AK VRZ

Verkehrsrechnerzentralen

Technische Anforderungen

Gesamtsystem

Ersteller: Kappich+Kniß

Autor: Dipl.-Ing. H.C. Kniß
Dipl.-Inform. R. Schmitz
Dipl.-Ing. C. Westermann

Version: 2.0
Stand 01.03.2018
Status: akzeptiert
PID:
Submodell: SE-02.00.00.00.00-TAnf-1.0
Dokument: Systementwicklung
VS-Einstufung: SE-02.00.00.00.00-TAnf-1.0 [Technische Anforderungen].doc
---

Projekt ID AG: AK VRZ
Projekt ID AN: BAST.06.006 / BAST.07.007

Kappich+Kniß Systemberatung Verkehr und Technik

Im Auftrag der Bundesanstalt für Straßenwesen BASt

1 Allgemeines

Verteilerliste

Entfällt. Dokumentverteilung entsprechend aktuellem Projektverteiler.

Versionsübersicht

Nr.	Datum	Version	Änderungsgrund	Bearbeiter
1	25.10.04	0.1	Ersterstellung	Kniß Schmitz Westermann
2	28.01.05	0.2	Ergänzungen	Westermann
3	14.02.05	0.3	Änderung entsprechend PAWG vom 11.02.2005	Kniß
4	16.09.05	0.4	Überarbeitung	Westermann
5	21.10.05	0.5	Überarbeitung	Westermann
6	12.04.06	1.0	Überführung in den Zustand "akzeptiert"	Westermann
7	16.02.18	1.1	Einarbeitung Zertifizierung 2017	Copei
8	01.03.18		Überführung in den Zustand "akzeptiert"	Kniß

Tabelle 1-1: Versionsübersicht

Änderungsübersicht

Nr.	Version	geändertes Kapitel	Beschreibung der Änderung
1	0.1	alle	Ersterstellung
2	0.2	3	Kapitel"Allgemeine Anforderungen an die Persistierung von Daten" ergänzt.
3	0.2	4	Kapitel "Betriebsmeldungen" ergänzt.
4	0.3	3 4	Kapitel 3 "Allgemeine Anforderungen an die Persistenz von Anwendungsdaten" und Kapitel 4 "Betriebsmeldungen" detailliert.
5	0.4	3	Kapitel 3 "Allgemeine Anforderungen an die Persistenz von Anwendungsdaten" überarbeitet
6	0.5	3	Klargestellt, dass die Modellierung durch den Applikationsentwickler und nicht durch die Applikation erfolgen muss
7	1.1	2, 5	Ergänzung der Mindeststandards gemäß der Zertifizierung sowie Erwähnung der BetrInfo für Ausgaben

Tabelle 1-2: Änderungsübersicht

Kurzbeschreibung

In diesem Produkt werden die allgemeinen technischen Anforderungen genannt, die keinem Element der technischen Architektur zu den in Tabelle 1-2 aufgeführten Segmenten gesondert zugeordnet werden können. Weiter werden in diesem Produkt die technischen Anforderungen an das Gesamtsystem definiert.

Damit werden in diesem Produkt die Allgemeinen und technischen Anforderungen an das Gesamtsystem abgedeckt, auf die in den technischen Anforderungen an die in Tabelle 1-2 aufgeführten Segmenten und deren SW-Einheiten verwiesen wird.

In Tabelle 1-2 sind alle Segmente mit ihren Nummern und Bezeichnungen aufgeführt. Die Detaillierung der technischen Anforderungen an die einzelnen Segmente sowie deren SW-Einheiten/HW-Einheiten erfolgt in den durch die PID referenzierten Dokumente.

Segment			
Nr.	Bezeichnung	Kürzel	PID
1	Datenverteiler	DaV	SE-02.01.00.00.00-TAnf
2	Kommunikation mit externen Stellen	KEx	SE-02.02.00.00.00-TAnf
3	Archivsystem	ArS	SE-02.03.00.00.00-TAnf
4	Datenübernahme und Aufbereitung	DUA	SE-02.04.00.00.00-TAnf
5	Intelligente Analyseverfahren	IAV	SE-02.05.00.00.00-TAnf
6	Intelligente Bewertungsverfahren	IBV	SE-02.06.00.00.00-TAnf
7	Steuerung	Ste	SE-02.07.00.00.00-TAnf
8	Parametrierung und Konfiguration	PuK	SE-02.08.00.00.00-TAnf
9	Protokolle und Auswertungen	PuA	SE-02.09.00.00.00-TAnf
10	System	Sys	SE-02.10.00.00.00-TAnf
11	Verwaltung	VeW	SE-02.11.00.00.00-TAnf
12	SWPÄ-Tools	PAT	SE-02.12.00.00.00-TAnf
13	Bedienung und Visualisierung	BuV	SE-02.13.00.00.00-TAnf

Tabelle 1-2: Segmentübersicht

Inhalt

Abkürzungen

siehe Dokument „Abkürzungen“.

Definitionen

siehe Dokument „Glossar“.

Verzeichnis der Tabellen

Referenzierte Dokumente

[Afo]	Anwenderforderungen AK VRZ, Dokument „SE-02.00.00.00.00-Afo“, aktueller Stand
[SysArc]	Systemarchitektur AK VRZ, Dokument „SE-02.00.00.00.00-SysArc“, aktueller Stand
[TAnfDaV]	Technische Anforderungen an den Datenverteiler AK VRZ, Dokument „SE-02.01.00.00.00-TAnf“, aktueller Stand
[TAnfPuK]	Technische Anforderungen an Parametrierung und Konfiguration AK VRZ, Dokument "SE-02.08.00.00.00-TAnf", aktueller Stand
[TAnfSys]	Technische Anforderungen an das System AK VRZ, Dokument „SE-02.10.00.00.00-TAnf“, aktueller Stand
[TAnfVeW]	Technische Anforderungen an die Verwaltung AK VRZ, Dokument „SE-02.11.00.00.00-TAnf“, aktueller Stand
[SSÜb]	Schnittstellenübersicht AK VRZ, Dokument „SE-02.00.00.00.00-SSÜb“, aktueller Stand
[SSB]	Schnittstellenbeschreibung AK VRZ, Dokument „SE-02.00.00.00.00-SSB“, aktueller Stand
[MARZ]	Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen, Ausgabe 1999.
[VMOD97]	Der Bundesminister des Innern, Entwicklungsstandard für IT-Systeme des Bundes Vorgehensmodell, Juni 1997, KBSt, Koordinations- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung.

2 Anforderungen an die Starterschnittstelle von Applikation

Alle SW-Einheiten, die im Sinne der Festlegungen der Systemarchitektur Applikationen darstellen, müssen bestimmte benannte Aufrufparameter, mit denen sich das (Start-) Verhalten der Applikation in bestimmten Bereichen einstellen lässt, unterstützen, indem sie bestimmter in der Systemarchitektur identifizierte Schnittstellen verwenden.

Spezifische Startparameter sowie detailliertere Beschreibungen des Verhalten der SW-Einheit ist der jeweiligen BetrInfo zu entnehmen.

2.1 Verwendung der Schnittstelle DatenverteilerApplikationsfunktionen-Starter

Durch Verwendung der Schnittstelle DatenverteilerApplikationsfunktionen-Starter stehen der Applikation folgende Aufrufparameter zur Verfügung:

Aufrufparameter zum Aufbau einer Datenverteilerverbindung

-datenverteiler = adresse
: Adresse des Datenverteilers.
Eindeutige Adressenangabe, über die die Datenverteiler-Applikationsfunktionen der Applikation die Kommunikation zum zugeordneten Datenverteiler aufnimmt. Bei Verwendung von TCP/IP wird die Adresseangabe durch die Angabe der IP-Adresse und (durch Doppelpunkt getrennt) der TCP-Portnummer spezifiziert.

-konfigurationsBereich = pid
: Anzubindende Konfiguration
Spezifiziert die Konfiguration für Konfigurationsanfragen und die Datenhaltung für Archivanfragen (optional, wenn kein Wert angegeben wird, wird der Defaultwert des zugeordneten Datenverteilers genommen).

-benutzer = zeichenkette
: Benutzerangabe
Benutzerkennung, die beim Verbindungsaufbau zum Datenverteiler übergeben wird (optional, wenn keine Benutzerkennung angegeben wird, wird diese abgefragt).

-authentifizierung = dateiname
: Authentifizierungsinformation
Pfadangabe zu der Datei, in der das zugehörige Passwort abgelegt ist (optional, wenn keine Angabe erfolgt, wird das Passwort abgefragt).

-authentifizierungsVerfahren = zeichenkette
: Verschlüsselungsverfahren
Klassenname des Verschlüsselungsverfahrens, mit dem der vom Datenverteiler gesendete Zufallstext über das Passwort chiffriert wird (optional, wenn keine Angabe erfolgt, wird das zur Zeit einzig unterstützte HMAC-MD5-Verfahren zur Verschlüsselung genutzt).

Aufrufparameter zur Kommunikationsüberwachung

-timeoutSendeKeepAlive = zahl

Timeout Versand Keep-Alive-Telegramm
Zeit in Sekunden, nach der spätestens ein Keep-Alive-Telegramm versendet wird, wenn keine sonstigen Telegramme während dieser Zeit versendet wurden. Der angegebene Wert dient als Vorschlag für den Datenverteiler, der den zu verwendenden Wert festlegt. (optional, wenn keine Angabe erfolgt, wird ein Default-Wert als Vorschlag an den Datenverteiler übertragen).

-timeoutEmpfangeKeepAlive = zahl

Timeout Empfang Keep-Alive-Telegramm
Timeoutzeit in Sekunden, in der spätestens ein Telegramm empfangen werden muss. Wird diese Zeit ohne Empfang von Telegrammen überschritten, wird die Verbindung zum Kommunikationspartner terminiert. Der angegebene Wert dient als Vorschlag für den Datenverteiler, der den zu verwendenden Wert festlegt. (optional, wenn keine Angabe erfolgt, wird ein Default-Wert als Vorschlag an den Datenverteiler übertragen).

-durchsatzPruefung = zeichenkette

Optionaler Parameter zur Durchsatzprüfung.
Durch Doppelpunkt getrennte Parameter zur Durchsatzprüfung.

-durchsatzPruefung=PufferFüllgrad:PrüfIntervall:MindestDurchsatz

Dabei gibt der Parameter PufferFüllGrad die Belegung des Sendepuffers in Prozent (Default 75 %) an, ab der zyklisch die Durchsatzprüfung erfolgen soll; der Parameter PrüfIntervall gibt das Messinterval in Sekunden (Default 60 Sekunden) an, nachdem der Durchsatz jeweils ermittelt werden soll; der Parameter MindestDurchsatz legt den Schwellwert in Bytes pro Sekunde (Default 3000 Bytes pro Sekunde) fest, mit dem der ermittelte Durchsatz verglichen wird. Ist der ermittelte Durchsatz kleiner als der MindestDurchsatz wird die Verbindung zum Datenverteiler terminiert. Die angegebenen Werte dienen als Vorschläge für den Datenverteiler, der die zu verwendenden Werte festlegt. (optional, wenn keine Angabe erfolgt, werden Default-Werte als Vorschlag an den Datenverteiler übertragen).

Aufrufparameter zur Spezifikation der zu behandelnden Daten

-lokaleSpeicherungKonfiguration = zeichenkette

Speicherungsort der lokalen Konfigurationsinformationen
Über diesen Parameter kann festgelegt werden, ob Konfigurationsinformationen persistent zwischengespeichert werden sollen. Wenn an dieser Stelle ein Verzeichnis angegeben wird, werden die Dateien mit den zwischengespeicherten Konfigurationsinformationen an dieser Stelle abgelegt. Fehlt dieser Parameter erfolgt keine Zwischenspeicherung.

-aspekt zeichenkette

Aspektmanipulation

Die Anmeldung der durch die Applikation benötigten und gelieferten Daten erfolgt über die Angabe der entsprechenden Attributgruppe mit dem dabei betrachteten Aspekt. Diese Angaben sind in den Applikationen als Default vorgegeben. Über diesen optionalen Aufrufparameter kann der Aspekt jeder zu verwendenden Attributgruppe geändert werden. Damit besteht die Möglichkeit, den Datenfluss beim Start einer Applikation zu modifizieren und damit beispielsweise einen weiteren Prozess in eine Bearbeitungskette einzufügen.

Beispiel

Eine Applikation erwartet als Default das Eingangsdatum die Attributgruppe "atg.verkehrswerte" mit dem Aspekt "asp.logischPLgeprueft". Diese Attributgruppe wird z.B. durch die Applikation PL-Prüfung FG 1 geschrieben. Über den Aufrufparameter

-aspekt=atg.verkehrswerte:asp.logischPLgeprueft:asp.formalPLgeprueft

wird eine Änderung des Datenflusses bewirkt. Die Applikation erhält nun als Eingangsdaten die Attributgruppe "atg.verkehrswerte" mit dem Aspekt "asp.formalPLgeprueft", die von einer anderen Applikation (PL-Prüfung formal) geschrieben wird.

-simVariante = zahl

Simulationsvariante

Simulationsvariante für Telegramme, mit der die Applikation arbeitet (optional, wenn die Applikation im Normalbetrieb (keine Simulation) arbeiten soll, muss keine Angabe der Simulationsvariante erfolgen. Der Wert wird implizit auf 0 gesetzt. Durch Angabe eines Wertes im Bereich zwischen 1 und 999 kann die Applikation für eine bestimmte Simulationsvariante gestartet werden).

2.2 Verwendung der Funktionsbibliothek zur Steuerung von Diagnoseausgaben

Durch Verwendung der Funktionsbibliothek stehen der Applikation folgende Aufrufparameter zur Steuerung von Diagnoseausgaben zur Verfügung:

-debugLevelStdErrText = level

Diagnoseausgabe auf die Standardausgabe

Mit diesem Aufrufparameter wird der Level der Debugausgaben auf die Standardausgabe festgelegt. Die möglichen Level sind in Kapitel 2.2.1 "Level der Diagnoseausgaben" aufgeführt.

-debugLevelFileText = level

Diagnoseausgabe in eine Textdatei

Mit diesem Aufrufparameter wird der Level der Debugausgaben in eine Textdatei festgelegt. Die möglichen Level sind in Kapitel 2.2.1 "Level der Diagnoseausgaben" aufgeführt.

-debugLevelFileXML = level

Diagnoseausgabe in eine XML-Datei

Mit diesem Aufrufparameter wird der Level der Debugausgaben in eine XML-Datei festgelegt. Die möglichen Level sind in Kapitel 2.2.1 "Level der Diagnoseausgaben" aufgeführt.

-debugLevelFileExcel = level

Diagnoseausgabe in eine Excel-Datei
Mit diesem Aufrufparameter wird der Level der Debugausgaben in eine Excel-Datei festgelegt. Die möglichen Level sind in Kapitel 2.2.1 "Level der Diagnoseausgaben" aufgeführt.

-debugLevelFileHTML = level

Diagnoseausgabe in eine HTML-Datei
Mit diesem Aufrufparameter wird der Level der Debugausgaben in eine HTML-Datei festgelegt. Die möglichen Level sind in Kapitel 2.2.1 "Level der Diagnoseausgaben" aufgeführt.

-debugFilePath = level

Festlegung des Verzeichnisses für Diagnoseausgaben
Mit diesem Aufrufparameter wird das Verzeichnis festgelegt, in das die Dateien mit den Diagnoseausgaben geschrieben werden. Die Namen der Dateien werden generisch gebildet..

2.2.1 Level der Diagnoseausgaben

Folgende Level für Debugausgaben sind zu unterstützen:

ERROR	FEHLER ist der höchste Level; die Verwendung sollte ausschließlich für schwerwiegende "echte" Fehler erfolgen.
WARNING	Verwendung für Warnungen, die vom Programm zwar noch abgefangen werden können, aber unbedingt behoben werden müssen.
INFO	Verwendung für Infoausgaben (z.B. Status des Programms, verwendete Startparameter etc.).
CONFIG	Verwendung für Konfigurationsinformationen (z.B. angemeldete Objekte etc.).
FINE	Verwendung für programmnahe Ausgaben zur Verfolgung des Programmablaufs.
FINER	Wie bei FINE, aber feinere Ausgabe.
FINEST	Wie bei FINER, aber mit allen Details.
ALL	Schaltet die Ausgabe aller Level ein
OFF	Schaltet die Ausgabe aller Level aus

3 Allgemeine Anforderungen an die Persistierung von Anwendungsdaten

Jede Applikation ist für die Persistenz der zur Initialisierung benötigten Anwendungsdaten selber zuständig.

Abbildung 3-1 skizziert den Datenfluss einer Applikation. Eine Applikation benötigt zur Initialisierung Informationen aus der Konfiguration und Parameter, die z.B. die Algorithmen steuern, wie aus den Eingangsdaten die Ausgangsdaten berechnet werden. Im Betrieb verarbeitet die Applikation Online-Daten; d.h. die Applikation meldet sich als Empfänger für Eingangsdaten und als Quelle für die ermittelten Ausgangsdaten an. Während des Betriebs muss die Applikation auf Parameteränderungen reagieren. Die Parameter werden von der Parametrierung unter dem Aspekt "Soll" verteilt und stehen der Applikation zur Verfügung, die sich auf die von ihr benötigten Parameter anmelden muss.

Sämtliche Daten können vom Archivsystem archiviert werden.

Da ein System aber grundsätzlich ohne Archivsystem arbeiten können muss, dürfen Applikationen die zur Initialisierung benötigten Daten nicht aus dem Archivsystem anfragen[1].

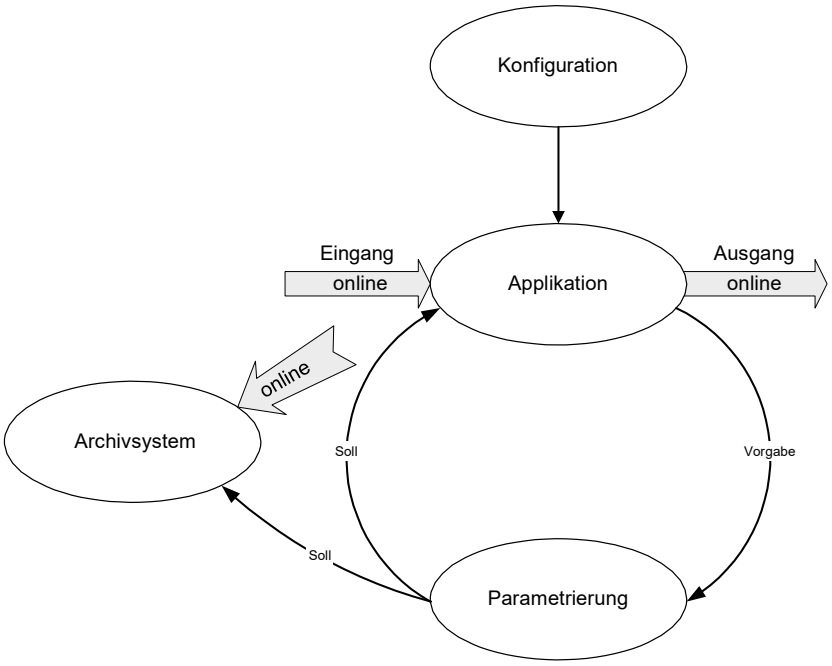

Abbildung 3-1: Datenfluss Applikation

[1] Bestimmte Algorithmen benötigen eine bestimmte Anzahl von Datensätzen, bevor sie Ergebnisse liefern können. Dazu steht über die Applikationsfunktionen eine Methode zur Verfügung, Daten mit Historie anzumelden [TAnfDaV]. Diese Methoden greifen natürlich auf ein Archivsystem zu, aber die Applikation muss gewährleisten, dass sie, auch wenn diese Daten nicht verfügbar sind, fehlerfrei lauffähig ist. Im einfachsten Fall wird die Applikation zunächst bekannt geben, dass sie "keine Daten" liefern kann, bis sie dann den ersten ermittelbaren Ergebnisdatensatz publiziert.

Da auf das Archivsystem nicht zugegriffen werden kann und jedes System über eine Parametrierung verfügen muss, bietet es sich für den Applikationsentwickler an, benötigte Daten, die die Applikation selber erzeugt und die zur Initialisierung benötigt werden (also über einen Neustart persistent sein müssen) als Parameter zu modellieren.

Betrachten wir beispielsweise als Applikation das Automatische Lernen von Ganglinien. Diese Applikation erzeugt auf Basis gemessener Verkehrswerte historische Ganglinien durch Verschmelzung mit der vorher ermittelten Ganglinie. Das bedeutet, dass diese Applikation beim Neustart die jeweils letzte Ganglinie benötigt. Wenn die Ganglinien als Parameter behandelt werden, würde die Applikation jede neue verschmolzene Ganglinie als Vorgabeparameter an die Parametrierung senden. Die Parametrierung verteilt diese unter dem Aspekt Soll an alle Applikationen, die diese Information benötigen (z.B. die Ganglinienprognose). Wenn das Automatische Lernen von Ganglinien neu gestartet werden muss, dann meldet sie sich auf die benötigten die Ganglinien repräsentierenden Parameter unter dem Aspekt Soll an.

Über diesen Mechanismus wird gewährleistet, dass die benötigten initialen Daten beim Neustart der Applikation zur Verfügung stehen.

Abbildung 3-2 skizziert den Datenfluss einer Applikation bei der Simulation mit historischen Daten.

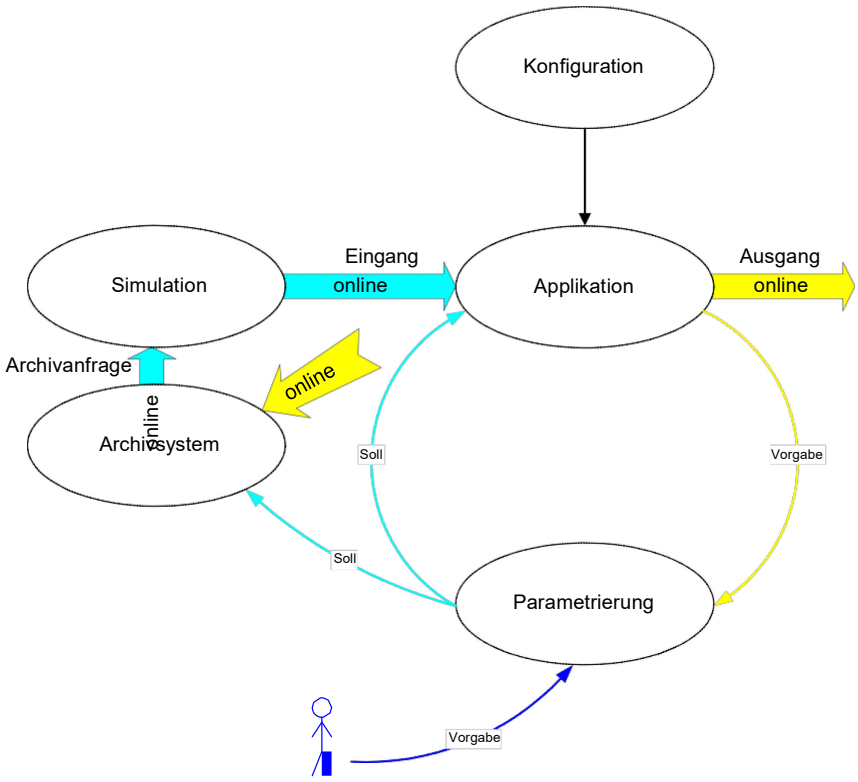

Abbildung 3-2: Datenfluss Applikation bei Simulationen

In diesem Fall werden die zu verarbeitenden Eingangsdaten durch den Simulationsdatengenerator bei dem Archivsystem angefragt und unter der entsprechenden Simulationsvariante eingespeist.

Die Parametrierung stellt die für die Simulation benötigten Parameter unter der entsprechenden Simulationsvariante zur Verfügung [TAnfPuK]. Dabei werden die entsprechenden Parameter

der Standardvariante (normaler Betrieb der Applikationen) auf die Simulationsvariante umkopiert.

Die Parametrierung verwendet für das Umkopieren immer die aktuellen Parameter. Der Benutzer hat aber die Möglichkeit, die Parameter der entsprechenden Variante vor dem eigentlichen Lauf der Simulation anzupassen. Er kann z.B. die zum simulierten Zeitpunkt gültigen Parameter vorgeben.

Bei der Beschreibung einer Simulationsstrecke muss spezifiziert werden, welche Simulationsdaten als Eingangsinformationen (Eingangsdaten und Parameter) benötigt werden [TAnfVeW]. Der Simulationsdatengenerator speist die spezifizierten Eingangsdaten ein und die Parametrierung stellt die angegebenen Parameter zur Verfügung. Damit kann durch die Spezifikation der Simulationsstrecke auch festgelegt werden, welche Parameter für die Simulation nicht als von außen während der Simulation änderbare Parameter sondern als "einfache" Eingangsdaten betrachtet werden sollen. Im Beispiel zum automatischen Lernen der Ganglinien können die historischen Ganglinien auch durch den Simulationsdatengenerator erzeugt werden. Damit würde die Simulation direkt mit den historischen Daten ablaufen.

Persistierung von Daten, die nicht simuliert werden müssen

Daten, die eine Applikation zusätzlich benötigt und die nicht simuliert werden müssen, können durch die Applikation intern vorgehalten werden. Dabei handelt es sich z.B. um Skriptdefinitionen zur Verarbeitung von Eingangsdaten, welche nicht in der Konfiguration oder Parametrierung vorgehalten werden sollen oder andere Informationen zu externen Schnittstellen, von denen klar ist, dass sie nicht für Simulationen benötigt werden.

4 Betriebsmeldungen

Zum Erzeugen von Betriebsmeldungen müssen die Bibliotheksfunktionen der SW-Einheit Funktionsbibliothek aus dem Segment System genutzt werden [TAnfSys].

Bestimmte Meldungen (z. B. DE-Fehlermeldungen, Brandmeldungen) sind zusätzlich über eine weiter Attributgruppen/Aspektkombination als Quelle zu versenden[2].

5 Mindeststandards, Anforderungen und Vorgaben an das System

Bei der Entwicklung einer SWE ist darauf zu achten, dass einzelne Funktionalitäten testbar entwickelt werden. Dies hat zur Folge, dass alle Funktionalitäten über eine Schnittstelle für automatisierte Tests zugänglich sein müssen. Für die konkrete Implementierung bedeutet dies:

- Als Testframework JUnit oder ähnliches verwenden.
- Automatisierte Tests sollten als Benutzerszenario implementiert werden.
- Klassen, Methoden sowie Attribute sollten ausreichend dokumentiert sein.
    - Klassen: Welche Aufgabe erfüllt die Klasse in der Anwendung
    - Methode: Welche Teilaufgabe löst diese Methode im Hinblick auf die Gesamtfunktionalität
    - Attribut: Welche Daten werden hier gespeichert.

Durch die lose Kopplung des Datenverteilers kann jede SWE ersetzt werden, sofern die gegebenen Schnittstellen implementiert sind und die allgemeinen Anforderungen an den Datenverteiler beachtet werden.

[2] Dies muss unabhängig von den Mechanismen der Betriebsmeldungen erfolgen.

Folgende Punkte muss die SWE im Hinblick auf Installierbarkeit erfüllen:

- Installation muss betriebssystemunabhängig sein.
- Eine Installation sollte ohne Rückstände zu entfernen sein.

Eine ausgewählte alternative Software muss alle Funktionen und Anforderungen erfüllen die ihre zu ersetzendes Gegenstück bereits erfüllt. Die genauen Anforderungen können den entsprechenden Dokumenten entnommen werden.

Beim Entwicklungsprozess ist stehts darauf zu achten, dass einzelne Teile der Software modular implementiert werden. Dadurch wird eine einfache Erweiterbarkeit bzw. Austauschbarkeit einzelnen Implementierungen gewährleistet.

Für die Prüfumgebung ist zu beachten, dass einzelne Tests reproduzierbar und deterministisch erstellt werden. Daher sollten Softwaretests mithilfe von JUnit, oder einem ähnlichen Testframework, abgebildet werden.

Beim Entwicklungsprozess ist zu beachten, dass keine plattformspezifischen Funktionen verwendet werden. Da dies im Konflikt mit der Anforderung der Plattformunabhängigkeit stehen würde. Sollte es nicht möglich sein eine solche Funktion zu umgehen muss sichergestellt sein das andere Betriebssysteme diese Funktionen in einer anderen Art ebenfalls implementieren.

Für die Einbindung eines externen Systems sind die Anforderungen der DaV-DaF Kopplung zu beachten. Gemäß dem Dokument [TAnfDaV].