



Generieren vs. Interpretieren

Die andere Seite von MDSD

ex|Xcellent
solutions

völter.de
ingenieurbüro für softwaretechnologie

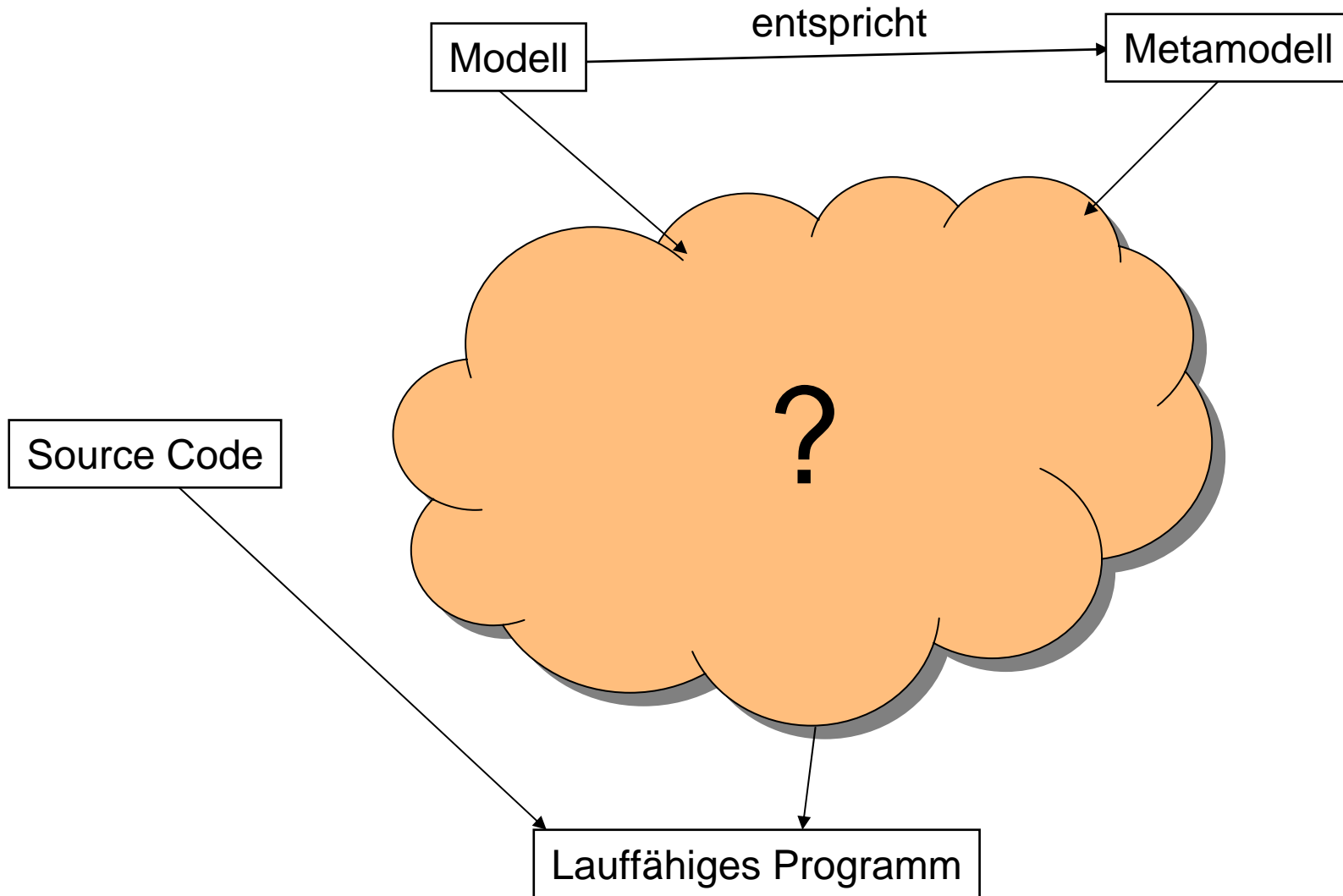


- Begriffsklärung
 - Generiert vs. Generisch
 - Gemeinsamkeiten
 - Generieren
 - Interpretieren
- Beispiel: Statemachine
 - Gemeinsamkeiten
 - Generierender Ansatz
 - Generischer Ansatz
- Praxisbeispiele
 - Typische Anwendungsszenarien
 - Konkrete Projektbeispiele
- Vergleich und Bewertung
- Fazit

Generiert vs. Generisch

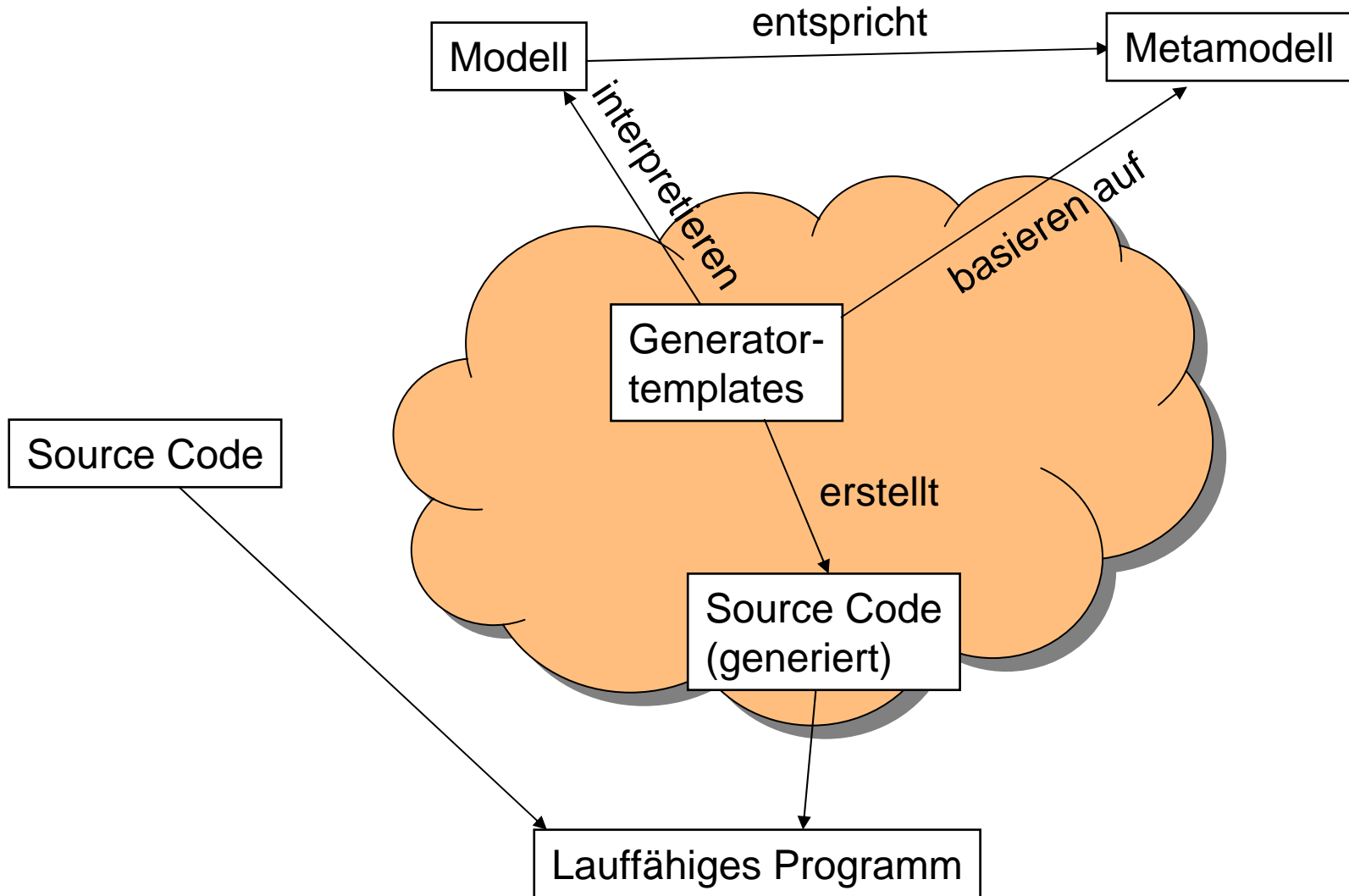
- *Generiert*
 - **Erzeugen** von Quelltext, Konfigurationsdateien und anderer textueller Artefakte aus einem Modell heraus
 - **Statisch**
 - Vor dem **Kompilieren**
 - Ergebnis: Viel (generierter) spezifischer Quelltext
- *Generisch*
 - **Interpretieren** von (Modell-)Informationen im Programm
 - **Dynamisch**
 - Zur **Laufzeit**
 - Ergebnis: Wenig (handgeschriebener) allgemeiner Quelltext

- **Abstraktion** des Problembereichs
- Definition eines **Metamodells**
- **Instantiierung** des Metamodells
- **Auswertung** des Metamodells
- Gleiches **Ziel**
 - Vermeidung wiederholter und manueller Programmierbarkeit

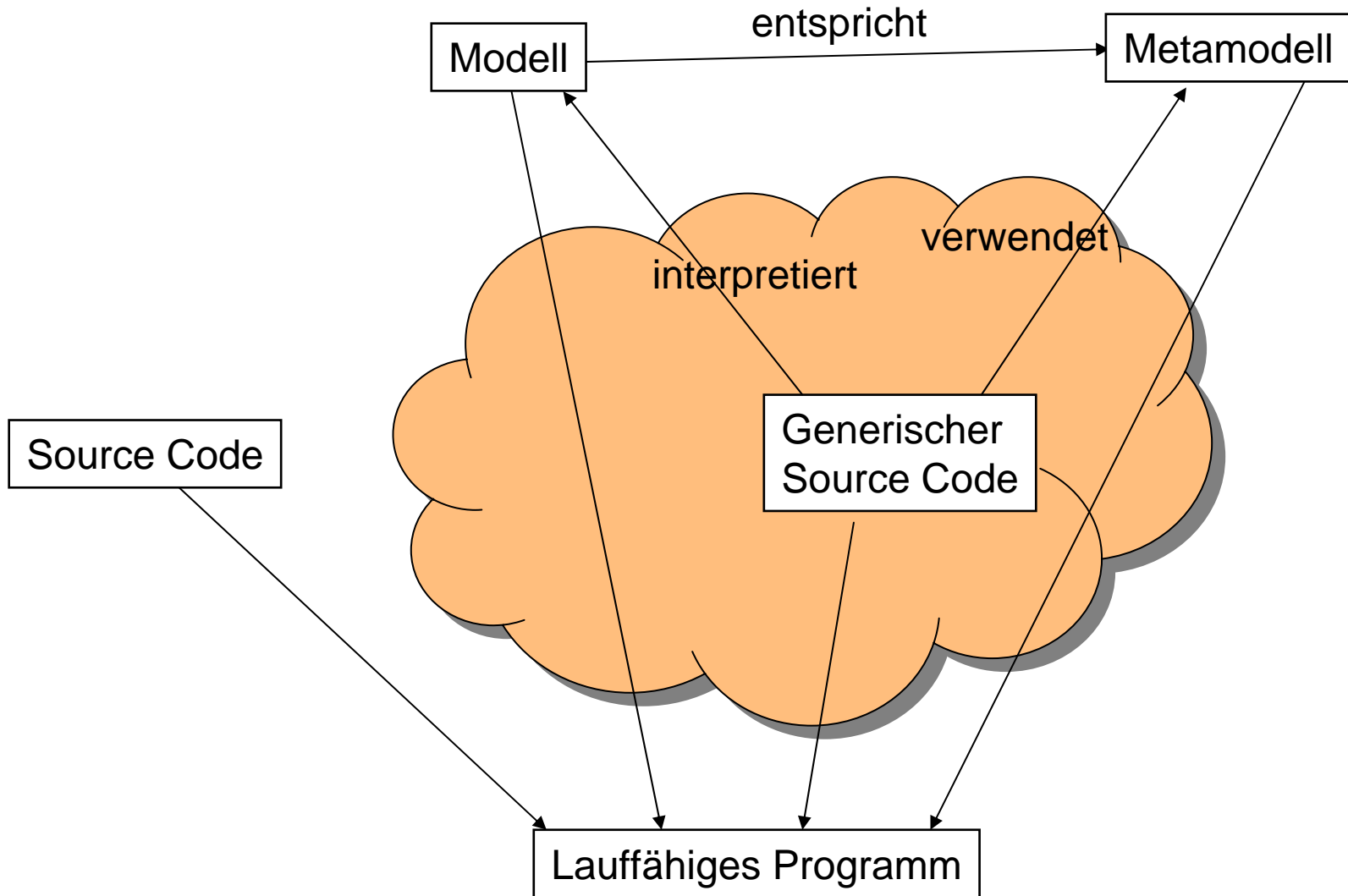


Generieren: Vorgehensweise

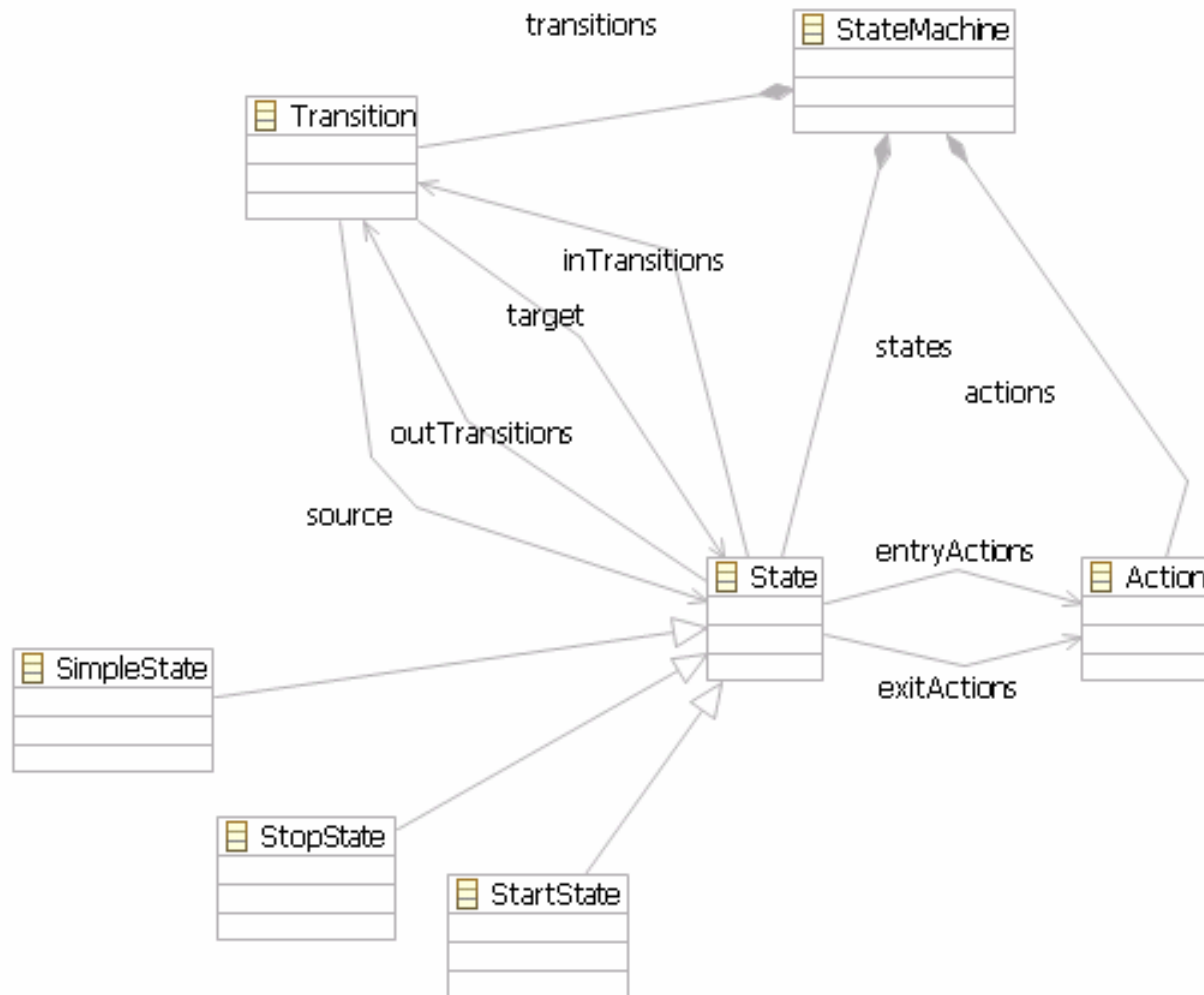
- Erstellen einer **nicht-generischen Implementierung**
 - Auch genannt Referenzimplementierung
- **Abstraktion** der Implementierung
 - Definition des **Metamodells**
 - Erstellung von Generator-Templates zur Generierung des Codes aus dem Modell
- In der Praxis zusätzlich:
 - Herausfaktorisierung der allen Instanzen **gemeinsamen Bestandteile** in Oberklassen und Bibliotheken
 - Generierung von **Hooks**, um manuell zu schreibenden Code zu integrieren



- Erstellen einer **nicht-generischen Implementierung**
- **Abstraktion** der Implementierung
 - Definition des **Metamodells**
 - Gemeinsame Bestandteile in parametrisierte Klassen und Methoden => **Generischer Source Code**
- Erstellung der **Grundkomponente** zum Laden des Modells
- In der Praxis zusätzlich: *Generierung* von
 - Basisklassen für Hooks und Callbacks
 - Zugriffsklassen auf das Modell und Metamodell



Statemachine: Gemeinsam Metamodell



Statemachine [Generiert] (1)

Generator Templates

```
Actions.xpt Statemachine.xpt X
«IMPORT simpleSM»
«EXTENSION templates::GeneratorUtil»

«DEFINE file FOR StateMachine»
«FILE basePath() + "/Abstract" + name.toFirstUpper() + ".java"-»
package «basePath()»;

public abstract class «implBaseClassName()» implements «actionsInterfaceName()» {

    private «statesEnumName()» currentState = «initialState.stateId(this)»;
    private boolean terminated = false;

    public void handleEvent( «eventsEnumName()» event ) {
        if ( terminated ) throw new RuntimeException( "this sm is terminated!" );

        switch ( currentState ) {
            «FOREACH states AS s->»
            case «s.shortStateId()»:
                «FOREACH s.transitions AS t->»
                if ( event == «t.event.eventId(this)» ) {
                    «EXPAND executeTransition(this) FOR t»
                    break;
                }
                «EXPAND handleIllegalTransition»
            «ENDFOREACH»
            break; // break out if no suitable transition has been found!
        «ENDFOREACH»
        }
    }

    public «statesEnumName()» getCurrentState() {
        return currentState;
    }
}
«ENDFILE»
«ENDDDEFINE»

«DEFINE handleIllegalTransition FOR StateMachine»
«ENDDDEFINE»

«DEFINE executeTransition(StateMachine sm) FOR Transition»
«FOREACH actions AS a->»
    this.«a.methodName()»();
«ENDFOREACH»
    currentState = «to.stateId(sm)»;
«ENDDDEFINE»
```

- Der **blaue Text** wird direkt in die Zieldatei generiert.
- Die **lila großgeschriebenen Wörter** sind Keywords der xPand Template-Sprache
- **Schwarzer Text** sind Modell-Zugriffe
- DEFINE...END-DEFINE Blöcke werden **Templates** genannt

Statemachine [Generiert] (2)

Generierter Code

```
*ExampleStateMachineImplBase.java x ExampleGenericStateMachineImplementation.java
public abstract class ExampleStateMachineImplBase
    extends oaw4.demo.emf.statemachine.platform.AbstractStateMachine {
    public static final int STATE_START = 1000 + 0;
    ...
    public static final int STATE_NOTAUS = 1000 + 5;
    public static final int EVENT_RUN = 2000 + 0;
    ...
    public static final int EVENT_INIT = 2000 + 4;

    public ExampleStateMachineImplBase() { currentState = STATE_INIT; }

    public void trigger(int event) {
        if (terminated) { throw new RuntimeException("bin schon tot!"); }

        switch (currentState) {
            case (STATE_INIT):
                if (event == EVENT_RUN) {
                    System.out.println("transitioning from init into passive");
                    action2();
                    currentState = STATE_PASSIVE;
                }
                break;
            case (STATE_PASSIVE):
                if (event == EVENT_CALLRECEIVED) {
                    System.out.println("transitioning from passive into calling");
                    currentState = STATE_CALLING;
                    action1();
                }
                if (event == EVENT_STOP) { terminated = true; }
                break;
            case (STATE_CALLING):
                if (event == EVENT_CALLTERMINATED) {
                    System.out.println("transitioning from calling into passive");
                    action2();
                    currentState = STATE_PASSIVE;
                }
                break;
        }
    }

    protected abstract void action1();
    protected abstract void action2();
}
```

Statemachine [Generiert] (3)

Integration manuellen Codes

```
ExampleStateMachineImplementation.java X  
  
public class ExampleStateMachineImplementation extends  
    ExampleStateMachineImplBase {  
  
    public int action1Counter;  
    public int action2Counter;  
  
    protected void action1() {  
        action1Counter++;  
    }  
  
    protected void action2() {  
        action2Counter++;  
    }  
  
}
```

Statemachine [Generisch] (1)

Abstraktion der Implementierung

```
ExampleStateMachineImplBase.java X
public void trigger(int event) {
    if (terminated) {
        throw new RuntimeException("bin schon tot!");
    }

    switch (currentState) {
    case (STATE_INIT):

        if (event == EVENT_RUN) {
            System.out.println("transitioning from init into passive");

            action2();

            currentState = STATE_PASSIVE;
        }

        if (event == EVENT_NOTAUS_INIT) {
            terminated = true;
        }

        break;

    case (STATE_PASSIVE):

        if (event == EVENT_CALLRECEIVED) {
            System.out.println("transitioning from passive into calling");

            currentState = STATE_CALLING;

            action1();
        }

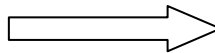
        if (event == EVENT_STOP) {
            terminated = true;
        }

        if (event == EVENT_NOTAUS_PASSIVE) {
            terminated = true;
        }

        break;

    case (STATE_CALLING):

        if (event == EVENT_CALLTERMINATED) {
```



```
GenericStateMachine.java X
package oaw4.demo.emf.statemachine.platform;

import sm.Action;

public abstract class GenericStateMachine {
    protected StateMachine sm = null;
    protected State currentState = null;
    protected boolean terminated = false;

    protected GenericStateMachine(StateMachine sm) {}

    public State getCurrentState() {}

    public boolean isTerminated() {}

    public State trigger(Transition transition) {
        if (terminated) throw new RuntimeException("bin schon tot!");

        if (transition.getSource().equals(currentState)) {
            for (Object o : currentState.getExitActions()) {
                perform((Action)o);
            }
            currentState = transition.getTarget();
            for (Object o : currentState.getEntryActions()) {
                perform((Action)o);
            }
            if (currentState instanceof StopState) {
                terminated = true;
            }
        }

        return currentState;
    }

    protected abstract void perform(Action a);
}
```

Statemachine [Generisch] (2)

Generierung von Applikations-Hooks

```
GenericRoot.xpt X
«IMPORT sm»
«EXTENSION templates::java»

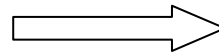
«DEFINE Root FOR StateMachine»
  «FILE "Generic"+implBaseClassName()+".java"»
  public abstract class Generic«implBaseClassName()»
    extends oaw4.demo.emf.statemachine.platform.GenericStateMachine {

    public Generic«implBaseClassName()»(sm.StateMachine sm) {
      super(sm);
    }

    public void perform( sm.Action a ) {
      «FOREACH actions AS a»
        if ( a.getName().equals("«a.name»") ) {
          «a.name»();
        }
      «ENDFOREACH»
    }

    «FOREACH actions AS a»
      protected abstract void «a.name»();
    «ENDFOREACH»

  }
«ENDFILE»
«ENDDEFINE»
```



```
GenericExampleStateMachineImplBase.java X

public abstract class GenericExampleStateMachineImplBase extends
oaw4.demo.emf.statemachine.platform.GenericStateMachine {

  public GenericExampleStateMachineImplBase(sm.StateMachine sm) {
    super(sm);
  }

  public void perform(sm.Action a) {
    if (a.getName().equals("action1")) {
      action1();
    }

    if (a.getName().equals("action2")) {
      action2();
    }
  }

  protected abstract void action1();

  protected abstract void action2();
}
```

Statemachine [Generisch] (3)

Präsentation und Interaktion durch den Benutzer

```

GenericStateMachineTest.java
import java.io.BufferedReader;

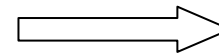
public class GenericStateMachineTest {
    public static void main(String[] args) {
        ExampleGenericStateMachineImplementation sm =
            new ExampleGenericStateMachineImplementation(loadStateMachine());

        while(!sm.isTerminated()) {
            printOptions(sm);
            int index = readTransition();
            sm.trigger((Transition)sm.getCurrentState().getOutTransitions().get(index));
        }
        System.out.println("Your StateMachine has terminated.");
    }

    public static void printOptions(ExampleGenericStateMachineImplementation sm) {
        System.out.println("You are in State " + sm.getCurrentState().getName());
        System.out.println("action1Counter is " + sm.action1Counter);
        System.out.println("action2Counter is " + sm.action2Counter);
        System.out.println("Your options are:");
        for (int i=0; i<sm.getCurrentState().getOutTransitions().size(); i++) {
            System.out.println("(" + i + ") " +
                ((Transition)sm.getCurrentState().getOutTransitions().get(i)).getName());
        }
    }

    public static int readTransition() {
    }

    private static StateMachine loadStateMachine() {
    }
}
    
```



```

Console Problems Javadoc
<terminated> GenericStateMachineTe:
You are in State init
action1Counter is 0
action2Counter is 0
Your options are:
(0) run
>> 0
You are in State passive
action1Counter is 0
action2Counter is 1
Your options are:
(0) callReceived
(1) stop
>> 0
You are in State calling
action1Counter is 1
action2Counter is 1
Your options are:
(0) callTerminated
>> 0
You are in State passive
action1Counter is 1
action2Counter is 2
Your options are:
(0) callReceived
(1) stop
>> 0
You are in State calling
action1Counter is 2
action2Counter is 2
Your options are:
(0) callTerminated
>> 0
You are in State passive
action1Counter is 2
action2Counter is 3
Your options are:
(0) callReceived
(1) stop
>> 1
Your StateMachine has terminated.
    
```

Typische Anwendungsszenarien

- Generierung
 - Quellcode, der bestimmten APIs genügen muss
 - Konfigurationsfiles
 - Dokumentation
 - Build Files
- Interpretiert
 - Verhalten (State Machines, Rule Engines, ...)
 - Datenvalidierung
 - Datentransformation
 - GUIs

Praxisbeispiele: Rahmenbedingungen

- Fachliches Umfeld: Verwaltungssystem für Teile aus dem Bereich Fahrzeugelektrik (Kabel, Stecker, Dichtungen, etc.)
- Technisches Umfeld:
 - Oracle Datenbank, > 100 Tabellen
 - O/R Mapping Ansatz (TopLink)
 - Rich Client (Swing) zur Dateneingabe
 - Thin Client (wingS) zur Datenrecherche
- Systemumfeld
 - Benutzer mit unterschiedlichen Sichtweisen auf die Daten
 - Datenimport/export mit mehreren Systemen
 - Datenmodell und Systemumgebung im Wandel

Praxisbeispiele: Modellinformationen

- Pre-EMF-Ära
- Pragmatischer Ansatz mit eigenem Metamodell
 - Klassen, Attribute, Datentypen, Beziehungen, Constraints, ...
- Generiert als Konstanten

```
SonderleitungMeta.java
package de.excellent.connect.business.bibliotheksdaten.generated.meta;

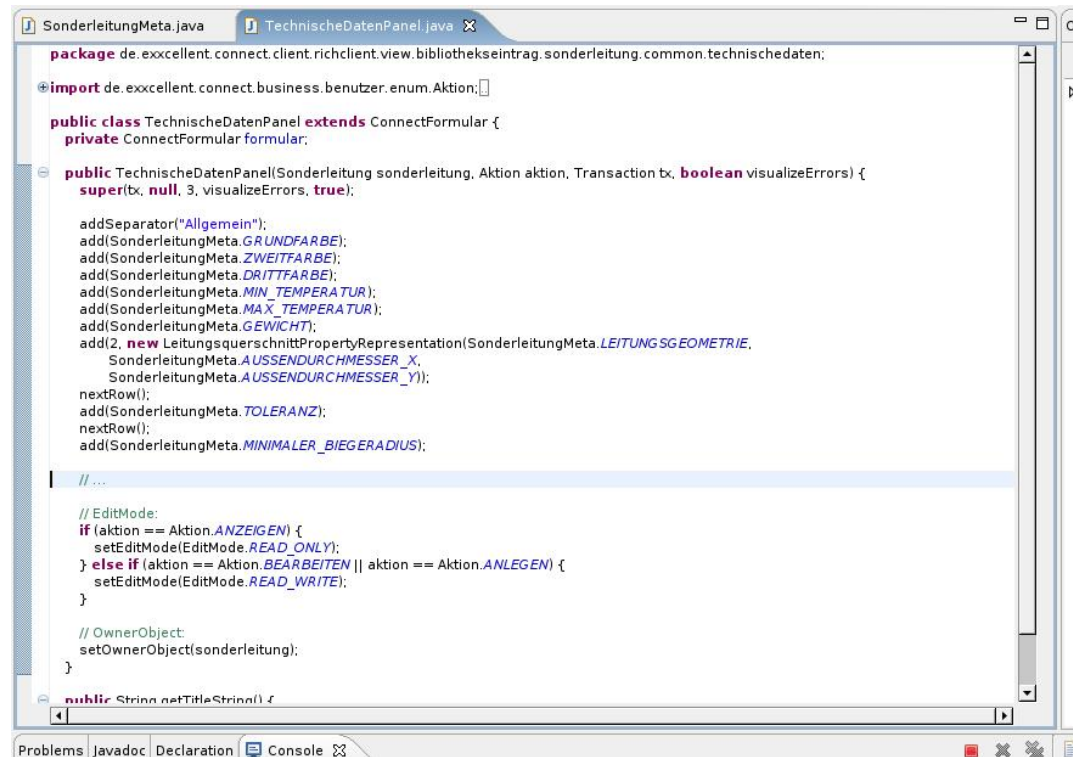
import de.excellent.connect.business.basisdaten.bo.Symbol;

public class SonderleitungMeta implements Bean {

    public final static SonderleitungMeta CLASS = new SonderleitungMeta();
    private Collection properties = null;
    public final static ConstraintAssociationImpl MINIMALERBIEGERADIUS_WERTMITEINHEITVOLLSTAENDIGCONSTRAINT = new de.excellent
    public final static ConstraintAssociationImpl MINIMALERBIEGERADIUS_WERTMITEINHEITGROESSERNULLCONSTRAINT = new de.excellent
    public final static AttributeProperty MINIMALER_BIEGERADIUS = new AttributePropertyImpl(CLASS, "minimalerBiegeradius", false, false
    public final static ConstraintAssociationImpl GEWICHT_WERTMITEINHEITVOLLSTAENDIGCONSTRAINT = new de.excellent.plexx.impl.mo
    public final static ConstraintAssociationImpl GEWICHT_WERTMITEINHEITGROESSERNULLCONSTRAINT = new de.excellent.plexx.impl.mo
    public final static AttributeProperty GEWICHT = new AttributePropertyImpl(CLASS, "gewicht", false, false, Arrays.asList(new Object[] {
    public final static RelationProperty BESTANDEILE = new RelationPropertyImpl(CLASS, "bestandteile", false, false, null, Sonderleitungs
    public final static ConstraintAssociationImpl GRUNDFARBE_SONDERLEITUNGGRUNDFARBECONSTRAINT = new de.excellent.plexx.impl.r
    public final static RelationProperty GRUNDFARBE = new RelationPropertyImpl(CLASS, "grundfarbe", false, false, Arrays.asList(new Obj
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERY_WERTMITEINHEITVOLLSTAENDIGCONSTRAINT = new de.excellent
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERY_WERTMITEINHEITGROESSERNULLCONSTRAINT = new de.excellent
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERY_LEITUNGSGEOMETRIECONSTRAINT = new de.excellent.plexx.impl
    public final static AttributeProperty AUSSENDURCHMESSER_Y = new AttributePropertyImpl(CLASS, "aussendurchmesserY", false, false
    public final static AttributeProperty SCHIRMUNG_VORHANDEN = new AttributePropertyImpl(CLASS, "schirmungVorhanden", true, true, n
    public final static AttributeProperty TOLERANZ = new AttributePropertyImpl(CLASS, "toleranz", false, false, null, new Object[] { {de.e
    public final static ConstraintAssociationImpl DRITTFARBE_SONDERLEITUNGDRITTFARBECONSTRAINT = new de.excellent.plexx.impl.mo
    public final static RelationProperty DRITTFARBE = new RelationPropertyImpl(CLASS, "drittfarbe", false, false, Arrays.asList(new Object
    public final static ConstraintAssociationImpl CADNR_MINMAXLENGTHCONSTRAINT = new de.excellent.plexx.impl.model.ConstraintAssc
    public final static ConstraintAssociationImpl CADNR_UNIQUECONSTRAINT = new de.excellent.plexx.impl.model.ConstraintAssociation
    public final static AttributeProperty CAD_NR = new AttributePropertyImpl(CLASS, "cadNr", false, false, Arrays.asList(new Object[] { CAI
    public final static ConstraintAssociationImpl MANTELISOLATION_MINMAXLENGTHCONSTRAINT = new de.excellent.plexx.impl.model.Cor
    public final static AttributeProperty MANTELISOLATION = new AttributePropertyImpl(CLASS, "mantelisolation", false, false, Arrays.asLisi
    public final static RelationProperty VERWENDUNG_IN_SONDERLEITUNGSKOMPONENTEN = new RelationPropertyImpl(CLASS, "verwendun
    public final static AttributeProperty VERDRILLUNG_VORHANDEN = new AttributePropertyImpl(CLASS, "verdrillungVorhanden", true, true,
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERX_WERTMITEINHEITVOLLSTAENDIGCONSTRAINT = new de.excellent
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERX_WERTMITEINHEITGROESSERNULLCONSTRAINT = new de.excellent
    public final static ConstraintAssociationImpl AUSSENDURCHMESSERX_LEITUNGSGEOMETRIECONSTRAINT = new de.excellent.plexx.impl
    public final static AttributeProperty AUSSENDURCHMESSER_X = new AttributePropertyImpl(CLASS, "aussendurchmesserX", false, false
    public final static RelationProperty ALLE_BESTANDEILE_DEEP = new RelationPropertyImpl(CLASS, "alleBestandteileDeep", true, false,
    public final static ConstraintAssociationImpl BIEGEWECHSELANFORDERUNG_MINMAXVALUECONSTRAINT = new de.excellent.plexx.impl
    public final static AttributeProperty BIEGEWECHSELANFORDERUNG = new AttributePropertyImpl(CLASS, "biegewechselanforderung", fal
```

Praxisbeispiel 1: UI-Aufbau (1)

- Definition des (statischen) UIs anhand von Business-Attributen
- Keine UI-Elemente (Textfelder, etc.) im Quelltext
- Einheitliches Look&Feel
- Sehr effiziente UI Entwicklung



```
package de.excellent.connect.client.richtclient.view.bibliothekseintrag.sonderleitung.common.technischdaten;

import de.excellent.connect.business.benutzer.enum.Aktion;

public class TechnischeDatenPanel extends ConnectFormular {
    private ConnectFormular formular;

    public TechnischeDatenPanel(Sonderleitung sonderleitung, Aktion aktion, Transaction tx, boolean visualizeErrors) {
        super(tx, null, 3, visualizeErrors, true);

        addSeparator("Allgemein");
        add(SonderleitungMeta.GRUNDFARBE);
        add(SonderleitungMeta.ZWEITFARBE);
        add(SonderleitungMeta.DRITTFARBE);
        add(SonderleitungMeta.MIN_TEMPERATUR);
        add(SonderleitungMeta.MAX_TEMPERATUR);
        add(SonderleitungMeta.GEWICHT);
        add(2, new LeitungsquerschnittPropertyRepresentation(SonderleitungMeta.LEITUNGSGEOMETRIE,
            SonderleitungMeta.AUSSENDURCHMESSER_X,
            SonderleitungMeta.AUSSENDURCHMESSER_Y));
        nextRow();
        add(SonderleitungMeta.TOLERANZ);
        nextRow();
        add(SonderleitungMeta.MINIMALER_BIEGERADIUS);

        // ...

        // EditMode:
        if (aktion == Aktion.ANZEIGEN) {
            setEditMode(EditMode.READ_ONLY);
        } else if (aktion == Aktion.BEARBEITEN || aktion == Aktion.ANLEGEN) {
            setEditMode(EditMode.READ_WRITE);
        }

        // OwnerObject:
        setOwnerObject(sonderleitung);
    }

    public String getTitleString() {
```

Praxisbeispiel 1: UI Aufbau (2)

The screenshot shows a software application window titled "Connect" with a menu bar containing "Leitungen", "Kontaktierung", "Verwaltung", "Verwendungszweck", "E/E", "Zusatzeile", "Suche", "Sonstiges", "Import", and "Fenster". Below the menu bar are buttons for "Save", "Save all", "Print", and "Save & Close". The main window title is "B55 - Sonderleitung bearbeiten".

The form contains the following sections and fields:

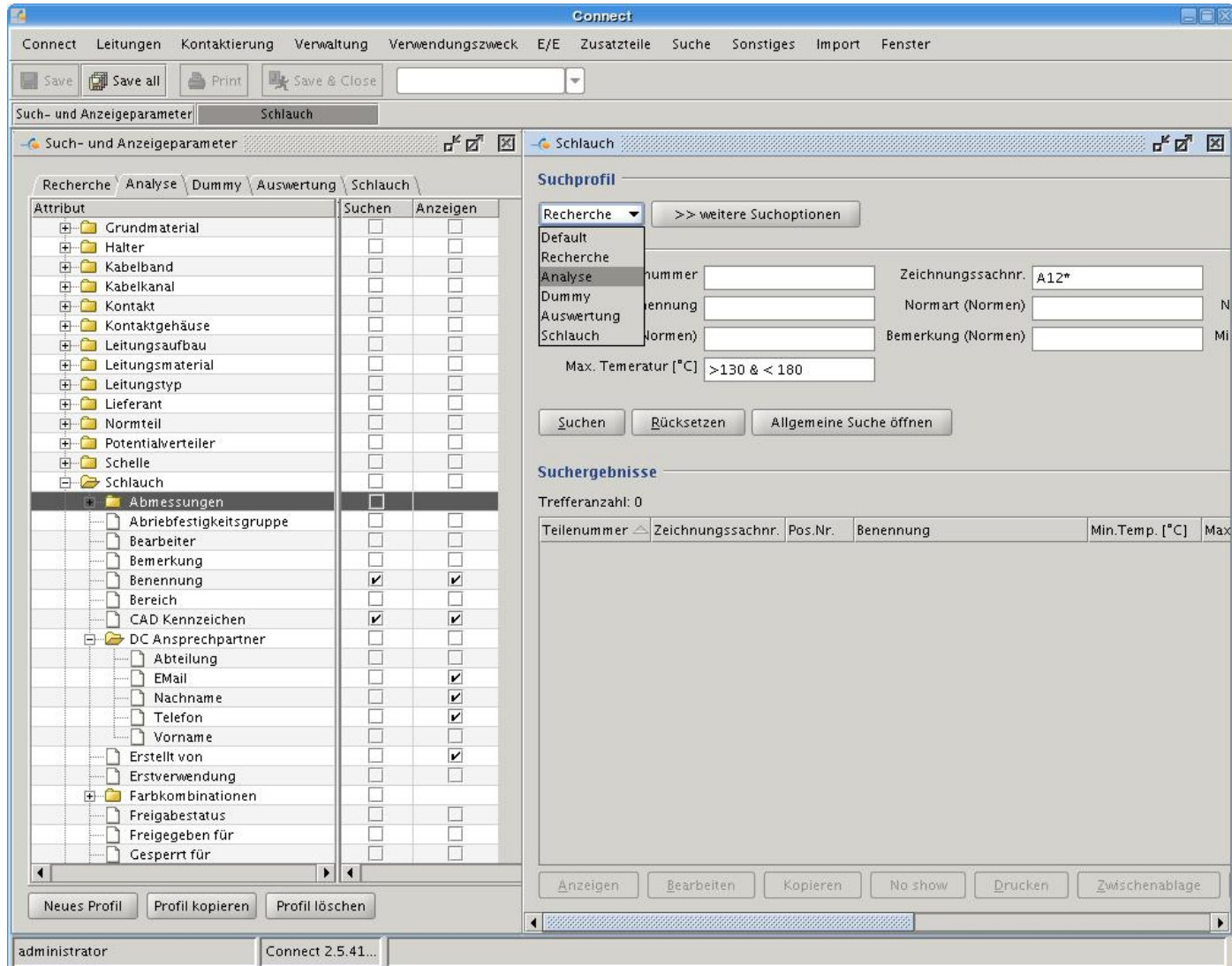
- Identification:** Cad Nr. (1) [B55], Teilenummer [123], Status [1], and a "Prüfen" button.
- Area:** Bereich * Pkw, KZ
- Descriptions:** Benennung [de] (1) [Einzel-1000-2000], Benennung [en] (1) [Single 1000/2000], and Änderungstext * []
- Technical Data:** A tabbed interface with "Allgemein" selected. Fields include Grundfarbe (1) [BK [schwarz]], Zweitfarbe [], Drittfarbe [], Min. Temperatur [°C] (2) [-40], Max. Temperatur [°C] (2) [50], Gewicht (2) [20] g/m, Außendurchmesser (1) [rund] [4.25] x [] mm, Toleranz Außendurchm. (2) [+/-] [0.1], Min. Biegeradius statisch [] mm, Biegeschwelanf. statisch bei 20°C [], Dämpfung [] dB / m, Wellenwiderstand [] Ohm, and Mantelisolierung (2) [PVC].
- Normen:** Dbl Norm [EN 50424-1000], Mbn Norm [], Iec Norm [], and Sonstige Normen [].
- Bestandteilinformationen:** Anzahl Adern [9], Anzahl Schirme [1], Anzahl Verdrillung [0], Beilauf vorhanden , Schirmung vorhanden , and Verdrillung vorhanden .
- Verwendung:** Entwicklungsfreigabe [] and Entwicklungsfreigabe ab [YYYY-MM-DD] []

The status bar at the bottom shows "administrator" and "Connect 2.5.41..."

Praxisbeispiel 2: Suchkonfiguration (1)

- Benutzer suchen mit verschiedenen Sichten dieselben Daten
 - Leitungssatzentwickler => technische Daten
 - Teamleiter => Statusinformationen
 - Kundendienst => Teilenummer
 - ...
- Suchprofile ermöglichen es zu definieren:
 - Nach welchen Attributen gesucht wird
 - Welche Attribute in der Treffermenge angezeigt werden
 - Vom Benutzer selbst konfigurierbar
- Technische Auswirkungen:
 - Keine fest definierten Suchdialoge
 - Keine vordefinierten Queries

Praxisbeispiel 2: Suchkonfiguration (2)



Vergleich und Bewertung (1)

	Generiert	Generisch
Flexibilität zur Laufzeit	--	++

- Generik unumgänglich für Flexibilität zur Laufzeit
 - Die Alternative wäre Code generieren, Compilieren, Dynamisch laden ... nur in Ausnahmefällen praktikabel
- Reflection über das Modell möglich
- Änderungen des Verhaltens der Applikation durch Benutzerinteraktion
- Theoretisch sogar Änderung des Modells zur Laufzeit möglich!
 - Durchaus nicht unüblich, um Datenstrukturen anzupassen (User Defined Attributes)

	Generiert	Generisch
Performance	+	-

- Generierter Code ist statischer Quelltext
- Generischer Code wird gesteuert durch `if` Anweisungen, Polymorphismus oder andere generische Mechanismen
- Laufzeitnachteil kann je nach Anwendungsgebiet nennenswert (embedded, oder bei großen Datenmengen in der Enterprise-Welt) bis vernachlässigbar sein
- Flexibilität für Optimierungen bei Spezialfällen auch und insbesondere beim generischen Ansatz vorsehen!

	Generiert	Generisch
Entwicklungszyklen	-	+

- Generatorlauf als zusätzlicher Schritt kostet Zeit
- Größere Menge an Quelltext erfordert längere Compilezyklen
- Kaum Praxisrelevant, da
 - Auch beim generischen Ansatz meist Teile generiert werden müssen
 - Unterschiede erst bei sehr großen Projekten auftreten
- In jedem Fall lohnenswert:
 - Applikation in Teilkomponenten aufspalten, sodass bei Änderungen nur Teile generiert werden müssen

	Generiert	Generisch
Applikations-Startup	+	-

- Generierter Code liegt fertig kompiliert und vollständig vor
- Generik erfordert Lesen und Auswerten des Modells
- Bei EMF basierten Ansätzen muss auch das Metamodell gelesen werden
- Bei großen Metamodellen (z.B. UML2) kann das u.U. mehrere Sekunden dauern
 - Der Bau eines Interpreters für direkte UML2 Modelle ist aber generell nicht empfehlenswert (Interpreter wird sehr kompliziert)
 - Stattdessen lieber eine Transformation von UML2 ein eigenes Metamodell, Interpreter dann darauf aufsetzen

	Generiert	Generisch
Debugging	+	-

- Generierter Code ist sehr explizit
- Breakpoints lassen sich leicht und direkt setzen
- Generischer Code arbeitet auf Meta-Ebene
- Umdenken des Entwicklers erforderlich
- Breakpoints fast nur mit Conditions sinnvoll einsetzbar

	Generiert	Generisch
Entwicklungseffizienz	++	++

- Beide Ansätze gleichwertig
- Wesentlich effizientere Entwicklung gegenüber manuellem Programmieren
- Weniger Fehler

	Generiert	Generisch
Verifizierbarkeit	+	-

- In manchen Umgebungen muss der Quellcode bzgl. verschiedener Eigenschaften verifiziert werden:
 - Nutzung bestimmter Sprachkonstrukte
 - Deadlockfreiheit
 - Timing-/QoS Eigenschaften
- Dies ist bei generiertem Code oft einfacher, weil die algorithmische Komplexität meistens geringer ist
 - Der Interpreter enthält quasi alle Alternativen,
 - Wohingegen der generierte Code nur die enthält, die wirklich vorkommen

	Generiert	Generisch
Codeumfang	+/-	+/-

- Hohe Metamodellabdeckung (große Modellen mit recht kleinem Metamodell)
 - Generisch: kleiner, da nur der Interpreter ins Gewicht fällt
 - Generiert: größer, da jedes Modellelement in Code resultiert
- Geringe Metamodellabdeckung:
 - Generisch: größer, da der Interpreter „für alle Fälle“ deployt werden muss
 - Generiert: bei kleinen Modellen kleiner, bei größeren Modellen groß
- Kommt also ganz drauf an 😊

	Generiert	Generisch
Flexibilität zur Laufzeit	--	++
Performance	+	-
Entwicklungszyklen	-	+
Applikations-Startup	+	-
Debugging	+	-
Entwicklungseffizienz	++	++
Verifizierbarkeit	+	-
Codeumfang	+/-	+/-

- Generik als alternativer Ansatz für MDSD
- In manchen Situationen sogar unumgänglich
- Vor- und Nachteile gegenüber rein generatorbasierter MDSD
- In der Praxis immer Mischformen von generiert und generisch
- Best-of-both-Worlds Ansatz adaptieren!

Aber, ist das dann noch MDSD??

Oder: was ist die **Essenz** von MDSD?

- Modelle, die bestimmte Viewpoints (Aspekte) einer Anwendung **klar, vollständig, formal und automatisch verarbeitbar** beschreiben
- Darstellung der Sachverhalte in einer **für die beteiligten Stakeholder passenden Syntax**
- **Validierung von Modellen** bzgl. bestimmter, vom Metamodell und Constraints definierter Eigenschaften überprüfen
- **Automatische Erstellung von Software** aus Modellen zwecks Konsistenz und Arbeitersparnis
 - Entweder per Generierung
 - Oder per Interpreter