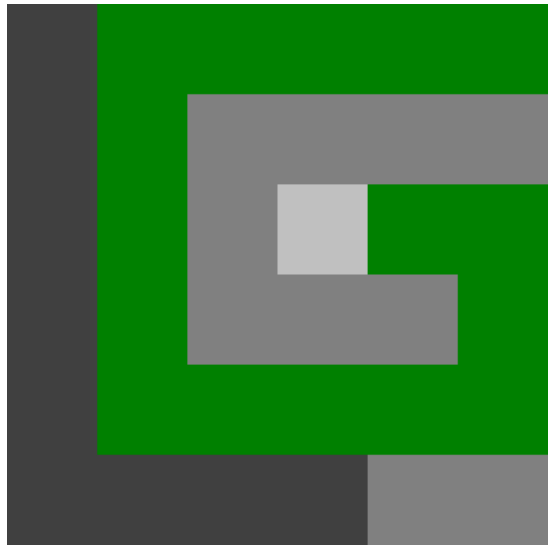
TRENDS IN DER IT

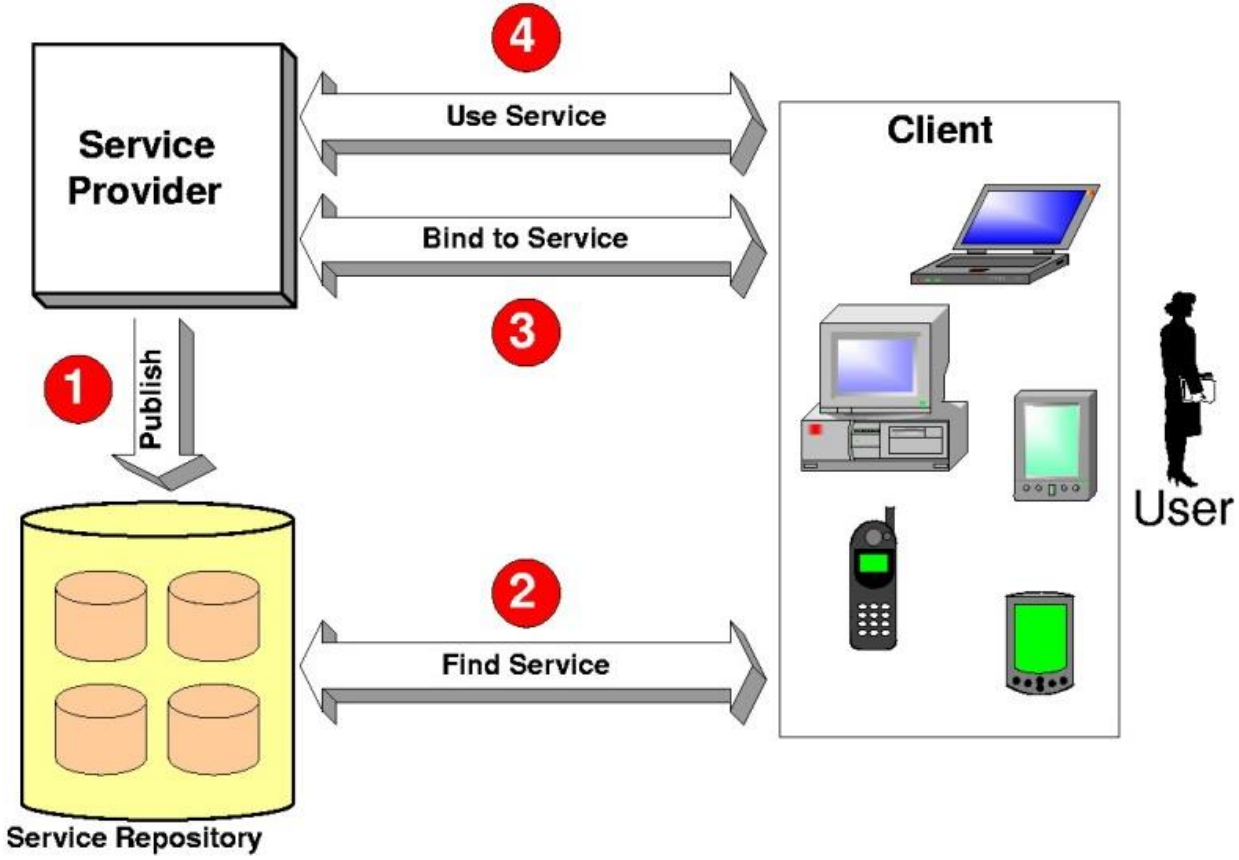


Kapitel 7
Neues im Bereich Softwareentwicklung und OO-Technik

Das Update für Experten

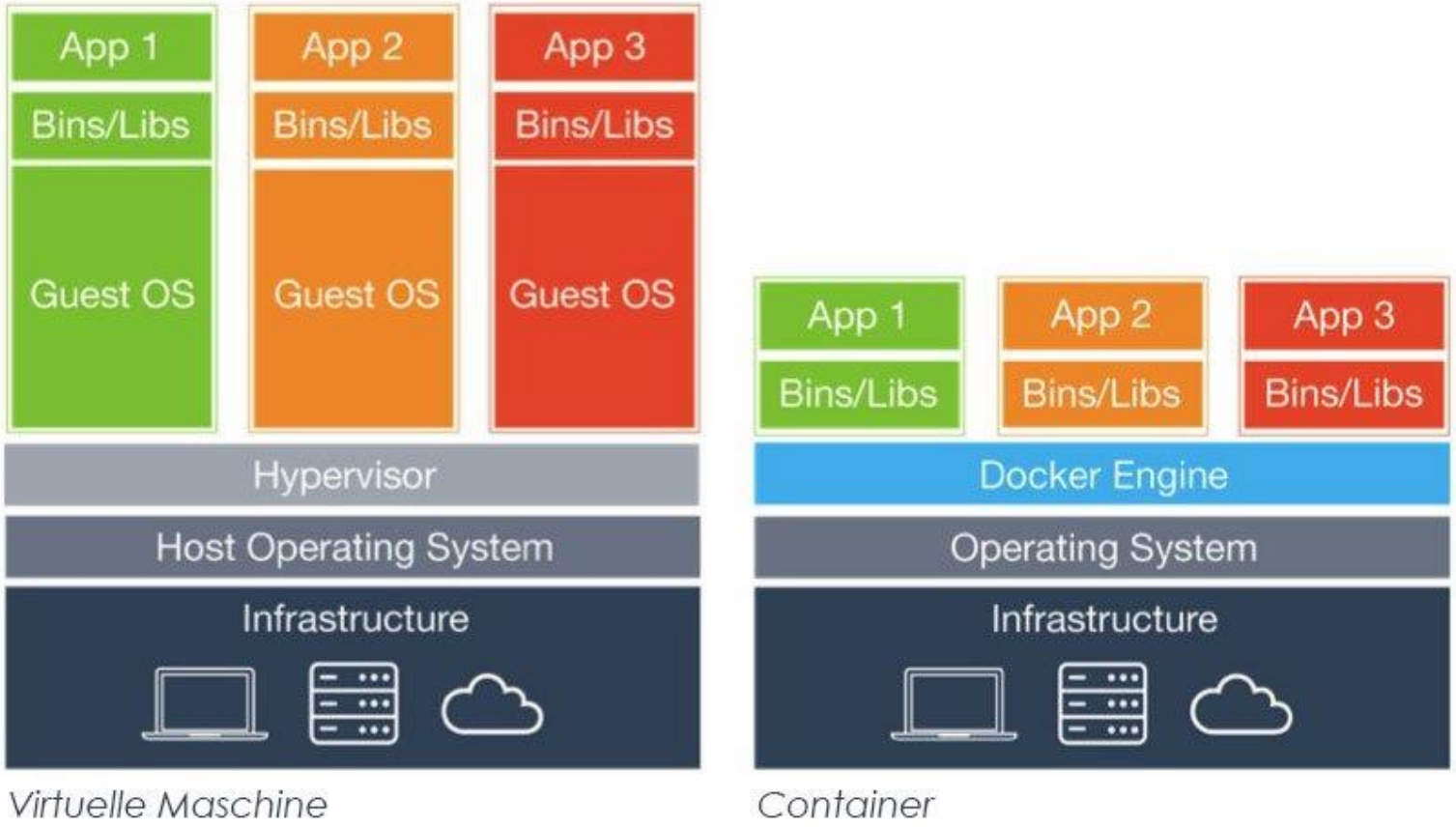
Web Services

- Verbraucher müssen Web Services suchen und finden können
- Ein Service muss seine Schnittstelle (Parameter und Protokoll) beschreiben, sodass er aufgerufen werden kann
- Ein Service muss die erlaubten Interaktionen beschreiben



Container-Technologie

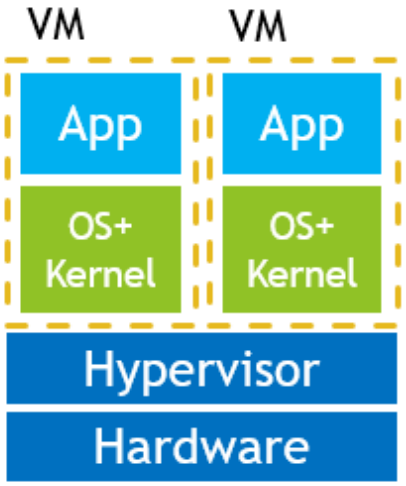
Betriebssystem



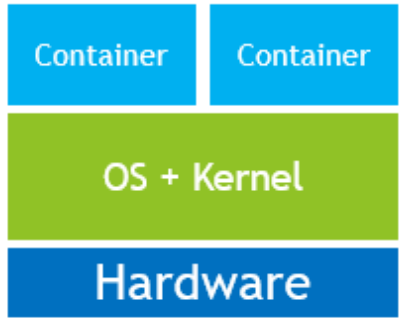
Container

Virtual Maschines

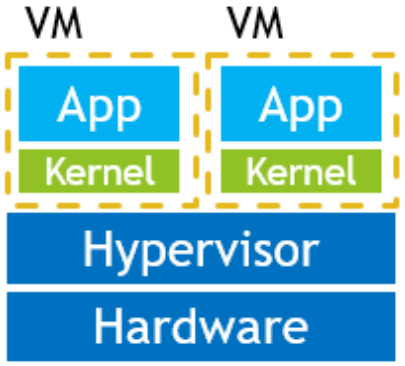
Unikernels



Virtual Machines



Linux Containers



Unikernels

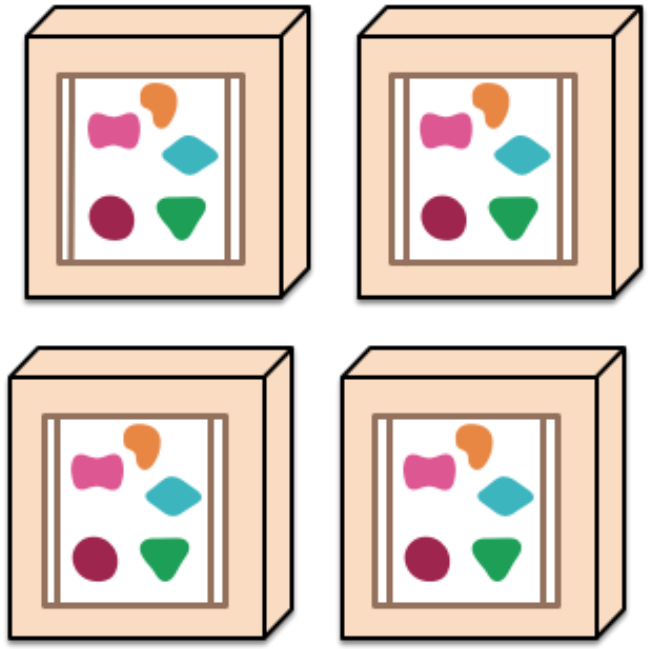
Monoliths and Microservices

Die wesentliche Eigenschaft von Microservices ist das unabhängige Deployment.

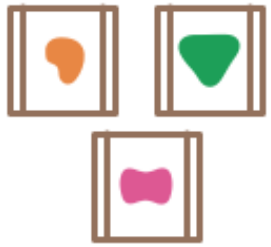
Kaum ein Thema ist im Moment so ein Hype wie Microservices.

A monolithic application puts all its functionality into a single process...

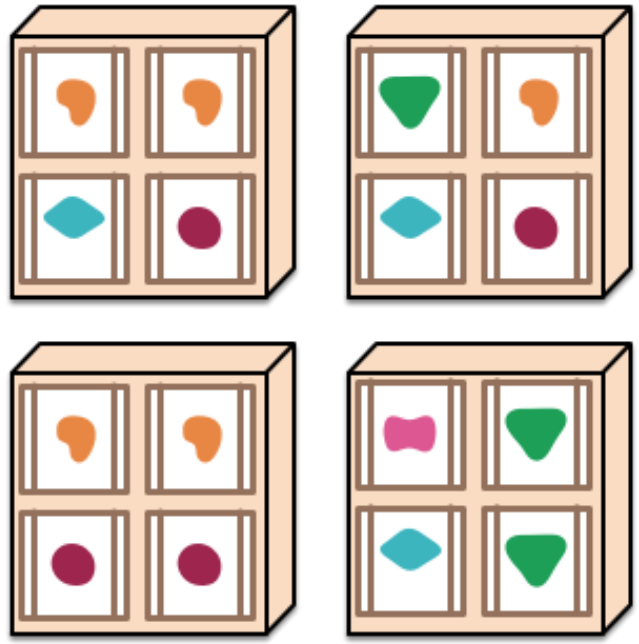
... and scales by replicating the monolith on multiple servers



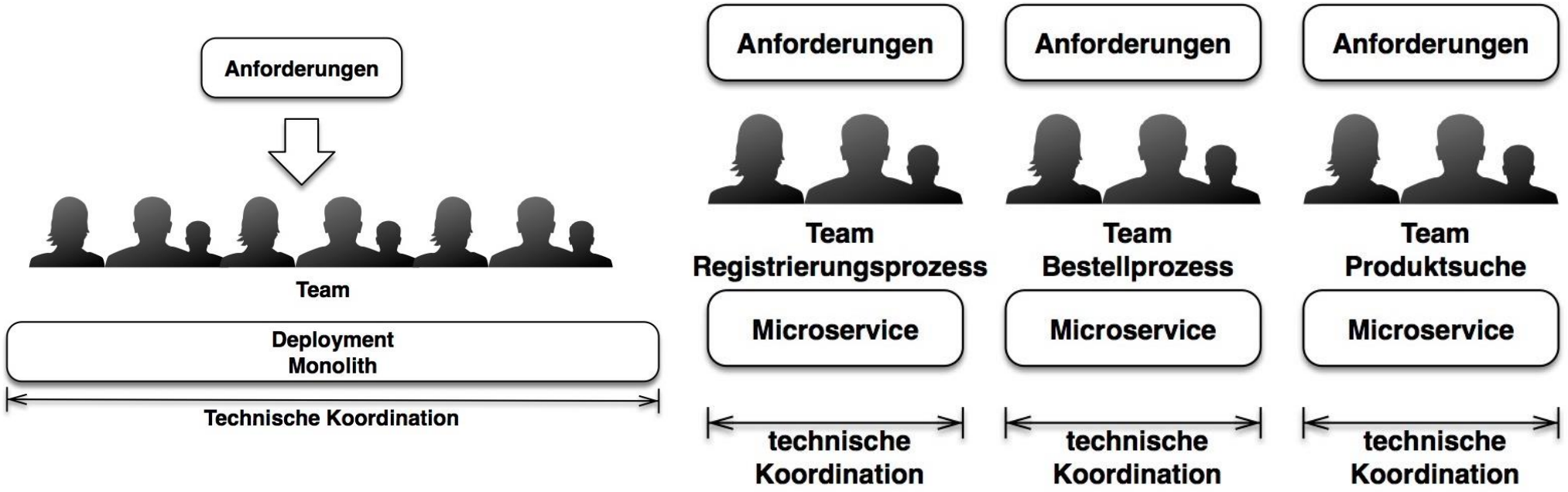
A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



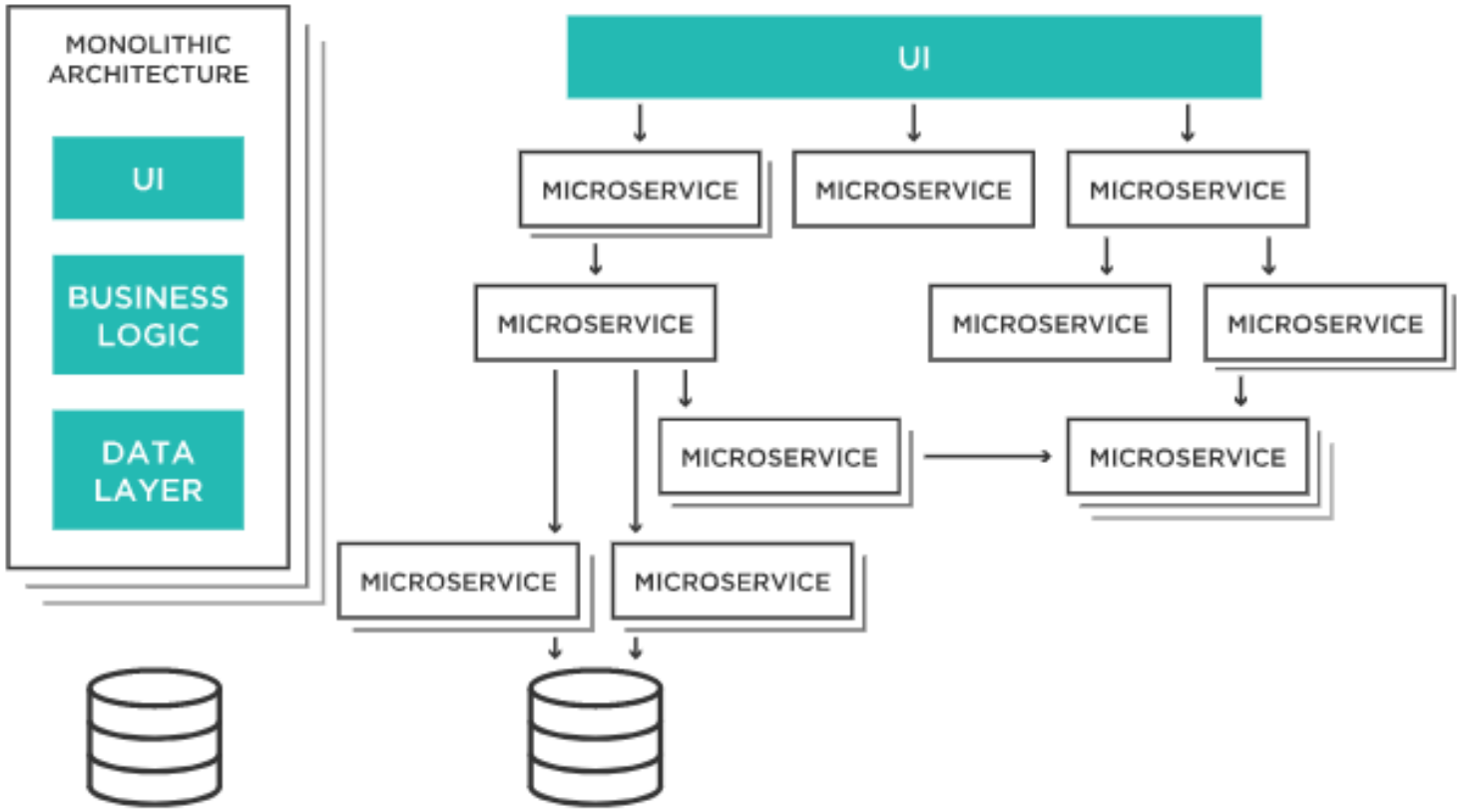
Monoliths and Microservices



Der Erfolg von Microservices ist auch gleichzeitig ein Problem.

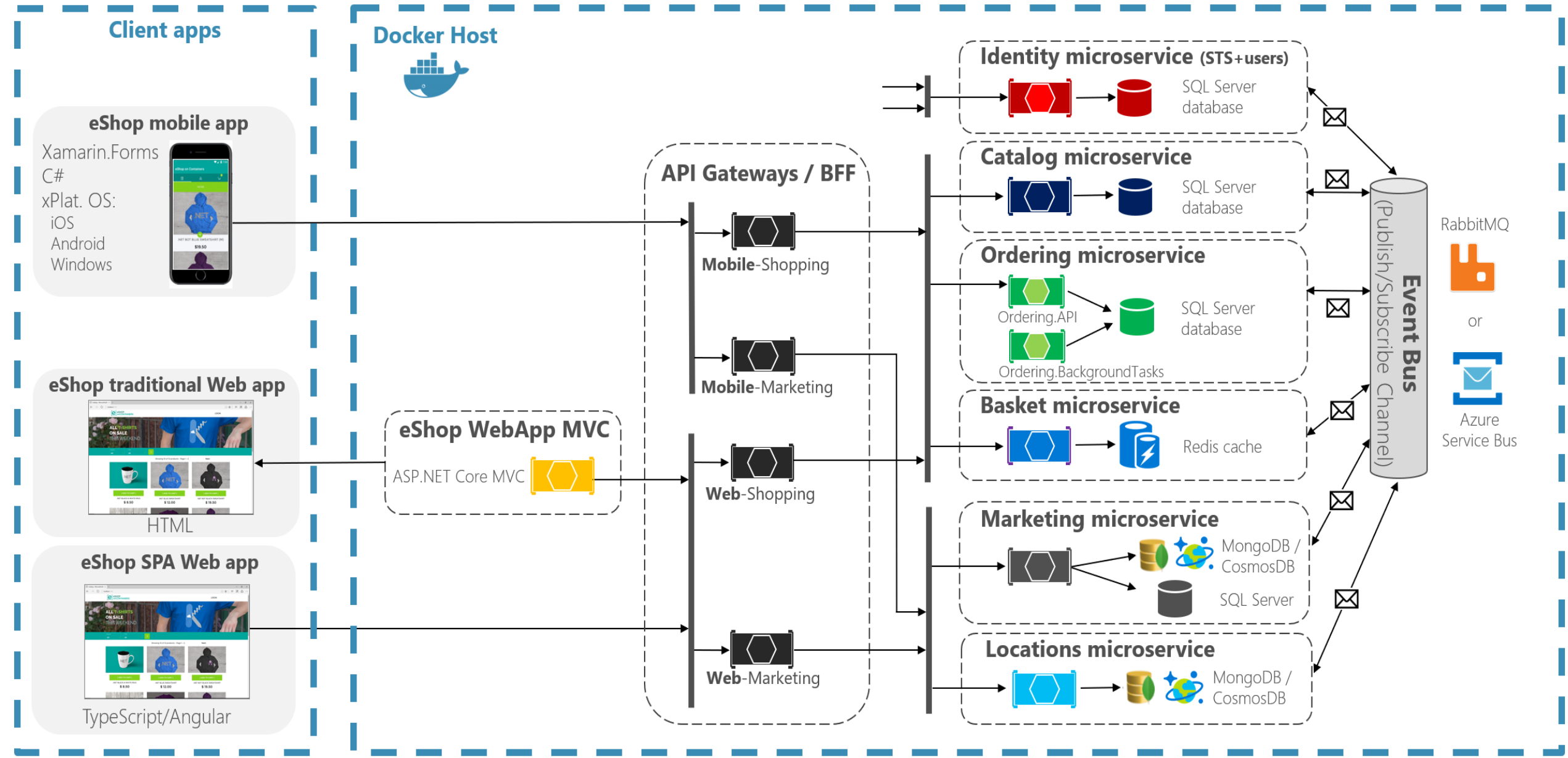
Container

und Microservice



eShopOnContainers reference application

(Development environment architecture)



Vorteile von Microservices (Quelle: wikipedia.org – 2017/11)

- Weil Microservices unabhängig voneinander verteilt und entwickelt werden können, können **Teams unabhängig voneinander arbeiten**. Das ermöglicht die **Skalierung agiler Entwicklungs-Prozesse**, ohne viel Kommunikations- und Koordinationsaufwand zu erzeugen.
- Microservices sind klein. Dadurch bleiben sie **übersichtlich und leicht weiterentwickelbar**. Bei Bedarf können sie, mit kleinem bis überschaubaren Aufwand, durch eine Neuimplementierung ersetzt werden.
- Oft schleichen sich bei Systemen ungewollte Abhängigkeiten ein und irgendwann geht die ursprüngliche Architektur vollständig verloren. Die Architektur des Microservices-Systems bleibt stabil, weil Abhängigkeiten zwischen Microservices über die API eingeführt werden müssen. Das ist aufwändig und passiert nicht aus Versehen.
- Weil die Microservices wartbar bleiben und auch die Architektur des Gesamtsystems erhalten bleibt, erlauben Microservices auch langfristig eine produktive Entwicklung des Systems.
- Microservices können **unabhängig voneinander skaliert** werden.
- Microservice-Systeme können gegen den **Ausfall** anderer Services abgesichert werden, so dass das Gesamtsystem robust ist.
- Continuous Delivery ist aufgrund der Größe der Microservices einfacher.
- Jeder Microservice kann mit einer anderen Technologie implementiert werden. Das vereinfacht Experimente mit neuen Technologien und verhindert das Veralten des Technologie-Stacks.
- Microservices können auch dazu genutzt werden, um **Legacy-Systeme zu erweitern**, ohne dabei zu viele Änderungen an der alten Code-Basis vornehmen zu müssen.
- Wenn Schlüsseldienste identifiziert wurden, können im Falle einer Überlastung unkritische Services reduziert oder abgeschaltet werden, um Ressourcen für kritische Services frei zu machen.

Nachteile von Microservices (Quelle: wikipedia.org – 2017/11)

- Die verteilte Architektur erzeugt **zusätzliche Komplexität**, vor allem Netzwerk-Latenzen, Lastverteilung oder Fehlertoleranz (siehe dazu auch: Fallacies of Distributed Computing).
- Da es mehr Systeme gibt die ausfallen können als bei monolithischen Services, steigt auch die Wahrscheinlichkeit, dass mindestens eine Komponente ausfällt. Gibt es beispielsweise 10 Services mit jeweils 99,9 % Ausfallsicherheit, so besteht für das **Gesamtsystem nur noch eine Ausfallsicherheit von $(99{,}9\ \%)^{10} = 99{,}0\ \%$**.
- Die Vielzahl an Services macht die Softwareverteilung und das Testen **komplexer**.
- Der Aufwand für die Migration bestehender Systeme ist beträchtlich und bedeutet in der Regel auch eine Anpassung der Kommunikationskultur in den beteiligten Organisationen.
- Das **Logging und Monitoring wird komplexer**, da mehrere Systeme involviert sind, welche ggf. unterschiedliche Logging- und Monitoringtechnologien einsetzen. Es sollten daher, zusätzlich zu dezentralen Logging- und Monitoringlösungen, zentrale Logging-, Monitoring- und OpsDB-Dienste eingesetzt werden.
- Da es sich um ein potenziell weltweit verteiltes System handelt, müssen nicht nur unterschiedliche Zeitzonen der Client-Anwendungen, sondern auch unterschiedliche Zeitzonen der Hosts berücksichtigt werden. Eine Zeitsynchronisierung zwischen den Hosts (z. B. mittels NTP oder noch besser PTP und die Verwendung passender Zeit-Bibliotheken (z. B. Joda Time oder Noda Time) wird damit zwingend notwendig.
- Da es sich bei Microservices um eine verteilte Architektur handelt, muss aufgrund des CAP-Theorems zwischen **Verfügbarkeit der Anwendung und der Datenkonsistenz** gewählt werden. Dem steht allerdings gegenüber, dass ein monolithischer Service im Fehlerfall, etwa bei einer Überlastung, ebenfalls nicht immer verfügbar ist. Zudem kann für Daten, sobald sie dem Nutzer angezeigt wurden, ebenfalls keine Konsistenz garantiertwerden.
- Da die Services in unterschiedlichen Programmiersprachen und Software-Stacks implementiert werden können, erhöhen sich die Anforderungen an die **Entwicklungswerkzeuge** und das Plattform-Management. Zudem muss die Funktionalität von Bibliotheken teilweise dupliziert werden.

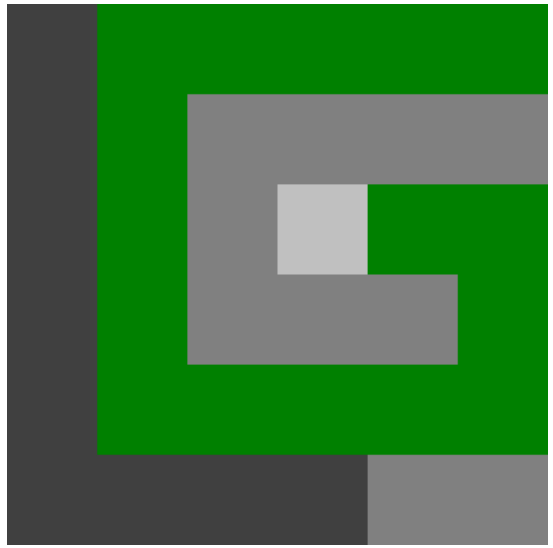
Stand: Open Source (OS)

- Im Office verloren!

- Betriebssysteme oder Teile vom Betriebssystem
  - Microsoft integriert immer mehr OS-Standards (XML, Docker, Protokolle usw.)
  - Android, Standard-Linux
- Datenbank (Internet-DB und kleine/mittlere Lösungen)
- Tools (Browser, Zip, PDF, Monitoring, Tuning, Grafik, Sicherheit usw.)
- Fachapplikationen

- Internetplattformen (Wikipedia)?

TRENDS IN DER IT



Objektorientierung und Datenbanken

Das Update für Experten

Datenbank-Technologien

- Netzwerk-Datenbanken
- hierarchische Datenbanken
- relationale Datenbanken

- objektrelationale Datenbanken
- objektorientierte Datenbanken

- mehrdimensionale Datenbanken
- HANA Datenbanken / In-Memory-Datenbank
- native XML-Datenbanken

Anbieter von OODBMS

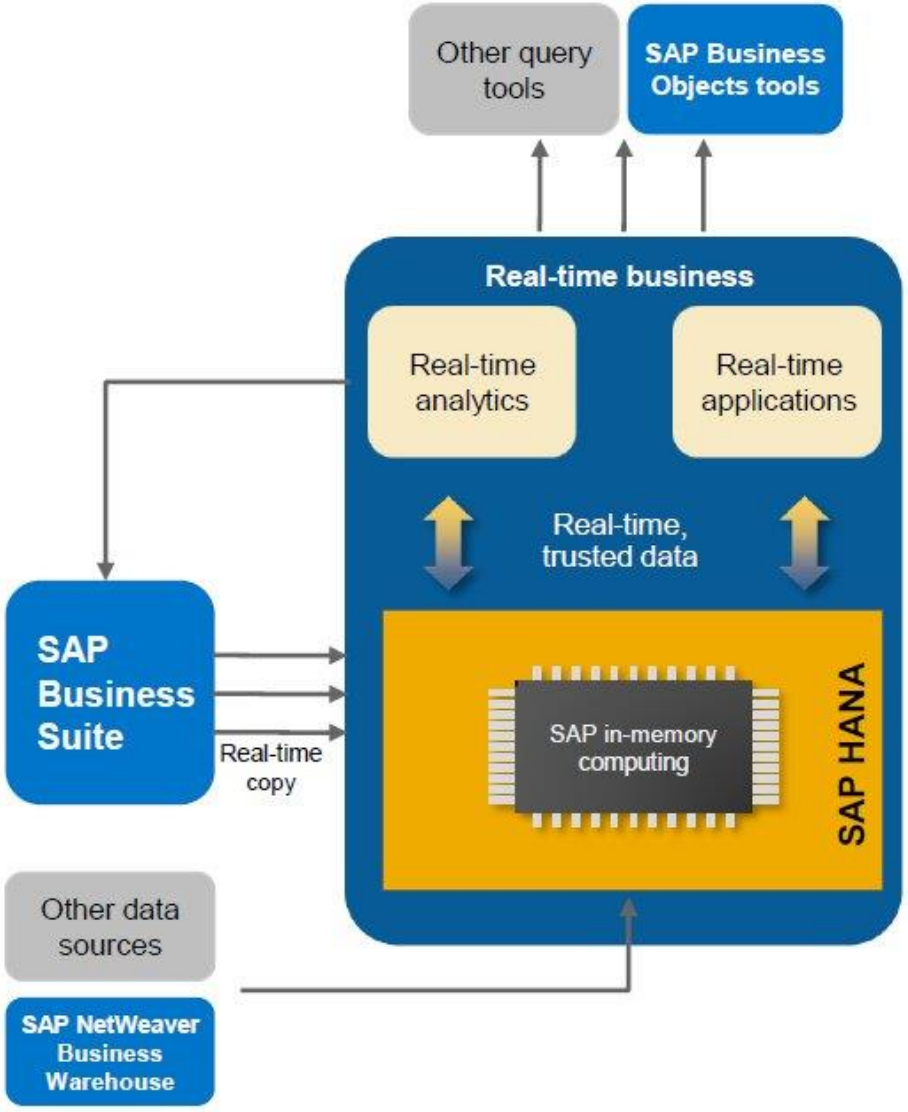
- POET
- Omniscience
- ObjectStore
- O2
- CA Jasmine
- J2EE Application Server

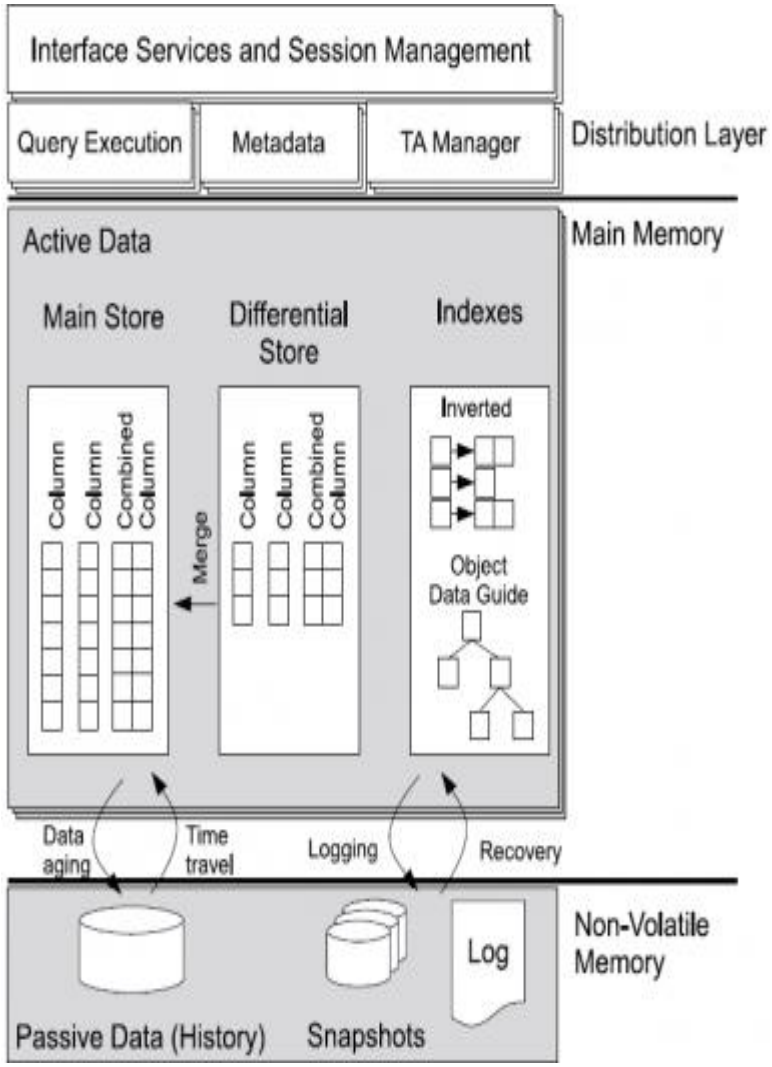
- Abfragesprache OQL

HANA

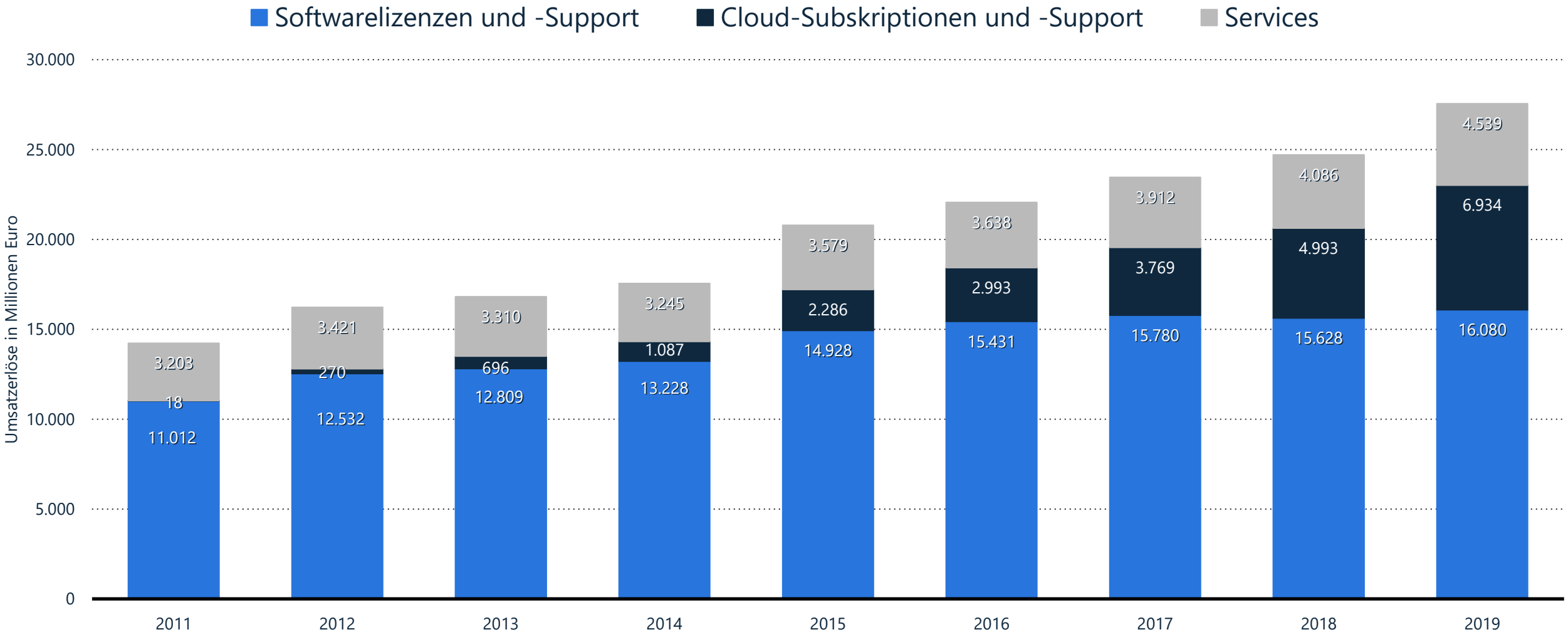
- In-Memory-Datenbank
- Appliance (Kombination aus Hardware und Software)
- Anfangs für
  - Business-Intelligence-
  - Business-Analytics-Anwendungen
- Seit 01/13 für die gesamte SAP Business Suite
- Die wichtigsten Vorteile:
  - Schnellere und mehrfach Durchführung von komplexe Prozesse wie Materialbedarfsplanungsläufe, Kapazitätsplanungen oder Simulationen
  - Durchführung von Analysen und Abfragen beliebiger Granularität, Aggregation und Dimension direkt und in Echtzeit
  - Verknüpfung aller SAP-Applikationen über ein DM-System
- Höhere Lizenz- und Hardwarekosten!

SAP HANA Appliance Architektur





Umsatz des Unternehmens SAP nach Segmenten in den Jahren 2011 bis 2019 (in Millionen Euro)



Quelle(n): SAP

Oracle Exalytics versus SAP HANA

- Oracle Exalytics abgestimmt auf Oracle BI Foundation Suite und Essbase
- SAP HANA technisch und funktionell wesentlich weiter als Oracle Exalytics (inkrementelle Deltabeladungen, transaktionalem Kontext …)
- Oracle Exalytics max. 1 Terabyte Arbeitsspeicher

- Weitere In-Memory-Werkzeuge sind von IBM, MicroStrategy, SAS bereits am Markt erschienen

Native XML-Datenbanken/Tamino

- Record Locking und Transaktionen problematisch, wenn mehrere Systeme zugreifen
- sich verändernde Zugriffe nicht in XML Query Language enthalten
- eher Dokumentenmanagement als Datenbank, dann aber vorteilhaft

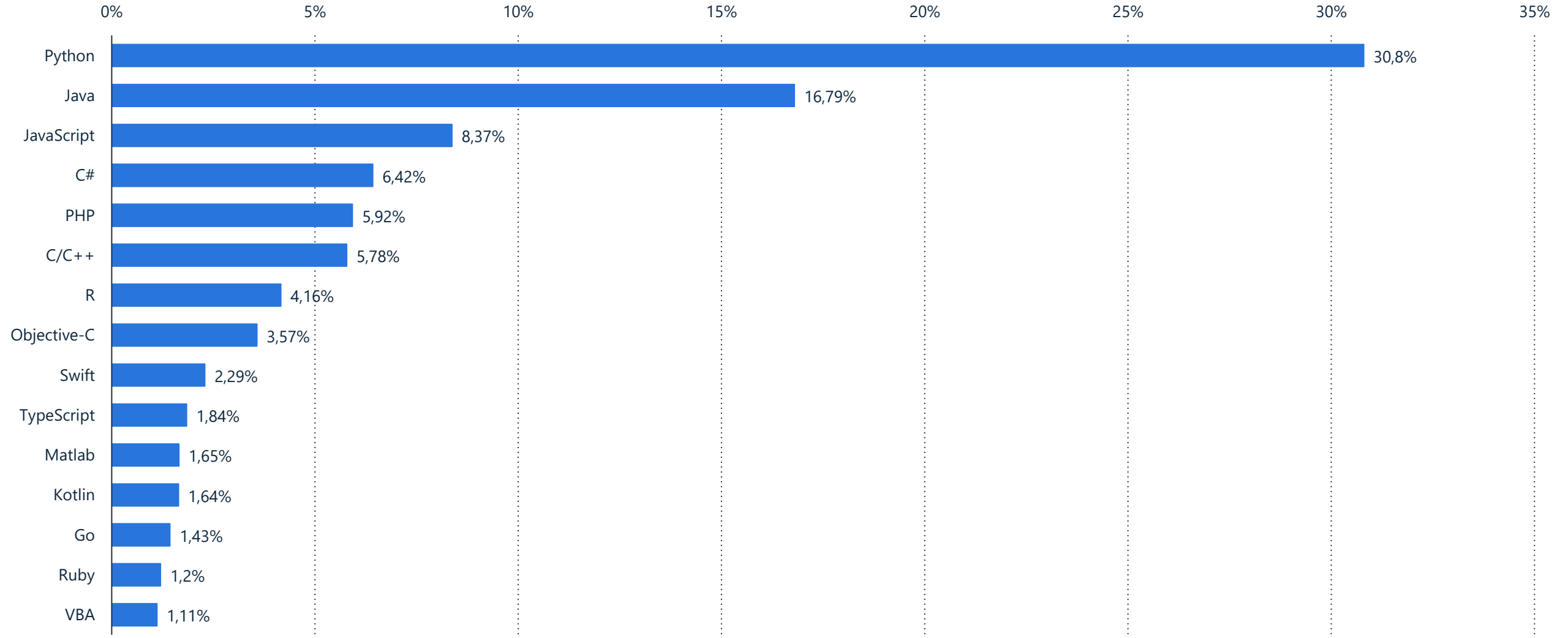
Entwicklung der Programmiersprachen

- C++
  - relativ stabil
- Visual Basic
  - fortschreitende Objektorientierung
  - mit .NET gleichmächtig zu C#
- C# und J#
  - sehr jung
- ABAP/4
  - fortschreitende Objektorientierung
- Smalltalk
  - Die erste ausgereifte OO-Sprache

JAVA

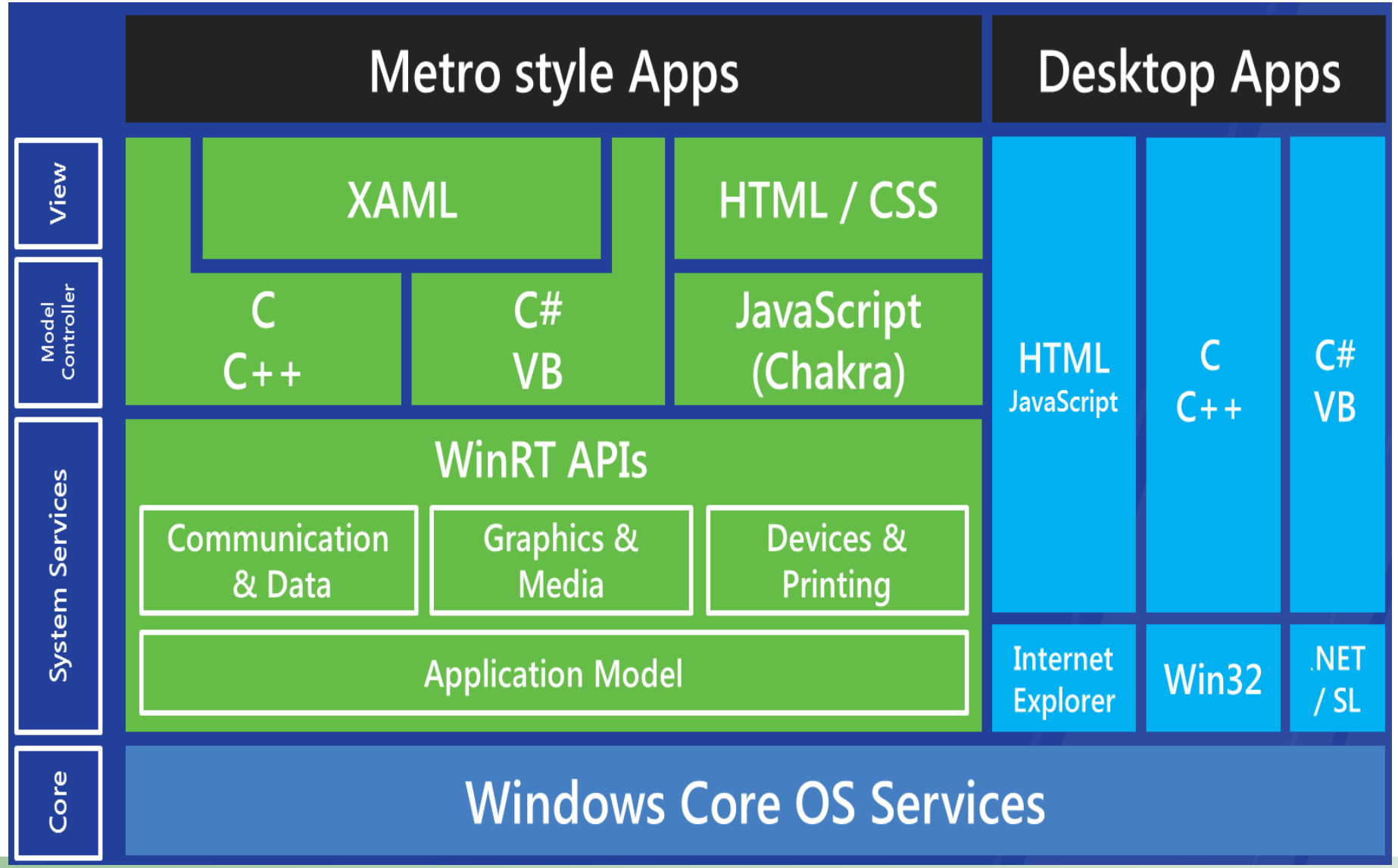
- Java-Webanwendungen
- Java-Desktop-Anwendungen
- Java-Applets for web
- Apps
- Compiler
  - Bytecode-Compiler
  - Native Compiler

Die beliebtesten Programmiersprachen weltweit laut PYPL-Index im November2020



Quelle(n): PYPL (PopularitY of Programming Language Index)

Windows 10 Plattform
HTML5 + JavaScript = .NET ade?



Eclipse

- Open Source
- Programmierwerkzeug zur Entwicklung von Software
- basiert auf Java-Technologie
- Für Eclipse gibt es zahlreiche Erweiterungen
- Eclipse unterstützt die Rich Client Plattform

- Die aktuelle Version ist 4.4 (Neon) – Juni-2016 (4.6.2 seit 22.12.2016)
  (Oxygen angekündigt für den 28. Juni 2017)

- Eclipse wird von allen große IT-Herstellern unterstützt und ist für die wichtigsten Systeme verfügbar.
- Alternativen: IntelliJ IDEA und NetBeans

MS Visual Studio 2017

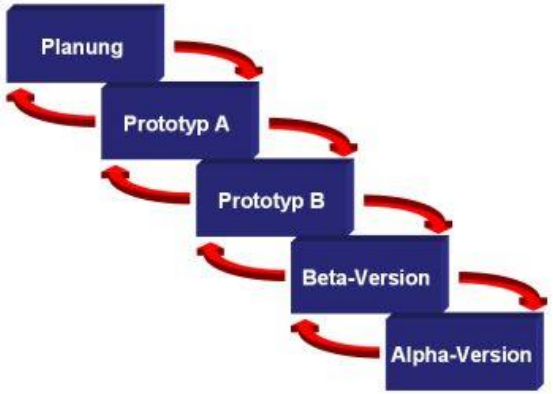
- Erstellen mehrschichtiger Anwendungen für Windows, das Web, SmartPhones und Pocket PCs
- integrierte Visual Database Tools für die Entwicklung von Datenbanken, Tabellen, gespeicherten Prozeduren und mehr
- integrierter Datenbankberichts-Designer und -Viewer
- Entwicklung, Debugging und Bereitstellung mehrschichtiger Anwendungen
- integriertes XSLT-Debugging
- neue Microsoft-Versionen

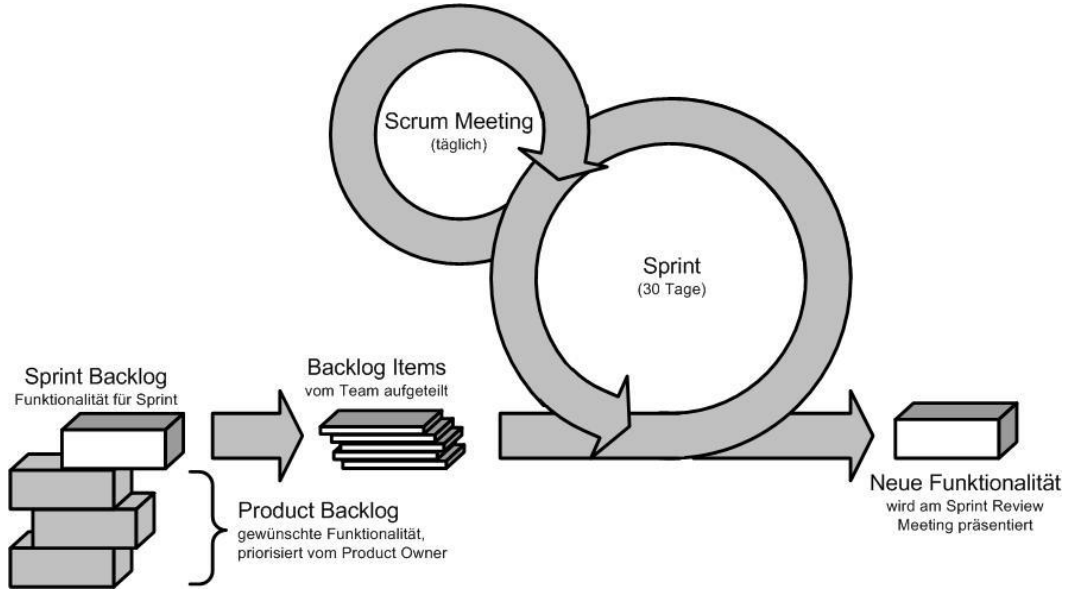
OO-Programmierung

- UML
  - Werkzeug zur Analyse und zum Design objektorientierter Anwendungen
- Design Patterns
  - Entwurfsmuster für verteilte Anwendungen
- eXtreme Programming (Agile Alliance…)
  - neue Ansatzmethode für Entwicklungsprojekte
  - fortlaufende Iterationen und den Einsatz mehrerer Einzelmethoden

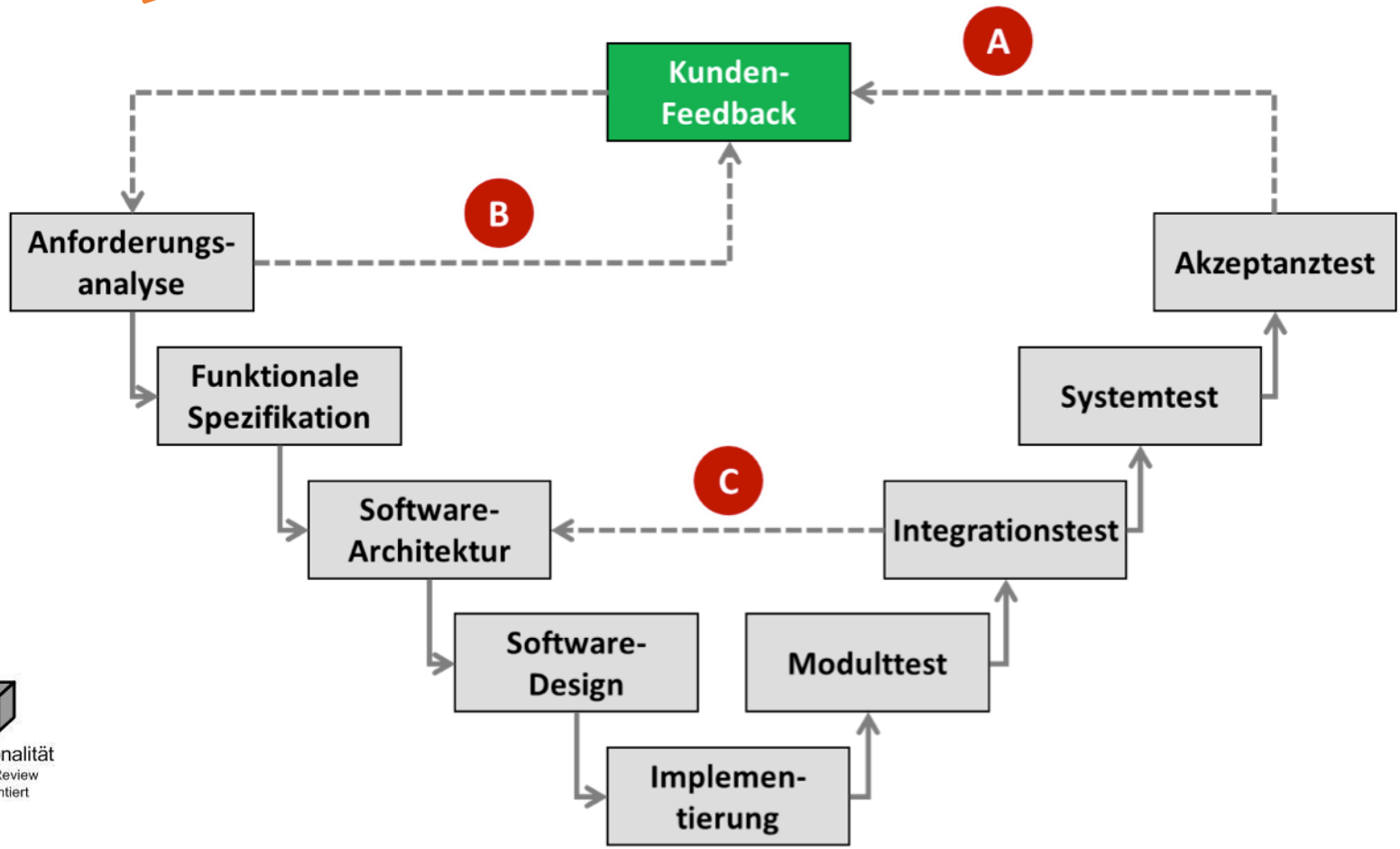
Agile Softwareentwicklung

- scrum
- kanban
- pmbok
- pmi
- …





Testing?



Agile Project Management