«Conway's Law»

Gesetz von Conway

Das Gesetz von Conway ist eine nach dem US-amerikanischen Informatiker Melvin Edward Conway benannte Beobachtung, dass die *Strukturen von Systemen* durch die Kommunikationsstrukturen der sie umsetzenden Organisationen vorbestimmt sind. Es wurde von Conway 1968 folgendermaßen formuliert:

> *"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."*
>
> *„Organisationen, die Systeme entwerfen, […] sind gezwungen, Entwürfe zu erstellen, die die Kommunikationsstrukturen dieser Organisationen abbilden."*

– Melvin E. Conway

Bsp.:

Angenommen, ein Unternehmen wird mit der Umsetzung eines großen Softwaresystems beauftragt. Das beauftragte Unternehmen hat *drei Entwicklergruppen*, **E1**, **E2** und **E3**, die in dem Projekt *zusammenarbeiten*. Das Gesetz von Conway besagt nun, dass das entwickelte Softwaresystem wahrscheinlich aus *drei großen Subsystemen* **S1**, **S2** und **S3** bestehen wird, die in einer jeweils anderen Entwicklergruppe umgesetzt werden. Wichtiger noch, *die Qualität und Art der Schnittstellen zwischen den Subsystemen (**S1–S2, S1–S3, S2–S3**) werden der Qualität und Art der zwischenmenschlichen Kommunikation* zwischen den entsprechenden Entwicklergruppen (**E1–E2, E1–E3, E2–E3)** entsprechen.

«Brooks's Law»

Brooks's Law lautet: *"Adding manpower to a late software project makes it later"* und stammt von Fred Brooks, einem Manager bei IBM und Autor des 1975 erschienenen Buchs "The Mythical Man-Month".

Die grundlegende Erfahrung, dass ein verzögertes Projekt, insbes. Softwareprojekt, nicht durch den vermehrten Einsatz von Arbeitskraft beschleunigt werden kann, sondern sogar noch verlängert wird, ist zwar leicht zu begründen, aber nur schwer Entscheidern zu vermitteln.

Hauptgrund für die Verringerung der Arbeitsgeschwindigkeit durch Aufstockung der Arbeitskapazität ist die Notwendigkeit, neue Mitarbeiter einzuführen. Dies können am schnellsten die bisherigen Bearbeiter des Projekts tun. Während sie den neuen Kollegen den Stand des Projekts erklären, arbeiten weder sie noch die neuen Mitarbeiter am Projekt. Das Projekt verzögert sich dadurch noch weiter.

Zweiter Grund ist der erhöhte Koordinierungsaufwand bei einer Vergrößerung des Projektteams.

Der dritte Grund besteht darin, dass Aufgaben nicht beliebig geteilt werden können, sondern eine minimale Größe aufweisen. Wenn mehr Personen da sind als zuteilbare Aufgaben, fallen nur noch zusätzliche Koordinationsaufgaben ohne Projektfortschritt an.

Da diese Gründe allgemein und nicht nur für Softwareprojekte zutreffen, kann das Gesetz von Brook als grundsätzlich universell gelten. Einschränkungen gelten dort, wo ein Projekt bereits von Beginn an mit zu geringen Ressourcen geplant war oder wo bestimmte kritische Vorgänge tatsächlich reine

Volumenvorgänge sind (z.B. kann die Geschwindigkeit der Personenbefragung bei einem Marktforschungsprojekt durch Erhöhung der Call-Center-Kapazität sehr wohl beschleunigt werden).

«Parkinson's Law»

Nach dem Parkinsonschen Gesetz dehnt sich die Arbeit in dem Maße aus, wie die *zur Fertigstellung vorgesehene Zeit* verstreicht. Entweder braucht man länger als nötig, um eine Aufgabe zu erledigen, oder man schiebt die Aufgabe vor sich her und erledigt sie erst kurz vor dem Abgabetermin.

Am bekanntesten ist das Parkinsonsche Gesetz zum Bürokratiewachstum, erstmals veröffentlicht 1955.[1] Es lautet:

"Work expands so as to fill the time available for its completion."

„Arbeit dehnt sich in genau dem Maß aus, wie Zeit für ihre Erledigung zur Verfügung steht."

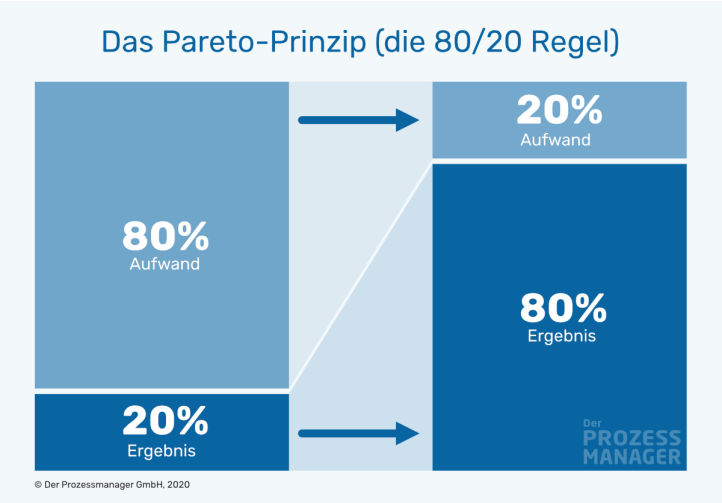
– und nicht in dem Mass, wie komplex sie tatsächlich ist.

Beispiel:

Eine ältere Dame, die einen halben Tag dafür braucht, ihrer Nichte eine Postkarte zu schicken (Postkartenauswahl, Brillen- und Adressensuche, Textverfassung, Entscheidung, ob für den Weg zum Briefkasten ein Schirm mitzunehmen ist). Den Kontrast bildet der vielbeschäftigte Mann, der die gleiche Aufgabe in drei Minuten an seinem Schreibtisch erledigt.

«Pareto's Fallacy»

Das nach Vilfredo Pareto benannte Pareto-Prinzip (oder die Pareto-Methode) befasst sich mit der Beziehung zwischen Aufwand und Ergebnis: 80% der Wirkung können durch 20% der beteiligten Faktoren erreicht werden.



Die **Pareto-Methode** stellt die Beziehung zwischen Aufwand und Ergebnis bzw. zwischen Einsatz und Ertrag dar. **Es besagt, dass 80% der Wirkung durch 20% der beteiligten Faktoren erreicht werden können**.

Anders ausgedrückt sind 20 Prozent des Aufwands für 80 Prozent des Ergebnisses verantwortlich (80 20 Regel). Daraus ergibt sich wiederum, die verbleibenden 20% der Ergebnisse 80% des Aufwands verantworten.

Zwar trifft dieses Zahlenverhältnis nicht immer und überall exakt zu, aber die darin ausgedrückte Gesetzmäßigkeit kann bei der Festlegung, Dokumentation und Darstellung von Prioritäten nützlich sein. Der Nutzen des Pareto-Prinzips besteht darin, sich auf die Aufgaben zu konzentrieren, die den größten Einfluss haben. Hierdurch können Produktivität und Effizienz gesteigert werden.

Im Projektmanagement verringert diese Methode zum Selbstmanagement die Komplexität. Da Aufgaben priorisiert werden, die nur 20% der Zeit in Anspruch nehmen, aber 80% der Ergebnisse liefern, ist die Arbeit konzentrierter. Bei dem Ziel, 100% aller Aufgaben erfüllen zu wollen, besteht die Gefahr, dass der Fokus verloren geht.

https://der-prozessmanager.de/aktuell/wissensdatenbank/pareto-prinzip

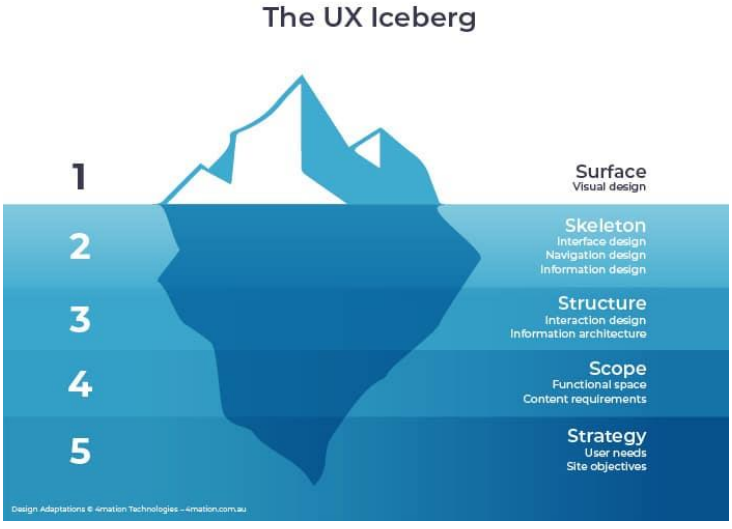
«Sturgeon's Revelation»

Sturgeons Gesetz (englisch Sturgeon's law) ist eine **Hypothese**, die auf der Offenbarung (englisch: Sturgeons Revelation) beruht, laut der **knapp 90 Prozent von allem Mist ist** (englisch: ninety percent of everything is crap). Diese Beobachtung bezieht sich dabei vor allem, aber nicht ausschließlich auf kulturelle Güter wie Musik, Filme und Bücher. Die These wurde von Theodore Sturgeon, einem US-amerikanischen Science-Fiction-Autor und Kritiker, geprägt. Diese geht auf Sturgeons Beobachtung zurück, dass die Science-Fiction von Kritikern oft als minderwertig verspottet wurde, dass aber die meisten Beispiele von Werken aus anderen Bereichen ebenfalls als minderwertig angesehen werden können und dass sich die Science-Fiction in dieser Hinsicht nicht von anderen Kunstformen unterscheidet.

https://de.wikipedia.org/wiki/Sturgeons_Gesetz

«The Iceberg Fallacy»

"Die Kosten für die Entwicklung eines neuen Softwareprodukts sind die einzigen ungefähr 25% der Gesamtbetriebskosten, die das Management sieht und budgetiert."



«The Peter Principle»,

Das **Peter-Prinzip** ist eine These von Laurence J. Peter. Sie lautet: "**In a hierarchy every employee tends to rise to his level of incompetence**." (deutsch: „***In einer Hierarchie neigt jeder Beschäftigte dazu, bis zu seiner Stufe der Unfähigkeit aufzusteigen***.") Sie wurde mit eigenen Notizen zusammen mit Raymond Hull in dem Buch *The Peter Principle* formuliert, das 1969 bei William Morrow in New York erschien. Es zählt zu den Klassikern der nordamerikanischen Managementliteratur. Die deutsche Erstausgabe erschien 1970 unter dem Titel Das Peter-Prinzip oder Die Hierarchie der Unfähigen im Rowohlt Verlag.

«Eagleson's Law»

Any code of your own that you haven't looked at for six or more months, might as well have been written by someone else.

Jeder eigene Code, den Sie sechs oder mehr Monate lang nicht überprüft haben, könnte genauso gut von jemand anderem geschrieben worden sein.

Viele halten Eagleson für einen Optimisten und meinen, dass sechs Monate ein großzügiger Zeitrahmen sind. Wie dem auch sei, Eaglesons Gesetz unterstreicht die Notwendigkeit einer klaren, effektiven Kommentierung und klarer Kodierungsstandards. Schließlich könnte nicht einmal der ursprüngliche Programmierer später unordentlichen Code entziffern.

«Greenspun's 10th Rule of Programming»

Any sufficiently complicated C or Fortran program contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of CommonLisp

Jedes hinreichend komplizierte C- oder Fortran-Programm enthält eine ad hoc, informell spezifizierte, fehlerbehaftete, langsame Implementierung der Hälfte von Common Lisp.