anderScore
trust in competence

DSLs mit Xtext entwerfen

17.08.2012, A. Arnold

Inhalt

Was sind DSLs?

anderScore trust in competence

- Domain Specific Languages
- für einen Anwendungsfall (Domäne) entworfen
- Grafisch oder textuell

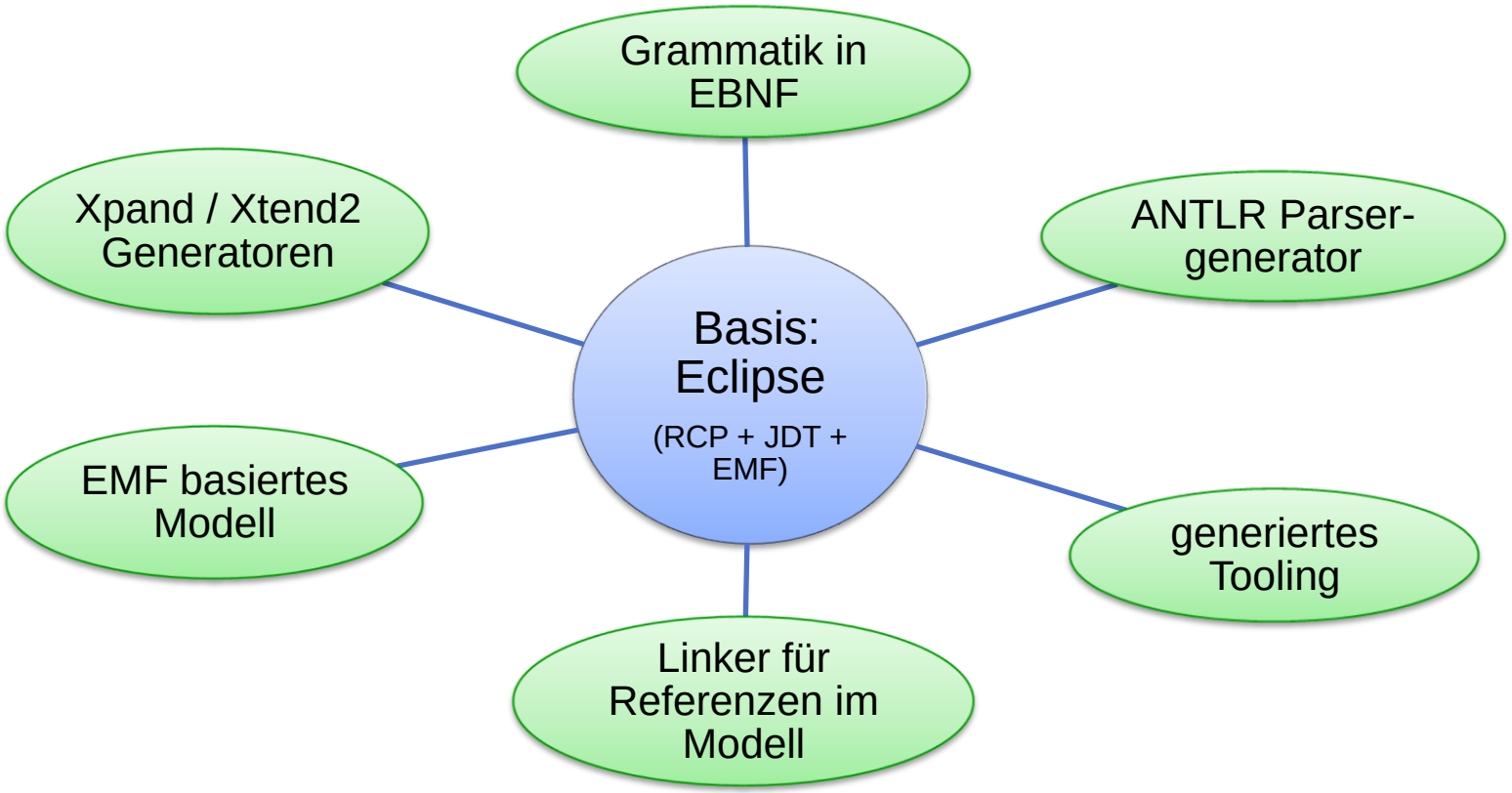
anderScore
trust in competence

XTEXT KONZEPTE

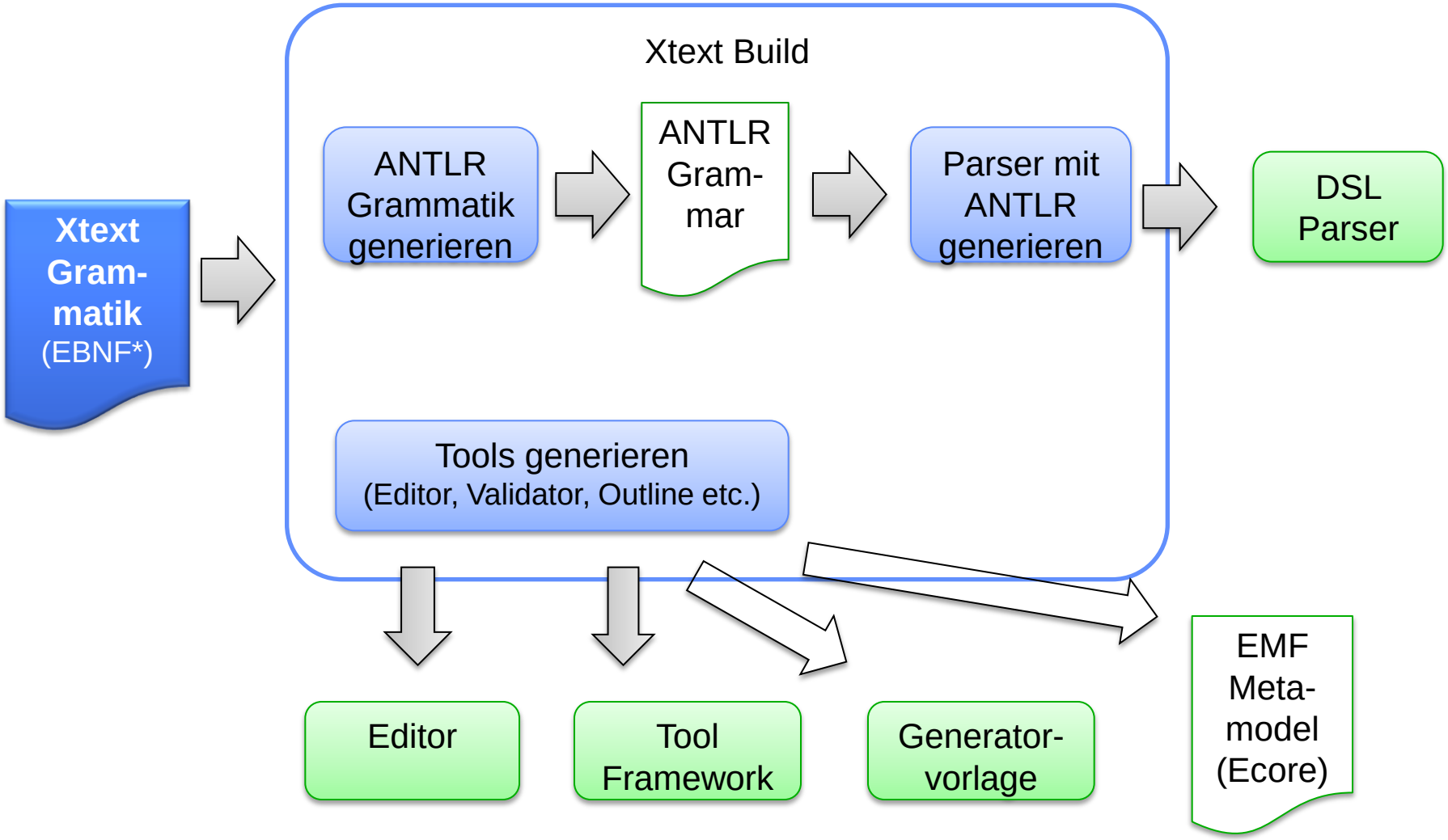
Die Idee …

anderScore trust in competence

DSL Workbench für textuelle DSLs + Codegenerierung

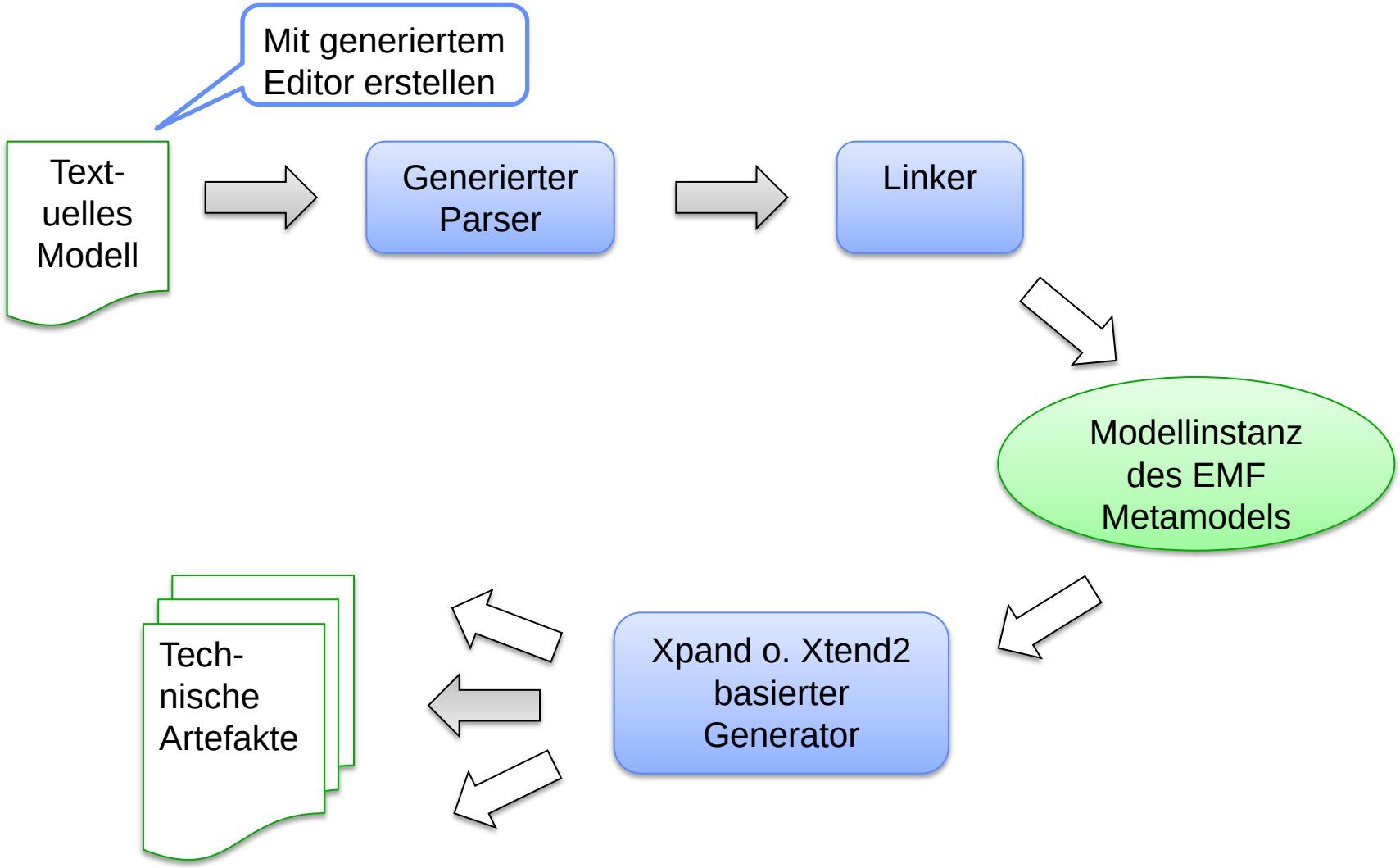


Sprache definieren



*EBNF: Erweiterte Backus-Naur Form (Form in der Grammatiken beschrieben werden)

Modell erstellen + nutzen



anderScore
trust in competence

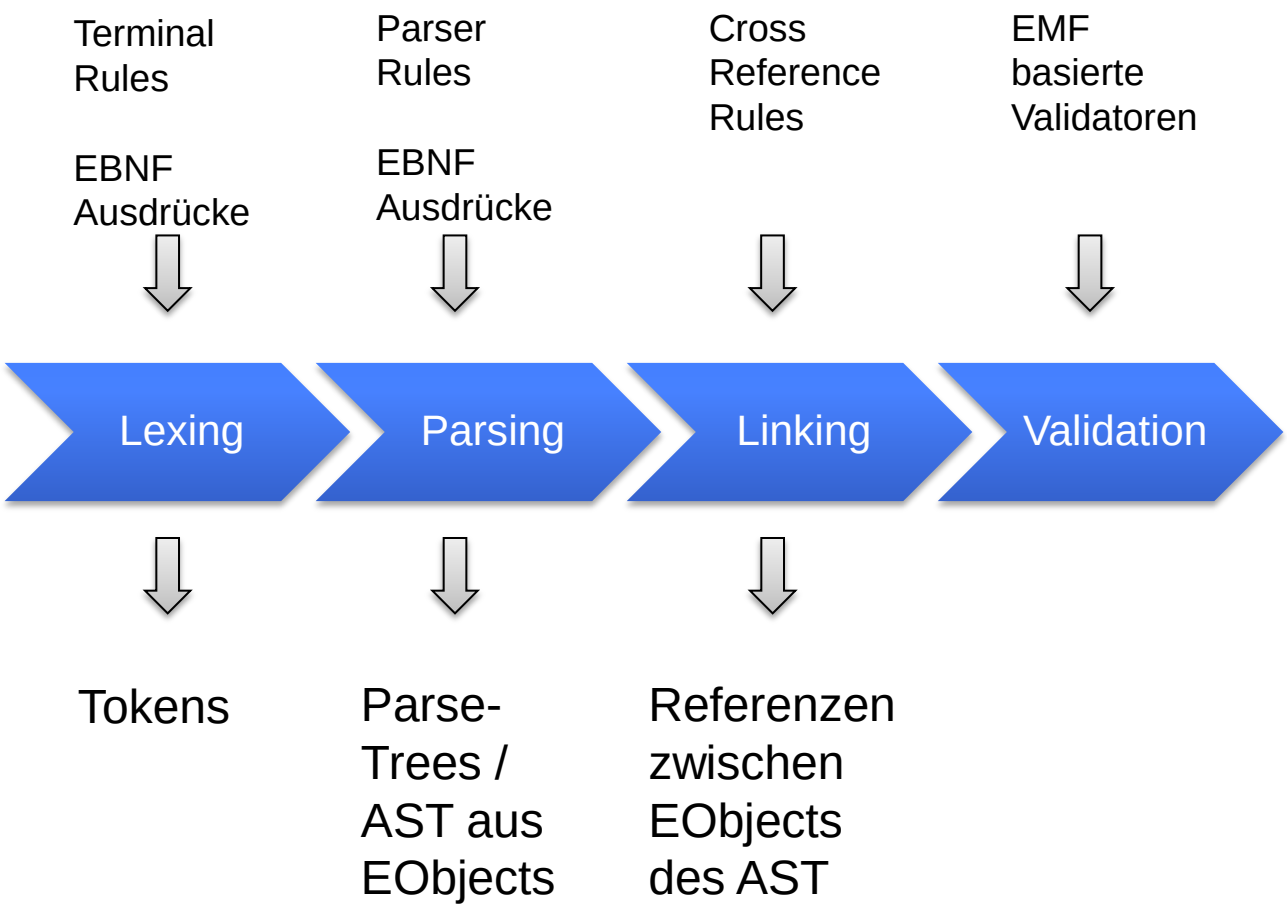
Parser und Aufbau von Grammatiken

EINFÜHRUNG INS SPRACHDESIGN

Grammatiken

- DSLs sind formale Sprachen
- Formale Sprachen werden durch Grammatiken beschrieben
  - Struktur der Grammatik ist ein AST – Abstract Syntax Tree

Parser Phasen



Terminal Rules

Der Lexer braucht eine Beschreibung der elementaren Textbausteine

- Terminal: atomares Element (Token) einer Grammatik
  - Gruppieren Zeichenfolgen zu Tokens
  - gelten überall
  - Lexer erzeugt einzelne Tokens anhand von Terminal Rules

Bsp.:

```
terminal ID : ('^')?  ('a'..'z' | 'A'..'Z' | '_')  ('a'..'z' | 'A'..'Z' | '_' | '0'..'9')*;
```

Eine **ID** besteht aus Buchstaben, Ziffern und _,
beginnt immer mit einem Buchstaben
oder ^ und einem Buchstaben

Ausdrücke

Grammatiken benötigen Ausdrücke

- „Erweiterte Backus Naur Form“
- Kardinalitäten festlegen:

kein Operator (Default)	genau einmal
?	optional
+	beliebig oft, mind. einmal
*	beliebig oft, optional

- ( ) zum Festlegen der Assoziativität der Ausdrücke (Gruppierung)

Ausdrücke Teil2

- Character Ranges: ..

  Bsp.:

  ```
  terminal INT returns ecore::EInt: ('0'..'9')+;
  ```

- Wildcards: . (Punkt)
  - hier kann ein beliebiges Zeichen stehen

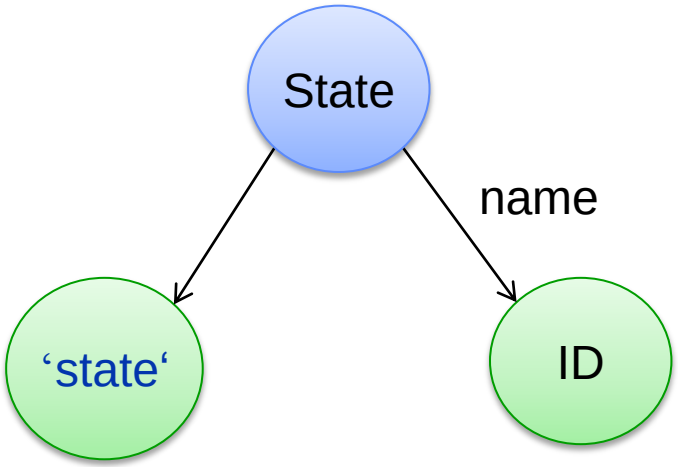
- Alternativen:
  - getrennt durch |

  Bsp.:

  ```
  terminal WS : (' ' | '\t' | '\r' | '\n')+;
  ```

Parser Rules

Parser Rules beschreiben die Struktur



State:
'state' name=ID

Parser Rules sind kontextuell (gelten **nicht** überall)

Parser Rules - Schlüsselwörter

Schlüsselwörter helfen Strukturen zu erkennen

- Art von Terminal Rules
- stehen in ‘ ‘

Bsp.: ‘*keyword*‘

keyword ist Schlüsselwort,
hilft dem Parser beim Erkennen der anzuwendenden Parser Rule

Rule Calls

Rules kann man wiederverwenden

- Regel aus anderen Regeln aufrufen
- Entsprechen einer EMF EClass des Metamodels

Bsp.:

```
terminal DOUBLE : INT '.' INT;
```

INT ist eine andere Terminal Rule

=>

ein DOUBLE besteht aus einem INT, einem Punkt und noch einem INT

Parser Rules - Assignments

anderScore trust in competence

Parsing

Durch Zuweisungen werden Werte im Modell (AST) gehalten

- Wert einem Attribut zuweisen
  - `=` einfache Zuweisung
  - `+=` Listenzuweisung, mind. Ein Element
  - `=?` Boolsche Zuweisung, Wert true, wenn rechte Seite der Regel vorhanden

Bsp.:

State :

```
'state' name=ID
('actions' '{' (actions+=[Command])+ '}')?
(transitions+=Transition)*
'end';
```

Attribut name bekommt Wert des Terminals ID zugewiesen

Enum Rules

anderScore trust in competence

Parsing

- Aufzählung gültiger Werte
  - Erzeugen eine EMF Enumeration

Bsp.:

```
enum ChangeKind :
    ADD | MOVE | REMOVE
;
```

Überall wo ein ChangeKind stehen kann, kann “ADD”, “MOVE” o. “REMOVE” stehen

Cross References

Cross Refs verknüpfen Modellelemente zu einem Graph

- Attribut erhält Querverweis auf anderes Modellelement
  - verweisen auf andere Parser Rule
  - in eckigen Klammern
  - Auflösung erfolgt beim Linking
    - Linking per Default anhand des Wertes im Attribut „name“

Bsp.:

Transition :

event=[Event] '=>' state=[State]

;

Attribut **event** verweist auf eine Instanz von **Event,** im Text steht die ID (der Name) des **Event**s

anderScore
trust in competence

HANDS ON

Referenzen

- Xtext SDK
  - http://www.eclipse.org/Xtext/download.html