WAS IST DIESES DDD UND WOBEI KANN ES MIR HELFEN?

JUG Saxony Vortrag am 18.04.2024
von Ferdinand Ade und Lennart Golubski

Das sind wir

Ferdinand Ade

Software Engineer & Consultant

Lennart Golubski

IT Consultant & Developer

an was denkt ihr, wenn ihr “Domain Driven Design” hört?

an was denkt ihr, wenn ihr “Domain Driven Design” hört?
https://www.menti.com/alyz5wpa8soa

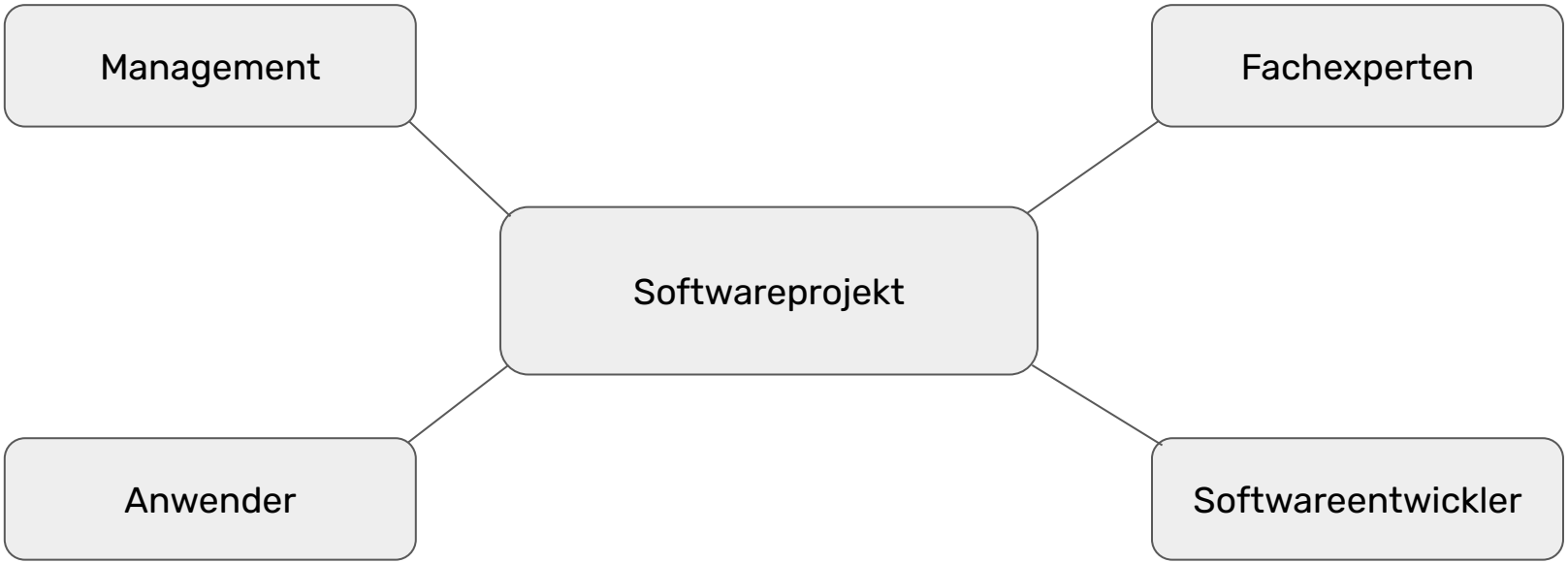
In welcher Rolle seid ihr unterwegs?

Storytime: Wie ist Ferdi zu DDD gekommen?

was ist hier los?

```
fun limit(x: Int, y: Int, z: Int, type: String): Boolean {
    if (type == "CREDIT") {
        return x - z > 0
    } else {
        return y - z > 0
    }
}
```

Stakeholder in Softwareprojekten



“It’s developers’ (mis)understanding, not expert knowledge that gets released in production.”

Alberto Brandolini

Was meint ihr?

"You ship what's in your developers' heads"

Andrea Magnorsky

"You ship what's in your code, not what's in your developers' heads"

Tudor Girba

Kontext, Kontext, Kontext

was ist eine “Karte”?

Ein Begriff - verschiedene Bedeutungen

Karte

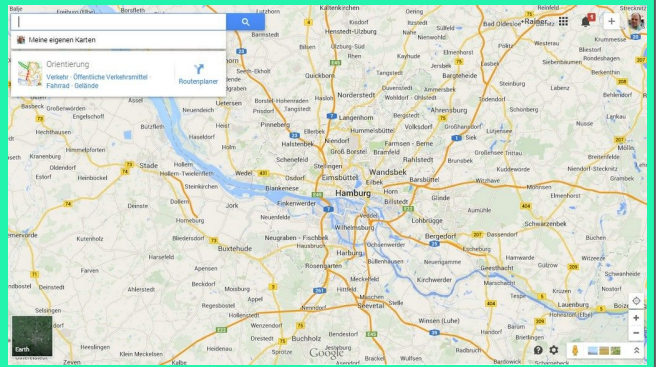
Bereich:
Fussball

Karte

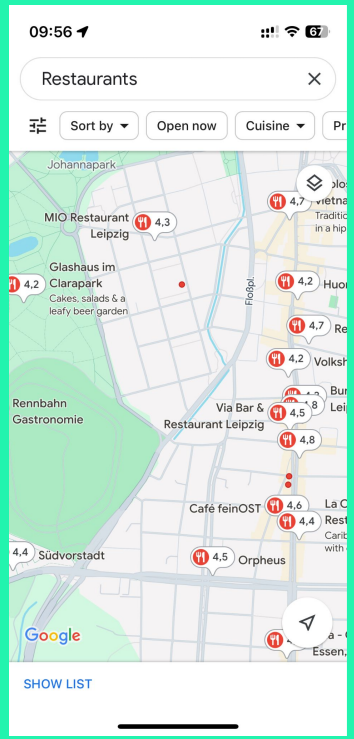
Bereich: Gespräch zwischen Geschäftspartnern

Karte

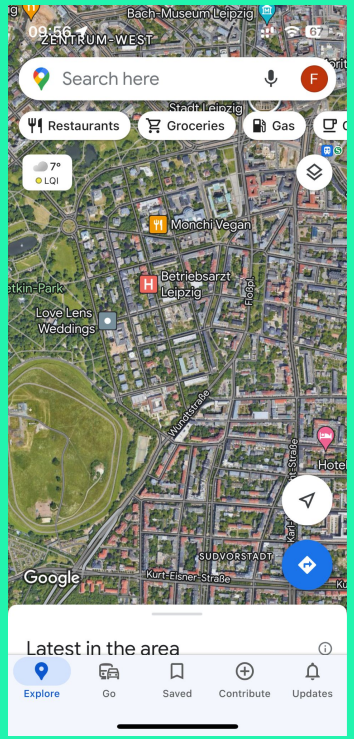
Bereich:
Google Maps



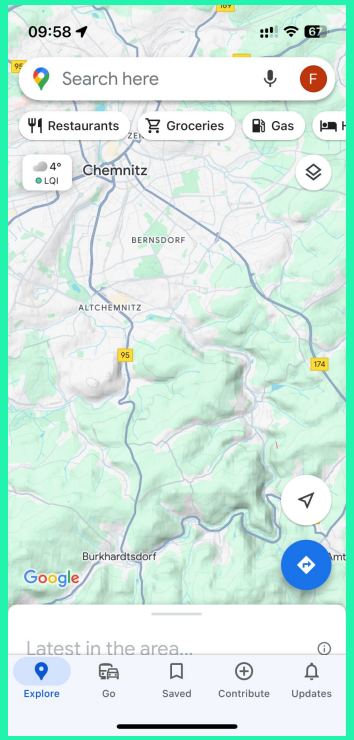
Eine Bedeutung - verschiedene Anwendungen und Detailgrade



Bereich: Hunger



Bereich: Spazier-gang



Bereich: Radtour

Welche Kartenansicht ist die “echte”? Welche ist die beste?

“All models are wrong, but some are useful”

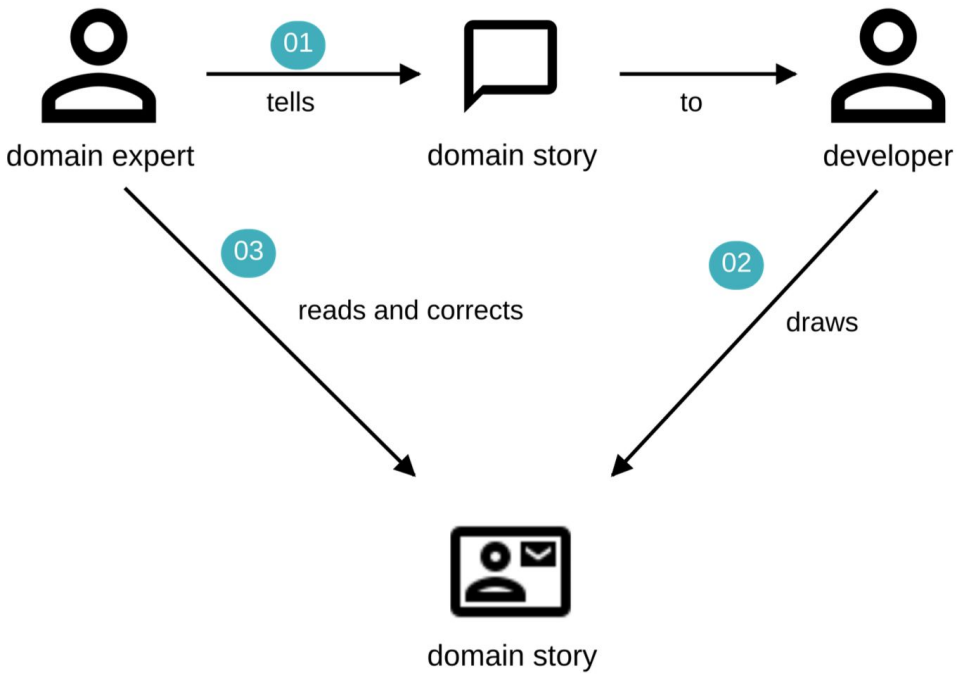
George E. P. Box

“A model is a simplification. It is an interpretation of reality that abstracts the aspects relevant to solving the problem at hand and ignores extraneous detail.”

Eric Evans

Knowledge Crunching mit “Domain Storytelling”

Domain Storytelling



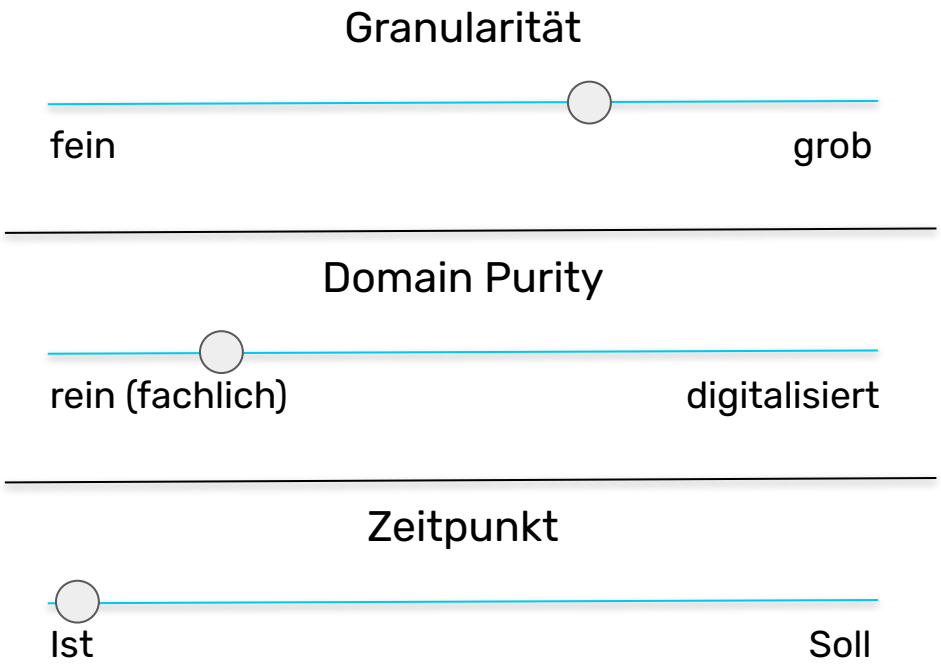
Kollaborative Modellierungstechnik

Geschichten erzählen und visualisieren

Transportiert Wissen von Domänenexperten zu allen, die an der Entwicklung von Software beteiligt sind

Eine Geschichte ist ein Pfad / Schreibe mehrere Geschichten für mehrere Pfade

Domain Storytelling - Scope Faktoren



Scope einer Domain Story setzt sich zusammen aus den drei Faktoren

die jeweiligen Ausprägungen sollten vorher für eine Story festgelegt werden

innerhalb einer Story sollten die Ausprägungen eingehalten werden

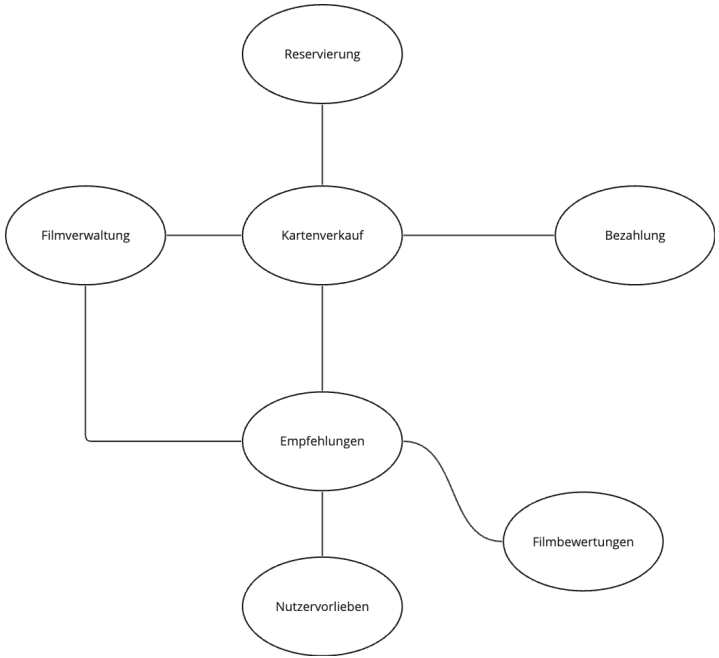
Wrap-up Domain Storytelling

Und wie bekommen wir das ganze in den Code?

Bounded Contexts

Bounded Contexts finden

Context Map

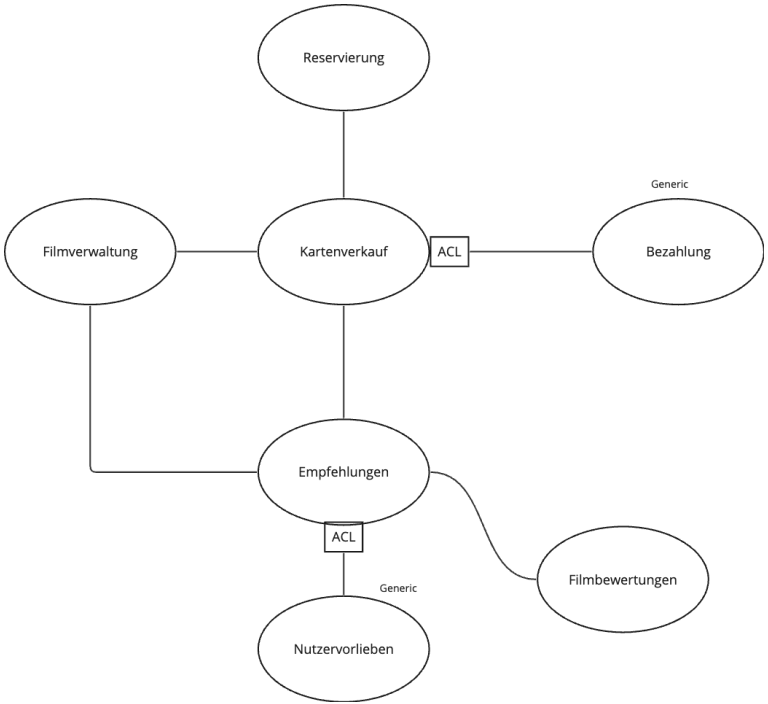

Reservierung
Filmverwaltung
Kartenverkauf
Bezahlung
Empfehlungen
Filmbewertungen
Nutzervorlieben

Subdomains

Subdomains

Generic

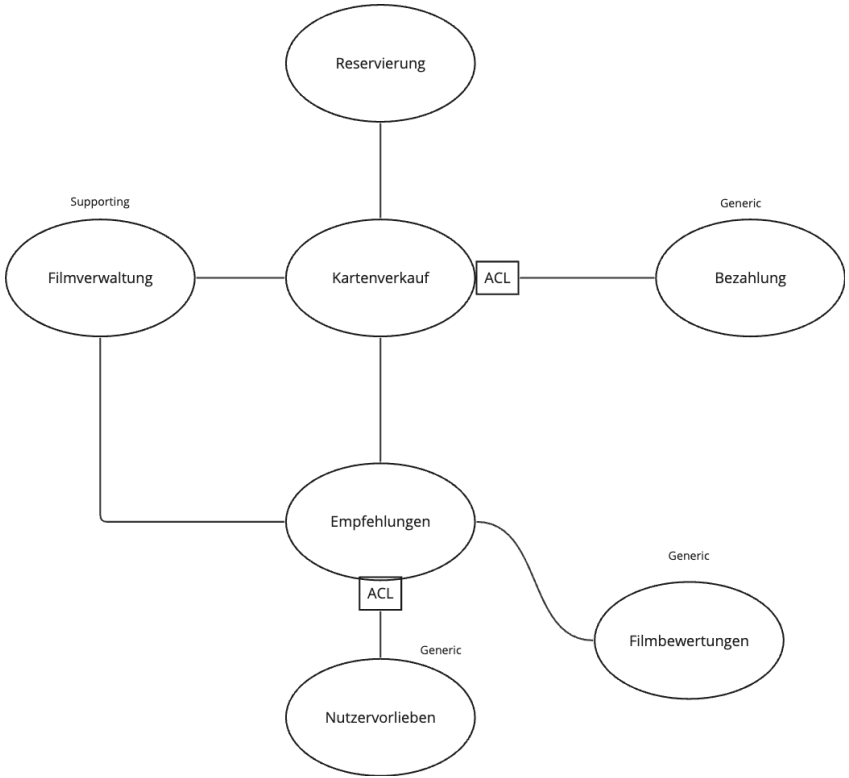
Context Map


Reservierung
Generic
Filmverwaltung
Kartenverkauf
ACL
Bezahlung
Empfehlungen
ACL
Filmbewertungen
Generic
Nutzervorlieben

Subdomains

Supporting

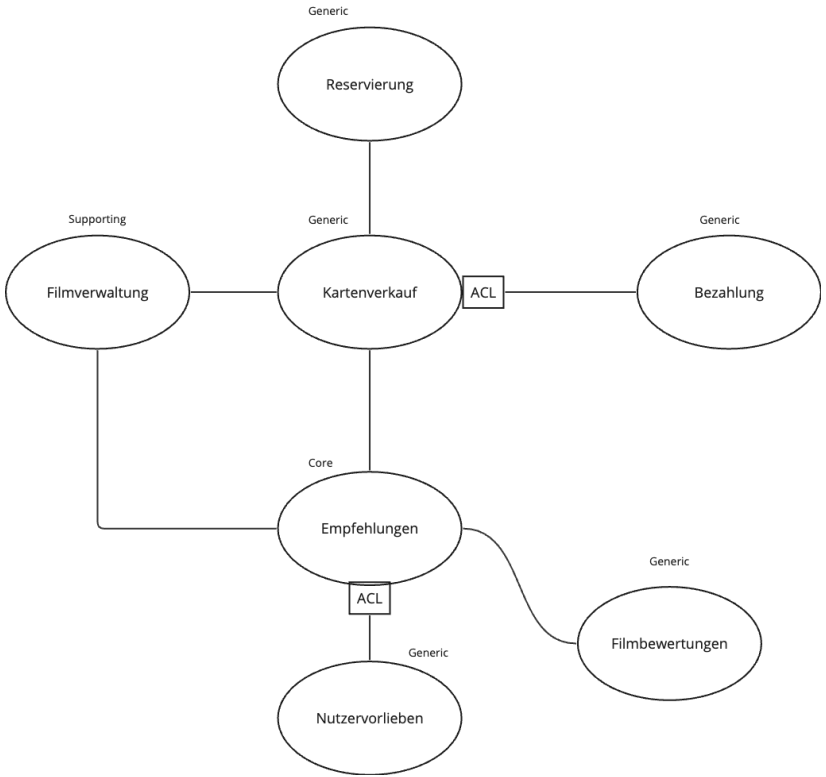
Context Map


Reservierung
Supporting
Filmverwaltung
Kartenverkauf
ACL
Generic
Bezahlung
Empfehlungen
ACL
Generic
Nutzervorlieben
Generic
Filmbewertungen

Subdomains

Core

Context Map

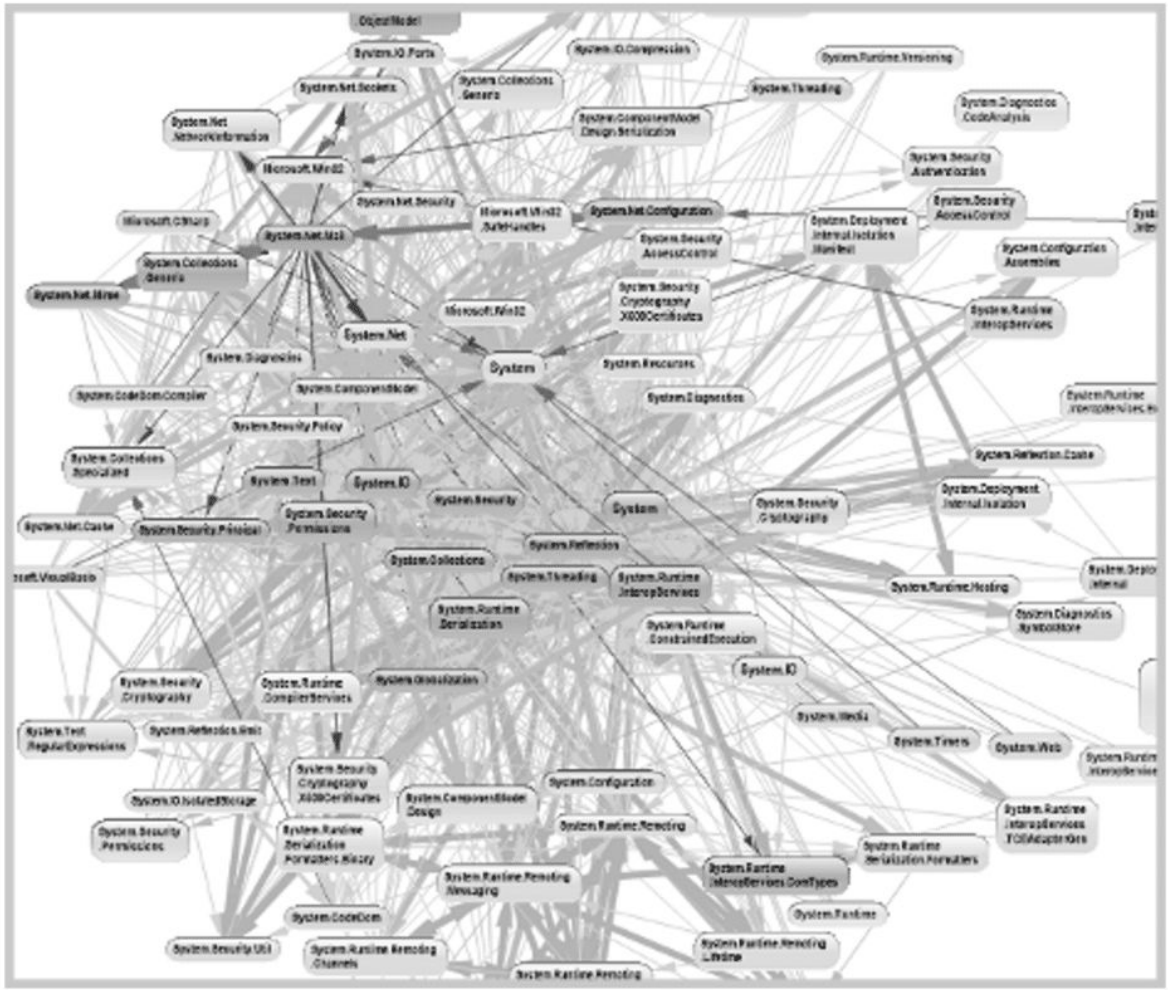

Generic
Reservierung
Supporting
Filmverwaltung
Generic
Kartenverkauf
ACL
Generic
Bezahlung
Core
Empfehlungen
ACL
Generic
Filmbewertungen
Generic
Nutzervorlieben

Core, Supporting, Generic-

Was bedeutet das für uns Techies?

“Big Ball of Mud”

Brian Foote und Joseph Yoder (1999)


System.IO.Ports
System.Net.Sockets
System.Collections.Generic
System.Net.NetworkInformation
Microsoft.Win32
System.Net.Security
Microsoft.Win32.SafeHandles
System.Net.Configuration
System.Net.Mail
System.Collections.Generic
System.Net.Mime
Microsoft.Win32
System.Net
System.Diagnostics
System
System.Resources
System.CodeDom.Compiler
System.ComponentModel
System.Diagnostics
System.Security.Policy
System.Collections.ObjectModel
System.Text
System.IO
System.Security
System
System.Net.Cache
System.Security.Principal
System.Security.Permissions
System.Reflection
System.Collections
System.Threading
System.Runtime.InteropServices
System.Runtime.Hosting
System.Runtime.Serialization
System.Runtime.ConstrainedExecution
System.Security.Cryptography
System.Runtime.CompilerServices
System.Reflection.Emit
System.IO
System.Media
System.Timers
System.Web
System.Text.RegularExpressions
System.Security.Cryptography.X509Certificates
System.ComponentModel.Design
System.Configuration
System.IO.IsolatedStorage
System.Runtime.Remoting
System.Security.Permissions
System.Runtime.Serialization.Formatters.Binary
System.Runtime.InteropServices.ComTypes
System.Runtime.Remoting.Messaging
System.Runtime
System.CodeDom
System.Security.Util
System.Runtime.Remoting.Channels
System.Runtime.Remoting.Lifetime
System.Runtime.Remoting

Ab in den Code…

lokale vs globale Optimierung

kurzfristig vs langfristig

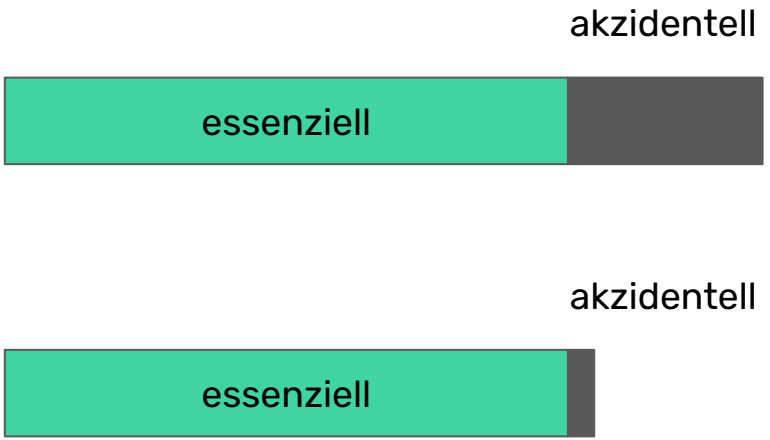
Essenzielle vs. akzidentelle Komplexität

essenziell:

- **die Mindest-Komplexität**
- **“liegt im Wesen der Sache”**
- **niemals auflösbar**

akzidentell:

- **nicht “notwendig”**
- **kann viele Gründe haben**



-> Gesamtkomplexität ist geringer

Gründe für akzidentelle Komplexität

Missverständnisse bei der Analyse der Fachdomäne	**Sachen werden gebaut, die keiner braucht**
schlechtes Design oder schlechte Architektur	**unpassende oder veraltete Technologie**

DDD hilft dabei, akzidentelle Komplexität zu verhindern

Auf was kommts also an bei DDD?

Kommunikation, Modellgrenzen, Komplexität

Auf was kommts also an bei DDD?

ihr müsst nicht immer alle Tools einsetzen, um von den Ideen in DDD zu profiti

Was fehlt noch?

ein Rätsel

```
fun limit(x: Int, y: Int, z: Int, type: String): Boolean {
    if (type == "CREDIT") {
        return x - z > 0
    } else {
        return y - z > 0
    }
}
```

die Auflösung

```
fun limit(kreditRahmen: Int, guthaben: Int, transaktion: Int, cardType: CardType): Boolean {
    return when (cardType) {
        CardType.CREDIT -> kreditRahmen - transaktion > 0
        CardType.DEBIT -> guthaben - transaktion > 0
    }
}

enum class CardType {
    CREDIT, DEBIT
}
```

Quellen & Einflüsse

- Talks:
  - Bounded Contexts - Eric Evans - DDD Europe 2020 - https://www.youtube.com/watch?v=am-HXycfalo
  - https://virtualddd.com/sessions/managing-domain-knowledge-with-chris-simon/
- Blogs & Artikel:
  - https://www.domainlanguage.com/wp-content/uploads/2016/05/DDD_Reference_2015-03.pdf
  - http://www.laputan.org/mud/mud.html
  - https://medium.com/nick-tune-tech-strategy-blog/domains-subdomain-problem-solution-space-in-ddd-clearly-defined-e0b49c7b586c
  - https://github.com/ddd-crew/welcome-to-ddd
- Bücher:
  - Learning Domain-Driven Design; Vlad Khononov; 2021
  - Domain Storytelling; Hofer, Schwentner; 2020
  - Domain Driven Transformation; Lilienthal, Schwentner; 2023
- Bilder:
  - https://www.dfb.de/fileadmin/_dfbdam/103359-csm_69029-imago19186247h_15a0cb748a.jpg
  - https://www.prinux.com/wp-content/uploads/2015/12/2Schriftarten.jpg
  - https://i.computer-bild.de/imgs/5/0/3/5/0/1/1/Google-Maps-neu-1024x576-a64c65951bf168b3.jpg
  - https://developeronfire.com/assets/images/AlbertoBrandolini.jpg
  - https://codeopinion.com/wp-content/uploads/2021/12/2-1.png


codecentric
Creating the digital future together.
Junior Developer und Consultant (w/d/m)
Berlin, Bielefeld, Erfurt, Frankfurt am Main, Hamburg, Karlsruhe, Leipzig, Münster, Nürnberg & Stuttgart

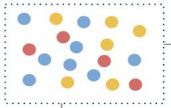
Fragen

Feedback
https://www.menti.com/alyz5wpa8soa

DDD Onepager

Domain Driven Design On A Page

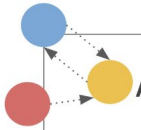
DDD is a philosophy for developing software systems that encourages
Domain Thinking at each step of the Software Development Lifecycle



Domain Discovery

Exploratory DDD

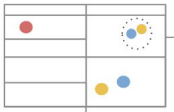
- Model as-is, to-be, and could-be states of the domain
- Model collaboratively and visually so that the whole team* learns the domain and contributes to solution ideas
- Example: Big Picture EventStorming



Software Architecture

Strategic DDD

- Bounded Contexts: Split a large software system into specialised models aligned to areas of the domain
- Integrate bounded context using domain events**
- Identify strategically significant core domains



Software Design

Tactical DDD

- Create models in code which align to the team's shared understanding of the domain
- Use appropriate patterns in each context: entities aggregates, event sourcing, etc

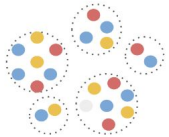
* whole team: all roles involved in product development including business experts
** domain events: business-relevant happening communicated via (technical) events or commands

github.com/ddd-crew/

DDD Mindset

DDD Doctrine

Principles and practices that are almost universally applicable in DDD



Make the implicit Explicit

Avoid ambiguity and improve communication by making all details visible.

Cultivate a shared language

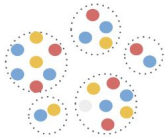
Create a common language within each bounded context to foster improved collaboration

Learning never stops

There is always more to learn about the domain so a learning mindset is essential

Explore multiple models

When the benefits of a better model are significant, don't stop at the first idea



Focus on concrete Scenarios

Avoid creating beautiful designs that fail to solve the problem by challenging designs with concrete scenarios

Design is Evolutionary

Because learning never stops there is a constant stream of feedback which can be used to improve the design

github.com/ddd-crew/