

Kapitel 6: software follows function - Softwareentwicklung nach industriellen Maßstäben

Prof. Dr.-Ing. Frank Thomas



Themenschwerpunkte

R

Zukunftsorientierte IT-Integration in der Logistik

Kapitel 1:

Warenidentifikation - Anwendung in der Logistik

Kapitel 2:

Datenkommunikation in der Intralogistik

Kapitel 3:

Systemarchitektur für Intralogistiklösungen / Modularisierung von Förderanlagen

Kapitel 4:

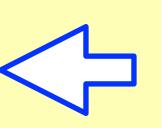
Gestaltung und Einsatz innovativer Material-Flow-Control-Systeme (MFCS)

Kapitel 5:

Transparenz und Vernetzung der Geschäftsprozesse

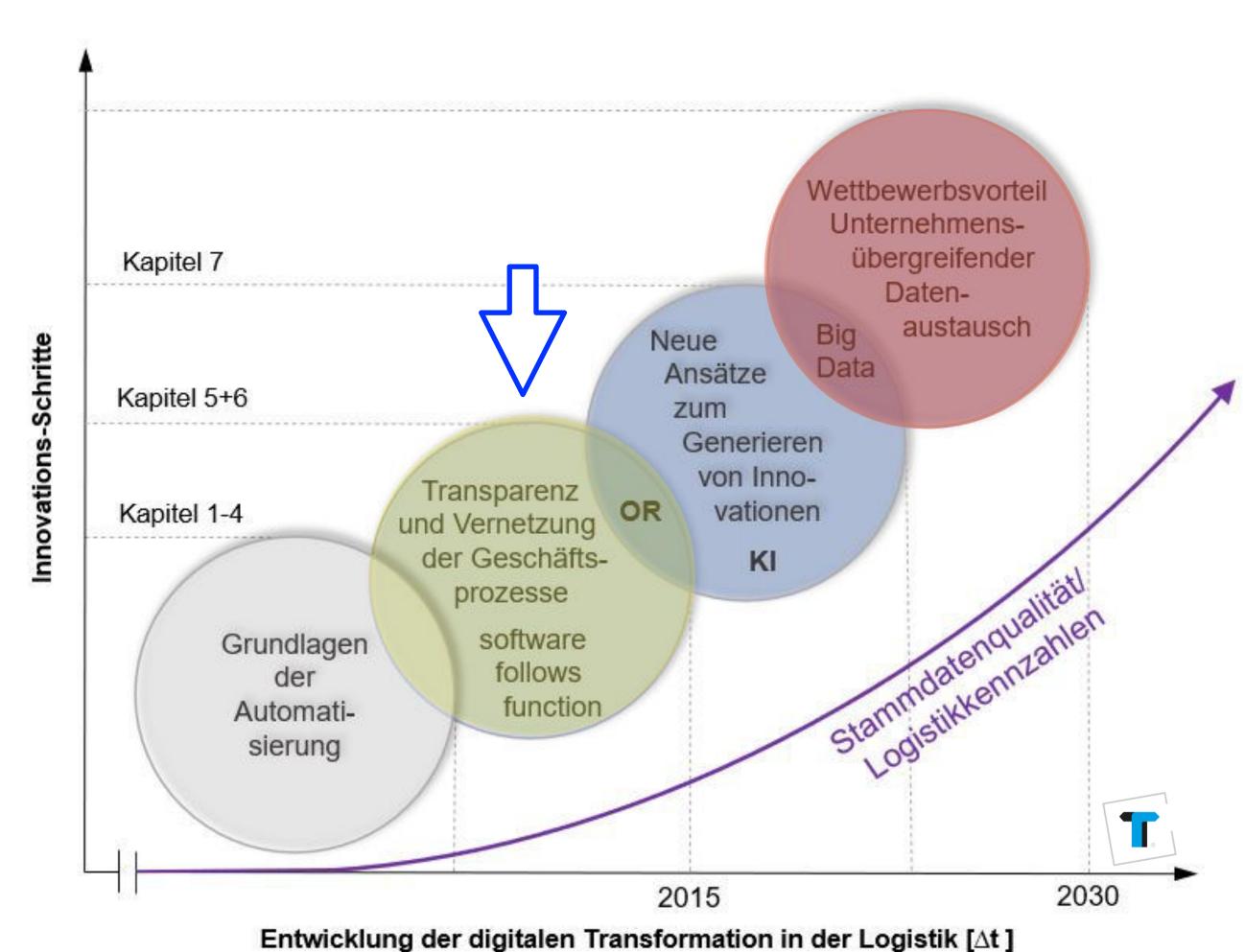
Kapitel 6:

software follows function - Software-Entwicklung nach industriellen Maßstäben



Kapitel 7:

Neue Ansätze zum Generieren von Innovationen



Lösungsansatz Adaptive IT

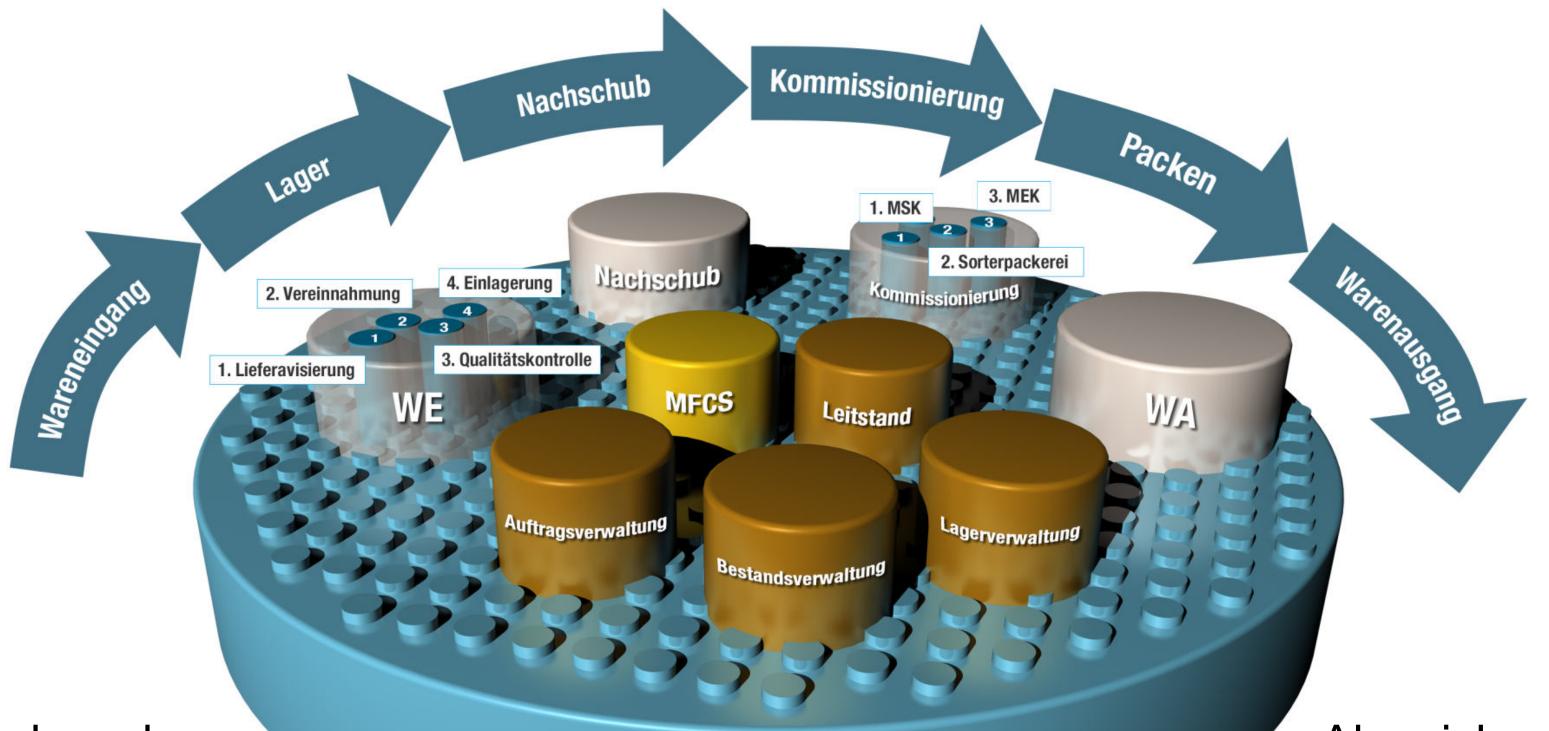


In den vorausgegangenen Kapiteln wurde mit der Herleitung der Geschäftsprozessmodule die Basis für die Entwicklung wiederverwendbarer adaptiver IT-Prozessbausteine geschaffen.

Komponenten-Architektur (Adaptive Prozessbausteine)

Veredelung der Standardprozesse auf neue Anforderungen





Immer wiederkehrende Geschäftsprozesse werden mit Bausteinen der Standardmodule abgebildet.

MIDDLEWARE: TUP STANDARD

Abweichende Geschäftsprozesse werden über Plugins und Veredelungsmodule abgebildet.

Im Skript Abbildung 5.3: Bestandteile der Softwarearchitektur einer adaptiven IT-Lösung

Definition Softwarearchitektur nach IEEE 1471



Wiederverwendbarkeit, Änderbarkeit und Erweiterbarkeit eines Softwaresystems wird durch die Softwarearchitektur bestimmt.

Die Softwarearchitektur ist die grundlegende Organisation eines Systems und wird durch ...

- ☐ ihre Komponenten
- ☐ die Beziehungen untereinander und zur Umgebung
- udie Prinzipien, die den Entwurf und die Evolution leiten,

verkörpert.

Norm zur Architekturbeschreibung von Software-Systemen (freigegeben 21. September 2000)

Software-Technik



In der Softwaretechnik sind unter anderem zwei Entwicklungen festzustellen:

☐ Komplexität der Software:

Der Trend geht dahin, dass Software immer komplexer wird.

Daraus folgt eine zunehmende Bedeutung der Softwarearchitektur.

Eine "gute" Softwarearchitektur schafft Transparenz.

□ Software muss sich an ständige Veränderungen anpassen:

Neue Marktanforderungen oder Kundenwünsche müssen während oder nach der Entwicklungsphase ohne großen Aufwand umsetzbar sein

(Entwicklungsprozess mit objektorientierten Werkzeugen).

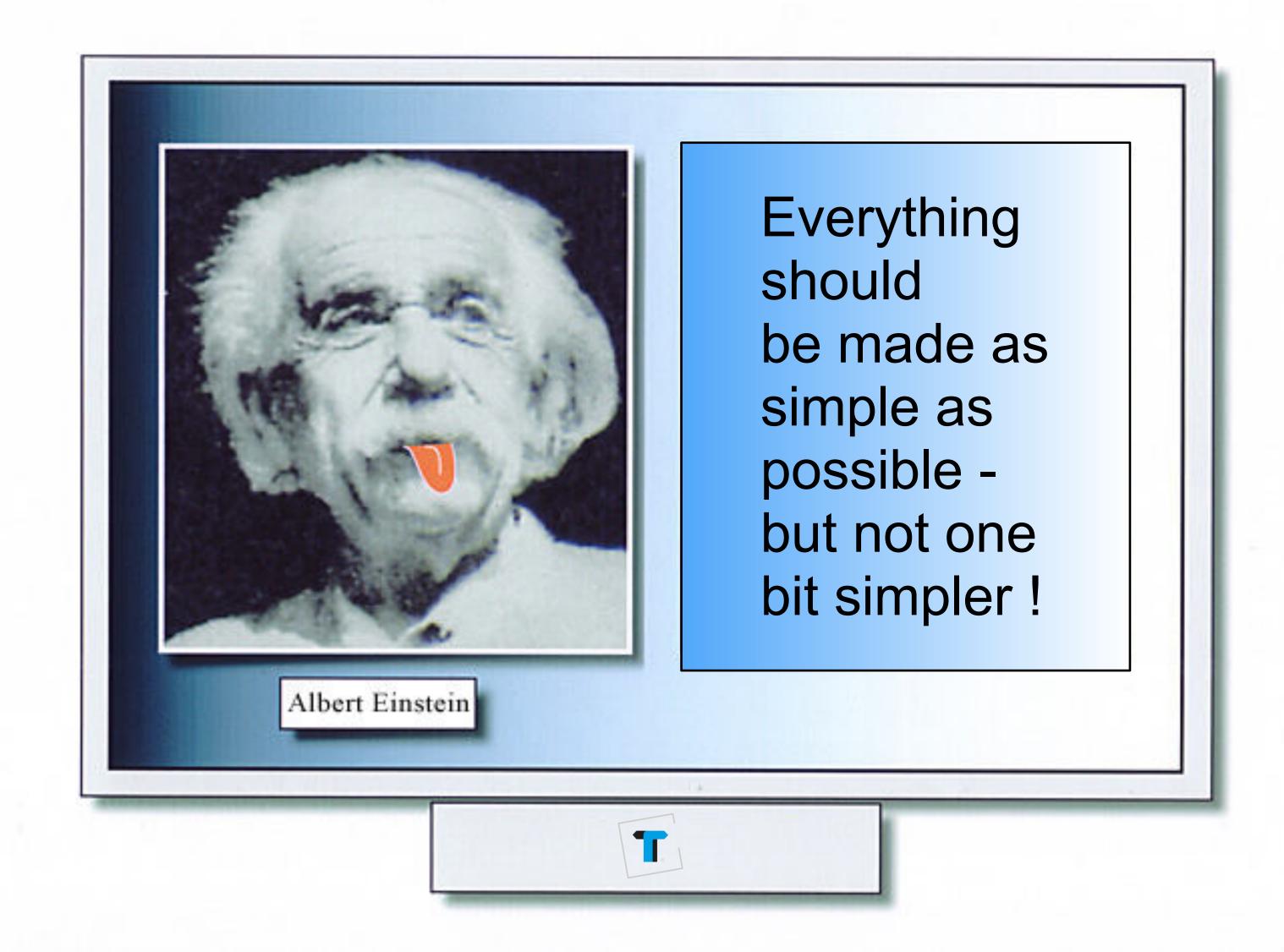
Es kann sonst ein "Big Ball of Mud" entstehen, eine gewucherte

Software. Eine "gute" Software wirkt dem entgegen.

Komplexität beherrschen



Hier hilft der Ansatz:



Objektorientierung



In der Softwareentwicklung werden objektorientierte Methoden umgesetzt:

- □ Zur Verbesserung von Produktivität, Wartbarkeit und der Software-Qualität
- Damit adaptive IT-Prozessbausteine möglichst oft wiederverwendet werden können (Objektorientierte Framework).

Die entwickelten Objekte sollen die "reale Welt" abbilden.

A warehouse is not a warehouse



Die Informatik sorgt nicht für das Verständnis des Problems, sondern gibt Methoden an, auf die dann jedoch die Logistiker angewiesen sind, um ihre Kerngeschäftsprozesse eines WMS einer Lösung zuzuführen.

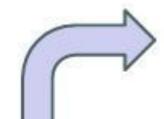
software follows function



Software follows function gilt dann, wenn in der Planungsphase alle Projektanforderungen dokumentiert wurden und gemeinsam im interdisziplinären Team aus Logistik-Planern, von dem Kunden/Nutzer und dem IL (Implementierungs-Leiter) unterschrieben wurden. Das Know-How des IL wird frühzeitig in die Prozessgestaltung und Anforderungsaufnahme mit einbezogen.

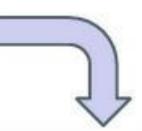
Agile Vorgehen Softwareentwicklung nach industriellen Maßstäben in der IT **Planungs-Analyse-**Lasten-Daily **Entwicklungs-**Stand Up phase und heft, unteraufgaben Deployment: alle Konstruktions-1 Woche Ziele, schriebenes zu den Anforderungen phase **Anforder-Pflichtenheft** Prozesssind dokumentiert ungen schritten Entwicklungs-Kapitel 6.2.3 Kapitel 5.11 Team IL+PL PL, Intergeben IL, IT-Stake-Stakedisziplinäres **Terminplan Entwicklungs**holder holder Team entlang der team + IL + PL **Prozess-**IL Aufgabenpakete fortschritte* während des Deployments **Projektlaufzeit Detaillierter** Inkrement inkl. **Terminplan Iteratives Vorgehen** Ziel **Endtermin** angepasst am **Termin** Release als **Detaillierte Projektfortschritt Einzuplanendes** Ticket im Jira Projektkosten **Budget** Kunde testet und gibt direkt software follows function! Planungsphase Logistik Feedback

Risikokarte



Mangelhafte Beschreibung der erforderlichen Leistungen zur Erreichung der Ziele des Projektes

. .





Personal & Management Ungenügende Mitarbeiterfluktuation Benutzerbeteiligung Kommunikations-Unzureichende Unterstützung durch risiko außerhalb des Projekts Auftraggeber Ungenügende Machtkämpfe Erfahrung des **Projektleiters** Kommunikations-Unpassendes Knowrisiko innerhalb des how der Mitarbeiter Projekts (Einbindung IT zu spät)

Problematische	Unterschiedliche
Eigentumsver-	Ziele der
hältnisse	Verantwortlichen
Nichtbeachten der	Kein eindeutig
Unternehmens-	definiertes
struktur	Vorgehensmodell







Budgetproblematik

Projektleiterwechsel/

Projektleiterausfall

Risikomanagement in der Pflichtenheft-Phase



Die linken Felder "Personal & Management" "Umgebung" und "Anforderung" bilden die Vorgängerrisiken, während der rechte Block möglicher Nachfolger aufdeckt.

Das alleinige Aufdecken von Risiken ist noch nicht ausreichend, um ein erfolgreiches Risikomanagement zu implementieren.

Faktoren, die den Projekterfolg beeinträchtigen Mangelnde Kommunikation



Warum scheitern SW-Entwicklungsprojekte?

unvollständige Anforderungen 13,1 %

Einbeziehung von Benutzern 12,4 %

Erwartungen 9,9 %

Change Requests 8,7 %

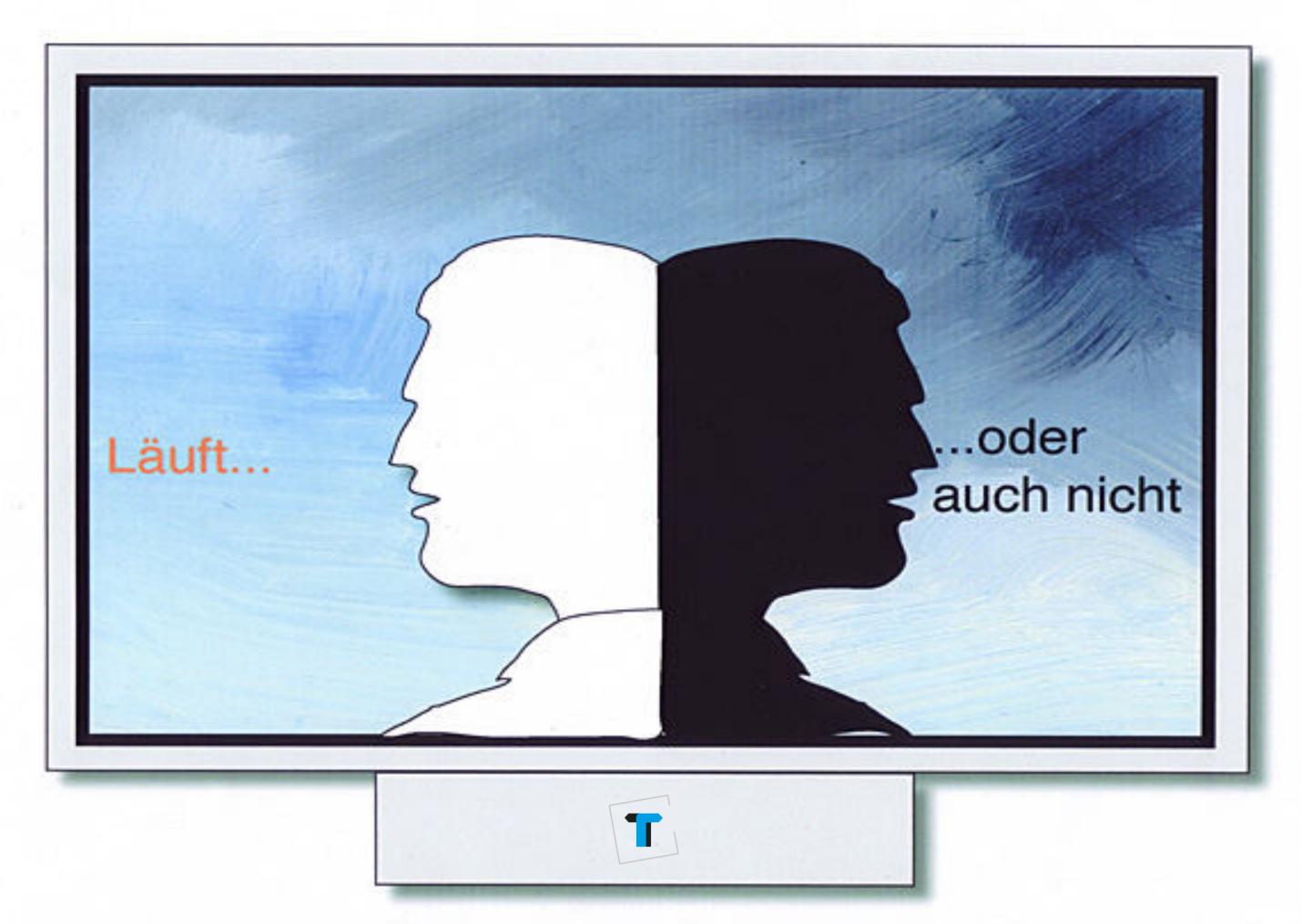
in Summe: 44,1 %

44,1 % der Faktoren, die den Projekterfolgt beeinträchtigen, hängen direkt oder indirekt mit Anforderungen zusammen.

Quellen:
TÜV Rheinland / Berlin / Brandenburg
Standish Group

Neue Software - Belastungsprobe für Unternehmen!





Software wird zunehmend zu einem Synonym für Scheitern

Qualität der Software-Architektur



Als Maß für die Qualität der Software-Architektur gelten:

☐ Mittelbares Maß:

Performance, Sicherheit, Verfügbarkeit und Zuverlässigkeit, Robustheit, Funktionsumfang, Benutzbarkeit.

☐ Unmittelbares Maß:

Flexibilität, Testbarkeit, Integrierbarkeit, Wartbarkeit, Änderbarkeit, Portierbarkeit, Skalierbarkeit, Wiederverwendbarkeit.

Frage nach der richtigen Software-Architektur!

Softwaretechnik (I)



Helmut Balzert definiert Softwaretechnik als zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.

Zielorientiert bedeutet die Berücksichtigung von:

Kosten, Zeit und Qualität.

Softwaretechnik (II)

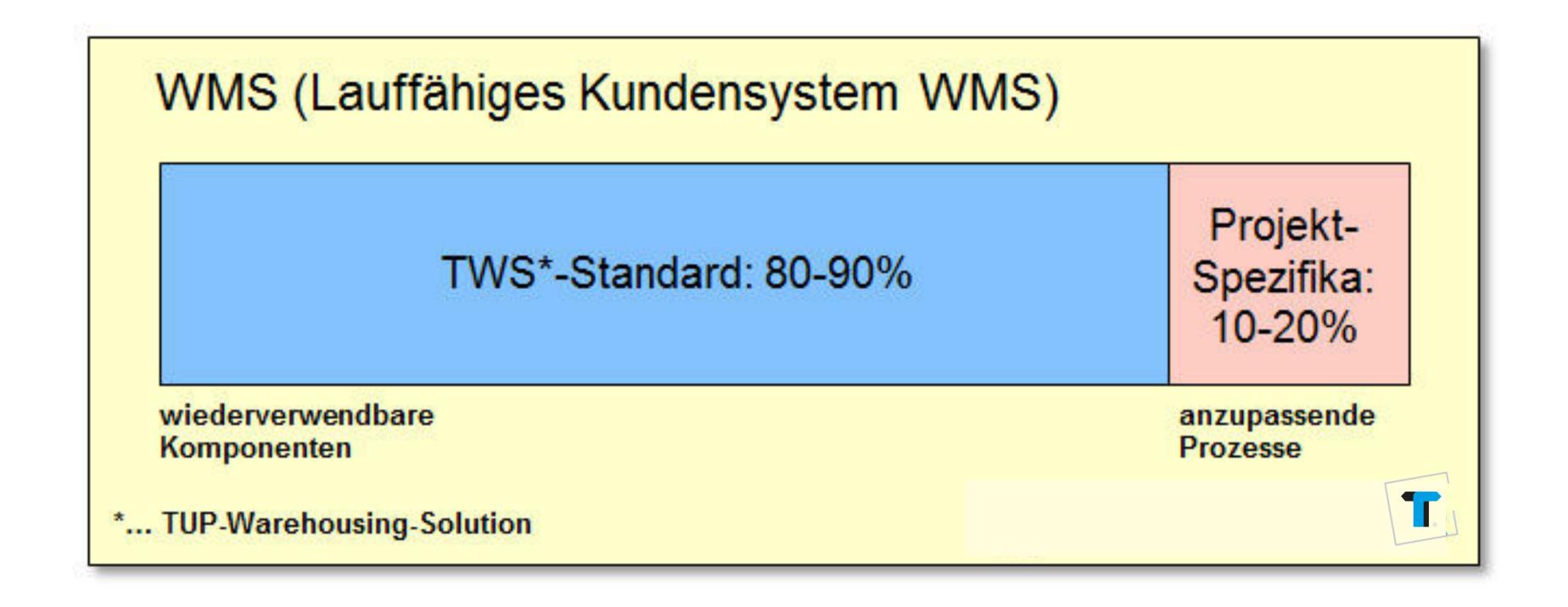


Zwei Eigenschaften einer adaptiven IT-Lösung sind für ein lauffähiges Kundensystem notwendig

- ☐ die Wiederverwendung von IT-Prozessbausteinen
- und die Anpassung "nicht"-gleichartiger Geschäftsprozesse durch Veredelung initiiert sofort die Frage nach der richtigen Softwarearchitektur

Präferierte Zielvorstellung für ein Lauffähiges Kundensystem







Frage nach der richtigen Software-Architektur

Entwicklungsprozess mit objektorientierten Werkzeugen



Entwicklungsprozess mit objektorientierten Werkzeugen:

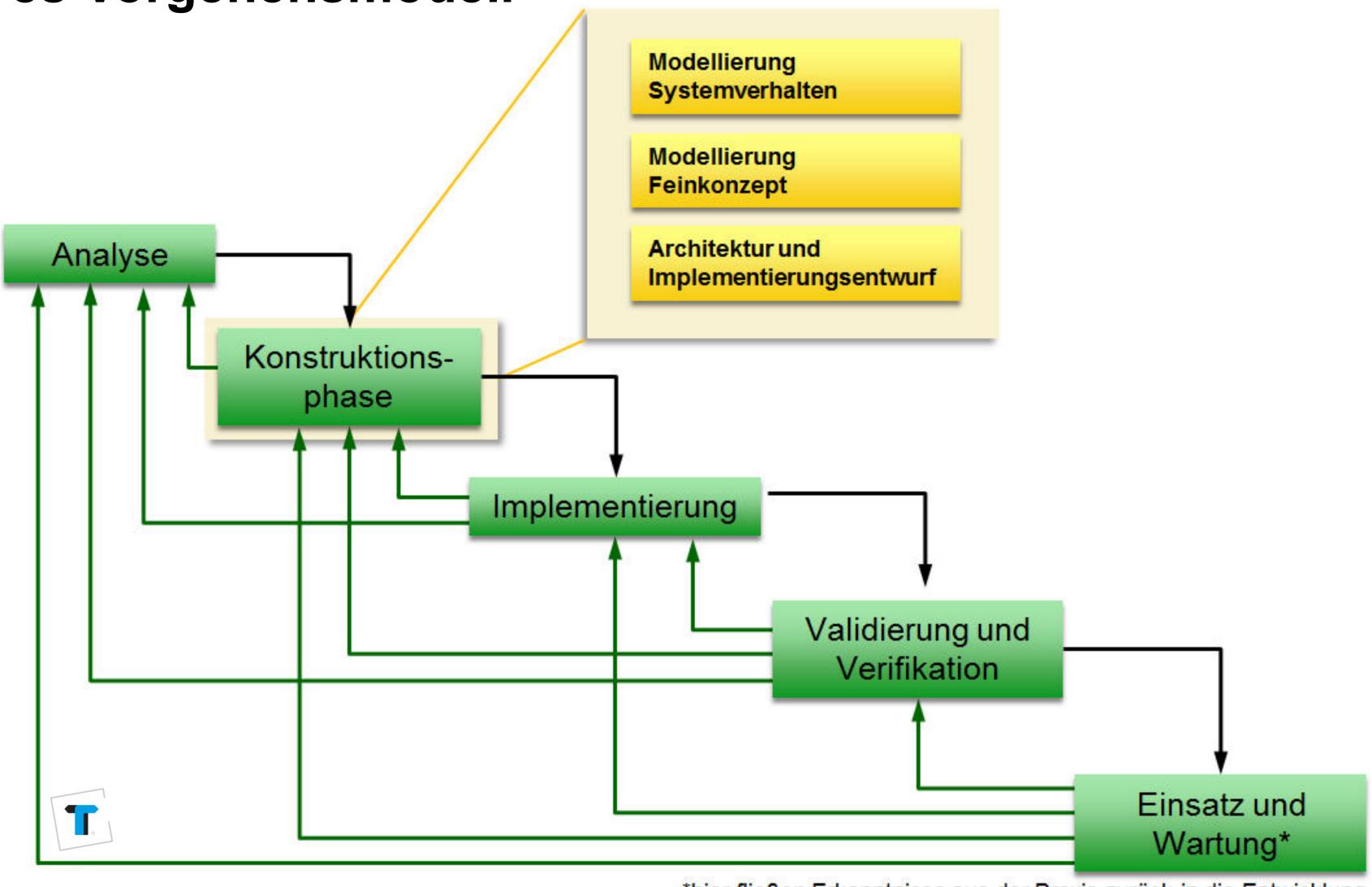
Neu dabei ist:

Man verlässt das Wasserfallprinzip

Im iterativen Prozess nimmt man in jeder Phase Unvollständigkeiten bewusst in Kauf

Die Rückkehr zu jeder Phase wird durch Werkzeuge (case tools) unterstützt, die eine permanente Konsistenzprüfung des Gesamtsystems zulassen Iteratives Vorgehensmodell



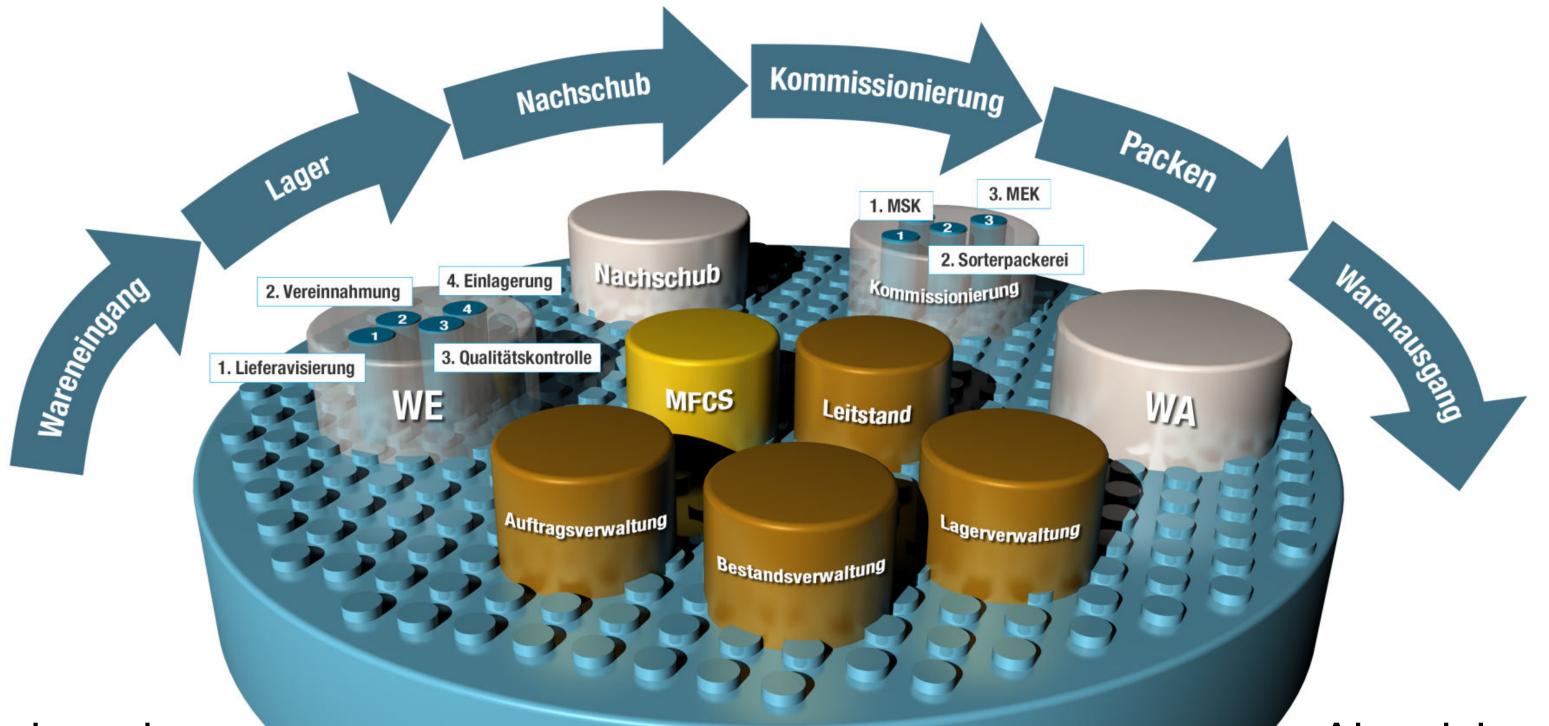


*hier fließen Erkenntnisse aus der Praxis zurück in die Entwicklung

Komponenten-Architektur (Adaptive Prozessbausteine)

Veredelung der Standardprozesse auf neue Anforderungen





Immer wiederkehrende Geschäftsprozesse werden mit Bausteinen der Standardmodule abgebildet.

MIDDLEWARE: TUP STANDARD

Abweichende Geschäftsprozesse werden über Plugins und Veredelungsmodule abgebildet.

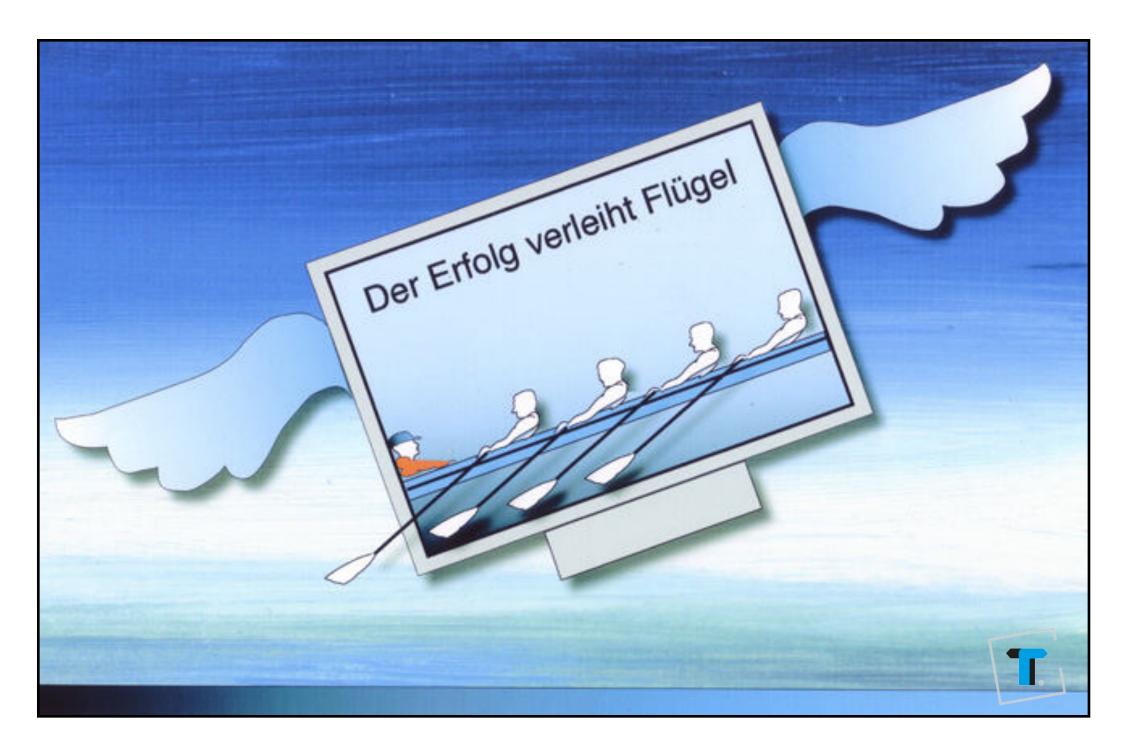
Im Skript Abbildung 5.3: Bestandteile der Softwarearchitektur einer adaptiven IT-Lösung

Erfolg verleiht Flügel!



- das Team macht etwas Spannendes
- etwas wofür Gehirnschmalz gebraucht wird
- wo Ideen zählen
- auf **jeden** kommt es persönlich an
- das Team verstärkt sich durch gute, hochmotivierte Mitarbeiter.

"Wer will keinen Erfolg haben"?



Vorgehensmodell: Prinzip der agilen Methoden



Die agilen Komponenten zielen auf die Software-Entwicklung, und nicht auf die Feinspezifizierung der Anforderungserfüllung, die durch den Softwareanbieter entlang der Prozesskette als Pflichtenheft und als Vertragsbestandteil verankert wird.

Die Basis der agilen Vorgehensweise bildet ein vollständiges, gut dokumentiertes Pflichtenheft und unterschriebenes Pflichtenheft wie bei den klassischen Vorgehensmodellen.

Vorgehensmodell: Prinzip der agilen Methoden



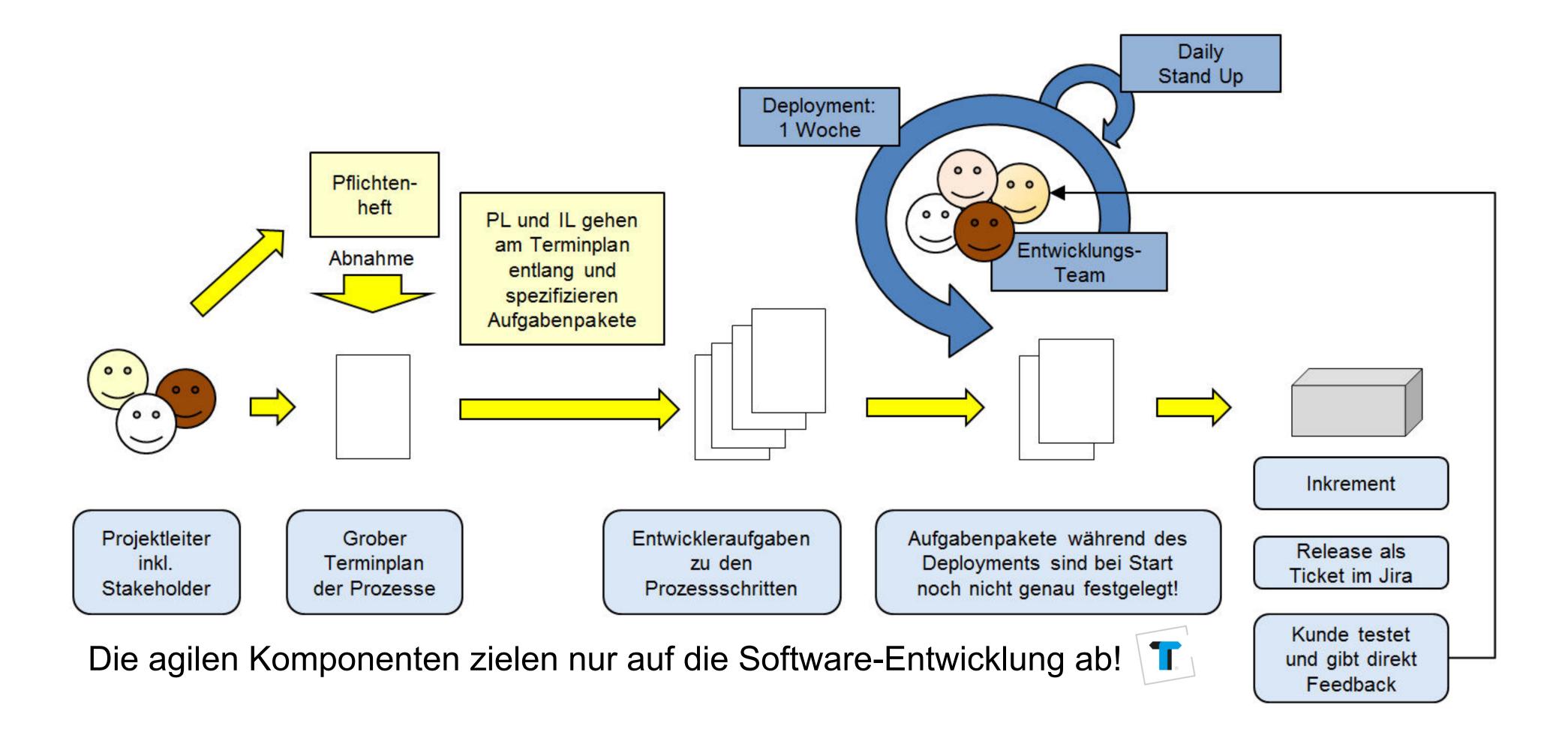
 Hier gilt "Software Follows Function", wenn die Geschäftsprozessmodule detailliert beschrieben sind und damit die Voraussetzung für eine adaptive IT-Lösung bilden.

(siehe auch Kapitel 6.2 Softwaretechnik)

Agile Vorgehen Softwareentwicklung nach industriellen Maßstäben in der IT **Planungs-Analyse-**Lasten-Daily **Entwicklungs-**Stand Up phase und heft, unteraufgaben Deployment: alle Konstruktions-1 Woche Ziele, schriebenes zu den Anforderungen phase **Anforder-Pflichtenheft** Prozesssind dokumentiert ungen schritten Entwicklungs-Kapitel 6.2.3 Kapitel 5.11 Team IL+PL PL, Intergeben IL, IT-Stake-Stakedisziplinäres **Terminplan Entwicklungs**holder holder Team entlang der team + IL + PL **Prozess-**IL Aufgabenpakete fortschritte* während des Deployments **Projektlaufzeit Detaillierter** Inkrement inkl. **Terminplan Iteratives Vorgehen** Ziel **Endtermin** angepasst am **Termin** Release als **Detaillierte Projektfortschritt Einzuplanendes** Ticket im Jira Projektkosten **Budget** Kunde testet und gibt direkt software follows function! Planungsphase Logistik Feedback

Agile Vorgehensweise bei TUP





Objektorientierung



In der Softwareentwicklung werden objektorientierte Methoden umgesetzt:

- □ zur Verbesserung von Produktivität, Wartbarkeit und der Software-Qualität
- ☐ damit adaptive IT-Prozessbausteine möglichst oft wiederverwendet werden können. (Objektorientierte Framework)

Die entwickelten Objekte sollen die "reale Welt" abbilden.

Objektorientiertes Strukturmuster - Wiederverwendbare Transportverwaltung



Das Standardmodul "Transportverwaltung" ist die zentrale Instanz zur Überwachung, Beauftragung und Koordination aller Transportaufträge und Transportressourcen. (siehe im Skript Kapitel 2.1 und ff.)

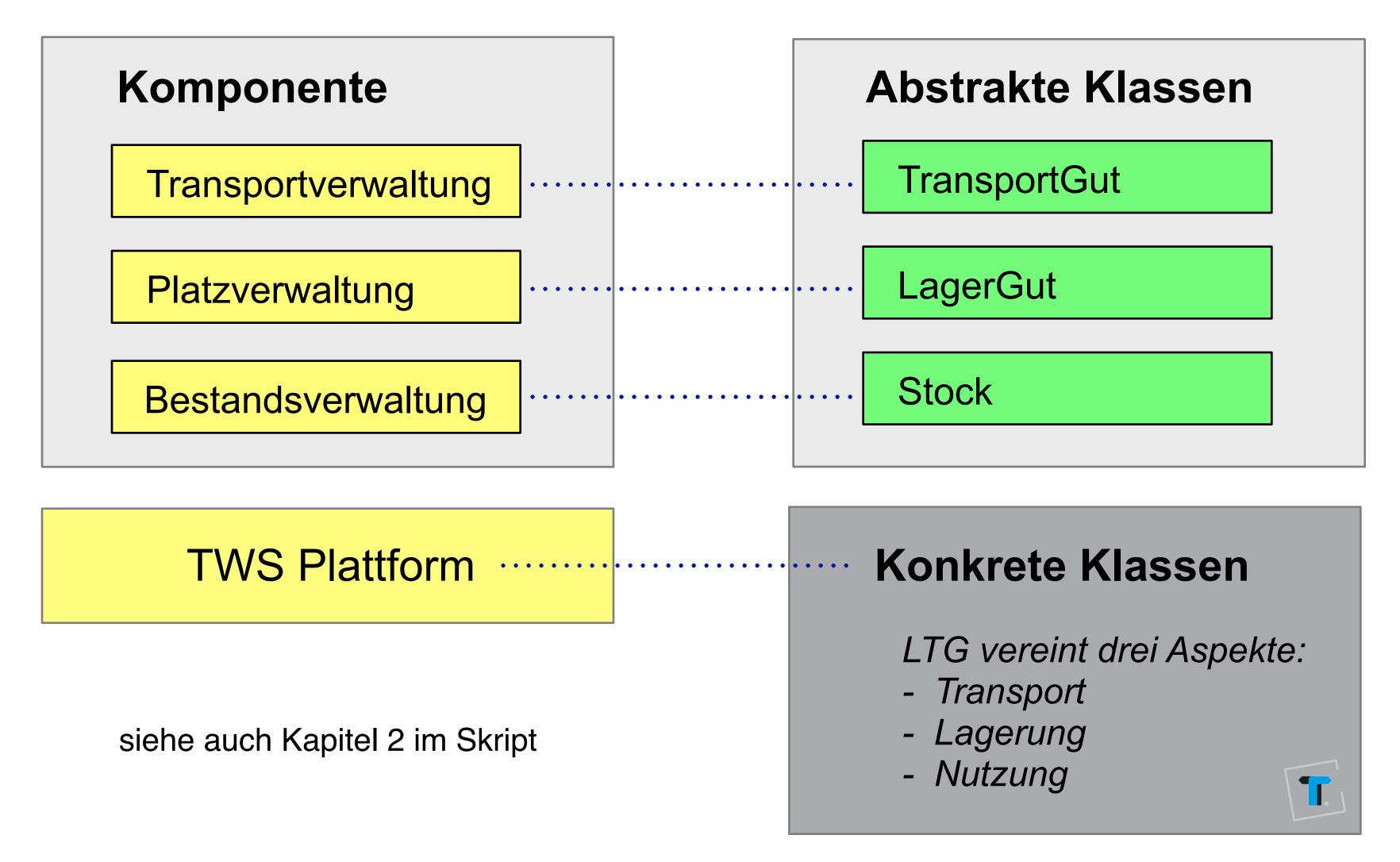
Objektorientiertes Strukturmuster - Wiederverwendbare Transportverwaltung



Das Standardmodul "Transportverwaltung" ist die zentrale Instanz zur Überwachung, Beauftragung und Koordination aller Transportaufträge und Transportressourcen.

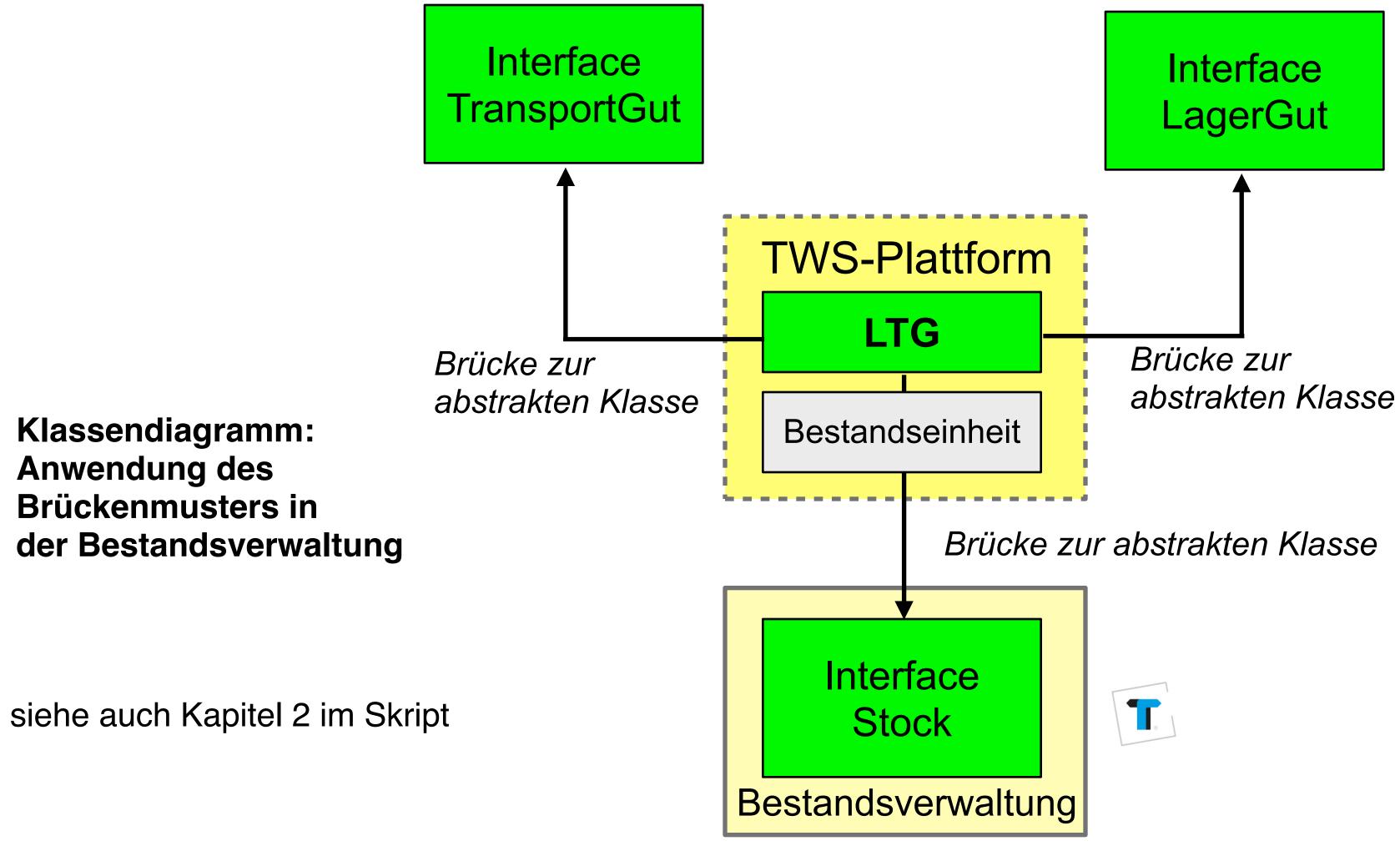
Klasse LTG (LagerTransportGut)

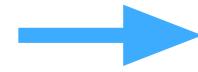




Verbindung von Transportverwaltung und Platzverwaltung über die Klasse LTG



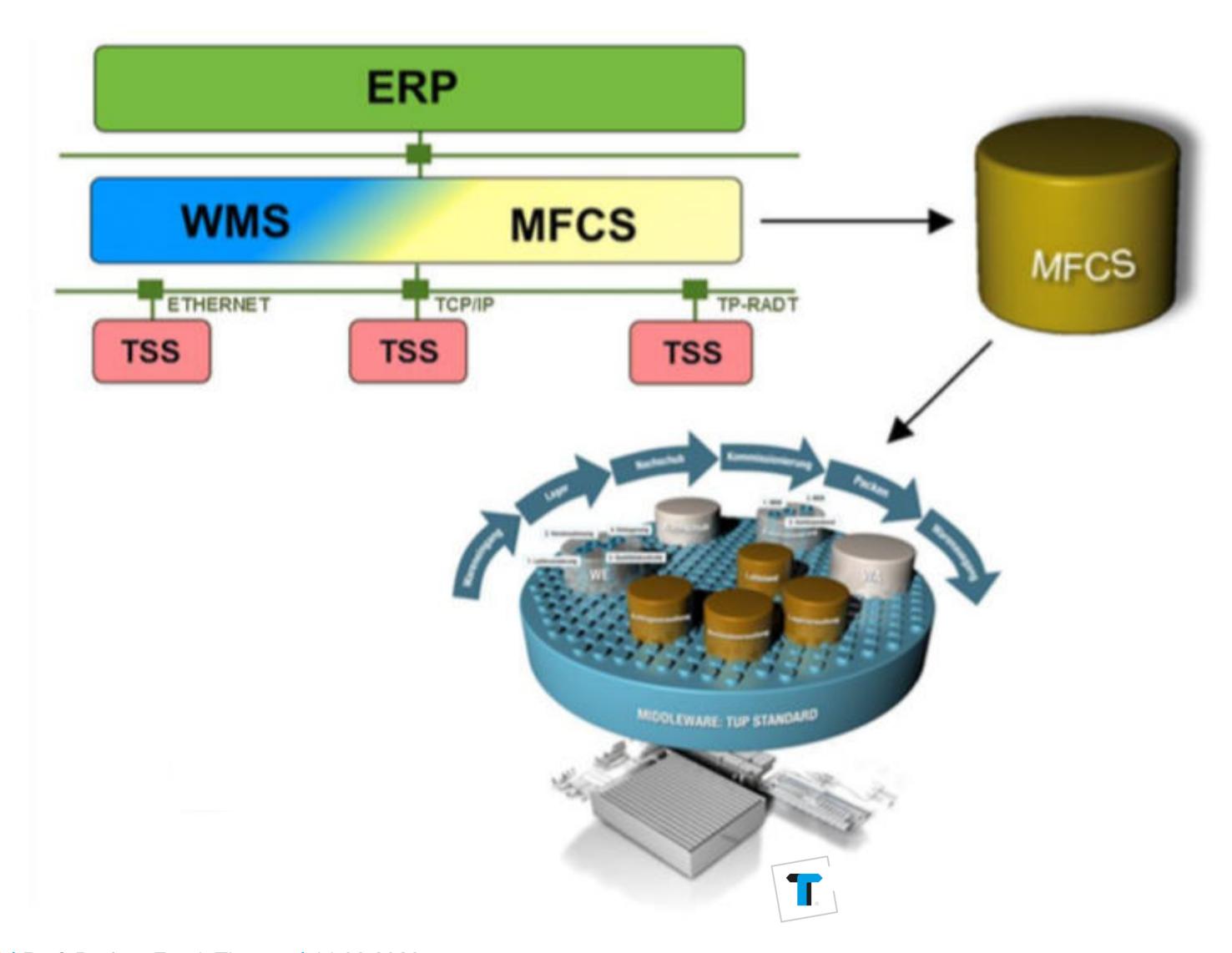




Softwareentwicklung nach industriellen Maßstäben, erhöht die Planungsintelligenz bei Intralogistik-Systemen

Einordnung des MFCS in die Systemlandschaft (Innovativer Ansatz)





Die Abstrakte Fabrik - angewendet auf den Lagerplatz im Kommissionierlager



In der Lagerplatzverwaltung (siehe auch Kapitel 5.4) ist es möglich, Lagergüter mit unterschiedlicher Einlagerstrategie einzulagern.

Kunde A: möglichst nahe an den Gassenköpfen

Kunde B: es sollen zuerst Lagerplätze in optimaler

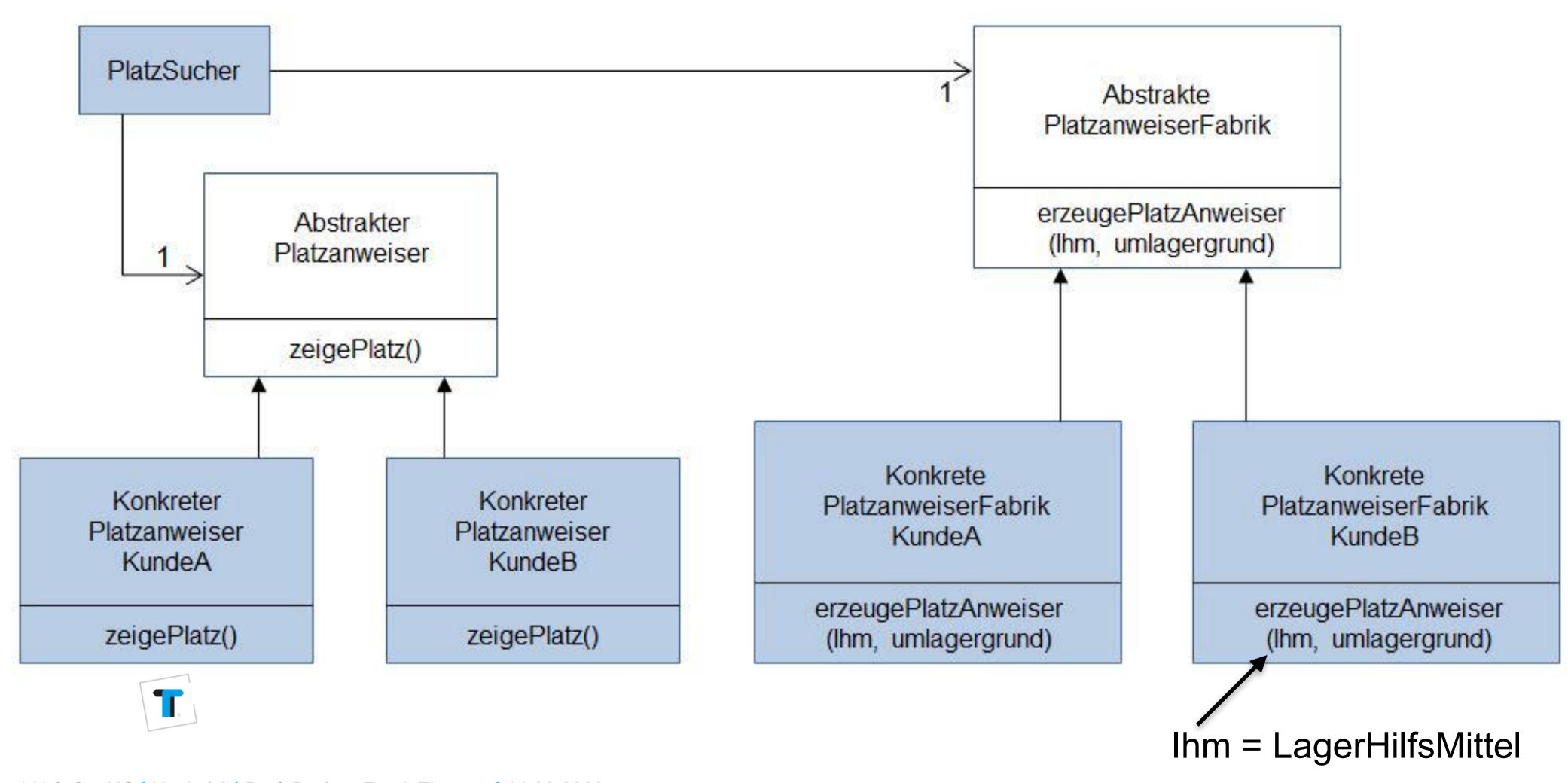
Griffposition belegt werden

Zusätzlich soll eine geeignete Einlagerstrategie abhängig vom LHM des Lagerguts möglich sein.

Klassendiagramm

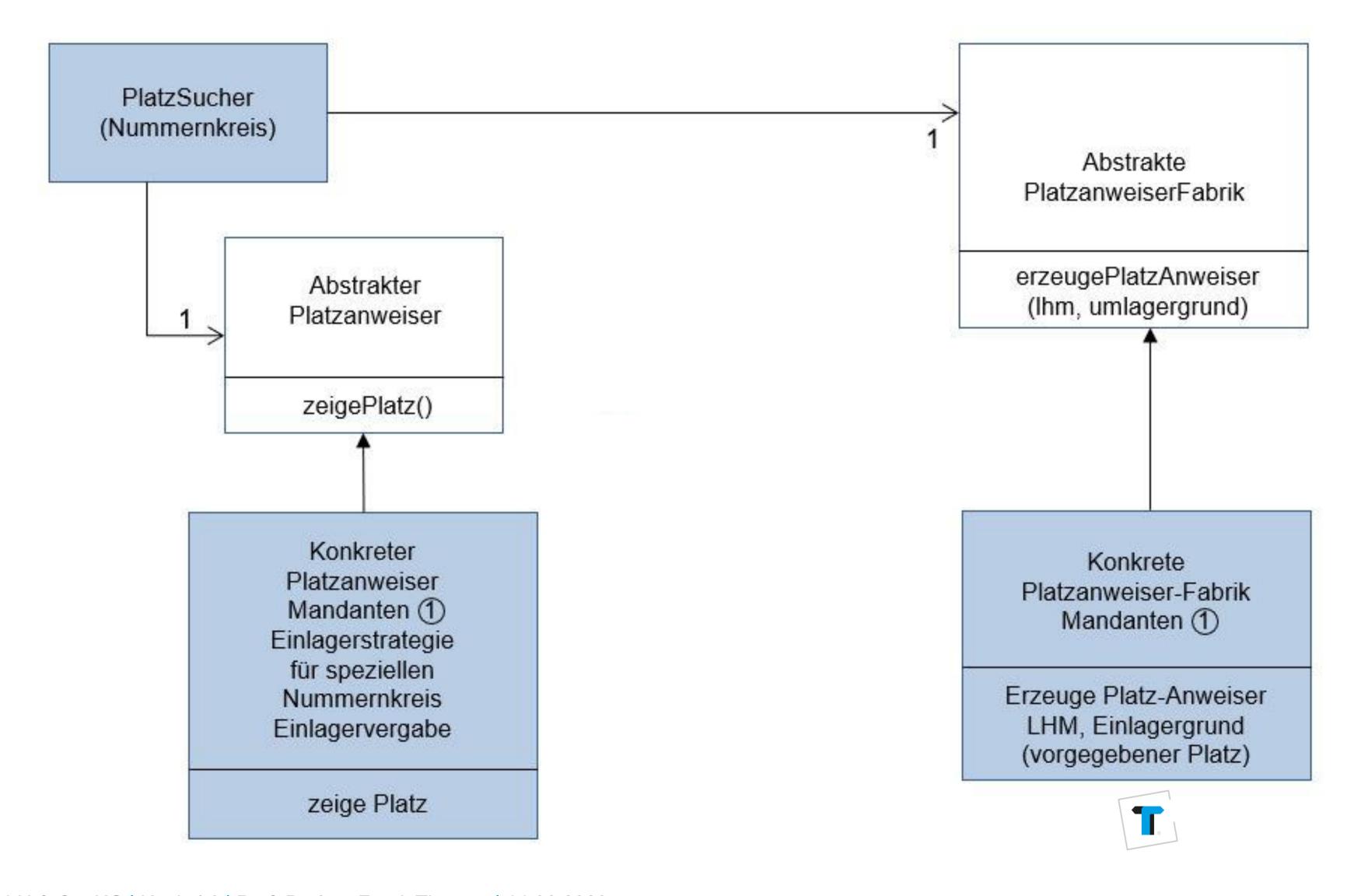
Anwendung des abstrakten Fabrikmusters in der Platzverwaltung Anwendung nach Gamma et al. (2004)





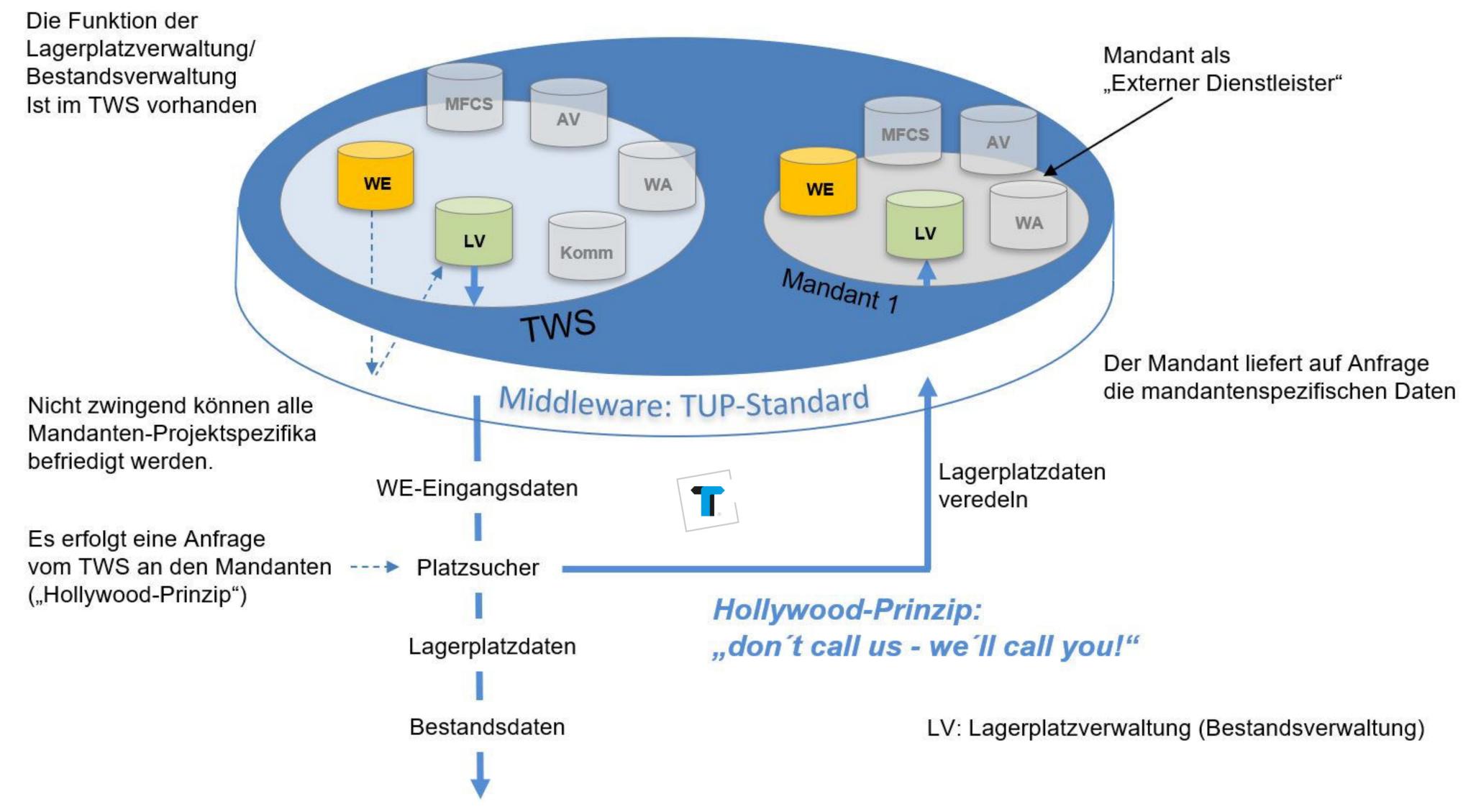
Framework: Abstrakte Fabrik - angewendet auf die Lagerplatzverwaltung (Bestandsverwaltung)





Hollywood-Prinzip: Framework Abstrakte Fabrik





Abstrakte Platzanweiser-Fabrik



Entkopplung zwischen einem "Platzsucher" von einem Dienstleister "Platzanweiser"

Aufgabenverteilung zwischen "Suchen" und "Platzstrategie"

Es gilt: In der "Platzanweiser-Fabrik" wird entschieden, welcher Typ von "Platzanweiser" verwendet wird.

Frameworks sind in Software gegossene Konzepte und Lösungen

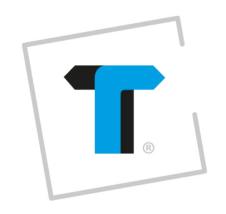


Frameworks sind in Software gegossene Konzepte und Lösungen auf der Basis von detailliert erarbeiteten Geschäftsprozessen (Kapitel 5)

Diese Konzepte helfen dem Entwickler beim Modellieren der Anwendungsdomäne, und beim Erkennen der relevanten Beziehungen unter den Objekten.

Für das Entwicklungsteam ergeben sich daraus Vorteile: Sie finden einen großen Teil des Grundwissens als integralen Bestandteil des Frameworks

Framework (Rahmen, Gerüst, Skelett)



Durch ein objektorientiertes Framework wird eine Software-Architektur für eine Anwendung vorgegeben.

- □ Damit werden die Struktur wesentlicher Klassen und Objekte, sowie ein Modell Kontrollfluss, in der Anwendungsdomäne festgelegt (Hollywood-Prinzip)
- ☐ In diesem Sinne werden Frameworks mit dem Ziel einer Wiederverwendung von Architekturen entwickelt und genutzt

Vorteile für das Entwicklungsteam

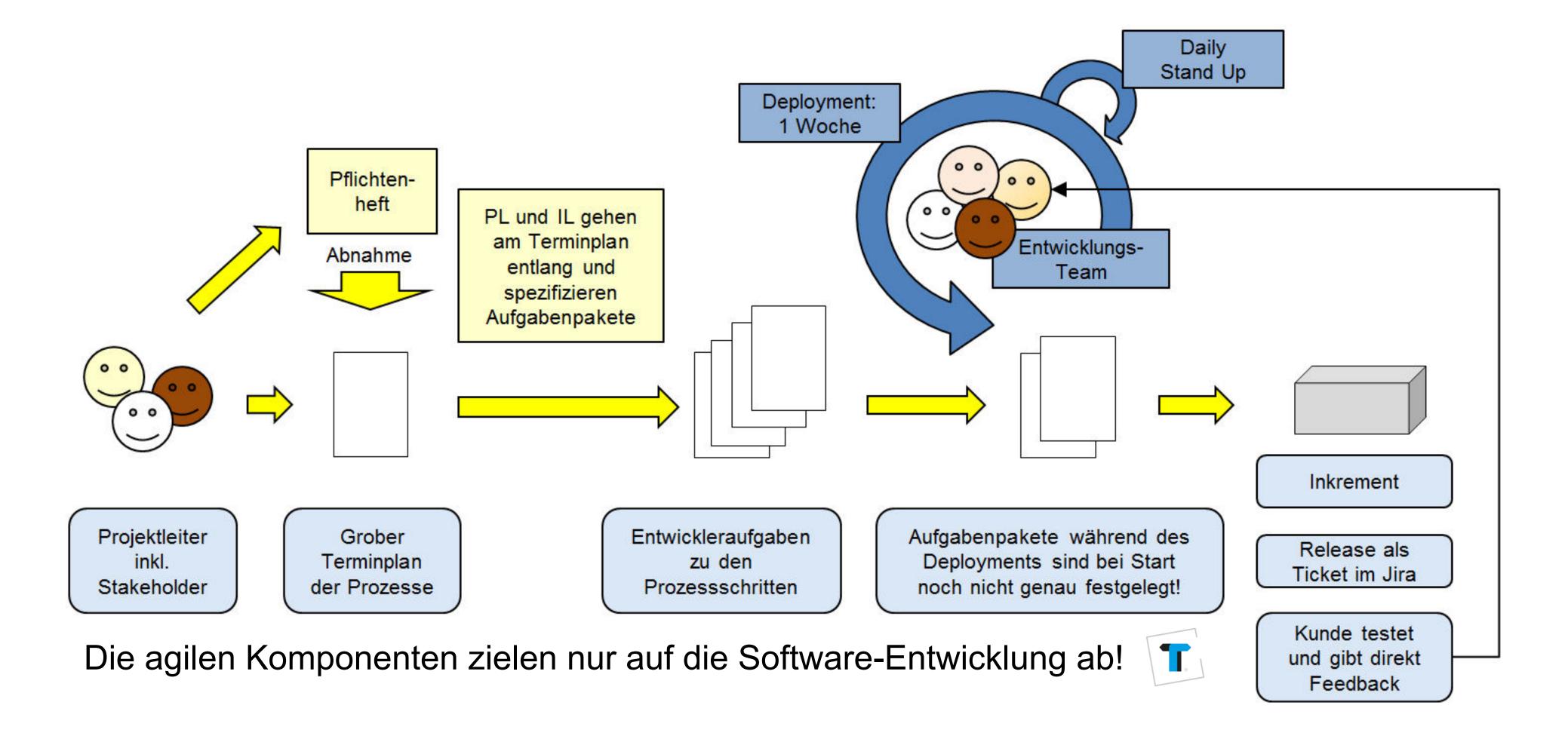


Nach dem Vorgehensmodell "Prinzip der agilen Methoden" können die spezifischen Arbeitspakete entsprechend den Anforderungen an die Entwickler aufgeteilt werden:

- ☐ Für einen mit der Anwendungsdomäne vertrauten Entwickler, führt dies zu einem beschleunigten Projektfortschritt
- □ Ein Unerfahrener Entwickler ist in der Lage Anwendungen zu schreiben, ohne die Anwendungsdomäne vollständig verstanden zu haben.
 Das Grundwissen ist integraler Bestandteil des Framework

Agile Vorgehensweise bei TUP

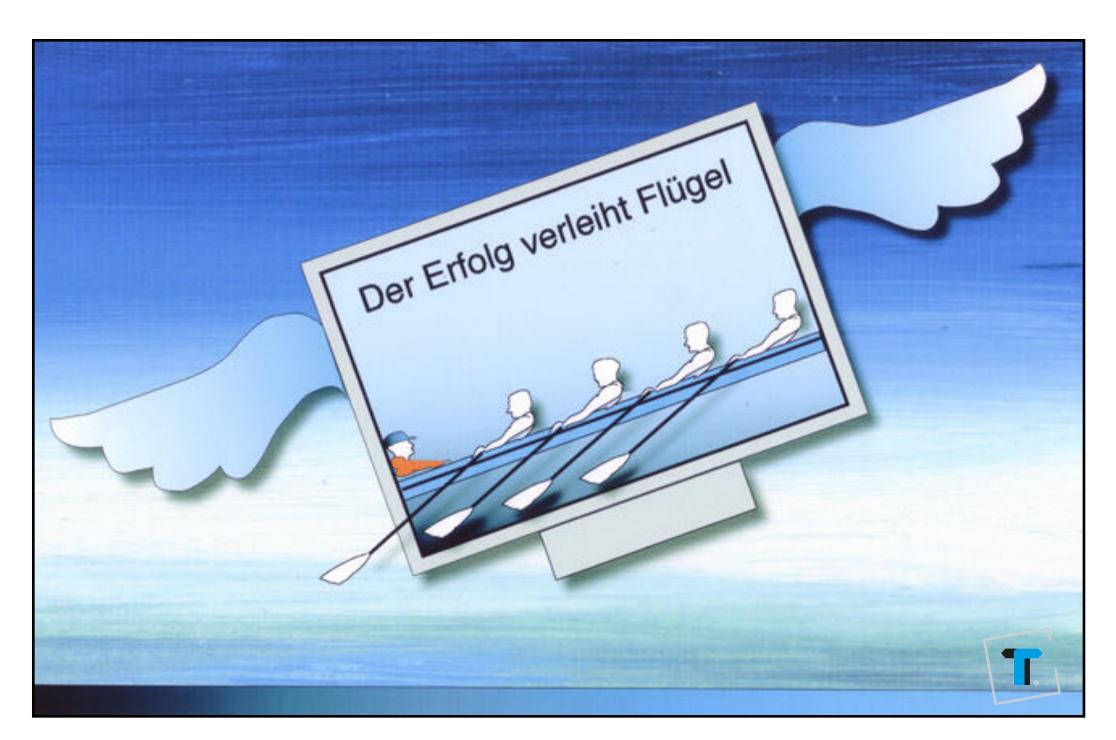




Erfolg verleiht Flügel!



- das Team macht etwas Spannendes
- etwas wofür Gehirnschmalz gebraucht wird
- wo Ideen zählen
- auf jeden kommt es persönlich an
- das Team verstärkt sich durch gute, hochmotivierte Mitarbeiter.
 - "Wer will keinen Erfolg haben"?



Vorteile einer abstrahierten wiederverwendbaren Anwendungsdomäne



- ☐ Frameworks können als Absicherung von Investitionen verstanden werden
- ☐ Erarbeitetes Wissen über das Anwendungsgebiet (hier WMS/MFCS) bleibt in der Firma erhalten
- Ein gutes Framework ermöglicht Anwendungen zu entwickeln, ohne alle Details der dazu benötigten Ressourcen verstehen zu müssen.

Frameworks = Components + Pattern



Die Struktur gestaltet sich in Form einer Beziehung der Anwendung als spezifischer Instanziierung des Frameworks (z.B. "best practice Komponente" im WE)

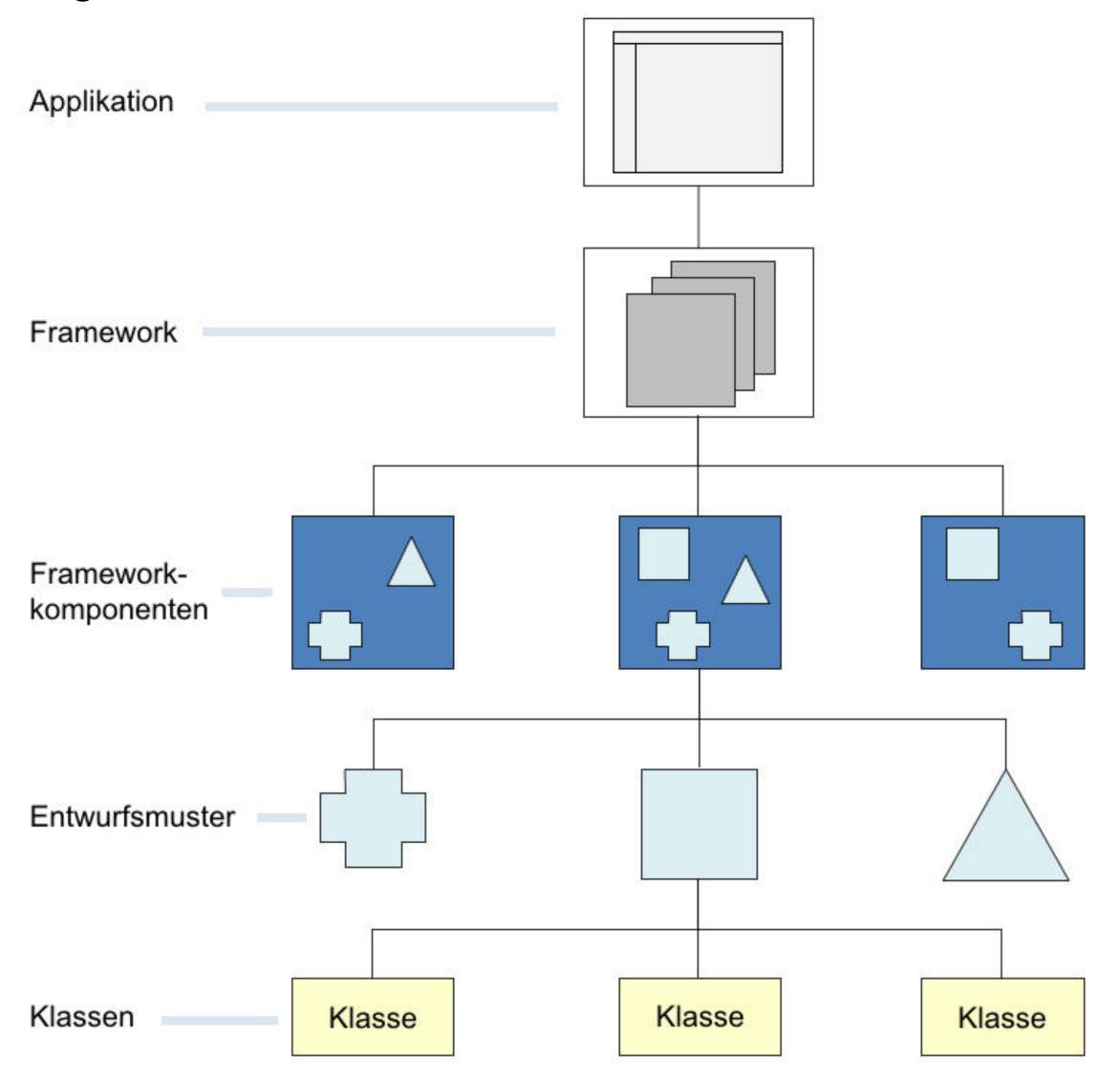
- dem Framework
- den Entwurfsmuster
- den Komponenten Klassen

Klassen und Entwurfsmuster vereinigen sich zu Komponenten. Zusammen bilden alle Komponenten das eigentliche Framework.

Komponenten des Framework

Struktureller Aufbau / Beziehungen





Bildquelle: John (1997)

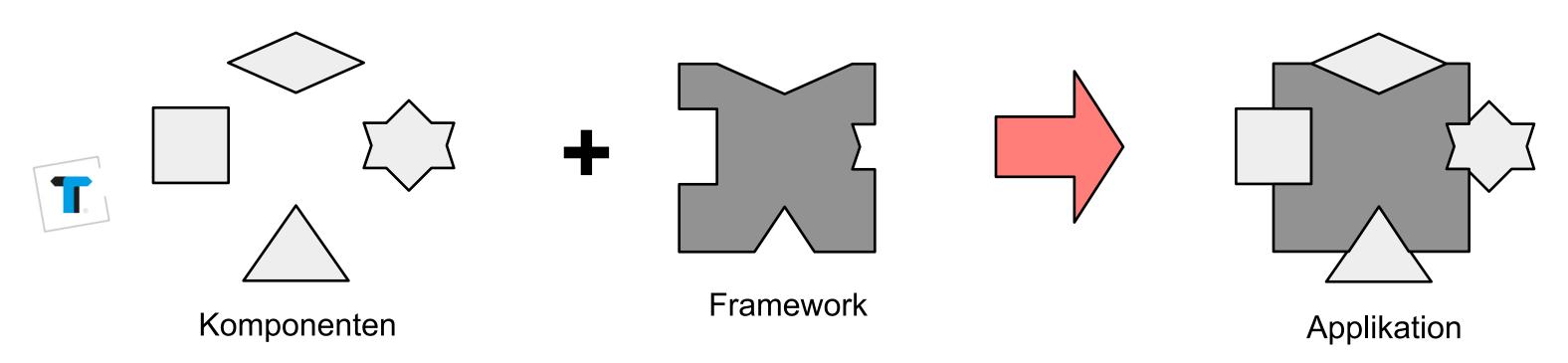
Frameworks

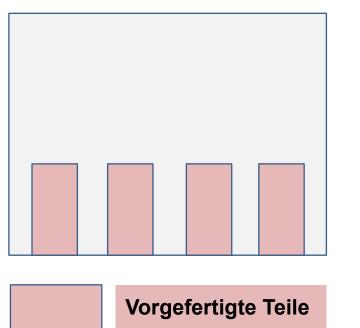


Um komplexe objektorientierte adaptive Architekturen zu beherrschen, gibt es den zentralen Ansatz des objektorientierten Frameworks (Rahmenwerk).

Ein Framework im allgemeinen Sinne ist ein wiederverwendbares System, welches in fertige und halbfertige Subsysteme untergliedert ist. Es legt dabei die Struktur dieser Systeme und Subsysteme fest.

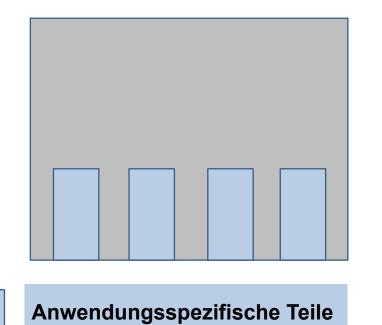
- Wiederverwendung von Design
- Zur Entwicklung von ähnlichen Anwendungen

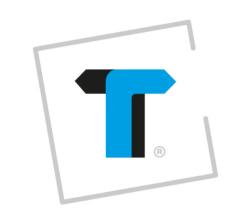




Klassenbibliothek

Frameworks





Ablaufsteuerung nicht vordefiniert

Kontrollfluss durch Anwendung

Unabhängige wiederverwendbare Module

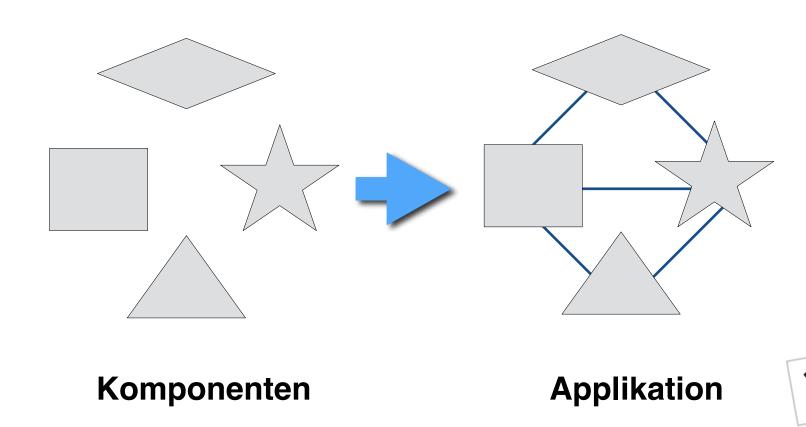
Mehrfachverwendung von Funktionalität

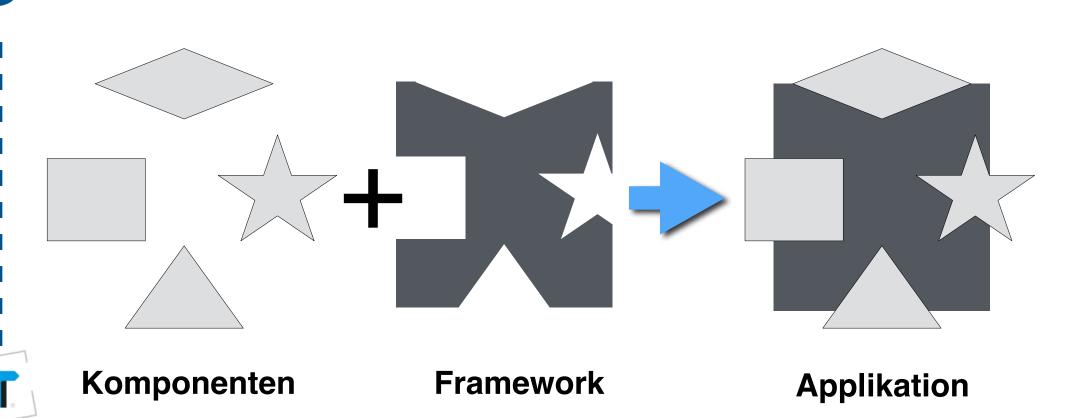
Ablaufsteuerung im wesentlichen vordefiniert
Kontrollfluss durch Framework ("Hollywood Prinzip")

Verbund zusammenhängender Klassen
Mehrfachverwendung von Struktur und Funktionalität

Applikationsentwicklung:

Applikationsentwicklung:





Softwarearchitektur



Eine adaptive IT erfordert eine modulare Softwarearchitektur, die in ihrer Eigenschaft

- skalierbar
- anpassbar
- erweiterbar
- und wiederverwendbar

sein muss.

Ziel ist eine nach den Architektur-Prinzipien ausgerichtete Systemarchitektur.

Architektur Prinzipien



Die Bausteine (Geschäftsprozessmodule) sollten

- □ nach Aufgabenbereichen getrennt sein (Separation of Concerns)
- ☐ die Ausprägung von Black-Boxes haben (Information-Hiding)
- und einen hohen Abstraktionsgrad aufweisen (Abstraktionsprinzip)

Separation-of-Concerns-Prinzip



Wichtigster Einsatz ist die Unterstützung der Modularisierung:

Dabei sind Teile einer anwendungsbestimmten Software - z.B.: Im Geschäftsprozessmodul "Zweistufige Kommissionierung". Hier die Aufgabe Mensch-Maschine-Kommunikation während des Rundgangs zu identifizieren und als eine wiederverwendbare Komponente zu kapseln.

Ein komplexes Geschäftsprozessmodul wird auch aus der Sicht "des Prinzip der agilen Methoden" in verständliche handhabbare Arbeitspakete (Komponenten) zerlegt.

Trennung von anwendungsbestimmter und technikbestimmter Software



A: Anwendungsbestimmte Software kann immer dann wiederverwendet werden, wenn vorhandene Anwendungslogik ganz oder teilweise benötigt wird.

T: Technikbestimmte Software kann immer dann wiederverwendet werden, wenn ein neues - z.B. ein Fördersystem dieselben technischen Komponenten einsetzt.

Information-Hiding-Prinzip

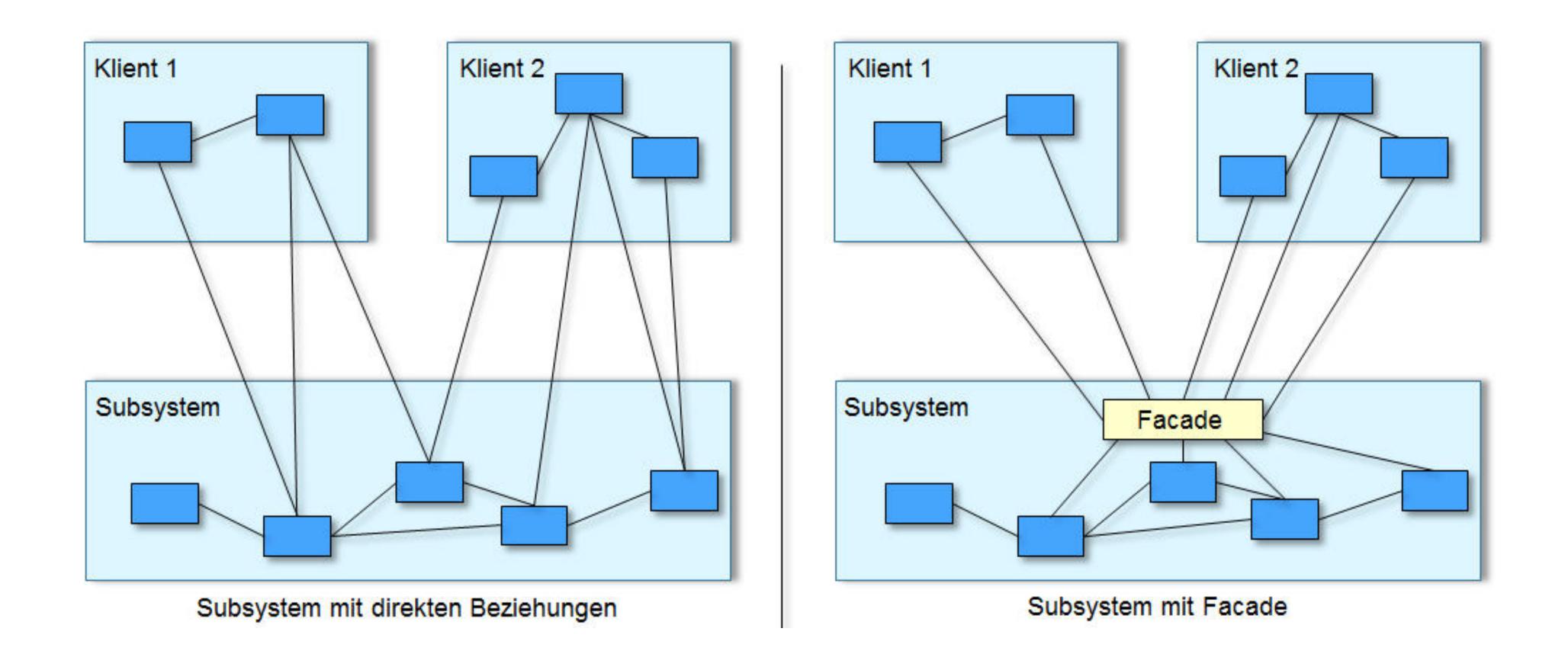


Ein Baustein wird gekapselt und ist nach außen durch wohl definierte Schnittstellen bekannt.

- □ Black-Box-Prinzip, die Interna sind nicht sichtbar, ausschließlich die Schnittstellen damit können Interna quasi beliebig oft geändert werden (MP-Rundgang, EP-Rundgang)
- Unterstützt auch größere Strukturen Beispiel: Facade-Entwurfsmuster schützt ein ganzes Subsystem

Subsystem ohne und mit Facade





Abstraktionsprinzip



Das Prinzip hierbei wichtige Aspekte zu identifizieren und unwichtige Details zu vernachlässigen (Schnittstellenabstraktion)

Trennung von Schnittstelle und Implementierung:

- □ Damit sich ein Klient auf die Schnittstelle verlassen kann, soll die Schnittstelle separat von der Implementierung betrieben werden
- ☐ Anwendung findet das, wenn die Schnittstelle standardisiert ist

Modularitätsprinzip



Eine Softwarearchitektur sollte aus Bausteinen bestehen, die funktionale Verantwortung klar abgegrenzt sind, d.h. leicht austauschbar und in sich geschlossen.

Das Modularitätsprinzip dient der

- Änderbarkeit
- Erweiterbarkeit
- Wiederverwendbarkeit von Bausteinen einer Architektur

Aufteilung in Anwendungsdomäne und Middleware



In einem ersten Schritt wird die Software getrennt nach dem Aspekt:

- Technikbestimmter Software und
- Anwendungsbestimmter Software

Lauffähiges Kundensystem

Anwendungsdomäne Intralogistik

Middleware

Die Anwendungsdomäne bewältigt alle Problemfälle der Intralogistik.

Middleware



- Anwendungsunabhängige Technologie
- Vermittelt zwischen fachlicher Anwendungssoftware und Betriebssystem / Hardware
- Hat den größten Wiederverwendungsgrad
- □ Die Datenbank ist über die Middleware verbunden. Das Modul Persistence hat die Aufgabe in weitgehender transparenter Weise die Verbindung zwischen Anwendung und Datenbank herzustellen

Middleware

Die Middleware (Zwischenanwendung) bezeichnet eine anwendungsunabhängige Technologie, die in fachlicher Anwendungs- software und Betriebssystem / Datenbank / Hardware vermittelt.

Sie bietet notwendige und hilfreiche Dienste für die Implementierung und Laufzeit von Anwendungssoftware.

Sie ist aber nicht dem Bereich der Anwendungssoftware zuzuordnen.

Das Modul Persistence hat die Aufgabe, die Verbindung zwischen Anwendung und relationaler Datenbank herzustellen.





Anwendungsdomäne



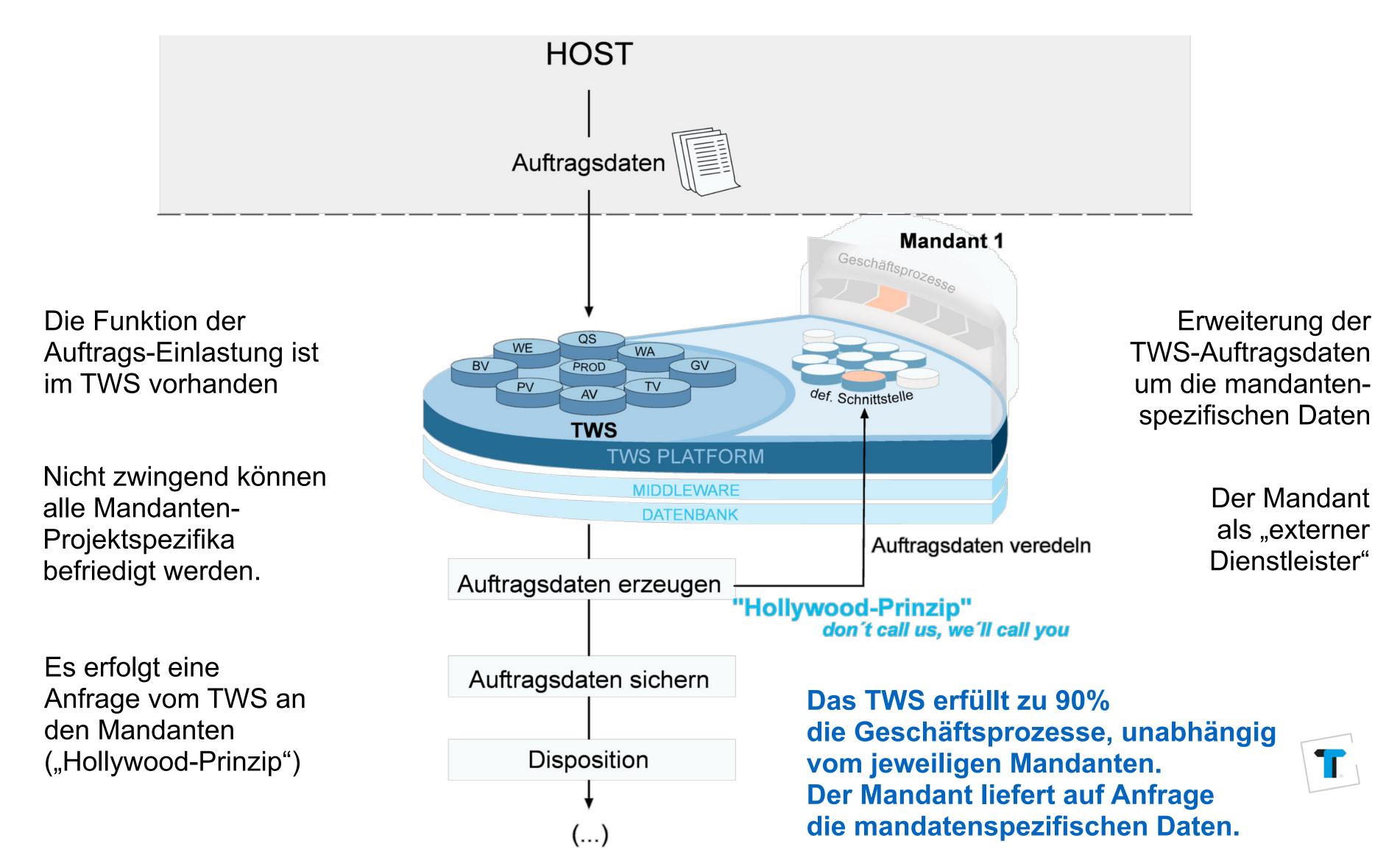
Nach dem Geschichtspunkt der Wiederverwendung strukturiert sich die Anwendungsdomäne in

- ☐ Komponenten (z.B. Best-Practice-Komponente)
- ☐ TWS-Plattform ist das Bindeglied der abstrakten unabhängigen Komponenten und konkretisiert die Teilaspekte der Komponenten
- ☐ Plugins müssen nicht mehr den Ansprüchen an Flexibilität genügen

Die Projektspezifische Software definiert zusammen mit den Komponenten der TWS-Plattform und die Plugins das lauffähige System

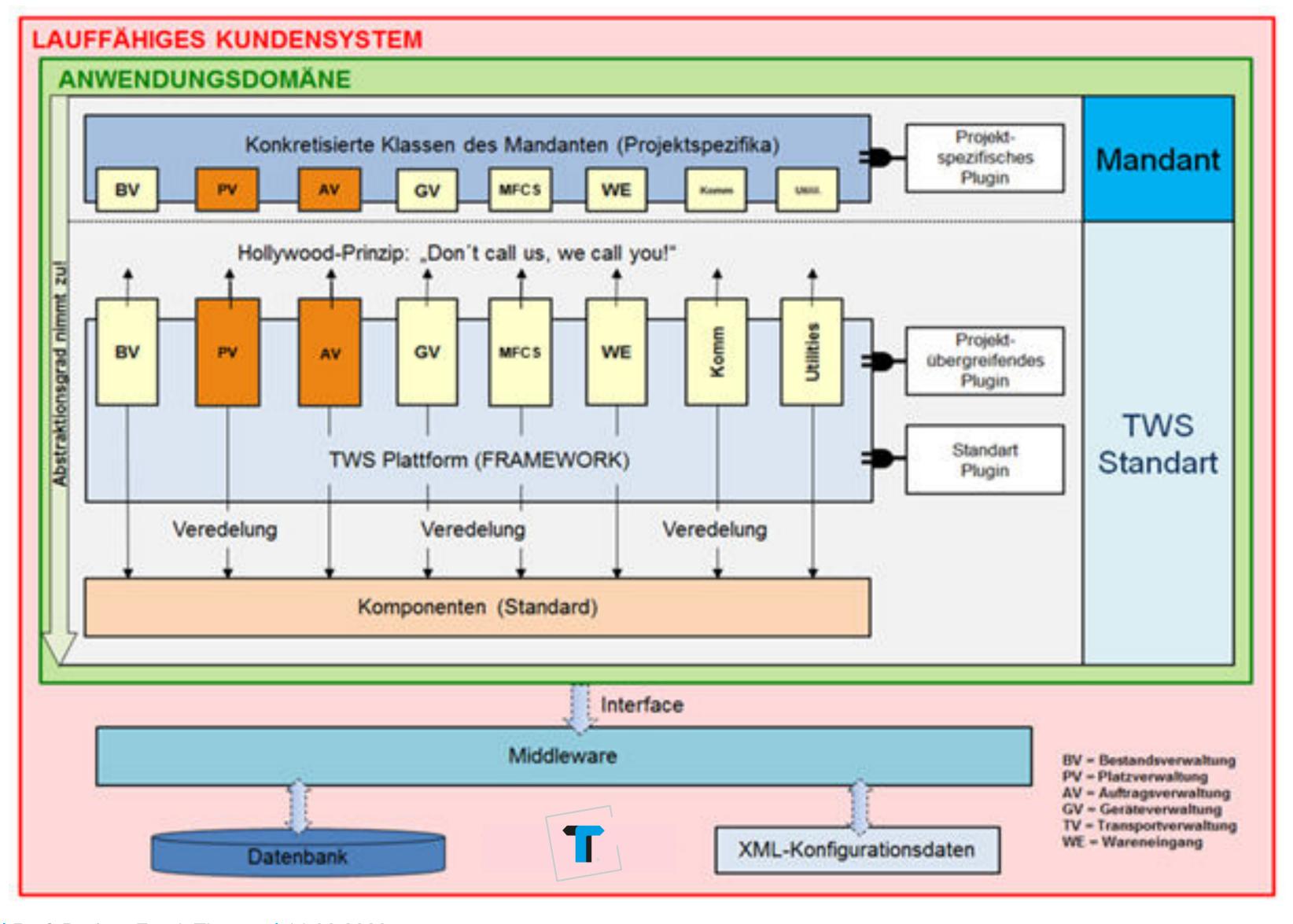
Das TWS und das Hollywood-Prinzip





TWS - Lauffähiges Kundensystem





Framework (Rahmen, Gerüst, Skelett)

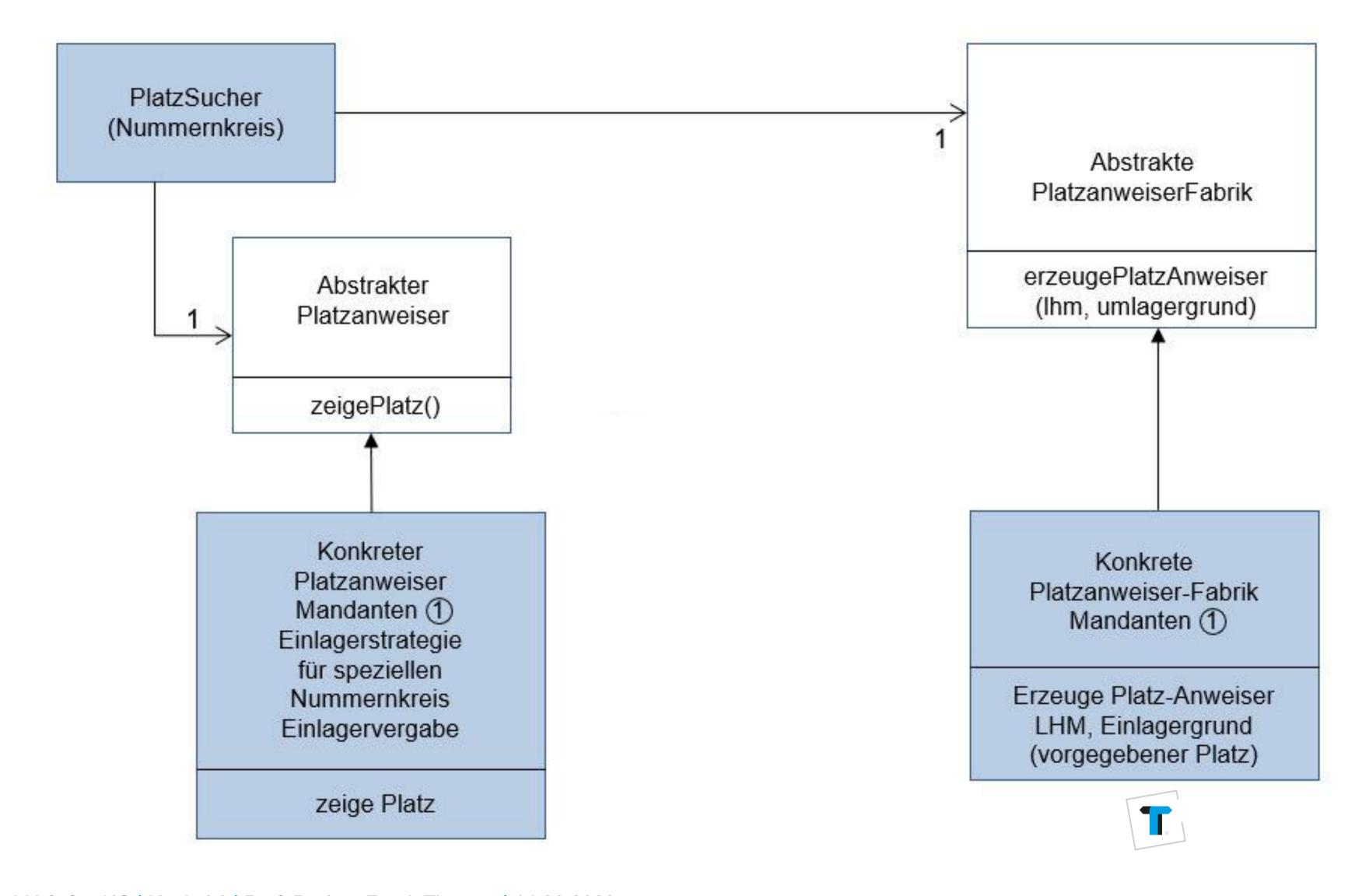


Durch ein objektorientiertes Framework wird eine Software-Architektur für eine Anwendung vorgegeben.

- □ Damit werden die Struktur wesentlicher Klassen und Objekte, sowie ein Modell Kontrollfluss, in der Anwendungsdomäne festgelegt (Hollywood-Prinzip)
- In diesem Sinne werden Frameworks mit dem Ziel einer Wiederverwendung von Architekturen entwickelt und genutzt

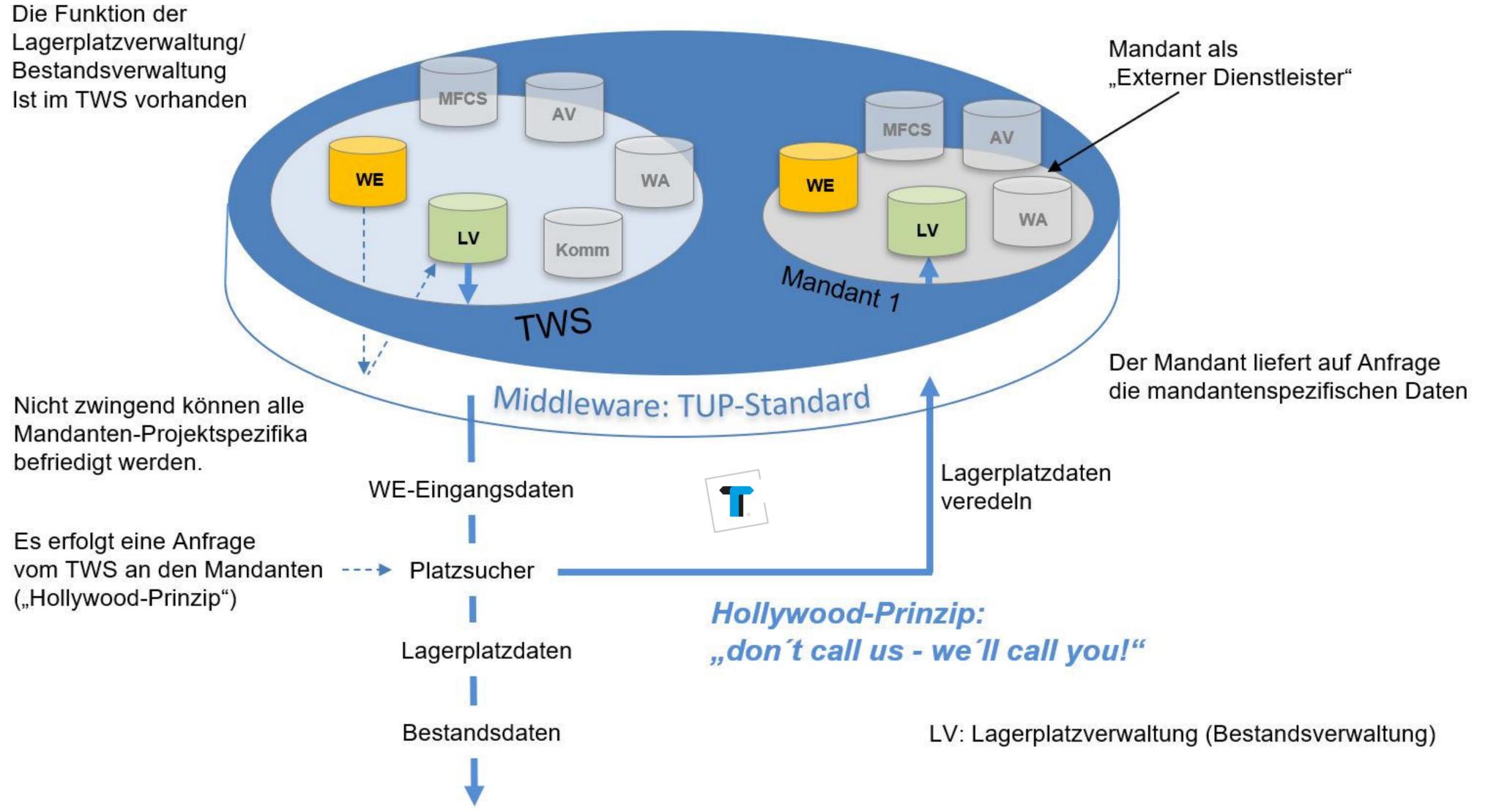
Framework: Abstrakte Fabrik - angewendet auf die Lagerplatzverwaltung (Bestandsverwaltung)





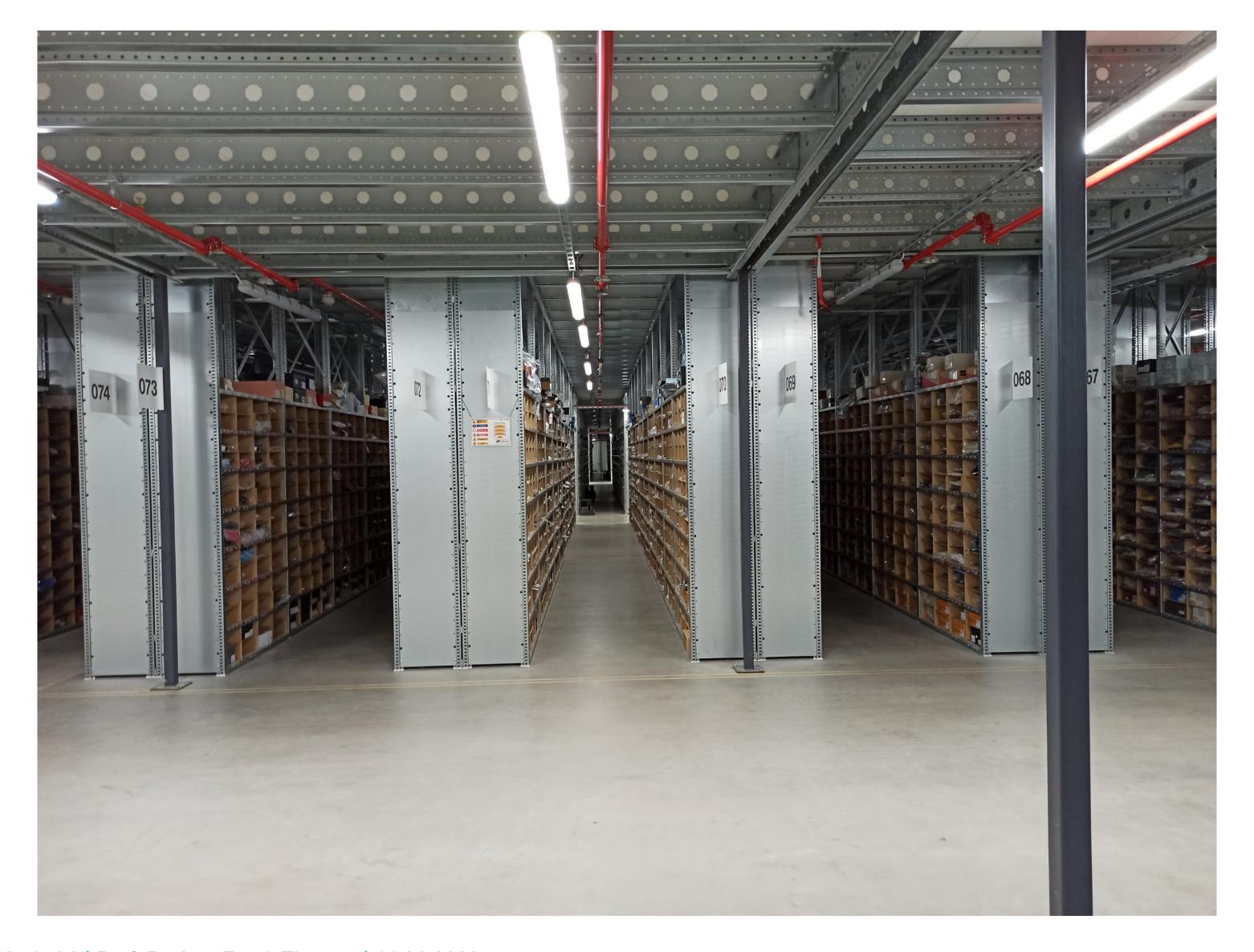
Hollywood-Prinzip: Framework Abstrakte Fabrik





Kommissionierlager Liegeware Textilien





Kommissionierlager Liegeware Textilien



