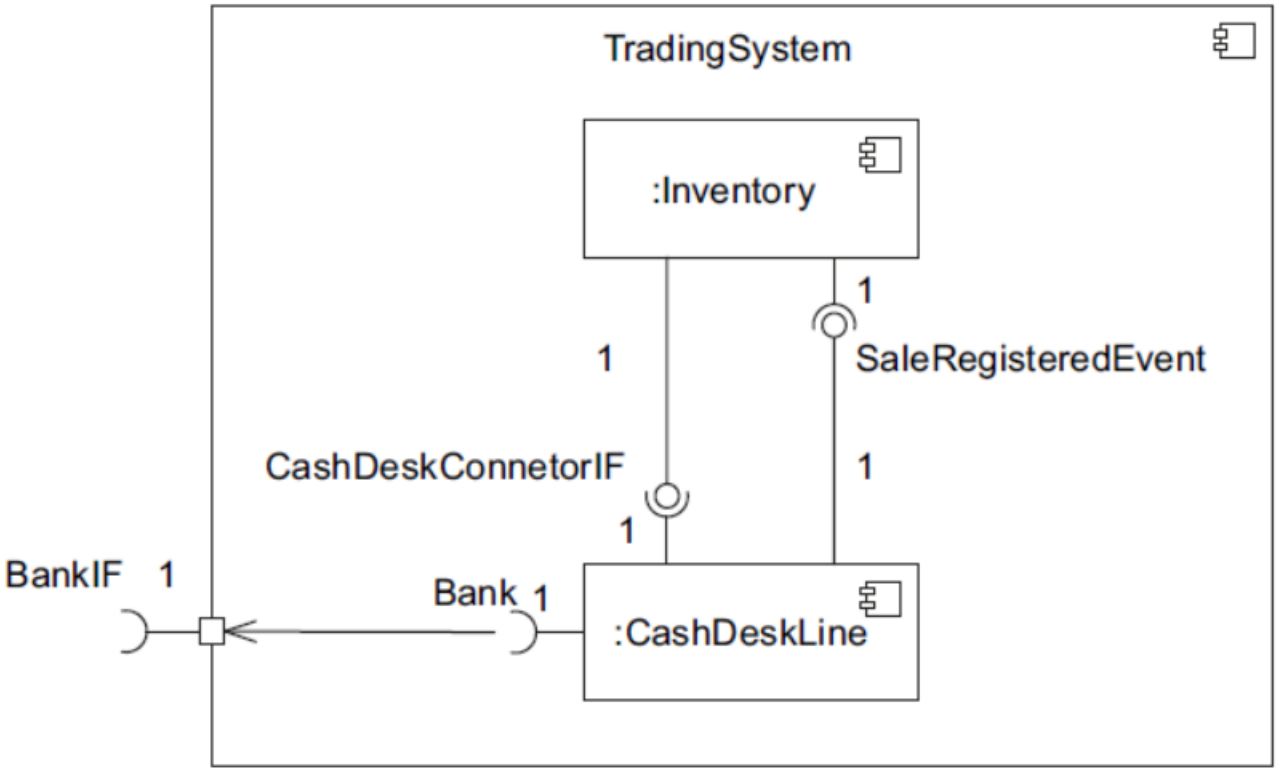
Grobaufbau – Schablonenartige Sichtenbeschreibung (4) Aufbau der Beschreibung

Zurich University of Applied Sciences
School of Engineering

- 5. Hintergrundinformationen:
  - Zum Verständnis des konkreten Aufbaus einer Sicht sind textuell beschriebene Hintergrundinformationen wichtig.
  - Begründen Sie damit beispielsweise konkrete Entwurfsentscheidungen.
  - Typische Hintergrundinformationen sind:
    - Begründungen für die gewählte Struktur oder die ausgewählte Alternative
    - Ergebnisse von Analysen oder Voruntersuchungen zu bestimmten inhaltlichen Systemaspekten

Beispiel Schablonenartige Sichtenbeschreibung (1)

- Kurzbeschreibung
  - Übersicht von CoCoME für den Betrieb einer Supermarktkasse.
  - Ein UML-Kompositionsstrukturdiagramm zeit die oberste Bausteinebene mit den Softwarebausteinen Inventory und CashDeskLine
- Diagramm



Beispiel Schablonenartige Sichtenbeschreibung (2)

Elementbeschreibung

- El

Element	Typ	Beschreibung
`TradingSystem`	UML-Komponente	Umrahmende UML-Komponente von CoCoME. Sie enthält ein Informationssystem zur Lagerverwaltung sowie ein eingebettetes System für die Kasse(n).
`Inventory`	UML-Part	Komponente für das Informationssystem zur Lagerverwaltung
`CashDeskLine`	UML-Part	. . .
`CashDeskConnectorIf`	Interface	Schnittstelle, über die die UML-Komponente `Inventory` mit der UML-Komponente `CashDeskLine` kommuniziert. Sie enthält Methoden, um Produktinformationen zu erhalten, wie z.B. Beschreibung und Preis. Der Barcode des Produkts dient hierfür als Eingabe.
`SaleRegisteredEvent`	Interface	. . .

- Variabilität
  - Kassensysteme sollen für verschiedene Installationen konfiguriert werden können
    - es gelten dann jeweils spezifische Plausibilitäten.

Beispiel Schablonenartige Sichtenbeschreibung (3)

- Hintergrundinformationen
  - Analysen:
    - Durch Prototypen wurde gezeigt, dass der Barcode-Scanner bei CoCoME eventuell eine Fehlerquelle darstellen könnte. Aus diesem Grund sind während der Entwicklung passende Tests durchzuführen und im Endsystem Maßnahmen für eine erhöhte Fehlertoleranz dieses Bereichs vorzusehen.
    - Ob die CoCoME-Software tatsächlich bis auf Ortsebene konfigurierbar sein soll, ist noch mit den zuständigen Stakeholdern zu klären. Es sind hier ggf. Maßnahmen zur prototypischen Evaluierung durch Testnutzer von CoCoME vorzusehen.
  - Annahmen:
    - Andere Bausteine von CoCoME lassen keine besonderen Fehlerquellen, Sicherheitsrisiken oder Leistungsengpässe erwarten.
  - Referenzen auf zugehörige Sichten:
    - Verfeinerte Bausteinsicht von CoCoME
    - Laufzeitsichten und Verteilungs- bzw. Infrastruktursicht

4.3.4 Kontextsicht oder Kontextabgrenzung (1)

- Bindeglied zwischen der textuellen oder grafischen Anforderungsbeschreibung und der späteren Architektur.
- beschreibt das Umfeld eines Systems und die Beziehungen bzw. Zusammenhänge mit diesem Umfeld und dient damit allen Beteiligten als Einstiegspunkt und Landkarte für das zu beschreibende System.
- Bei der Darstellung der Kontextsicht liegt somit der Schwerpunkt auf Schnittstellen zu den umliegenden Systemen (Nachbarsysteme).

Kontextsicht oder Kontextabgrenzung (2)

- Für die Kontextsicht sind folgende Elemente von Bedeutung:
  - externe Akteure (Nachbarsysteme und Benutzer);
  - das zu entwickelnde System selbst;
  - alle Schnittstellen zu externen Akteuren (alle Nachbarsysteme bzw. Benutzer) einschliesslich:
    - Art der Schnittstelle: z.B. Online, Batch, USB oder Datei sowie die über diese Schnittstelle übertragenen Daten oder Ressourcen sowie ggf. genutzte Services oder Funktionen;
    - verwendete Kommunikationsprotokolle;
    - verwendete Kommunikationsmuster, z.B. synchron, asynchron.

Kontextsicht oder Kontextabgrenzung (3)

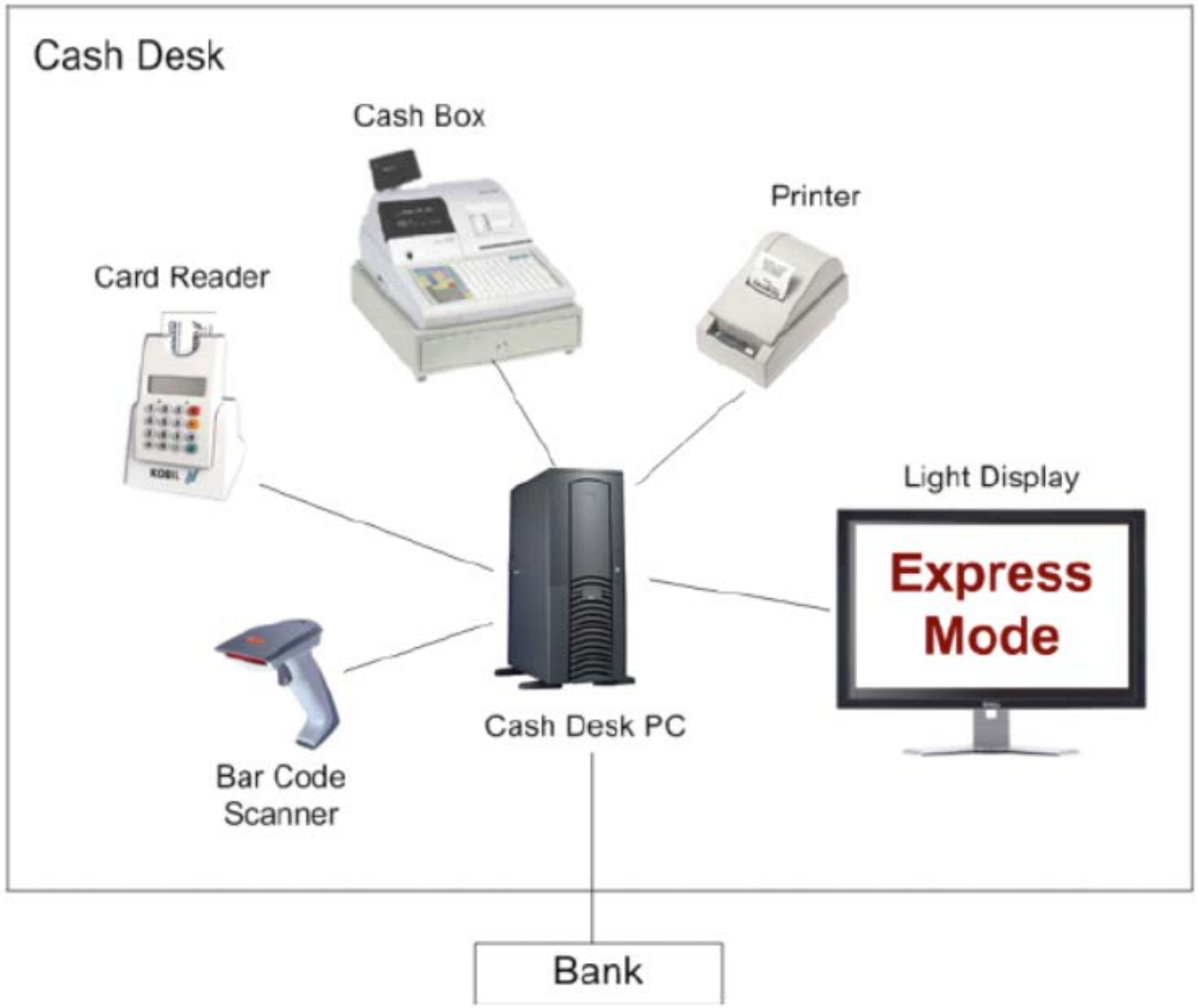
- Die Schnittstellen zu Nachbarsystemen gehören zu den kritischsten Aspekten eines Projekts.
- Entsprechend bedeutsam ist die Kontextsicht.
- Häufige Stakeholder dieser Sicht sind u.a.:
  - Projektleitung
  - Anforderungsanalysten (als »Input-Geber«)
  - Systemanalysten (als »Input-Geber«)
  - Fach- oder Domänenexperten (als »Input-Geber«)
  - Design und Entwicklung
  - Test
  - ggf. nachgelagert Administration bzw. Betrieb
  - Controlling (Kostenstellenzuordnung der Entwicklungskosten)
  - bei »Produkten« ggf. Vertrieb, Marketing

Kontextsicht oder Kontextabgrenzung (4)

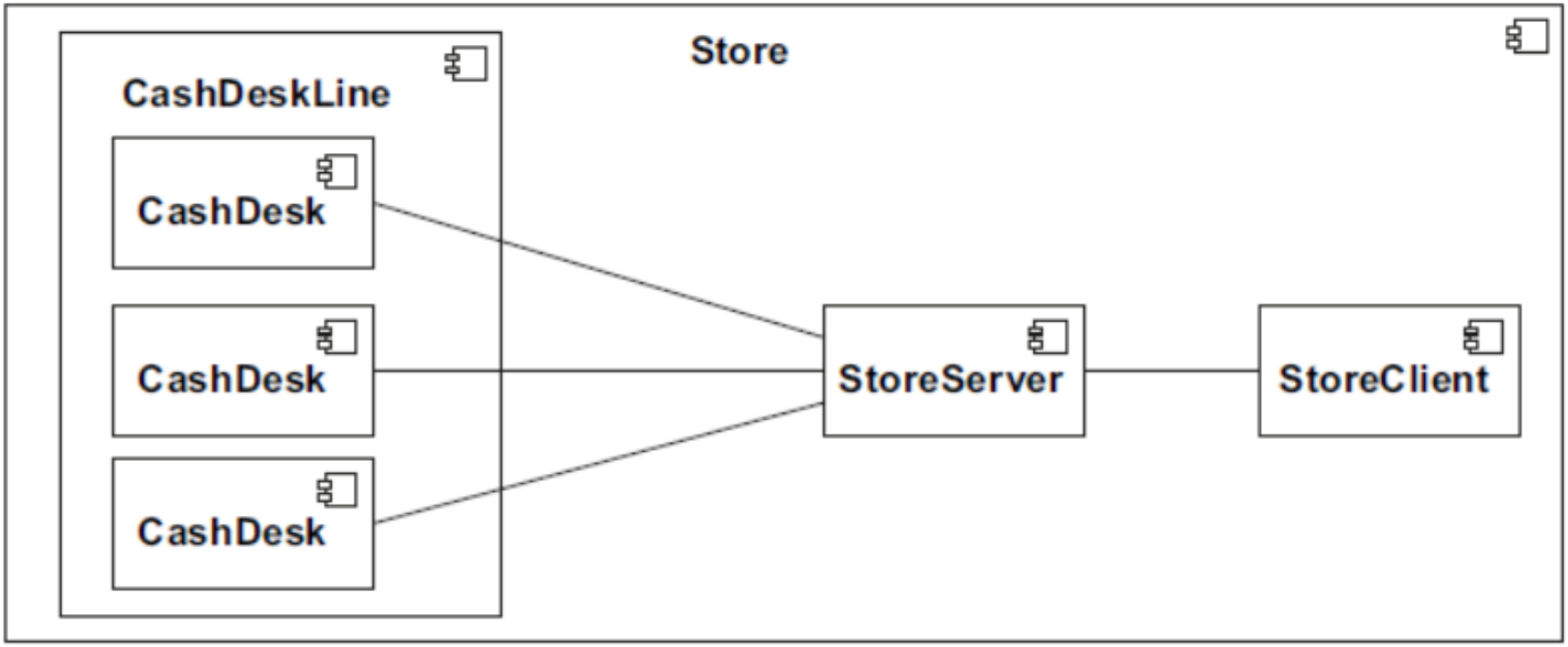
- Beschreibungen der Kontextsicht erfolgen vor allem durch
  - Kontextdiagramme und
  - Listen von Nachbarsystemen mit deren Schnittstellen.

Typ	Beschreibung
UML-Komponente UML-Part	UML-Komponenten und UML-Parts stellen als wesentliche Top-Level-Elemente Bausteine dar, für die klare (ggf. extern sichtbare) Schnittstellen sehr wichtig sind. Sie sind die wichtigsten Symbole der Kontextsicht.
UML-Knoten	Speziell für eine (ergänzende) Darstellung des (technischen) Verteilungs- bzw. Infrastrukturkontexts des zu beschreibenden Softwaresystems können auch UML-Knotensymbole genutzt werden. Speziell zur Verbindung von Knoten dürfen hier neben UML-Abhängigkeitsbeziehungen auch UML-Assoziationen genutzt werden.
UML-Akteur	Zur Darstellung des Bezugs des zu beschreibenden Softwaresystems zu wichtigen Benutzerrollen wird der Typ des UML-Akteurs genutzt.
Schnittstellen zur Außenwelt («Abhängigkeitsbeziehung«)	Diese dienen der Darstellung des Daten- oder Kontrollflusses zwischen den Systemen und Stakeholdern der Außenwelt. Verwenden Sie UML-Beziehungen («Abhängigkeiten«). Diese beinhalten bei Bedarf Informationen über Schnittstellenart, Kommunikationsprotokolle, Kommunikationsmuster und übertragene Objektart. Jede Schnittstelle sollte in der Kontextsicht einen aussagekräftigen Namen tragen.
Legende/Kommentar	Verbale Legenden bzw. Erläuterungen erscheinen als Kommentar im Diagramm.

Kontext eines Kassenarbeitsplatzes (für Stakeholder)



Kontext eines Kassenarbeitsplatzes (für Entwickler)



4.3.5. Bausteinsicht

Zurich University of Applied Sciences

zh aw School of Engineering

- Zeigt statische Strukturen eines Softwaresystems
- Zeigt Beziehungen zwischen den Bausteinen
- Zeigt die Zerlegung der Bausteine in Sub-Bausteine
- Ein Baustein sollte folgende Merkmale haben:
  - Name
  - Verantwortlichkeit bzw. Zweck
  - Schnittstelle
  - Verweis auf seine Implementierung
- Stakeholder der Bausteinsicht
  - alle an Architektur, Entwurf, Erstellung und Test der Software beteiligte Projektmitarbeiter,
  - zusätzlich die Qualitätssicherung
  - Dem Projektmanagement hilft die Bausteinsicht bei der Erstellung von Arbeits- oder Aktivitätsplänen.

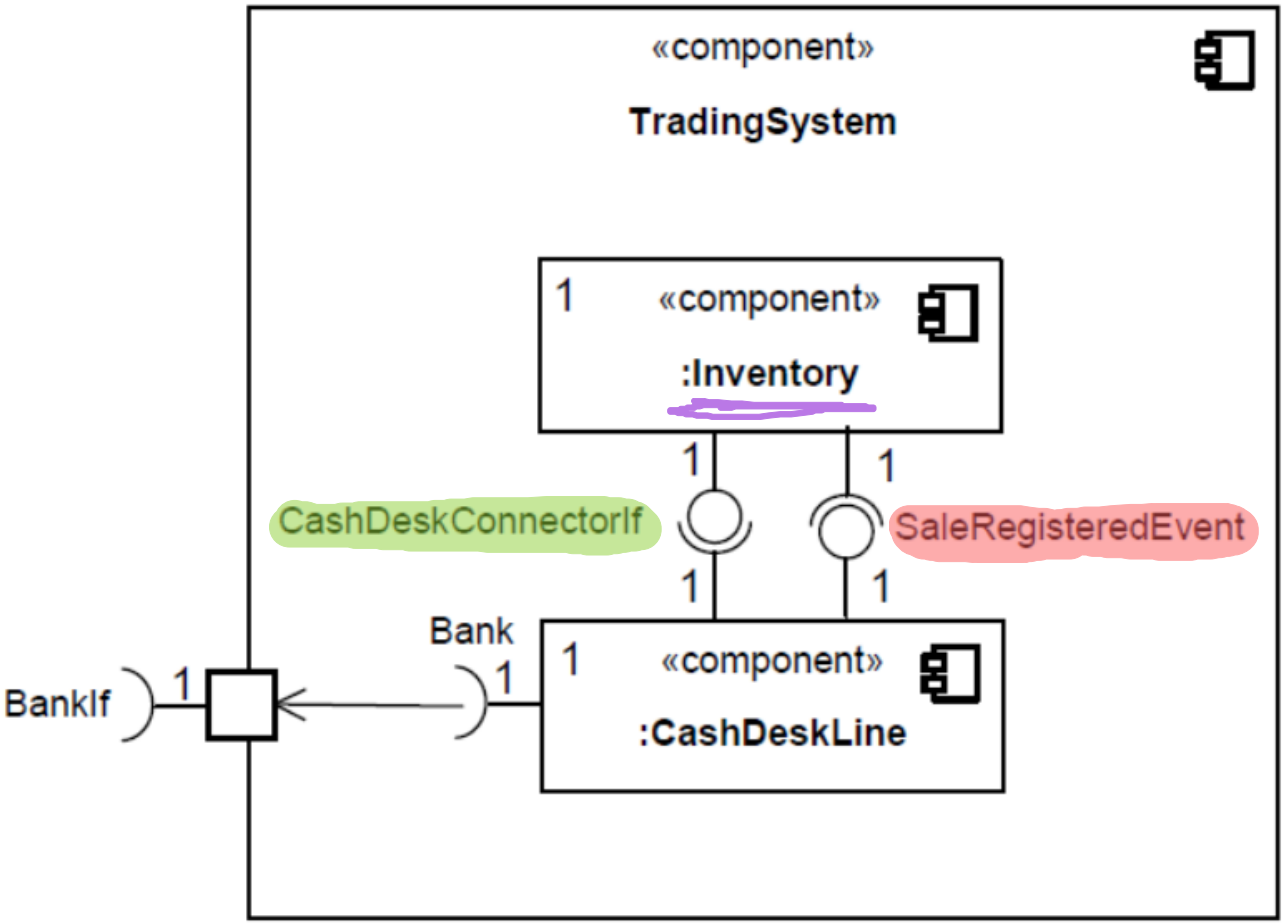
Beschreibungselemente der Bausteinsicht

- Vorallem UML Komponentensymbole mit Schnittstellen in Blackbox oder Whiteboxdarstellung

Typ	Beschreibung
UML-Komponenten (mit Schnittstellen)	UML-Komponenten stellen als wesentliches Top-Level-Element Bausteine dar, für die klare (ggf. extern sichtbare) Schnittstellen besonders wichtig sind.
UML-Pakete	UML-Pakete stellen eine logische Kapselung oder Abstraktion von anderen Elementen dar, die auf Architekturebene nicht genauer spezifiziert werden soll bzw. muss. Sie stellen gemeinsam mit Komponenten ebenfalls ein Top-Level-Element der Bausteinsicht dar.
UML-Klassen	UML-Klassen können (müssen jedoch nicht) ergänzend insbesondere in Verfeinerungsschritten als Bausteine zum Einsatz kommen.
UML-Beziehungen	UML-Beziehungen kommen zwischen UML-Komponenten, zwischen UML-Paketen oder zwischen UML-Klassen zum Einsatz.
Legende/Kommentar	Verbale Legenden bzw. Erläuterungen erscheinen als Kommentar im Diagramm.

Bausteinsicht Component TradingSystem

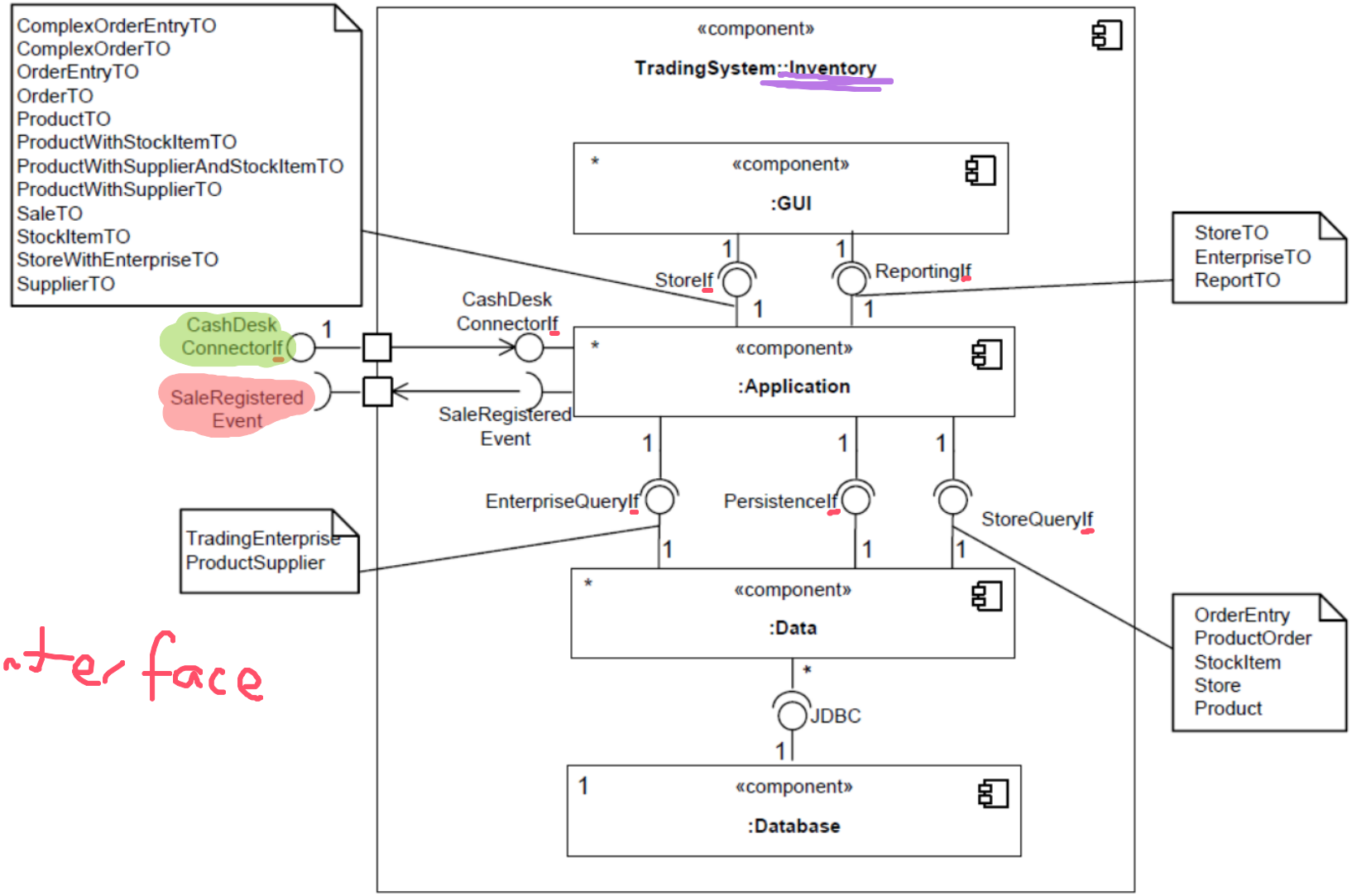
- In den nachfolgenden Folien wird eine (De-) Komposition der Komponente Inventory gezeigt



Baustein TradingSystem::Inventory

Zurich University of Applied Sciences

zh aw School of Engineering



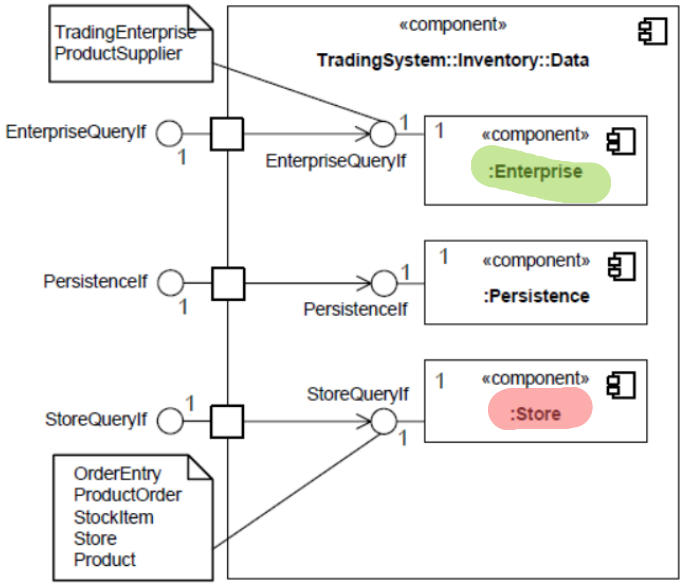
If = Interface

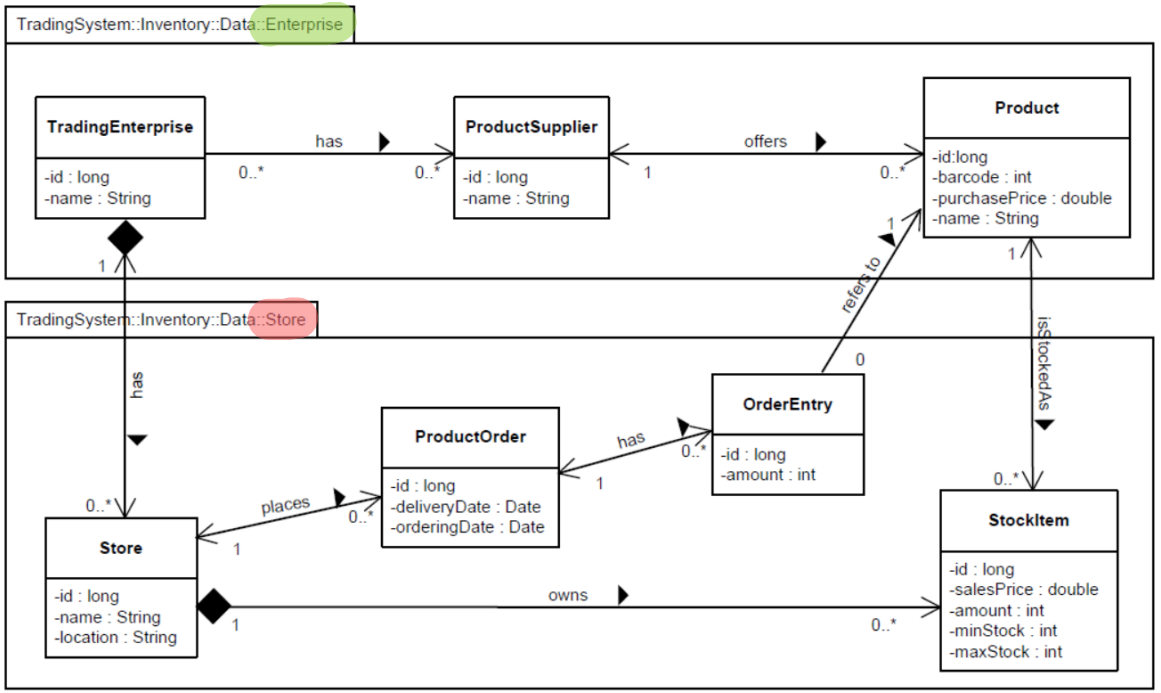
Baustein TradingSystem::Inventory::Data

Zurich University of Applied Sciences

School of Engineering

- Bausteinsicht (White Box)
- Datamodel:
  - TradingSystem::Inventory::Data::Enterprise
  - TradingSystem::Inventory::Data::Store



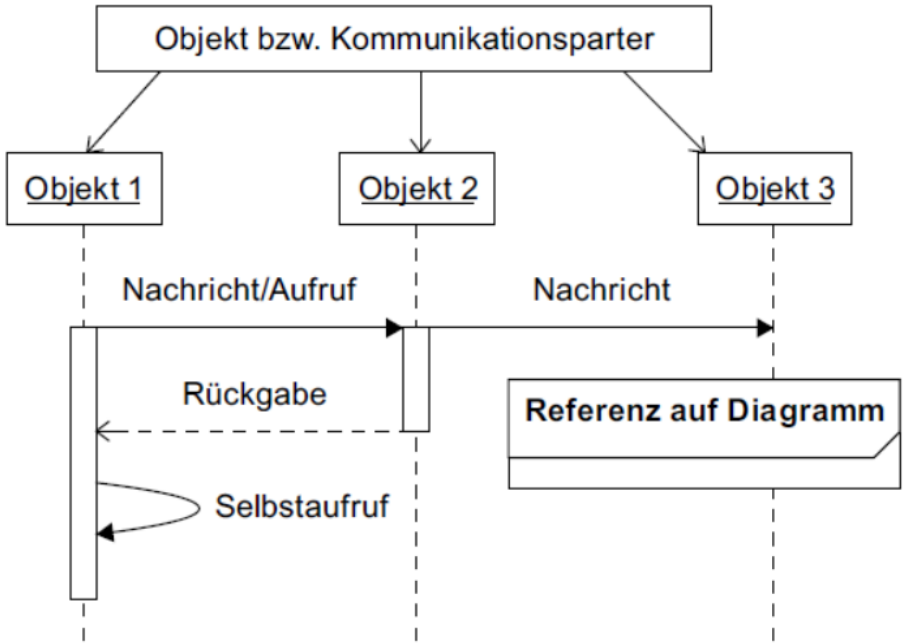


4.3.6 Laufzeitsicht

- Die Laufzeitschicht beschreibt das Zusammenwirken der Bestandteile des Softwaresystems zur Laufzeit
- Berücksichtigung von Aspekten wie
  - Systemstart
  - Laufzeitkonfiguration
  - Administration des Systems
- Beschreibung der Laufzeitsicht für ausgewählte Use Cases

Laufzeitsicht (Stakeholder und Elemente)

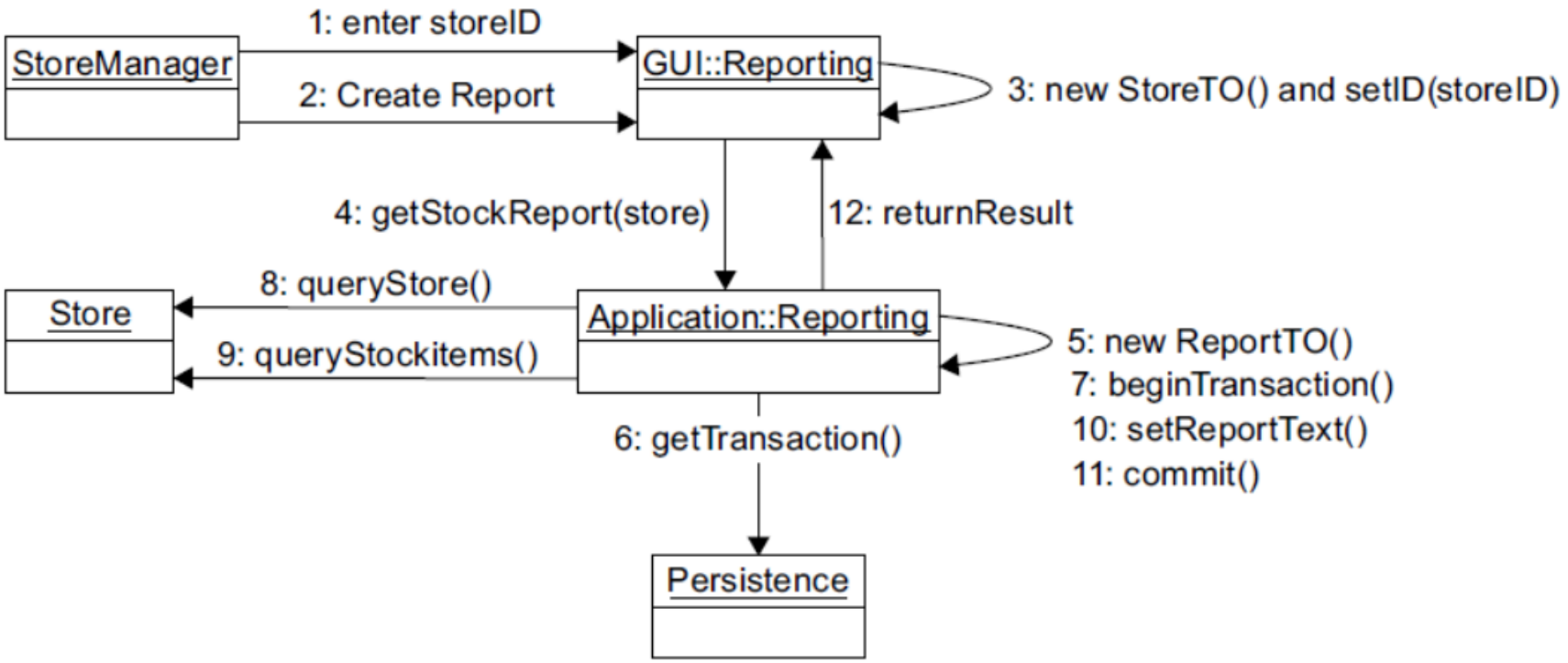
- Zielgruppe für die Laufzeitsicht
  - Entwickler, Tester, Systemarchitekten
- Beschreibungselemente
  - UML Aktivitätsdiagramme, Kommunikationsdiagramme und Sequenzdiagramme
  - Flussdiagramme, Pseudocode oder verbale Beschreibung



Wichtige Elemente eines Sequenzdiagramms

Typ	Beschreibung
Objekt/Kommunikations-partner	Waagerechte Rechtecke stellen allgemein Objekte bzw. Kommunikationspartner in Sequenzdiagrammen dar. Solche Kommunikationspartner können u.a. UML-Komponenten, z.B. aus der Bausteinsicht, oder auch Klassen sein.
Senkrechte gestrichelte Linien mit Rechtecken	Senkrechte gestrichelte Linien mit Rechtecken geben Lebenslinien der Objekte bzw. Kommunikationspartner an.
Pfeile	Verschiedene Aufrufarten bzw. Nachrichten zwischen Kommunikationspartnern, z.B. synchrone und asynchrone Methodenaufrufe, werden durch Pfeile gekennzeichnet.
Diagrammreferenzen	Bei größeren Abläufen kann es sinnvoll sein, Referenzen auf andere Diagramme zu nutzen.
Schleifen/Bedingungen	Schleifen und Bedingungen sind gebräuchliche Kontrollablaufstrukturen. Sie kommen oft erst bei Verfeinerungen der oberen Architekturebene sinnvoll zum Einsatz.
Legende/Kommentar	Verbale Legenden bzw. Erläuterungen werden als Kommentar im Diagramm hinterlegt.

Kommunikationsdiagramm



4.3.7 Verteilungssicht

- Zeigt die technische Umgebung in der ein Softwaresystem abläuft
  - Komponenten aus der Bausteinsicht werden in Knoten der Verteilungssicht platziert
  - Beschreibung enthält auch Systemsoftware, Hardwarebausteine, Application Server, DBMS, Netzwerkverbindungen
- Artefakte wie .war, .ear, .jar, exe werden den Ausführungsknoten zugeteilt wie Server, Device, etc.
- Stakeholder
  - Betreiber des Systems
  - Systemarchitekten
  - Entwickler
  - Tester

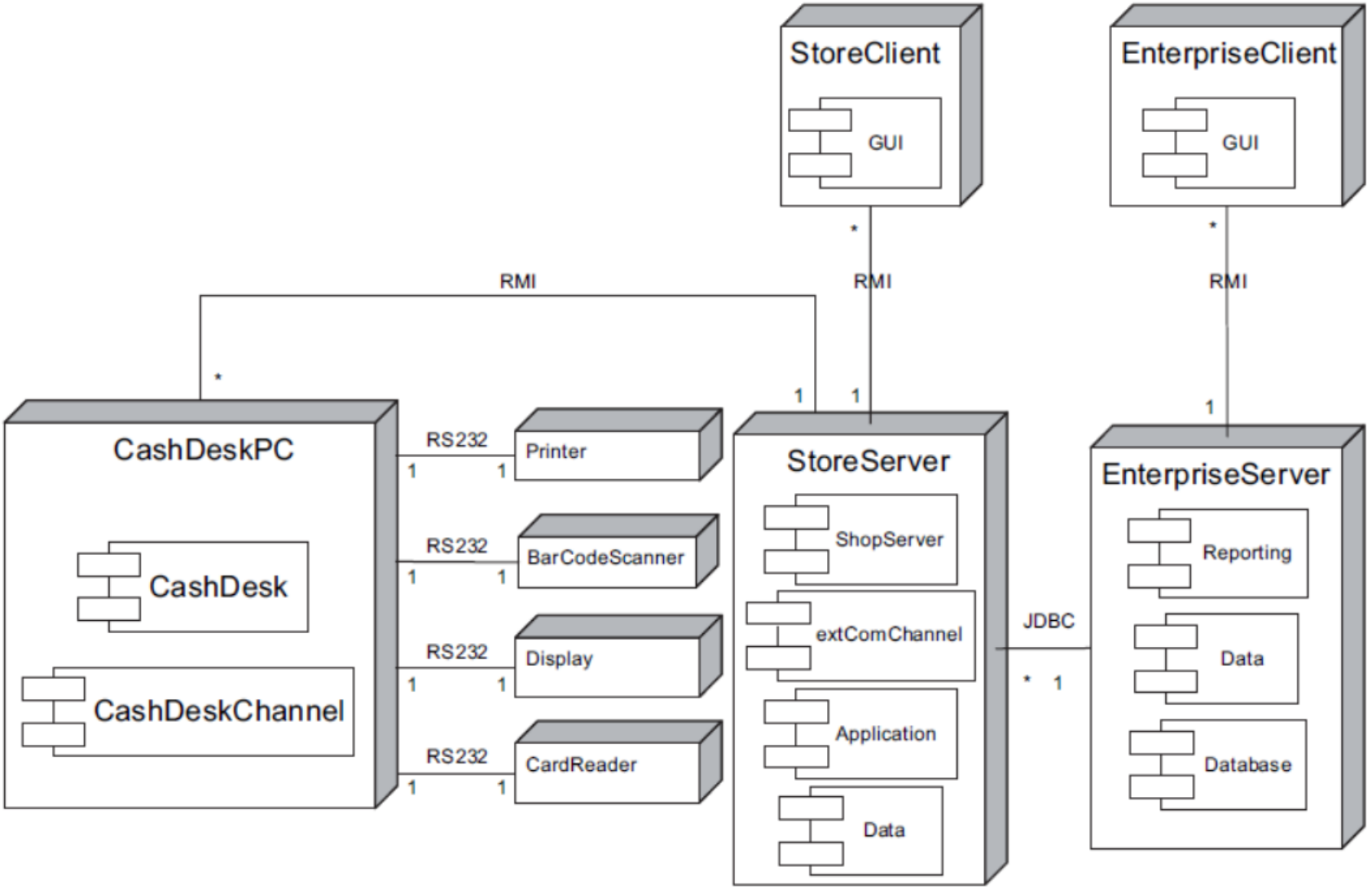
Beschreibungselemente

- Meistens UML Deployment Diagramme

Typ	Beschreibung
UML-Knoten	UML-Knoten sind das zentrale Element dieser Sicht. Auf bzw. in ihnen laufen andere Bausteine ab.
Kanäle/Kommunikations-pfade	Kanäle bzw. Kommunikationspfade zwischen Knoten werden als reguläre UML-Assoziationen dargestellt. Sie können die Verbindung näher beschreiben, z.B. »1-Gbit/s-Ethernet« oder »Physische CD«.
UML-Komponente	UML-Komponenten werden typischerweise auf Knoten platziert, um darzustellen, dass sie dort z.B. in einem Application Server unter UNIX ablaufen.
UML-Paket	UML-Pakete dienen zur Darstellung von Gruppen, Mengen oder Strukturen von Bausteinen, die analog zu UML-Komponenten auf bzw. in Knoten platziert werden. Dies könnten z.B. EAR-»Deploy-ables« aus Java EE sein. →

Typ	Beschreibung
UML-Abhängigkeits-beziehung	Diese dienen der Darstellung von Beziehungen zwischen UML-Knoten.
Legende/Kommentar	Verbale Legenden bzw. Erläuterungen erfolgen als Kommentar im Diagramm.

Beispiel CoCoME



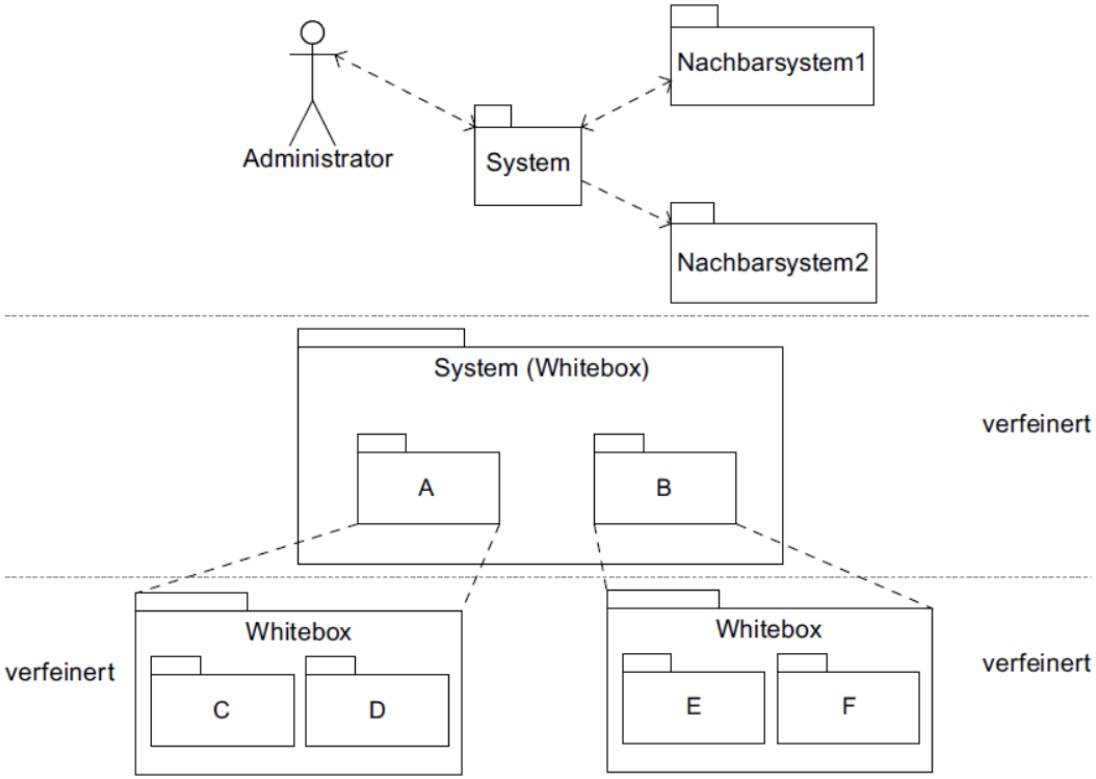
4.3.8 Wechselwirkung Architektursicht

- Der Entwurf einer Architektursicht beeinflusst andere Sichten.
  - Änderungen einer Sicht ziehen Anpassungen anderer Sichten nach sich.
  - Deshalb iterativer Entwicklungsprozess an, bei dem nach jeder Änderung alle abhängigen Sichten aktualisiert werden
- Wechselwirkungen dokumentieren weil:
  - Entwurfsentscheidungen werden nachvollziehbar.
  - Auswirkungen von Änderungen werden leichter erkennbar.
  - Zusammenhänge zwischen Systemteilen werden leichter verständlich.
- Startpunkt sind die Kontextsichten, die das Softwaresystem als Blackbox mit seinen Nachbarn interagieren lassen.
  - Anschliessend Baustein und Laufzeitsicht ableiten
  - In der Verteilungssicht werden Elemente der Bausteinsicht platziert

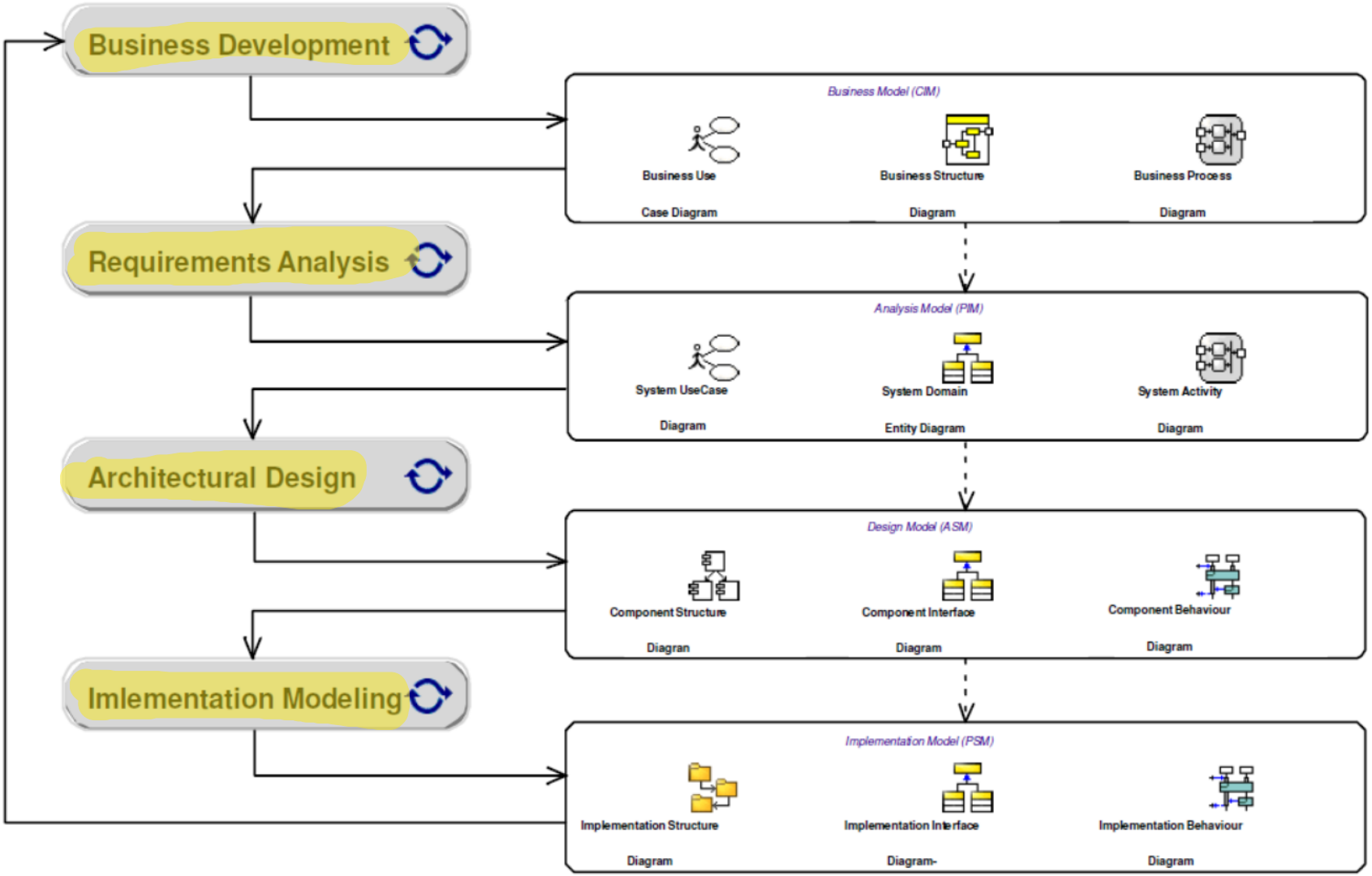
4.3.9. Hierarchische Verfeinerung von Sichten

Zurich University of Applied Sciences
School of Engineering

- Hierarchie beginnt mit der Kontextsicht in der Blackbox Darstellung (oberste Ebene)
  - Dokumentation der externen Schnittstellen
  - Anschliessend Verfeinerung als Whitebox Darstellung



Verfeinerung im Rahmen des SW-Prozesses



4.4. Technische oder querschnittliche Konzepte

Zurich University of Applied Sciences
School of Engineering

- Technische oder querschnittliche Konzepte umfassen:
  - Ablaufsteuerung
  - Fehlerbehandlung
  - Integration
  - Internationalisierung
  - Persistenz
  - Verteilung
- Herausforderungen
  - Z.B. Fehlerbehandlung: Durchgängige Anwendung
  - Sicherheit: vielschichtiges Thema
    - in Bezug auf Authentifizierung und Autorisierung
    - Schutz vor Angriffen

4.5. Architektur und Implementierung

- Aus der schrittweisen hierarchischen Verfeinerung soll ablauffähiger Code entwickelt werden
  - Wo ist die Schnittstelle zwischen Architektur und Entwurf?
  - … weiter wichtig: Zusammen mit Team Definition von Code Conventions, Gestaltung der Entwicklungsumgebung, Build Richtlinien, etc
- Beispiel CocoME
  - Design: Abbildung der Struktur in Packages, Schnittstellen sollen sich im Package befinden, Implementierungsklassen in einem Subpackage
  - Build: Ant verwenden mit Build Datei build.xml-> CoCoME.jar
  - Run: CoCoME.jar kann über die Konsole gestartet werden, in .properties können Einstellungen vorgenommen werden.

4.6 Übliche Dokumententypen (1)

- Die Beschreibung der Architektur kann in diversen Artefakten erfolgen
  - Architekturüberblick -> Kurzfassung
  - Dokumentenübersicht -> Verzeichnis aller architekturrelevanten Dokumente
  - Übersichtspräsentation -> Foliensatz
  - Architekturtapete -> 1-n Poster
  - Handbuch zur Doku -> Struktur der Doku wird erläutert
  - Dokumentation der externen Schnittstellen
  - Templates

Übliche Dokumententypen (1)

- Zentrale Architekturbeschreibung
  - Aufgabenstellung, Ziele (Vision), Qualitätsanforderungen und Stakeholder
  - Technische und organisatorische Rahmenbedingungen
  - Sichten, Entscheidungen, verwendete Muster
  - Technische Konzepte
  - Qualitätsbewertungen
  - Erkannte Risiken
  - usw.

4.7. Praxisregeln zur Dokumentation

- Regel 1: »Schreiben aus der Sicht des Lesers«
- Regel 2: »Unnötige Wiederholung vermeiden« - DRY
- Regel 3: »Mehrdeutigkeit vermeiden«
- Regel 4: »Standardisierte Organisationsstruktur bzw. Schablonen«
- Regel 5: »Begründen Sie wesentliche Entscheidungen schriftlich«
- Regel 6: »Überprüfung auf Gebrauchstauglichkeit«
- Regel 7: »Übersichtliche Diagramme«
- Regel 8: »Regelmässige Aktualisierungen«

4.8. Beispiele weiterer Architektur-Frameworks

- Bisher angeschaut:
  - ANSI/IEEE 1471-2000/ISO/IEC 42010:2011,
  - pragmatischen iSAQB-Ansatz (arc42)
- Weitere Architektur-Frameworks zur Beschreibung von Softwarearchitekturen
  - 4+1 (Kruchten)
  - Department of (US) Defense Architectural Framework (DoDAF)
  - Model Driven Architecture der (OMG-MDA)
  - RM-ODP (Reference Model of Open Distributed Processing,)
  - Standards und Architekturen für E-Government-Anwendungen (SAGA) als Framework der deutschen Bundesverwaltung
  - SAPs-EAP - Enterprise Architecture Framework
  - Enterprise: The Open Group Architecture Framework (TOGAF®)
  - Enterprise: Zachman Framework (IBM)

Fragen und Zusammenfassung

- 4.3. Sichten und Schablonen
- 4.4. Hierarchische und querschnittliche Konzepte
- 4.5. Architektur und Implementierung
- 4.6. Übliche Dokumententypen zu Softwarearchitekturen
- 4.7. Praxisregel zur Dokumentation
- 4.8. Beispiele weitere Architekturframeworks

Zurich University
of Applied Sciences

zh
aw

School of
Engineering