Projekte. Beratung. Spezialisten.

IKS
Individuelle Softwarelösungen

Domain Driven Design

Stabile Basis für langlebige Software

IKS-Thementag „Softwareengineering heute“

Autor: Christoph Schmidt-Casdorff

06.11.2018

IKS Gesellschaft für Informations-und Kommunikationssysteme mbH

T. +49 2103-5872-0 | www.iks-gmbh.com

Agenda

- Warum Domain Driven Design ?
- Strategisches Design - Konzepte im Problemraum
- Strategisches Design - Konzepte im Lösungsraum
- Taktisches Design und Architektur
- Essentials
- Abschluss

Agenda

- **Warum Domain Driven Design ?**
- Strategisches Design - Konzepte im Problemraum
- Strategisches Design - Konzepte im Lösungsraum
- Taktisches Design und Architektur
- Essentials
- Abschluss

Warum Domain Driven Design?

Software soll Geschäftsanforderungen lösen

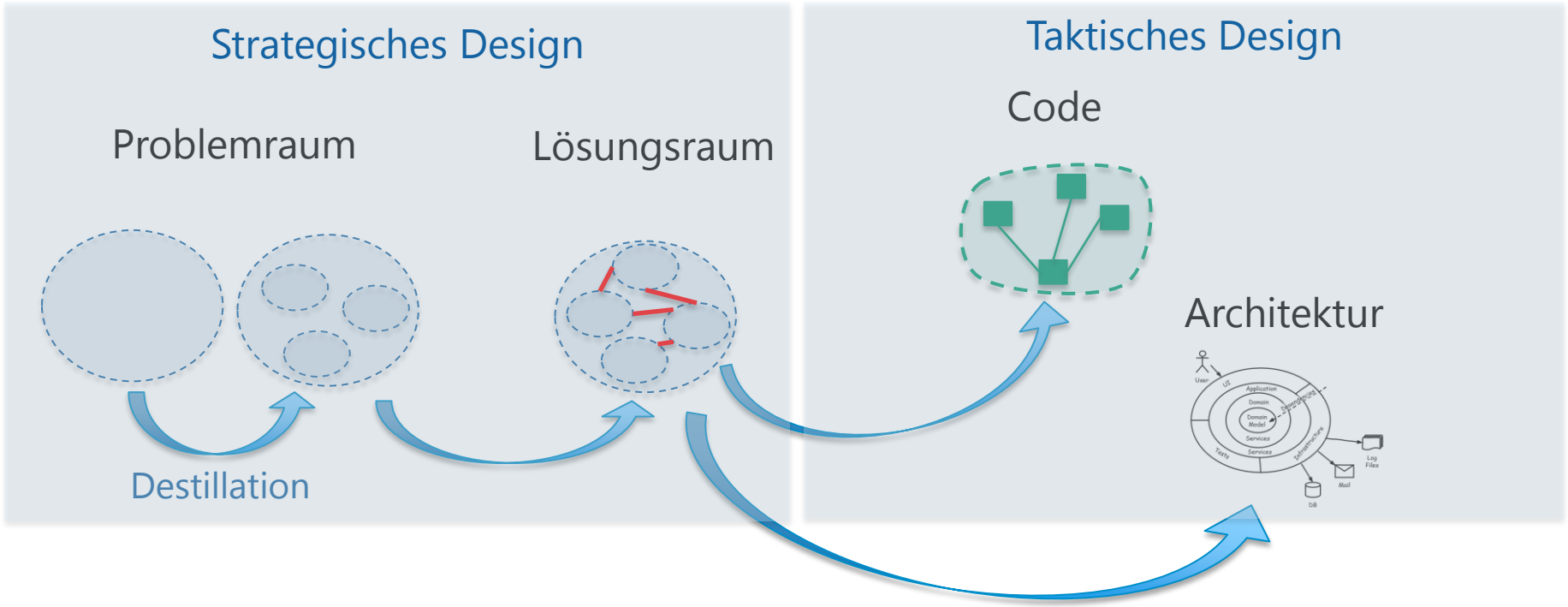
=> Fokus auf die Fachlichkeit

Fokus auf die Fachlichkeit

- Software dient zur Umsetzung von Geschäftsanforderungen
  - Nicht umgekehrt
- Fachlichkeit ist langlebiger als Technologien
  - Fachlichkeit bringt Stabilität in Software
- Fachlichkeit ist der gemeinsame Nenner
  - . . . aller am Entwicklungsprozess Beteiligten
- Domänenwissen begleitet die Softwareentwicklung
  - Softwareentwicklung startet und endet mit dem Wissen der Domänen-Experten

Domäne ist das Problemfeld, für das eine Lösung gefunden werden soll

Fachlichkeit im Zentrum der Softwareentwicklung



Fachliche Kriterien bestimmen das Vorgehen

Allgemeingültige Sprache

Domain Driven Design

- *Domain Driven Design* beschreibt einen Prozess
- *Domain Driven Design* gibt Verfahren/Techniken unterstützend an die Hand
- *Domain Driven Design* ist eine Denkungsart/Philosophie
  - Beschreibt Grundsätze zur Ausrichtung einer Softwareentwicklung an der Fachlichkeit
- Fachlichkeit steht im Zentrum der Softwareentwicklung
  - Design ist isoliert von Technologie
    - Über Technologie wird immer nachrangig entschieden
  - Auf das Verständnis der Fachlichkeit wird viel Energie verwendet
- Design und Software spiegeln die Fachlichkeit wieder
  - Nachvollziehbar
  - Methodisch

Domain Driven Design – Was ist anders/besser?

IKS Individuelle Softwarelösungen

- Fachlichkeit und Software korrespondieren nachvollziehbar
  - Dies ist durch die Methode vorgegeben
  - In welchem Softwarebaustein ist diese Fachlichkeit umgesetzt?
  - Für welche Fachlichkeit ist dieser Baustein zuständig?
- Softwarelösung skaliert mit der Komplexität
  - Integration einer fachlichen Änderungen darf nicht mit der Größe/Komplexität des Systems komplizierter werden
  - Wachsende Komplexität bleibt wartbar
- Zusammenarbeit steht im Mittelpunkt
  - Fachexperten und Entwicklung arbeiten Hand in Hand
  - Fachexperten und Entwicklung kommunizieren technologiefrei
  - Zusammenarbeit verstärkt und vereint unterschiedliche Fähigkeiten

Worum geht es bei Domain Driven Design?

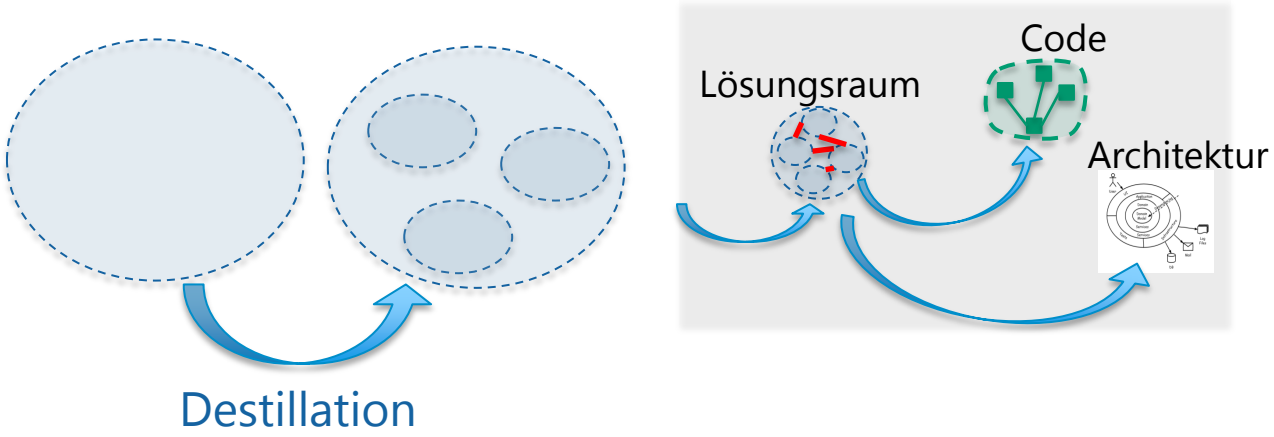
- Es geht um Sprache
- Es geht um Zusammenarbeit
- Es geht um Methoden und Verfahren
- Es geht um eine Änderung von Denkmustern

Agenda

- Warum Domain Driven Design ?
- **Strategisches Design - Konzepte im Problemraum**
- Strategisches Design - Konzepte im Lösungsraum
- Taktisches Design und Architektur
- Essentials
- Abschluss

Fachlichkeit im Zentrum der Softwareentwicklung

Problemraum



Allgemeingültige Sprache

Fachliche Dekomposition

Gemeinsames Verständnis ist alles

- Gemeinsame Sprache ist Schlüssel zum gemeinsamen Verständnis
  - Eine gemeinsames Verständnis zu entwickeln ist mühselig aber notwendig
  - Klärung der fachlichen Begriffe
- Klarheit der gemeinsamen Sprache ist Indikator für die Tiefe des Verständnisses
  - Unklare Begrifflichkeit deutet auf tieferliegende, verdeckte Konzepte hin
  - Mache Implizites explizit

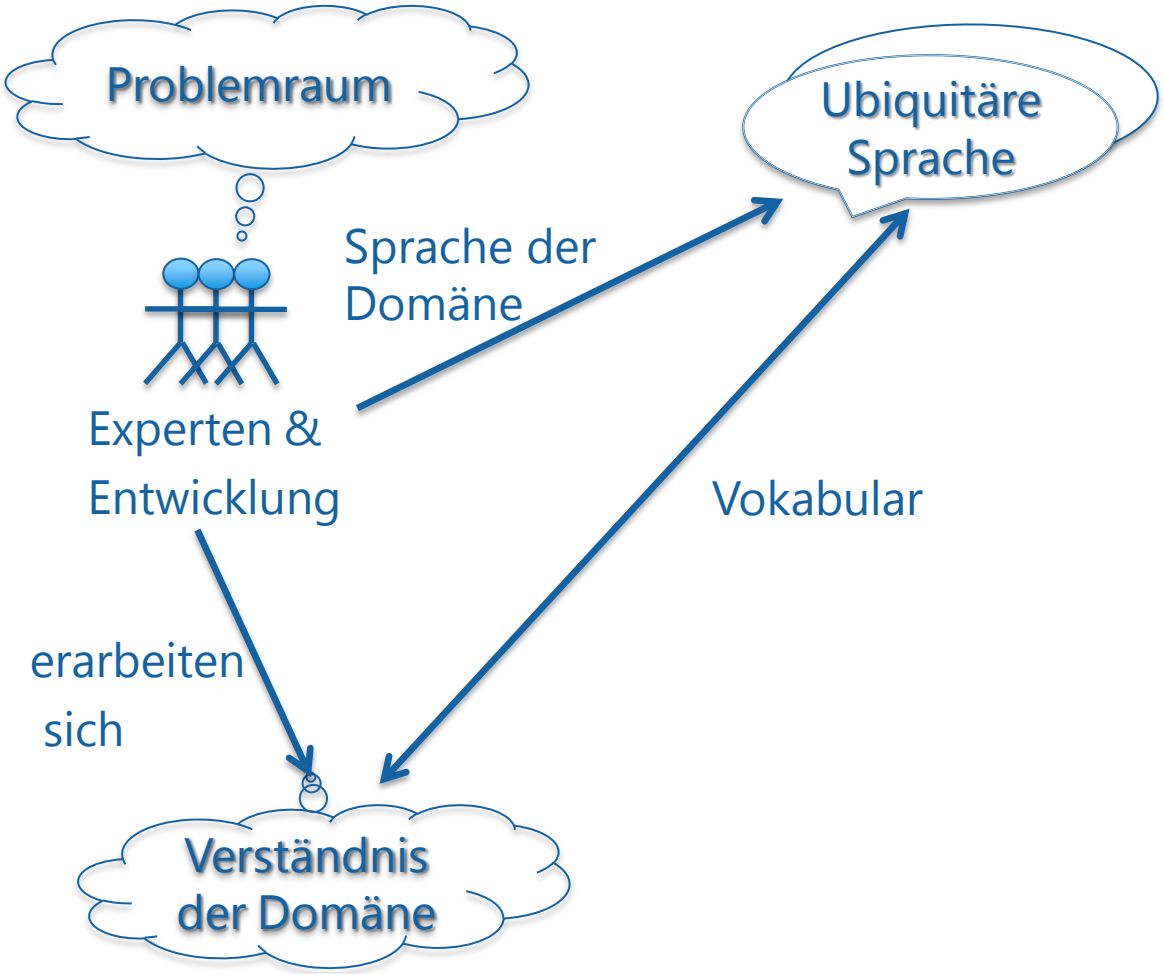
Gemeinsame Sprache ist alles

- Gemeinsames Verständnis durch gemeinsam entwickelte Sprache
  - *Ubiquitous Language*
  - Eindeutige Bedeutung innerhalb der *Ubiquitous Language*
- Ubiquitär = „allumfassend, überall vorhanden, allgegenwärtig“
  - Jeder im Projekt muss die Sprache sprechen und verstehen können
  - Alle relevanten Sachverhalte müssen sich durch die Sprache beschreiben lassen
  - Die Sprache ist in allen Phasen der Entwicklung präsent
  - Die Sprache lebt
- Die gemeinsame Sprache ist eines der wichtigsten Konzepte in DDD
  - Grundlage für gemeinsames Verständnis
  - Grundlage für durchgängiges Verständnis
  - Indikator für Durchdringung und Stabilität (der Beschreibung) der Domäne

Ubiquitäre Sprache

- Ubiquitäre Sprache entsteht während der Modellierung
  - Initial während der Analyse des Problembereichs
  - Lebt während aller Phasen der Softwareentwicklung
- Ubiquitäre Sprache muss explizit aufgeschrieben und gepflegt werden
  - Z.B. Wiki
- Ubiquitäre Sprache gehört zur Welt der Entwicklung wie auch der Domäne
  - Klärung der unterschiedlichen Sichten/Konzepte zwischen Fachexperten und Entwicklung
- Ubiquitäre Sprache durchdringt alle Phasen und Artefakte
  - … wird durchgängig genutzt
    - auch in *Code, Tests,…*
  - Ubiquitäre Sprache ist daher deutlich mehr als ein Glossar

Domain Driven Design – Problemraum



Nach *Practicing Domain-Driven Design , Millet, Leanpub*

Dekomposition des Problemraums

- Teile und Herrsche
  - Problembereich wird in abgeschlossene, kohärente, logische *Sub-Domain* aufgeteilt
  - Nach fachlichen Kriterien
  - Klare fachliche Zuordnung
  - Eine fachliche Quelle/ein fachlicher Treiber
  - Klare Abgrenzung einer Domäne

DDD-Bezeichnung für die Partitionen im Problemraums

- … drückt sich durch die ubiquitäre Sprache aus
  - Erkenntnisse während der Dekomposition beeinflussen die ubiquitäre Sprache
- … ist im Fluss
  - Eine gewisse Stabilität ist Voraussetzung für die nächsten Schritte
- … entsteht kooperativ, übergreifend, iterativ
  - DDD stellt Verfahren/Techniken zur Unterstützung der Dekomposition bereit
  - Insbesondere Methoden um *Core Domain* zu finden

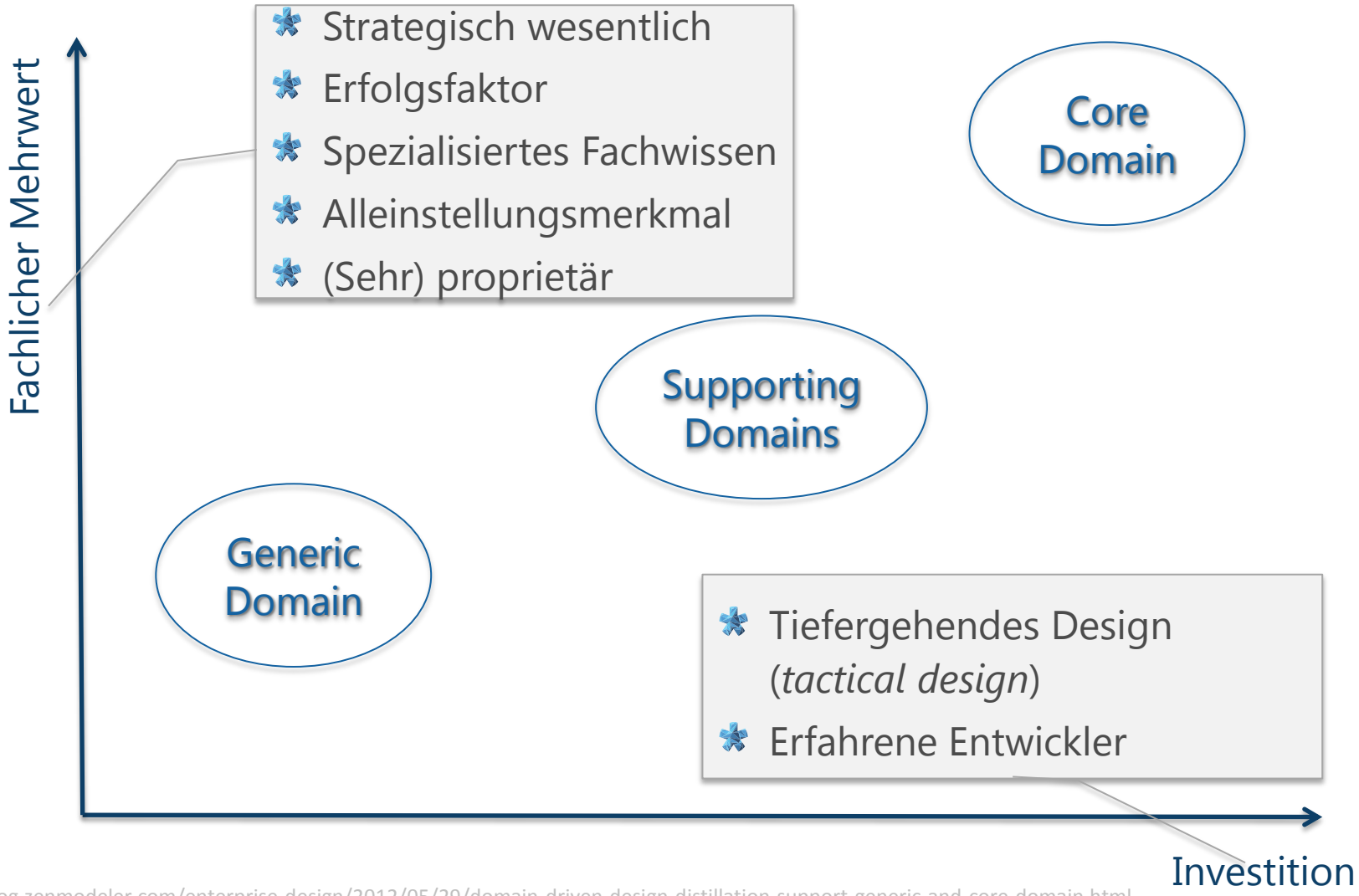
Dekomposition des Problemraums

- … orientiert sich an ***business capabilities***
- … kann zugeordnete Geschäftsobjekte besitzen
- … kann Bezug zu Geschäftsvorgängen besitzen

(Fachliche) Kompetenzen, um Geschäftsziele zu erreichen

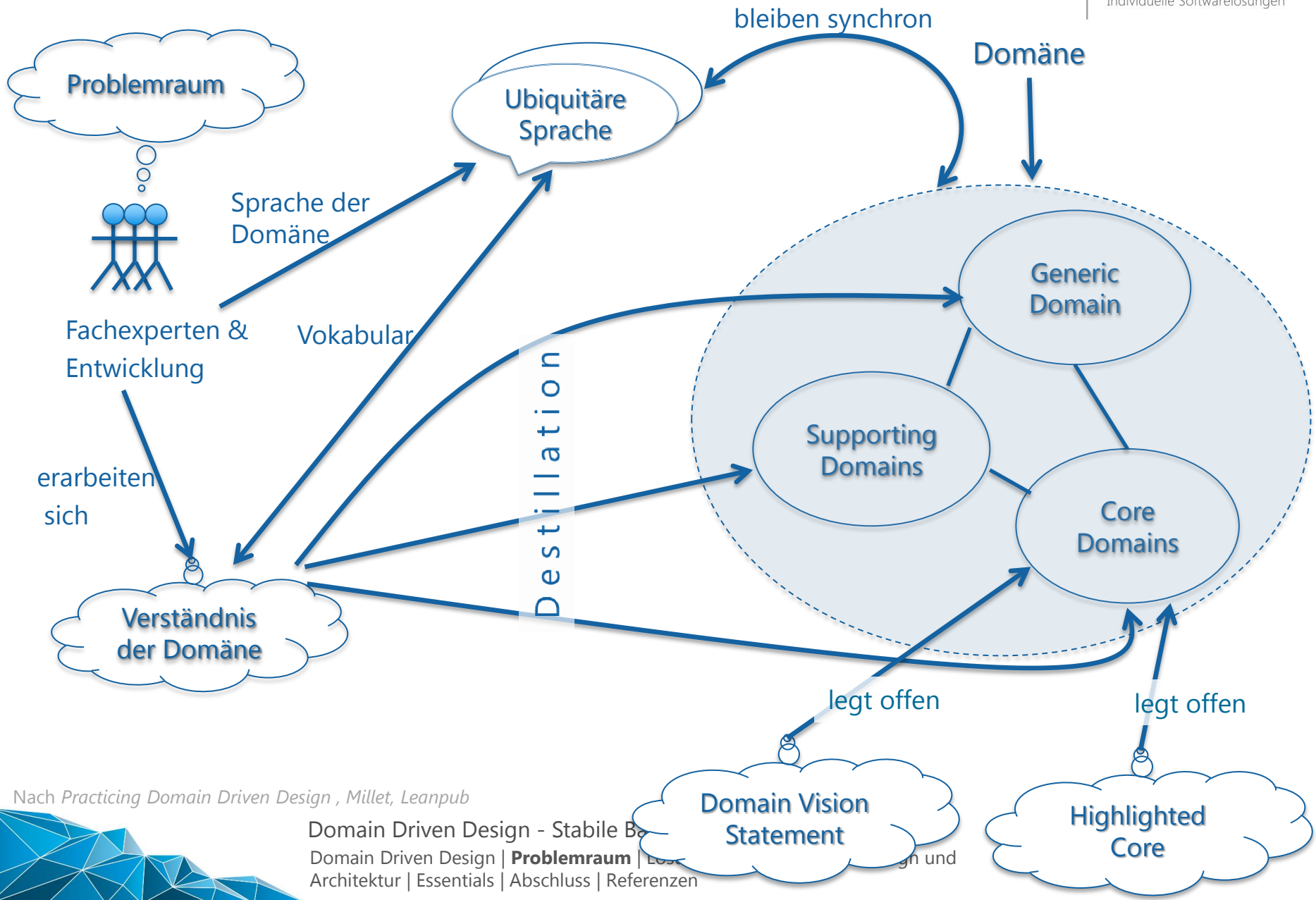
- Bewertet Subdomains nach Relevanz für mein System
  - Bewertung nach Kritikalität/Wichtigkeit für die Geschäftsanwendung
  - Destillation (Literaturbegriff : *distillation*)

Prioritäten nach fachlichem Mehrwert



http://blog.zenmodeler.com/enterprise-design/2012/05/29/domain-driven-design-distillation-support-generic-and-core-domain.html

Domain Driven Design – Problemraum



Nach *Practicing Domain Driven Design , Millet, Leanpub*

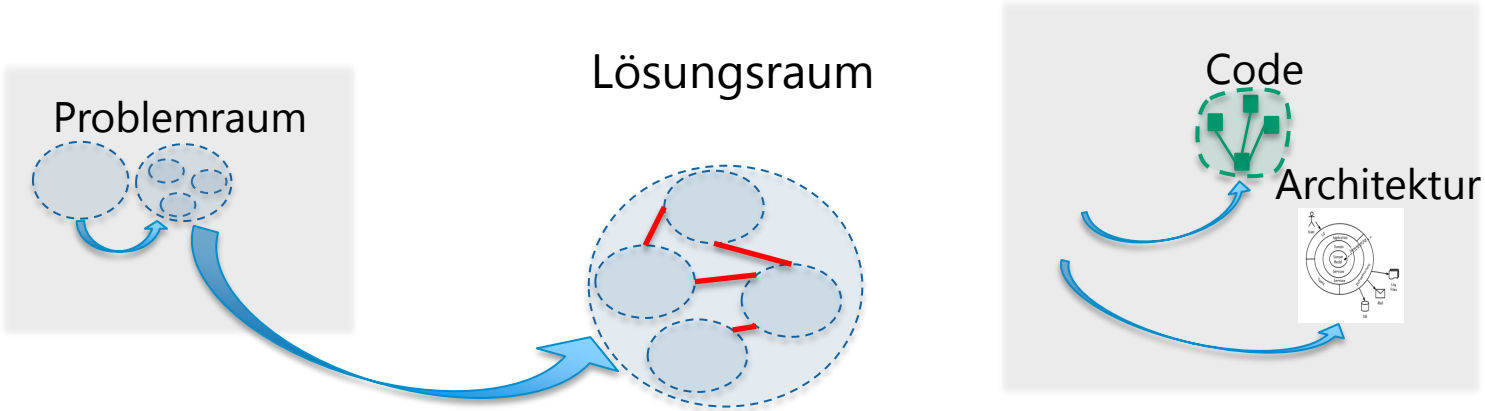
Wie werden die Ziele der Dekomposition erreicht?

- Experten und Entwicklung arbeiten zusammen
  - Kooperativ
  - Kommunikativ
- DDD ist ein andauernder Lernprozess über die Fachlichkeit
  - Iterativ
  - Explorativ
  - Kompetenzgewinn spiegelt sich in Ergebnissen wieder
- Sprache treibt und bestimmt das Vorgehen
  - Ubiquitäre Sprache beschreibt das gemeinsame Verständnis der Domäne
  - Ubiquitäre Sprache bestimmt die Dekomposition (*Destillation*)

Agenda

- Warum Domain Driven Design ?
- Strategisches Design - Konzepte im Problemraum
- **Strategisches Design - Konzepte im Lösungsraum**
- Taktisches Design und Architektur
- Essentials
- Abschluss

Fachlichkeit im Zentrum der Softwareentwicklung



Fachliche Kriterien bestimmen das strategische Design

Begriffe und sprachlicher Kontext

Kollaboration

Überführe das WAS in das WIE - Lösungsraum

- Schritt 1: Überführe *Sub-Domains* in den Lösungsraum
  - *Bounded Context*
- Schritt 2 : Beschreibe die Zusammenarbeit der einzelnen *Bounded Contexts*
  - *Context Mapping (of Bounded Contexts)*

Bedeutung und sprachlicher Kontext

- Wo kommt Mehrdeutigkeit von Begriffen her?
- Begriffe leben in unterschiedlichen sprachlichen Kontexten
  - Haben dort unterschiedliche Bedeutung
  - Klassiker: Auftrag, Kunde, Stammdaten, Person, Bestellung, Produkt, Buch
- Bedeutung ist immer auf einen (sprachlichen) Kontext bezogen
  - Dieser Kontext ist in der gemeinsamen Sprache festzuhalten
- DDD fördert explizit unterschiedliche sprachliche Kontexte
  - Diese bleiben getrennt, werden nicht vereinheitlicht

Beispiel für einen sprachlichen Kontext

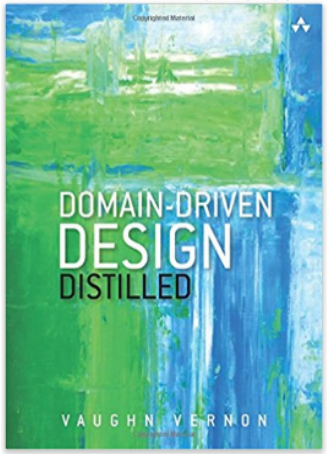
IKS Individuelle Softwarelösungen

Buchbestellung

Fremdsprachige Bücher | Erweiterte Suche | Bestseller | Neuheiten & Vorbesteller | Angebote | Englische Bücher | Französische Bücher | Spanische Bücher | Italienische Bücher | Niederländische Bücher | Alle Fremdsprachen

Domain-Driven Design Distilled und über 4,5 Millionen weitere Bücher verfügbar für **Amazon Kindle**. Erfahren Sie mehr

Fremdsprachige Bücher › Computer & Internet › Informatik



Alle 2 Bilder anzeigen

Domain-Driven Design Distilled (Englisch) Taschenbuch – 23. Mai 2016

von Vaughn Vernon (Autor)

10 Kundenrezensionen

Alle 2 Formate und Ausgaben anzeigen

Kindle Edition	**Taschenbuch**
EUR 21,24	**EUR 24,99**

Lesen Sie mit unserer **kostenfreien App**

Lieferung Freitag, 6. Apr.: Bestellen Sie innerhalb 3 Stunden und 29 Minuten per **1-Tages-Zustellung** an der Kasse. Siehe Details.

45 neu ab EUR 23,38 | 1 gebraucht ab EUR 33,29

Domain-Driven Design (DDD) software modeling delivers powerful results in practice, not just in theory, which is why developers worldwide are rapidly moving to adopt it. Now, for the first time, there's an accessible guide to the basics of DDD: What it is, what problems it solves, how it works, and how to quickly gain value from it.

Concise, readable, and actionable, ***Domain-Driven Design Distilled*** never buries you in detail–it

Mehr lesen

Teilen

EUR 24,99

Alle Preisangaben inkl. deutscher USt.

Weitere Informationen.

Kostenlose Lieferung.

Auf Lager.

Verkauf und Versand durch Amazon.

Geschenkverpackung verfügbar.

Menge: 1

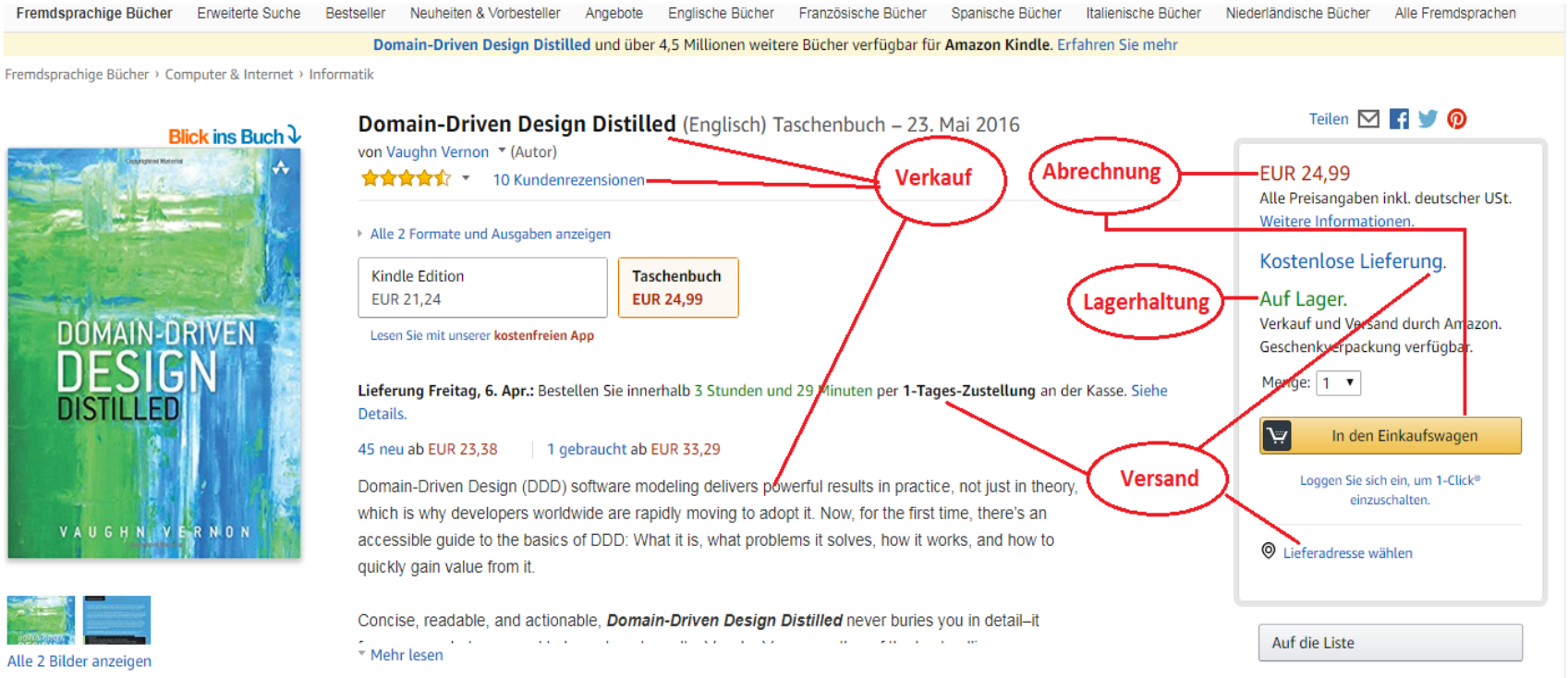
In den Einkaufswagen

Loggen Sie sich ein, um 1-Click® einzuschalten.

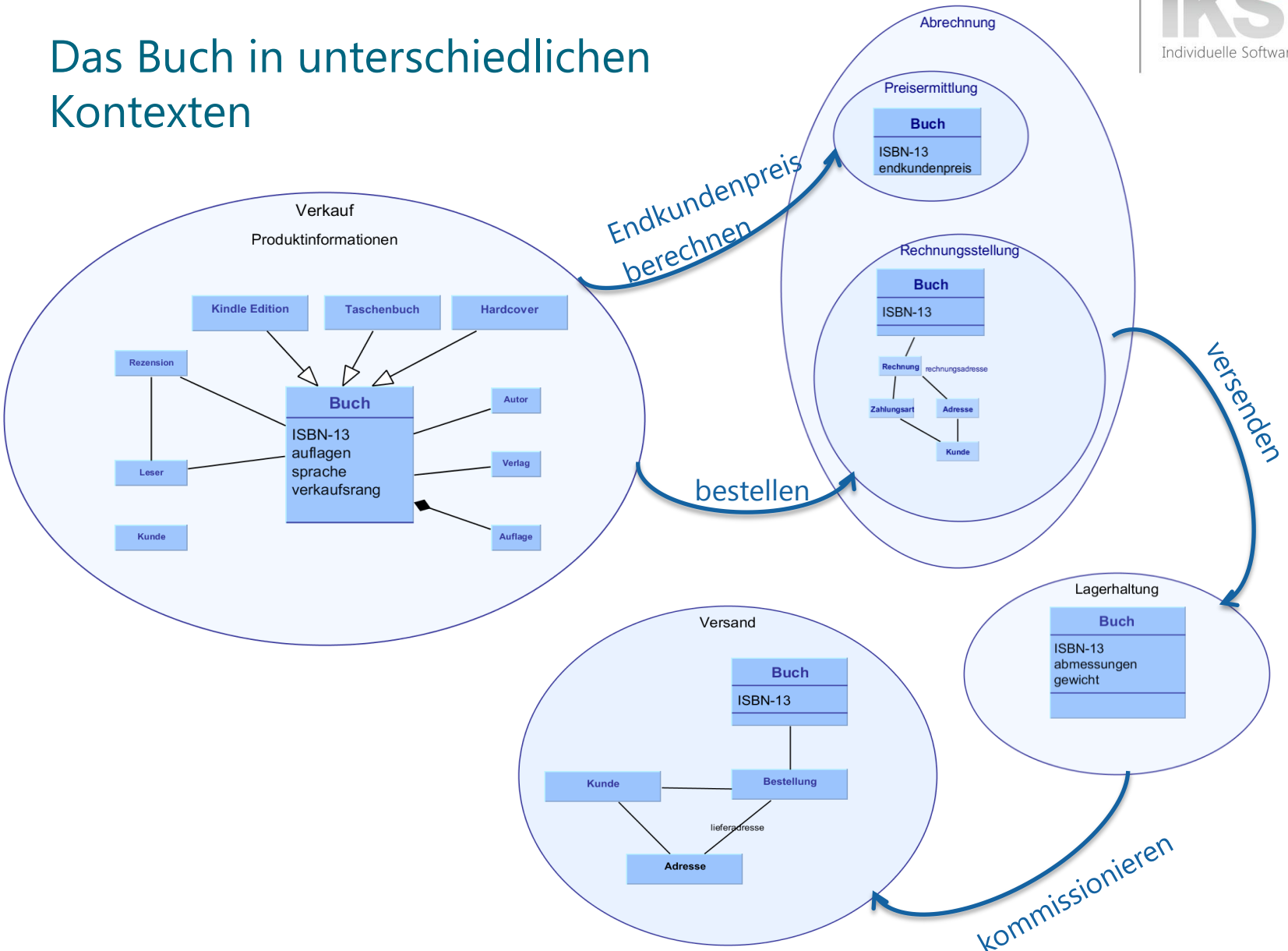
Lieferadresse wählen

Auf die Liste

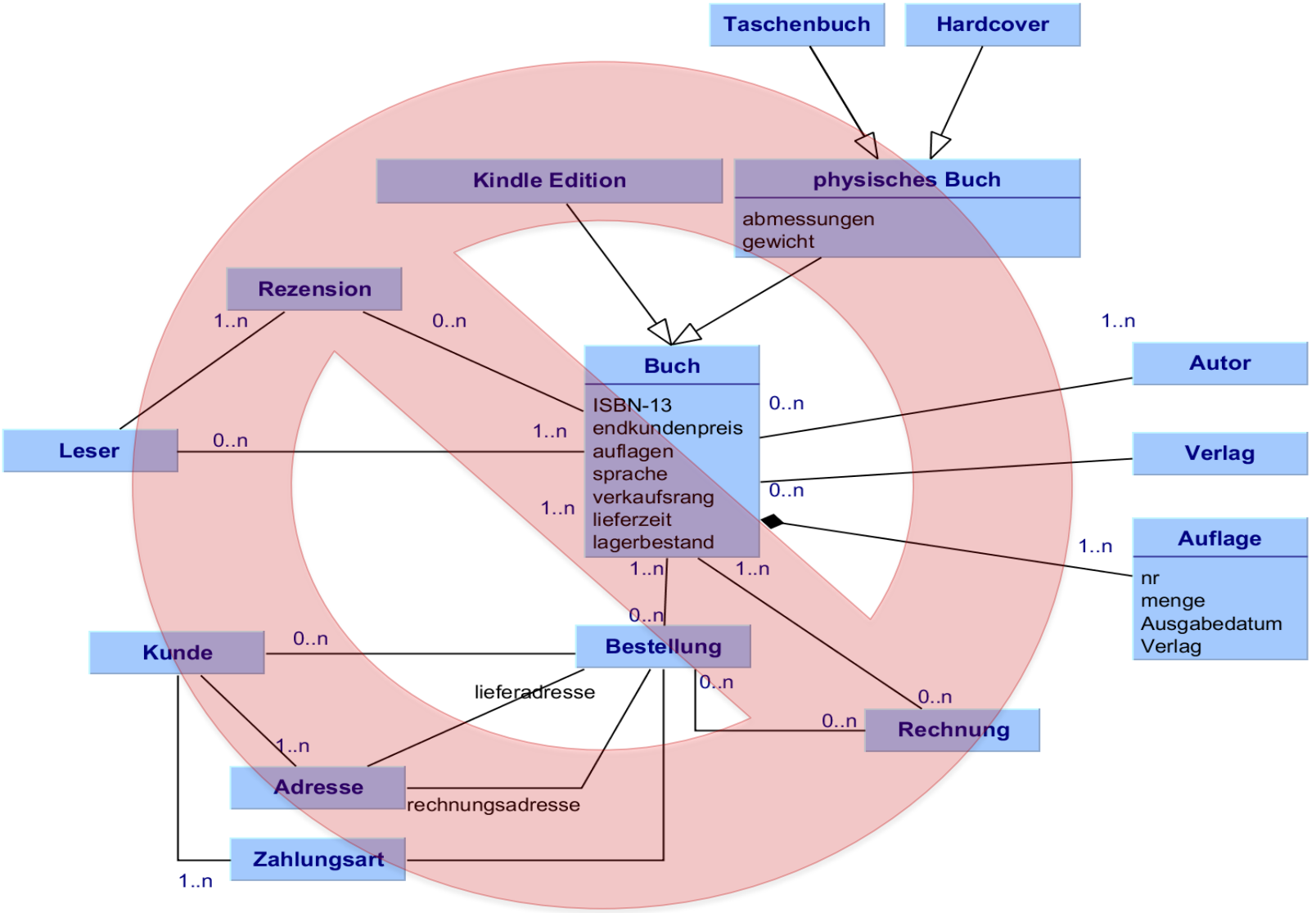
Buchbestellung



Das Buch in unterschiedlichen Kontexten



Buch - Unternehmensmodell



Buch in unterschiedlichen Kontexten

- Das Konzept *Buch* wird auf die einzelnen Bounded Contexts verteilt
  - Es hat in jedem Kontext eine unterschiedliche Bedeutung
  - ISBN13 hält alle Ausprägungen zusammen
- *Buch* kann sich in jedem Kontext unabhängig weiterentwickeln
- Dieses Verständnis benötigt unabdingbar die ubiquitäre Sprache
  - Bounded Context als sprachlicher Kontext

Bounded Context (1)

- Bounded Context repräsentiert Sub-Domänen im Lösungsraum
  - Faustregel : Eine Sub-Domäne wird zu einem Bounded Context
  - *Bounded Contexts* tragen Rahmenbedingungen Rechnung
- Bounded Context ist ein eigener sprachlicher Kontext
  - Begriffe bezüglich des Kontexts können in einem anderen Kontext eine andere Bedeutung haben
- Bounded Context repräsentiert fachlich unabhängige Bereiche
  - abgeleitet aus *Sub-Domains*
- Bounded Context hat explizit formulierte Grenzen
  - Bounded Contexts kommunizieren (nur) über öffentliche Schnittstellen

Bounded Context (2)

- Bounded Context ist (weitestgehend) unabhängig
  - Das Konzept der unabhängigen Sprachkontexte forciert Autonomie
  - Autonomie ist der Schlüssel zur Beherrschung von langlebigen Systemen
- Bounded Context ist kohärent
  - So kompakt wie möglich
  - So umfangreich wie notwendig
  - Faustregel: *Bounded Context* orientiert sich an einem *Aggregate*
- Bounded Context wird durch ein Team entwickelt
  - Domänenexperten, Entwickler, Test, …

Autonomie, Autonomie, Autonomie ….

Autonomie, um …

- … fachliche Entscheidungen innerhalb eines *Bounded Context* zu treffen
- … Auswirkungen des (Daten-)modells auf *Bounded Context* zu beschränken
  - Kleine lokale Datenmodelle versus Unternehmensmodell

Context Mapping und Collaboration Pattern

DDD dokumentiert die Beziehung der Bounded Contexts

Context Mapping

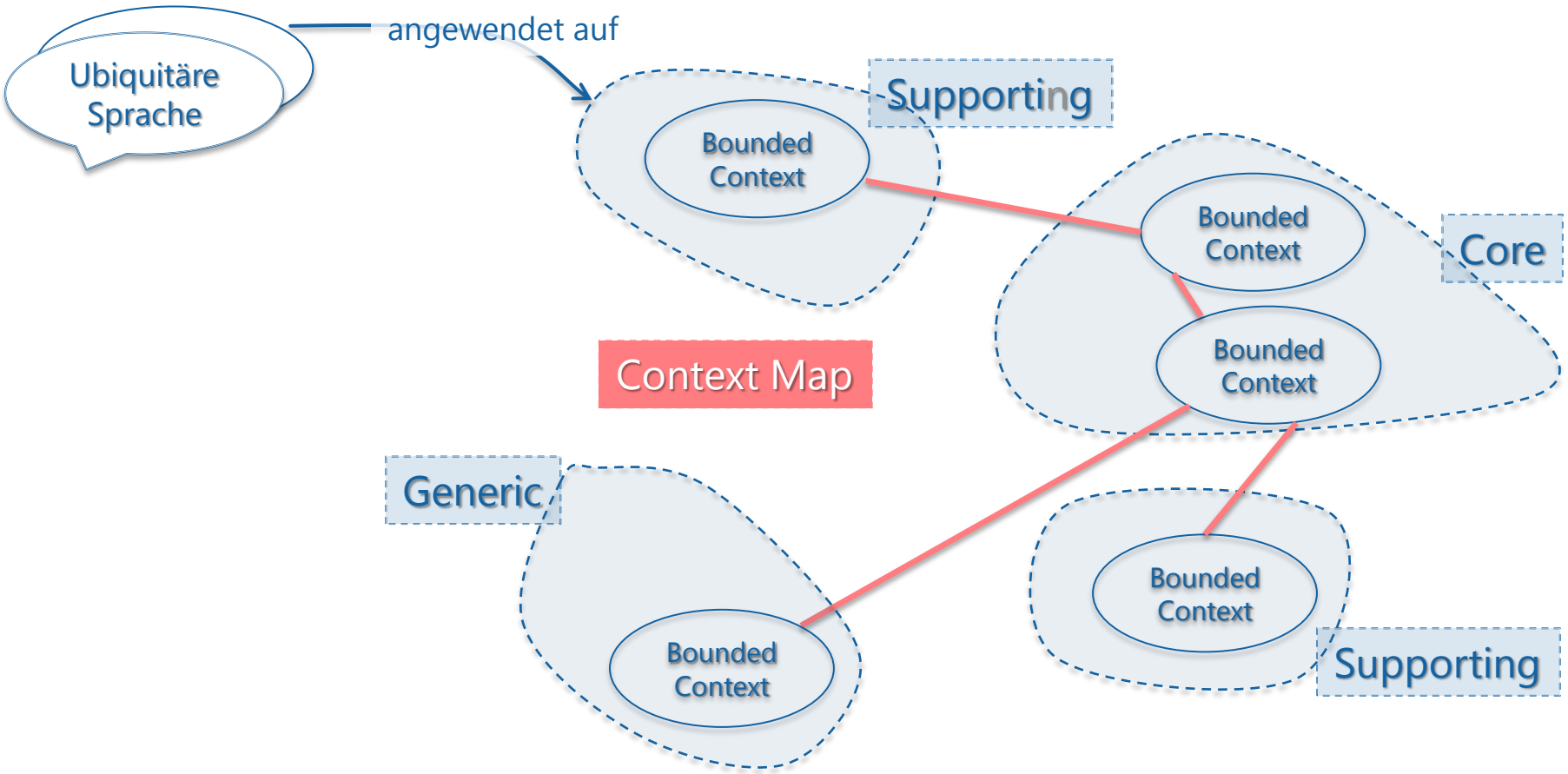
DDD beschreibt Muster und Klassifizierung der Zusammenarbeit

Collaboration Pattern

Überblick behalten - Context Map

- Dokumentation, wie die einzelnen *Bounded Contexts* zusammenspielen
  - Wer konsumiert wen?
  - Bietet Überblick über die Landschaft der *Bounded Contexts*.
- *Context Map* klärt die Beziehung zwischen *Bounded Contexts* und *Teams*
  - Welches Team zu welchem *Bounded Context*?
- *Context Map* beschreibt Kommunikationsmuster
  - In welcher Form kooperieren die Teams?
  - DDD beschreibt Muster der Zusammenarbeit*: Collaboration Patterns*

Domain Driven Design - Lösungsraum



Nach *Practicing Domain-Driven Design , Millet, Leanpub*

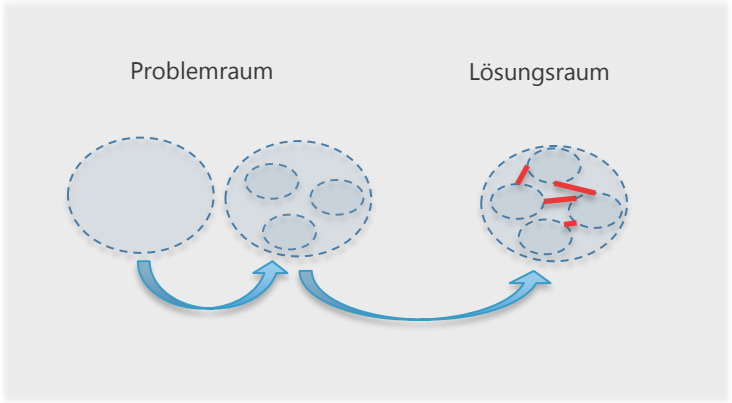
Wie werden die Ziele des strategischen Designs erreicht?

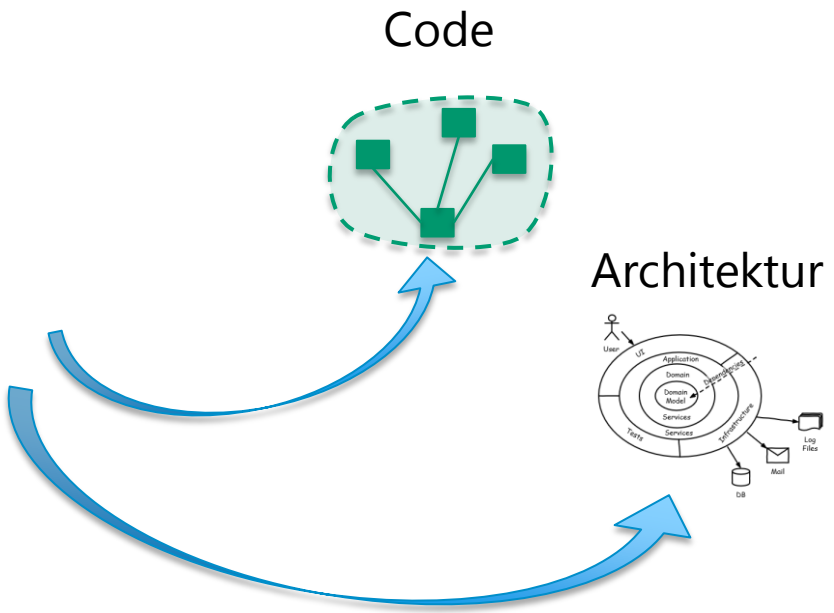
- Fachexperten und Entwicklung arbeiten zusammen
  - Kooperativ
  - Kommunikativ
- DDD ist ein andauernder Lernprozess über die Fachlichkeit
  - Iterativ
  - Explorativ
  - Kompetenzgewinn spiegelt sich in Ergebnissen wieder
- Sprache treibt und bestimmt das Vorgehen
  - Ubiquitäre Sprache beschreibt das gemeinsame Verständnis der Domäne
  - *Bounded Context* bestimmt einen sprachlichen Kontext

Agenda

- Warum Domain Driven Design ?
- Strategisches Design - Konzepte im Problemraum
- Strategisches Design - Konzepte im Lösungsraum
- **Taktisches Design und Architektur**
- Essentials
- Abschluss

Fachlichkeit im Zentrum der Softwareentwicklung





Trennung von Fachlichkeit und Technologie

Fachliche Durchgängigkeit durch tiefgehende Modellierung

Agenda

- Warum Domain Driven Design ?
- Strategisches Design - Konzepte im Problemraum
- Strategisches Design - Konzepte im Lösungsraum
- Taktisches Design und Architektur
- **Essentials**
- Abschluss

Erosion ihrer langlebigen Software

- Es dauert immer länger um Releases freizugeben
- Es ist fast nicht mehr möglich neue Technologien zu integrieren
- Fachliche Änderungen verstreuen sich über Ihre gesamte Anwendung
  - Oft ist gar nicht klar, welche Teile des Systems betroffen sind
- Kleine Änderungen haben große Auswirkungen
  - Zum Beispiel aufwendige Abnahme des Gesamtsystems
- Ihre Datenbankstruktur ist unübersichtlich
  - Es ist nicht klar, welche Tabellen miteinander zu tun haben

Dann wird es Zeit, über *Domain Driven Design* nachzudenken

Domain First – Domäne steht im Zentrum

- Vorgehen stellt Fachlichkeit in den Mittelpunkt des Designs
  - Kollaboration zwischen Fachexperten und Entwicklung
  - Gemeinsames Verständnis der Fachlichkeit
- *Domain Driven Design* zielt auf eine fachliche Durchgängigkeit/Integrität
  - Bruch zwischen fachlicher Anforderung und technischer Umsetzung ist behoben
  - Fachliche Durchgängigkeit unterstützt die Beherrschung langlebiger Systeme
- *Domain Driven Design* zielt auf ein System autonomer Bausteine
  - Autonomie ist der Schlüssel zur Beherrschung langlebiger Systeme

Domain First – Domäne steht im Zentrum

- Domänen-orientiertes Design
  - Explizites Design der reinen Fachlichkeit/Domänen
- Domänen-orientierte Dekomposition
  - *Domain Distillation* (*Problemraum*)
  - *Bounded Context* (*Lösungsraum*)

Domain Driven Design

DDD beschreibt

- Methoden und Prozesse
  - Domänen und Destillation
  - Ubiquitäre Sprache
  - Strategisches Design
  - Taktisches Design

Siehe [Evans]

- Verfahren und Techniken, um
  - Fachexperten und Entwicklung zusammenarbeiten zu lassen
  - Domäne in Subdomänen zu zerlegen
  - Die richtige Core-Domäne zu finden
  - Bounded Context voneinander abzugrenzen
  - Eine Bounded Context en Detail zu designen

Einsatz von DDD will überlegt sein

- Strategisches Design
  - Ubiquitäre Sprache, Domänen-Destillation und Bounded Context
  - . . . sind für das gesamte System zuträgliche Konzepte
  - Designansätze tragen in allen Domänen/Projekten

- *Domain Driven Design* in voller Ausprägung
  - Eignet sich für fachlich komplexe Systeme
  - D.h. inkl. taktischem Design, Architektur + DDD-Prozess
  - Konzentration auf *Core Domain*
  - Ist sehr aufwendig

- *Domain Driven Design* ist nicht die Lösung aller Probleme

Auswirkungen von *Domain Driven Design*

- Entwickler bekommen ein neues Rollenverständnis
- Fachexperten bekommen ein neues Rollenverständnis
- Teams stehen im Zentrum der Entwicklung
- Zusammenarbeit in crossfunktionalen Teams
- Entwicklungsprozess muss sich anpassen
- *Agiles Mindset* unterstützt den Erfolg von Domain Driven Design

Wie kann es losgehen?

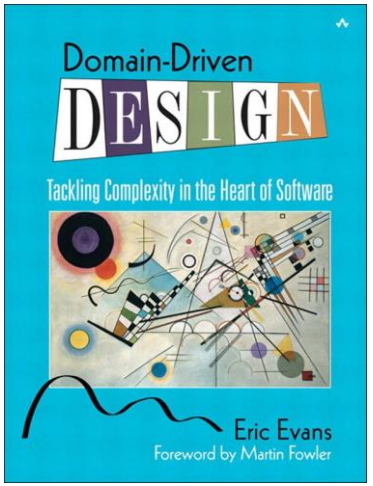
- Fachexperten müssen den Wert von Domain Driven Design sehen
  - Müssen ggfs. überzeugt werden
- Entwickeln Sie ein ubiquitäre Sprache
- Lassen Sie Raum für Exploration
  - Die „richtige" Lösung muss sich entwickeln können
  - Anforderungen so lange hinterfragen bis diese stabil sind
- Fangen Sie klein an
  - Piloten, um die Verfahren einzuüben

Abschließende Bemerkungen

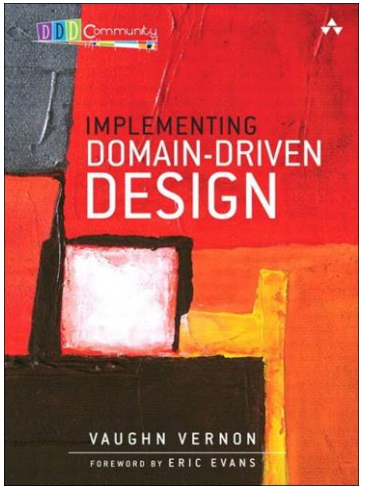
- Exakte Ausprägung von *Domain Driven Design* hängt vom Umfeld ab
  - Zusammenarbeit zwischen Fachexperten und Entwicklung
  - Einbettung in agiles Vorgehen
- *Domain Driven Design* benötigt Erfahrung
  - Wir können nur grundlegende Techniken und Vorgehensweisen vorstellen
  - Siehe *Analysis Patterns: Reusable Object Models* von *Martin Fowler*
- *Domain Driven Design* stabilisiert ihre Systeme
- Es lohnt sich

Leseempfehlungen

Domain-Driven Design: Tackling Complexity in the Heart of Software von *E. Evans*



Implementing Domain-Driven Design von *Vaughn Vernon*



Vielen Dank!

Fragen?

Weiterführende Literatur

- https://weblogs.asp.net/ngur/Business-capabilities-or-business-processes-
- http://codebetter.com/gregyoung/2012/02/29/the-context-game-2/
- https://hackernoon.com/how-to-define-service-boundaries-251c4fc0f205
- https://hackernoon.com/wrong-ways-of-defining-service-boundaries-d9e313007bcc
- https://www.infoq.com/articles/ddd-contextmapping

- https://leanpub.com/Practicing-DDD; Scott Millett
- Patterns, Principles, and Practices of Domain-Driven Design; Scott Milet
- Rechtin E. and Maier M. The art of systems architecting, CRC Press, 2002, ISBN 0-8493-0440-7.
- Implementing Domain-Driven Design; Vaughn Vernon
- https://www.microsoftpressstore.com/articles/article.aspx?p=2248811
- *Analysis Patterns: Reusable Object Models* von *Martin Fowler*

Impulsvorträge für Ihr Unternehmen

IKS
Individuelle Softwarelösungen

- Überblick über das gesamte Angebot an Impulsvorträgen unter: www.iks-gmbh.com/impulsvortraege

- Ihr Nutzen:
  - Unabhängiges, aktuelles Expertenwissen.
  - Individuell auf Ihr Publikum und Ihr Unternehmen zugeschnittene Vorträge.
  - Referenten mit langjähriger und branchenübergreifender Expertise in der IT-Beratung.
  - Praxisnahe Vorträge, die aus Projektarbeit entstanden sind, frei von Produktwerbung.
  - Ideale Ergänzung für Ihre Führungskräftetreffen, Abteilungsmeetings, Hausmessen, Innovation Days, Konferenzen, Open Spaces, Kick-off-Meetings oder Zukunftsworkshops.

IKS
Individuelle Softwarelösungen

WWW.IKS-GMBH.COM

Projekte. Beratung. Spezialisten.

IKS
Individuelle Softwarelösungen

IKS Gesellschaft für Informations-und Kommunikationssysteme mbH

T. +49 2103-5872-0 | www.iks-gmbh.com