Entwicklung einer Klassenbibliothek zur Erzeugung autokorrelierter Zufallszahlen

SAD – Software Architecture Documentation

Abteilung Informatik

Hochschule für Technik Rapperswil

|  |
| --- |
| Herbstsemester 2017 |

Autor(en): Anthony Delay

Philipp Bütikofer

Betreuer: Prof. Dr. Andreas Rinkel

Lukas Kretschmar

# Inhalt

[1. Einführung 3](#_Toc497306342)

[1.1 Zweck 3](#_Toc497306343)

[1.2 Gültigkeitsbereich 3](#_Toc497306344)

[1.3 Referenzen 3](#_Toc497306345)

[2. .NET Core – Architektonische Ziele und Einschränkungen 4](#_Toc497306346)

[2.1 Technische Plattform 4](#_Toc497306347)

[2.2 Performance 4](#_Toc497306348)

[2.3 Statistische Tests 4](#_Toc497306349)

[2.4 Simio Integration 4](#_Toc497306350)

[3. Namespaces (Systemübersicht) 5](#_Toc497306351)

[4. Assembly: ARTA.core 6](#_Toc497306352)

[4.1 Beschreibung 6](#_Toc497306353)

[4.2 Hsr.Arta 7](#_Toc497306354)

[4.2.1 Beschreibung 7](#_Toc497306355)

[4.2.2 Factory Pattern 7](#_Toc497306356)

[4.3 Hsr.Fitting 8](#_Toc497306357)

[4.3.1 Beschreibung 8](#_Toc497306358)

[4.4 Hsr.Test 8](#_Toc497306359)

[4.4.1 Beschreibung 9](#_Toc497306360)

[4.5 Hsr.Util 9](#_Toc497306361)

[4.5.1 Beschreibung 9](#_Toc497306362)

[4.6 Hsr.Math 9](#_Toc497306363)

[4.6.1 Beschreibung 9](#_Toc497306364)

[5. Assembly:ARTA.simio 9](#_Toc497306365)

[5.1 Beschreibung 9](#_Toc497306366)

[6. Assembly: ARTA.statistics 9](#_Toc497306367)

[6.1 Beschreibung 10](#_Toc497306368)

# Einführung

## Zweck

Dieses Dokument zeigt die Architektur über die Studienarbeit «Entwicklung einer Klassenbibliothek zur Erzeugung von autokorrelierten Zufallszahlen» auf.

## Gültigkeitsbereich

Dieses Dokument ist im Rahmen der Studienarbeit 2017 «Entwicklung einer Klassenbibliothek zur Erzeugung autokorrelierter Zufallszahlen» gültig.

## Referenzen

|  |  |
| --- | --- |
| **Bezeichnung** | **Link** |
| Apache Math3 Library | <http://commons.apache.org/proper/commons-math/index.html> |
| .Net Numeric Library | <https://numerics.mathdotnet.com/> |
|  |  |

# .NET Core – Architektonische Ziele und Einschränkungen

Dieser Abschnitt beschreibt Anforderungen und Einschränkungen, welche signifikanten Einfluss auf die Architektur von «Arta.Core» mit sich bringen.

## Technische Plattform

Das Fundament soll «.Net Standard» bilden. Ziel ist es, eine möglichst schlanke Klassenbibliothek zu erzeugen. Diese soll möglichst auf allen Plattformen einsetzbar sein. «.Net Standard» stellt eine modulare Version des «.Net Framework» dar, welche auf plattformübergreifende Portierbarkeit ausgelegt ist.

JARTA referenziert und nutzt einige externe mathematische Bibliotheken von Apache. Viele dieser Klassen werden für JARTA selbst nicht benötigt und vergrössern den Umfang der Applikation unnötig.

Die Bibliothek von Apache lässt sich nicht für die .Net-Umgebung benutzen, deswegen wurden Alternativen gesucht und mit «Math.Net Numerics» gefunden. Diese Bibliothek ist nicht identisch zur Apache Bibliothek und es mangelt teilweise an notwenigen Funktionen. Die fehlenden oder divergenten Klassen werden durch Selbstverfasste ergänzt, um die volle Funktionalität des JARTA Projektes zu gewährleisten.

## Performance

Dem Projekt sind keine Performance-Constraints gesetzt. Wichtig sind die Qualität Aspekte, der Output dieser Arbeit soll dem Output des ARTA Prozesses entsprechen, die Laufzeit ist vorerst unwesentlich. Es wird aber versucht das Projekt möglichst schlank zu halten und mittels «.Net.standard» flexibel in der Wahl der Plattform zu sein.

## Statistische Tests

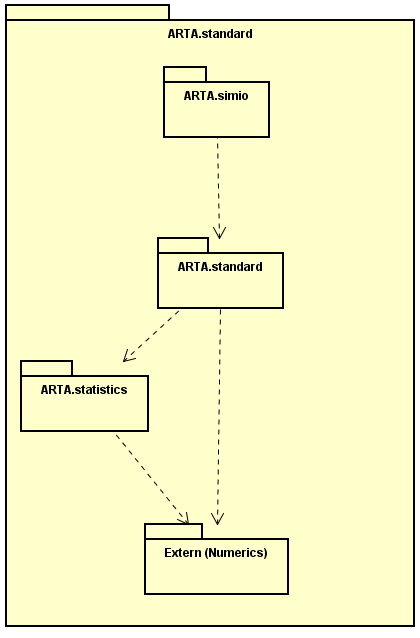
Dieses Thema wird erst in der Konzeption 2 behandelt.

## Simio Integration

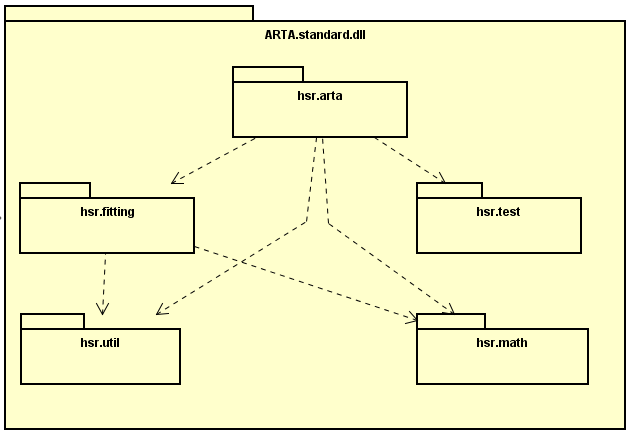
Dieses Thema wird erst in der Konzeption 2 behandelt.

# Namespaces (Systemübersicht)

Das Endprodukt beinhaltet vier Assemblies (siehe Grafik) und referenziert die Bibilothek «Math Numerics» von .Net. Die einzelnen Assemblies werden weiter unten detaillierter beschrieben.



# Assembly: ARTA.core

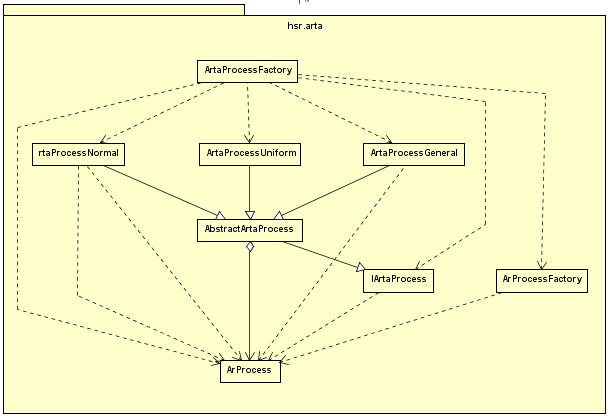


## Beschreibung

Dieses Assembly enthält die nötigen Klassen und Interfaces um den ARTA-Prozess korrekt abzubilden. Das Assembly ist wiederrum in mehrere Folder unterteilt.

Die Struktur orientiert sich stark an der des JARTA Codes und wurde wenn möglich gleich übernommen.

## Hsr.Arta



### Beschreibung

Dieses Subsystem kümmert sich ausschliesslich um den ARTA Prozess. Mittels zwei Factory Pattern werden Autoregressive- und ARTA Prozesse kreiert. Dieses Pattern findet in Klassenbibliotheken oft gebrauch, da das Projekt flexibler und ausbaufähiger wird.

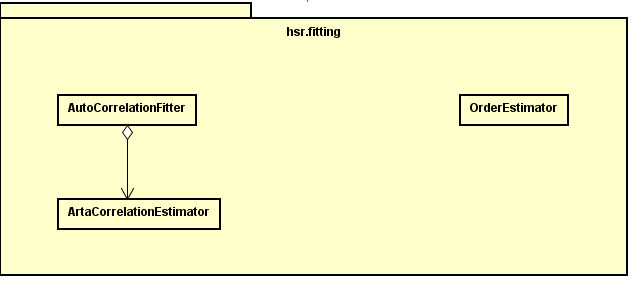
Der ARTA Prozess unterscheidet zwischen drei Distributionen:

* Normal
* Uniform
* General

Um einen ARTA Prozess zu erzeugen muss der Korrelationskoeffizient «(-1;1),(-1,1)» und die Distribution der Methode «CreateArtaProcess» übergeben werden. Der ArtaProcess generiert die Zufallszahlen einzeln mittels der Methode «next()».

### Factory Pattern

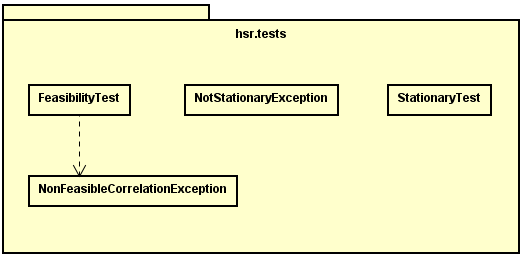
## Hsr.Fitting



### Beschreibung

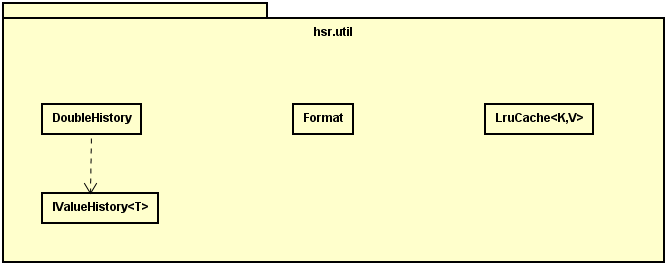
Dieses Subsystem bietet Methoden zur Schätzung der erwarteten Autokorrelationen eines ARTA-Prozesses, des zugrundeliegenden autoregressiven Prozesses und der im ARTA-Prozess verwendeten Verteilung. Diese Schätzung beruht auf der Gauss-Kronrod-Quadraturregel.

## Hsr.Test



### Beschreibung

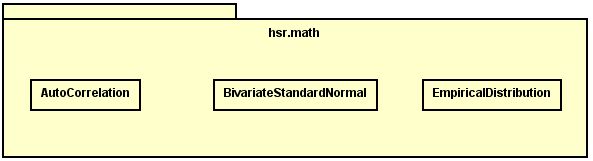
## Hsr.Util



### Beschreibung

Dieses Subsystem bietet hilfreiche Klassen die für den ARTA Prozess nötig sind. Die Klasse Format bietet zum Beispiel diverse Konstanten und Formatvorlagen für Darstellungen von Datentypen. Die DoubleHistory imitiert ein Stack und verwaltet jeweils einen Double Array.

## Hsr.Math



### Beschreibung

Dieses Subsystem bietet weitere mathematische Klassen an, die in der externen Bibliothek nicht vorhanden sind aber für den ARTA-Prozess unabdingbar sind.

# Assembly:ARTA.simio

Dieses Assembly regelt die Integration in Simio und wird erst in der Konzeptionsphase 2 bearbeitet.

## Beschreibung

# Assembly: ARTA.statistics

Dieses Assembly prüft den Output von ARTA.core und wird erst in der Konzeptionsphase 2 bearbeitet.

## Beschreibung