Themenspeicher für

Certified Professional for Software Architecture®
Expert Level

2024.1-rev0-DE-20240125

Inhaltsverzeichnis

Die Abkürzung "e. V." ist Teil des offiziellen Namens des iSAQB und steht für "eingetragener Verein", der seinen Status als juristische Person nach deutschem Recht beschreibt. Der Einfachheit halber wird iSAQB e. V. im Folgenden ohne die Verwendung dieser Abkürzung als iSAQB bezeichnet.

1. Themenspeicher

1.1. Auswirkungen von DDD auf das Unternehmen

Themeneinreicher: Hermann Woock

Status: Freigegeben

Hintergrund

Software Projekte werden mit der Zeit in der Regel zäher, zerbrechlicher und fehleranfälliger. Der Grund liegt oft in zu vielen Abhängigkeiten, schlechter Kohäsion und hoher Kopplung. Domain-Driven Design (DDD) verspricht hier Abhilfe.

DDD ist aber mehr als nur das Schneiden von Komponenten. Vieles davon wird oft übersehen, wodurch viele seiner Stärken nicht ausgenutzt werden können: Deep Insight, Break Through, Supple Design, Large Structures.

Außerdem ist DDD ein sehr agiler Ansatz, der bei richtiger Umsetzung starken Einfluss auf Arbeitsabläufe und Firmenstrukturen hat.

Aufgabenstellung

Durch die Bearbeitung des Themas soll gezeigt werden, welche Probleme mit DDD gelöst und was mit DDD erreicht werden kann.

Ebenso sollen aber auch die Auswirkungen aufs Unternehmen gezeigt werden, oder warum DDD scheitern kann. Welche Voraussetzungen müssen gegeben sein, damit DDD seinen Nutzen bringt?

Das Thema soll u. a. aus Sicht einer Organisation, des Managements, der Software-Entwickler und der Anwender beleuchtet werden.

Referenzen

Evans, E. (2003). Domain-Driven Design: Tackling Complexity in the Heart of Software. Pearson ITP.

Vernon, V. (2013). Implementing Domain-Driven Design. Pearson ITP.

Gruppengröße

Das Thema sollte von einer Gruppe mit 3-5 Personen bearbeitet werden.

1.2. Dokumentation von SW Architekturen

Themeneinreicher: Hermann Woock

Status: Freigegeben

Hintergrund

Dokumentation ist in der Softwareentwicklung genauso wichtig wie unbeliebt. In der Vergangenheit und unter Einfluss des Wasserfallmodells entstanden oft unübersichtliche Papierwüsten. In der agilen Softwareentwicklung hingegen wird Dokumentation oft als Ballast und überflüssig angesehen. Neue Medien und das Erkennen der Wichtigkeit für eine saubere Architekturdokumentation werfen ein neues Licht auf die Dokumentation.

Was sind Qualitätsmerkmale für eine gute Dokumentation der Architektur und wie führt man diese Erkenntnisse in ein Projekt ein? Welche Erfahrungen mit der Einführung und dem täglichen Gebrauch können an andere Teams weitergegeben werden? Welche Schwierigkeiten traten auf und wie wurde damit umgegangen. An welchen Stellen gibt es noch ungelöste Fragen und Probleme?

Aufgabenstellung

Folgende Fragen sollen behandelt werden:

- Welche Formen eignen sich für die Dokumentation von SW-Architekturen (Format, Ablageort, Struktur, verwendete Notationen)?
- Wie nutzt man Dokumentation als Kommunikationsmittel und nicht als Ablage?
- Wie werden Entscheidungen und Prinzipien dokumentiert?
- Wie können die Artefakte der SW-Architekturdokumentation mit anderen Artefakten (Anforderungen, Fehlerberichte, Testfälle, …) verknüpft werden (Traceability)?
- Wie identifiziert man die notwendigen Inhalte und bestimmt den Detaillierungsgrad der Dokumentation?

Referenzen

Zörner, S. (2021). Software-Architekturen dokumentieren und kommunizieren. Hanser Verlag

Starke, G. & Hruschka P. (2022). Arc42 in Aktion: Praktische Tipps zur Architekturdokumentation. Carl Hanser Verlag

Homepage of the ADR GitHub organization. https://adr.github.io/

Brown, S. The C4 model for visualising software architecture. https://c4model.com/

Gruppengröße

Das Thema sollte von einer Gruppe mit 3-6 Personen bearbeitet werden.

1.3. Einsatz unterschiedlicher NoSQL Datenbanken im Projekt

Themeneinreicher: Hermann Woock

Status: Freigegeben

Hintergrund

NoSQL-Datenbanken sind längst kein Novum mehr. In vielen Projekten sind sie Standard, in vielen aber auch nicht.

Wer bislang nur mit relationalen Datenbanken zu tun hatte, kann sich oft nicht vorstellen, wie der Einsatz von NoSQL-Datenbanken gelingen kann:

- Welche Möglichkeiten stecken in solchen Datenbanken und was bringen sie für einen Gewinn?
- Welche Typen von NoSQL Datenbanken gibt es und wofür können sie eingesetzt werden?
- Gibt es Gefahren, auf die man achten sollte?

Das Thema ist interessant für Architekten, Entwicklung, Betrieb, DB-Administratoren

Aufgabenstellung

Es sollen diese Fragen beantwortet werden:

- Welche Probleme lassen sich mit NoSQL-Datenbanken lösen und welche neuen Probleme bekomme ich?
- Wie kann man den besten Nutzen aus NoSQL-Datenbanken ziehen?
- Welche Fähigkeiten und Wissen sind erforderlich?
- Wie müssen NoSQL-Datenbanken in die Architektur berücksichtigt werden?

Weitere Themen in diesem Zusammenhang sind Event Sourcing, Eventually Consistency, Skalierbarkeit, Ausfallsicherheit, Lizenzen oder BASE

Ein paar gelungene Beispiele aus der Praxis mit Tipps und Tricks würden weiterhelfen.

Referenzen

Fowler, M., & Sadalage, P. J. (2012). NoSQL distilled. Pearson Education.

Gruppengröße

Das Thema sollte von einer Gruppe mit 3-5 Personen bearbeitet werden.

1.4. Erfahrungen mit Event-Driven Architekturen

Themeneinreicher: Hermann Woock

Status: Approved

Hintergrund

Nicht erst seit der Veröffentlichung von Vernons Buch „Implementing Domain-Driven Design“ mit dem Vorwort von Eric Evans sind Event-Driven Architekturen von Interesse. Event-Driven Architekturen (EDA) bieten ein hohes Maß an Flexibilität in der Software. Sei es im Programmablauf oder in der Datenerfassung. EDAs werden in vielen Orten bereits erfolgreich eingesetzt. Auch Cloud Provider bieten Lösungen an. Immer mehr Frameworks kommen hierzu auf den Markt. Monitoring, BI und viele weitere Bereiche machen von EDAs Gebrauch.

Event-Driven Architekturen helfen dabei, flexible Systeme zu schaffen, die sich an Veränderungen anpassen und Entscheidungen in Echtzeit treffen können. Ereignisse werden erfasst, sobald sie von Ereignisquellen wie Internet-of-Things-Geräten (IoT), Anwendungen und Netzwerken eintreten, sodass Event-Produzenten und Event-Consumer Status- und Reaktionsinformationen in Echtzeit austauschen können. In diesem Kontext wird auch häufig über die Konzepte des Event Sourcings und CQRS gesprochen, die weitere Herausforderungen bieten, aber auch spannende Vorteile versprechen.

Aufgabenstellung

Diese Fragen sollen behandelt werden:

- Für welche Einsatzzwecke eignet sich dieser Architekturstil?
- Welche Muster und Technologien können eingesetzt werden?
- Wie ordnen sich folgende Punkte in das Thema ein: CQRS, Event Sourcing, SAGA, Messaging, Reactive Programming, Kafka?
- Wie löst man folgende Probleme?
  - Überholung
  - Idempotenz
  - Versionierung
  - Inhalte von Events

Referenzen

Vernon, V. (2013). Implementing Domain-Driven Design. Pearson ITP.

Domogalla, M. (2022, April 21). Software-Architektur.tv: Softwarearchitektur MIT events, event sourcing UND CQRS. Developer. https://www.heise.de/news/software-architektur-tv-Softwarearchitektur-mit-Events-Event-Sourcing-und-CQRS-7061126.html

Gruppengröße

Das Thema sollte von einer Gruppe mit 3 bis 7 Personen bearbeitet werden.

1.5. Frontend-Entwicklung in großen Projekten

Themeneinreicher: Johannes Hochrainer

Status: Freigegeben

Hintergrund

Große Softwaresysteme werden in der Regel von mehreren Entwicklungsteams umgesetzt. Während sich im Backend Feature-Teams durchgesetzt haben, wird in vielen Projekten das Frontend von einem einzigen Team entwickelt. Doch auch im Frontend setzt sich immer mehr der Microservice-Gedanke durch: Unabhängige Teams arbeiten an einem fachlichen Frontend-Baustein und können ihren Baustein unabhängig von den anderen Teams entwickeln und ausliefern. Für die Umsetzung dieser sogenannten Microfrontend-Architektur gibt es verschiedenste methodische und technische Ansätze.

Aufgabenstellung

Folgende Fragen sollen beantwortet werden:

- Wie kann ein Frontend flexibel und effizient mit verschiedenen Entwicklungsteams umgesetzt werden? Dazu gehören u. a. Folgende:
  - Micro Frontends
  - Self-Contained Systems
  - Backends for Frontends Pattern
  - Organisationsinterne Komponentenbibliotheken (ähnlich Angular Material)
  - Webpack Module Federation
  - Monorepos (z. B. mit Nx)
- Wie baut man ein bestehendes monolithisches Frontend zu einem sauber modularisierten, parallel entwickelbaren Frontend um?

Referenzen

- Micro Frontends, https://micro-frontends.org/
- Self Contained Systems, https://scs-architecture.org/
- Backends for Frontends Pattern - Cloud Design Patterns, https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends
- Angular Material, https://material.angular.io/
- Nx Monorepo, https://nx.dev/
- Webpack Module Federation, https://webpack.js.org/concepts/module-federation/

Gruppengröße

Das Thema sollte von einer Gruppe mit 3 bis 5 Personen bearbeitet werden.

1.6. Maschinelles Lernen in der Praxis

Themeneinreicher: Hermann Woock

Status: Freigegeben

Hintergrund

Maschinelles Lernen (ML) entwickelt sich immer mehr von einer Fiktion zu einer nicht mehr wegzudenkender Technik. Wer sich nicht früh mit ML beschäftigt, für den kann der Zug später vielleicht schon abgefahren sein. Dennoch ist ein Investment sehr riskant.

Einsatzgebiete sind sehr vielfältig: Baubranche, Automobilindustrie, Finanzwesen, Behörden, Logistik, Marketing. Die Umsetzung erfordert allerdings neue Talente, und wirft Fragen auf. Niemand sollte diese Entwicklung verschlafen.

Im Rahmen der Digitalisierung oder der stärker werdenden Konkurrenz braucht es neue Lösungen, bei denen ML eine gute Hilfe anbieten kann. Im Bereich von Business Intelligence oder Big Data braucht es neue Verfahren, bei denen ML unterstützen kann.

1.7. Aufgabenstellung

Der Beitrag soll Mut machen, einen Einstieg in dieses Thema zu finden.

Es gibt viele Dinge zu klären:

- Welche Probleme lassen sich mit ML lösen?
- Welcher Aufwand steckt dahinter?
- Welche Fähigkeiten und Wissen sind bei der Umsetzung erforderlich?
- Welche rechtlichen Einflüsse müssen beachtet werden?
- Was sind die Erfolgsfaktoren?

Referenzen

Gruppengröße

Das Thema sollte von einer Gruppe mit 3-7 Personen bearbeiten werden.

1.8. Migration zu Microservices unter Kubernetes

Themeneinreicher: Hermann Woock

Status: Freigegeben

Hintergrund

Märkte werden komplexer, Software, wird im Bereich Business und Technologie komplizierter. Das nötigt Firmen immer häufiger dazu, die Lösung für ihre SW-Projekte in einer Microservices-Architektur (µService) zu suchen.

µServices werden in der Regel mit Kubernetes (K8s) realisiert. Kubernetes ist ein hoch kompliziertes und komplexes Framework. Entwickler sind mit deren Installation, Nutzung und Betrieb häufig überfordert. Projekte können so leicht scheitern oder unrentabel werden. Ein intensiver Austausch zwischen den Rollen Anforderungsmanagement, Architektur, Entwicklung, Test und Betrieb ist deshalb erforderlich. Fehlende Agilität und starre Strukturen im Unternehmen kommen erschwerend hinzu. Erste Ansätze gibt es dafür im DevOps. Weitere werden nötig.

Aufgabenstellung

Die Teilnehmer sollen die Probleme, Herangehensweise und Lösungen aus ihren Projekten aufarbeiten und einen Leitfaden erstellen.

Referenzen

Liebel, O. (2023). Skalierbare Container-Infrastrukturen: Das Handbuch für Administratoren (4. Ausgabe). Rheinwerk Computing.

Gruppengröße

Das Thema soll von einer Gruppe mit 6-7 Personen bearbeiten.

1.9. Modellierungssprachen

Themeneinreicher: Johannes Hochrainer

Status: Freigegeben

Hintergrund

Seit über 20 Jahren gilt UML als der Standard für die Modellierung von Software-Systemen. Andere Modellierungssprachen positionieren sich entweder als leichtgewichtige Alternativen (z. B. C4), als vollständigen Ersatz oder für einen bestimmten Anwendungsbereich bzw. Domäne (z. B. BPMN, SysML, ArchiMate).

Die Verwendung von Modellierungssprachen ist nicht immer beliebt. Sie werden oft als zu kompliziert oder aufwändig angesehen. Dennoch lassen sich große Projekte ohne Modellierung nicht umsetzen. Zu spät wird oft erkannt, dass man die Zusammenhänge im Projekt nicht mehr durchschaut und ein geeignetes Modell hier hätte helfen können. Noch besser wäre es gewesen, schon beizeiten mit der Modellierung zu beginnen, um die Kommunikation im Projekt zu fördern.

Aufgabenstellung

Die Themenarbeitsgruppe soll folgende Fragen diskutieren und die Erkenntnisse aufbereiten:

- Welche Erfahrungen wurden in den eigenen Projekten mit UML und deren Alternativen gemacht?
- Wann reichen informelle Modelle und wann ist eine formelle Modellierungssprache zu empfehlen?
- Wie lassen sich Entwicklungsteams davon überzeugen zumindest teilweise UML zu verwenden?
- Welche bewährte Praktiken gibt es beim Einsatz von UML?

Referenzen

- UML (Unified Modelling Language), https://www.uml.org/
- C4 model, https://c4model.com/
- FMC (Fundamental Modelling Concepts), http://www.fmc-modeling.org
- ArchiMate, https://www.opengroup.org/archimate-forum/archimate-overview
- Semantics of a Foundational Subset for Executable UML Models (fUML)
- Action Language for Foundational UML (ALF)
- Precise Semantics of UML Composite Structures (PSCS)
- Precise Semantics of UML State Machines (PSSM)

Gruppengröße

Das Thema sollte von einer Gruppe mit 4-7 Personen bearbeitet werden.

1.10. Sind SOLID-Prinzipien noch zeitgemäß?

Themeneinreicher: Johannes Hochrainer

Status: Freigegeben

Hintergrund

In den letzten Jahren gab es vermehrt Stimmen, die behaupteten, dass die SOLID-Prinzipien nicht mehr zeitgemäß wären. So sorgte etwa 2021 Dan North für Aufregung, als er behauptete, jedes einzelne SOLID-Prinzip wäre falsch. Nach etwas Zeit legte er nach und schlug als Alternative seine sogenannten „CUPID-Properties“ vor:

- Composable
- Unix Philosophy
- Predictable
- Idiomatic
- Domain-based

Aufgabenstellung

Die Themenarbeitsgruppe soll folgende Fragen klären:

- Welche Kritikpunkte gibt es an SOLID und können sie wissenschaftlich bestätigt werden?
- Erfüllt CUPID alle Anforderungen, um als Orientierung bei der Software-Entwicklung zu dienen?
- Gibt es Gemeinsamkeiten zwischen SOLID und CUPID?
- Gibt es Einsatzgebiete, die sich besonders für SOLID oder für CUPID eignen?
- Gibt es auch andere Sammlungen von Prinzipien, an denen sich Entwickler orientieren können?

Referenzen

- Principles Of Object Oriented Design, http://wiki.c2.com/?PrinciplesOfObjectOrientedDesign
- CUPID—the back story, https://dannorth.net/2021/03/16/cupid-the-back-story/
- CUPID—for joyful coding, https://dannorth.net/2022/02/10/cupid-for-joyful-coding/

Gruppengröße

Das Thema soll von einer Gruppe mit 3-5 Personen bearbeiten werden.