embarc Software Consulting GmbH
SIGS DATACOM FACHINFORMATIONEN FÜR IT-PROFESSIONALS

architektur SPICKER

Übersichten für die konzeptionelle Seite der Softwareentwicklung

MEHR WISSEN IN KOMPAKTER FORM:
Weitere Architektur-Spicker gibt es als kostenfreies PDF unter www.architektur-spicker.de

NR. 6

Agile Architektur

Softwarearchitektur wird in agilen Kontexten dynamischer, kleinteiliger, verteilter. Dieser Spicker fasst die wichtigsten Aspekte zusammen.

IN DIESER AUSGABE

Worum geht's? (Herausforderungen/Ziele)

- Agile Denkweise bringt Cross-Funktionalität, Iterativität, Flexibilität. Wie verändert sich die Architekturdisziplin dadurch?
- Agile Vorhaben sind schlank - auch in der Vorabarbeit. Wie kann trotzdem fundiert und gezielt an der Architektur gearbeitet werden?
- Agile Vorgehensmodelle sparen mit Aussagen zur Architektur. Wie geht man mit Architekturaufgaben und der Rolle Architekt um?
- In großen Entwicklungsvorhaben sind Kommunikation und ad-hoc Entscheidungen schwieriger. Wie ist agile Architekturarbeit hier sinnvoll möglich?

Grundlegende agile Ideen & Architektur

Die Top-3 Links zur agilen Denkweise:

TOP-3 LINKS

- Die Prinzipien des agilen Manifests: goo.gl/TXWaQt
- Der komplexe Bereich des Cynefin-Frameworks: goo.gl/ZNt87M
- Der OODA-Zyklus für schnelle Entwicklungsprozesse und Feedback: goo.gl/exBftJ

Was bedeuten zentrale agile Ideen aus diesen Quellen für die Architekturarbeit?

➔ **Agile Architektur** ist durch **Anforderungen getrieben**, vom **Aufwand** her dem Problem **angemessen**, von aktuellen Erkenntnissen zu **Zusammenarbeit** und Vorgehen beeinflusst und gut mit der **iterativen Entwicklung verzahnt**.

Zentrale Ideen von Agilität	Relevante Architekuraspekte
Iteratives Vorgehen	Kein Big Upfront Design, stattdessen schlanke Architekturvision
	Architekturtreiber im Backlog
	Entscheidungen zum letzten vernünftigen Moment (LVM)
Feingranulares Feedback	Tests für qualitative Eigenschaften
	Regelmäßige Reflexion zu Qualitätszielen
	Stetige Identifikation technischer Schulden
Cross-Funktionalität/ Team-Verantwortung	Verteilung der Rolle Architekt
	Verfahren für Konsensentscheidungen
	Prinzipien für die Architekturrichtung
Transparenz & direkte Kommunikation	Informativer Arbeitsplatz inkl. Architekturwand
	Ad-hoc Architekturworkshops
	Communities of Practice
Reaktionsfähigkeit/ Flexibilität	Vertikaler Architekturstil & Domänenorientierung
	Tiefe technische Isolation
	Self-Service Plattform und Infrastruktur
	Technische Exzellenz und Fokus auf Wartbarkeit
Produktorientierung	Evolutionäre Architektur
	Weiche Architekturstandards und Eventual Integrity
	Anti-Zähigkeit in der Architekturumsetzung

Methodische Aspekte
- Detailliert beschrieben in **Vorgehensmuster für Softwarearchitektur**
Stefan Toth
2. Auflage 2015
Carl Hanser

Technische Aspekte
- Microservices/Self-Contained-Systems
- Containerization
- Public/Private Cloud (siehe Spicker Nr. 5)

Organisatorische Aspekte
- Langfristige Ausrichtung auf Themen
- Weiche Governance
- siehe Seite 4

Besonders interessant bei Skalierung (große Vorhaben, hohe Dynamik)

Konzeption und Vorab-Arbeit

Architekturvision

Möglichst schlanke Zusammenstellung architektonischer Treiber (Was?) und Ideen (Wie?) als Gegenstück zur fachlich orientierten Produktvision.

Was?

- **Systemkontext** (Abgrenzung)
- **Rahmenbedingungen**
- **Qualitätsanforderungen** (priorisiert)
- **Risiken** (fachlich und technisch)

Grundlage für Architekturarbeit - Kontextunabhängig wichtig!

Wie?

- **Basistechnologien** (inkl. Frameworks etc.)
- **Konzepte, Muster, Prinzipien**
- **Fachliche Strukturierung** (+Domänenmodell)
- **Koordinations- u. Kommunikationslösungen**
- **Integrationslösungen u. Schnittstellen**
- **Persistenzstrategien u. Datenmodell**

Erste Ausgestaltungsideen* - Umfang und Detail je nach Kontext!

*Ziel ist die Schätzbarkeit des Systems (Makroebene) – NICHT die finale Festlegung!

Skalierungstreiber

- Hohe Qualitätsanforderungen
- Enger Projektrahmen (Zeit, Budget)
- Große Entwichlungsmannschaft
- Hoher räumlicher Verteilungsgrad
- Neue Technologien
- Weniger Erfahrung im Lösungsspektrum
- Dünner technischer Rahmen
- Viele (externe) Abhängigkeiten
- Abweichung von Standardarchitektur
- Vorhanden Zielkonflikte

Die Top-2 Unterschiede zu Big-Upfront Design (BUFD):

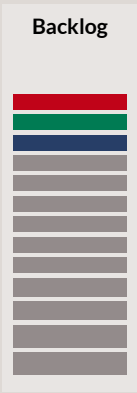
1. **Schlankere Ausprägung:** Die Detaillierung erfolgt risikoorientiert später, in den Iterationen. Offene Punkte sind OK wenn geplant bearbeitbar.
2. **Kandidaten statt Entscheidungen:** Finale Entscheidungen nur in uninnovativen/ bekannten Bereichen ohne Risiko. Kommuniziert werden „Kandidaten".

Architekturarbeit in den Iterationen

Arbeit in **frühen Iterationen** sollte neben erster sinnvoller Funktionalität, **möglichst viele Architekturansätze, möglichst dünn berühren**. Ziel ist, Aspekte der Architekturvision möglichst bald und schmerzfrei zu widerlegen. Grundlage hierfür: Das „**Walking Skeleton**".

„A Walking Skeleton is a tiny implementation of the system that performs a small end-to-end function. It need not use the final architecture, but it should link together the main architectural components. The architecture and the functionality can then evolve in parallel."

Alistair Cockburn



Architekturansätze

Fachliche Strukturierungselemente
✗ .....
☐ .....

Konzepte, Prinzipien, Muster
✗ .....
✗ .....

Basistechnologien u. Frameworks
✗ .....
✗ .....
✗ .....

Koordinations- u. Kommunikationslsg.
✗ .....
☐ .....

Integrationslsg. u. Schnittstellen
✗ .....

Persistenzstrategien u. Datenmodelle
✗ .....
✗ .....

Plattform u. Infrastrukturelemente
✗ .....
☐ .....

In weiteren Verlauf einer Produktentwicklung vor allem **Qualitätsziele** als Planungs- und Kommunikationsgrundlage verwenden. Hochlevel-Checkliste für die eigene **Reife der iterativen Architekturarbeit**:

Stufe 1	Stufe 2	Stufe 3	Stufe 4	Stufe 5
Grobe Qualitätsaussagen sind im Team bekannt und finden Beachtung (z.B. in Reviews)	Explizite Events wie Reflexionsmeetings oder Bewertungen prüfen regelmäßig den Status der Qualitätserreichung	Qualitätsszenarien werden verwendet, um mit Stakeholdern und dem PO zu kommunizieren	Qualitätsszenarien werden entweder als Akzeptanzkriterien oder als eigene Stories im Backlog geführt	Alle architektonischen Tätigkeiten sind durch Qualitätsziele motiviert (inkl. Technischer Schulden)

(Nicht-funktionale) Tests und CI/CD Techniken geben Feedback zur Zielerreichung: grob → feingranular

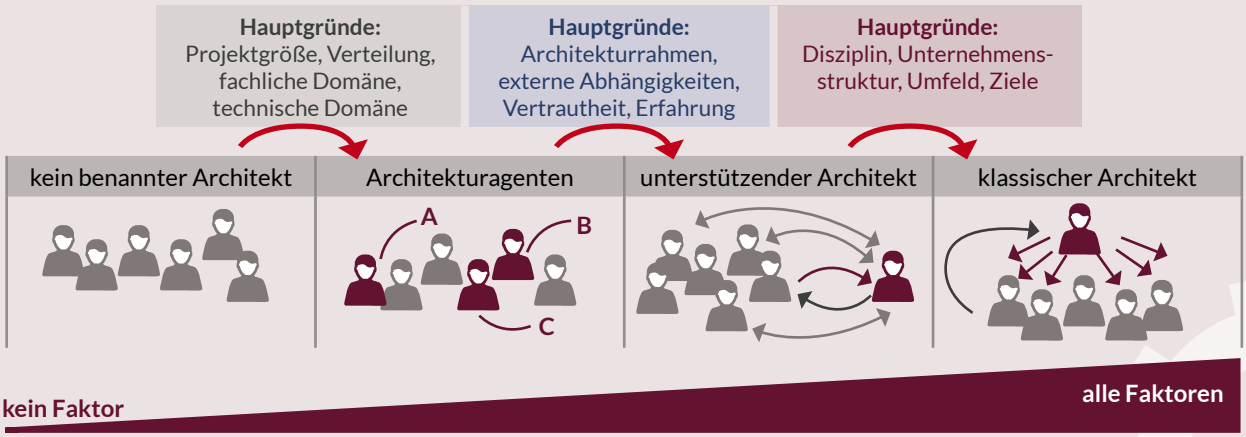
→ Qualitätsszenarien: siehe Spicker Nr. 4 – Architektur-Reviews

Die Rolle Architekt

Neben dem klassischen Architekten gibt es einige Möglichkeiten die Architekturrolle gemeinsam auszufüllen.
Die „richtige“ Wahl ist am Umfeld und der Aufgabe festzumachen – den „Architektenfaktoren“.

Architektenfaktoren

- **Projektgröße:** mehrere Teams
- **Verteilung:** geografisch verteilt
- **Fachliche Domäne:** komplex, neu
- **Technische Domäne:** schwierig, herausfordernd, neu
- **Architekturrahmen:** muss erst geschaffen werden
- **Externe Abhängigkeit:** hoch
- **Vertrautheit:** erstes Projekt in dieser Zusammensetzung
- **Erfahrung:** viele unerfahrene Entwickler
- **Disziplin:** Verantwortungsübernahme mangelhaft
- **Unternehmensstuktur:** stark hierarchisch
- **Umfeld:** reguliert oder von Standards bestimmt
- **Ziele:** Architekturziele in Konflikt (auch zu Projektzielen)



Taktiken um weiter links und damit dynamischer zu agieren ohne Architektur zu vernachlässigen:

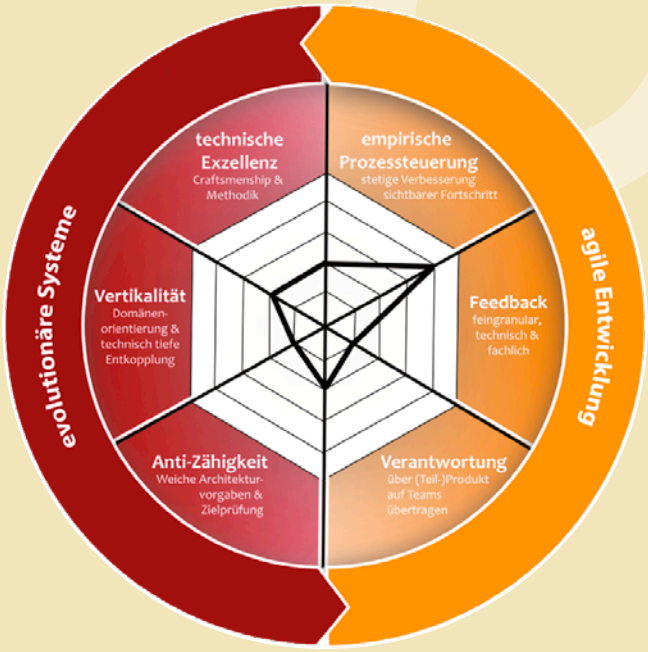
- **Informativer Arbeitsplatz:** Sichtbare Architekturartefakte (z.B. Architekturwand) als Kommunikationsbasis
- **Gemeinsame Entscheidung:** Konsensbasierte Entscheidungsverfahren zur Verantwortungsstärkung
- **Wiederholte Reflexion:** Explizite Events prüfen den Status der Qualitätserreichung und synchronisieren
- **Architekturcommunities:** Der Architekturdisziplin verschriebene Veranstaltungen verbreitern Wissen
- **Architekturarbeit im Backlog:** Architekturaufgaben werden in Form von Qualitätsszenarien sichtbar und verteilbar
- **Architekturprinzipien:** Transportieren wichtige Erkenntnisse und Denkweisen – machen Architekturarbeit ähnlicher
- **Qualitative, automatisierte Tests:** Feedback zur Erreichung von Architekturzielen macht Verantwortung real

Skalierung von Agilität & Evolutionäre Architektur

Die größten agilen Herausforderungen bei Vorhaben mit 5 Teams und mehr sind die **Beibehaltung der Reaktionsfähigkeit** und gesund **verteilte Verantwortung** ohne Flaschenhälse. Um diese Herausforderungen zu meistern müssen organisatorisch/methodische Aspekte von Agilität mit den richtigen technisch/architektonischen Konzepten verheiratet werden:

„The technical architecture is hugely important for the way we are organized. The organizational structure must play in harmony with the technical architecture. Many companies can't use our way of working because their architecture won't allow it.“

Henrik Kniberg (über Spotify)



Das **ADES**-Framework (**A**gile **D**elivery and **E**volutionary **S**ystems) verzahnt technische und organisatorische Aspekte, um Agilität in größeren Produkt- und Unternehmenskontexten effektiv zu machen: www.ADES-Framework.org

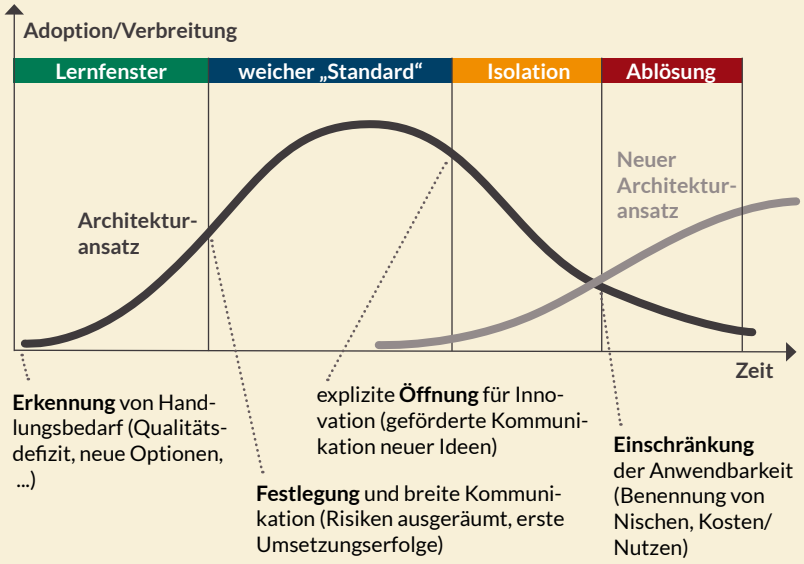
Evolutionäre Architektur

Die linke Seite des ADES-Frameworks.

Evolutionäre Systeme/Architektur	Das Gegenmodell: Projektorientierte Entwicklung
Produktgedanke: Langfristige Versorgung eines Themas mit einer sich wandelnden Lösung	**Projektgedanke:** Zeitlich beschränkte Systementwicklung, gefolgt von Wartungsphasen und Neuentwicklung
Möglichst **konstante Teams** langfristig mit Thema/Produkt verknüpft	**Wechselnde Mannschaften** für Entwicklung und Wartung.
Sich technisch **wandelnde Architekturbasis** (keine kompletten Neuentwicklungen)	**Konstante Architekturbasis** für einen Projekt/ Wartungszyklus.
Technisch **tiefe Entkopplung** zwischen (Sub-)Domänen für kleinteilige Änderung	Eher **technische Standardisierung** und Vereinheitlichung (kein Muss aber üblich)
Eventual Integrity – Integrität der Lösung ist gegeben, wenn sich gute Ideen durchsetzen	**Standards First** – Nach Analyse bereits fixe Vorgaben, danach unerwünschte Abweichung/Governance
Gleichbeibende Qualität über die Zeit	**Schwankende Qualität** über Projekt/Wartungszyklus
Konstantes Investment in ein Thema, oft jährliche Budgets	Investment über Projekte und **zeitlich beschränkte Budgets**

➔ **Evolutionäre Ansätze** eignen sich vor allem bei **neueren Themen**, Vorhaben mit **höherem Innovations- oder Marktdruck**. Der Heimat von agilen Ansätzen eben…

Evolutionäre Betrachtung einer Architekturfragestellung über die Zeit:



Wichtige Konzepte agiler Architektur den Phasen zugeordnet:

- **Letzter vernünftiger Moment:** Entscheidungen werden so spät wie sinnvoll möglich getroffen um das Lernfenster zu vergrößern und kostspielige Fehler in der Breite zu vermeiden.
- **Vertikalisierung/technische Isolation:** Geringere technische Standardisierungstiefe ermöglicht das „Ausprobieren“ neuer Ansätze in realen Umfeldern, mit einer kleinen isolierten Fachlochkeit.
- **Anti-Zähigkeit:** Die momentan beste Lösung wird in der Anwendung so vereinfacht, dass Entwickler nicht aus Faulheit abweichen. Statt harter Governance wird die Zielerreichung geprüft.
- **Communities of Practice:** Haben Architekturansätze den Zenith überschritten ist der Austausch unter den Entwicklern und die Beschäftigung mit Trends besonders wichtig.
- **Eventual Integrity:** Abweichung und Innovation ist immer zugelassen. Über Communities verbreitete Ideen, die sich gegen Zähigkeit breit durchsetzen führen schlussendlich zu Integrität.

Weitere Informationen

- Practices for Scaling Lean and Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum“, Craig Larman, Addison Wesley 2010
- ADES Framework: www.ADES-Framework.org
- Die Top-3 Links zur agilen Denkweise (siehe Seite 1)
- Spotify Culture: https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1

Wir freuen uns auf Ihr Feedback: spicker@embarc.de

http://architektur-spicker.de

embarc Software Consulting GmbH
http://www.embarc.de
info@embarc.de

SIGS DATACOM FACHINFORMATIONEN FÜR IT-PROFESSIONALS
http://www.sigs-datacom.de
info@sigs-datacom.de