Domänen Driven Design

Exkurs – Vorlesung Services Engineering

Inhalt

- Motivation und Ziele
- Aspekte zum Domain Driven Design (kurz DDD)
- Vertiefung mit Hilfe einer Übung
  - DDD und Web-APIs
  - DDD und Microservices
  - Bezüge zu existenten Domänenmodellen – (hier TMF)
- Weiterführende Quellen

Motivation und Ziele

Typische Probleme

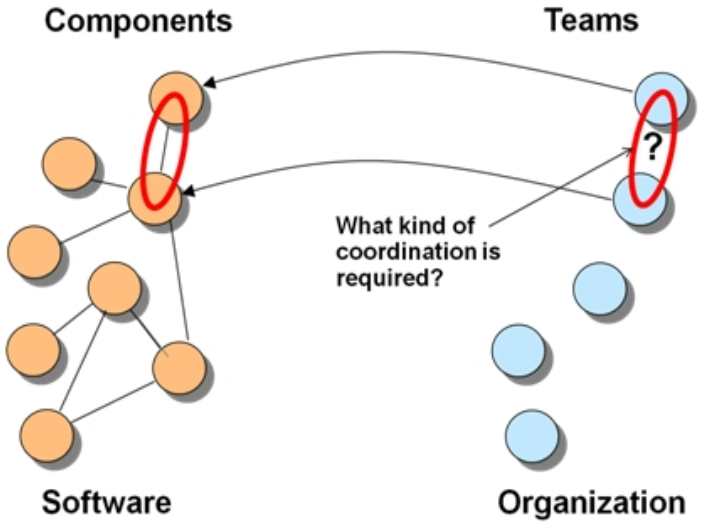
- Technologiebessenheit der Entwickler …
- Zu starker Fokus auf die Datenbank …
- Namen für Objekte und Operationen korrespondierenden nicht mit fachlichen Aufgaben
- Monolithische Architekturansätze
- Fachlogik gehört nicht in die Benutzerschnittstelle bzw. in die Datenhaltung
- Services sind untereinander stark gekoppelt

In Anlehnung an: Vernon, V.: Domain-Driven Design kompakt, dpunkt.verlag, 2017 (Übers. Lilienthal, C.; Schwentner, H.)
Quelle der Abbildung: https://www.dpunkt.de/buecher/12841/9783864904394-domain-driven-design-kompakt.html

Conway's Law,

"Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure"

[Conway 1968]



Quelle: Conway, M. E. "How Do Committees Invent?" Datamation 14, 4 (April 1968): 28–31.,
Genutzte Sekundärquelle: Bass, L. et al.: A Workshop on Architecture Competence, S. 15 | CMU/SEI-2009-TN-005

Hintergründe zum DDD

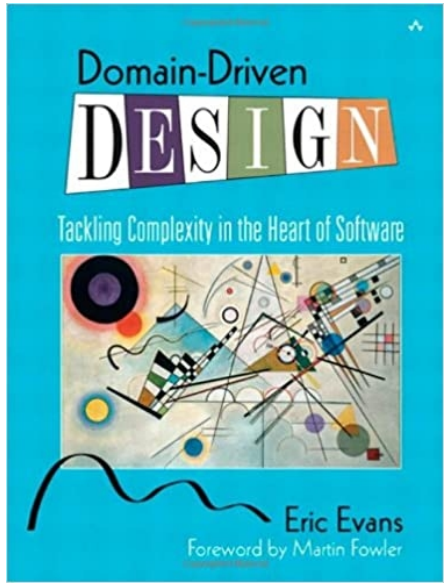
"*Für die meisten Softwareprojekte sollte der Fokus auf der jeweiligen Domäne und Domänenlogik liegen. Und das Design komplexer Domänen sollte auf Modellen aufbauen.*"

Diese Kernaussage von DDD kann man für die meisten Domänen noch etwas schärfer formulieren: "*Wenn man ein Problem nicht ohne Software lösen kann, kann man es mit Software (meist) auch nicht.*"

Quelle: Martincevic, NDDD: Context is King – Kein Context, keine Microservices,
https://www.informatik-aktuell.de/entwicklung/methoden/ddd-context-is-king-kein-context-keine-microservices.html

Aspekte im DDD

Aspekte im DDD

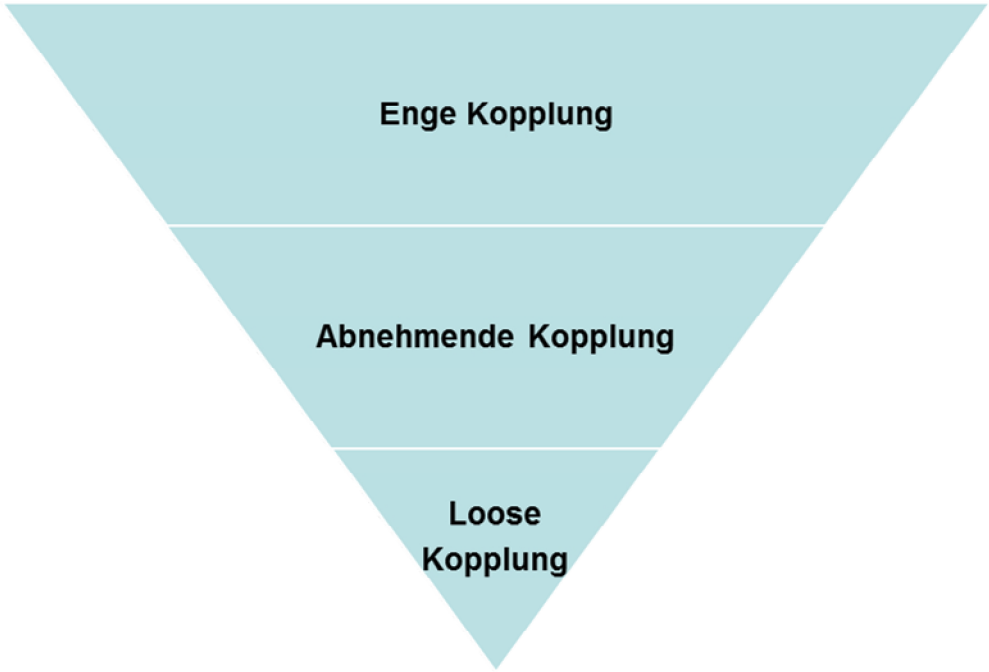


- Vermeidung monolithischer Ansätze.
- Aufteilung bzw. Schneidung komplexer Systeme.
- Bilden autonomer bzw. entkoppelter Einheiten.
- Fokus auf Modularität und Agilität.
- Fachliche Domäne als Treiber von z.B. Microservices.
- Analyse des Problem- (Fachlich) und Lösungsraums (Technisch).
- Modellbasierte Analyse orientiert an Architekturmustern.

Quelle der Abbildung: Evans, E.: Domainen-Driven-Design: Tackling Complexity in the heart of Software, Addison-Wesley, 2004, Blick ins Buch: https://www.amazon.de/Domain-Driven-Design-Tackling-Complexity-Software/dp/0321125215

Schnittstellen und Context Mapping

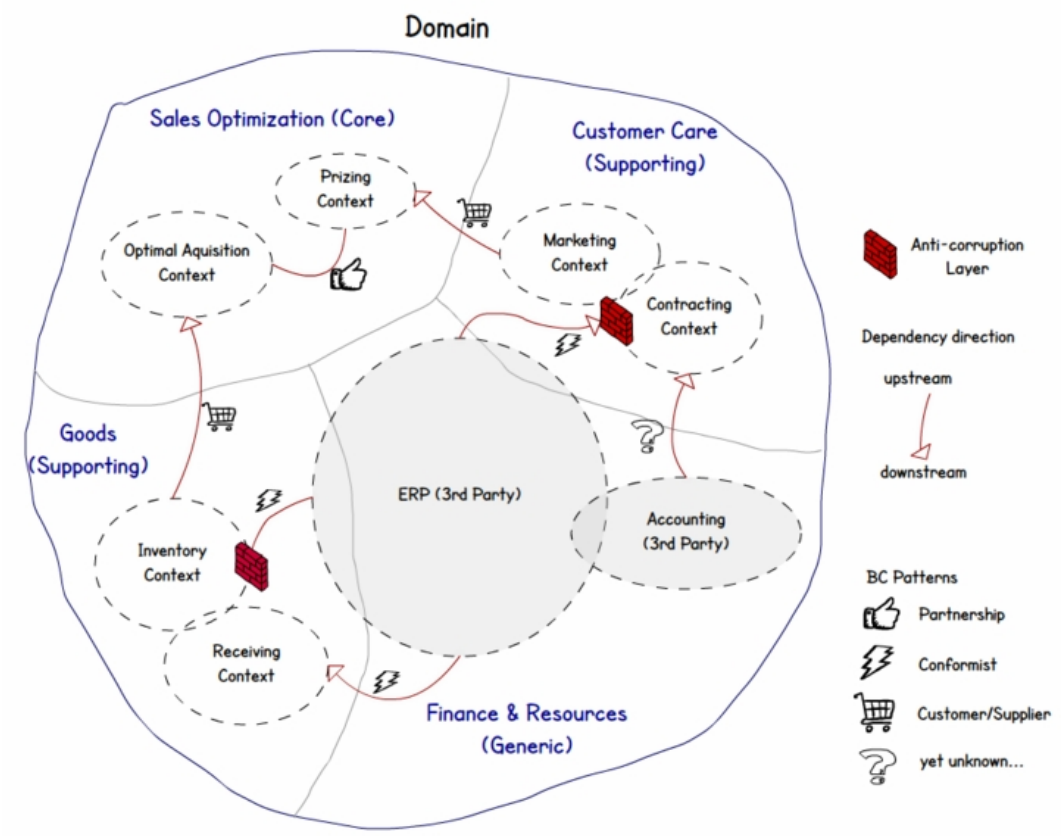
- Shared Kernel
- Customer/Supplier
- Published Language
- Open-Host-Service
- Anitcorruption Layer
- Conformist
- Seperate Ways



Quelle: Lilienthal, C.: Langlebige Software-Architekturen, S. 103-104, dpunkt.verlag, Heidelberg 2017
Originalquelle: Evans, E.: Domainen-Driven-Design: Tackling Complexity in the heart of Software, Addison-Wesley, 2004

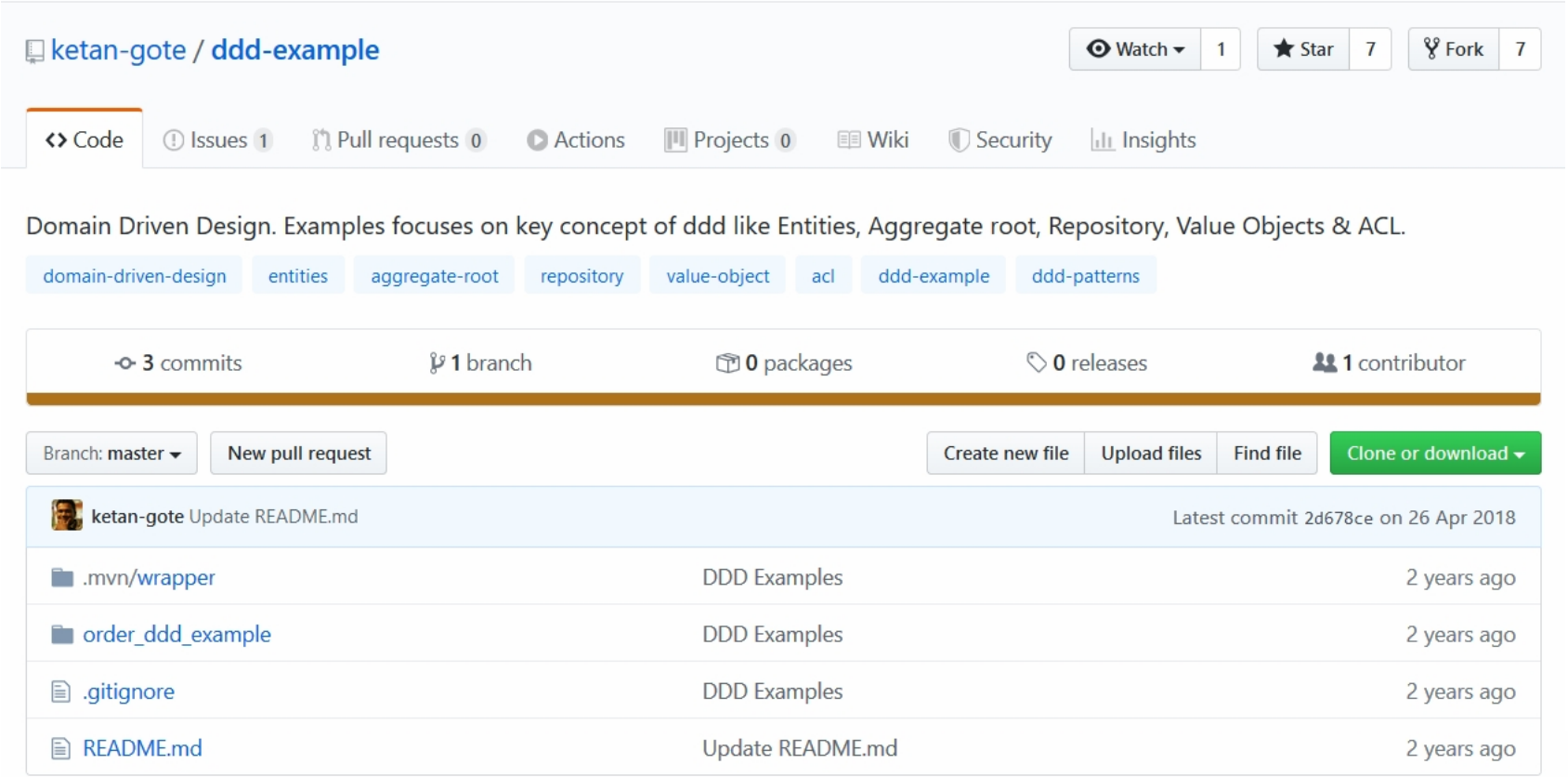
Beispiel einer Context Map

(Quelle: Nino Martincevic)



Quelle der Abbildung: Martincevic, N.: Context is King – Kein Context, keine Microservices, https://www.informatik-aktuell.de/entwicklung/methoden/ddd-context-is-king-kein-context-keine-microservices.html

DDD Beispiel unter GitHub



Quelle der Abbildung: GitHub – Beispiel eines DDD abgebildet mit Java,
https://github.com/ketan-gote/ddd-example

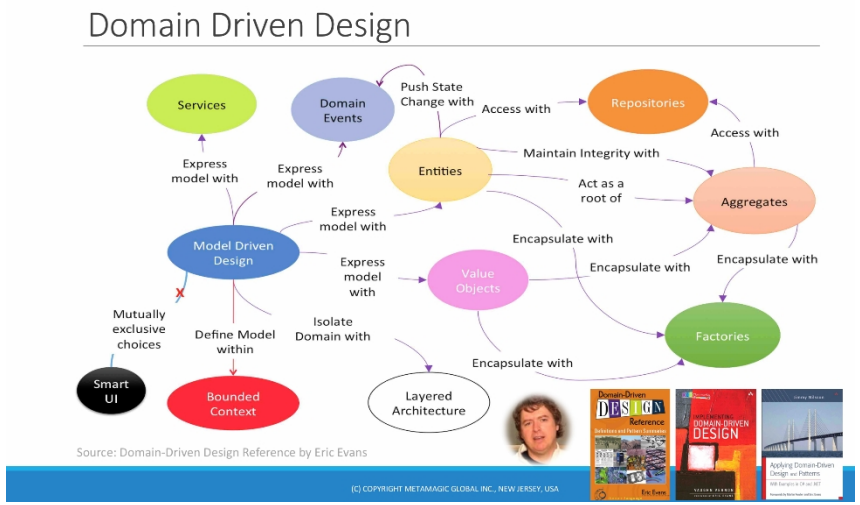
Möglichkeit zur Vertiefung

Ziele der Übung

- Idee des Domänen Driven Design (kurz DDD) erschließen.
- Möglichkeiten des DDD im Bezug auf Web-APIs (REST-APIs).
- Bewertung des TMF-Frameworks hinsichtlich des DDD.
- Bedarf einer fachlichen Herleitung von Web-APIs erkennen.

Aufgaben A

- Machen Sie sich mit der grundlegenden Idee des DDD vertraut.
- Was ist unter dem Begriff des „bounded bontext“ zu verstehen?
- Was wird unter dem Begriff der Fachsprache (ubiquitous language) im DDD verstanden?
- Warum sollte die Modularität von Web-APIs aus fachlichen Anforderungen erwachsen?

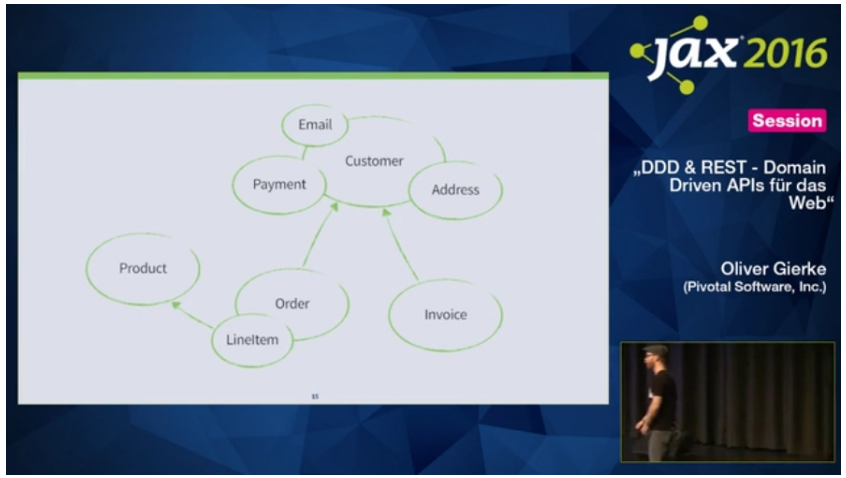


Quelle: https://github.com/ketan-gote/ddd-example

Originalquelle: Eric Evans, Domänen Driven Design

Aufgaben B

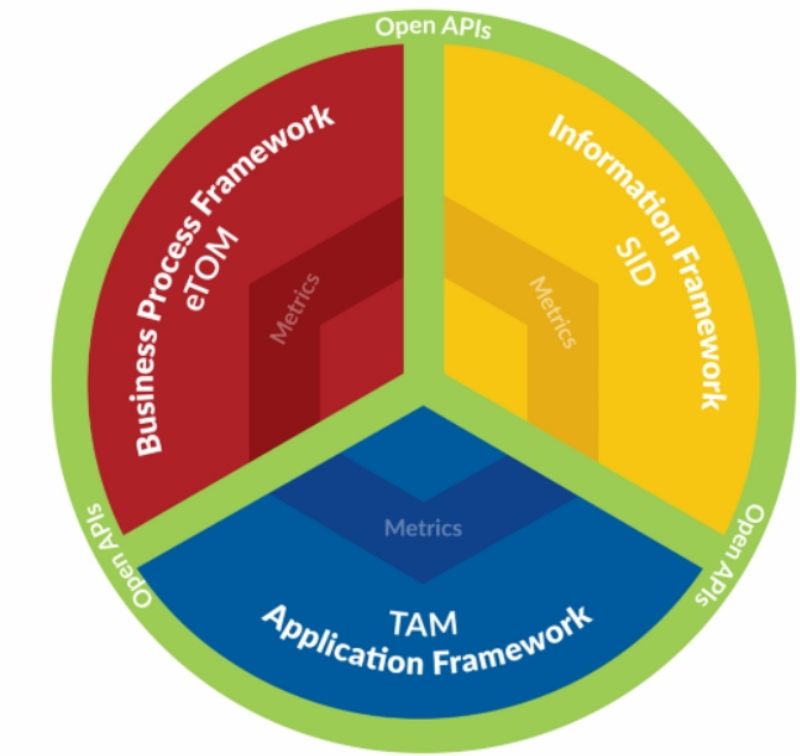
- Schauen Sie sich das Video (65 min.) „*Grundlegendes Domain-driven Design für Microservices*“ von von Oliver Gerke auf der W-JAX 2016 an.
- Geben Sie die aus ihrer Sicht wichtigsten Kernaussagen (mind. 10) dieses Vortrags stichpunktartig wieder!



Oliver Gierke: DDD & REST – Domain Driven APIs für das Web, https://jaxenter.de/ddd-rest-45713, 30. August 2016

Aufgaben C

- Welche Bezüge und Problembereiche des DDD sehen Sie im Zusammenhang mit der Idee des TMF-Frame-work (eTOM, SID und TAM)?
- Identifizieren Sie potentielle Bezüge zu den Open-API Spezifikationen des TMF bzw. in welcher Weise sollte das DDD bei diesen Services berücksichtigt sein?



Quelle der rechten Abbildung: Frameworx 19.5 - https://www.tmforum.org/frameworx-homepage, interaktive Modellerläuterung: http://casewise.tmforum.org/evolve/statics/frameworx/index.html?_ga=2.6214451.1210546848.1585908342-126195025.1585908342

Quellen

Interessante Quellen zum nachlesen

- Martincevic, N.: Context is King – Kein Context, keine Microservices, https://www.informatik-aktuell.de/entwicklung/methoden/ddd-context-is-king-kein-context-keine-microservices.html, 22. November 2016
- Gierke, O.: DDD & REST – Domain Driven APIs für das Web, https://jaxenter.de/ddd-rest-45713, 30. August 2016
- Roden, G.: Domain-driven Design erklärt, https://m.heise.de/developer/artikel/Domain-driven-Design-erklaert-3130720.html?seite=all, 11. März 2016