Reactive Programming

Programmierparadigma für antwortbereite und benutzerfreundliche Anwendungen

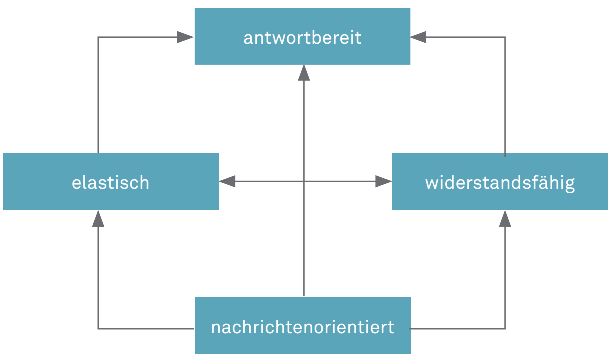
Reaktive Anwendungen sind anpassungsfähiger und in jeder Hinsicht skalierbarer als herkömmliche Anwendungen. Sie sind einfacher weiterzuentwickeln. Sie reagieren zuverlässiger und eleganter auf Fehler. Sie vermeiden hohe Ausfallkosten. Und sie stellen dem Benutzer durch ihre fortwährende Antwortfreudigkeit eine optimale User Experience bereit.

Definition

Das Programmierparadigma des Reactive Programmings beruht auf dem Reactive Manifesto. Das Manifest entstand aufgrund des Bedarfs nach benutzerfreundlichen, immer verfügbaren und schnell antwortenden Systemen.

Das Kernstück von Reactive Programming ist eine Stream-basierte Verarbeitung einer beliebigen Abfolge von unspezifischen Ereignissen. Dabei kann alles als Ereignis verstanden werden, sowohl Benutzereingaben oder abgeschlossene Datenübertragungen, als auch Änderungen an Variablen, Datenstrukturen, Caches und vieles weitere. Die Stream-basierte Denkweise erlaubt eine asynchrone Verarbeitung aller Ereignisse mit nur wenigen Codezeilen, bietet eine komfortable Fehlerbehandlung und unterstützt eine parallele Verarbeitung.

Reactive Programming besticht durch seine Flexibilität. So lässt es sich problemlos mit anderen Programmierparadigmen wie Asynchronous Programming, Functional Programming oder Dependency Injection koppeln.



Reactive Programming

Anforderungen
- Steigende User Experience
- Erhöhte Antwortzeiten
- Verteile IT-Landschaften
- Immer verfügbare Systeme

Programmierung
- Java, JavaScript, Python, Scala, .NET, Swift, Lisp, C, C++, Objective C, Go, ...
- Arbeit mit Streams reduziert Code und Komplexität, erlaubt parallele Verarbeitung und asynchrone Fehlerbehandlung

Programmierparadigmen
- Dependency Injection
- Functional Programming
- Asynchronous Programming

Schlagworte
- Reactive Manifesto
- Event-Driven Architecture
- Streams
- Actor Model
- Communicating Sequential Processes
- Vert.X
- Akka

Referenzszenario

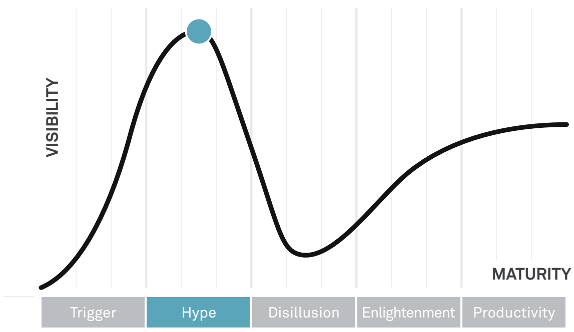
Ein Unternehmen möchte eine Anwendung kontinuierlich weiterentwickeln und deren Features kurzfristig erweitern, anpassen oder streichen können. Dabei sollen aktuelle Marktentwicklungen und Kundenwünsche innerhalb weniger Tage und Wochen berücksichtigt werden. Die ständige Verfügbarkeit der Anwendung sowie die Kundenzufriedenheit wurden als entscheidende Erfolgsfaktoren bestimmt. Mit der Entscheidung für das Programmierparadigma des Reactive Programmings schaffen die Architekten alle wesentlichen Voraussetzungen, um diese Ziele zu erreichen. Das Paradigma erfordert allerdings eine grundlegen Neuentwicklung der Anwendung, die dann wie ein Organismus wachsen kann. Das Unternehmen muss lediglich noch die Entwickler in Reactive Programming schulen.

Business Impact

Reaktive Anwendungen sind für den Anwender nicht als solche zu erkennen, haben durch ihre Antwortbereitschaft aber einen maximal positiven Einfluss auf die Benutzerzufriedenheit. Gesteigerte Benutzerzufriedenheit ist insbesondere im Rahmen der Digitalen Transformation entscheidend, um Kunden langfristig binden und sich gegen den Wettbewerb behaupten zu können. Gleichzeitig unterstützt die lose Koppelung nahezu sämtlicher Komponenten die schnelle Implementierung neuer Features.

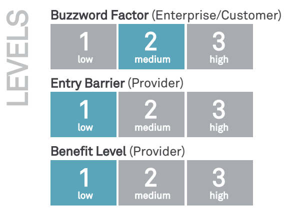
Reifegrad

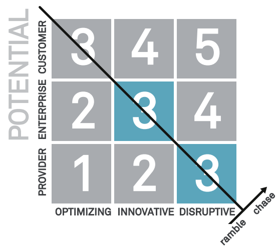
Reactive Programming auf Ebene von Anwendungskomponenten ist bereits langjährig etabliert, erlebt aktuell aber dennoch einen Hype aufgrund des Stream-Konzepts. Die Adaption dieses Trends auch auf Ebene von Anwendungen oder ganzen Systemlandschaften befindet sich mitten im Hype.



Marktübersicht

Reactice Programming ist überall (engl. ubiquitous) und in jeder Programmiersprache verfügbar. Meist setzen dedizierte Frameworks den Gedanken der Stream-basierten Verarbeitung für die verschiedenen Programmiersprachen um, etwa RxJS für JavaScript, RxJava für Java oder Rx.NET für C#. Es reicht allerdings nicht aus, nur ein passendes Framework zu wählen. Stattdessen muss sich der reaktive Ansatz durch die gesamte Anwendungsarchitektur ziehen und sogar Systemgrenzen überschreiten können. Unternehmen wie Airbnb, Netflix, Github und Trello setzen Reactive Programming erfolgreich in ihren Anwendungen um und profitieren von stabilen und hochverfügbaren Plattformen sowie sehr zufriedenen Kunden.





Alternativen

Benutzerfreundliche, hochverfügbare, fehlertolerante und performante Anwendungen lassen sich generell auch ohne Reactive Programming entwickeln. Allerdings lohnt es sich meistens nicht, eigene Ansätze zu erfinden. Denn das Risiko steigt, dass die grundsätzlichen Erwartungen an eine Anwendung hinsichtlich Benutzerfreundlichkeit und Verfügbarkeit nicht erfüllt werden können.

Pro	Contra
Geringe Einstiegshürde für Architekten und Entwickler	Wirtschaftlicher Vorteil nicht messbar
In allen Programmiersprachen umsetzbarr	Für den Kunden nicht direkt ersichtlich
Große Auswahl fertiger Frameworks	Viele alternative Programmierstile
Hohe Kundenzufriedenheit	Denkweise von Streams muss verstanden werden

https://msg.direct/techrefresh

Stand September 2017

msg systems ag
Robert-Bürkle-Straße 1 | 85737 Ismaning/München | Telefon: +49 89 96101-0 | Fax: +49 89 96101-1113 | www.msg.group | info@msg.group