Konfigurationsverwaltung

**Begriffsdefinitionen**

**Software-Einheiten** = Ein Stück Software, das im Software-Lebenslauf entsteht und für die Entwicklung, den Betrieb oder die Wartung des Software-Systems relevant ist. Eine Software-Einheit kann unabhängig von anderen Software-Einheiten bearbeitet, gespeichert und ersetzt werden. Sie besteht nicht aus kleineren Software-Einheiten, sie ist also im Sinne der Verwaltung atomar

**Versionen** = Der Vorgänger einer Software-Einheit wird durch einen Nachfolger, eine neue Version, ersetzt. Dabei wird die Einheit in irgendeinem Sinne besser gemacht, z.B. durch eine Korrektur, eine Erweiterung oder eine Anpassung. Um Verwechslungen zu vermeiden, wird der feste Teil der Bezeichnung übernommen, die mehrstufige Versionsnummer jedoch erhöht, was bedeutet, dass mehrere Versionen in einer zeitlichen Ordnung hintereinander stehen.  
Weiterhin wird gemäß des IEEE-Standards zwischen Version und Revision unterschieden: anders als bei einer Version wird bei einer Revision eine Software-Einheit nicht komplett ersetzt sondern repariert.

**Varianten** = Eine Variante entsteht, wenn bei dem Verkauf von Software-Systemen an mehrere Kunden jeweils spezifische Anpassungen gemacht werden. Somit stehen Varianten im Gegensatz zu Versionen zeitlich nicht hintereinander, sondern nebeneinander. Mehrere Varianten eines Systems bilden zusammen eine Variantenfamilie.

**Konfigurationen** = Eine Konfiguration stellt eine Menge von Software-Einheiten dar. Diese sind für einen definierten Zweck zusammengestellt und passen, sowie auch das Geflecht der Beziehungen zwischen den Einheiten, zueinander.   
Wie viele Software-Einheiten in einer Konfiguration enthalten werden sein müssen, bestimmt ein Auswahlverfahren: zum Beispiel stellt eine 1-aus-N-Auswahl die Auswahl von genau einer Einheit aus der Variantenfamilie dar. Bei mehreren Einheiten aus derselben Familie spricht man von einer N-aus-M-Auswahl.   
Auch können Konfigurationen zwei Varianten darstellen, wenn sie unterschiedliche Varianten einer Variantenfamilie enthalten.

**Baseline** = eine Konfiguration, die so zusammengestellt und geprüft ist, dass sie als stabil gilt und so als Bezugspunkt für die weitere Entwicklung geeignet ist. Eine Baseline *kann* somit ein lauffähiges System bilden.

**Release** = im Gegensatz zur Baseline muss das Release ein lauffähiges System bilden, da es eine Baseline ist die mit dem Zweck definiert wurde, diese Konfiguration an die Kunden auszuliefern.

**Konfigurationsverwaltung =** diejenige Rolle oder Organisationseinheit, die die Software-Einheiten und Konfigurationen identifiziert, verwaltet, bei Bedarf bereitstellt und ihre Änderungen überwacht und dokumentiert. Dazu gehört auch die Rekonstruktion älterer Software-Einheiten und Konfigurationen, sowie die Versions- und Variantenverwaltung.  
Die Konfigurationsverwaltung hat somit ein Monopol auf die Bereitstellung der Software, was bedeutet, dass alle Entwickler ihre Software-Einheiten an diese übergeben und so eindeutig identifiziert, erfasst, archiviert und vor Änderungen geschützt werden.   
  
Aufgaben der Konfigurationsverwaltung:

* **Versionskontrolle** = die erstellten Versionen und Varianten aller Software-Einheiten werden über lange Zeiträume verwaltet.
* **Konfigurationskontrolle** = es können beliebige Konfigurationen des Software-Systems erstellt werden, wenn die benötigten Software-Einheiten vorliegen.
* **Konstruktion ausführbarer Programme** =der Build-Prozess ( = aus Quellcode werden ausführbare Programme erzeugt) wird mit den Konfigurationsinformationen gesteuert und automatisiert.
* **Änderungskontrolle** =bedeutet, dass alle Änderungen an den Software-Einheiten überwacht und alle Prüfresultate verwaltet werden.
* **Koordination der Teamarbeit** = durch eine gemeinsame Referenzumgebung und die Konflikterkennung oder -vermeidung wird die Zusammenarbeit der Entwickler unterstützt.

**→ Referenzumgebung** = diese sorgt dafür, dass nur geprüfte Software-Einheiten als Systembausteine verwendet werden.   
Ein Entwickler erstellt, modifiziert und überprüft Software-Einheiten in seiner persönlichen **Konstruktionsumgebung**, die Fertigen übergibt er dann der Referenzumgebung. Die Software-Einheiten darin sind der Konfigurationsverwaltung unterstellt und können somit nicht verändert werden.   
**Prüfumgebung** = darin werden die Software-Einheiten geprüft. Eine Einheit, in der Fehler entdeckt werden, erhält den Status „verworfen“ und wird zur Korrektur an den Entwickler zurückgegeben. Wird eine Einheit mit „gut“ bezeichnet, kann diese an die **Integrationsumgebung** weitergegeben werden. In dieser Umgebung wird das System aus guten Einheiten schrittweise zusammengebaut. Wird eine fehlerhafte Einheit entdeckt, wird diese erneut geprüft, erst wenn diese korrigiert ist, wird die Integration fortgesetzt.   
**Systemtestumgebung** = nach einem entstandenen Release-Kandidaten, erfolgt in dieser Umgebung der Systemtest.  
**Release-Umgebung** = wird der Release-Kandidat ohne Befunde getestet, geht der Release an die Release-Umegbung.   
**Produktionsumgebung** = in dieser Umgebung wird das Release produktiv eingesetzt.

**Konfigurationsdatenbank** = darin werden alle Informationen zu den Software-Einheiten und den definierten Konfigurationen abgelegt.