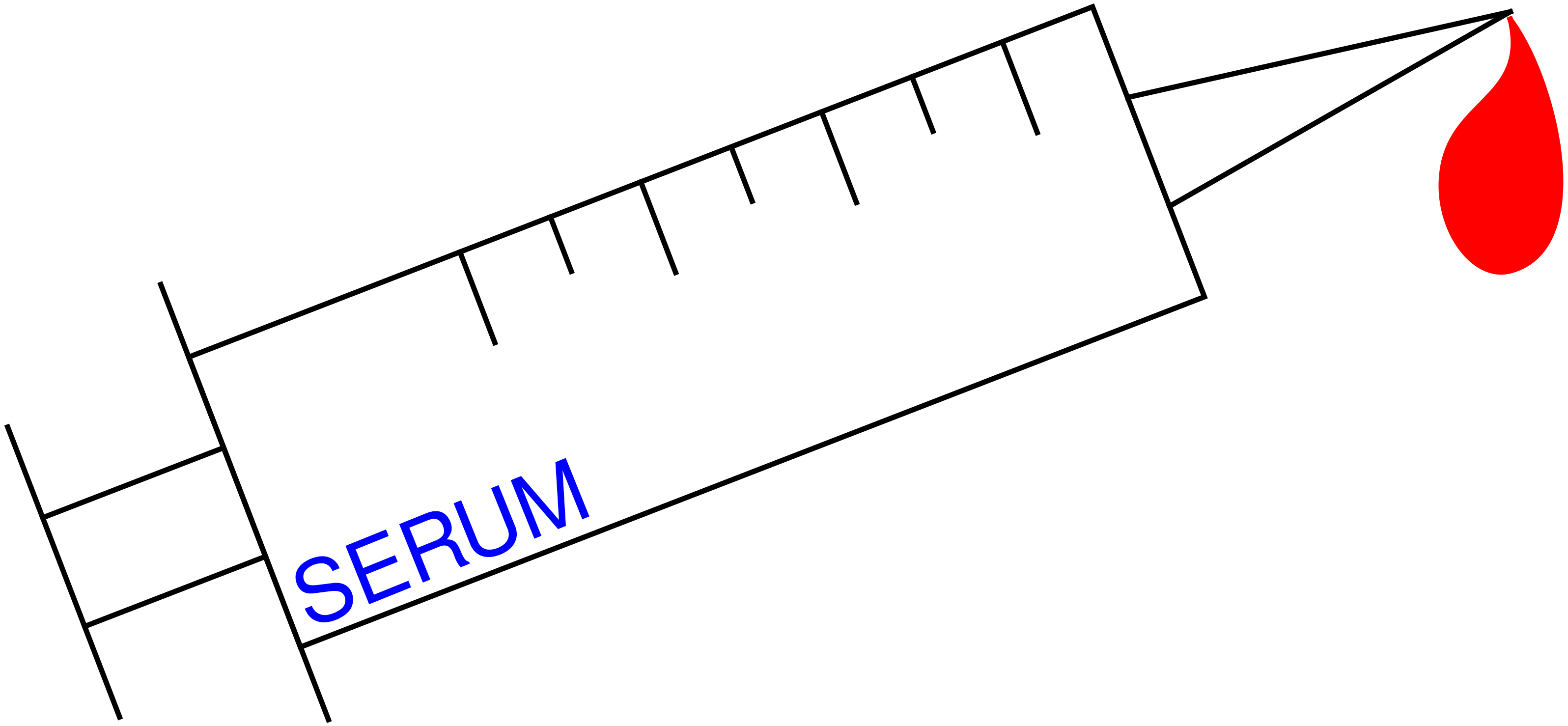
SERUM - Software Engineering Repository using UML



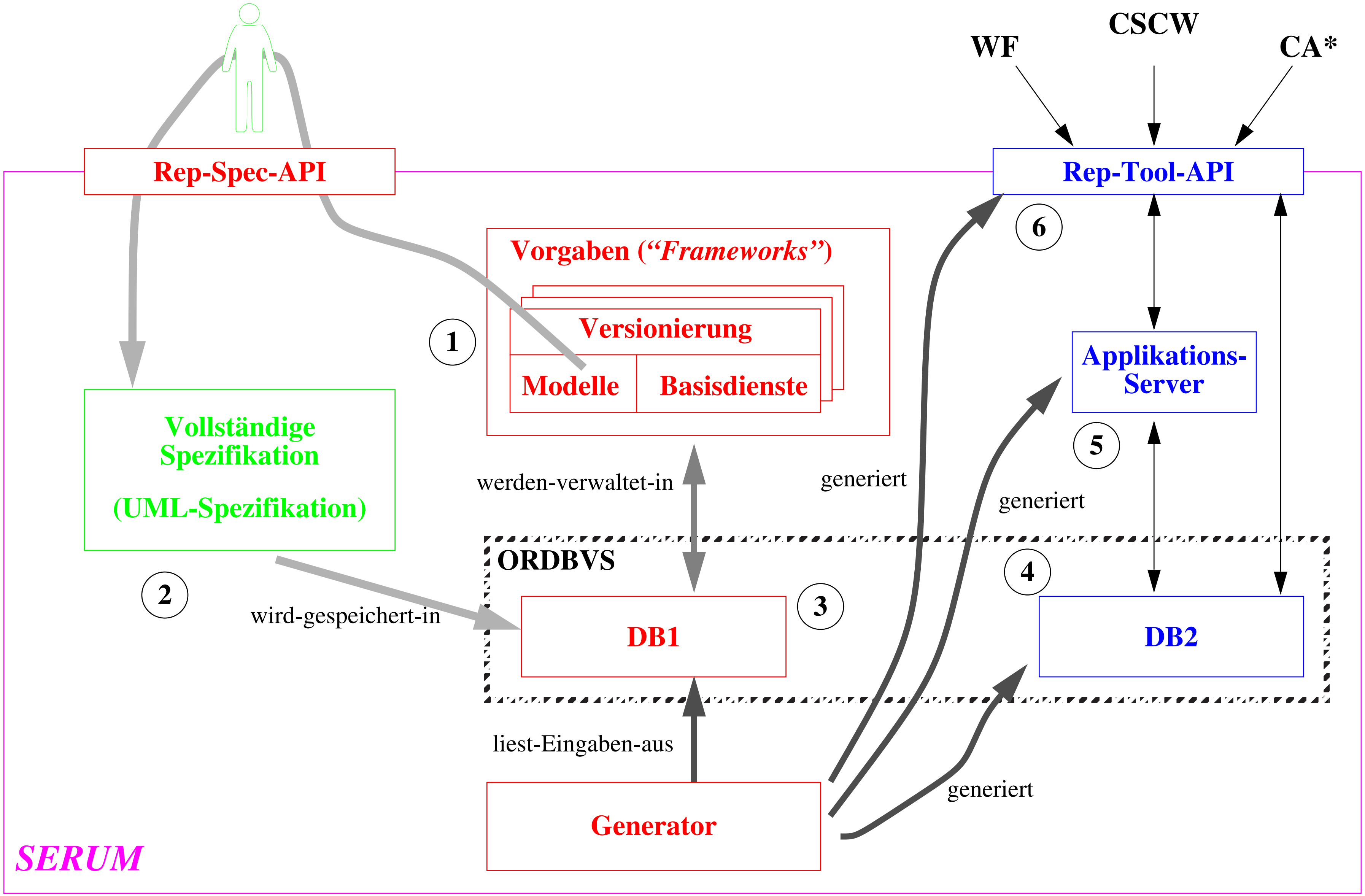
Framework-Ansatz

Beispiel: Software-Entwicklungsumgebung

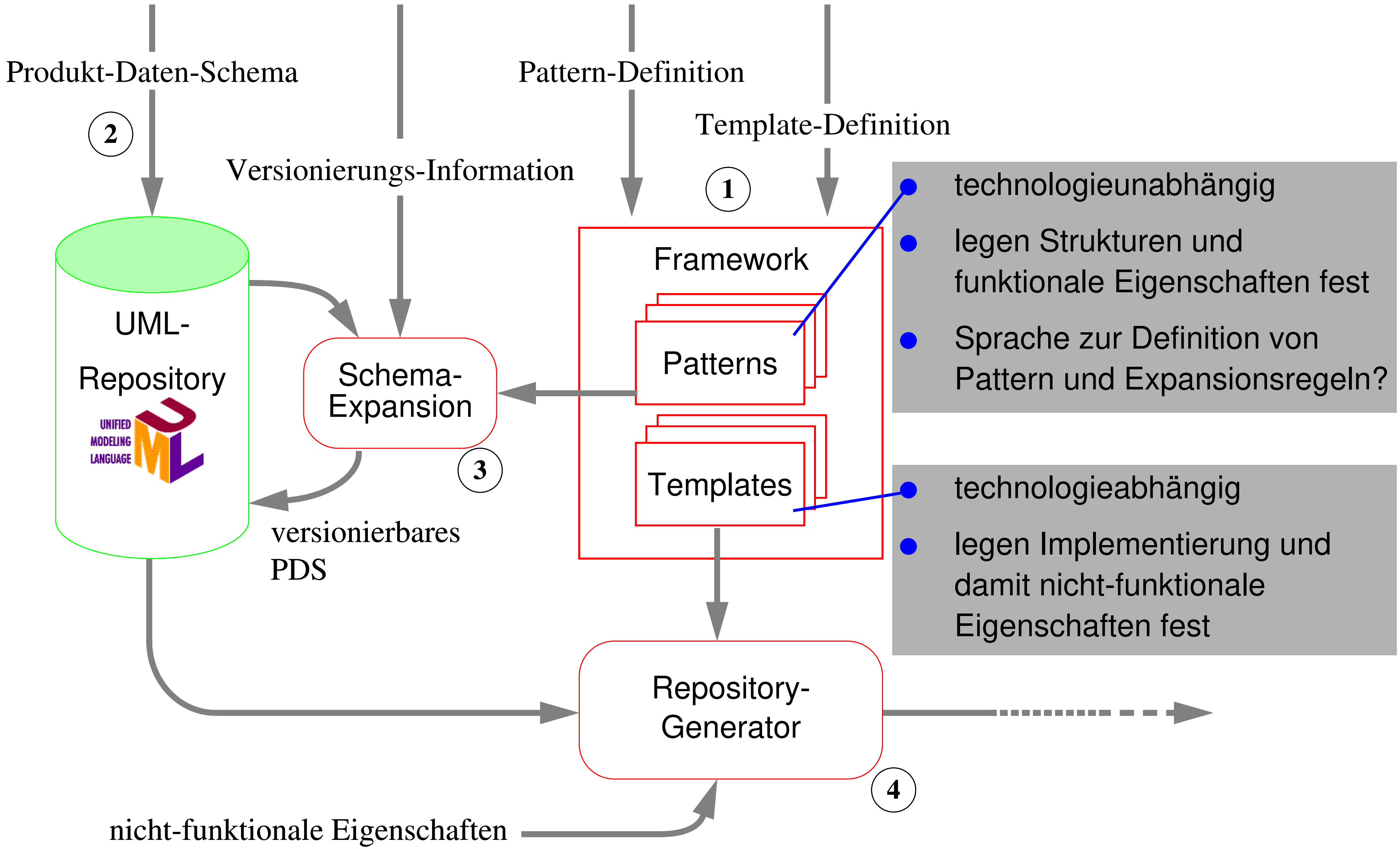
Signalmechanismus des Versionierungs-Framework

Realisierung des Signalmechanismus

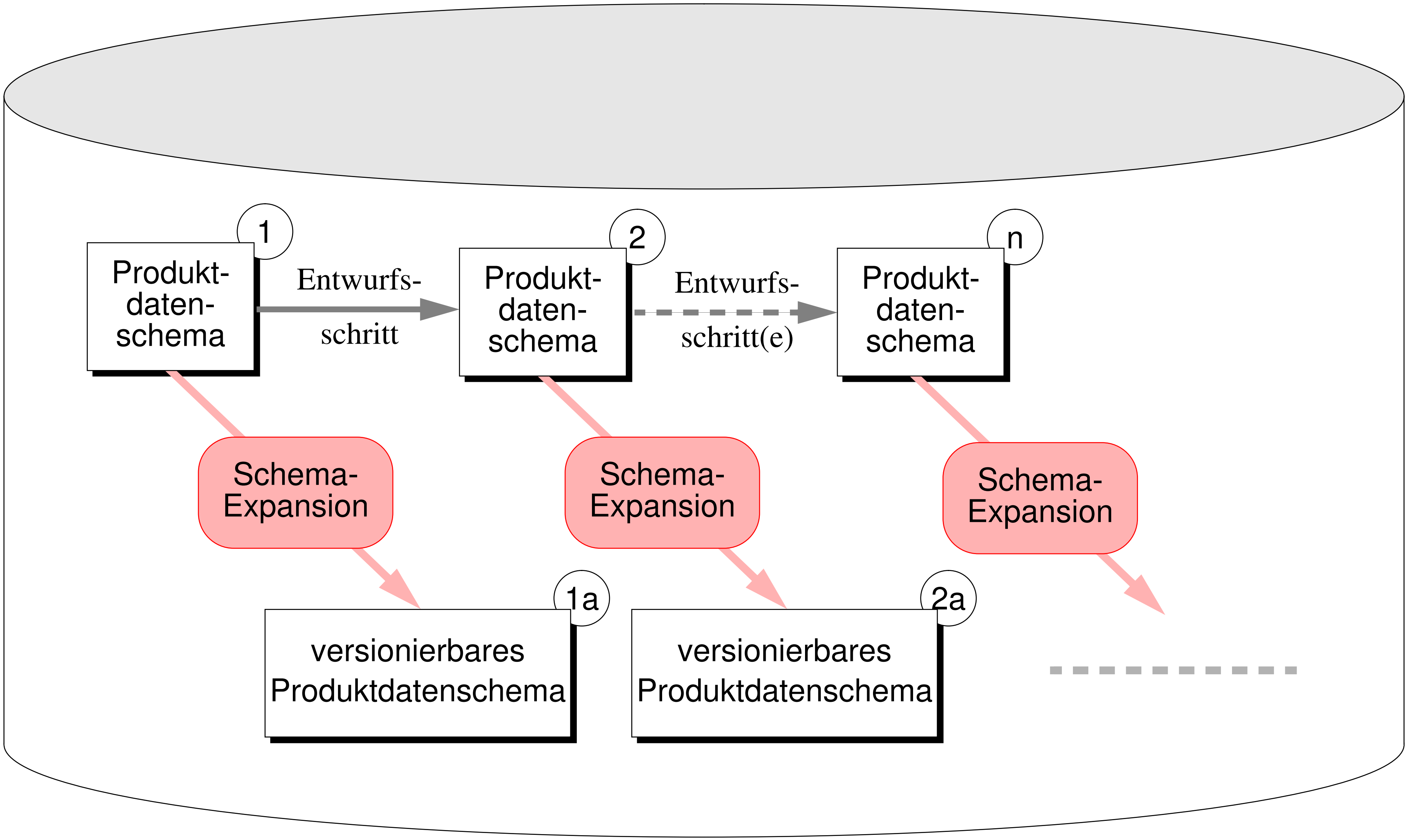
Framework-Ansatz für Repositories



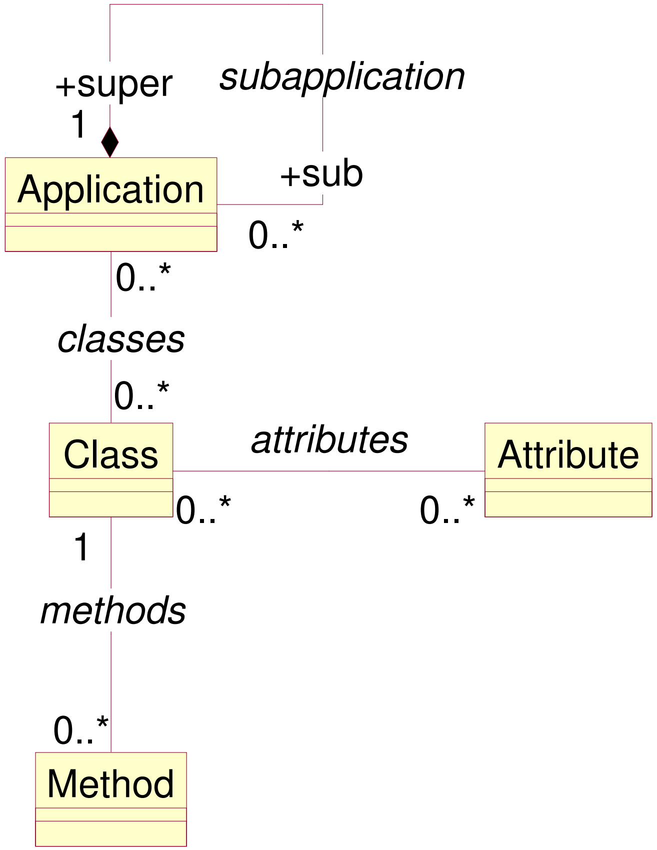
Spezifikation und Erzeugung von Repositories



Versionierung im UML-Repository



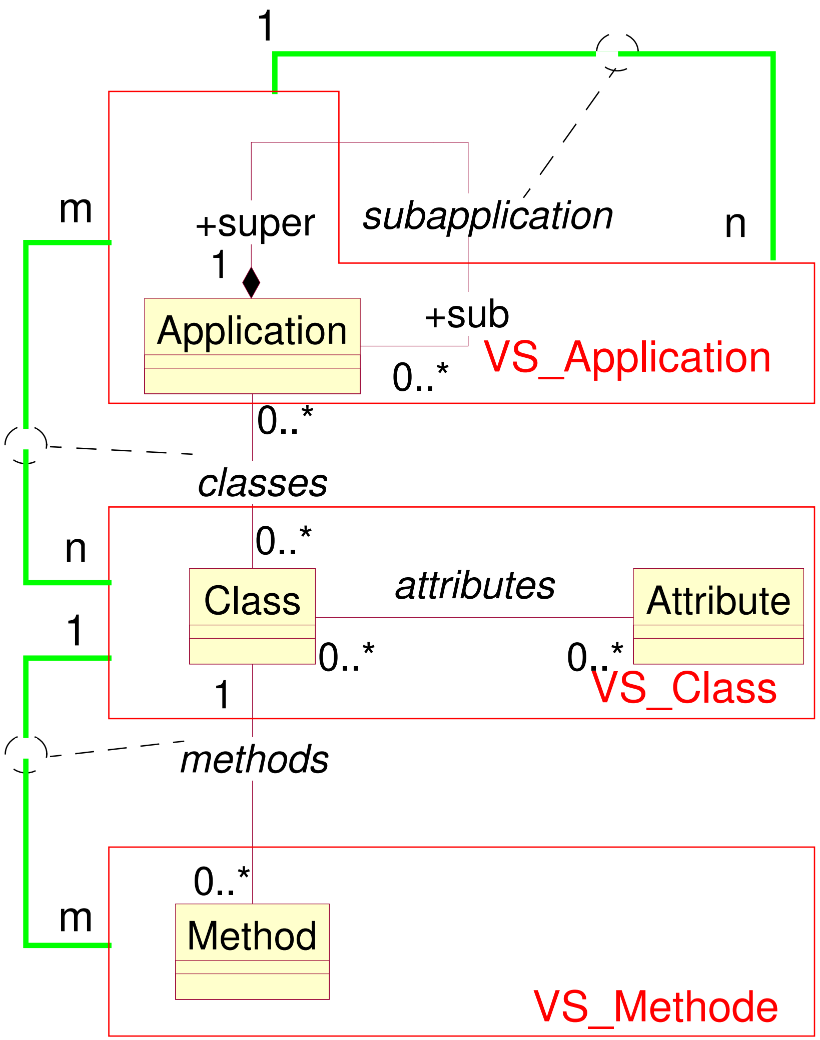
Produktdatenschema „Software-Entwicklungs-Umgebung“



SEU verwaltet die Klassen eines Softwareentwicklungsprojekts

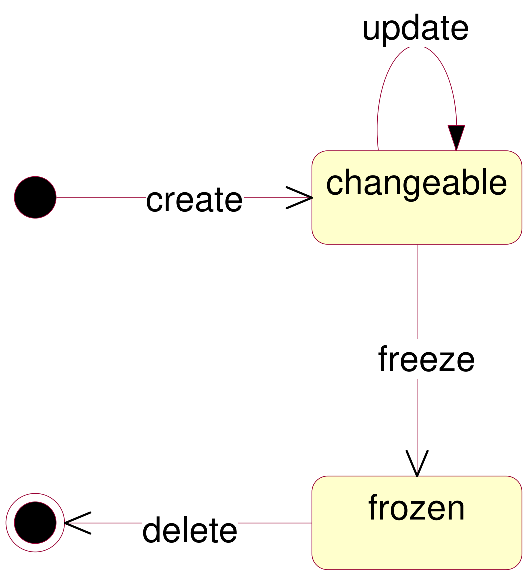
- *Application*
  - faßt Menge von Klassen zusammen
  - kann hierarchisch strukturiert werden (subapplication)
- *Class*
  - verwaltet die Klassen des Projekts
- *Attribute*
  - steht stellvertretend für die Strukturdefinition einer Klasse
- *Method*
  - steht stellvertretend für die Verhaltensdefinition einer Klasse

Einheiten der Versionierung



- *VS_Application*
  - Applikationen werden versioniert
  - die Beziehungen auf PDS-Ebene zwischen Super- und Subapplikation muß auf Versionsebene nachgezogen werden
- *VS_Class*
  - der Schemaausschnitt mit der Klassenstruktur wird geschlossen versioniert
  - die Beziehung zur Applikation auf PDS-Ebene muß auf der Versionsebene nachgezogen werden
- *VS_Method*
  - Verhalten ändert sich häufig und wird daher eigenständig versioniert
  - die Beziehung zur Klasse auf PDS-Ebene muß auf der Versionsebene nachgezogen werden

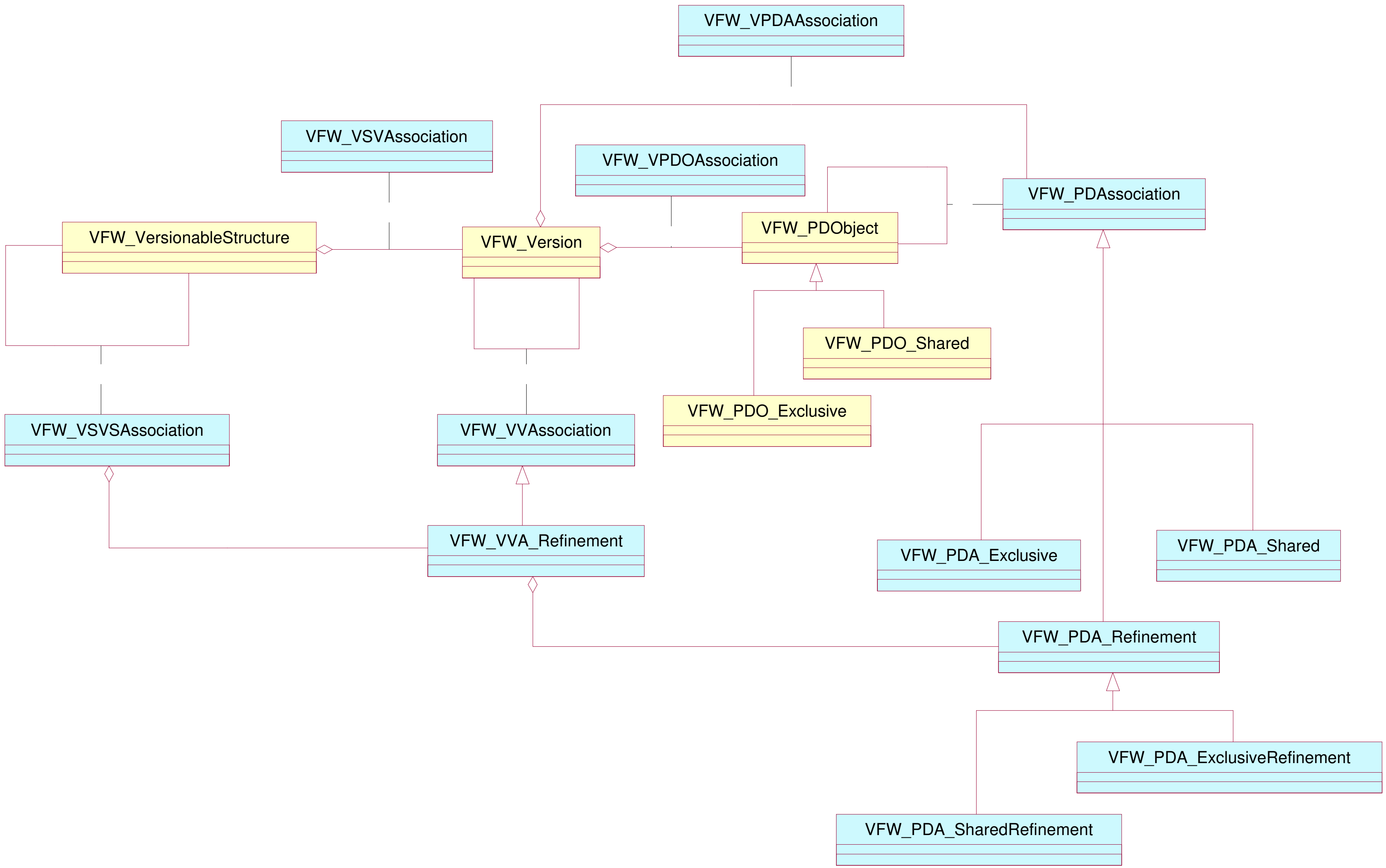
Versionierungssemantik



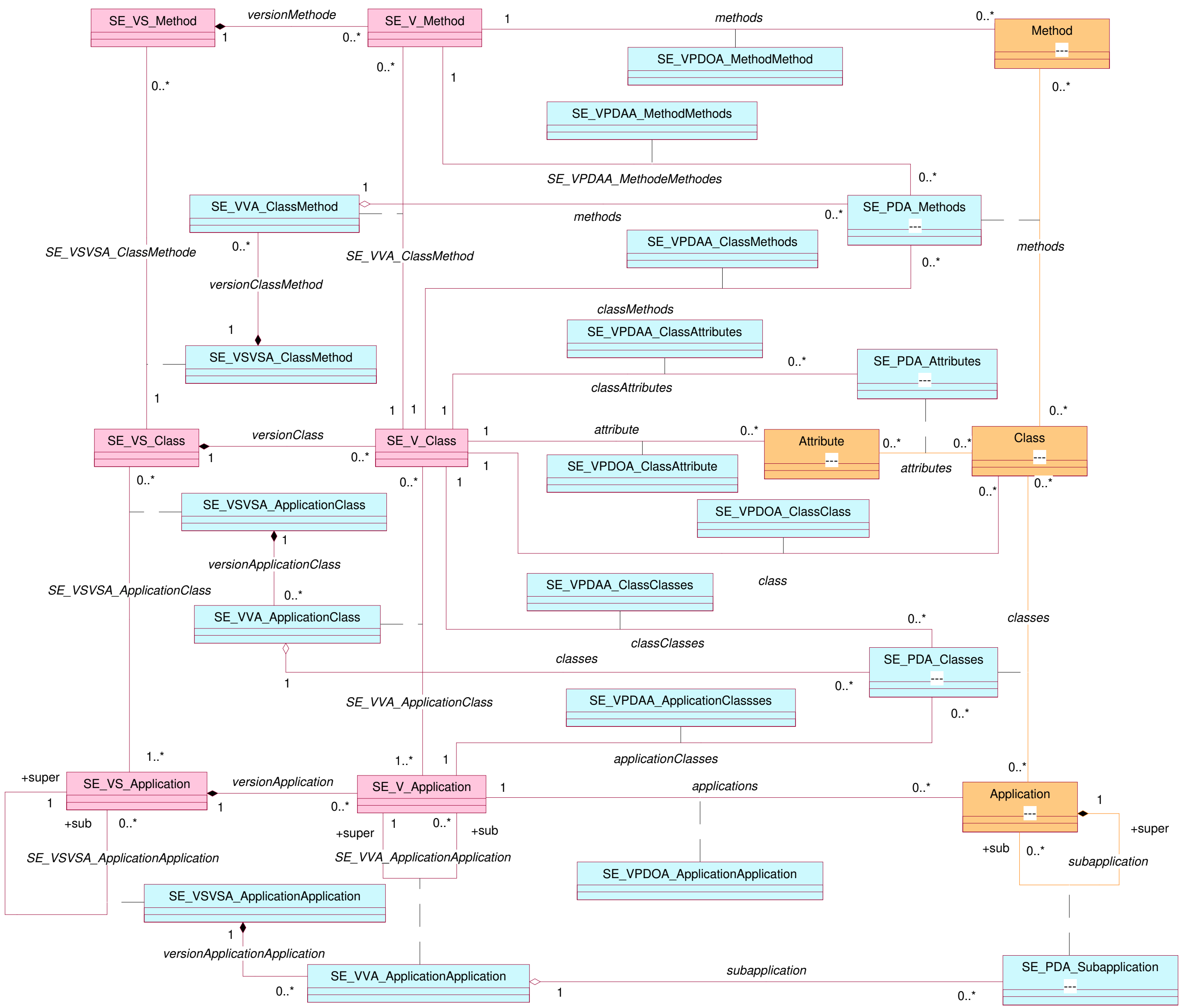
State-Diagramm zu VS_Application

- der Lebenslauf einer Version umfaßt zwei Zustände:
  - „*changeable*“
    - ☞ die Version kann noch verändert werden
  - „*frozen*“
    - ☞ die Version kann nicht mehr verändert werden
- ❑ *VS_Application*
  - der Wechsel nach „*frozen*“ ist nur möglich, wenn alle zugehörigen Subapplikationen und Klassen ebenfalls in diesem Zustand sind, ansonsten wird er unterbunden
    - ☞ isFrozen() wird zu „*true*“ ausgewertet
- ❑ *VS_Class*
  - der Wechsel nach „*frozen*“ ist jederzeit möglich
- ❑ *VS_Methode*
  - wechselt die zugehörige Version von *VS_Class* in den Zustand „*frozen*“ so wechseln implizit auch die Versionen von *VS_Method* in diesen Zustand
  - Versionen von *VS_Method* können nicht explizit in den Zustand „*frozen*“ versetzt werden

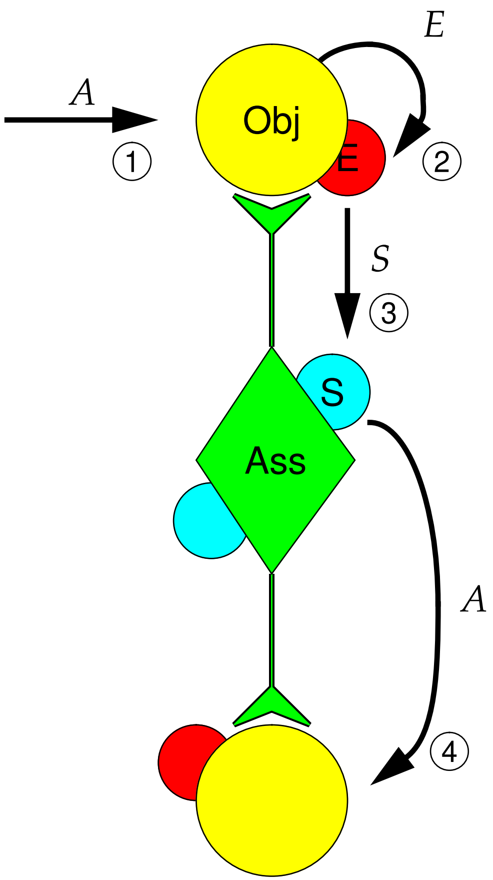
Versionierungs-Pattern



Versionierbares Produktdatenschema „SE-Repository“



Signalmechanismus



Action

- entsprechen den Operationen auf Datenobjekten
- lösen Ereignisse aus
  - jeweils eines vor und nach der Ausführung der Operation
  - Verfeinerungen ... ?

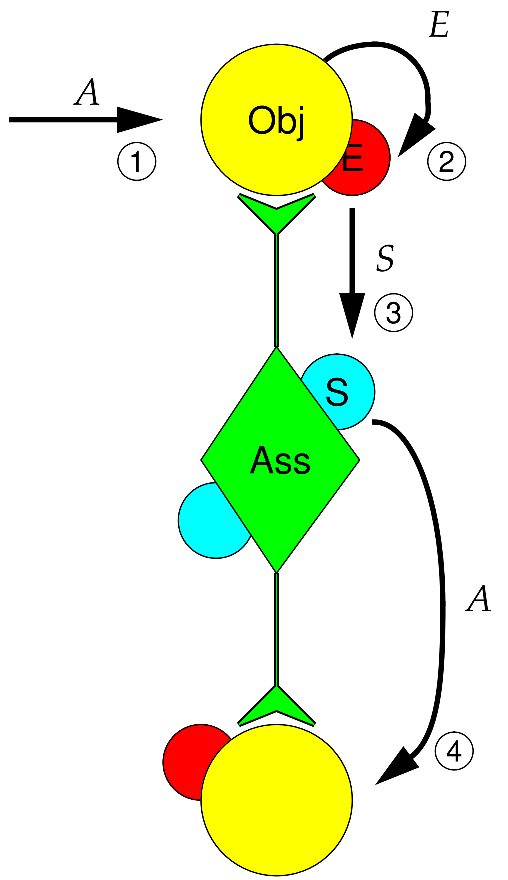
Event

- informieren den Event-Handler über das Eintreten eines Ereignisses auf dem zugehörigen Datenobjekt
- tragen Information über den Auslöser des Ereignisses
  - Welche Aktion hat das Ereignis ausgelöst?
  - Wie ist die Aktion parametrisiert?
  - Welcher Zustand hat das Ereignis ausgelöst?

Signal

- informieren den Signal-Handler über ein für ihn relevantes Ereignis
- tragen Information über das eingetretene Ereignis
  - Auslöser, Parameter, ...

Signalmechanismus



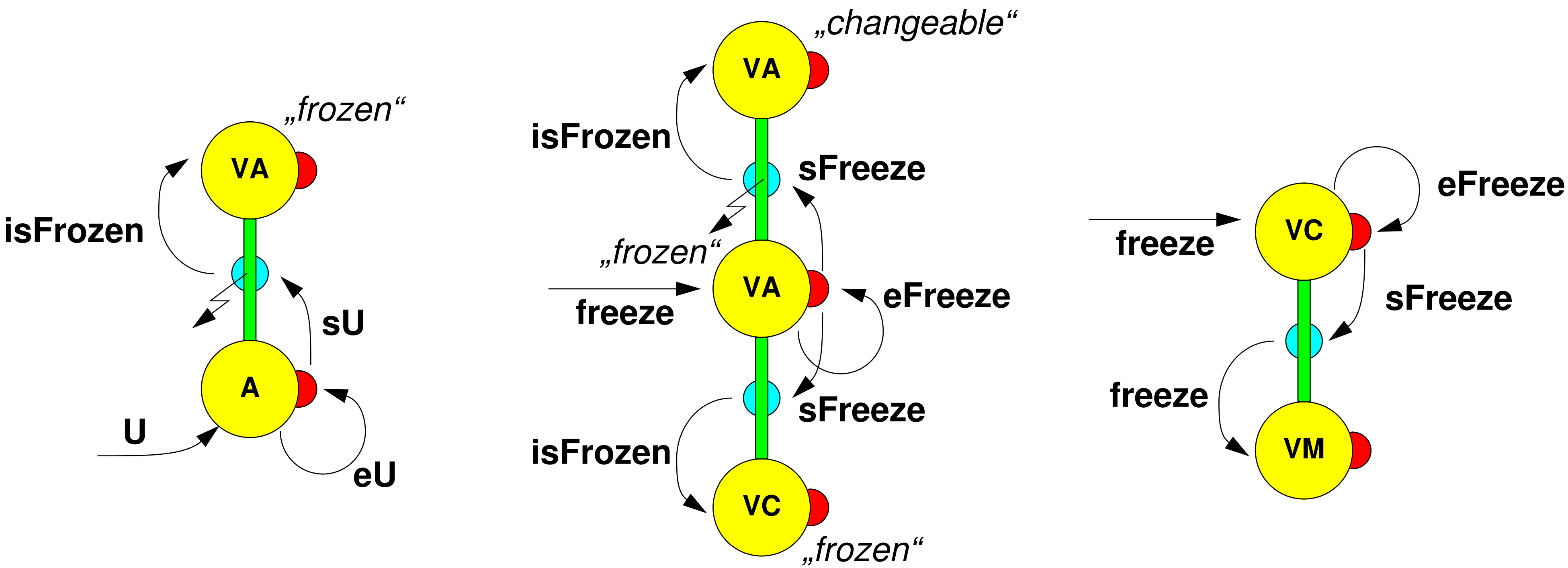
Event-Handler

- verwaltet die Signal-Handler (An- und Abmelden)
- kennt die für einen Signal-Handler relevanten Ereignisse
  - Was bedeutet „relevant“?
    - Ereignis
    - auslösende Aktion
    - Parameter der Aktion
    - Objektzustand
    - aktueller Kontext
    - ...
- signalisiert den Signal-Handler das Eintreten eines für sie relveanten Ereignisses

Signal-Handler

- reagiert auf Signale durch Aufruf von Aktionen
  - bei(m) gleichen / beiden / anderen Objekt(en)?
  - Wann wird welche Aktion ausgeführt?
  - Wovon hängt dies ab?

Kontrolle der Versionierungssemantik



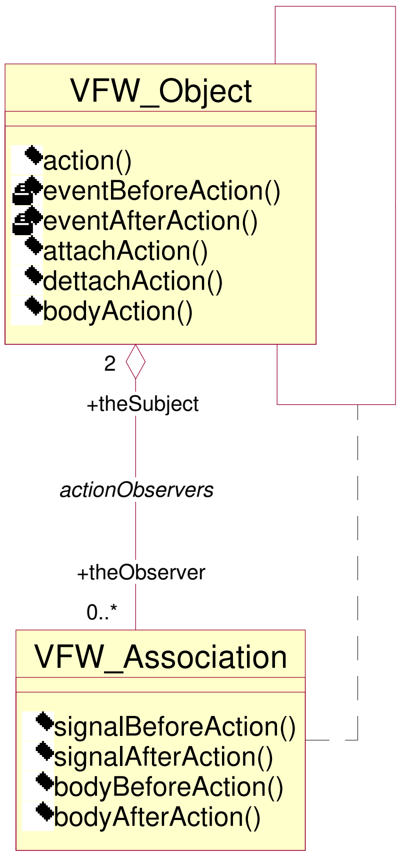
Version von VS_Application ist festgeschrieben

Version von VS_Application ist festgeschrieben

Versionen von VS_Method werden implizit festgeschrieben

☞ Versionierungssemantik wird durch die Beziehungen zwischen den Objekten festgelegt

Anwendung des Design-Pattern „Observer“



VFW_Object

- realisiert „Subject“ des DP „Observer“
- realisiert den Event-Handler

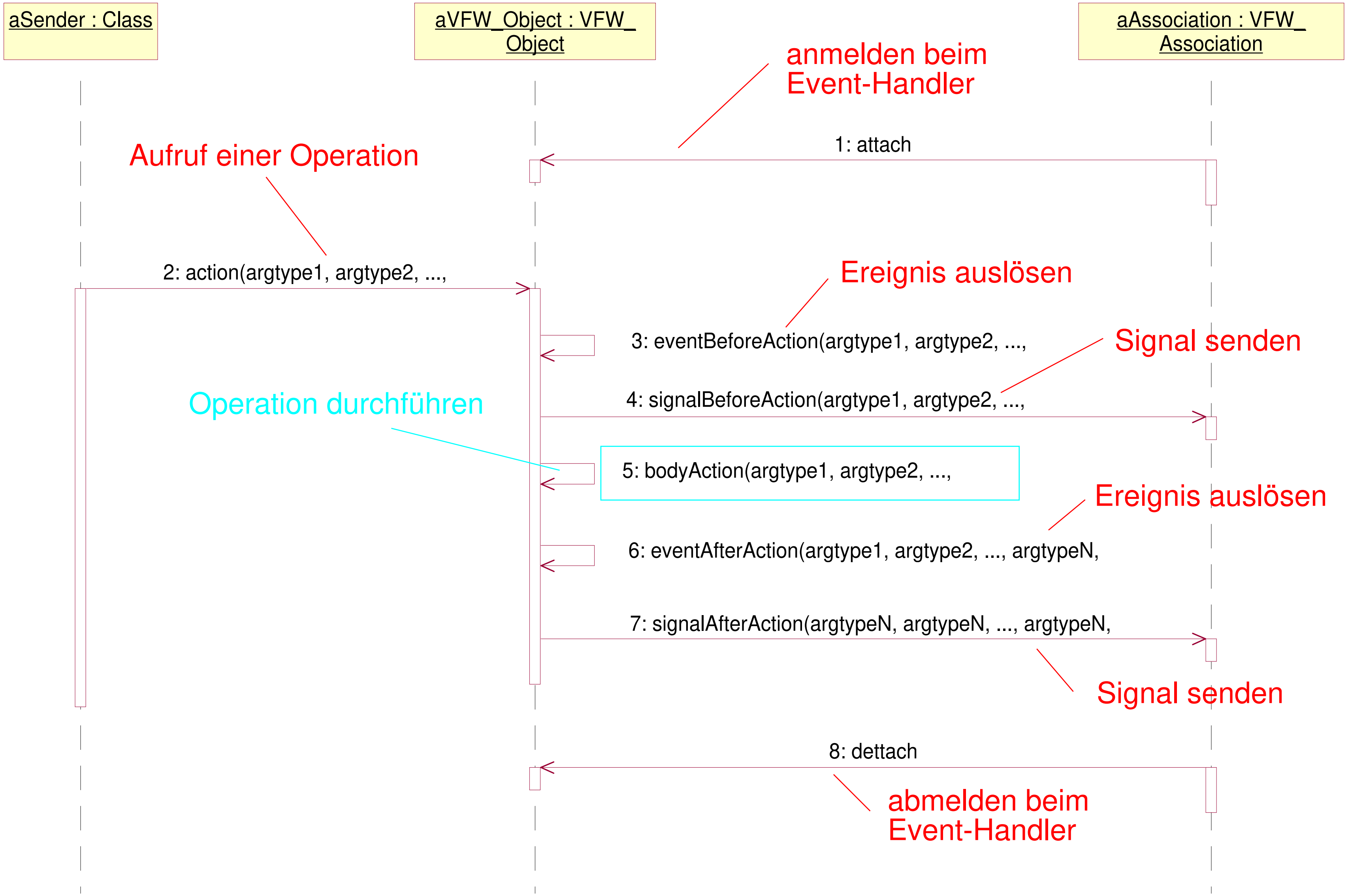
VFW_Association

- realisiert „Observer“ des DP „Observer“
- reagiert auf Signale aus beiden Richtungen
  - Verfeinerung in Subklassen möglich (unidirektional, ...)!
- realisiert den Signal-Handler
- Versionierungssemantik wird vollständig von den Beziehungen realisiert

Entwurfsrichtlinien

- alle Datenobjekte sind Instanzen von VFW_Object
- alle Beziehungsobjekte sind Instanzen von VFW_Association

Protokoll des Signalmechanismus



Ausblick

- Verfeinerung des Signalmechanismus
  - Wann ist ein Framework „sicher“?
  - Kontrolle der Ereignis-/Signal-Flut
  - Ist die Modellierung von „State-Maschines“ in UML zur Spezifikation der Signalverarbeitung verwendbar?
- Einbettung in SQL3, bzw. Realisierung als Informix DataBlade
- Realisierung eines Basis-Repository (Diplomarbeiten)
  - Ausgangpunkt: UML-Core-Konzepte
  - Entwicklung einer geeigneten Versionierungsstrategie für das UML-Core Schema
  - Vorschlag für Pattern und Templates
- Realisierung der Werkzeuge, um das Basis-Repository mit den Mitteln des Basisrepository zu erstellen
- Durchführung beider Strategien im Rahmen eines SFB-Experiments (Praktikum)
  - Welche Gewinne bei Aufwand, Fehlerhäufigkeit, etc kann durch den Ansatz erreicht werden
- Weitergehende Konkretisierung und Verfeinerung des Ansatzes
- Anwendung des Ansatzes auf andere Bereiche (Dienst zur Ablaufsteuerung, ...)