## Was ist ein System?

**IEEE-Standard 610.12-1990**

|  |
| --- |
| A collection of components organized to accomplish a specific function or set of functions. |

Das charakterisiert recht treffend die wesentlichen Eigenschaften eines Systems, die auch intuitiv so zu erwarten sind. Ein System besteht aus Bausteinen bzw. Komponenten, wie z .B. Hardware-, Software- oder mechanischen Bausteinen.

Bereits in diesem Systembegriff findet sich die Vorstellung wieder, dass Systeme in Bausteine zerlegt werden können. Darüber hinaus soll entsprechend der Definition ein System einem bestimmten Zweck dienen. Dies spiegelt das Grundverständnis des Ingenieurwesens wider, für den Menschen nutzbringende Dinge zu erschaffen.

## Was ist ein Software intensives System?

|  |
| --- |
| **softwareintensives System** besteht aus einer Menge von Bausteinen, die so zusammengestellt sind, dass sie gemeinsam den Zweck des Systems erfüllen. Bausteine, die vollständig oder zu wesentlichen Teilen aus Software bestehen, übernehmen dabei essenzielle Aufgaben zur Erfüllung des Systemzwecks. Der Softwareanteil des Systems besteht dabei aus einer Menge von Programmen und weiteren Prozeduren und Daten sowie zugehöriger Dokumentation. |

Wie bereits dargestellt wurde, ist der Entwurf der Softwarearchitektur ein wichtiger und kritischer Schritt in der Softwareentwicklung. Die Softwarearchitektur hat unmittelbare Auswirkung auf die Parameter des magischen Vierecks: die gewünschte Funktionalität, die damit verbundenen Qualitätseigenschaften, der notwendige Aufwand zur Systemerstellung und die Zeitdauer bis zur Bereitstellung des Systems. Letztlich geht es um die Frage, wie große Systeme zu strukturieren sind, sodass die geforderten Parameter des magischen Vierecks erreicht werden können.

Aber hat denn jedes softwareintensive System überhaupt eine Softwarearchitektur? Ebenso wie auch Bauwerke, die ohne das Mitwirken eines Bauingenieurs entstanden sind, besitzt jedes softwareintensive System eine Architektur, auch wenn diese nicht explizit entworfen und umgesetzt wurde. Leider ist in Entwicklungsprojekten noch viel zu häufig zu beobachten, dass Softwarearchitektur nicht explizit entworfen wird. Die Konsequenzen sind oft folgenschwer.

Anforderungen an Software verändern sich während der Softwareentwicklung, aber auch über den ganzen Lebenszyklus hinweg, egal wie gut sie dokumentiert sind. Sich ändernde Anforderungen beeinflussen das Entwicklungsprojekt.

So müssen beispielsweise Projektpläne geändert oder das Budget muss angepasst werden. Dies alles läuft allerdings ins Leere, wenn sich die bereits realisierten Softwareanteile den gewünschten Änderungen widersetzen. Eine gute Softwarearchitektur allerdings ermöglicht das leichte Ändern von bestehenden oder das einfache Einbringen von neuen Funktionalitäten, ohne die Qualitätseigenschaften der existierenden Software zu gefährden.

Softwarearchitektur ist somit von enormer Bedeutung für eine erfolgreiche Softwareentwicklung. Worauf aber sind diese Zusammenhänge begründet? Diese Frage wollen wir anhand von zwei Teilfragen andiskutieren. Warum hat jede Software eine Architektur? Und warum ist die Softwarearchitektur mitentsche dend für eine erfolgreiche Softwareentwicklung?

Wesentlicher Bestandteil einer Softwarearchitektur ist die meist hierarchische Zerlegung des Systems in Teilsysteme oder Bausteine. Die Existenz einer solchen Zerlegungsstruktur ist auch bereits in der Natur von softwareintensiven Systemen enthalten. Wie in der Begriffsdefinition oben bereits aufgeführt, besteht ein softwareintensives System aus einer Menge von Bausteinen, die entsprechend organisiert

werden, um den Zweck des Systems zu erfüllen. Damit ist die Geburtsstunde der Architektur bereits mit dem Begriff eines Systems untrennbar verbunden. Somit hat jedes System und auch jedes softwareintensive System implizit oder explizit eine Architektur.

Diese inhärente Verzahnung, dass die Softwarearchitektur maßgeblich die Systemstruktur definiert und umgekehrt, ist auch der Grund dafür, dass Softwarearchitektur mitentscheidend für eine erfolgreiche, Softwareentwicklung ist. Die Struktur von Bauwerken legt fest, welche Teile tragende Bestandteile sind und welche nicht. Will man an einem Gebäude etwas verändern, ohne dass dabei tragende Bestandteile betroffen sind, ist das in der Regel problemlos möglich. Müssen allerdings tragende Bestandteile verändert werden, so ist nicht absehbar, ob und wie das realisierbar ist und welche Kosten dadurch entstehen.

Analoges gilt für die Softwarearchitektur: Sie definiert über die Festlegung der Systemstruktur die tragenden Elemente in der Software. Damit gibt die Softwarearchitektur den Rahmen für zukünftige Veränderungen vor. Ergeben sich im Laufe des Entwicklungsprojekts oder darüber hinaus im Lebenszyklus der Software Änderungs- oder Neuerungswünsche, dann ist das problemlos möglich, solange die tragenden Grundpfeiler der Software erhalten bleiben. Andernfalls gilt das Gleiche wie bei Bauwerken: Kosten, Zeit und resultierende Qualität sind nur schwer abzuschätzen und stehen in der Regel nicht in einem vernünftigen Kosten-Nutzen-Verhältnis.

Dieser banale, aber fundamentale Zusammenhang zwischen dem softwareintensiven System, der inhärent existierenden Softwarearchitektur und den sich daraus ergebenden Einschränkungen bezüglich des magischen Vierecks der Softwareentwicklung begründen die enorme Bedeutung und Tragweite der Softwarearchitektur im Rahmen der Softwareentwicklung.

## Bedeutung der Softwarearchitektur für ein softwareintensives System

Eine einzelne, allgemein akzeptierte Definition für Softwarearchitektur gibt es nicht. Zur Einstimmung in den Wildwuchs der Softwarearchitekturdefinitionen anbei eine der exotischeren Definitionen, die deshalb aber nicht weniger treffend ist:

|  |
| --- |
| Software architecture is a framework for change. |

Das Software Engineering Institute der Carnegie Mellon University (SEI) hat auf einer eigens dafür eingerichteten Website inzwischen über 150 Definitionen dazu gesammelt [SEI Def]. Bei den ausgearbeiteten Definitionen lässt sich zunehmend ein Konsens erkennen. Dieser Konsens, dem wir uns anschließen wollen, spiegelt sich auch in der Definition des ISO/IEC/IEEE-Standards 42010:2011, Systems and software engineering – Architecture description, wider:

|  |
| --- |
| <**system**> fundamental concepts or properties of a system in its environment embodied in ist elements, relationships, and in the principles of its design and evolution. |

Statt eine grundsätzlich neue Definition für diesen zentralen Begriff im Rahmen des vorliegenden Buches einzuführen, orientieren wir uns an diesem Standard, ergänzen ihn aber an einigen wenigen Stellen, z .B. um den Begriff einer Schnittstelle oder den Bezug zur Entwicklungsorganisation, da diese aus unserer Sicht für Softwarearchitekturen wesentlich sind.

|  |
| --- |
| Die **Softwarearchitektur** definiert die grundlegenden Prinzipien und Regeln für die Organisation eines Systems sowie dessen Strukturierung in Bausteinen und Schnittstellen und deren Beziehungen zueinander wie auch zur Umgebung. Dadurch legt sie Richtlinien für den gesamten Systemlebenszyklus, angefangen bei Analyse über Entwurf und Implementierung bis zu Betrieb und Weiterentwicklung, wie auch für die Entwicklungs- und Betriebsorganisation fest. |

Dieses Verständnis von Softwarearchitektur beinhaltet zwei wesentliche Aspekte. Zum einen findet sich der auf das System bezogene konstruktive Aspekt einer Softwarearchitektur, die den Aufbau respektive die Zerlegung eines softwareintensiven Systems in Bausteine, Schnittstellen sowie deren Beziehungen zueinander und zur Umgebung beschreibt. Darüber hinaus enthält die Definition aber auch einen anderen, auf die Vorgehensweise bezogenen Aspekt: Die Softwarearchitektur beeinflusst auch die Entwicklungsorganisation und den Systemlebenszyklus und gibt deshalb vor, welche Prinzipien und Regeln dabei zu beachten sind.

Daraus resultiert auch die Konsequenz, dass Softwarearchitekturziele durchaus auch längerfristige Ziele sein können, die über Projektziele und deren zeitlicher Horizont, der meist an die Projektlaufzeit gekoppelt ist, hinausreichen. Softwarearchitektur kann somit auch eine Investition in den gesamten Systemlebenszyklus bedeuten und umfassen, der sich unter Umständen erst nach Abschluss des zugehörigen Entwicklungsprojekts amortisiert.

Quelle Baiswissen für Softwararchitekten 4. Auflage