LS 1 – Infos - Python

Python - Schlüsselwörter (Keywords)

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Python - Eingebaute Funktionen (Build In Functions)

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Phyton Standard Bibliothek (Standard Library)

https://docs.python.org/3.8/library/

Sonstige Informationsquellen

https://www.python.org/
https://docs.python.org/3/
https://www.w3schools.com/python/
https://py-tutorial-de.readthedocs.io/de/python-3.3/#
https://www.python-kurs.eu/python3_kurs.php

LS 1 – Infos – Struktogramme

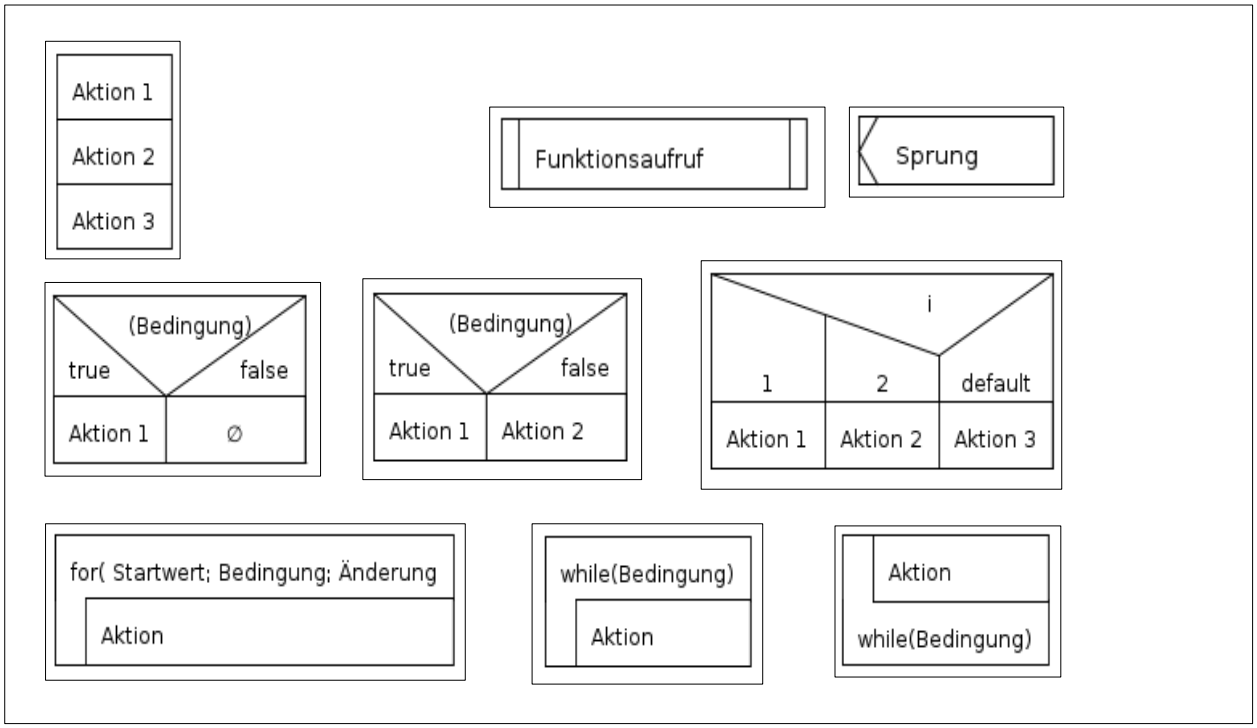
Struktogramme

- Diagrammtyp zur Darstellung von Programmentwürfen im Rahmen der strukturierten Programmierung
- 1972/73 von Isaac Nassi und Ben Shneiderman entwickelt.
- In der DIN 66261 genormt
- Sollen keine programmsprachenspezifischen Konstrukte enthalten
- Sollen leicht verständlich sein

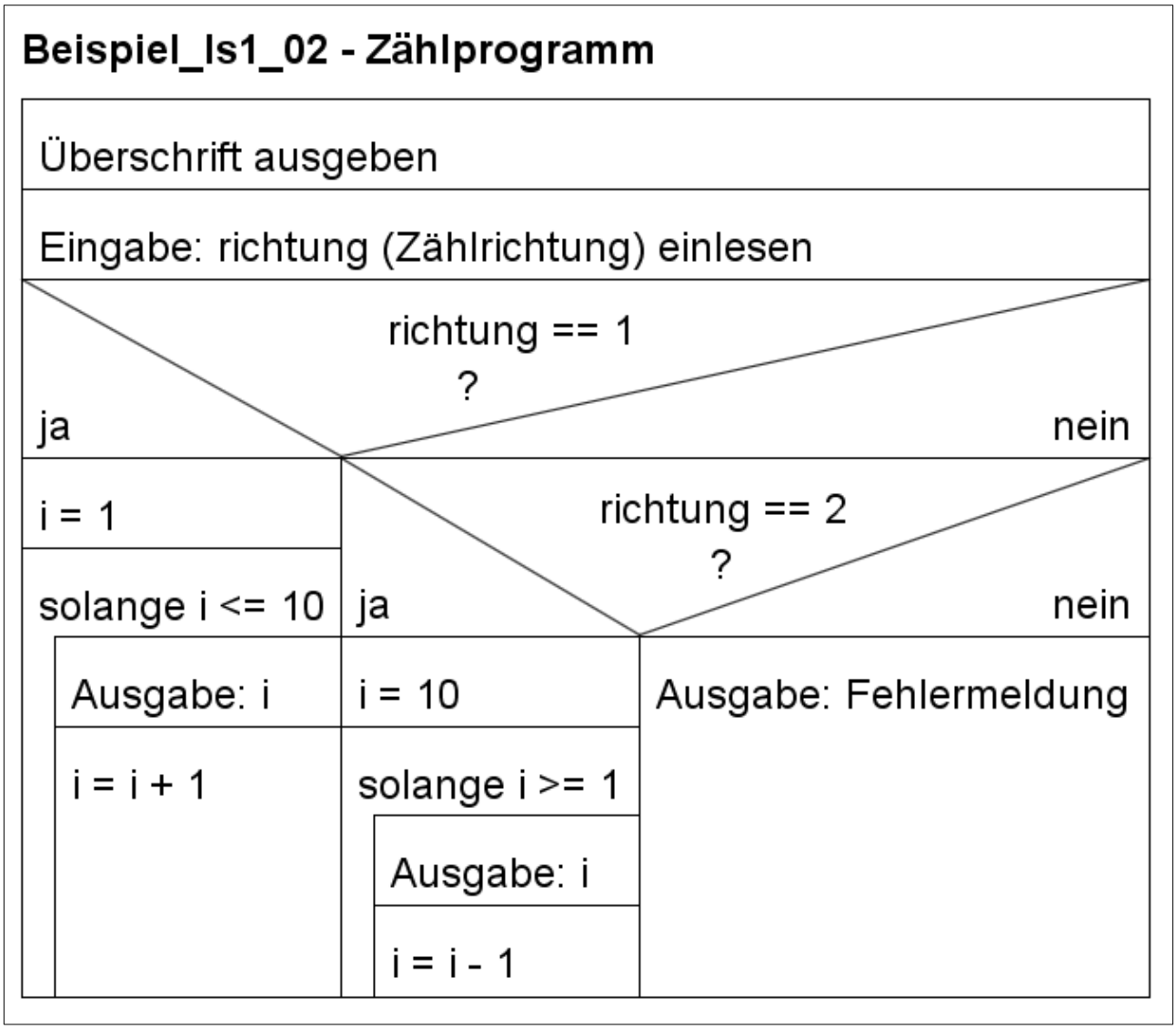
Werkzeugunterstützung

- Zur Erstellung von Struktogrammen gibt es spezielle Programme. Diese bieten oft neben einem Editor zur Erstellung des Struktogramms auch Codegeneratoren für verschiedene Programmiersprachen.
- Als Freeware ist unter anderem verfügbar:
  - Structorizer - zur Beispielerstellung verwendet
  - NSD

Notation



Beispiel


Beispiel_ls1_02 - Zählprogramm
Überschrift ausgeben
Eingabe: richtung (Zählrichtung) einlesen
richtung == 1 ?
ja
nein
i = 1
solange i <= 10
Ausgabe: i
i = i + 1
richtung == 2 ?
ja
nein
i = 10
solange i >= 1
Ausgabe: i
i = i - 1
Ausgabe: Fehlermeldung

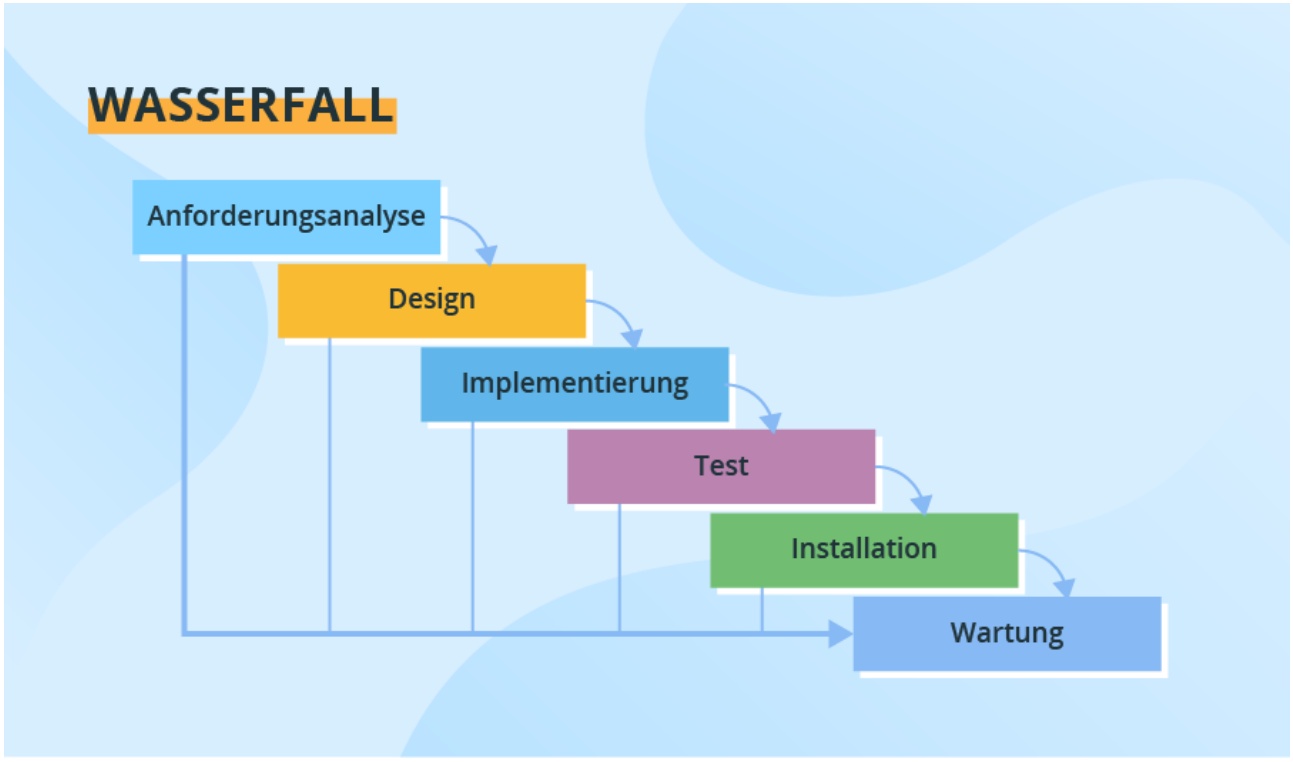
LS 1 – Infos – Vorgehensmodelle

Quelle: ScienceSoft
https://www.scnsoft.de

Traditionelle Vorgehensmodelle
- **Wasserfallmodell**
- **V-Modell (Validierungs- und Verifizierungsmodell)**

Agile Vorgehensmodelle
- **Scrum**
- **Extreme Programmierung (XP)**

Wasserfallmodell

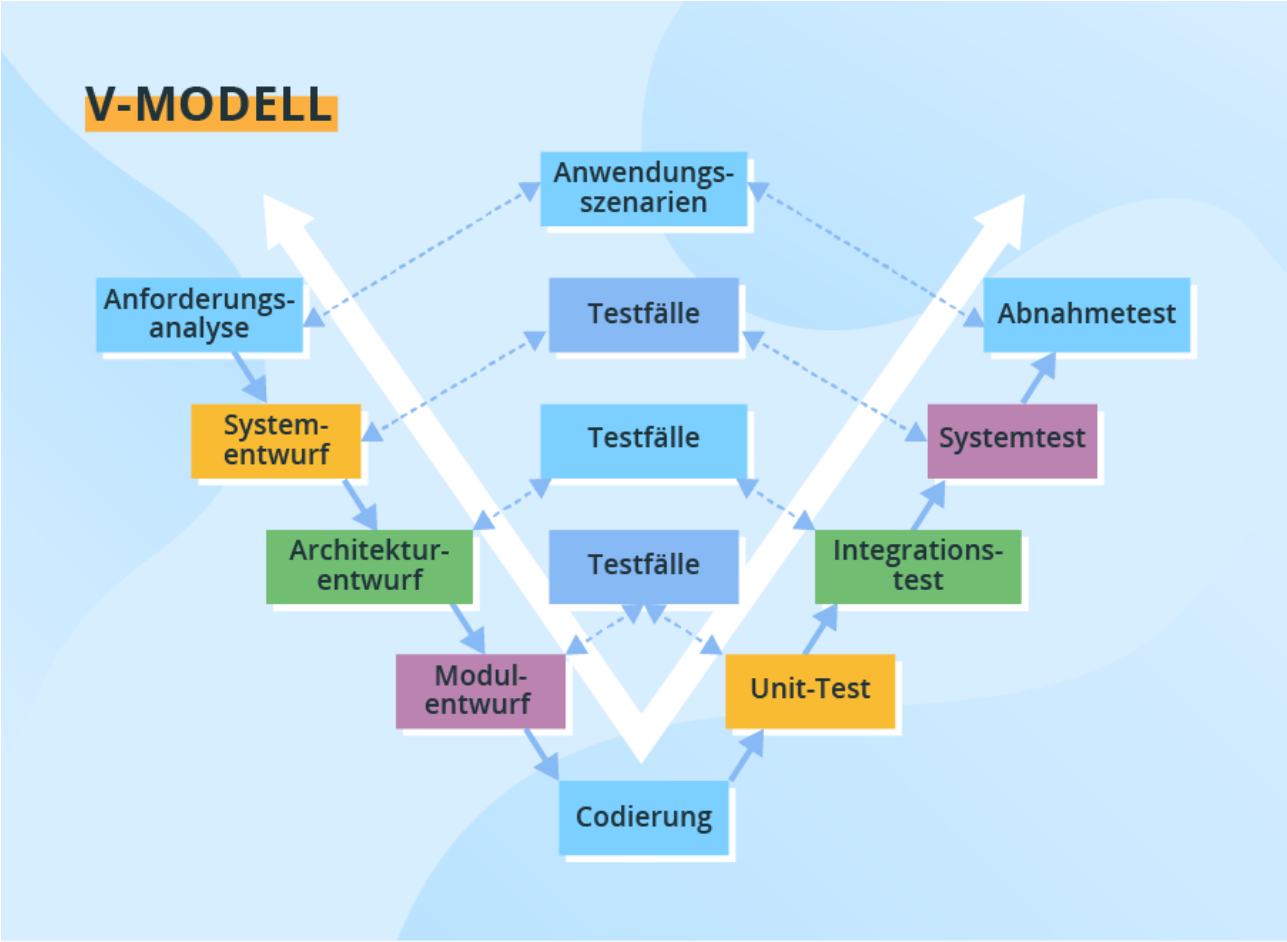


Bei diesem sequentiellen Vorgehensmodell ist der Entwicklungsprozess in aufeinanderfolgenden Phasen organisiert: Analyse, Entwurf, Codierung, Test, Installation, Wartung. Dabei wird jede einzelne Phase stark dokumentiert.

Sobald die vorhergehende Stufe beendet wird, beginnt die nächste, die auf konkreten Ergebnissen der vorherigen Phase basiert. Deshalb gibt es keine Möglichkeit, beispielsweise, Softwareanforderungen während der Entwicklungsphase neu zu bewerten.

Es ist auch unmöglich, die fertige Software zu sehen und zu testen, bis die letzte Entwicklungsstufe vollständig abgeschlossen ist, was zu hohen Projektrisiken und unvorhersehbaren Ergebnissen führt. Deshalb werden Fehler nur am Ende des Projektes in der Testphase gefunden. Deren umgehende Behebung erfordert einen größeren Aufwand und führt zu steigenden Gesamtkosten.

V-Modell (Validierungs- und Verifizierungsmodell)

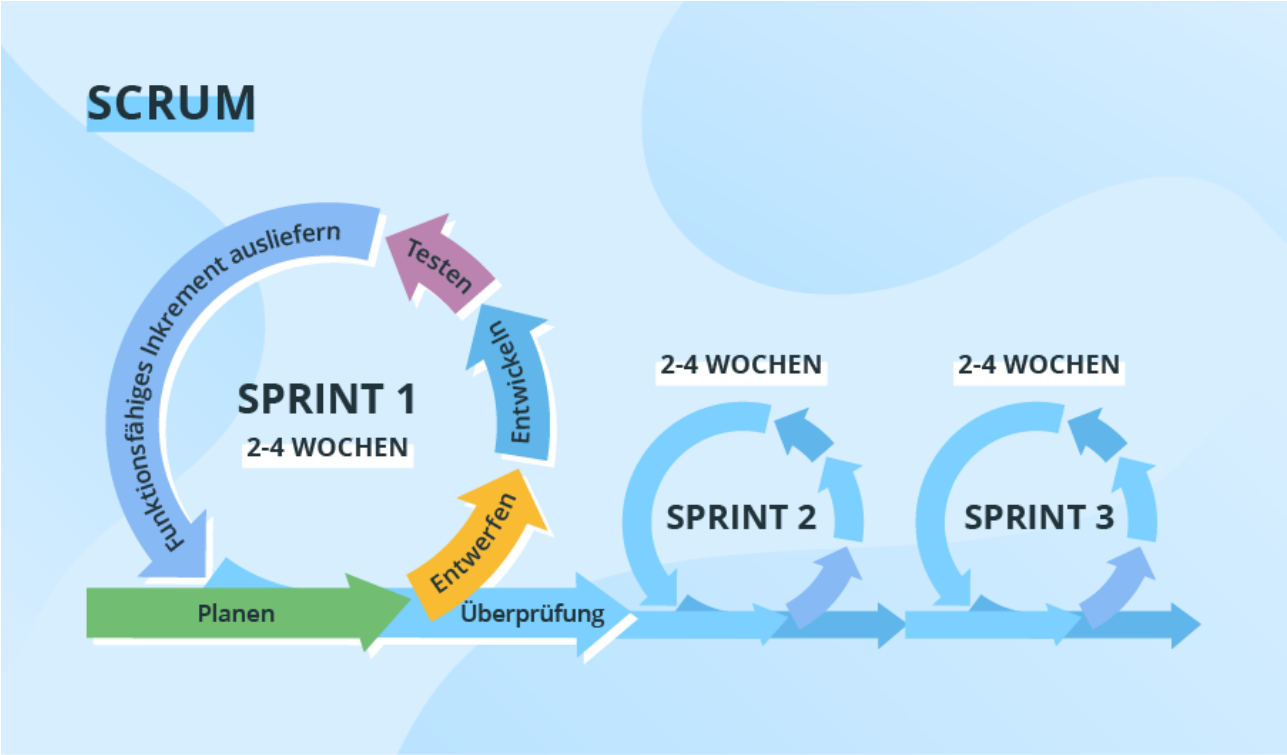


Das V-Modell ist ein weiteres Vorgehensmodell, bei dem alle Phasen nacheinander, also linear, durchlaufen werden. Aber für jede Stufe werden entsprechende Testaktivitäten definiert.

Einerseits ermöglicht das, die Softwarequalität effizienter zu kontrollieren und damit Projektrisiken auf ein Minimum zu reduzieren. Andererseits gehört das V-Modell dadurch zu einem der kosten- und zeitaufwändigsten Modelle.

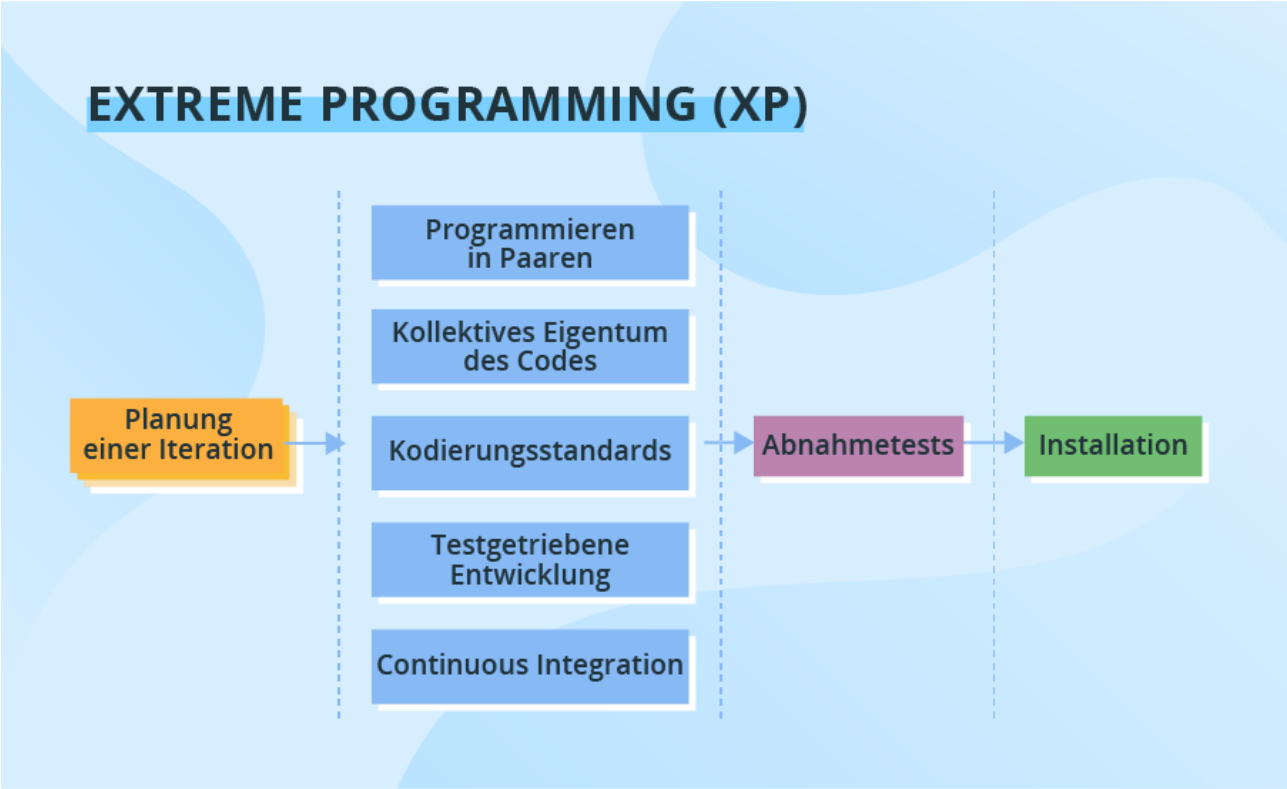
Auch wenn Fehler in Anforderungsspezifikation, Code und Architektur frühzeitig erkannt werden, ist es mmer noch teuer und schwierig, Änderungen während der Entwicklung zu implementieren. Wie beim Wasserfallmodell werden alle Anforderungen zu Beginn erfasst und können nicht geändert werden.

Scrum



Scrum ist eine der beliebtesten Agilen Entwicklunsmethoden, welche die Aufteilung der gesamten Projektlaufzeit in kurze Iterationen (sog. Sprints) ermöglicht. Jede Iteration beginnt mit der sorgfältigen Planung und endet mit dem Sprint Retrospektive, wo der vorherige Sprint analysiert und bewertet wird. Nachher folgt der nächste Sprint, der auf den Ergebnissen des vorherigen Sprints gebaut ist (z. B. Kundenfeedback, neuen Anforderungen an das Produkt und mehr). Eine Iteration dauert üblicherweise von 2 bis 4 Wochen. Nachdem die Aktivitäten für den nächsten Sprint festgelegt worden sind, können keine Änderungen vorgenommen werden.

Extreme Programmierung (XP)



Bei Extreme Programmierung (XP) stehen kurze Iterationszyklen und Kundeneinbindung im Mittelpunkt. Eine typische Iteration dauert 1-2 Wochen. Dieses Vorgehensmodell orientiert sich auf Kundenforderungen sowie -bedürfnisse und ermöglicht, Kunden intensiv zu involvieren, benötigte Änderungen in den Entwicklungsprozess zu integrieren.

Diese Flexibilität erschwert erheblich die Auslieferung hochwertiger Software. Um diesen Anforderungen zu begegnen, kommen bei der XP bewährte Praktiken ins Spiel: Programmieren in Paaren, testgetriebene Entwicklung und Testautomatisierung, Continuous Integration, kleine Releases, einfaches Softwaredesign und Vorschriften zur Einhaltung von Kodierungsstandards.

LS 1 – Infos – Kontextdiagramm

Sinn/Aufgabe

- Darstellung/Modellierung der Systemumgebung

Graphische Darstellung

- Zu entwickelndes System wird als Prozess (Kreis) in der Mitte dargestellt.
- Mit dem System interagierende Elemente werden als Terminatoren (Vierecke) dargestellt.
- Kommunikation zwischen System und interagierenden Elementen wird als Datenfluss (Pfeil) dargestellt.

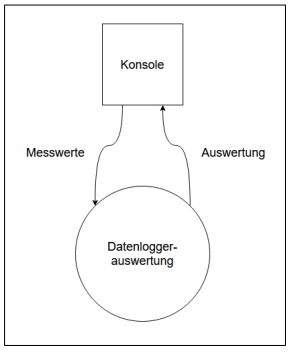
Hinweise zur Erstellung

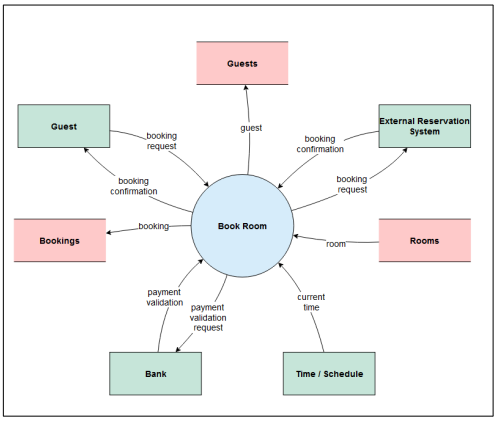
- Es gibt immer nur einen Prozess (Kreis) in einem Kontextdiagramm.
- Es kann mehrere Terminatoren (Vierecke) in einem Diagramm geben.
- Jeder Terminator muss durch mindestens einen Datenfluss (Pfeil) mit dem Prozess verbunden sein. Sonst ist der Terminator für das jeweilige System uninteressant und wird nicht dargestellt.
- Datenflüsse zwischen Terminatoren werden nicht dargestellt.

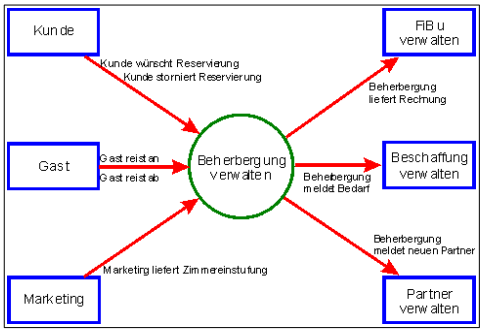
Herkunft

- Strukturierte Analyse

Beispiele







Weitere Informationsquellen

- http://www.infforum.de/themen/anforderungsanalyse/re-systemabgrenzung.htm
- https://de.wikipedia.org/wiki/Kontextdiagramm