Entwicklung einer Klassenbibliothek zur Erzeugung autokorrelierter Zufallszahlen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2017

Autor(en): Anthony Delay
Philipp Bütikofer
Betreuer: Prof. Dr. Andreas Rinkel
Lukas Kretschmar

Inhalt

1. Abstract [bis 20.12.2017]

2. Einführung und Motivation [bis 18.10.2017]

In der Simulation von Systemen werden Zufallszahlen zur Beschreibung der einzelnen Arbeitsschritte benötigt. Standardmässig werden diese Zufallszahlen so erzeugt, dass sie keine Autokorrelationen (Abhängigkeiten) aufweisen.
Die Realität sieht jedoch anders aus. Es hat sich gezeigt, dass in der Praxis häufig ebendiese Autokorrelationen auftreten. Aufgrund dieser Abhängigkeiten können simulierte und reale Ergebnisse stark voneinander abweichen. Im Rahmen der Studienarbeit HS2017/18 soll eine Klassenbibliothek (ARTA.Standard) entwickelt werden, welche es ermöglicht, autokorrelierte Zufallszahlen zu erzeugen. Der Grad der Autokorrelation kann selbst definiert werden. ARTA.Standard soll so implementiert werden, dass eine Einbindung in die Simulationssoftware Simio oder andere Simulationstools möglich ist.

3. Zugrundeliegende Arbeiten [bis 18.10.2017]

Als Fundament für die vorliegende Studienarbeit gelten die beiden Dokumente «Autoregressive to anything: Time-series input processes for simulation[1]» und «JARTA — A Java library to model and fit Autoregressive-To-Anything processes[2]».
Die erst genannte Publikation beschreibt den ARTA-Prozess auf der mathematischen Ebene. ARTA (Autoregressive-to-anything) stellt ein bewährtes Modell zur Erzeugung von zufällig generierten Zahlen, mit gegebener Randverteilung und einer Autokorrelation aufweisendem Muster dar. Entwickelt wurde das ARTA-Modells von Marne C. Cario und Barry L. Nelson.
Die zweite Publikation stellt eine Java Implementation vor, welche den ARTA-Prozess abbildet. Mit JARTA werden die Ansätze von ARTA in eine JAVA-Library abgebildet. An einem konkreten Beispiel einer Lagerhaussimulation zeigen Tobias Uhlig und Oliver Rose die Funktionsweise und Wichtigkeit der Abhängigkeiten, wenn es um das Modellieren von stochastischen Prozessen geht. Der Sourcecode von JARTA ist frei verfügbar.

4. Autokorrelation [bis 25.10.2017]

Dieser Abschnitt wird den Begriff der Autokorrelation erläutern und deren grundlegende Eigenschaften und Charakteristiken aufzeigen. Anschliessend wird auf die Bereiche, welche Autokorrelation aufweisen eingegangen. Um den Themenbereich abzuschliessen wird Autokorrelation anhand eines konkreten Beispiels aufgezeigt.

4.1 Definition

Autokorrelation setzt sich aus zwei Wörtern («Auto» und «Korrelation») zusammen. Der Wortteil Korrelation beschreibt dabei einen Zusammenhang zwischen mindestens zwei oder mehreren Merkmalen, Zuständen, Funktionen oder Ereignissen. Diese Merkmale können sich je nach Anwendungsgebiet sehr stark unterscheiden.
Das Präfix «Auto» zeigt auf, dass die Funktion oder Reihe mit sich selbst korreliert. Dies bedeutet, dass ähnliche oder gleiche Muster erkennbar sind. Bei Autokorrelation sind also die Werte einer Variable zum Zeitpunkt t mit den Werten derselben Variable in zeitlich vergangenen Perioden abhängig. Die Autokorrelation ist immer zeitabhängig. Der Zusammenhang zwischen Autokorrelation und Zeit kann in Form von Korrelationsfunktionen ausgedrückt werden. Eine Korrelationsfunktion zeigt an, wie viel Ähnlichkeit zwischen der ursprünglichen und der, um eine Zeit t, verschobenen Folge besteht.

[1] Modeling and generating multivariate time-series input processes using a vector autoregressive technique, 10.1145/937332.937333
[2] JARTA — A Java library to model and fit Autoregressive-To-Anything processes, 10.1109/WSC.2013.6721508

4.2 Korrelationskoeffizienten

Grundsätzlich gilt die Aussage, «Korrelation gilt als Mass eines Zusammenhangs». Dieses Mass kann numerisch in Form von Korrelationskoeffizienten ausgedrückt werden und beantwortet die Frage nach der Stärke und der Richtung des Zusammenhangs. Bei Korrelationskoeffizienten handelt es sich um Zahlen, welche in einem Intervall zwischen -1 und 1 liegen. Eine Korrelation die den Koeffizienten 1 aufweist wird als perfekte positive, bei -1 als perfekte negative Korrelation bezeichnet. Je weiter sich der Korrelationskoeffizient dem Wert 0 nähert, umso schwächer ist die Korrelation. Ein Korrelationskoeffizient mit dem genauen Wert 0 bedeutet, dass keine Korrelation vorhanden ist und die Werte perfekt verteilt sind.

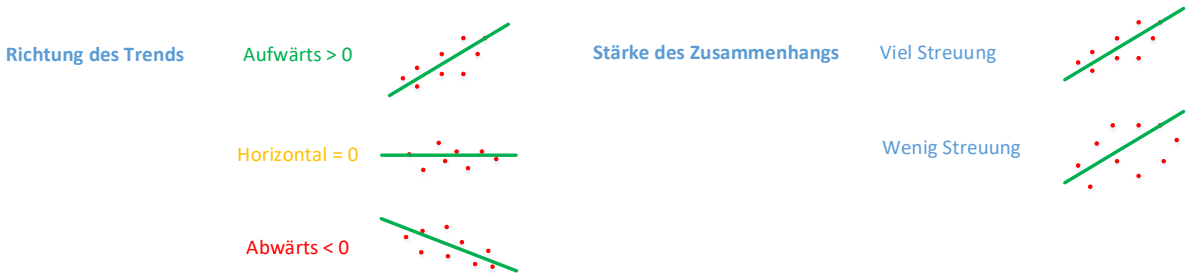


Abbildung 1: Korrelationskoeffizient

Im Zusammenhang mit dem Begriff Korrelationskoeffizient taucht der Ausdruck Pearson-Korrelation auf. Dieser ist nach Karl Person benannt, welcher das Mass der Korrelation in Form des Korrelationskoeffizienten in Zusammenarbeit mit Auguste Bravais entwickelt hat. Dies ist daher speziell erwähnenswert, da auf den Algorithmus von Pearson innerhalb der in dieser Arbeit erzeugten Klassenbibliothek zurückgegriffen wird.

Autokorrelation kann durch mathematische Formeln ausgedrückt werden, jedoch wird sie in jedem Anwendungsbereich domänenspezifisch definiert.

4.3 Anwendungsbereiche

Autokorrelation kann in verschiedenen Gebieten vorgefunden werden. Als die signifikantesten gelten Statistik, Signalanalyse, Informationstheorie und die Softwaretechnik.

Autokorrelation in der Statistik: In der Statistik wird durch die Autokorrelation das Mass des Zusammenhangs zwischen zwei Zufallsvariablen beschrieben. Am häufigsten wird dieses Mass in Form der Korrelationskoeffizienten (Pearson) angegeben.

Autokorrelation in der Signalanalyse und Bildverarbeitung: In diesem Anwendungsgebiet wird eine Autokorrelationsfunktion genutzt, um die Korrelation eines Signales mit sich selbst zu unterschiedlichen Zeitverschiebungen eingesetzt. Somit kann beispielsweise der Zusammenhang zwischen Faltung und Autokorrelation aufgezeigt werden. In der Bildverarbeitung wird die zeitliche Komponente durch eine örtliche ersetzt. Dadurch lässt sich beispielsweise Objekterkennung realisieren.

Autokorrelation in der Softwaretechnik: Anwendung findet die Autokorrelation hier im sogenannten Korrelationstest. Dieser beschreibt ein Verfahren, welches die Plausibilität einzelner Parameter einer Funktion und deren Kombinationen überprüft.

Autokorrelation in der Informationstheorie: *[TODO] Kryptographie erwähnen & Kurzbeschreibung zur Informationstheorie*

4.4 Partielle Korrelation

Unter der partiellen Korrelation versteht man das nicht-berücksichtigen von Dritteinflüssen. Eine Korrelation zwischen zwei statistischen Werten a und b kann unter Umständen auf einen gemeinsamen Faktor c zurückgeführt werden. Um diesen Effekt auszuschalten kann das Konzept der partiellen Korrelation eingesetzt werden. Durch eine partielle Korrelation wird der dritte Faktor entweder ausgeschaltet oder gezielt kontrolliert, so dass dieser das Resultat nicht verfälschen kann.

4.5 Durbin-Watson-Test

Die gebräuchlichste Methode um die Existenz von Autokorrelation zu belegen stellt der Durbin-Watson-Test dar. Durch diese Art statistischer Test kann geprüft werden, ob eine Autokorrelation der 1. Ordnung vorliegt. Autokorrelation erster Ordnung bedeutet, dass aufeinanderfolgende Glieder der Reihe bzw. ihrer Residualgrössen[3] korrelieren. Das Ergebnis eines DW-tests ist ein numerischer Wert im Bereich von 0 bis 4.

Wert des Tests	Korrelationskoeffizient	Bedeutung
d = 2	0	Keine Autokorrelation
d = 0	1	Perfekte positive Autokorrelation
d = 4	-1	Perfekte negative Autokorrelation

Der DW-Test ist durch den folgenden Term definiert:

$$d = \frac{\sum_{t=2}^{T}(\varepsilon_t - \varepsilon_{t-1})^2}{\sum_{t=1}^{T}\varepsilon_t^2}$$

Ausdruck	Bedeutung
$\sum_{t=2}^{T}$	Summiert alle Sequenzglieder zwischen t = 2 und T, wobei t und T die Anzahl aller Beobachtungen ist. Die Anzahl der Beobachtungen entspricht dem Start und -Endwert der Zeitreihe.
$(\varepsilon_t - \varepsilon_{t-1})^2$	Entsprechen den Residuen/Werte der Reihe.
$\sum_{t=1}^{T}\varepsilon_t^2$	*[TODO]*

[3] Vertiefter Einblick Residuum: https://de.wikipedia.org/wiki/Residuum_(Statistik)

4.6 Beispiel Autokorrelation

Folgend dargestellt ist ein Beispiel zur Autokorrelation aus dem Bereich der Kryptographie. In der Kryptographie stellt die Autokorrelation eine Kennzahl für die Ähnlichkeit von Teilen eines Dokuments dar. Mithilfe der Autokorrelation kann unter Umständen die Schlüssellänge eines verschlüsselten Dokuments ermittelt werden.
Als Grundlage dient die Verschlüsselungsmethode Vigenère-Chiffre. Bei diesem Verfahren handelt es sich um ein Substitutionsverfahren, wobei der Klartext in Monogramme (einzelne Zeichen) zerlegt wird und diese anschliessend durch Geheimtextzeichen substituiert werden.
In diesem Beispiel wird vom Standardalphabet mit 26 Buchstaben ausgegangen. Daraus wird eine Matrix (Vigenère-Quadrat) erstellt, welches die 26 Buchstaben immer um eine Position verschoben darstellt.

Klartext

Schlüssel	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Geheimtext
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	

Tabelle 1 Vigenère-Quadrat

Anschliessend muss ein Schlüssel gewählt werden. Dieser sollte möglichst lang und aus einer möglichst zufälligen Sequenz der Buchstaben des Alphabets bestehen. Der Kreuzungspunkt der einzelnen Buchstaben des Klartextes und des Schlüssels können nun innerhalb des Vigenère-Quadrats abgelesen werden und ergeben so das neue, verschlüsselte Zeichen.

Mithilfe der Software Cryptool[4] können solche einfache Verschlüsselungsverfahren aufgezeigt und analysiert werden. Cryptool verwendet folgende Autokorrelationsfunktion C(t), welche die Ähnlichkeit einer Folge[5] (s[i]) = s[1], s[2], …, s[n] und der um t Stellen verschobenen Folge (s[i+t]= s[1 + t], s[2 + t], s[n + t].

$$C(t) = \frac{(A(t) - D(t))}{n}$$

Wobei A(t) = Anzahl der übereinstimmenden Glieder der Folgen s[i] und s[i + t] im betrachteten Abschnitt und D(t) = Anzahl der nicht übereinstimmenden Glieder derselben Folgen und Abschnitt ist. n beschreibt die Länge der Sequenz.
Zur Veranschaulichung werden diese Formeln in ein Autokorrelationsdiagramm umgesetzt.

4.6.1 Beispiel 1 – starke Autokorrelation

Schlüssel:
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Klartext:
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXY ZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWX YZ
Verschlüsselter Text:
ACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKM OQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYACEGIKMOQSUWYA CEGIKMOQSUWY

Dadurch, dass die Vigenère-Chiffre auf Substitution und Verschiebung der einzelnen Zeichen basiert, korrelieren die einzelnen Zeichen nach einer gewissen Zeit bzw. Verschiebung miteinander. Das erste, triviale, Beispiel zeigt eine sehr starke Autokorrelation, da der Klartext lediglich ein Vielfaches des Schlüssels ist.

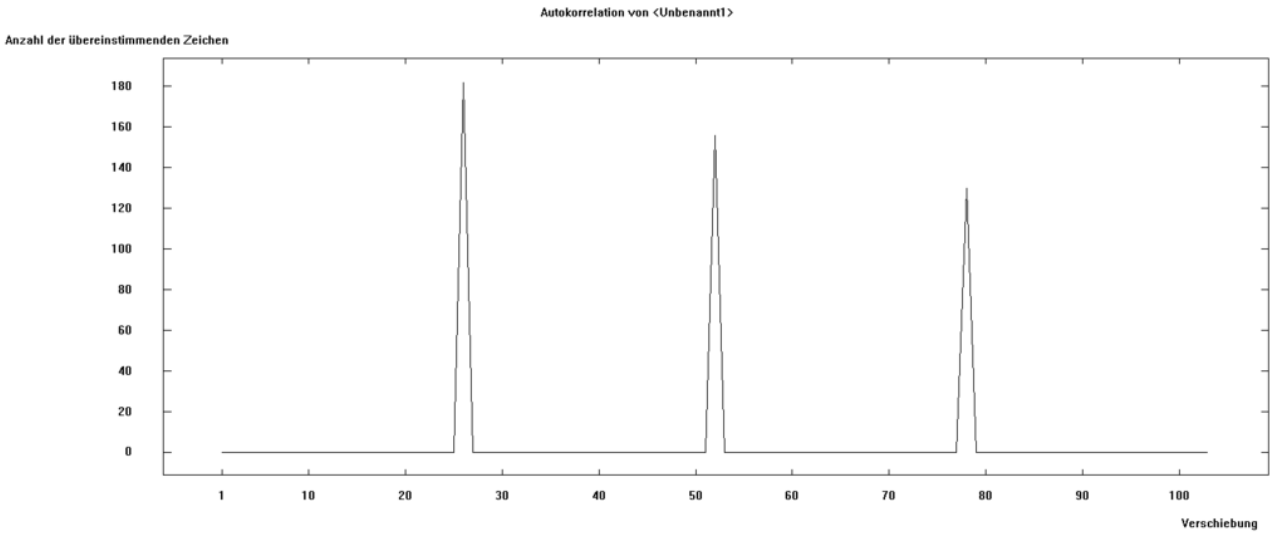


Figure 1 Autokorrelation des unverschlüsselten Textes, Bsp. 1

Wird nun die Autokorrelation des unverschlüsselten Textes betrachtet, so kann man die Verschiebung um 26 Zeichen klar erkennen.

[4] https://www.cryptool.org/de/cryptool1
[5] Die Indexierung der Folge ist 1 - basiert

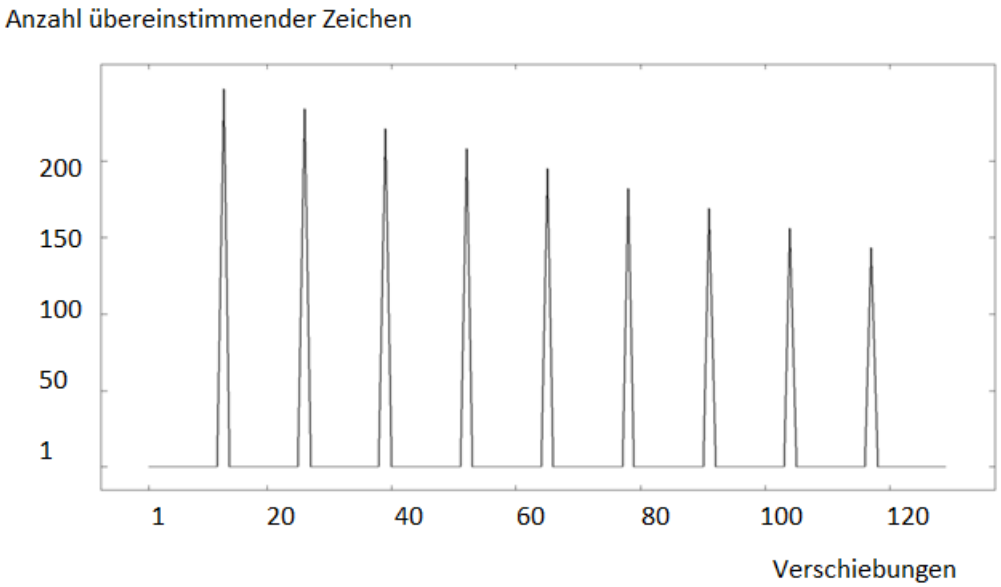


Figure 2 Autokorrelation verschlüsselter Text, Bsp.1

Man würde erwarten, dass auch beim verschlüsselten Text die Autokorrelation bei 26 Verschiebungen am stärksten ist. Jedoch ist dies ein Trugschluss. Die Autokorrelation ist bei 13 Verschiebungen deutlich am stärksten. Dies kommt daher, dass das Vigenère-Quadrat eine Diagonale bildet, an welcher das Alphabet neu startet.

Weiter ist der Vergleich zwischen verschlüsseltem Text und des Klartextes spannend. Dort kann gesehen werden, dass «ABCDEFGHIJKLMNOPQRSTUVWXYZ» der Zeichenkette «ACEGIKMOQSUWYACEGIKMOQSUWY» entspricht. Es wird erkannt, dass genau nach 13 Zeichen wiederrum das Zeichen «A» auftaucht, was auf die obengenannte Struktur des Vigenère-Quadrat hinweist.

4.6.2 Beispiel 2

Schlüssel:
AUTOKORRELATION

Klartext:
Die Giraffen[6] sind eine Gattung der Säugetiere aus der Ordnung der Paarhufer. Ursprünglich wurde ihr mit Giraffa camelopardalis und der Trivialbezeichnung Giraffe nur eine einzige Art zugewiesen. Molekulargenetische Untersuchungen zeigen jedoch, dass die Gattung wenigstens vier Arten mit sieben eigenständigen Populationen umfasst. Die Giraffen stellen die höchsten landlebenden Tiere der Welt. Zur Unterscheidung vom verwandten Okapi werden sie auch als Steppengiraffen bezeichnet.

Verschlüsselter Text:
Dcx Usfrwjpn lqbq ecgs Qokkyyg wmf Fäuaxhssiv efs wmf Brxgixu uvv Aatzvhfyk. Ibggiürrlbkv julws svi dme Gbzosfu vowscftlrwizvs ogr nsi Kvtvbizoetxwmvelrr Gbzosfy gib szei pighwte Ukh jixvatelmb. Zofxyezrikpnxbwfcbx Ixhviwfcacbteh sssuve npdhkv, qaml rss Xrxeugo krnczgdsej ztek Ifgeh fwd gzvfpn xqurnmmäbnwxvr Aoiczntchbob ldjlslb. Rve Abfktwvr dtxtzrn xbs röqyjxpn eibqlyusxrve Xtekm rrr Qxzd. Nli Yytxzgphybrebx msx vxzknnxmsx Cbrtt wxzrrn mbs kity ews Lbscpygusfrwjpn umnriwaboh.

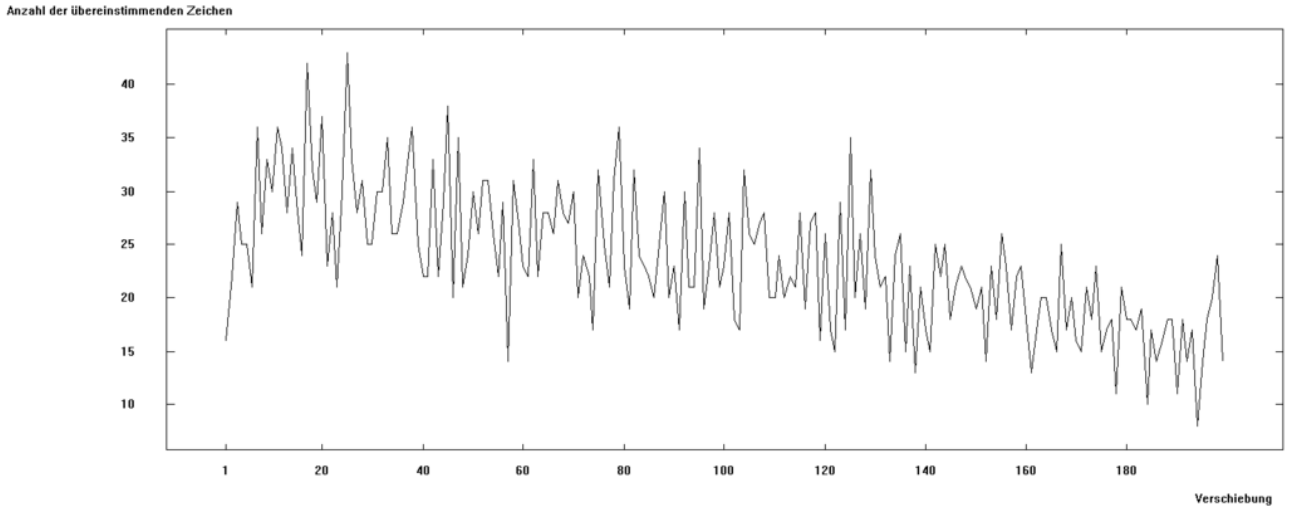


Abbildung 2 Autokorrelation des Klartextes

[6] Quelle : https://de.wikipedia.org/wiki/Giraffen

Auf den ersten Blick vermittelt dieses Diagramm einen willkürlichen Eindruck. Jedoch können auch hier autokorrelierte Strukturen erkannt werden. Diese sind nicht mehr nach einer fixen Anzahl Zeichen erkennbar wie in Beispiel 1, trotzdem sind vereinzelte, sehr starke Korrelationen ersichtlich.

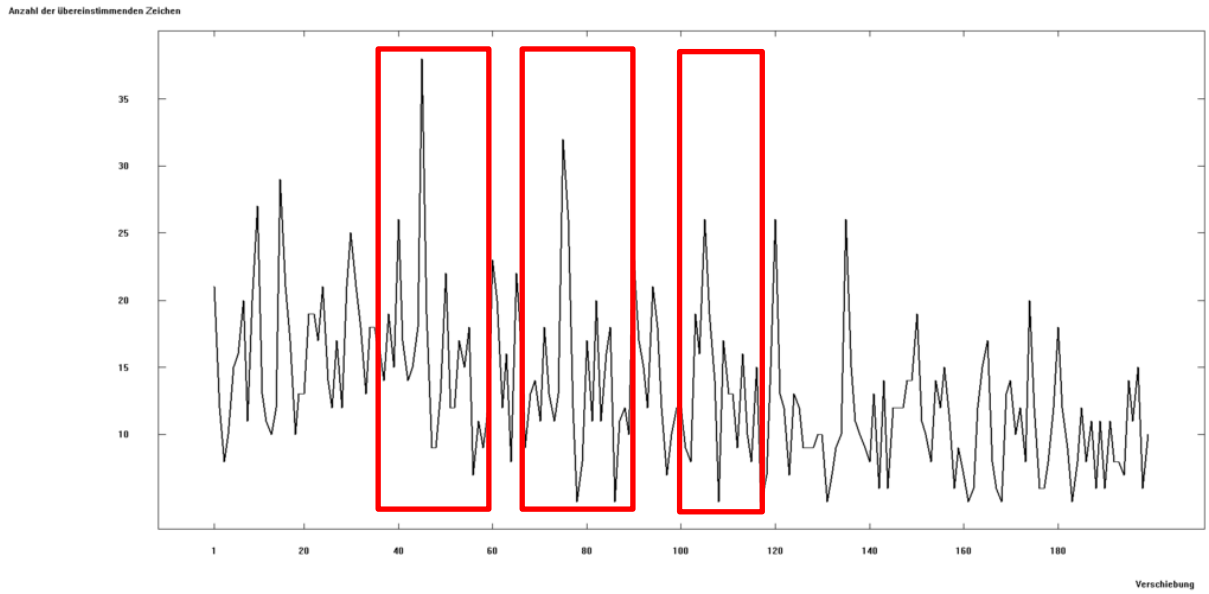


Figure 3 Autokorrelation des verschlüsselten Textes, Bsp. 2 - Giraffen

5. Autoregressive to anything [bis 18.11.2017]

Dieses Kapitel befasst sich mit dem ARTA-Prozess, welcher die Grundlage des Projektes darstellt. Anhand mathematischer und graphischer Elemente sollen die mitwirkenden Komponenten veranschaulicht werden. Folgende Grafik bildet die einzelnen Bestandteile des ARTA-Prozesses ab. In den folgenden Kapiteln wird auf die grundlegenden Elemente eingegangen.

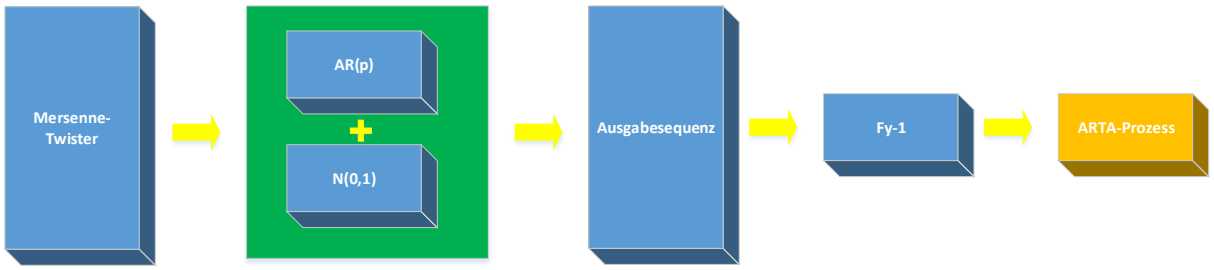


Figure 4 Grafische Darstellung der Bestandteile eines ARTA-Prozesses

5.1 Zufallszahlen – Mersenne-Twister

Der ARTA-Prozess benötigt eine Inputsequenz. Diese entstammt aus einem Zufallszahlengenerator. Die Generierung der Zufallszahlen basiert auf dem Algorithmus des Mersenne-Twister, entwickelt von Makoto Matsumoto und Takuji Nishimura, 1997. Der Algorithmus existiert in zwei Varianten, die hier eingesetzte wird MT 19937 genannt. Die andere Variante wird TT8800 genannt, arbeitet grundsätzlich nach dem gleichen Prinzip, kann jedoch nur eine kleinere Datenmenge verarbeiten. Zusätzlich ist seine Periode kleiner.

Mersenne-Twister weist drei Eigenschaften auf, welche ihn für die vorliegende Implementation qualifizieren.

1. Er weist eine extrem lange Periode auf. Dies ist ein Kriterium, welches die Güte des Generators beschreibt. Die Periodenlänge des Mersenne-Twister beträgt $p = 2^{19937} – 1$ (Mersenne-Primzahl).
2. Alle Werte bzw. Bits der Ausgabesequenz sind hochgradig gleichverteilt. Im Fall des Mersenne-Twister erfolgt diese Verteilung bis zur 623 Dimension[7]. Daraus resultiert eine extrem geringe Korrelation zwischen den aufeinanderfolgenden Zufallszahlen.
3. Der Algorithmus ist schnell. Eine Ausnahme bilden hier Rechenarchitekturen bzw. -Systeme, welche nur über einen sehr begrenzten Arbeitsspeicher verfügen.

ARTA.Standard implementiert den Mersenne-Twister innerhalb der Klasse MersenneTwister. Im folgenden Abschnitt wird anhand des Codes die Funktionsweise des zugrundeliegenden Algorithmus erklärt.
Die Grundlage bildet eine Zahlensequenz. Die Startwerte liegen bei Y_1 bis Y_N, wobei N = 624.Die ersten 624 Werte sind im Idealfall echte Zufallszahlen, jedoch funktioniert der Algorithmus auch mit Pseudozufallszahlen. ARTA.Standard erzeugt diese Zufallszahlen innerhalb der Klasse RandomSource, wobei es sich in diesem Fall lediglich um Pseudozufallszahlen handelt. Die weiteren Werte mit N > 624 werden folgendermassen berechnet:

$$h = Y_{i-N} - Y_{i-N} \bmod 2^{31}\ Y_{i-N+1} \bmod 2^{31}$$
$$Y_i = Y_{i-227}\ \text{XOR}\ h/2\ \text{XOR}\ ((h \bmod 2) * 0x9908B0DF)$$

[7] N-Dimensional: Wird die Ausgabesequenz in Tupel von je n Zahlen zerlegt, so sind diese gleichverteilt im n-dimensionalen Raum.

Abschliessend wird ein Tempering durchgeführt, dadurch wird die Gleichverteilung der Zufallszahlen sichergestellt.

Mersenne-Twister Algorithmus (Tempering)	Implementation

$X = Y_i \text{ XOR } Y_i / 2^{11}$
$Y = x \text{ XOR } ((x * 2^7) \& 0x9D2C5680)$
$Z = y \text{ XOR } ((y * 2^{15}) \& 0xEFC60000)$
$Z_i = z \text{ XOR } z / 2^{18}$
	`x ^= y >> 11;`
	`y = y ^ (y << 7 & - 0x9D2C5680;`
	`z ^= y << 15 & - 0xEFC60000;`
	`z ^= z >> 18;`
	`return z;`

5.2 Zeitreihen / AR-Prozesse

Eine Zeitreihe[8] beschreibt eine Sequenz von Werten, welche sich an eine bestimmte Struktur halten. Diese Struktur wird durch einen Zeitkoeffizienten definiert. Die einzelnen Werte sind an den entsprechenden Zeitpunkt gebunden. Folgendes Beispiel soll die Grundidee einer Zeitreihe verdeutlichen.

$$Y_t = \frac{1}{2^t}$$

t - Werte	0	1	2	3	4	5
$Y_t = \frac{1}{2^t}$	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$

Tabelle 2 Beispiel Zeitreihe

Die Werte der Zeitreihe Y weisen zum Zeitpunkt t immer die Hälfte des vorhergegangenen Wertes des Zeitpunktes t – 1 auf.
Bei einem AR-Prozess muss der Wert zum Zeitpunkt t nicht nur vom Wert des Zeitpunktes t – 1 abhängen, sondern es ist denkbar, dass er auch vom Wert des Zeitpunktes t – 2 abhängt. Solche autoregressiven Prozesse werden in folgendermassen beschrieben: AR(p). Der Parameter p gibt dabei die höchste zeitliche Verzögerung (Lag) an. Beim obigen Beispiel ist dieser Lag gleich 1. Daher kann die Zeitreihe als AR(1) beschreiben werden.

Ein ARTA-Prozess modelliert eine stationäre Zeitserie. Die Basis bildet dabei ein stationärer, autoregressiver Gaussprozess (AR). Die Werte einer autoregressiven Zeitreihe hängen nicht systematisch vom vorhergegangen Werte ab, sondern können auch von Werten zu einem früheren Zeitpunkt abhängen. Der ARTA-zugrundeliegende AR-Prozess ist folgendermassen definiert:

$$\mathbf{AR(p) = \{Z_t; t = 1, 2, ..., n\} \text{ wobei } Z_t = \alpha_1 Z_{t-1} + \alpha_2 Z_{t-2} + \alpha_p Z_{t-p} + \varepsilon_t}$$

Z_t definiert den stationären AR(1)-Prozess, $\boldsymbol{\varepsilon_t}$ steht für zufällige, unabhängige Zufallsvariable einer Normalverteilung N(0, 1). Die Varianz wird so angepasst, dass ein entsprechende Prozess Z_t generiert werden kann.
Folgendes Codefragment zeigt die Berechnung des nächsten Sequenzgliedes eines AR(p)-Prozesses auf. «whiteNoseProcess» beschreibt hierbei die Normalverteilung N(0,1) bzw. $\boldsymbol{\varepsilon_t}$ in obigem Beispiel.

```
public double Next()
{
        double value = whiteNoiseProcess.sample();
        for(int i = 0; i < alphas.Length; i++) {
                value = value + alphas[i] * values.get(i);
        }
        values.add(value);
return value;
}
```

Codefragment 1 AR-Prozess - Next()-Methode

[8] Quelle: Zeitreihenanalyse- Einstieg und Aufgaben von Thomas Mazzoni, FernUniversität in Hagen

TODO: Vertiefung Normalverteilung **[Varianz] = 1 – $\alpha_1 r_1$ – $\alpha_2 r_2$ - … -$a_p r_p$**

wobei r_h die angestrebte Autokorrelation für den Lag h darstellt. Nun kann der Output des AR(p)-Prozesses durch die CDF (Cumulative Distribution Function) in gleichmässig verteilte Werte transformiert werden. Wird nun die Inverseverteilungsfunktion auf die sich ergebenden Werte angewendet, führt dies zu einem Prozess mit der gewünschten Randverteilung.

5.3 Verteilungen

[TODO] Genaue Beschreibung zu den einzelnen Verteilungen, Überarbeitung von UniformDistribution, neuer Termin 15.11.2017

5.3.1 Normalverteilung

5.3.2 Empirische Verteilung

5.3.3 UniformDistribution

Eine UnifromDistribution (stetige Gleichverteilung) beschreibt eine stetige Wahrscheinlichkeitsverteilung. Dies bedeutet, dass Werte auf einem Intervall eine konstante Wahrscheinlichkeitsdichte aufweisen. Demnach ist gegeben, dass alle gleichlangen Teilintervalle ebenfalls dieselbe Wahrscheinlichkeit besitzen.

5.4 ARTA und Autokorrelation [bis 01.11.2017]

Dem AR-Prozess liegt eine natürliche autokorrelierte Struktur zugrunde. Diese ist durch den Lag (Zeitverzögerung), welche durch den Parameter p ausgedrückt wird, gegeben. Die Herausforderung liegt nun darin, diese Autokorrelation auf den darüberliegenden ARTA-Prozess zu transformieren. Um dies zu bewerkstelligen wird auf die Yule-Walker-Methode zurückgegriffen.

5.4.1 Yule-Walker-Gleichungen

Durch eine Yule-Walker-Gleichung kann die Ordnung und die korrespondierenden Korrelationskoeffizienten eines AR-Prozesses identifiziert werden. Dieser Vorgang wird durch die Klasse OrderEstimator übernommen.

5.4.2 PearsonsCorrelation [bis 1.11.2017]

Die Klassen AutoCorrelation und PearsonsCorrelation übernehmen die Funktion zur Errechnung der Korrelationskoeffizienten sowie der Erzeugung der Korrelationsmatrizen. Durch die Formel von Pearson kann leicht der Korrelationskoeffizient r zwischen zwei Variablen ermittelt werden. Folgendes numerische Beispiel soll dies verdeutlichen.

$$r = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{n}}{\sqrt{\left(\sum X^2 - \frac{(\sum X^2)}{n}\right)\left(\sum Y^2 - \frac{(\sum Y^2)}{n}\right)}}$$

X – Werte	1	2	5	5	8
Y – Werte	1	3	4	4	4

Mithilfe der obenstehenden Formel können folgende Werte errechnet werden:

Ausdruck	Bedeutung/numerischer Wert
$\sum XY$	(1)(2) + (3)(5) + (4)(5) + (4)(8) = 69
$\sum X$	1 + 3 + 4 + 4 = 12
$\sum Y$	2 + 5 + 5 + 8 = 20
$\sum X^2$	$1^2 + 3^2 + 4^2 + 4^2 = 42$
$\sum Y^2$	$2^2 + 5^2 + 5^2 + 8^2 = 118$
n	Anzahl der Werte/ Länge der Zahlenreihe

Werden die errechneten Werte nun in die Gleichung eingesetzt, kann der Pearson-Korrelationskoeffizient ermittelt werden:

$$r = \frac{69 - \frac{12*20}{4}}{\sqrt{\left(42 - \frac{(12)^2}{4}\right)\left(118 - \frac{(20)^2}{4}\right)}} = 0.866$$

Das folgende Codefragment zeigt die Berechnung der Korrelationskoeffizienten, so wie sie in dieser Arbeit umgesetzt ist. Weiter übernehmen diese Klassen die Erzeugung der partiellen Korrelationskoeffizienten und der Korrelationsmatrizen.

```
public static double[] CalculateAcfs(double[] data, int maxLag)
{
        double[] accs = new double[maxLag + 1];
        for (int lag = 0; lag <= maxLag; lag++) {
                accs[lag] = CalculateAcf(data, lag);
        }
return accs;
}
```

Codefragment 2 Berechnung der Korrelationskoeffizienten

6. ARTA.Standard [bis 15.11.2017]

ARTA.Standard soll einerseits eine Klassenbibliothek als Grundlage der Modellierung stochastischer Prozesse darstellen, andererseits die Möglichkeit zur Integration in die Simulationssoftware Simio bereitstellen. Auf en folgenden Seiten sind die einzelnen, relevanten Klassen und Algorithmen dargelegt, welche essentiell zur Realisierung beitragen.

6.1 Domain-Modell

[TODO] Domainmodell aus SAD einfügen und entsprechend beschreiben

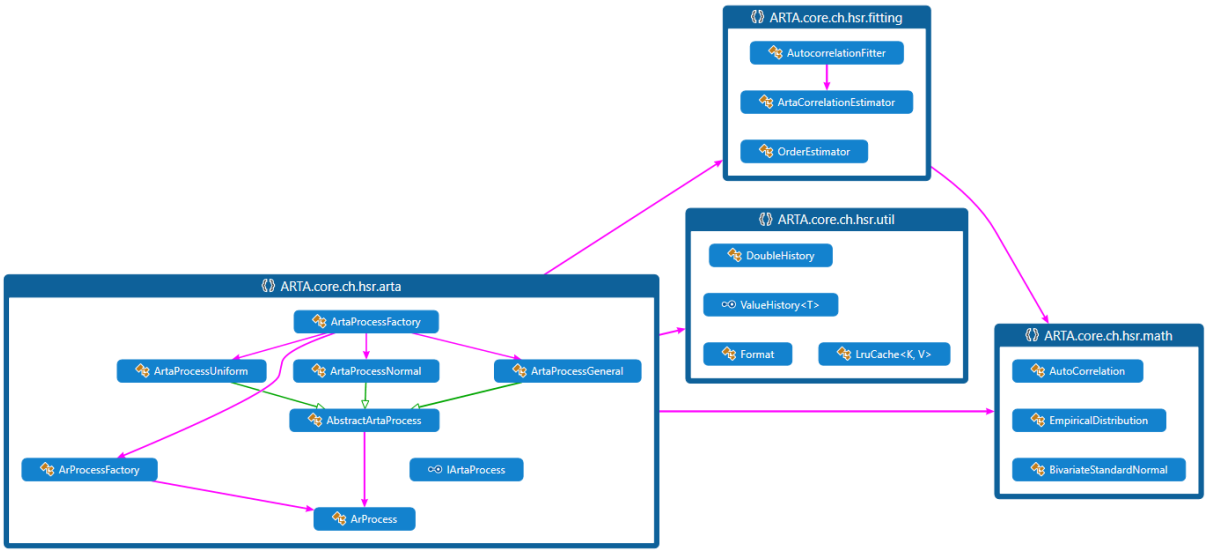


Abbildung 3 Klassendiagramm ARTA.Standard

6.2 Implementation

ARTA.Standard greift auf die Sammlung mathematischer Funktionen und Klassen der MathNet.Numerics[9]-Library zurück. Diese stellt eine Vielzahl an ausgewählten Klassen und Funktionen bereit, welche zur Modellierung des ARTA-Prozesses essentiell sind.
Die Kernkomponente liefern die beiden Klassen ArtaProcessFactory und ArProcessFactory, welche den ARTA-Prozess und den darunterliegenden AR(p)-Prozess erzeugen. Die ArProcessFactory erzeugt den AR-Prozess mithilfe eines Zufallszahlengenerators (hier Mersenne-Twister) und gegebenen Autokorrelationskoeffizienten. Somit kann der Grad der Autokorrelation entsprechend frei gewählt werden, solange die Koeffizienten in den entsprechenden Wertebereichen liegen. Die ArtaProcessFactory nimmt den erzeugten AR-Prozess und eine Randverteilung entgegen um den entsprechenden Prozess zu erzeugen.
[TODO] während Implementationsphase konstant erweitern

[9] https://numerics.mathdotnet.com/
https://github.com/mathnet/mathnet-numerics

Folgendes Codefragment (Auszug aus ArProcessFactory.cs) zeigt die Erzeugung eines neunen AR-Prozesses.

```
///<summary>
///Erzeugt einen AR-Prozess mit den gegebenen Korrelationskoeffizienten.
///Passt die Alpha-Werte in eine Normalverteilung ein, mit dem Mittelwert 0 und der Varianz kleiner 1
///</summary>
public static ArProcess CreateArProcess(double[] arAutocorrelations, RandomGenerator rng)
{
        //Erzeugt eine Korrelationsmatrix und gibt die Reihe mit Index 0 als double[] zurück
        double[] alphas = ArAutocorrelationsToAlphas(arAutocorrelations);

        /*
        Errechnet die Varianz aus den gegebenen Korrelationskoeffizienten und den erzeugten Alpha-
        Werten
        */
        double variance = CalculateVariance(arAutocorrelations, alphas);

        /*
        Erzeugt eine Normalverteilung der zufällig erzeugten Werte des Zufallszahlen-generators,
        untere  Grenze 0.0, obere Grenze @variance. Wendet die Umkehrfunktion der Normalverteilung    an
        um die gewünschte Randverteilung zu erhalten.
        */
        NormalDistribution whiteNoiseProcess = new NormalDistribution(rng, 0.0, Math.Sqrt(variance),
                                                NormalDistribution.DEFAULT_INVERSE_ABSOLUTE_ACCURACY);

return new ArProcess(alphas, whiteNoiseProcess);
}
```

Codefragment 3 ArProcessFactory.CreateArProcess()

Auf der Basis des erzeugten AR-Prozesses kann die ArtaProcessFactory den entsprechenden ARTA-Prozess instanziieren.

6.3 Statistische Tests

Tests werden in einem separaten Assembly «StatisticalTests» abgebildet. Dabei handelt es sich lediglich um Tests der Klassenbibliothek an sich. Die Integration in Simio wird separat in Form eines Integrationstestes und verschiedener Szenarien getestet.

6.3.1 Durbin-Watson-Test - Implementation

Das Ziel der hier dargestellten statistischen Tests liegt darin, die Autokorrelation von den durch ARTA.Standard erzeugten Zufallszahlen nachzuweisen. Um dies zu beweisen, wird auf den Durbin-Watson-Test zurückgegriffen, welcher in der entsprechenden Klasse abgebildet ist.
[TODO] Codefrgmente aus DurbinWatson.cs einfügen

6.3.2 ARTAProcess Tests

Weitere Tests sollen die Vollständigkeit und Funktionalität des abgebildeten ARTA-Prozesses abdecken. Dazu werden verschiedene ARTA-Prozesse mit verschiedenen Parameter erzeugt und anschliessend geprüft, ob die resultierenden Werte den Erwartungen entsprechen. Diese Tests decken ebenfalls die verschiedenen Verteilungen ab. *[TODO] Sobald vollständige Tests vorhanden, Codefragment ergänzen*

```
RealDistribution distribution = new ExponentialDistribution(1.0);
double[] artaCorrelationCoefficients = { 0.3, 0.3, -0.1 };
IArtaProcess arta = ArtaProcessFactory.CreateArtaProcess(distribution, artaCorrelationCoefficients);

double[] data = new double[10000];
for (int i = 0; i<data.Length; i++) {
        data[i] = arta.Next();
}
int maxLag = 10;
double[] acfs = AutoCorrelation.CalculateAcfs(data, maxLag);
double[] pacfs = AutoCorrelation.CalculatePacfs(acfs);
```

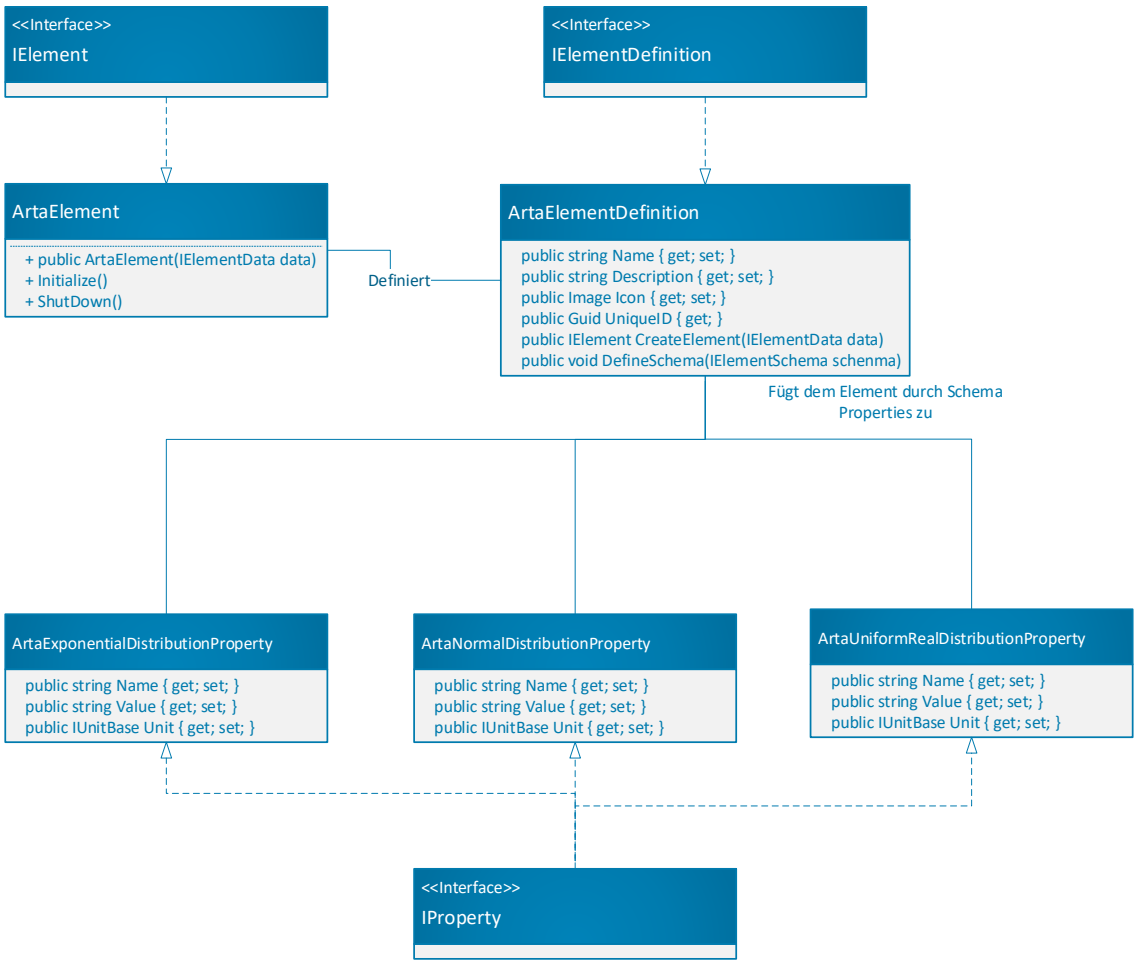
Codefragment 4 Beispiel eines Tests der ARTAProcessFactory

6.3.3 Grenzen von ARTA

Ein weiterer Aspekt soll die Grenzen von ARTA aufzeigen. Damit ist gemeint, dass auf die angegebenen Schwächen, welche im Dokument «JARTA — A Java library to model and fit Autoregressive-To-Anything processes[10]» genannt sind. *[TODO] Weiterführen, Grenzen aufzeigen*

6.4 Integration Simio

Die Integration in die Simulationssoftware Simio ist im Assembly «Arta.Simio» umgesetzt. Die Grundlage bildet ein von Simio bereitgestelltes Visual-Studio-Template. Dieses gibt die Grundstruktur entsprechend vor. Für die Implementation wurde das Template «User-AddIn» verwendet.
Innerhalb der Klasse *[Classname]* wird ein ArtaElement erzeugt. Das ArtaElement enthält Properties welche später den Arta-Prozess definieren (Korrelationskoeffizienten). Weiter sind drei spezifische Properties implementiert, welche die jeweiligen Verteilungen bereitstellen.
Innerhalb von Simio kann nun ein ArtaElement erzeugt werden und dies als InterarrivalTime-Property einer Source übergeben werden.
[TODO]Pro/Kontra des UserAddins
[TODO]Verifikation, dass richtige Zeitabstände genommen werden.
[TODO] während Implementation genauer beschreiben. Termin 15.11.2017
[TODO] Typisierung dokumentieren, verschiedene Properties aufzeigen
[TODO] Bis 13.11.2017 Klassendiagramm Arta.Simio verbessern



[10] JARTA — A Java library to model and fit Autoregressive-To-Anything processes, 10.1109/WSC.2013.6721508

7. Test und Auswertung [[bis 25.11.2017]

7.1 Simulationsumgebung

7.2 Eigene Simulation

[TODO] geeigneten Titel finden

7.3 Resultate

8. Anwendungsfall und Simulation [bis 13.12.2017]

9. Fazit und Ausblick [bis 20.12.2017]

10. Literaturverzeichnis und Referenzen

11. Abbildungsverzeichnis

12. Codefragmente