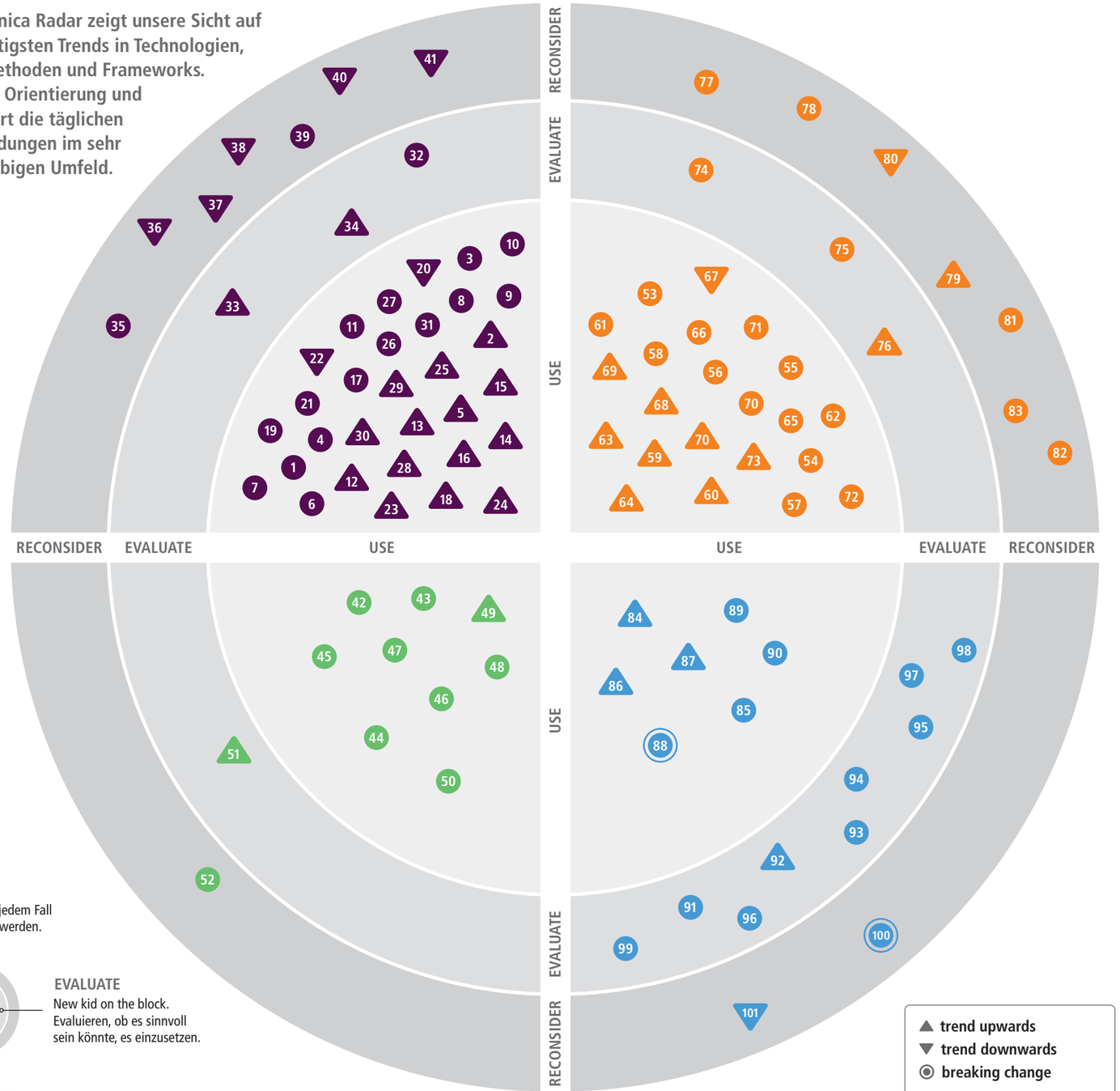
TECHNICA RADAR – VOL. 1

bbv
MAKING VISIONS WORK.

Der Technica Radar zeigt unsere Sicht auf die wichtigsten Trends in Technologien, Tools, Methoden und Frameworks. Er bietet Orientierung und erleichtert die täglichen Entscheidungen im sehr schnelllebigen Umfeld.



Concept or theme

USE
1 Accessibility (Barrierefreiheit)
2 Adaptive Design ▲
3 Big Data
4 Chatbot / Conversational Interfaces
5 Clean Architecture / Ports & Adapters / Hexagonal Architecture / Onion Architecture ▲
6 Clean Code
7 Cloud Architecture
8 Collective (Code) Ownership
9 Content Strategy
10 DevOps
11 Domain Driven Design
12 Emergent Architecture ▲
13 Event Sourcing ▲
14 Evolutionary Architecture ▲
15 Functional Programming ▲
16 Gamification ▲
17 Information Architecture
18 Internet of Things (IoT) ▲
19 IT Security
20 Lean Startup ▼
21 MVP, MMP, Earliest Testable/Useable/Loveable Product
22 Object Oriented Programming (OOP) ▼
23 Product Life Cycle Based Development ▲
24 Progressive Web Applications (PWA) ▲
25 Reactive Programming ▲
26 REST
27 Self-contained Systems
28 Serverless Architectures ▲
29 Software Modernisation ▲
30 Web Components ▲
31 OPC Unified Architecture

EVALUATE
32 Advanced Analytics und Machine Learning
33 Crowdfunding ▲
34 Mixed (Virtual) Reality ▲

RECONSIDER
35 Crowdtesting
36 Microservices ▼
37 Private Cloud ▼
38 Responsive Design ▼
39 Service Oriented Architecture (SOA)
40 Story Points ▼
41 N-Tier Architecture ▼

Tools

USE
42 Cumulocity IoT
43 Microsoft Azure
44 Amazon Web Services (AWS)
45 Cloud Computing
46 Docker
47 Kubernetes
48 Monitoring
49 Software as a Service (SaaS) ▲
50 OpenShift

EVALUATE
51 Cognitive Services ▲

RECONSIDER
52 Apache Kafka

USE
Dies kann in jedem Fall angewendet werden.

EVALUATE
New kid on the block. Evaluieren, ob es sinnvoll sein könnte, es einzusetzen.

RECONSIDER
Technologien und Rahmenbedingungen ändern sich und müssen manchmal neu betrachtet werden.

▲ **trend upwards**
▼ **trend downwards**
◉ **breaking change**

Beobachtete Trends im Markt und in Projekten. Die Analyse unserer Experten dazu auf der Folgeseite.

Method or technique

USE
53 Acceptance Test Driven Development (ATDD) / Behavior Driven Development (BDD)
54 Architecture Tradeoff Analysis Method (ATAM)
55 Continuous Delivery
56 Continuous Deployment
57 Continuous Integration
58 Continuous Refactoring / Refactor Mercilessly
59 Cost of Delay ▲
60 Event Storming ▲
61 Kanban
62 Large Scale Scrum (LeSS)
63 Liberating Structures ▲
64 Mob Programming ▲
65 Pair Programming
66 Risk Management
67 Scrum ▼
68 Service Design ▲
69 Systems Thinking ▲
70 Test Driven Development (TDD)
71 Usability Test
72 User Research
73 User Centered Design ▲

EVALUATE
74 Disciplined Agile 2.0 aka DAD (Disciplined Agile Delivery)
75 Scrum@Scale
76 Infrastructure as Code ▲

RECONSIDER
77 BPMN 2.0
78 Burn down/up chart
79 Design Thinking ▲
80 Effort Estimation ▼
81 Nexus
82 Scaled Agile Framework (SAFe)
83 UML

Libraries, Frameworks, Programming Languages

USE
84 .NET 5 ▲
85 BenchmarkDotNet
86 gRPC ▲
87 Python ▲
88 WPF ◉
89 Spring Framework
90 C++

EVALUATE
91 Azure IoT Edge
92 Blazor ▲
93 F#
94 GO Programming Language
95 GraphQL
96 Kotlin
97 R
98 Rust
99 WebAssembly

RECONSIDER
100 .NET Framework (Full-Framework NICHT Core) ◉
101 Windows Communication Foundation (WCF) ▼

Die Analyse der wichtigsten Trends

▲ trend upwards ▼ trend downwards ◉ breaking change

Concept or theme

USE

2 Adaptive Design ▲

Im Vergleich zu Responsive Design, bei dem sich das UI bei jeder Änderung an die Bildschirmgrösse anpasst, verwendet Adaptive Design fixe Bildschirmgrössen, an denen sich das UI neu ausrichtet. Damit ist Adaptive Design flexibler und reduziert die Komplexität in der Entwicklung im Vergleich zu Responsive Design.

5 Clean Architecture / Ports & Adapters / Hexagonal Architecture / Onion Architecture ▲

Unsere Erfahrungen in unseren Mandaten zeigen, dass das Einhalten dieser Architekturpattern zu flexibleren und wartbareren Systemen führt.

6 Clean Code

Obwohl die Idee bereits einige Jahre alt ist, sehen wir die Konzepte und Ideen hinter Clean Code immer noch als valid. Wir trennen in unserer Bewertung bewusst Konzept und Autor.

12 Emergent Architecture ▲

In einer komplexen Umgebung ist es unmöglich, eine vollständige Architektur zu entwerfen, dennoch sollte sie die aktuellen architekturellen Probleme lösen. Solch eine Architektur wird fortwährend weiterentwickelt (emerge), indem Architekturprinzipien wie Modularisierung und Clean Architecture kontinuierlich angewendet werden, statt die Architektur vor der Entwicklung zu entwerfen.

13 Event Sourcing ▲

Event Sourcing ist hilfreich, um Aktionen in einer Business-Domäne zu verstehen und zu modellieren. Wenn ein Event in einem Geschäftsprozess ausgelöst wird, kann ein System auf diesen reagieren. Wenn diese Events gespeichert werden, wird zusätzlich Wissen aufgebaut durch die Analyse des Eventlogs des Systems.

14 Evolutionary Architecture ▲

In einer komplexen Entwicklungsumgebung muss die Software-Architektur flexibel bleiben. Aufgrund konstanter Änderungen der eingesetzten Technologien, der Cloud-Dienste und Business-Prozesse müssen gewisse Teile der Architektur ersetzt, andere verbessert werden.

15 Functional Programming ▲

Es ist einfacher geworden, mit funktionalen Sprachen zu arbeiten. Viel mehr Entwickler nutzen funktionale Programmiersprachen und deren Vorteil der Typsysteme, um die Geschäftsdomäne zu modellieren. Der Fokus auf «Immutability» und zustandslosem Design vereinfacht die Implementation und Verständlichkeit der Businesslogik.

16 Gamification ▲

Gamification hat den Hype-Status überschritten und findet den Weg in die Realität. Anstelle einer rein visuellen Umsetzung von Gamification (Badges, Scores) sollten grundlegende Mechaniken und Prinzipien von Gamification angewendet werden. So können Mikro- und Makrolösungen der Gamification systematisch eingesetzt werden, um intrinsische Motivation mit extrinsischen Triggern anzusprechen. Dieser Ansatz führt zu messbar besserem User-Engagement oder adaptiertem Nutzerverhalten (z. B. Erfolgsraten, Zeit, um eine Aufgabe zu erledigen).

18 Internet of Things (IoT) ▲

IoT hat sich etabliert und ist zum Teil des IT-Lösungsraumes geworden. IoT wächst stetig und entwickelt sich weiter.

20 Lean Startup ▼

Falls Sie in einem Startup arbeiten, verwenden Sie es. Wenn Sie im Kontext einer etablierten Firma arbeiten, sollten Sie berücksichtigen, dass Sie sehr wahrscheinlich Kultur und Einstellung ändern müssen, damit es funktioniert. Wir beobachten, dass die meisten Organisationen nicht fähig sind, diese Veränderungen durchzuführen.

22 Object Oriented Programming (OOP) ▼

Die Evolution der Programmier-Paradigmen geht weiter. Für die meisten Systeme (z. B. Web Backend) ist das funktionale Paradigma besser geeignet, da Immutability der Standard ist. Auch die algebraischen Datentypen sind eine bessere Alternative, um die Domäne zu beschreiben.

23 Product Lifecycle-based Development ▲

Wie in der klassischen Industrie, so muss sich die Software-Entwicklungsindustrie über den ganzen Software-Produkt-Lebenszyklus Gedanken machen. Dies kann die Gesamtheit der Kosten einer Investition verkleinern, die über ihren kompletten Lebenszyklus anfallen (total cost of ownership). Im Vergleich zu stabilen Projektteams führen stabile Produktteams zu weniger Spannungen. Weitere Resultate sind eine kontinuierliche, stabile Weiterentwicklung, sich evolvierende Systeme und stabileres Domänenwissen.

24 Progressive Web Applications (PWA) ▲

Mit der Verbesserung der Web-Technologien durch Notifikationen und einen lokalen Persistenz-Layer werden PWA mehr und mehr verwendet, um native Applikationen abzulösen und Desktop-Applikationen mit On- und Offline-Fähigkeiten umzusetzen.

25 Reactive Programming ▲

Das reaktive Programmiermodell passt zu modernen eventgetriebenen Geschäfts-Szenarien. Die weitreichende Technologieunterstützung hilft Teams, dieses Modell zu adaptieren.

28 Serverless Architectures ▲

Mehr und mehr Applikationen werden in der Cloud betrieben. Gleichzeitig wachsen die Fähigkeiten von Cloud-Diensten. Mit diesen neuen Möglichkeiten werden Serverless-Architektur-Ansätze wichtiger, speziell in hochdynamisch skalierenden Szenarien.

29 Software Modernisation ▲

Um mit den sich schneller ändernden Erwartungen von Nutzern und Unternehmen mitzuhalten, müssen Wege gefunden werden, um existierende IT-Systeme kontinuierlich zu modernisieren, um neue und entstehende Technologien sowie neue Geschäftsmodelle nutzen zu können.

30 Web Components ▲

Diese verkleinern die Komplexität der Web-Entwicklung. Durch das Definieren von Web Components werden etliche Teile der Webseite einfacher.

EVALUATE

33 Crowdfunding ▲

Mit Crowdfunding können finanzielle Risiken beim Umsetzen von neuen Services und Produkten reduziert werden. Crowdfunding kann für kleine bis mittlere Unternehmen eine Möglichkeit bieten, neue Produkte zu finanzieren.

34 Mixed (Virtual) Reality ▲

Aufgrund der verfügbaren und erschwinglichen Geräte für Endanwender können immer mehr Anwendungsfälle für «mixed reality» gefunden und realisiert werden. Die Design-Tools, um Inhalte zu erstellen, sind weder intuitiv noch einfach zu bedienen, daher ist die Erstellung von guten Inhalten für «mixed reality» noch aufwändig und teuer.

RECONSIDER

36 Microservices ▼

Microservices wurden vom Hype zur Realität. Die Hoffnung, dass sie die Lösung für alles sind, hat sich nicht bewahrheitet. Fakt ist, dass der Aufwand, um all diese Services zu orchestrieren, sehr gross und teilweise sogar unwartbar ist. «Microservices», die eine Business-Komponente kapseln, sind trotzdem sehr nützlich – unter Berücksichtigung der «Fallacies of distributed Systems».

37 Private Cloud ▼

Die Hauptgründe für eine Private Cloud sind Datenschutz und rechtliche Vorbehalte gegenüber der Public Cloud. Wir beobachten in den meisten Branchen aber sich stetig abbauende Vorbehalte, besonders aufgrund der Investitionen in den Datenschutz seitens der Public-Cloud-Anbieter. Die konstante Innovation bei Public-Cloud-Anbietern ermöglicht Zugriff auf neue Dienste. Das ermöglicht rasch das Erschaffen und Testen neuer Produkte, ohne grosse initiale Kosten für die Infrastruktur und den Wissensaufbau.

38 Responsive Design ▼

Responsive Design ist stark, bleibt jedoch in der Erstellung und Umsetzung kompliziert und ist daher teuer und nicht einfach in der Wartung. Es wird durch Adaptive-Design-Lösungen verdrängt werden, die simpler in der Erstellung, Entwicklung und Wartung sind.

40 Story Points ▼

Fast niemand hat «Komplexität vergleichen» verstanden, daher zerfällt auch die Idee, Komplexitäten zu vergleichen und zu skalieren. Simples Zählen von Items (oder Stories) ist einfacher und gleich effektiv.

41 N-Tier Architecture ▼

Anstelle von typischen n-tier-Architekturen werden service-basierte Systeme entwickelt.

Tools

USE

49 Software as a Service (Saas) ▲

Die Digitalisierung von IT-Administratoren. Reduziert die Kosten, um IT-Infrastruktur bereitzustellen und zu warten.

EVALUATE

51 Cognitive Services ▲

Die Verwendung von ML-/AI-Mechanismen ist viel einfacher geworden, ebenso sie zu implementieren und zu integrieren. Damit ist die Einstiegsbarriere gesunken, sie in Produktionssystemen einzubinden. Die verfügbaren Services sind meistens auf ein weites Einsatzgebiet optimiert und nicht auf spezifische Anwendungsfälle.

Method or technique

USE

59 Cost of Delay ▲

Immer mehr Firmen sehen die Vorteile einer schnellen Produkteinführung (time-to-market), um früh Geld zu verdienen. Der ökonomische Effekt von Verzögerungen (cost of delay) wird immer mehr ein Faktor für die Priorisierung von Produktinkrementen.

60 Event Storming ▲

Machen Sie abstrakte Dinge für Entwickler und Wirtschaftler (business person) fassbar, indem Sie zusammen die Geschäftsdomäne erforschen und verstehen. Gleichen Sie das verwendete Vokabular aneinander an, und definieren Sie gemeinsam verwendete Begriffe.

63 Liberating Structures ▲

Die «liberating structures» haben sich als hilfreich herausgestellt. Die Selbstorganisation in abgestecktem Rahmen hat gezeigt, dass Mitarbeitende sich mehr mit den Arbeitsergebnissen identifizieren. Wenden Sie es an, und Sie werden erstaunt sein von dem erzeugten Mehrwert.

64 Mob-Programming ▲

Wenn alle Teamfähigkeiten und das komplette Wissen bei einer massgeblichen Softwarekomponente einbezogen werden, lohnt es sich, Mob-Programming anzuwenden.

67 Scrum ▼

Entweder haben es Teams (oder ihre Umgebung) nicht verstanden, oder die Teams, die es verstanden haben, wenden es nicht mehr in der gleichen Weise an. Die Teams, die es verstanden haben, verbesserten ihre Art zu arbeiten und haben sich von den in Scrum enthaltenen Einschränkungen emanzipiert. Nebenbei – Scrum und SAFe funktionieren nicht wirklich zusammen.

68 Service Design ▲

Eine der Disziplinen, die aufgrund der Digitalisierung Produkte zu Services transformiert. Nicht nur das Produkt an sich ist Teil des Geschäfts, sondern auch die Wartung und die Administration des Produkts als ein Service.

69 System-Thinking ▲

System-Thinking ist sehr machtvoll für die Analyse und Konzeption von komplexen Systemen. Es bildet eine Brücke zwischen Businesslogik und dem Fluss und der Verarbeitung der Daten in Systemen. Die Präzision von systematischen Einschränkungen (es gibt kein «hier passiert die Magie») unterstützt Teams, Entscheidungen zu finden und bei Design-Kompromissen (trade-off) zu entscheiden.

73 User-Centered-Design ▲

Nutzer werden immer selbstbewusster. Mit den zunehmenden Alternativen an Lösungen, um seine Ziele zu erreichen, wird der Nutzer diejenige Lösung wählen, die am einfachsten zu verwenden ist. Am einfachsten für den Nutzer, und niemanden sonst!

EVALUATE

76 Infrastructure as Code ▲

Infrastrukturen werden komplizierter. Um die Qualität aufrecht zu erhalten und Migrationen zu automatisieren, werden Infrastrukturen in maschinen-lesbarem Format definiert und versioniert. Lösungen die eine Beschreibung wie die Infrastruktur aussehen soll sind zu bevorzugen. Ein trade-off zwischen manuell und IaC sollte bedacht werden.

RECONSIDER

79 Design-Thinking ▲

Wieso «Reconsider»? Immer mehr Unternehmen verfügen über Innovationsabteilungen, die grossartige Produktideen generieren. Doch meistens findet nur eine kleine Anzahl der Ideen den Weg in die Produktentwicklung. Auch wenn Workshops auf dem Markt mit dem Titel «Design-Thinking» angeboten werden, ist zu beachten, dass der gesamte Design-Thinking-Prozess Wochen oder Monate benötigt, bis die Methode Wirkung zeigt.

80 Effort Estimation ▼

In einem komplexen Umfeld, das, basierend auf der Definition von Komplexität, aus «unknown unknowns» und keinen im Voraus bekannten Ursache-Wirkung-Beziehungen besteht, werden wiederverwendbare Lösungen und eine zuverlässige Aufwandschätzung immer unwahrscheinlicher. Anstelle von vermeintlich genauen Plänen sollten Teams kontinuierlich an Produktinkrementen arbeiten, die den bestmöglichen Wert generieren.

Libraries, Frameworks, Programming Languages

USE

84 .NET 5 ▲

.NET Core ist die Zukunft und nach dem aktuellen .NET Core 3.x steht .NET 5 vor der Tür. Dies wird die einzige Version sein, welche weiterentwickelt wird.

86 gRPC ▲

Der Standard – basierend auf den Nutzungszahlen – für synchrone Kommunikation zwischen Services.

87 Python ▲

Aufgrund der Fähigkeiten, einfach strukturierte Daten zu prozessieren, erlebt Python eine Renaissance in den Bereichen Machine Learning und Big Data.

88 WPF ◉

Mit dem Erscheinen und seiner Modernisierung in .NET Core ist WPF zurück im Spiel.

EVALUATE

82 Blazor ▲

Eine in das .NET-Ökosystem integrierte Web-Entwicklungsumgebung. Der Pluspunkt: bessere Zugänglichkeit der Webentwicklung für .NET-Entwickler.

RECONSIDER

100 .NET Framework ◉

.NET Framework 4.8 wird die letzte Major Version vom .NET Framework sein. Microsoft markiert das Framework als «feature complete» und wird nur noch Bugfixes und Security Fixes releasen.

101 Windows Communication Foundation (WCF) ▼

Wenn du merkst, dass du ein totes Pferd reitest, steige ab! (WCF wird in .NET 5 obsolet sein)

Vol. 1 – Published: Nov 2020