CLOUD NATIVE COMPUTING

CLOUD-NATIVE

Unit:

Domain Driven Design

(2) Strategisches Design

Domains + Ubiquitous Language

PROF. DR.
NANE KRATZKE

creative commons

CLOUD NATIVE COMPUTING

Urheberrechtshinweise

Diese Folien werden zum Zwecke einer praktikablen und pragmatischen Nutzbarkeit im Rahmen der **CCo 1.0 Lizenz** bereitgestellt.

Sie dürfen die Inhalte also kopieren, verändern, verbreiten, mit eigenen Inhalten mixen, auch zu kommerziellen Zwecken, und ohne um weitere Erlaubnis bitten zu müssen.

Eine Nennung des Autors ist nicht erforderlich (aber natürlich gern gesehen, wenn problemlos möglich).

Diese Folien sind insb. für die Lehre an Hochschulen konzipiert und machen daher vom **§51 UrhG (Zitate)** Gebrauch.

Die CCo Lizenz überträgt sich nicht auf zitierte Quellen. Hier sind bei der Nutzung natürlich die Bedingungen der entsprechenden Quellen zu beachten.

Die Quellenangaben finden sich auf den entsprechenden Folien.

KAPITEL 14

Domain Driven Design

14.1 Fachlichkeit, Fachlichkeit, Fachlichkeit

14.2 Strategisches Design

- Subdomänen
- Ubiquitous Language
- Bounded Context
- Context Mapping

14.3 Taktisches Design

- Oft genutzte Pattern für Geschäftslogik
  (ETL, Active-Record, Domain Model, Event-Sourcing)
- Oft genutzte Architektur-Pattern
  (Layered Architecture, Ports & Adaptor, CQRS)

14.4 Zusammenfassung

INHALTE

Domain Driven Design

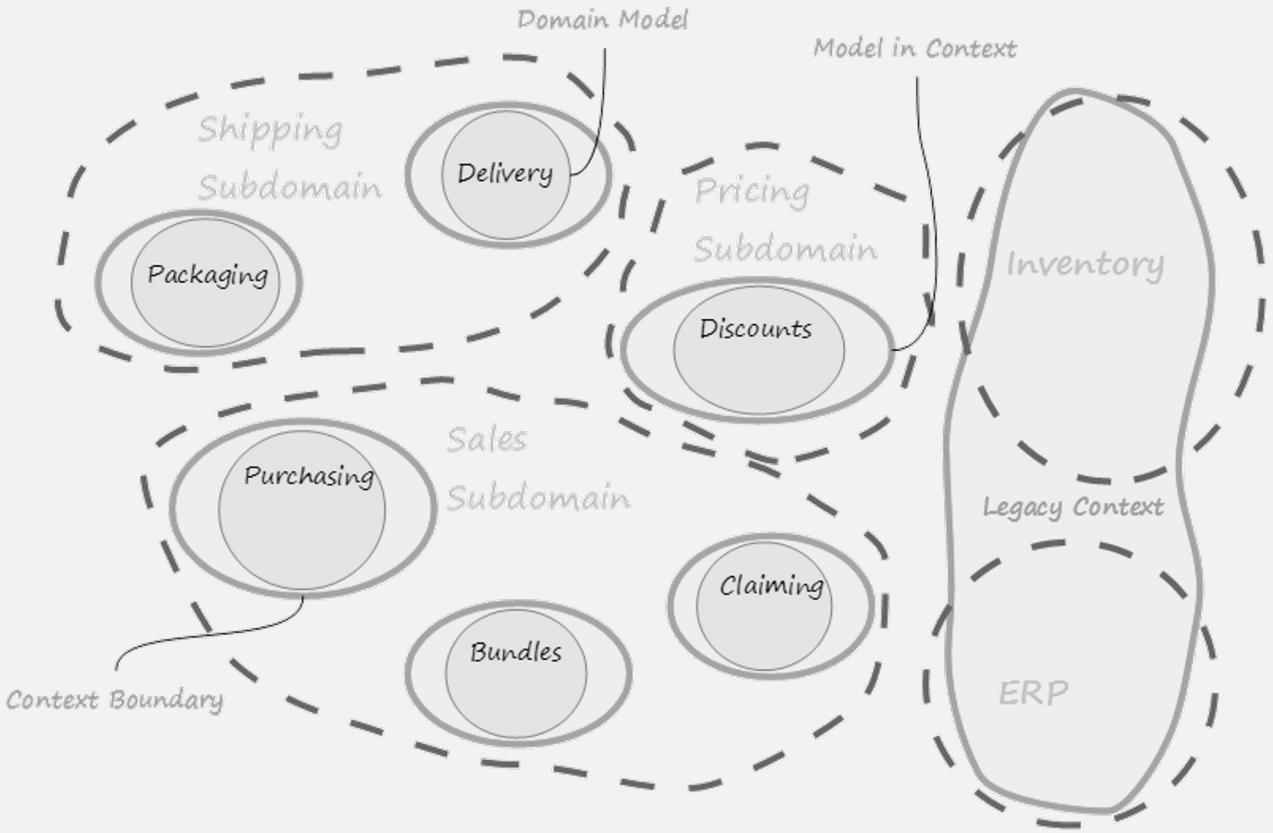
- Was ist das?
- Effektives Software Design
- Strategisches Design
- Taktisches Design

Strategisches Design

- Subdomains
- Ubiquitous Language
- Bounded Context
- Context Mapping

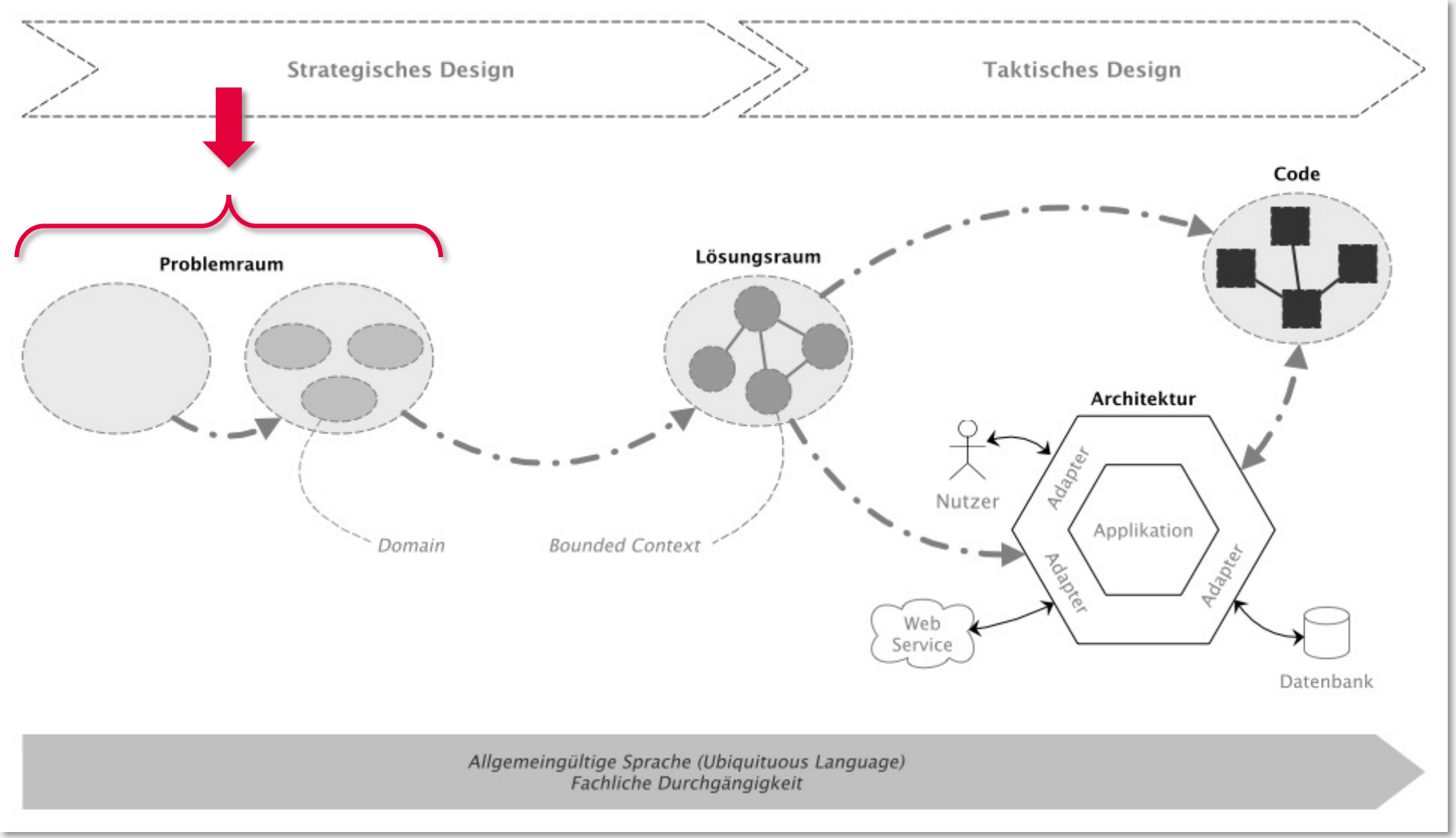
Taktisches Design

- Business Logic Pattern
- Architectural Pattern



DOMAIN DRIVEN DESIGN

Fachlichkeit als Treiber der Softwareentwicklung



STRATEGISCHES DESIGN

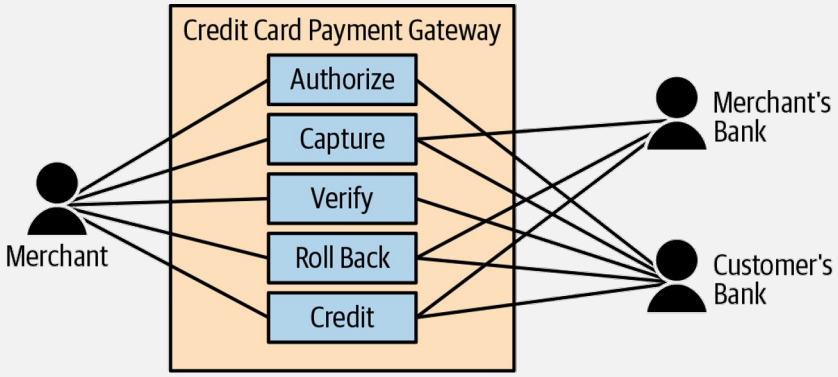
Analyse von Business Domains

Ein Geschäftsfeld (Business Domain) definiert den gesamten Tätigkeitsbereich einer Organisation oder eines Unternehmens. Im Allgemeinen ist es die Dienstleistung, die Einrichtungen Kunden anbieten. Ein Beispiel:

- DHL => Paketdienst
- Starbucks => Kaffee
- Karstadt => Einzelhandel

Ein Unternehmen kann in mehreren Geschäftsbereichen tätig sein. Zum Beispiel bietet Amazon sowohl Einzelhandels- als auch Cloud-Computing-Dienste an.

Unternehmen wandeln ggf. ihre Geschäftsbereiche auch im Verlaufe der Zeit. Nokia war bspw. im Laufe seiner Geschichte in so unterschiedlichen Bereichen wie Holzverarbeitung, Gummiherstellung, Telekommunikation und Mobile Devices Geräte tätig.



Um die Ziele seiner Geschäftsdomäne zu erreichen, muss ein Unternehmen in mehreren Subdomänen operieren. Eine Subdomäne ist ein feingranularer Bereich der Geschäftsaktivität. Alle Subdomänen zusammen ergeben die Geschäftsdomäne des Unternehmens.

Oft korrelieren Subdomains mit den Abteilungen oder anderen Organisationseinheiten des Unternehmens. Zum Beispiel könnte ein Online-Handelsshop Subdomains wie Katalogmanagement, Werbung, Buchhaltung, Support, Kundenbeziehungen, Lieferantenbeziehungen und andere enthalten.

Aus technischer Sicht ähneln Subdomänen einer Menge von zusammenhängenden Anwendungsfällen.

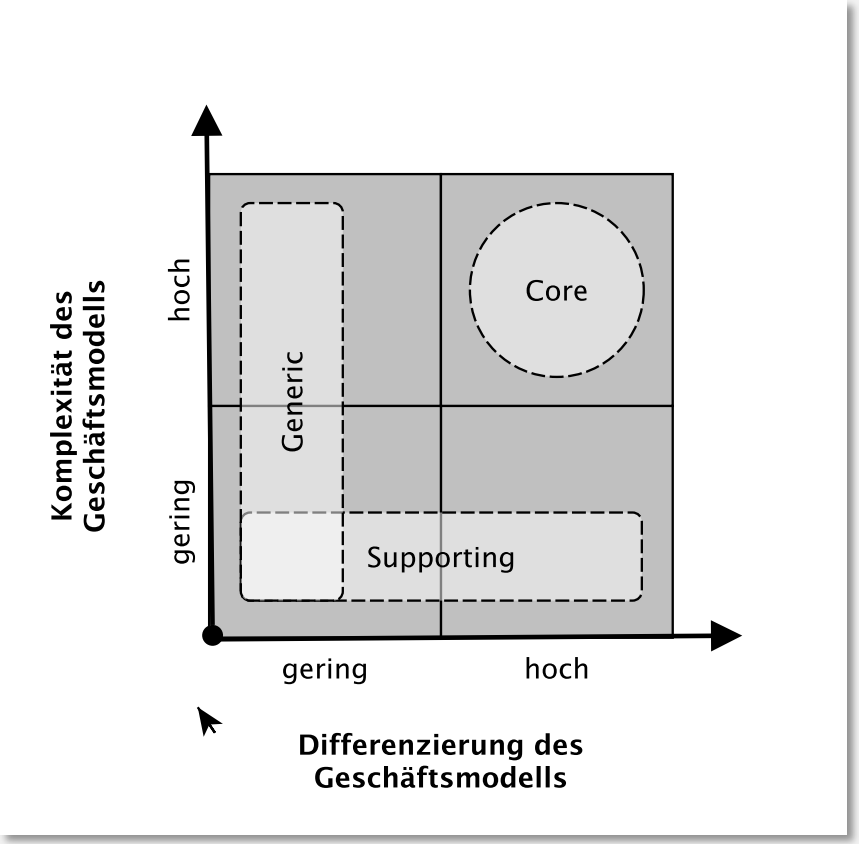
STRATEGISCHES DESIGN

Was ist eine Domäne?

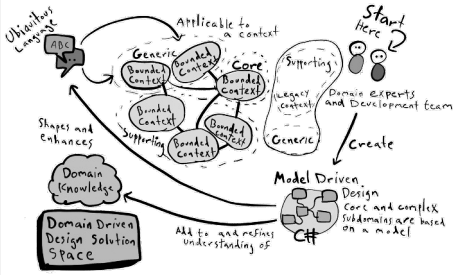
Unter einer Domäne versteht man im allgemeinen einen bestimmten Tätigkeits- oder Wissensbereich.

Das Domänenkonzept ist sehr breit und abstrakt. Um es konkreter und greifbarer zu machen, ist es sinnvoll, es in kleinere Teile aufzuteilen, die Subdomains genannt werden.

Solche Subdomains werden daher häufig in drei Kategorien unterteilt.



Alle Subdomänen, unabhängig von der Kategorie, sind wichtig für Gesamtlösungen. Fehlt eine Domäne wird die Lösung unvollständig sein. Die genannten Domänen erfordern jedoch einen unterschiedlichen Aufwand und können auch unterschiedliche Anforderungen an Qualität und Vollständigkeit haben.

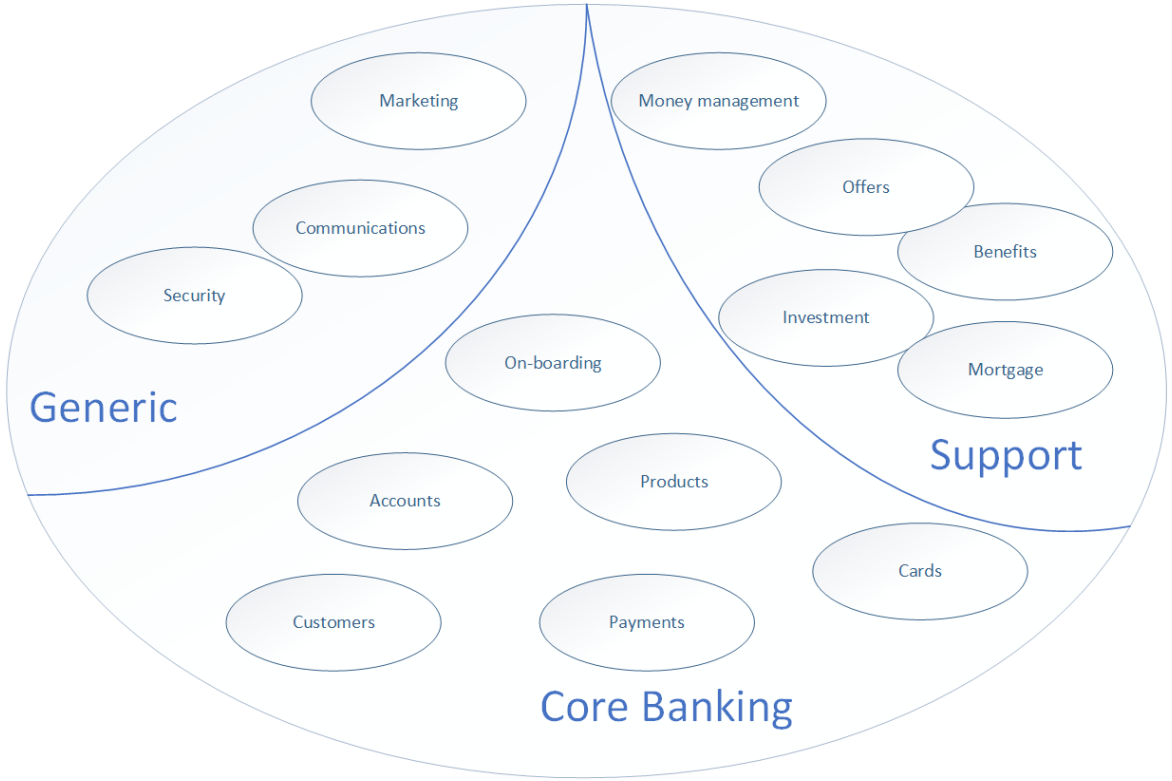


- *Core Domain (Kerndomäne)*
- *Supporting Subdomain*
- *Generic Subdomain*

STRATEGISCHES DESIGN

Was ist eine Core Subdomain? - Kerndomäne?

- **Kerndomäne** unterscheidet Unternehmen von Konkurrenten durch neue Produkte oder Prozessoptimierung
- Einfach zu implementierende Kern-Subdomänen bieten nur kurzlebigen Vorteil, daher sind sie **komplex mit hohen Eintrittsbarrieren**
- **Investitionen** in System- und Softwareentwicklung sollten **primär in die Kerndomäne fließen**
- Kern-Subdomains müssen intern implementiert werden und können nicht gekauft oder übernommen werden
- Auslagerung der Implementierung einer Core-Subdomain wäre unklug.



STRATEGISCHES DESIGN

Was ist eine Supporting Subdomain?

Im Gegensatz zu den Kern-Subdomains bieten **unterstützende Subdomains** keinen Wettbewerbsvorteil für Unternehmen. Sie unterstützen lediglich das Kerngeschäft des Unternehmens.

Beispiel:
Kern-Domain eines Online-Werbeunternehmens:

- Anpassung der Anzeigen an Besucher
- Optimierung der Anzeigenwirksamkeit
- Minimierung der Werbekosten

Supporting-Domain eines Online-Werbeunternehmens:

- Katalogisierung von Kreativmaterialien ist Teil der Kern-Subdomains
- Verwaltung von Kreativmaterialien hat keinen Einfluss auf Gewinne.

Komplexität und Änderungsgeschwindigkeit

- Unterschied zwischen Kern- und unterstützenden Subdomains ist die **Komplexität der Geschäftslogik** als Hauptunterschied
- Geschäftslogik von unterstützenden Subdomains basiert meist auf einfachen ETL-Operationen und CRUD-Schnittstellen
- Unterstützende Subdomains ändern sich im Gegensatz zu Kern-Subdomänen selten
- Kein geschäftlicher Wert durch Weiterentwicklung von unterstützenden Subdomains, da sie keinen Wettbewerbsvorteil bieten

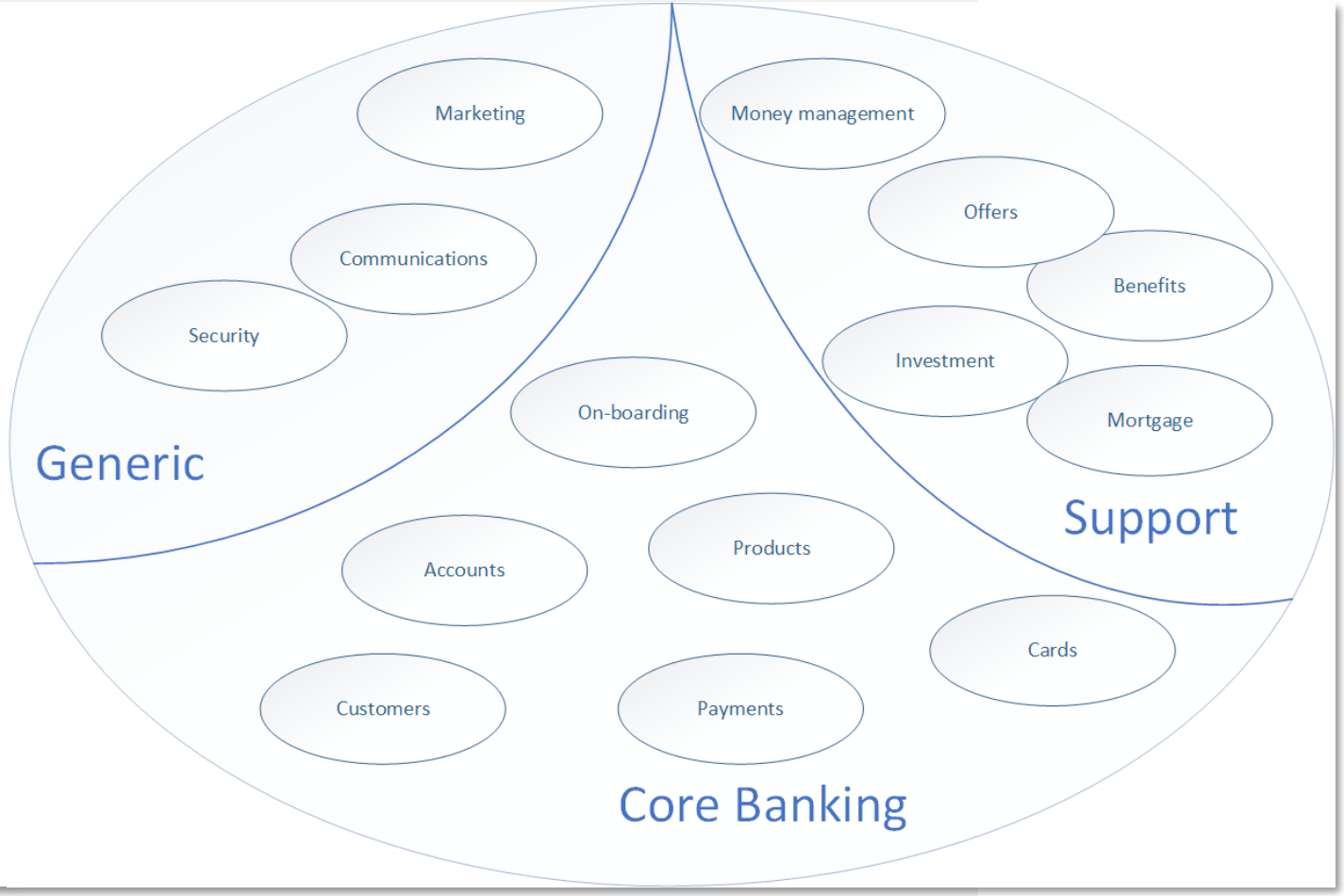
Implementierungsüberlegungen

- Mangels Wettbewerbsvorteil sollten unterstützende Subdomains nicht selbst implementiert werden
- Es gibt jedoch häufig keine vorgefertigten Lösungen für unterstützende Subdomains
- Daher müssen unterstützende Subdomains dennoch oft selbst implementiert werden
- Einfache Anwendungsentwicklungs-Framework sind dafür oft ausreichend
- Die Einfachheit der Geschäftslogik macht Supporting Subdomains zu einem naheliegenden **Kandidaten für das Outsourcing**

STRATEGISCHES DESIGN

Was ist eine Generic Subdomain?

- Generische Subdomains sind ebenfalls durch Komplexität gekennzeichnet
- Sie bieten jedoch einem Unternehmen **keinen Wettbewerbsvorteil**, da es bereits bewährte und weit verbreitete Implementierungen gibt
- Generische Subdomains sind Geschäftsaktivitäten, die alle Unternehmen auf die gleiche Weise durchführen
- Man versucht diese daher üblicherweise mit Standardsoftware und -systemen abzudecken
- Beispiele wären die Identitätsverwaltung mit Microsofts Active Directory oder Oracles Datenbankprodukte, oder die Produkte von SAPs wie R/3 oder S/4 HANA



STRATEGISCHES DESIGN

Kategorien von Domänen

	Core Subdomain	Supporting Subdomain	Generic Subdomain
Alleinstellungsmerkmal / Geschäftskritisch	ja	nein	nein
Unternehmensspezifisch	ja	ja	nein
Technische Komplexität (Einstiegshürde)	hoch	niedrig	mittel/ hoch
Technische Rate of Change	hoch	niedrig	niedrig
Existieren Lösungen / Produkte / Services	nein	teilweise	ja
Entwicklung In-House sinnvoll	ja	mögl.	nein
Out-Sourcing der Entwicklung ratsam	nein	mögl.	nein
Existieren COTS Produkte / Managed Services?	nein	teilweise	ja

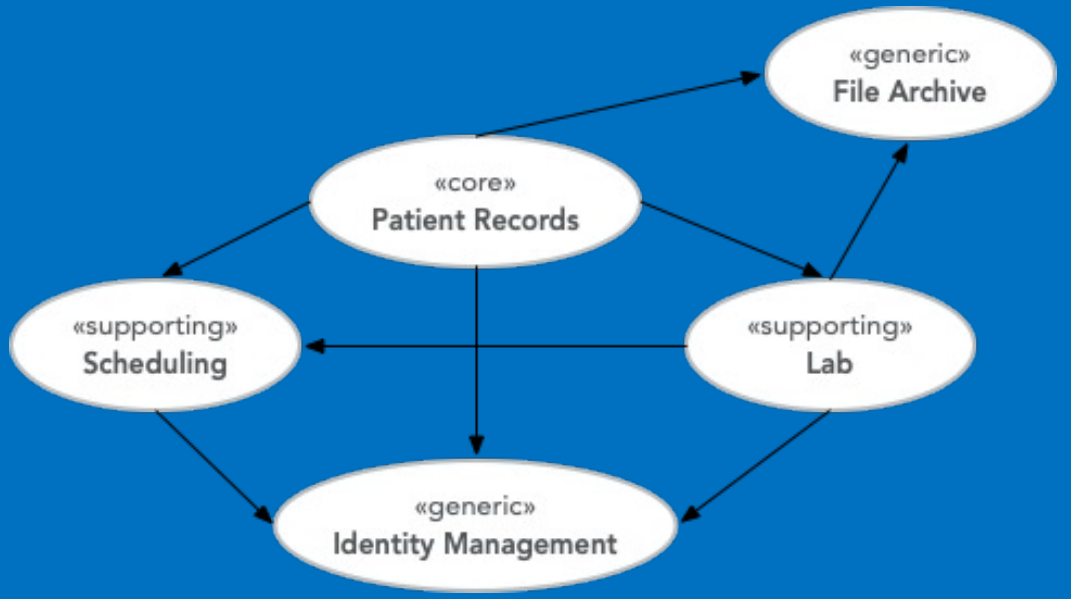
BEISPIEL

EMR-System (Electronic Medical Records)

Nehmen wir an, wir bauen ein EMR-System (Electronic Medical Records) für kleinere Kliniken für das die folgenden Subdomänen identifiziert worden sind:

- **Patient Records** für die Verwaltung der Patientenakten (persönliche Informationen, Anamnese, etc.).
- **Lab** für die Bestellung von Labortests und die Verwaltung der Testergebnisse.
- **Scheduling** für die Planung von Terminen.
- **File Archive** für die Speicherung und Verwaltung von Dateien, die den Patientenakten beigefügt sind (z. B. verschiedene Dokumente, Röntgenbilder, gescannte Papierdokumente).
- **Identitätsmanagement**, um sicherzustellen, dass die richtigen Personen Zugriff auf die richtigen Informationen haben.

Wie würden wir nun diese Subdomänen klassifizieren? Die offensichtlichsten sind das Dateiarchiv und das Identitätsmanagement, die eindeutig generische Subdomänen sind. Aber was ist mit den anderen? Das hängt davon ab, was dieses bestimmte EMR-System von den anderen auf dem Markt abheben soll.

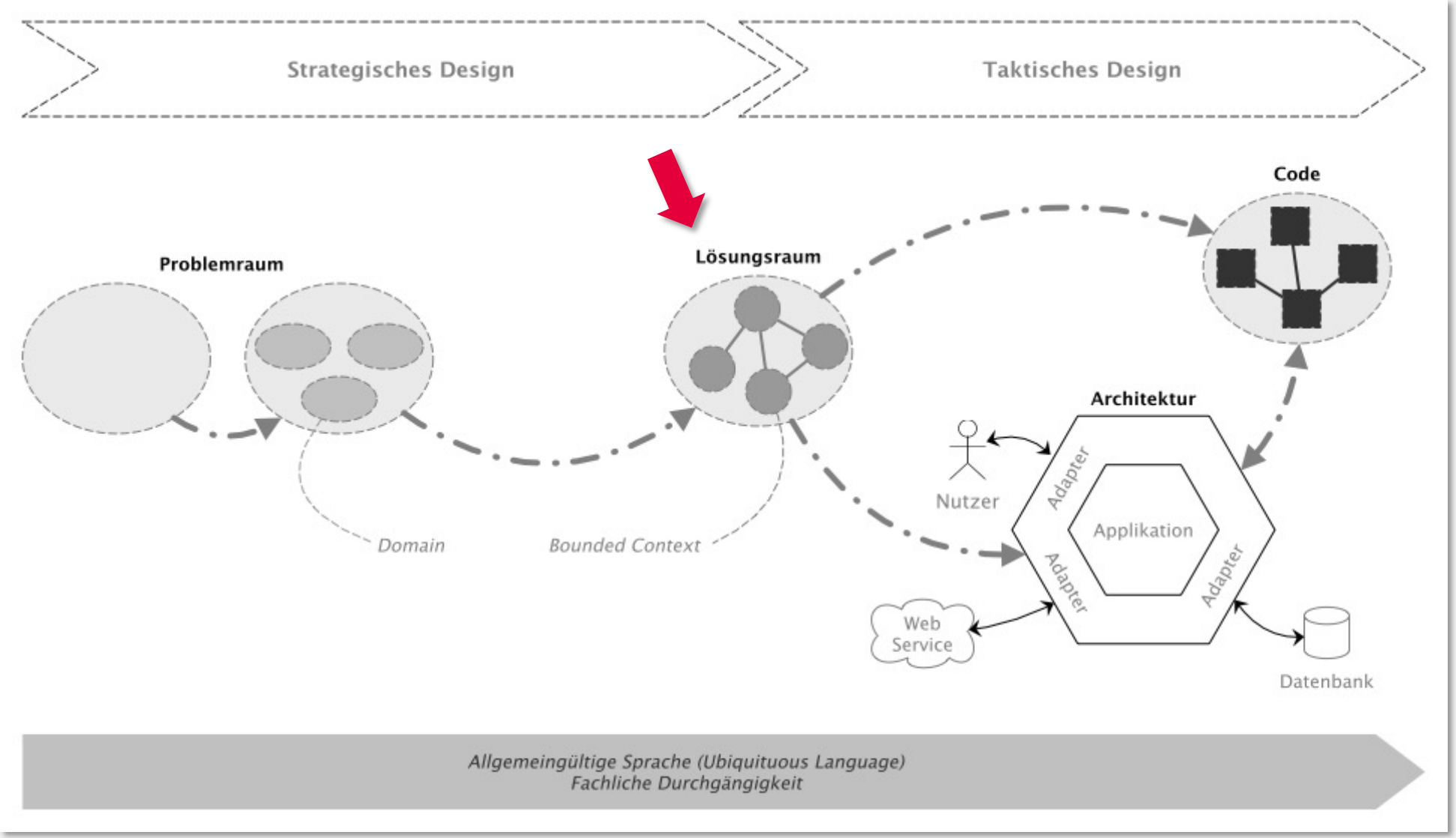


Bei einem EMR-System, kann man ziemlich sicher davon ausgehen, dass Patient Records eine Kerndomäne ist. Wenn allerdings ein EMR-System ebenfalls zum Ziel hat, Kliniken durch clevere und innovative Terminplanung effizienter zu machen, dann ist Scheduling ggf. auch eine Kerndomäne. Andernfalls ist es eine unterstützende Subdomäne, die durch eine bestehende Terminplanungs-Engine gut abdeckbar ist. Ähnlich bei der Subdomäne Labor: Sollte ein wesentlicher Teil unseres Business Case eine nahtlose Integration zwischen Patientenakten und Labor sein, dann ist das Labor höchstwahrscheinlich eine Kerndomäne. Andernfalls ist es eine unterstützende Subdomäne.

Bildquelle: Vladik Khononov, *What Is Domain-Driven Design?* O'Reilly Media, 2019

DOMAIN DRIVEN DESIGN

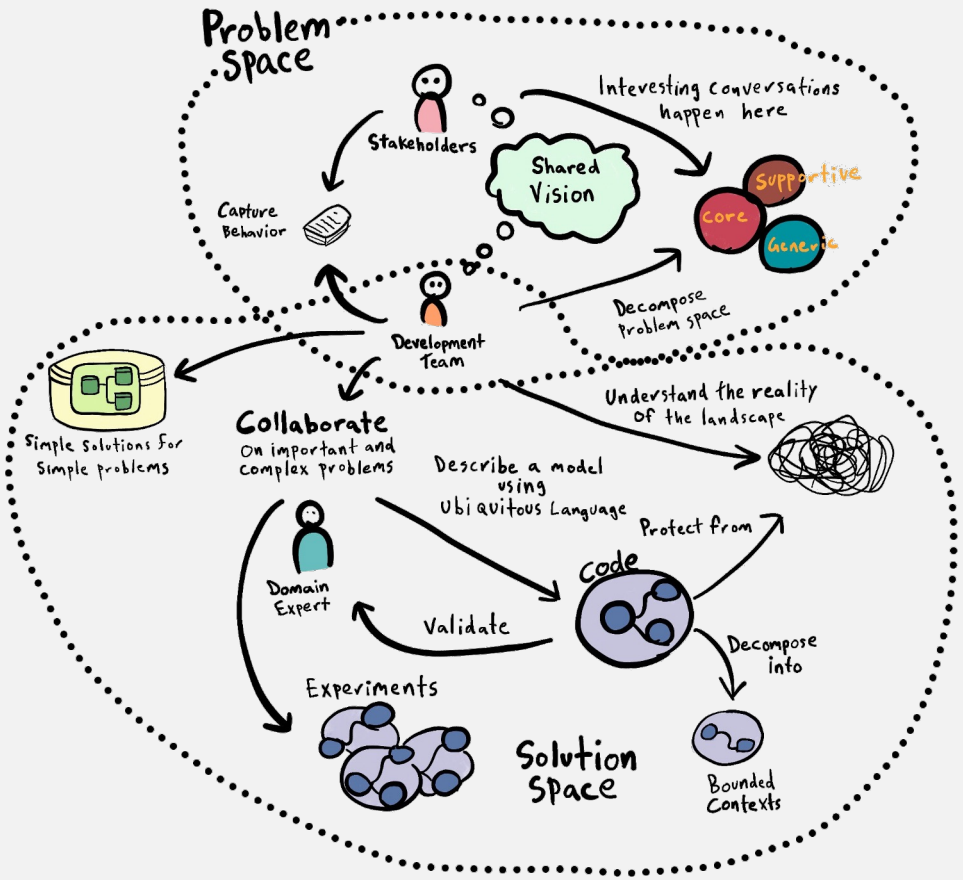
Fachlichkeit als Treiber der Softwareentwicklung



VOM PROBLEMRAUM ZUM LÖSUNGSRAUM

Ubiquitous Language

- Eine Domäne wird manchmal auch als "Problemraum" bezeichnet, da sie fachliche Probleme definiert, die eine Software lösen soll.
- Man unterscheidet daher auch zwischen Problemraum und Lösungsraum.
- Der **Problemraum** konzentriert sich auf **fachliche Probleme**
- Der **Lösungsraum** konzentriert sich auf **technische Lösungen**
- Die Subdomains gehören in den Problemraum.
- Die Beziehung zwischen dem fachlichen Problemraum und dem technischen Lösungsraum ist allerdings schwierig zu erhalten.



übliche SW-Entwicklungs-methodiken basieren meist implizit auf den folgenden übersetzungen:

- *Domänenwissen in Analysemodell*
- *Analysemodell in Anforderungen*
- *Anforderungen in Systementwurf*
- *Systementwurf in Quellcode*

Bei jedem dieser übergänge können übersetzungsfehler passieren.

KODIFIZIERUNG VON DOMÄNENWISSEN

CLOUD NATIVE COMPUTING

Ubiquitous Language

- Softwareentwicklungsmodelle beinhalten oft Tätigkeiten, bei denen **Domänenwissen in eine „ingenieursfreundliche" Form übersetzt** wird
- Solche Artefakte (z.B. in UML-Klassendiagramme) werden als Analysemodelle bezeichnet und beschreiben die Anforderungen an das System
- Analysemodelle dienen weniger dem Verständnis der Geschäftsdomäne als der Strukturierung der Lösung
- Solch ein Wissensaustausch zwischen Fachdomäne und Entwicklung ist gefährlich, da Informationen verloren gehen können
- Domänenwissen geht auf dem Weg zu Softwareingenieuren verloren, was für die Lösung von Geschäftsproblemen problematisch sein kann

Seit der Softwarekrise in den 1960ern haben viele Untersuchungen das Scheitern von Software-Projekten untersucht. Fast alle haben gezeigt, dass u.a. Kommunikation für den Projekterfolg unerlässlich ist.

Fast alle Softwareentwicklungsmodelle versuchen daher Domänenwissen von Domänenexperten an die Ingenieure weiterzugeben. Meist geschieht dies durch Vermittlerrollen, die Wissen von der Fachdomäne an die Entwicklung „durchreichen" und übersetzen: System-/Business-Analysten, Product Owner, Projektmanager, usw..

Das impliziert aber nicht automatisch eine **effektive**, d.h. auf Verständnis beruhende, Kommunikation.

Domain-driven Design versucht vor diesem Hintergrund das Wissen von Domänenexperten zu Softwareingenieuren mittels einer allgegenwärtigen – und gemeinsam entwickelten – Sprache zu bringen, der sogenannten Ubiquitous Language.

STRATEGISCHES DESIGN

Konzepte im Problemraum - Ubiquitous Language

CLOUD NATIVE COMPUTING

Gemeinsame Sprache ist Schlüssel zum gemeinsamen Verständnis

- Aufwändiger aber notwendiger Schritt
- Klärung der fachlichen Begriffe
- Klarheit der gemeinsamen Sprache ist Indikator für die Tiefe des Verständnisses des Problemraums.
- Unklare Begriffe deuten auf tieferliegende und verdeckte (ggf. unverstandene) Konzepte hin.
- Hierzu müssen implizite Annahmen explizit gemacht werden um eine eindeutige Bedeutung innerhalb der Ubiquitous Language festlegen zu können.

Ubiquitär = „allumfassend, überall vorhanden, allgegenwärtig"

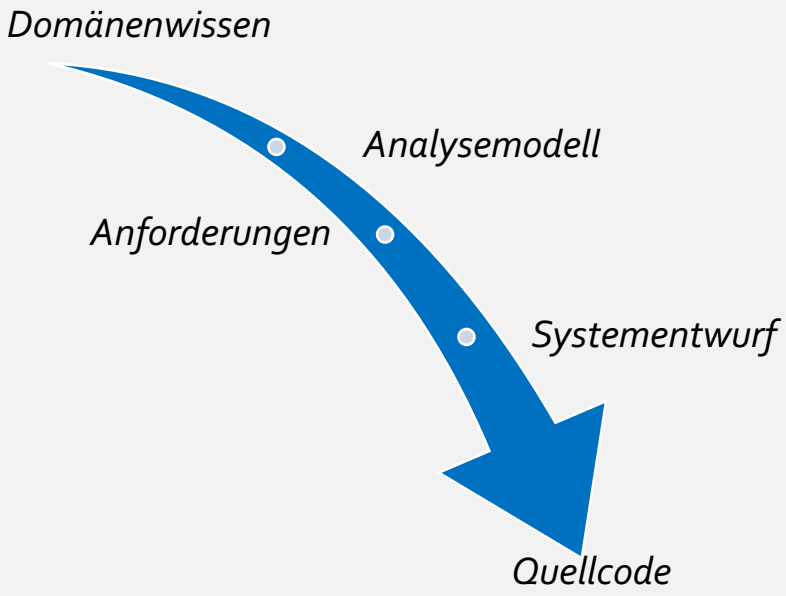
- Jeder im Projekt muss dieselbe Sprache sprechen und verstehen können
- Alle relevanten Konzepte müssen in der Sprache beschrieben werden können
- Die Sprache wird evolutionär weiterentwickelt und ist in allen Phasen der Entwicklung präsent (auch im Code!)

Eines der wichtigsten Konzepte in DDD

- Grundlage für gemeinsames Verständnis
- Grundlage für durchgängiges Verständnis
- Indikator für Durchdringung und Beschreibungsstabilität der Domäne

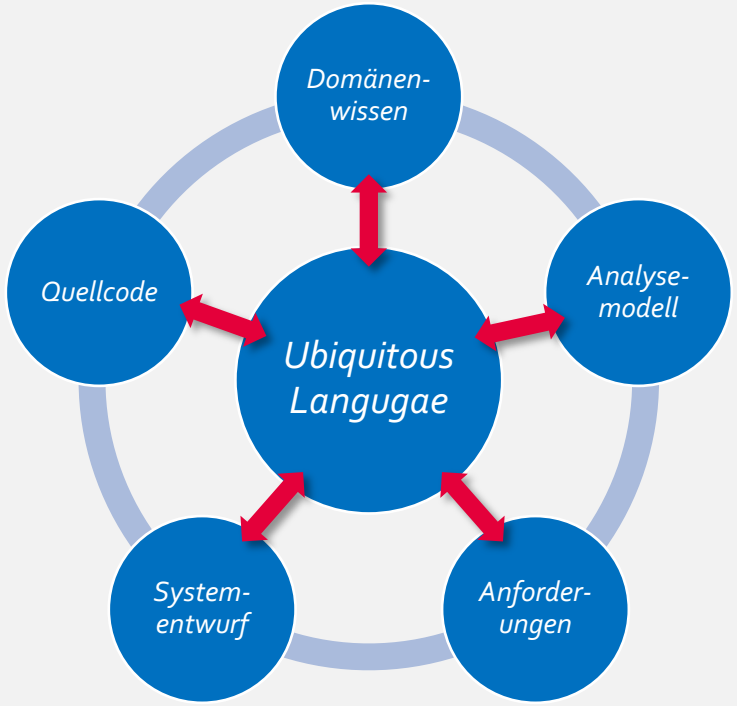
UBIQUITOUS LANGUAGE

Non - DDD



Anstatt das Domänenwissen ständig zu übersetzen, fordert DDD die Kultivierung einer einzigen Sprache zur Beschreibung der Geschäftsdomäne: die allgegenwärtige Sprache.

DDD



Durch die durchgängige Verwendung der allgegenwärtigen Sprache und ihrer Begriffe kann ein gemeinsames Verständnis zwischen allen Projektbeteiligten kultiviert werden. Die allgegenwärtige Sprache sollte daher nur aus Begriffen bestehen, die sich auf die Geschäftsdomäne beziehen.

CLOUD NATIVE COMPUTING

Ziel ist es nicht, Domänenexperten etwas über SW-Entwicklung (z.B. Singletons und abstrakte Fabriken) beizubringen.

Der Zweck der allgegenwärtigen Sprache ist es, das Verständnis und die mentalen Modelle der Geschäftsdomäne der Domänenexperten in leicht verständliche Begriffe für die Entwicklung zu fassen.

UBIQUITOUS LANGUAGE

Mehrdeutige und synonyme Begriffe

CLOUD NATIVE COMPUTING

Mehrdeutige Begriffe

Der englische Begriff "Policy" hat mehrere Bedeutungen: Er kann eine gesetzliche Vorschrift oder einen Versicherungsvertrag bedeuten. Die genaue Bedeutung ergibt sich meist aus dem Kontext. Problematisch sind Kontexte in denen bspw. Versicherungsverträge aufgrund gesetzlicher Vorschriften erforderlich sind. Welches Konzept meint Policy nun? Mit Versicherungen haben wir eine Domäne in der beide Konzepte zeitgleich vorkommen.

Ubiquitäre Sprache verlangt für solche Fälle eine einzige Bedeutung für jeden Begriff, und daher sollte "policy" explizit mit den beiden Begriffen "Regulierungsvorschrift" oder "Versicherungsvertrag" modelliert werden (aber nie Policy genannt werden).

Synonyme Begriffe

Im DDD können zwei Begriffe nicht austauschbar verwendet werden. Zum Beispiel verwenden viele Systeme den Begriff "Benutzer". Der Begriff "Benutzer" wird im normalen Sprachgebrauch jedoch häufig austauschbar verwendet: z. B. "Benutzer", "Besucher", "Administrator", "Konto" usw.

Synonyme Begriffe bezeichnen meist unterschiedliche Feinkonzepte. Bspw. beziehen sich die Begriffe "Besucher" und "Konto" technisch gesehen auf die Benutzer des Systems; in den meisten Systemen stellen jedoch nicht registrierte und registrierte Benutzer unterschiedliche Rollen dar und haben unterschiedliche Verhaltensweisen. Daten von "Besuchern" werden meist zu Analysezwecken verwendet, während "Konten" das System und seine Funktionen tatsächlich nutzen.

Es ist vorzuziehen, jeden Begriff explizit und in seinem spezifischen Kontext zu verwenden. Ein Verständnis der Unterschiede zwischen den verwendeten Begriffen ermöglicht die Erstellung einfacherer und klarerer Modelle und Implementierungen der Entitäten der Geschäftsdomäne.

UBIQUITOUS LANGUAGE

So

Angenommen, wir arbeiten an einem System zur Verwaltung von Online Werbekampagnen. Folgende Aussagen sind in der Sprache von Marketing-Domänenexperten formuliert.

- Eine Werbekampagne kann verschiedene kreative Formate (Videos, Banner, Animationen) anzeigen.
- Eine Kampagne kann nur veröffentlicht werden, wenn mindestens eine ihrer Formate aktiv ist.
- Verkaufsprovisionen werden für genehmigte Transaktionen abgerechnet.

So nicht

Die folgenden Aussagen sind rein technischer Natur und werden für Domänenexperten vermutlich unklar sein. Sie gehören nicht in eine Ubqiquitous Language.

- Der Anzeigen-iframe zeigt eine HTML-Datei an, die mittels einer URL verlinkten Content einbettet.
- Eine Kampagne kann nur veröffentlicht werden, wenn sie mindestens einen zugehörigen Datensatz in der Tabelle active-placements hat.
- Die Verkaufsprovisionen basieren auf korrelierten Datensätzen aus den Tabellen transactions und approved sales.

CLOUD NATIVE COMPUTING

DDD == Fachlichkeit, Fachlichkeit, Fachlichkeit

AUSBLICK

Domain Driven Design

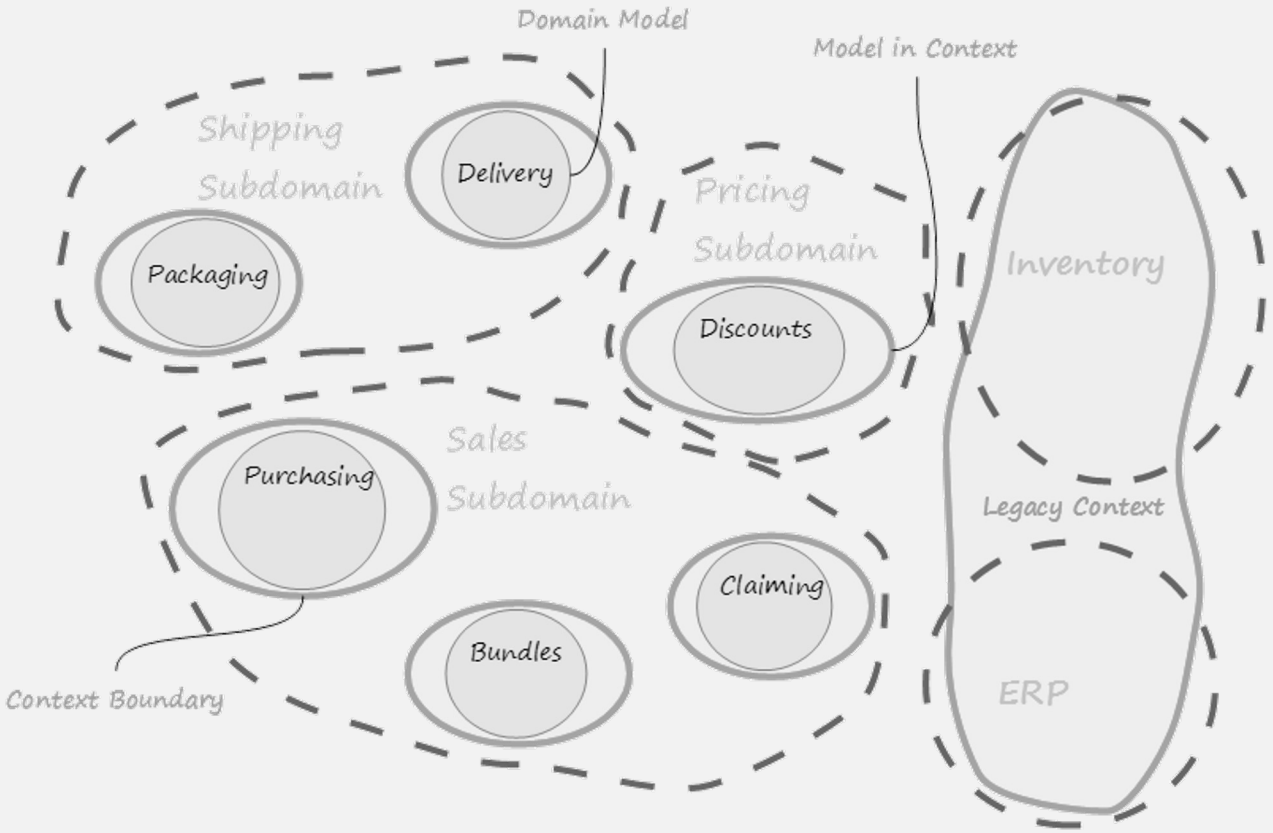
- Was ist das?
- Effektives Software Design
- Strategisches Design
- Taktisches Design

Strategisches Design

- Subdomains
- Ubiquitous Language
- **Bounded Context**
- **Context Mapping**

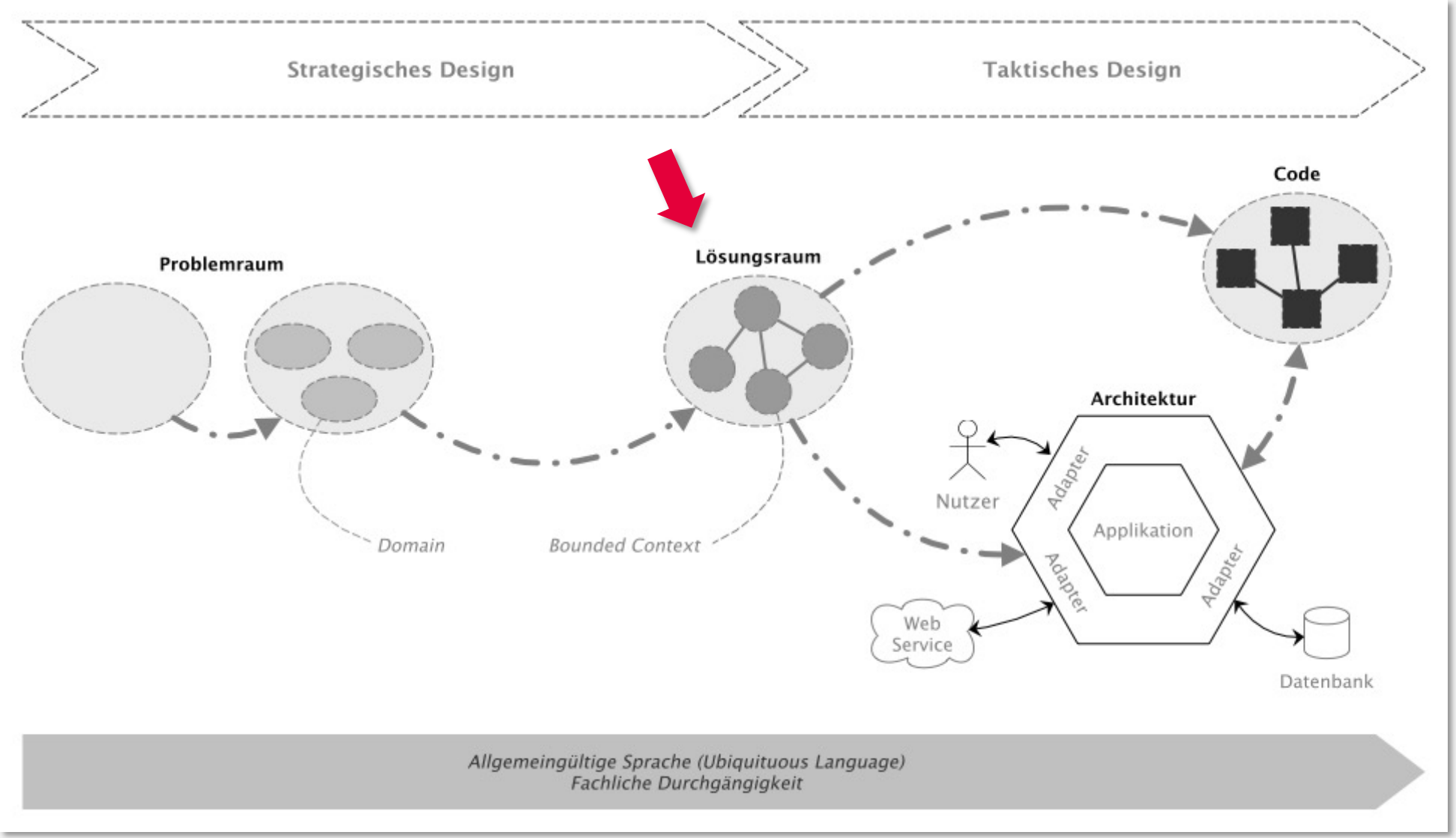
Taktisches Design

- Business Logic Pattern
- Architectural Pattern



DOMAIN DRIVEN DESIGN

Fachlichkeit als Treiber der Softwareentwicklung



KONTAKT

Disclaimer

Nane Kratzke

+49 451 300-5549

nane.kratzke@th-luebeck.de

kratzke.mylab.th-luebeck.de