

AnEdge

Werkzeug zur Analyse von TAA-Modulen

Stand 09.10.2007 12:03

Autor: TeamWiSE GmbH

Leerseite

Inhaltsverzeichnis

Inhaltsverzeichnis	3
1. Problemstellung.....	5
2. Zielsetzung AnEdge	6
2.1. Abgrenzung zum EDB-Check.....	6
3. Arbeitsweise	7
3.2. Übersicht	7
3.3. Aufruf von AnEdge.....	7
3.4. Ergebnisse der Durchführung.....	8
4. Darstellung der Analyseergebnisse	10
4.5. Übersicht	10
5. Analysedetails	12
5.6. States (Zustände)	12
5.7. Events (Ereignisse)	13
5.8. ParUse - Verwendung Parameter Objekte	15
5.9. LocUse - Verwendung lokale Objekte.....	16
5.10. GloUse - Verwendung globale Objekte.....	16
5.11. ObjUse - Verwendung Objekte Allgemein.....	17
5.12. ObjMod – Modifikation von Objekten	18
5.13. Cond – Bedingungen.....	20
5.14. LocStrg – Lokale Steuerungsteile.....	21
5.15. StrgVa – Steuerungsvariable	22
5.16. ParEnd – Parallelverarbeitung	23
5.17. CALL – Fehlerhafte Bausteinaufrufkonstrukte	24
5.18. Schwebe	25
5.19. GEVO – Fehlende Endekonstrukte	26
5.20. TRAN – Transaktionskonstrukte	26
5.21. CMNT– Comments	28
5.22. CtvUse – Ctv Konstrukte innerhalb von Arbeitsgängen.....	29
5.23. WflTra – Interaktionen innerhalb einer Infrastruktur-Transaktion	29

Leerseite

1. Problemstellung

Modulentwurf und –Entwicklung sind grundsätzlich iterative Tätigkeiten, bei denen ein fertiges Ergebnis oft erst nach mehreren Bearbeitungsschritten entsteht. Während der einzelnen Bearbeitungsschritte entstehen somit Zwischenergebnisse, die bewusst formal noch nicht vollständig ausformuliert sind. Beispiele dafür sind Steuerungen, in denen ein bestimmter Bedingungsweig zwar schon grafisch ausmodelliert ist, die formale Bedingung aber noch nicht gleichzeitig ausformuliert werden kann. Oder Parameterobjekte, die in der Schnittstelle angelegt sind, aber in der Implementierung noch nicht (oder nicht mehr) verwendet werden.

Die eingesetzten Entwicklungswerkzeuge unterstützen diese iterative Arbeitsweise und lassen formal unvollständige Beschreibungen zu.

Spätestens zu den Zeitpunkten, wo Entwicklungsstände in die Produktion übergeben werden, sollten Modulbeschreibungen aber formal vollständig und richtig sein.

Formal nicht korrekte Module können zu unterschiedlichen Laufzeitproblemen führen. Dies reicht von Performance-Einbußen für die laufzeittechnische Umsetzung von nicht mehr benötigten Konstrukten, über logische Anomalien bis hin zu Abstürzen wegen nicht ausführbarer Konstruktionen.

2. Zielsetzung AnEdge

Mit dem Werkzeug AnEdge werden die im Kapitel *Problemstellung* angesprochenen formalen Anomalien in Modulen erkannt und protokolliert.

Derzeit liegt der Schwerpunkt der Analyse auf Modulen, welche mit dem Werkzeug CEEdge erstellt wurden. Diese Steuerungsmodule können insbesondere auf Anomalien in der Implementierung untersucht werden, und auf Anomalien, die sich durch das Zusammenspiel Schnittstelle und Implementierung ergeben (Schnittstelle passt nicht zur Implementierung).

Module mit anderen Implementierungen können zwar auch Gegenstand der Analyse sein, die Möglichkeiten, Anomalien aufzudecken, sind dabei aber stark eingeschränkt, da die Analyse sich in solchen Fällen auf die Schnittstelle beschränkt und nicht die Implementierung einbezieht.

Im nachfolgenden Text ist deshalb auch von Anomalien und nicht von Fehlern die Rede.

2.1. Abgrenzung zum EDB-Check

Zur Analyse von Steuerungsbausteinen gibt es derzeit schon das Werkzeug EDB-Check. Die Zielsetzung dieses Werkzeugs liegt in der Untersuchung von Modulen auf Konsistenz zwischen den redundant gehaltenen Teilen einer Modulbeschreibung und den gültigen Inhalten der Entwicklungsdatenbank.

Die Funktionen beider Werkzeuge haben derzeit unabhängig voneinander ihre Berechtigung. Mittel- bis langfristig sollen aber alle Funktionen in dem Werkzeug AnEdge zusammengefasst werden.

3. Arbeitsweise

3.2. *Übersicht*

Um die als Eingabe für die Analyse benötigte Modulbeschreibung zu finden, durchsucht AnEdge die Entwicklungsdaten anhand der für die betreffende Umgebung gültigen Einstellung der SearchOrder. In den meisten Fällen wird dadurch der Zugriff auf die bereits in einer Ressourcen-Dlx abgelegte Modulbeschreibung, **nicht** auf die im Repository (Rochade) befindliche, erfolgen. Dies ist insbesondere deshalb wichtig, weil der Aufruf des Werkzeugs aus der Rochade-Oberfläche erfolgt, was leicht zu der Annahme führen kann, dass die Analyse deshalb auch auf das hier abgelegte Dokument erfolgen würde.

Die Analyse wird gesteuert durch eine Aufgabendatei, in welcher hinterlegt ist, welche Analysen für welche Baustein auszuführen sind.

Das Ergebnis der Analyse wird abgelegt in einer Datei im xml-Format, welche dann beliebig aufbereitet werden kann.

Daneben wird noch eine log-Datei erstellt, in der Meldungen, die die Durchführung des Analyseprozess betreffen, protokolliert werden. Die nachfolgende Darstellung zeigt die Zusammenhänge.

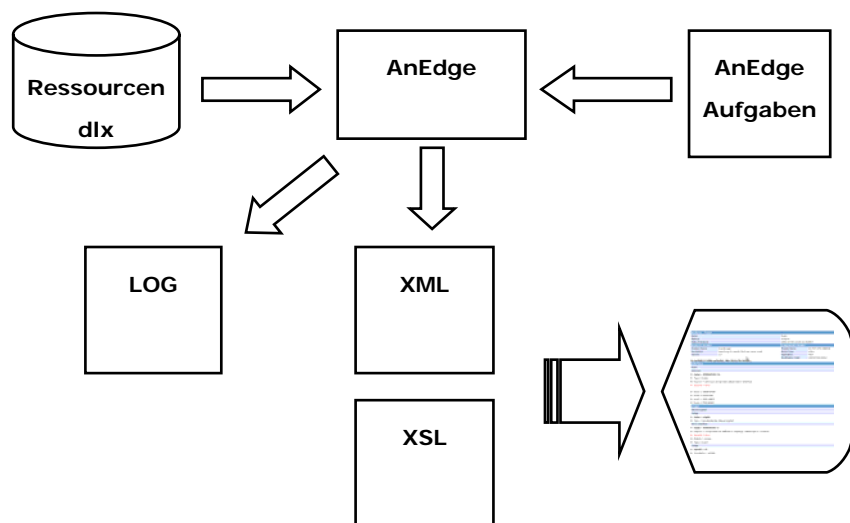


Abbildung 3-1: Arbeitsweise von AnEdge

3.3. *Aufruf von AnEdge*

Der Aufruf des Werkzeugs ist als eine Standard-Transformationsart für den Dokumenttyp DV_Modul eingerichtet und kann somit direkt aus dem Dokument heraus gestartet werden.

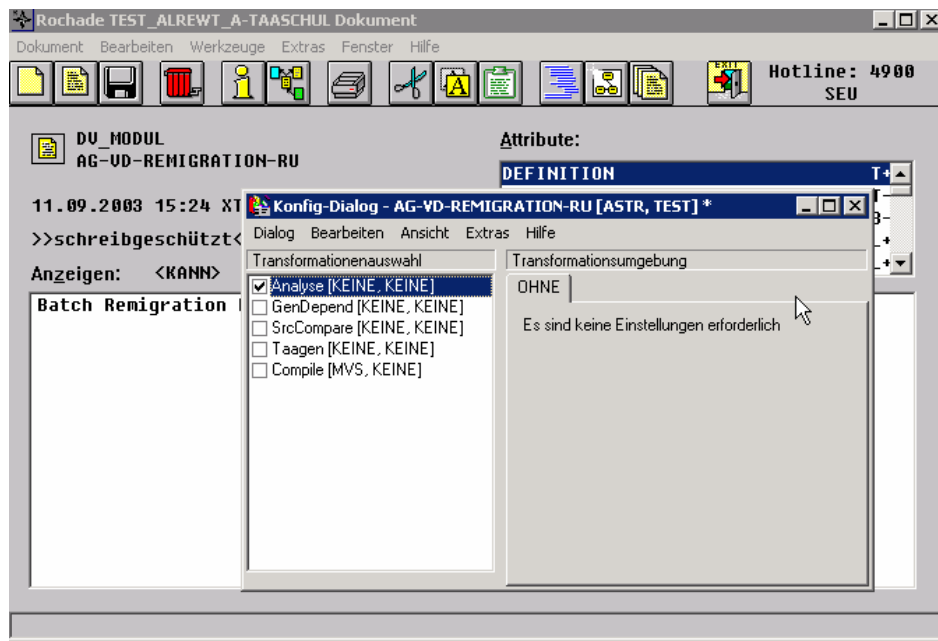


Abbildung 3-2: Aufruf des AnEdge aus Rochade

Über den Dokumentmanager können auch Analysen für mehrere Module angestoßen werden.*

3.4. Ergebnisse der Durchführung

3.4.1. Rochade-Protokoll

Wie bei jeder Transformation aus Rochade heraus, wird ein Protokoll erstellt, welches man unmittelbar nach der Durchführung der Analyse zur Ansicht angeboten bekommt. Hierbei ist darauf zu achten, dass der ausgewiesene Gesamtstatus in nahezu allen Fällen OK ist. Dies ist auch der Fall, wenn das zu analysierende Modul gar nicht auf den Ressourcen gefunden wurde. †

In diesem Fall steht in dem Ergebnisprotokoll ein Eintrag der folgenden Art:

Start: 23.09.2003 09:54:12

```
*****
KF0001I Dokument: ASTR AG-TST-ANED
KF0002I Transformation:Analyse Plattform:KEINE Variante:KEINE
00003624: taaload.c(1006): message 10x40002725: Resource 'SrvApp(TEST), Apps(TEST;TAA;ZENTKO;),
Type(MODL), Name(AG-TST-ANED)' could not be loaded from designated library. [0x04CAE070 (TAA.taaStart)]
00003624: taaimmod.c(1829): message 10x40002729: Failed to get named object 'AG-TST-ANED'.
[0x04CAE070 (TAA.taaStart)]
```

* Ein zentraler Aufruf des Werkzeug durch das Konfig-Management der Projekte (z. B. im Rahmen von Übergaben) ist zum jetzigen Zeitpunkt noch nicht vorgesehen.

† Dies ist immer dann der Fall, wenn für ein ganz neues Modul noch gar keine Ressourcen-Generierung durchgeführt wurde oder wenn eine Ressource bisher nur lokal generiert wurde und durch Pfadeinstellungen oder Kopiervorgänge diese lokalen Ressourcen für AnEdge nicht mehr sichtbar sind.

3.4.2. Analyse-Ergebnis

Die Ergebnisse der Analyse werden in dem lokalen Verzeichnis c:\tmp abgelegt. Dort gibt es für jedes Modul, welches einer Analyse unterzogen wurde, zwei Dateien. Die eigentlichen Analyse-Ergebnisse stehen in der xml-Datei.

Die Datei kann durch Doppelclick geöffnet werden. Das Ergebnis wird durch ein xsl-Stylesheet aufbereitet und im Browser angezeigt. Die nachfolgende Darstellung zeigt das typische Aussehen eines Analyse-Ergebnisses. Die Details dieser Ansicht werden weiter unten detailliert behandelt.

AnEdge Analyse	
Modulname	AG-TST-VTG-ANEDGE
Modultyp	ASTR
Letzte Änderung	20031104115725

Analyse - States
Es wurde(n) 12 Fehler gefunden. Hier klicken für Details...

Analyse - Event
Es wurde(n) 2 Fehler gefunden. Hier klicken für Details...

Analyse - Cond
Es wurde(n) 2 Fehler gefunden. Hier klicken für Details...

Abbildung 3-3: Aufbereitete Analyseergebnisse

3.4.3. Log-Datei

Neben dem Analyse-Ergebnis wird noch eine Log-Datei erzeugt, welche bei der Untersuchung von unerwarteten Analyse-Ergebnissen zur Hilfe herangezogen werden kann. Die Inhalte dieser Datei werden hier nicht im Detail dargestellt.

4. Darstellung der Analyseergebnisse

4.5. Übersicht

Das Werkzeug AnEdge unterstützt verschiedene Analysetypen. Beim Aufruf aus Rochade werden grundsätzlich alle Analysetypen durchgeführt. Für jedes analysierte Modul gibt es in dem aufbereiteten Gesamtergebnis einen Kopfteil. Daneben gibt es für jede Analyse einen eigenen Abschnitt in dem die gefundenen Anomalien dargestellt werden. Die Darstellung der Anomalien im Detail wird zunächst unterdrückt und kann durch Anklicken des Textes „Es wurde(n) n Fehler gefunden. Hier klicken für Details...“ sichtbar gemacht werden.

Die nachfolgende Abbildung zeigt diesen grundsätzlichen Aufbau eines Analyseergebnisses, hier am Beispiel des Analysetyps „States“.

AnEdge Analyse	
Modulname	AG-TST-VTG-ANEDGE
Modultyp	ASTR
Letzte Änderung	20031104115725

Analyse - States
Es wurde(n) 12 Fehler gefunden. Hier klicken für Details...

Abbildung 4-1: Grundsätzlicher Aufbau Analyseergebnisse

Das Datum „Letzte Änderung“ ist das Änderungsdatum des Moduls in Rochade zu dem Zeitpunkt der Generierung des Moduls auf die Ressourcen. Bei unerwarteten Analyseergebnissen kann hier geprüft werden, ob die in den Ressourcen gefundene und für die Analyse verwendete Version des Moduls mit der in Rochade gerade aktuellen Version übereinstimmt. Ansonsten sollten die Einträge in ihrer Bedeutung selbsterklärend sein.

Grundsätzlich werden die gefundenen Anomalien in die 3 Kategorien *Verwendung*, *WrongUsed* und *Unbenutzt* eingeordnet.

- Verwendung bedeutet, es wird etwas verwendet, was es nicht gibt.
- WrongUsed bedeutet, es wurde etwas definiert, aber nicht entsprechend seiner Bedeutung verwendet.
- Unbenutzt bedeutet, es wurde etwas definiert, das dann aber nie verwendet wird.

Nicht für jeden Analysetyp existieren alle 3 Kategorien.

Bei Anomalien, die in der Implementierung von Steuerungen gefunden werden, wird immer der untersuchte Steuerungssteil identifiziert (nur einmal für alle dort gefundenen Anomalien), und dann die Anomalie im Einzelnen mit zusätzlichen Angaben, die ein Auffinden der Stelle in der Steuerung erleichtern sollen.

Die nachfolgende Abbildung zeigt die grundsätzliche Darstellung einer Anomalie (hier am Beispiel des Analysetyps „States“.

Usage
Steuerungsteil
Cedge
• Name = empty
• Typ = Verarbeitender Steuerungsteil
Nicht in der Schnittstelle
• Name = NOTDEF
• Grund = Component not defined in interface, referenced in ModuleCall
• Schweregrad = Error
• Modul = IN-TST-VTG
• Typ = State
Cedge
• ConsID = 59
• ConsName = empty
• ConsText = Bedingung

Abbildung 4-2: Grundsätzliche Darstellung einer Anomalie

Die Einträge erklären sich wie folgt:

Die Anomaliekategorie ist *Verwendung*. Untersucht wurde ein mit CEdge erstellter *Steuerungsteil*. Dabei handelt es sich um den „*Verarbeitenden Steuerungsteil*“, welcher über keinen explizit vergebenen Namen verfügt, deshalb die Angabe *Name=empty*. Wird die Anomalie in einem explizit benannten lokalen Steuerungsteil gefunden, so enthält die Angabe *Name=* auch den Namen dieses Steuerungsteils.

Die Art der Anomalie ist „*Nicht in der Schnittstelle*“. Die Anomalie wurde erkannt für ein Konstrukt, in dem ein Zustand (State) mit dem Namen *NOTDEF* (Name=) referenziert wurde. Die Gewichtung der Anomalie ist vom Schweregrad „Error“.*

Danach finden sich Angaben zur Eingrenzung des Konstrukts, in dem die Abweichung erkannt wurde. In diesem Fall war in der CEdge Steuerung keine explizite Kommentierung zu dem Konstrukt hinterlegt worden, weshalb die Einträge *ConsName* und *ConsText* leer sind. Die *ConsId* ist die eindeutige Identifikation des Konstrukts in der CEdge Steuerung. Mit der CEdge Funktion *Bearbeiten/Gehe_Zu* kann der Benutzer sich direkt auf die Stelle im Steuerungsablauf positionieren, an der die Anomalie erkannt wurde.

* Derzeit werden nur Anomalien der Gewichtung Error dargestellt.

5. Analysedetails

5.6. States (Zustände)

5.6.1. Allgemein

Die Analyse States sucht nach Konstruktionen, in denen unbekannte Zustände in einer Steuerung verwendet werden, bzw. definierte Zustände niemals verwendet werden.

5.6.2. Kategorie Verwendung

Es werden Ende- oder Abbruchkonstrukte erkannt, in denen vergessen wurde, einen Zustand zu setzen. Ein konkretes Beispiel für ein solches Analyseergebnis sieht wie folgt aus:

```

Missing
  ● Grund = Missing information in Abbruch Construct
  ● Schweregrad = Error
  ● Was = State
  ● Wo = Abbruch Construct
  Cedge
    ● ConsID = 53
    ● ConsName = empty
    ● ConsText = Abbruch [Unnamed]
  
```

Abbildung 5-1: Beispiel fehlender Zustand in Konstrukt

Es werden Ende- oder Abbruchkonstrukte erkannt, in denen ein Zustand gesetzt wird, der in der Schnittstellendefinition des Moduls nicht enthalten ist. Ein konkretes Beispiel für ein solches Analyseergebnis sieht wie folgt aus:

```

Unbekannt
  ● Name = IRRTUM
  ● Typ = State
  ● Grund = Unknown component referenced in End Construct
  ● Schweregrad = Error
  Cedge
    ● ConsID = 99
    ● ConsName = empty
    ● ConsText = Abbruch IRRTUM
  
```

Abbildung 5-2: Beispiel unbekannter Zustand in Konstrukt

Es werden Konstrukte erkannt, in denen ein unbekannter Zustand eines *gerufenen* Moduls verwendet wird. Die ist z. B. in Bedingungsabfragen oder Schleifenbedingungen der Fall. Ein konkretes Beispiel für ein solches Analyseergebnis (hier: Abfrage im Bedingungs-zweig) sieht wie folgt aus:

```

Nicht in der Schnittstelle
  ▶ Name = NOTDEF
  ▶ Grund = Component not defined in interface, referenced in ModuleCall
  ▶ Schweregrad = Error
  ▶ Modul = IN-TST-VTG
  ▶ Typ = State
    Cedge
      • ConsID = 56
      • ConsName = SCHLEIFE
      • ConsText = Iterationsbeginn SCHLEIFE

```

Abbildung 5-3: Beispiel unbekannter Zustand in der Schnittstelle des gerufenen Moduls

5.6.3. Kategorie Unbenutzt

Es werden Zustände erkannt, welche zwar in der Schnittstelle des Moduls definiert sind, aber in der Implementierung niemals angesprochen werden. Ein konkretes Beispiel für ein solches Analyseergebnis sieht wie folgt aus:

```

Unbenutzt
  Wird nicht benutzt
  • Name = ZUSTAND24
  • Grund = Component defined but never used
  • Schweregrad = Info
  • Typ = State
  • Definiert in = AG-TST-VTG-ANEDGE

```

Abbildung 5-4: Beispiel für einen definierten aber nicht verwendeten Zustand

5.7. Events (Ereignisse)

5.7.1. Allgemein

Die Analyse Events sucht nach Konstruktionen, in denen unbekannte Ereignisse in der Steuerung verwendet werden bzw. definierte Ereignisse nicht verwendet werden.

5.7.2. Kategorie Verwendung

Es werden Ereignisse erkannt, welche in der Implementierung benutzt werden, die aber in der Schnittstelle nicht definiert sind.* Ein konkretes Beispiel für ein solches Analyseergebnis sieht wie folgt aus. Hier wurde das Ereignis OPERATION-1 in einer Bedingungsabfrage (Condition) verwendet.

* Es werden Konstrukte erkannt, in denen die Ereignisse anderer Module verwendet werden, welche in der Schnittstelle dieser anderen Module nicht definiert sind.

Not in interface

- **Name = OPERATION-1**
- Reason = Component not defined in interface, referenced in Condition
- **Severity = Error**
- Module = empty
- Type = Event

Cedge

- **ConsID = 97**
- ConsName = empty
- ConsText = Bedingung

Abbildung 5-5: Beispiel für die Verwendung nicht definierter Ereignisse

Es wird erkannt, wenn ein Modul aufgerufen wird mit einem Ereignis, welches in der Schnittstelle des Moduls gar nicht enthalten ist. Im nachfolgenden Beispiel wurde versucht, das Modul AF-TST-PRAEMIE mit dem Ereignis OPERATION-1 aufzurufen. Solche Situationen können entstehen, wenn Modulschnittstellen geändert werden, ohne dass die von dieser Änderung betroffenen Steuerungsbausteine neu generiert werden.

Modulaufruf

- **Modul = AF-TST-PRAEMIE**
- Grund = Module Call with Undefined Event
- **Schweregrad = Error**
- Typ = EFUN
- Projekt = TEST
- Ereignis = OPERATION-1

Abbildung 5-6: Beispiel für Modulaufruf mit unbekanntem Ereignis

5.7.3. Kategorie „Unbenutzt“

Es werden Ereignisse erkannt, welche zwar in der Schnittstellenbeschreibung des Moduls enthalten sind, aber in der Implementierung niemals angesprochen werden. Ein konkretes Beispiel für ein solches Analyseergebnis sieht wie folgt aus. Hier wurde das Ereignis FREIGEBEN in der Schnittstelle gefunden, aber nicht benutzt.

Wird nicht benutzt

- **Name = OPERATION-24**
- Grund = Component defined but never used
- **Schweregrad = Info**
- Typ = Event
- Definiert in = AG-TST-VTG-ANEDGE

Abbildung 5-7: Beispiel für nicht benutzte Ereignisse

5.8. ParUse - Verwendung Parameter Objekte

5.8.1. Allgemein

Es werden die Parameterobjekte ausgewiesen, welche zwar definiert sind, aber niemals in der Implementierung angesprochen werden. *Die Auswertung erfolgt auch ereignis-spezifisch, d. h. ein Parameterobjekt, welches nur für bestimmte Ereignisse definiert ist, und in einem dieser Ereignisse nicht angesprochen wird, wird auch ausgewiesen.[†]

5.8.2. Kategorie Unbenutzt

Die nachfolgende Abbildung zeigt die Analyseergebnisse für ein Parameterobjekt, welches in der Schnittstelle definiert ist, aber nicht verwendet wird. Hier sind für das Ereignis FREIGEBEN die Parameterobjekte Z0AAZZV und STPME zugeordnet, werden aber in diesem Ereignis gar nicht benutzt.

```

Nicht für dieses Ereignis benutzt
Ereignis
  • Name = FREIGEBEN
    Wird nicht benutzt
      • Name = Z0AAZZV
      • Grund = Component defined but never used
      • Schweregrad = Info
      • Typ = Parameter Object
      • Definiert in = AG-TST-VTG-ANEDGE
    Wird nicht benutzt
      • Name = STPME
      • Grund = Component defined but never used
      • Schweregrad = Info
      • Typ = Parameter Object
      • Definiert in = AG-TST-VTG-ANEDGE
  
```

Abbildung 5-8: Definiertes, aber nicht verwendetes Parameterobjekt

5.8.3. Kategorie „Unterschiedliche Nutzung“

Hier werden Parameterobjekte ausgewiesen, welche in der Schnittstelle des Moduls die Rolle REF haben und in der Implementierung dieses Datenobjekt bei einem Modulaufruf einem Datenobjekt mit der Rolle MOD zuweisen. Dabei handelt es sich **nicht** um einen Fehler (Schweregrad=Info).

* Ab Taa-Release 7.04 werden auch die Schnittstellen von Start- und CTV-Konstrukten mit in die Analyse einbezogen.

† Erst ab TAA-Release 7.04

Unterschiedliche Nutzung

Ereignis

- **Name = GENERIEREN**

Unterschiedliche Nutzung

- Name = PVGESL
- Grund = Different use in Context
- **Schweregrad = Info**
- Typ = REF
- Ereignis = GENERIEREN
- Definiert in = 0x00000001

5.9. LocUse - Verwendung lokale Objekte

5.9.1. Allgemein

Es werden die lokalen Datenobjekte ausgewiesen, welche zwar definiert sind, aber niemals in der Implementierung angesprochen werden. *

5.9.2. Kategorie „Unbenutzt“

Wird nicht benutzt

- **Name = LOBPO**
- Grund = Component defined but never used
- **Schweregrad = Info**
- Typ = Local Object
- Definiert in = AG-TST-VTG-ANEDGE

Abbildung 5-9: Definiertes, aber nicht benutztes lokales Datenobjekt

5.10. GloUse - Verwendung globale Objekte

5.10.1. Allgemein

Diese Analyse prüft die Verwendung der globalen Objekte. Ein globales Objekt gilt nicht dadurch als verwendet, wenn es in der Schnittstelle eines in der Steuerung gerufenen Bausteins ebenfalls definiert ist. Eine Verwendung wäre dadurch gegeben, dass das globale Objekt einem Parameterobjekt bei der Übergabe zugewiesen wird.

5.10.2. Kategorie Unbenutzt

Es werden, pro Ereignis, die globalen Objekte aufgelistet, die zwar in der Schnittstelle definiert sind, aber nicht verwendet werden. Die nachfolgende Abbildung zeigt ein Beispiel für das Analyseergebnis bei einem definierten aber nicht verwendeten globalen Datenobjekt.

Das Beispiel in der nachfolgenden Abbildung zeigt ein Globales Objekt ROOTIDS, welches in keinem Ereignis verwendet wird. Danach werden für jedes Ereignis die Globalen Objekte dargestellt, welche zwar zugeordnet sind aber nicht verwendet werden.

* Ab TAA-Release 7.04 werden auch die Schnittstellen von Start- und CTV-Konstrukten mit in die Analyse einbezogen.

Unbenutzt

Wird nicht benutzt

- Name = **ROOTIDS**
- Grund = Component defined but never used
- Schweregrad = Info
- Typ = Global Object
- Definiert in = AG-TST-VTG-ANEDGE

Nicht für dieses Ereignis benutzt

Ereignis

- Name = **FREIGEBEN**

Wird nicht benutzt

- Name = **SYSTVAR**
- Grund = Component defined but never used
- Schweregrad = Info
- Typ = Global Object
- Definiert in = AG-TST-VTG-ANEDGE

Abbildung 5-10: Definiertes, aber nicht verwendet globales Datenobjekt

5.11. ObjUse - Verwendung Objekte Allgemein

Diese Analyse bietet eine Zusammenfassung der Analysen ParUse, LocUse und GloUse sowie eine weitere, ereignisspezifische Untersuchung, welche Parameter oder Globalen Objekte in den Pfaden von Ereignissen verwendet werden, für die sie in der Schnittstelle gar nicht definiert sind.

5.11.1. Kategorie Unbenutzt

In dieser Analyse werden dieselben Ergebnisse erzeugt wie bei den Analysen ParUse, GloUse und LocUse, allerdings für alle Typen von Datenobjekten. Die nachfolgende Abbildung zeigt die Analyseergebnisse für ein Parameterobjekt und ein lokales Datenobjekt.

- Name = **PVGESL2**
- Reason = Component defined but never used
- Severity = Info
- Type = Object
- Defined in = AG-TST-VTG-ANEDGE

Not used

- Name = **LOBPO**
- Reason = Component defined but never used
- Severity = Info
- Type = Object
- Defined in = AG-TST-VTG-ANEDGE

Abbildung 5-11: Definierte aber nicht verwendete Datenobjekte

5.11.2. Kategorie WrongUsed

In der Schnittstelle können Parameterobjekte ereignisspezifisch zugeordnet werden. Dadurch kann es passieren, dass Datenobjekte in Ausführungspfaden angesprochen werden, aber für diesen Pfad das Parameterobjekt gar nicht übergeben wurde.

Ein Beispiel:

Das Parameterobjekt STPME ist in der Schnittstelle eines Moduls allen Ereignissen zugeordnet, bis auf das Ereignis GENERIEREN. Im Ausführungspfade dieses Ereignis wird jedoch ein Modul aufgerufen, welchem STPME als Datenobjekt übergeben wird. Da das Parameterobjekt zur Modellierungszeit bekannt ist, funktioniert diese Zuordnung auch (genauso funktionieren die Transformationen STRGEN und COMPILE MVS einwandfrei). Die nachfolgende Abbildung zeigt den Ausführungspfad für das Ereignis sowie den Aufruf des Moduls mit der Zuordnung des Parameterobjekts STPME.

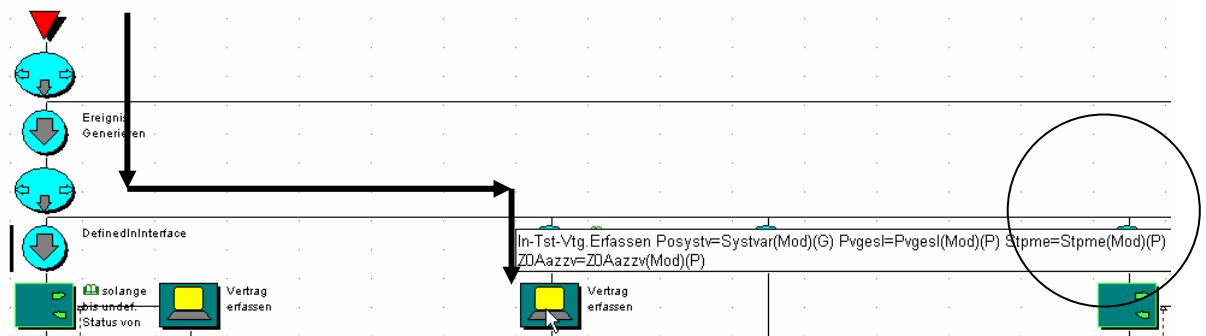


Abbildung 5-12: Ausführungspfad für ein Ereignis

Diese Situation wird von AnEdge erkannt und protokolliert so wie in der nachfolgenden Abbildung dargestellt.

Ereignis
<ul style="list-style-type: none"> Name = GENERIEREN
Not in interface
<ul style="list-style-type: none"> Name = STPME Reason = Component not defined in interface, referenced in GENERIEREN Severity = Error Module = AG-TST-VTG-ANEDGE Type = Object

Abbildung 5-13: Verwendete, aber nicht definierte Datenobjekte, pro Ereignis

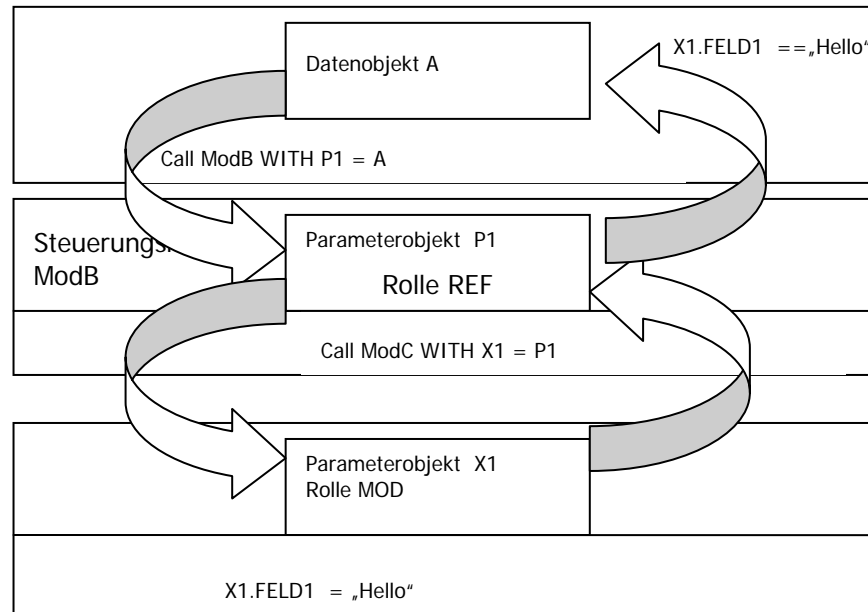
5.12. ObjMod – Modifikation von Objekten

5.12.1. Allgemein

Der Hintergrund dieser Analyse ist die Bedeutung der Parameterobjektrolle REF.

Diese Rolle zeigt lediglich an, dass dieses Modul **selbst** keine Änderungen an Dateninhalten vornehmen wird. Das bedeutet aber nicht automatisch, dass deshalb die Inhalte des Datenobjekts am Ende der Ausführung noch tatsächlich dieselben sein werden wie beim Aufruf des Moduls. Denn das Modul, welches das Datenobjekt in der Rolle REF in seiner Schnittstelle hat, kann durchaus andere Module aufrufen und das REF-Datenobjekt auf ein MOD-Datenobjekt in der Schnittstelle des gerufenen Moduls zuweisen. Diese können dann die Inhalte des Datenobjekts ändern und diese Änderungen würden dann auch durchgereicht an den Rufer. Dieser würde die geänderten Inhalte dann wiederum an seinen Rufer zurückgeben und damit hätte dieser geänderte Inhalte, trotz Rolle REF.

Die nachfolgende Abbildung zeigt diesen Zusammenhang. Obwohl das Steuerungsmodul ModB das Datenobjekt P1 mit der Rolle REF in der Schnittstelle hat, wird es die durch das Modul ModC darin geänderten Inhalte an seinen Rufer zurückgeben.



Die Rolle REF sollte bei Auswertungen des Repositories dafür verwendet werden können, herauszufinden, welche Module tatsächlich keine inhaltlichen Änderungen an Datenobjekten vornehmen wollen.

Verstanden worden ist die Rolle REF jedoch zum Teil als Zusicherung, dass bei Ende dieses Moduls der Inhalt des Parameterobjekts mit dieser Rolle auf jeden Fall unverändert zurückgegeben würde, unabhängig davon, welche Rollen dieses Datenobjekt irgendwo in der unter dem Modul liegenden Aufrufhierarchie hat.

Seit TAA-Release 7.04 gibt es beim Aufruf eines Bausteins die Möglichkeit, für jedes Datenobjekt anzugeben, ob lediglich die Inhalte des Datenobjekts übergeben werden (By Value) oder eine Referenz auf das Datenobjekt (BY REFERENCE)*.

Nur bei dem zuerst beschriebenen Verständnis der Rolle REF macht die Analyse einen Sinn, denn in diesem Fall müssten eigentlich alle Parameterobjekte in der Schnittstelle eines Steuerungsbausteins die Rolle REF haben, da Steuerungen die Inhalte von Parameterobjekten per Definition nicht verändern können.

Derzeit ist das Analyseergebnis deshalb ein Hinweis darauf, wo überall die Rolle MOD eigentlich gegen die Rolle REF ausgetauscht werden könnte.

5.12.2. Kategorie „WrongUsed“

Die nachfolgende Abbildung zeigt das Analyseergebnis für ein ereignisspezifisches Parameterobjekt in der Rolle MOD, welches tatsächlich nicht durch die analysierte Steuerung modifiziert wird.

* BY REFERENCE wird der Default sein, damit die bestehenden Aufrufe nicht zwingend umgestellt werden müssen.

```

Ereignis
  • Name = FREIGEBEN
    Falsch benutzt
      • Name = ZOAAZZV
      • Grund = Wrong use in Context
      • Schweregrad = Warning
      • Typ = MOD
      • Ereignis = FREIGEBEN
      • Definiert in = 0x00000000
  
```

5.13. Cond – Bedingungen

5.13.1. Allgemein

Bedingungen werden im Zusammenhang mit Schleifenkonstrukten und Bedingungskonstrukten verwendet.

5.13.2. Kategorie Verwendung

Fehlende Bedingungen in Schleifenkonstrukten werden als Anomalie der Gewichtung Warning ausgewiesen, da es ja durchaus sein kann, dass tatsächlich keine ausformulierte Bedingung für die Schleife notwendig ist, z. B. weil es sich um eine Einmalschleife handelt (Once) oder weil es sich um eine Endlosschleife (Forever) handelt, bei der alle Aussprungsbedingungen im Schleifenkörper explizit ausmodelliert sind. Die nachfolgende Abbildung zeigt ein Beispiel für das Analyseergebnis bei einer fehlenden Schleifenbedingung.

```

Steuerungsteil
  Cedge
    • Name = empty
    • Typ = Verarbeitender Steuerungsteil
  Missing
    • Grund = Missing information in Expression
    • Schweregrad = Warning
    • Was = Loop once
    • Wo = Expression
      Cedge
        • ConsID = 72
        • ConsName = SCHLEIFE
        • ConsText = Iterationsbeginn SCHLEIFE
  
```

Abbildung 5-14: Fehlende Bedingung in einem Schleifenkonstrukt

Fehlende Bedingungen in Bedingungskonstrukten werden als Anomalie der Gewichtung Error ausgewiesen, da jeder Bedingungsweig (abgesehen vom Else-Zweig) über eine ausformulierte Bedingung verfügen muss. Die nachfolgende Abbildung zeigt ein Beispiel für das Analyseergebnis bei einer fehlenden Bedingung in einem Bedingungskonstrukt.

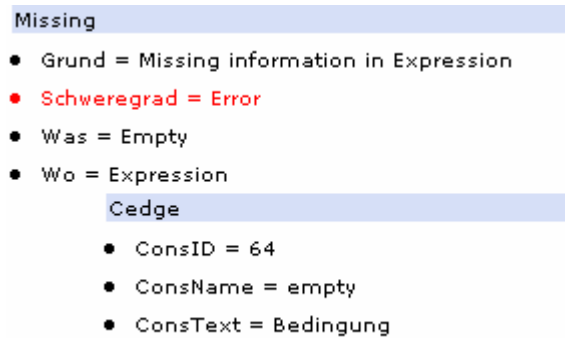


Abbildung 5-15: Fehlende Bedingung in einem Bedingungskonstrukt

5.14. LocStrg – Lokale Steuerungsteile

5.14.1. Allgemein

Es gibt unterschiedliche Typen von lokalen Steuerungsteilen. Die für ein Modul definierten Steuerungsteile können dem Übersichtsdialog von CEdge entnommen werden.

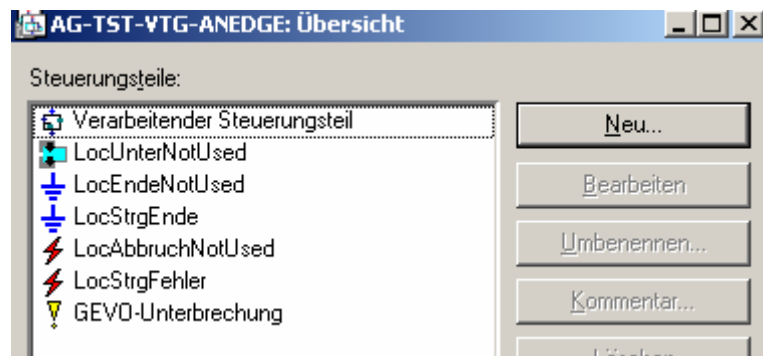


Abbildung 5-16: Steuerungsteile

Die unterschiedlichen Steuerungsteiltypen sind in dieser Darstellung durch unterschiedliche Symbole gekennzeichnet.

Die Analyse erkennt definierte, aber nicht verwendete Steuerungsteile.

5.14.2. Kategorie Unbenutzt

Die nachfolgende Abbildung zeigt die Analyseergebnisse für definierte, aber nicht verwendete Steuerungsteile.

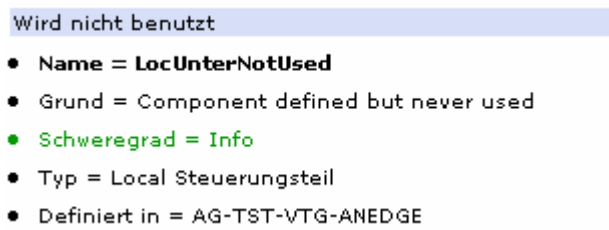


Abbildung 5-17: Definierter aber nicht verwendeter Steuerungsteil

5.15. StrgVa – Steuerungsvariable

5.15.1. Allgemein

Steuerungsvariable können in Steuerungen beim Durchlauf eines Konstruktes gesetzt werden. In Bedingungskonstrukten kann eine solche Steuerungsvariable abgefragt werden. Die nachfolgende Abbildung zeigt das Setzen eines Zustands über das Kontextmenü eines Konstrukts.

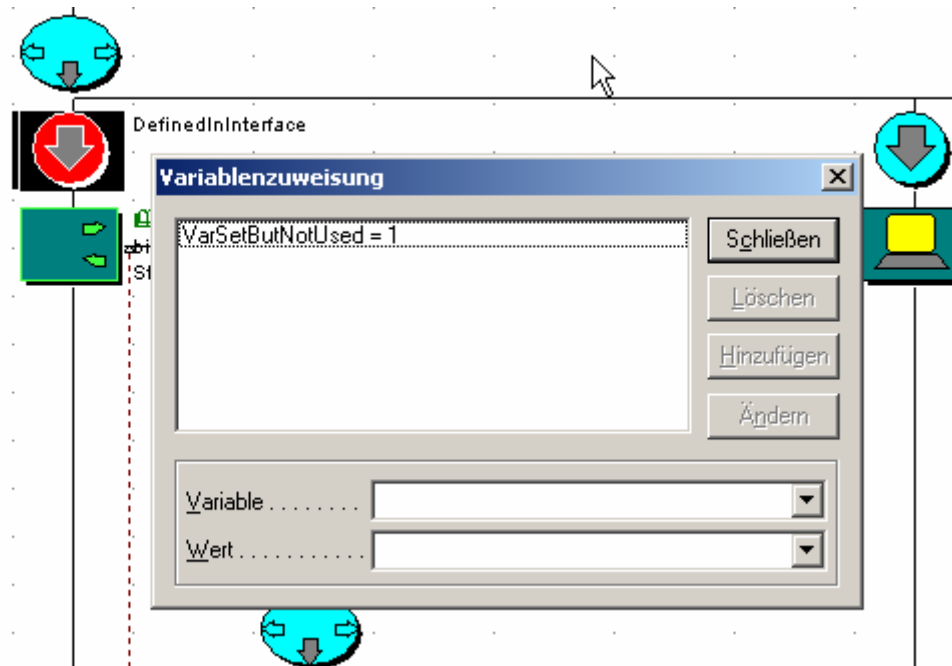


Abbildung 5-18: Setzen einer Steuerungsvariablen

5.15.2. Kategorie „Falsch benutzt“

Das Setzen und Abfragen von Steuerungsvariablen sollte innerhalb einer Steuerung immer paarweise auftreten, also wenn eine Steuerungsvariable gesetzt wird, sollte sie auch in einer Bedingung verwendet werden, und wenn eine Steuerungsvariable in einer Bedingung abgefragt wird, so muss sie auch in der Steuerung gesetzt werden. AnEdge prüft Steuerung genau auf diese Konsistenz hin.

Falsch benutzt

Nicht gesetzt

- **Name = VarUsedButNotSet**
- Grund = Component used but never set
- **Schweregrad = Error**
- Typ = Value : "1" used but not set.
- Definiert in = AG-TST-VTG-ANEDGE

Wird nicht benutzt

- **Name = VarSetButNotUsed**
- Grund = Component defined but never used
- **Schweregrad = Info**
- Typ = Value : "1" set but not used.
- Definiert in = AG-TST-VTG-ANEDGE

5.16. ParEnd – Parallelverarbeitung

5.16.1. Allgemein

Diese Analyse untersucht, ob es in parallel ablaufenden Verarbeitungszweigen Ende- oder Abbruchkonstrukte gibt, welche ja dazu führen würden, dass der eine Zweig die Verarbeitung beendet, während der andere Zweig noch abgearbeitet wird.

Zum besseren Verständnis wird nachfolgend ein Beispiel für eine Abweichung innerhalb eines Parallelkonstrukts dargestellt.

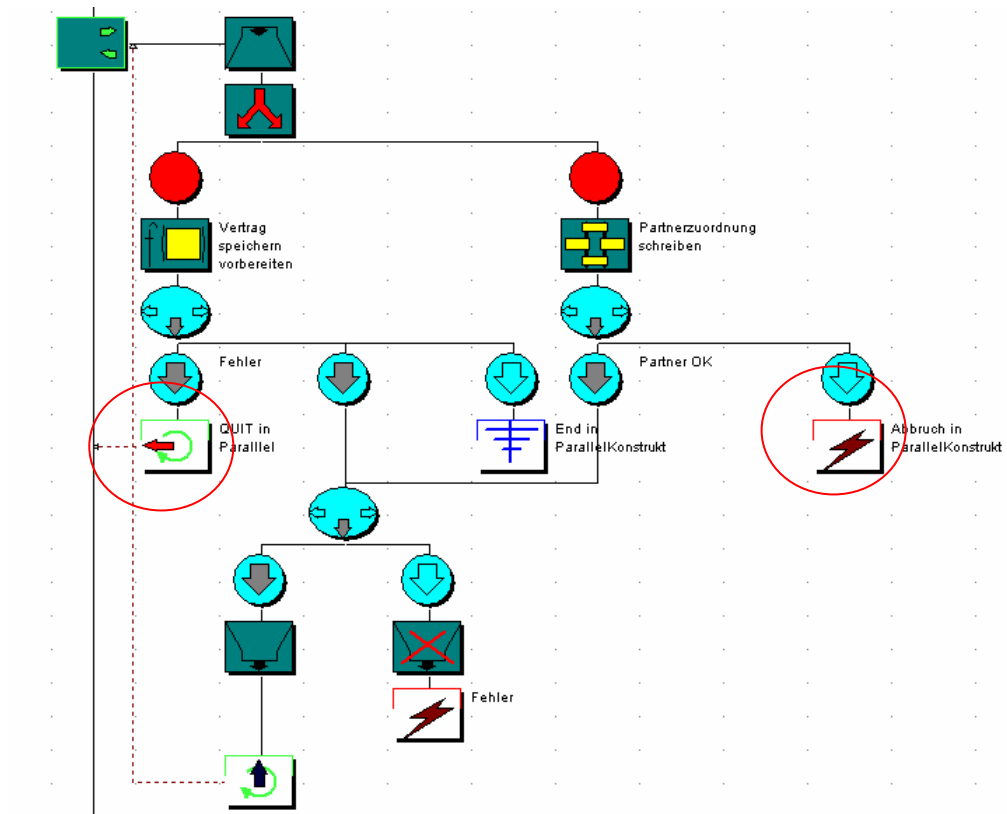


Abbildung 5-19: Parallelkonstrukt mit Fehlern

5.16.2. Kategorie Verwendung

Es wird erkannt wenn der Pfad einer Parallelverarbeitung mittels Quit oder Skip verlassen wird. Die ist immer dann der Fall, wenn Schleifenanfang außerhalb des Parallelverarbeitungspfades liegt. Die nachfolgende Abbildung zeigt die Protokollierung eines Quit-Konstrukts in einem solchen Pfad.

Iteration Quit	
• Grund =	Quit Construct exiting Thread
• Schweregrad =	Warning
Cedge	
• ConsID =	80
• ConsName =	empty
• ConsText =	Iterationsabbruch [Unnamed]

Abbildung 5-20: Protokollierung eines Quit-Konstrukts

Weiterhin wird erkannt, wenn eine Parallelverarbeitung durch ein Ende- oder Abbruchkonstrukt in mindestens einem der Verarbeitungspfade erfolgt. Die nachfolgende Abbildung zeigt die Protokollierung einer solchen Situation. Es gibt derzeit keine unterschiedliche Protokollierung von Ende- oder Abbruchkonstrukten.

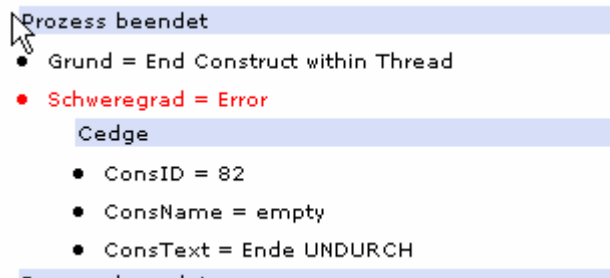


Abbildung 5-21: Protokollierung eines Ende- oder Abbruchkonstrukts

5.17. CALL – Fehlerhafte Bausteinaufrufkonstrukte

5.17.1. Kategorie Verwendung

In dieser Analyse werden fehlerhafte Bausteinaufrufkonstrukte erkannt und ausgewiesen. Das nachfolgende Beispiel zeigt das Analyseergebnis für eine fehlende Bausteinangabe in einem Modulaufrufkonstrukt.

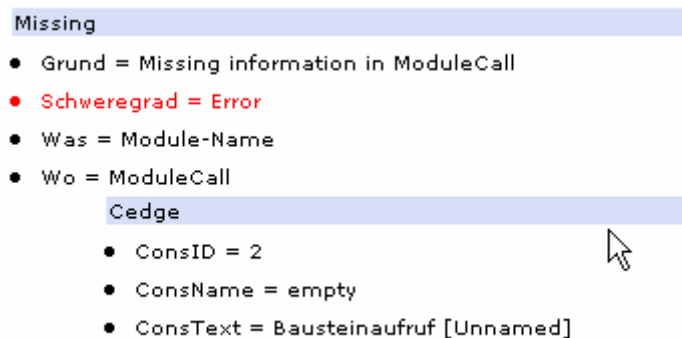


Abbildung 5-22: Protokollierung eines fehlenden Bausteinnamens im Aufrufkonstrukt

Weiterhin werden unvollständige Bausteinaufrufe protokolliert. Die nachfolgende Abbildung zeigt den Aufruf eines Bausteins mit dem Namen AF-TST-PRAEMIE, bei dessen Aufruf der Parameter ZOAAZZV keine Zuweisung erhalten hat.

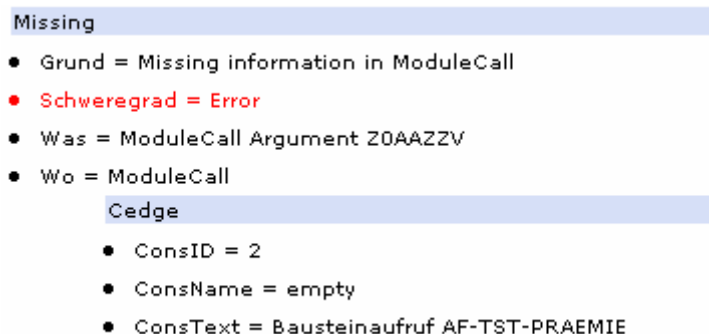


Abbildung 5-23: Protokollierung einer unvollständigen Parameter-Bestückung beim Bausteinaufruf

5.18. Schwebel

5.18.1. Allgemein

Untersucht werden die folgenden Anomalien von Klammerkonstrukten:

- öffnende Klammer ohne Namen
- öffnende Klammer ohne zugeordnete Datenobjekte,
- schließende oder verwerfende Klammer ohne Namen
- öffnende Klammer ohne schließende Klammer

5.18.2. Kategorie Verwendung

Das nachfolgende Beispiel zeigt das Analyseergebnis für eine Schwebeklammer, welcher keine Datenobjekte zugewiesen wurde.

```
Schwebe-Begin
• Grund = No Variables Set
• Schweregrad = Error
  Gefunden bei Konstrukt...
    • ConsID = 15
    • ConsName = empty
    • ConsText = Schwebe-Anfang GO-2
```

Abbildung 5-24: Beispiel Schwebeklammer ohne Datenobjekte

Das nachfolgende Beispiel zeigt das Analyseergebnis für eine schließende Schwebeklammer, welcher keine Name zugewiesen wurde. Das gleiche Ergebnis gibt es auch für eine öffnende Schwebeklammer, welcher kein Name zugewiesen wurde.

```
Schwebe-Close
• Grund = No Name Set
• Schweregrad = Error
  Gefunden bei Konstrukt...
    • ConsID = 17
    • ConsName = empty
    • ConsText = Schwebe-Ende [Unnamed]
```

Abbildung 5-25: Beispiel Schwebeklammer ohne Bezeichnung

5.18.3. Kategorie „Falsch benutzt“

Das nachfolgende Beispiel zeigt das Analyseergebnis für eine Schwebeklammer, welche zwar geöffnet wurde, für die es aber kein schließendes Konstrukt gibt. Ein Konstrukt zum Verwerfen (Rollback) dieser Schwebeklammer wurde gefunden.

```
Schwebe - GO-2
• Schweregrad = Error
• Grund = Schwebe "GO-2" opened, but not explicitly closed. Rollback found
```

Abbildung 5-26: Beispiel Schwebeklammern ohne Abschluss, aber mit Verwerfen

Dieses Beispiel zeigt eine geöffnete Schwebeklammern, welche überhaupt kein Gegenstück hat, weder Schließen noch Verwerfen.



Abbildung 5-27: Schwebeklammern offen, kein Schließen oder Verwerfen-Konstrukt vorhanden

5.19. GEVO – Fehlende Endkonstrukte

5.19.1. Kategorie „Verwendung“

Jede Gevo-Steuerung muss mit einem Gevo-Ende-Konstrukt abgeschlossen werden. Diese Analyse erkennt fehlende Gevo-Ende-Konstrukte. Das nachfolgende Beispiel zeigt die Protokollierung dieser Situation.

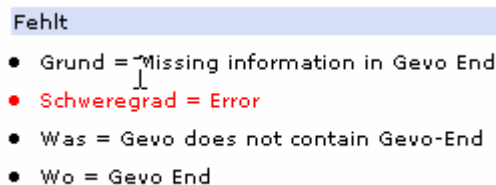


Abbildung 5-28: Protokollierung fehlendes Gevo-Ende-Konstrukt

5.20. TRAN – Transaktionskonstrukte

5.20.1. Allgemein

Es wird die Paarigkeit von Transaktionsklammern geprüft. Zu einer öffnenden Transaktionsklammern muss es eine schliessende Klammern geben.

Ausserdem sollten keine verarbeitenden Konstrukte mehr nach einer schliessenden Transaktion verwendet werden.

5.20.2. Kategorie „Verwendung“

Nach einer schliessenden Transaktionsklammern sollten keine Calls auf weitere Module mehr gemacht werden, da hierdurch unnötige Abbruchmöglichkeiten nach dem Bestätigen der Transaktionen geschehen können. In einem solchen Falle ist der Arbeitsgang wahrscheinlich nicht wiederaufsetzbar, weil die Transaktion kein zweites Mal durchlaufen werden kann.

Anedge erkennt deshalb, wenn nach einer Transaktionsklammern noch Bausteinaufrufe stattfinden und protokolliert jeden davon als „Error“.

```

Transaktion
• Grund = Transactions used, but call outside transaction
• Schweregrad = Error
  Gefunden bei Konstrukt...
  • ConsID = 57
  • ConsName = empty
  • ConsText = Bausteinaufruf AF-TST-PLAUSI
Transaktion
  
```

Abbildung 5-29: Bausteinaufruf nach schliessender Transaktionsklammer

Ein Sonderfall des Problems „Bausteinaufruf nach Transaktionsende“ ist die Verwendung des Transaktionskonstrukts innerhalb einer Schleife. Hier kann die statische Analyse nicht mit Sicherheit feststellen, ob nach dem Schliessen der Transaktionsklammer noch weitere Bausteinaufrufe dadurch folgen, dass die Schleife erneut durchlaufen wird. Solche Konstruktionen werden aus diesem Grund mit einem „Warning“ protokolliert.

```

Transaktion
• Grund = Transaction Construct within Iteration
• Schweregrad = Warning
  Gefunden bei Konstrukt...
  • ConsID = 54
  • ConsName = empty
  • ConsText = Transaktionsanfang
  
```

Abbildung 5-30: Transaktion innerhalb einer Schleifenkonstruktion

Ein weiterer Sonderfall des Problems „Bausteinaufruf nach Transaktionsende“ ist die Verwendung des Transaktionskonstrukts innerhalb einer Parallelverarbeitung. Hier kann die statische Analyse nicht mit Sicherheit feststellen, ob nach dem Schliessen der Transaktionsklammer noch weitere Bausteinaufrufe stattfinden. Solche Konstruktionen werden aus diesem Grund mit einem „Error“ protokolliert.

```

Transaktion
• Grund = Transaction Construct within Fork
• Schweregrad = Error
  Gefunden bei Konstrukt...
  • ConsID = 58
  • ConsName = empty
  • ConsText = Transaktionsanfang
Transaktion
  
```

Abbildung 5-31: Transaktion innerhalb einer Parallelverarbeitung

Weiterhin wird geprüft, ob es zu jeder öffnenden Transaktionsklammer auch eine schliessende Transaktionsklammer gibt, und umgekehrt. Wenn dies nicht zutrifft wird ein Analyseergebnis mit der Gewichtung „Error“.

Transaktion

- Grund = Open - Close mismatch. Not all transactions closed at end
- **Schweregrad = Error**

Gefunden bei Konstrukt...

- ConsID = 51
- ConsName = empty
- ConsText = Ende-Verarbeitung LocStrgEnde

Abbildung 5-32: Mismatch bei Transaktionsklammern (Schliessende Klammer fehlt)

ConsText = Transaktionsabbruch

Transaktion

- Grund = Open - Close mismatch. No Transactions left to Close
- **Schweregrad = Error**

Gefunden bei Konstrukt...

- ConsID = 60
- ConsName = empty
- ConsText = Transaktionsende

Abbildung 5-33: Mismatch bei Transaktionsklammern (Öffnende Klammer fehlt)

5.21. CMNT- Comments

5.21.1. Allgemein

Es wird geprüft, ob es zu einem Konstrukt einen Kommentar gibt.

5.21.2. Kategorie „Missing“

Anedge erkennt die Konstrukte bei denen kein Kommentar angegeben wurde.

Steuerungsteil

Cedge

- **Name = empty**
- Typ = Verarbeitender Steuerungsteil

Fehlt

- Grund = Missing information in Construct
- **Schweregrad = Info**
- Was = Comment
- Wo = Construct

Gefunden bei Konstrukt...

- ConsID = 51
- ConsName = empty
- ConsText = Ende-Verarbeitung LocStrgEnde
- ConsType = 1029

Abbildung 34: Fehlender Kommentar

5.22. CtvUse – Ctv Konstrukte innerhalb von Arbeitsgängen

5.22.1. Allgemein

Diese Analyse weist Ctv-Konstrukte aus, welche auf der Arbeitsgangebene (also nicht-GeVo) verwendet werden und die potenziell oder tatsächlich dazu führen, daß Dokumente gedruckt werden.

Dies kann problematisch sein, wenn nach dem erfolgten Druck, der Arbeitsgang unterbrochen wird. Die Wiederaufnahme des Arbeitsgangs könnte dazu führen, daß das Ctv-Konstrukt erneut durchlaufen wird und damit das Dokument zum zweiten Mal erstellt wird.

5.22.2. Kategorie „Falsch benutzt“

Anedge erkennt die Ctv-Konstrukte bei denen potenziell oder tatsächlich gedruckt wird.

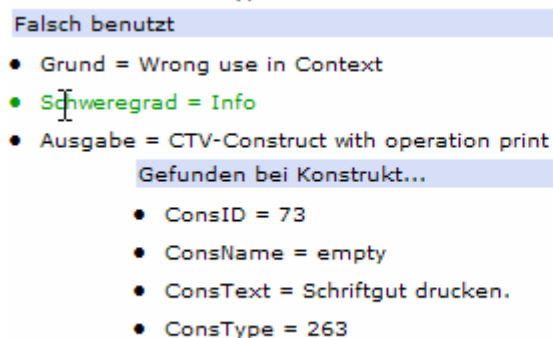


Abbildung 35: Ctv-Konstrukt mit Drucken im Arbeitsgang

5.23. WfiTra – Interaktionen innerhalb einer Infrastruktur-Transaktion

5.23.1. Allgemein

Für einen Arbeitsgang kann durch die Konfig-Einstellung „WorkflowTransaction“ gesteuert werden, ob die fachlichen Datenbankänderungen und die Änderungen an den Infrastruktur-Datenbanken (Kontext, GeVes etc.) gemeinsam durchgeführt werden. Dadurch können Diskrepanzen zwischen fachlichen und technischen GeVodaten vermieden werden, wenn es z.B. zu einem Absturz unmittelbar zwischen dem Schreiben der fachlichen und technischen Daten kommt.

Die für die gemeinsame Transaktionierung verwendete Technik verbietet es, dass innerhalb eines gemeinsam transaktionierenden Arbeitsgangs solche Konstrukte verwendet werden, die potenziell eine Benutzer-Interaktion darstellen.

Diese Anedge-Analyse sucht nach solchen Konstrukten in Arbeitsgangsteuerung.

5.23.2. Kategorie „Falsch benutzt“

Anedge erkennt die Ctv-Konstrukte bei denen potenziell oder tatsächlich gedruckt wird. In der nachfolgenden Abbildung ist das ein Baustein vom Typ elementare Interaktion und ein CTV-Baustein, der auch den Ctv-Dialog ermöglicht.

Ereignis - ERFASSEN

Falsch benutzt

- Grund = Wrong use in Context
- Schweregrad = Info
- Ausgabe = Interaction used in control which has UseWorkflowTransaction set

Gefunden bei Konstrukt...

- ConsID = 5
- ConsName = empty
- ConsText = Bausteinaufruf IN-TST-VTG
- ConsType = 262

Falsch benutzt

- Grund = Wrong use in Context
- Schweregrad = Info
- Ausgabe = CTV-Dialog used in control which has UseWorkflowTransaction set

Gefunden bei Konstrukt...

- ConsID = 74
- ConsName = empty
- ConsText = Schriftgut im CTV Dialog bearbeiten.
- ConsType = 263

Abbildung 36: Fehlerhafte Konstrukte in Workflow Transaktion