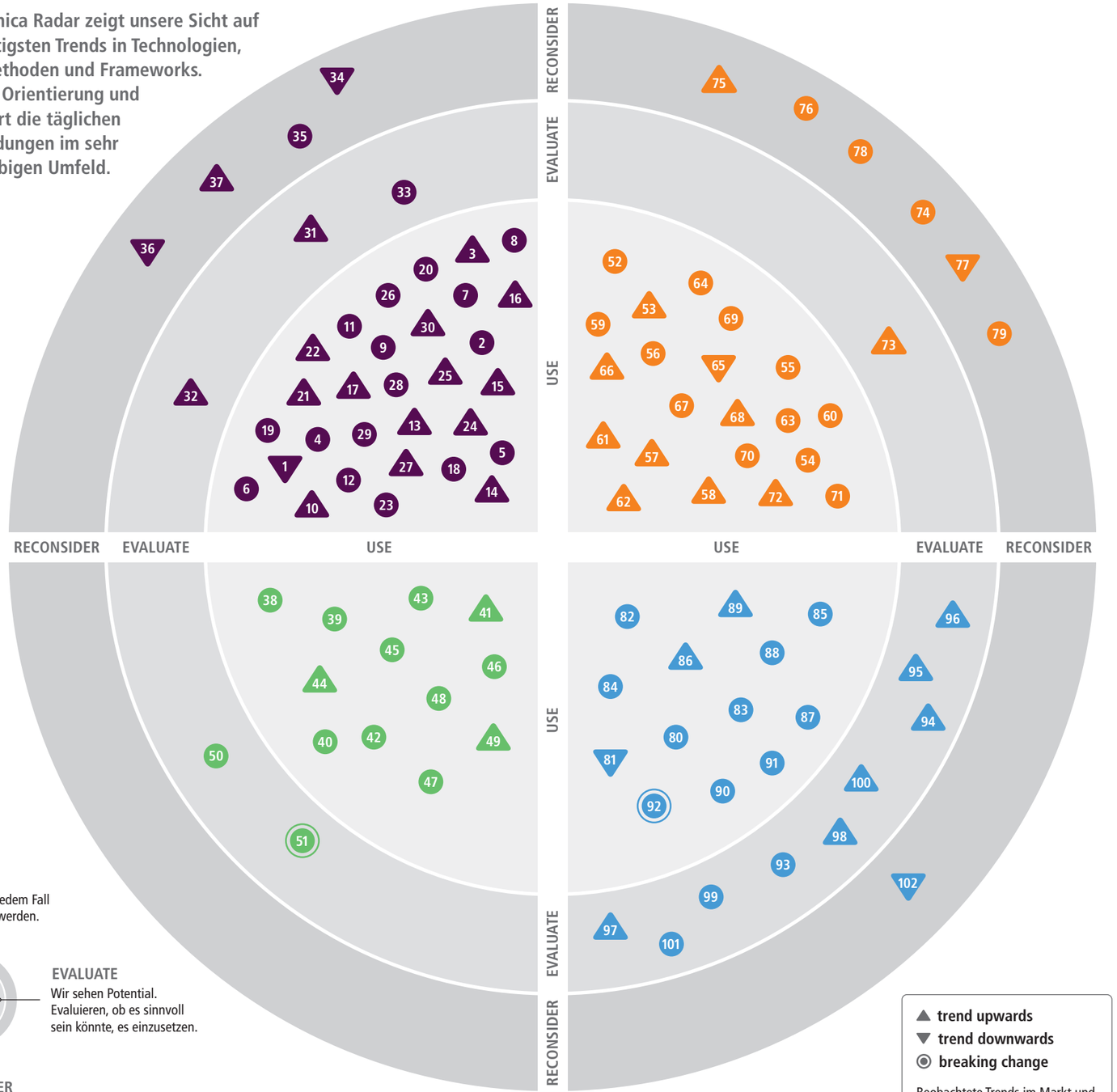
TECHNICA RADAR – VOL. 2

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 (UI) Pattern Library ▼
2 Accessibility (Barrierefreiheit)
3 Adaptive Design ▲
4 Advanced Analytics and Machine Learning
5 Clean Architecture / Ports & Adapters / Hexagonal Architecture / Onion Architecture
6 Clean Code
7 Content strategy
8 Conversational Interfaces
9 Customer Experience
10 Design System ▲
11 DevOps
12 Domain Driven Design
13 Edge Computing ▲
14 Emergent Architecture ▲
15 Event Sourcing ▲
16 Evolutionary Architecture ▲
17 Functional Programming ▲
18 Gamification
19 Hybrid Cloud
20 Information Architecture
21 Internet of Things ▲
22 IT Security ▲
23 OPC Unified Architecture
24 Product Life Cycle Based Development ▲
25 Progressive Web Apps (PWA) ▲
26 Representational State Transfer (ReST)
27 Security für Embedded Devices ▲
28 Self-contained Systems
29 Serverless Architecture
30 Web Components ▲

EVALUATE
31 Crowdfunding ▲
32 Mixed Reality ▲
33 WebAssembly

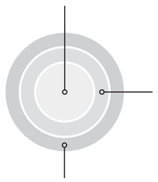
RECONSIDER
34 Chat-Bots ▼
35 Microservices
36 Responsive Design ▼
37 Strategy engineered ▲

Tools

USE
38 AWS
39 Azure DevOps
40 Cloud Computing
41 Dependency Vulnerability Scanning ▲
42 Docker
43 Figma
44 Infrastructure as Code ▲
45 Kubernetes
46 Microsoft Azure
47 Monitoring
48 OpenShift
49 Software as a Service (SaaS) ▲

EVALUATE
50 Cognitive Services
51 Windows 11 ◉

USE
Dies kann in jedem Fall angewendet werden.



EVALUATE
Wir sehen Potential. 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
52 Acceptance Test Driven Development (ATDD) / Behavior Driven Development (BDD)
53 Arc42 ▲
54 Architecture Tradeoff Analysis Method (ATAM)
55 Continuous Delivery
56 Continuous Refactoring / Refactor Mercilessly
57 Cost of Delay ▲
58 Event Storming ▲
59 Kanban
60 Large Scale Scrum (LeSS)
61 Liberating Structures ▲
62 Mob (Ensemble) Programming ▲
63 Pair Programming
64 Risk Management
65 Scrum ▼
66 Systems Thinking ▲
67 Service Design
68 Security engineering ▲
69 Test Driven Development (TDD)
70 Usability Test
71 User Research
72 User Centered Design ▲

EVALUATE
73 Ethical OS ▲

RECONSIDER
74 Burn down / up chart
75 Democratizing Programming ▲
76 Design Thinking
77 Effort Estimation ▼
78 Scaled Agile Framework (SAFe)
79 UML

Libraries, Frameworks, Programming Languages

USE
80 .NET 5/6
81 .NET Standard ▼
82 Angular
83 C++
84 Cumulocity IoT
85 Graph database
86 gRPC ▲
87 Java
88 Node.js
89 Python ▲
90 React
91 Spring Framework
92 xBehave.net ◉

EVALUATE
93 Azure IoT Edge
94 Blazor ▲
95 dapr ▲
96 F# ▲
97 Flutter ▲
98 GO Programming Language ▲
99 Kotlin
100 R ▲
101 Rust

RECONSIDER
102 .NET Framework (Full-Framework NICHT Core) ▼

Die Analyse der wichtigsten Trends

▲ trend upwards ▼ trend downwards ◉ breaking change

Concept or theme

USE

1 (UI) Pattern Library ▼
Pattern Libraries funktionieren insbesondere für einzelne Produkte mit geringer Komplexität und bei denen wenige Veränderungen vorgenommen werden. In einer Zeit, in der Dienste und Produkte zunehmend über mehrere Kanäle zugänglich sind und permanent weiterentwickelt werden, sind Pattern Libraries zu starr und in der Weiterentwicklung eingeschränkt. Pattern Libraries werden zunehmend durch Design Systeme abgelöst.

3 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.

10 Design System ▲
Ein Design System ist ein Regelwerk, um Elemente und Strukturen von Benutzeroberflächen zu konzipieren. Dabei werden Verhalten, Aussehen und Kombinierbarkeit von Attributen festgelegt, so dass das Design eines neuen Elements einer Benutzeroberfläche aus den Regeln abgeleitet werden kann. Dies ermöglicht es Teams, neue Elemente zu entwickeln, ohne die Konsistenz der Benutzerführung zu gefährden oder die Design-Prinzipien zu verletzen.

13 Edge Computing ▲
Edge Computing ist ein Lösungspattern, das Logik, Sicherheits- und Datenschutzaspekte an den Rand zur Cloud-Lösung verlagert. Zusätzlich eignet sich das Pattern, um das Datenvolumen in der Kommunikation mit der Cloud zu steuern. Im Kontext von IoT-Projekten ist Edge Computing nützlich, um Daten einzelner Gerätegruppen zusammenzufassen und die Kommunikation zu strukturieren.

14 Emergent Architecture ▲
In einer komplexen Umgebung ist es unmöglich eine Architektur zu finalisieren, 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.

16 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.

17 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.

21 Internet of Things ▲
IoT hat sich etabliert und ist zum Teil des IT-Lösungsraumes geworden. IoT wächst stetig und entwickelt sich weiter.

22 IT Security ▲
In Zeiten, in denen eine effektive Ransomware-Software für ein paar hundert Dollar im Darknet zu erwerben ist, ist das Thema IT-Security wichtiger denn je. Diverse Fälle in den Medien bestätigen die reale Bedrohung dieser Angriffe. Es gilt, sein Sicherheitsnetz gesamtheitlich zu überdenken, von Netzwerksegmentierung über die Passwortstrategie (keine universalen Passwörter) bis hin zur 2-Faktor-Authentisierung.

24 Product Life Cycle Based Development ▲
Wie in der klassischen Industrie, 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.

25 Progressive Web Apps (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.

27 Security für Embedded Devices ▲
Integrierte Systeme verlangen, dass immer mehr Embedded Devices miteinander vernetzt sind. Dies führt zu neuen potenziellen Sicherheitsrisiken. Die Konnektivität der Geräte erfordert eine sichere Verbindung zum Datenaustausch der Geräte untereinander.

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

EVALUATE

31 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.

RECONSIDER

37 Strategy engineered ▲
Die Technologie wird ein immer grösserer Treiber in der Gesamtstrategie eines Unternehmens. Heute sind Business und Technologie eng miteinander verwoben. Damit eine realistische Strategie erarbeitet werden kann, sollte eine enge Zusammenarbeit zwischen Business und Technologie etabliert werden. Beide Seiten müssen sich besser verstehen und mehr aufeinander eingehen. Daher sollte die Person mit Technologie-Verantwortung aktiv in die Strategieentwicklung miteinbezogen werden. Die Entwicklung einer Strategie wird so zu einem gemeinsamen und anhaltend iterativen Prozess.

Tools

USE

41 Dependency Vulnerability Scanning ▲
Mit jeder Abhängigkeit besteht die Gefahr, dass eine Sicherheitslücke in einer Software eingebunden wird. Der erste wichtige Schritt ist, die Abhängigkeiten immer auf dem aktuellen Stand zu halten. Der zweite Schritt ist, ist, mögliche Sicherheitslücken automatisiert zu erkennen. In vielen Ökosystemen gehört das zum Standard und ist gut integriert. Im .NET wurde das leider oft vernachlässigt. Seit .NET 5 wird eine Überprüfung der Abhängigkeiten gegen potenzielle Sicherheitslücken als Standardfunktion angeboten. Wir empfehlen dringend, diese zu nutzen.

44 Infrastructure as Code (IaC) ▲
Infrastrukturen werden komplizierter. Um die Qualität aufrecht zu erhalten und Migrationen zu automatisieren, werden Infrastrukturen in maschinen-lesbarem Format definiert und versioniert. Ein tradeoff zwischen manuell und IaC sollte bedacht werden. Der IaC Lösung sollte der Vorzug gegeben werden.

Method or technique

USE

53 Arc42 ▲
ArchiArchitektur-Dokumentationen wurden leider in letzter Zeit vernachlässigt. Arc42 bietet ein Template für die Struktur und Hilfestellung zum Erstellen der Architektur-Dokumentation. Dadurch ist die Dokumentation formalisiert und bietet die Möglichkeit einzelne Aspekte der Architektur strukturiert zu diskutieren und zu bewerten. Es liefert ein solides Grundgerüst in einem pragmatischen Ansatz, ist an individuelle Bedürfnisse flexibel anpassbar und lässt sich mit anderen Ansätzen wie z.B. dem C4 Modell kombinieren.

57 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.

58 Event Storming ▲
Machen Sie abstrakte Dinge für das gesamte Produktteam (Entwicklung, Buisness-Verantwortliche, Vertrieb, ...) fassbar, indem Sie zusammen die Geschäftsdomäne erforschen und verstehen. Gleichen Sie das verwendete Vokabular aneinander an, und definieren Sie gemeinsam verwendete Begriffe.

61 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.

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

65 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.

66 Systems Thinking ▲
System-Thinking ist sehr machtvoll für die Analyse und Konzeption von komplexen Systemen. Es bildet eine Brücke zwischen Businesslogik, 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.

72 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

73 Ethical OS ▲
Ethical OS ist ein Framework, das Startups und Produktentwicklern einen Fragekatalog an die Hand gibt, damit sie das Risiko möglicher moralischer Desaster ihrer Produkte minimieren. Diese Fragen betrachten Aspekte geplanter Produktfeatures, die das Produkt in einen moralisch fragwürdigen Kontext bringen. Beispiel: Das Feature zur Ortung eines verlorenen Mobiltelefons wird schlussendlich zur Überwachung einer Person eingesetzt.

RECONSIDER

75 Democratizing Programming ▲
Democratizing Programming und Low Code sind eine Möglichkeit, ohne Studium oder Ausbildung in Informatik die Business-Logik einer Applikation in Teilen zu automatisieren oder gesamte Applikationen zu entwickeln. Democratizing Programming ersetzt keine klassische IT-Entwicklung und sollte zusammen mit der Softwareentwicklung im Unternehmen eingeführt werden, um keine falschen Erwartungen zu wecken und Missverständnisse zu vermeiden. Dieses Konzept kann dabei unterstützen, Softwareentwicklung und Business einander näher zu bringen.

76 Design Thinking
Immer mehr Unternehmen verfügen über Innovationsabteilungen, die gute 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.

77 Effort Estimation ▼
In einem komplexen Umfeld ist vieles nicht voraussehbar. Daher ist eine zuverlässige Schätzung über einen längeren Zeitraum unmöglich. Anstelle von vermeintlich genauen Plänen und der Frage «Wie aufwändig ist das?» sollten Teams eher den Fokus auf «Was gewinnen wir?» legen und kontinuierlich an Produktinkrementen arbeiten, die den bestmöglichen Wert generieren.

Libraries, Frameworks, Programming Languages

USE

81 .NET Standard ▼
Der .NET Standard ist die formale Spezifikation von .NET API, welche in mehreren .NET Implementationen verfügbar sind. Von Microsoft wird empfohlen, net5.0 als Zielframework zu verwenden. Der .NET Standard wird nicht als «deprecated» bezeichnet und wird immer noch eingesetzt, wenn der Code mit dem .NET Framework oder mit .NET Core 3.x geteilt werden soll.

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

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

92 xBehave.net ◉
xBehave.net ist eine der Bibliotheken, die wie BDD Tests in .NET geschrieben werden. xBehave wird nicht mehr aktiv gepflegt und sollte daher nur noch mit Vorsicht verwendet werden, beziehungsweise eine Alternative gesucht werden. Bitte unterstützt eure eingesetzten OSS-Komponenten, wir tun es leider auch noch nicht.

EVALUATE

94 Blazor ▲
Eine in das .NET-Ökosystem integrierte Web-Entwicklungsumgebung. Der Pluspunkt: Bessere Zugänglichkeit der Webentwicklung für .NET-Entwickler.

95 Dapr ▲
Dapr verspricht als Distributed Application Runtime Entwickler, sich auf die Business-Logik zu fokussieren und die Querschnittsthemen delegieren zu können, wie die Anbindung und Kommunikation mit anderen Systemen. Wir sehen Analogien zu Serverless Computing, daher könnte Dapr eine gute Grundlage für zukünftige verteilte Systeme bieten.

96 F# ▲
Die prägnante, robuste «functional-first» Sprache F# hat eine viel kleinere Sprachspezifikation als C# und ist damit wohl einfacher zu erlernen und einzusetzen. Trotzdem ist sie als Vorreiter oft die Quelle für die neueren Sprachfeatures von C#, auch wenn sie nicht immer gleichwertig umgesetzt werden. F# kann für die Entwicklung eines Frontends, eines Backends oder bis zur Infrastruktur eingesetzt werden und genoss auch von Microsoft an der .NET Conf 2021 mit dem Thema «Focus on F#» mehr Aufmerksamkeit.

97 Flutter ▲
Die Maturität von Flutter wächst. Seit 2.0 können Applikationen für das Web entwickelt werden. Es vereinfacht die Entwicklung von Services über mehrere Plattformen hinweg.

100 R ▲
Die Themen Data Science und Machine Learning sind mittlerweile in der Entwicklung von digitalen Produkten sehr verbreitet. Im Zuge dessen gewinnt die Programmiersprache R für die statistische Auswertung und Verarbeitung von Daten an Bedeutung.