FernUniversität in Hagen

Graphische Entwicklungsumgebung für eine Echtzeitprogrammiersprache

Masterarbeit

Vorgetragen von: Andreas Enns
Studiengang: MSc Elektro- und Informationstechnik

Erstprüfer: Prof. Dr.-Ing. habil. Herwig Unger
Zweitprüfer: Dipl.-Inform. (Univ.) Marcel Schaible

Inhalt

- Einleitung

  Ziel, Motivation und Aufgaben

- Grammatiktransformation

  Überführung einer Grammatik in eine andere äquivalente Grammatik

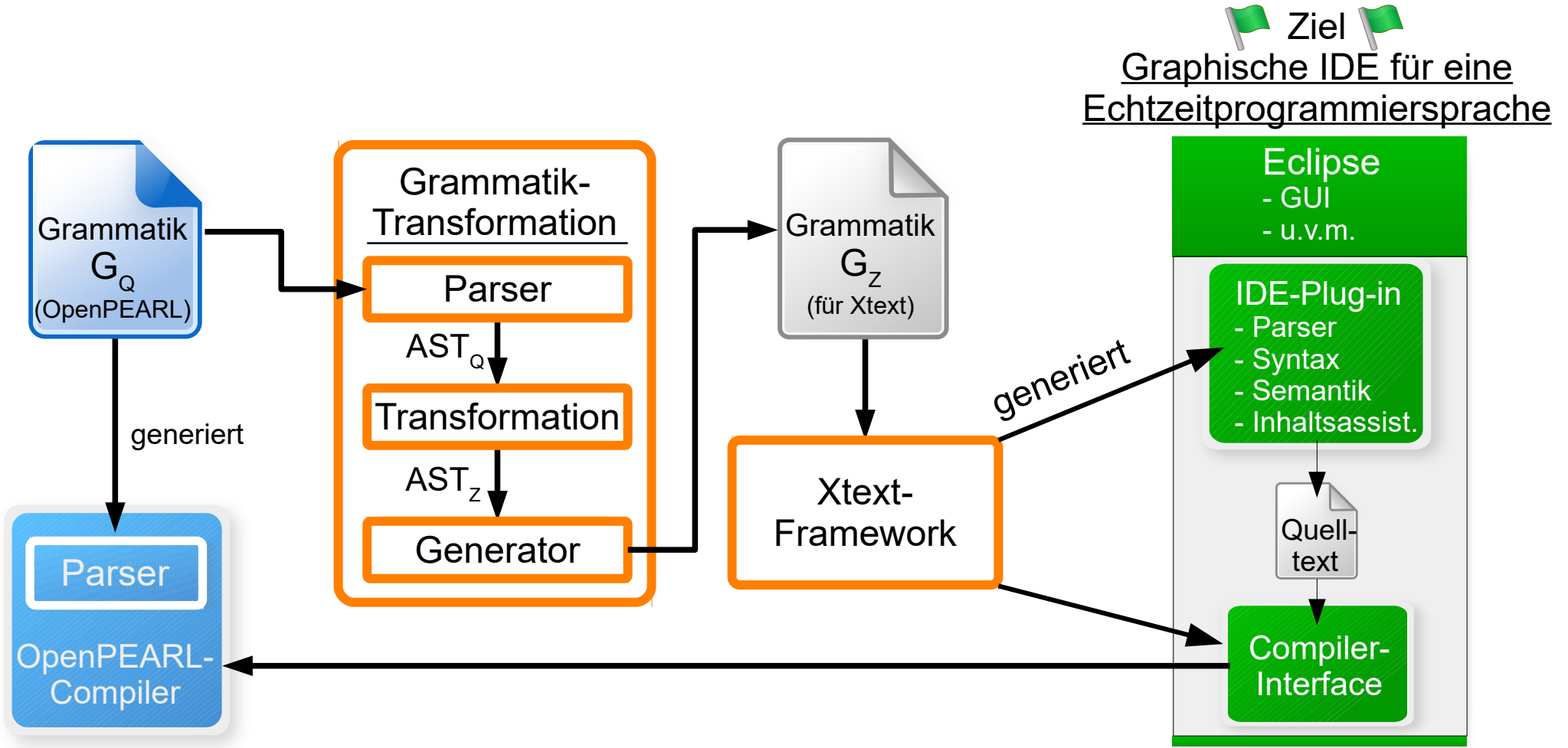
- Graphische Entwicklungsumgebung

  Integration der Sprache, Integration des Compilers und semantische Validierung

- Schlussfolgerungen

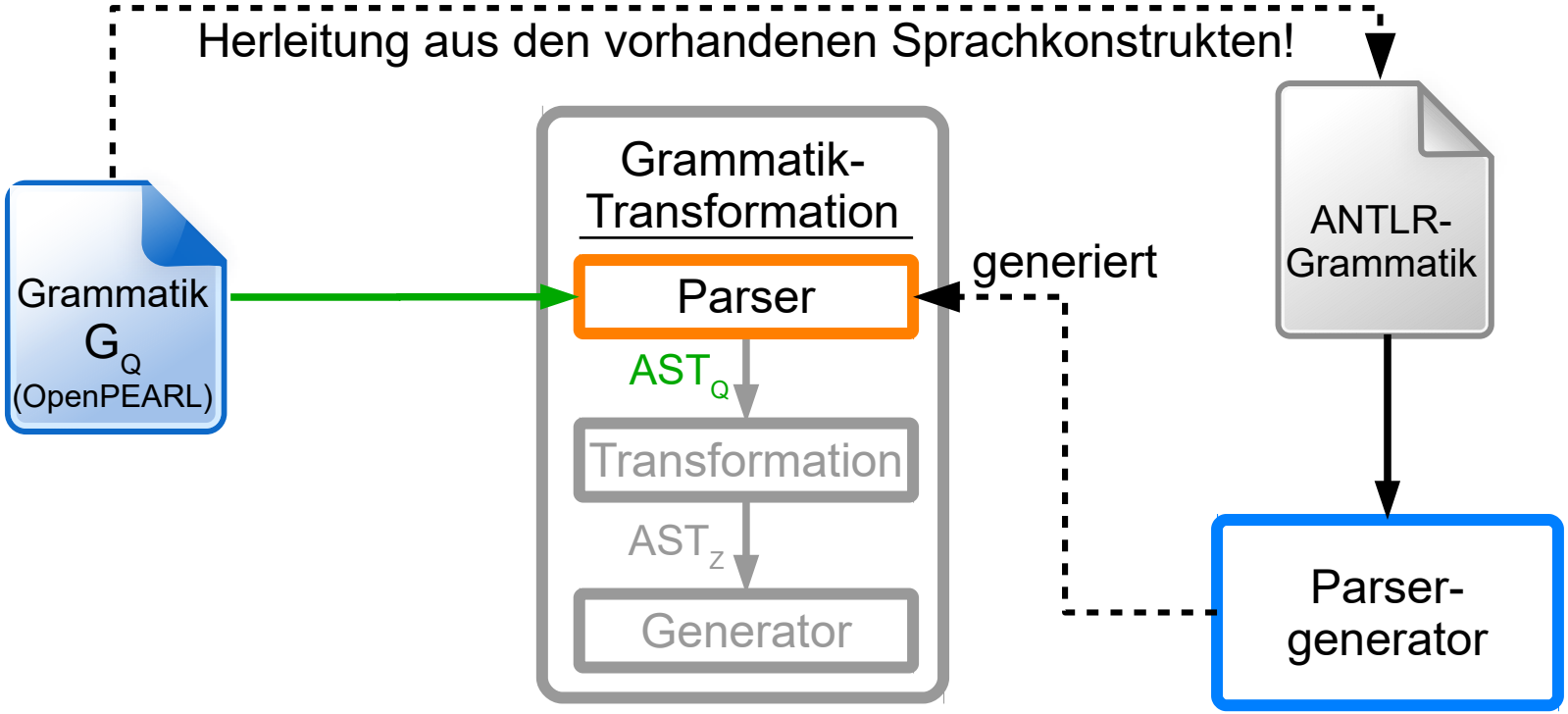
  Zusammenfassung und Ausblick

Ziel und Aufgaben



Parsen der Quellgrammatik

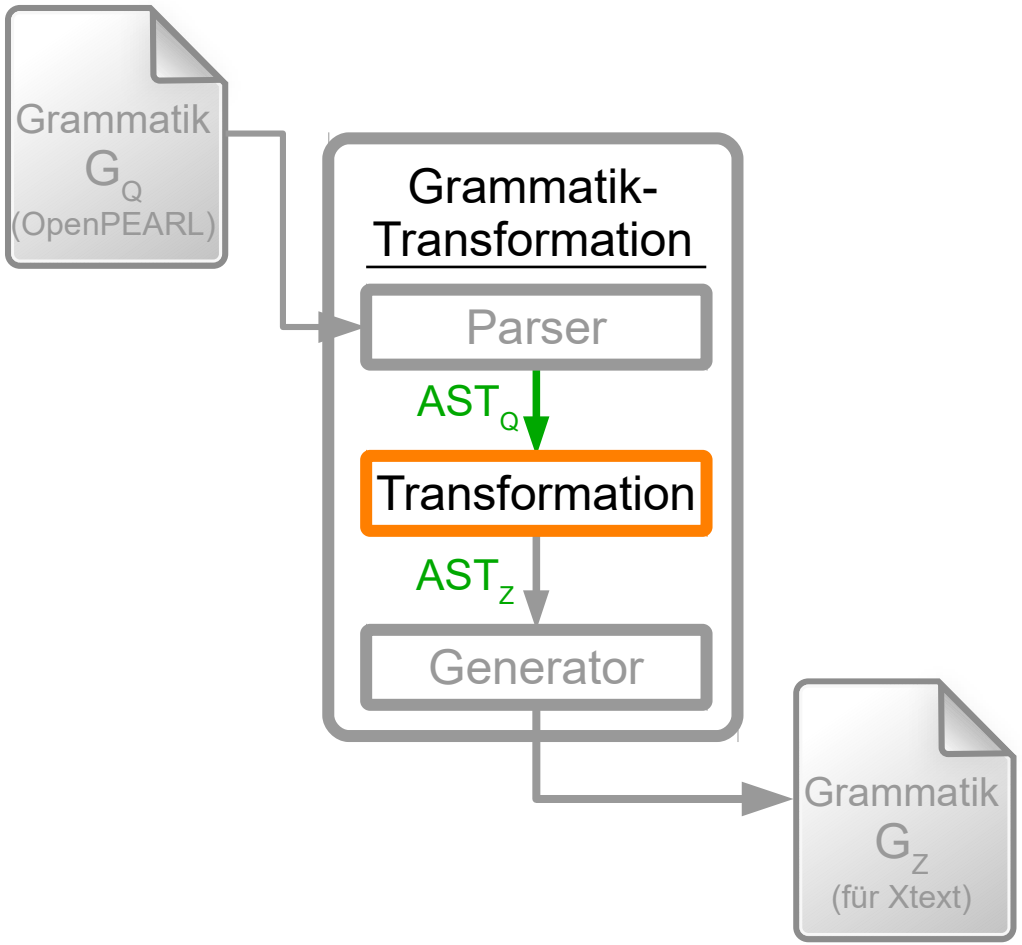
- Parser können unter Vorgabe einer Grammatik generiert werden!
- OpenPEARL-Grammatik ist in ANTLR verfasst! Zur Parsergenerierung ist die ANTLR-Grammatik erforderlich!



Transformation

- Beseitigung von …
  - nutzlosen Symbolen,

    (s. [1], [2] o. [4]: Kap. 3.2.2 u. 3.2.3)
  - Linksrekursionen,
  - Mehrdeutigkeiten.



Transformation: Beseitigung von Linksrekursionen

- Linksrekursive Grammatiken sind für linksableitende Parser ungeeignet!
- direkte Linksrekursion: $A \rightarrow Ax \mid y \quad \Rightarrow$ Transformation: $A \rightarrow yA', \quad A' \rightarrow xA' \mid \varepsilon$ ([1], [2])
- indirekte Linksrekursion: $A_1 \rightarrow A_2 \mid y, \quad A_2 \rightarrow A_1x \quad \Rightarrow$ Transformation: $A_1 \rightarrow A_2 \mid y, \quad A_2 \rightarrow A_2x \mid yx$ ([1], [2])

$A_2 \rightarrow yxA_2', \quad A_2' \rightarrow xA_2' \mid \varepsilon$

- Algorithmus zur Beseitigung von Linksrekursionen (nach [3], S. 258)

```
Bringe die Nonterminale in N in eine beliebige Reihenfolge von 1 bis n
for jedes i von 1 bis n do
 begin
  for jedes j von 1 bis i-1 do
   begin
    Ersetze jede Produktion der Form N[i] → N[j]a durch die Produktion N[i]→b1a|b2a|…|bka, wobei
    N[j]→b1|b2|…|bk alle aktuellen N[j] Produktionen sind. (a,b1,b2,..,bk ∈ (N ∪ T)*)
   end
   eliminiere direkte Linksrekursion in den N[i] Produktionen
 end
```

Transformation: Beseitigung von Linksrekursionen

- Schwächen des obigen Algorithmus:
  - Unnötige Ersetzungen = Unnötige strukturelle Änderungen!

$A_1 \rightarrow A_3 a_1$
$A_2 \rightarrow a_2 \mid a_3$
$A_3 \rightarrow A_2 a_4 \mid A_1 a_5 \quad \Rightarrow \quad A_3 \rightarrow a_2 a_4 \mid a_3 a_4 \mid A_3 a_1 a_5$

  - Grammatik sollte keine ε-Produktionen enthalten

$A_1 \rightarrow A_2 a_1 \mid a_2$
$A_2 \rightarrow A_3 A_1 a_3$
$A_3 \rightarrow a_4 \mid \varepsilon$

  - Grammatik sollte keine Zyklen enthalten

$A_1 \rightarrow A_2 a_1 \mid A_3$
$A_2 \rightarrow A_3 a_2 \mid a_3$
$A_3 \rightarrow A_1 \mid A_2 a_4 \quad \Rightarrow \quad A_3 \rightarrow A_3 a_2 a_1 \mid a_3 a_1 \mid A_3 \mid A_3 a_2 a_4 \mid a_3 a_4$

Transformation: Beseitigung von Linksrekursionen

Erweiterung des Algorithmus zur Beseitigung der Linksrekursion

```
Bringe die Nonterminale in N in eine beliebige Reihenfolge von 1 bis n
for jedes i von 1 bis n do
 begin
  for jedes j von 1 bis i-1 do
   begin
    if N[i] aus N[j] ableitbar? then
     begin
      Ersetze jede Produktion der Form N[i] → XN[j]a durch die Produktion N[i]→x1N[j]a|x2N[j]a|…|xmN[j]a
      und füge zusätzlich die Produktion N[i] → N[j]a hinzu, wobei X → x1 | x2 | … | xm alle X
      Produktionen außer X → ε sind! (mit X ∈ (N\{N[i],N[j]} ∪ T)* und X⇒*ε)

      Ersetze dann jede Produktion der Form N[i] → N[j]a durch die Produktion N[i]→b1a|b2a|…|bka, wobei
      N[j]→b1|b2|…|bk alle aktuellen N[j] Produktionen sind. (a,b1,b2,..,bk ∈ (N ∪ T)*)
     end
   end
   entferne Produktionen der Form N[i] → N[i]
   eliminiere direkte Linksrekursion in den N[i] Produktionen
 end
```

Transformation: Beseitigung von Mehrdeutigkeiten

- Beseitigung von Mehrdeutigkeiten
  - Beispiel aus nicht transformierten OpenPEARL-Grammatik:

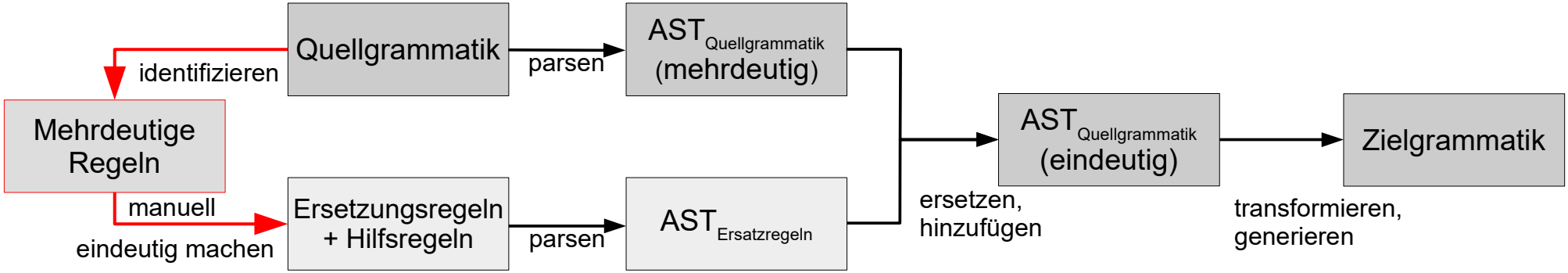
```
simpleType :
        typeInteger
      | typeFloatingPointNumber
      | typeBitString
      | typeCharacterString
      | typeTime
      | typeDuration
    ;
```

```
typeTime
    : type_clock
    | type_duration
    ;
```

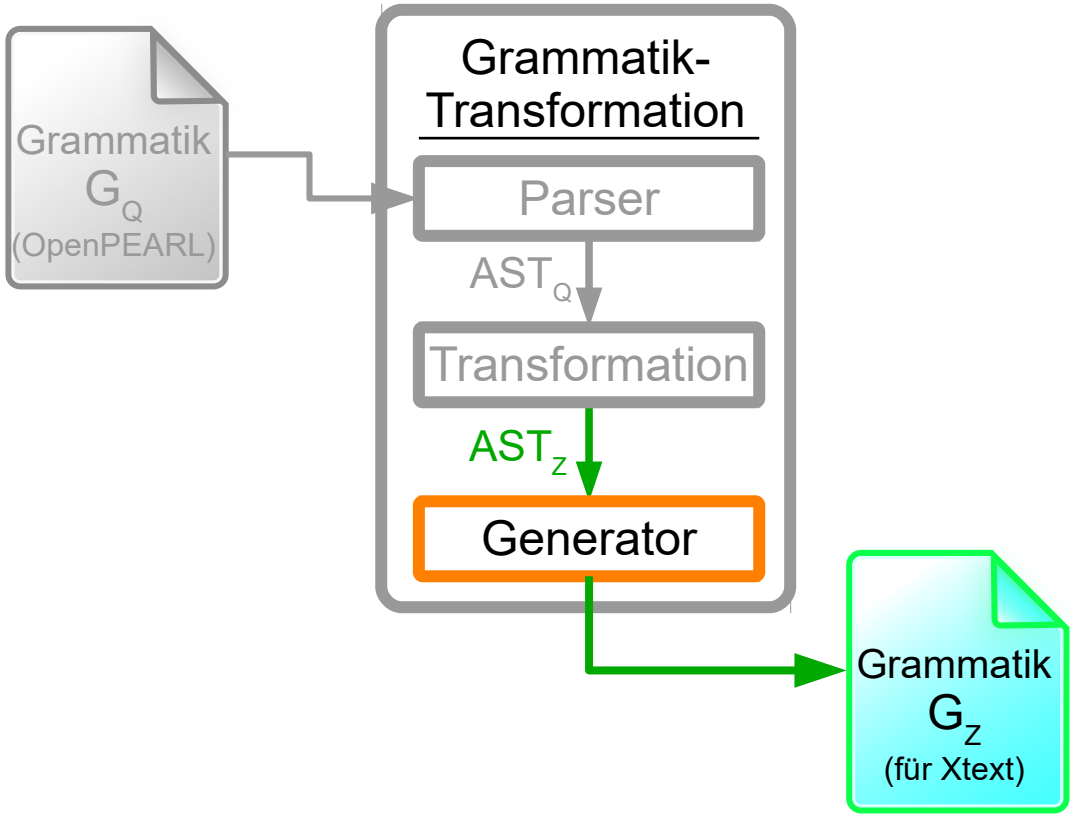
```
type_duration: ( 'DURATION' | 'DUR' );
```

```
typeDuration :
    'DURATION' | 'DUR'
    ;
```

  - Beseitigung mittels Ersetzung durch eindeutige Regeln



Generierung der Zielgrammatik

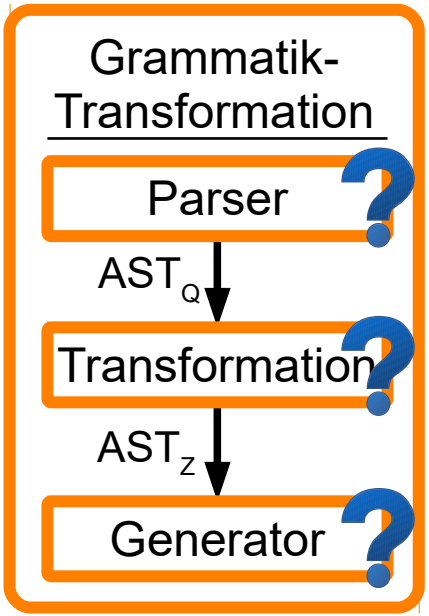


- Ausgabe muss den Notationsvorschriften der Sprache der Zielgrammatik entsprechen (Xtext-Notation)

Testen der Grammatiktransformation

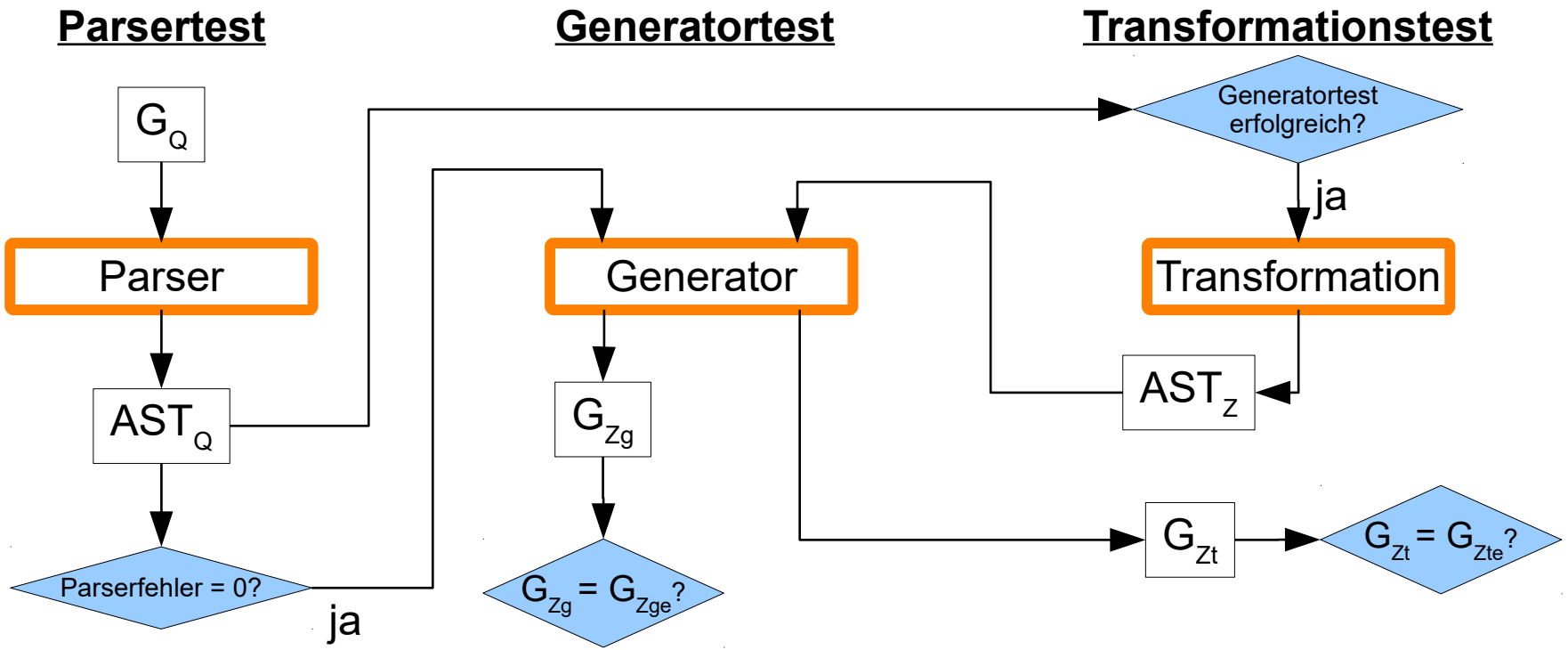
- Grundsätzliche Vorgehensweise
  - G_Q: Quellgrammatik, G_Z: Zielgrammatik, G_{Ze}: erwartete Zielgrammatik
  - G_Q zu G_Z transformieren
  - Wenn $G_Z = G_{Ze}$ ⇒ Test erfolgreich!
  - Wenn $G_Z \neq G_{Ze}$ ⇒ Test nicht erfolgreich!

→ Wo ist der Fehler?



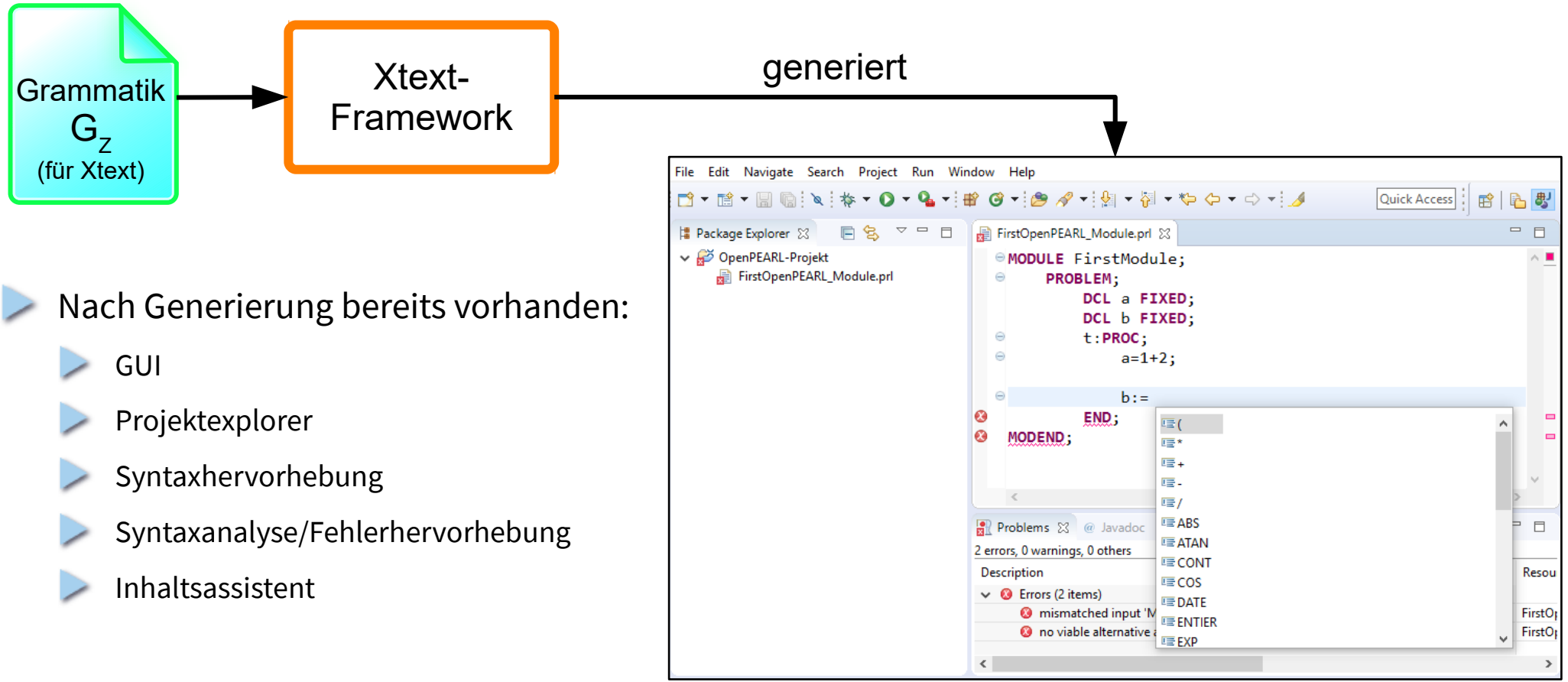
Testen der Grammatiktransformation

- Aufteilung in Parser-, Generator- und Transformationstests (Durchführung und Abhängigkeit in der genannten Reihenfolge)



Integration der Programmiersprache unter Eclipse

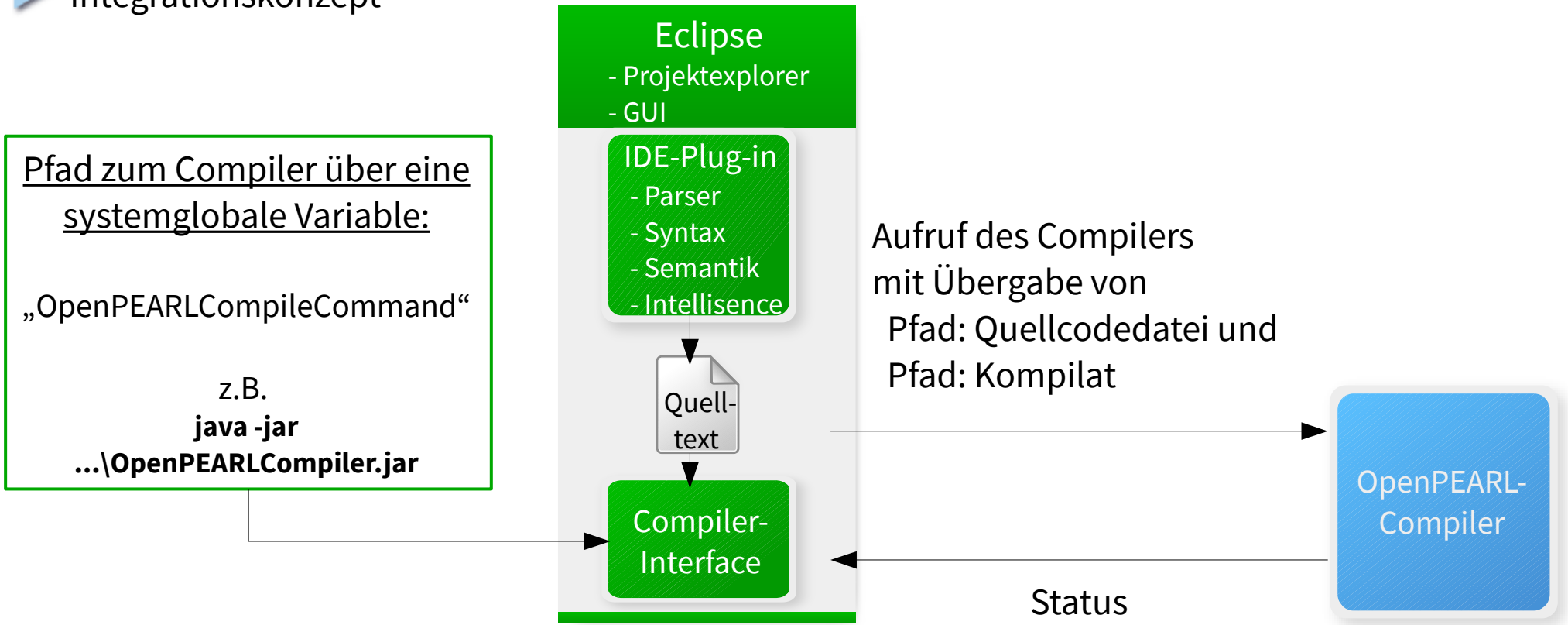
- Transformierte OpenPEARL-Grammatik in Xtext anwenden und Plug-in generieren lassen



- Nach Generierung bereits vorhanden:
  - GUI
  - Projektexplorer
  - Syntaxhervorhebung
  - Syntaxanalyse/Fehlerhervorhebung
  - Inhaltsassistent

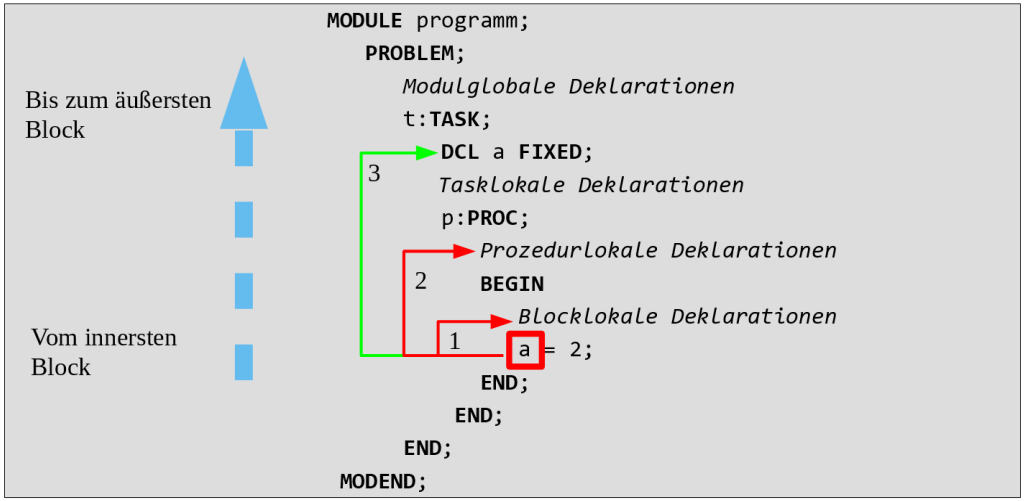
Compiler-Integration

- Integrationskonzept



Semantische Validierung

- Validierung von Bezeichnern innerhalb von Zuweisungen zum Programmierzeitpunkt!



- Umsetzung in Xtext? → @Check-Methoden!

```
@Check
public void validateIdentifierInAssignment(assignmentStatementOrCallStatement context) {
    // Wenn das assignmentStatementOrCallStatement-Objekt einen Bezeichner enthält
    // und es sich um eine Zuweisungsoperation handelt, dann suche nach der entsprechenden
    // Deklaration des Bezeichners innerhalb des Sichtbarkeitsbereichs!
}
```

Zusammenfassung und Bewertung der Ergebnisse

- Entwicklungsumgebung (auf Basis von Eclipse)
  - Umgesetzt mit Xtext und der transformierten OpenPEARL-Grammatik
  - Quelltext-Editor mit Syntaxhervorhebungen und -überprüfungen
  - Semantische Validierung: Bezeichner in Zuweisungsoperationen
  - Compiler: integriert als externe Anwendung, leicht austauschbar, Statusdarstellung in der Umgebung
  - Inhaltsassistent (kontextbezogen): Templates MODULE, TASK, PROCEDURE, BLOCK integriert
  - Einfacher Projektexplorer

- Programmierung mit PEARL wurde vereinfacht!
- Nach heutigen Maßstäben ist die entwickelte IDE jedoch noch recht rudimentär!

Zusammenfassung und Bewertung der Ergebnisse

- Grammatiktransformation
  - Herleitung ANTLR-Grammatik
  - Beseitigt nutzlose Symbole, Linksrekursionen und Mehrdeutigkeiten
  - Linksrekursionen können in Grammatiken mit Zyklen und ε-Produktionen beseitigt werden, Grammatikstruktur wird weitestgehend beibehalten
  - Zur Beseitigung von Mehrdeutigkeiten müssen eindeutige Regeln zur Verfügung gestellt werden
  - Transformation der Notation: ANTLR → Xtext!

- Automatisiert! → Dadurch schnelle Anpassung der OpenPEARL-IDE an die Weiterentwicklung des Compilers möglich!

Ausblick

- Erweiterung der semantischen Validierung
  - Ausweitung der Bezeichnervalidierung auf alle Anweisungstypen
  - Doppeldefinitionen
  - Typprüfungen
  - …
- Inhaltsassistent
  - Kontextbezogenes anbieten von deklarierten Variablen und Prozeduren
  - Weitere Templates
  - …
- Konzept SW-Projektorganisation im Projektexplorer
- Refactoring, Navigation im Quellcode, Debugging, etc.

Quellen

[1] P. Rechenberg, H. Mössenböck; Ein Compiler-Generator für Mikrocomputer; Carl Hanser Verlag München; 1988

[2] J.E. Hopcroft, J.D. Ullman; Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie; Oldenbourg Verlag München Wien; 2000

[3] Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman; Compiler Prinzipien, Techniken und Werkzeuge 2. aktualisierte Auflage; Pearson Studium; 2008

[4] Andreas Enns; Graphische Entwicklungsumgebung für eine Echtzeitprogrammiersprache; Masterarbeit an der Fernuni in Hagen; 2017