



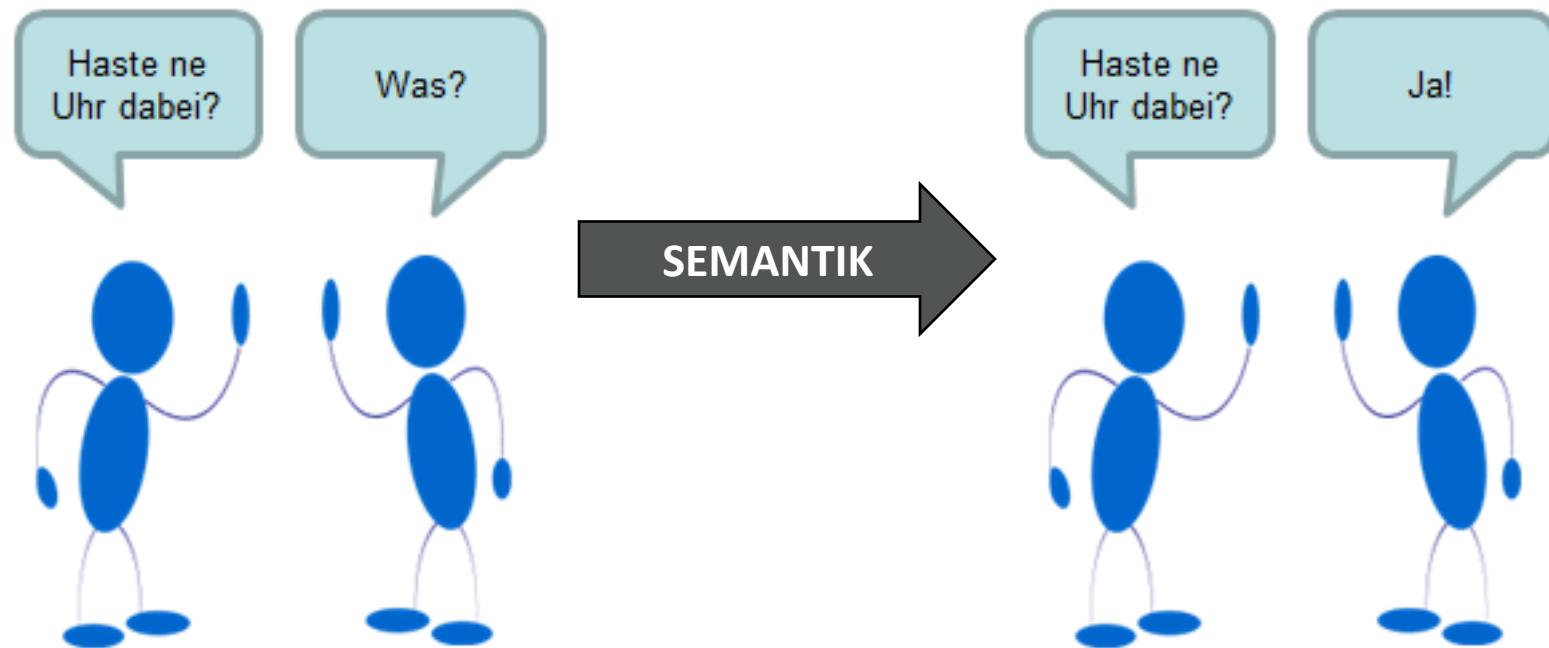
Software für Industrie 4.0 (Vorlesung & Übung)

Semantische Beschreibungen



Was ist Semantik?

- **Semantik = Bedeutungslehre**, nennt man die **Theorie von der Bedeutung von Zeichen**



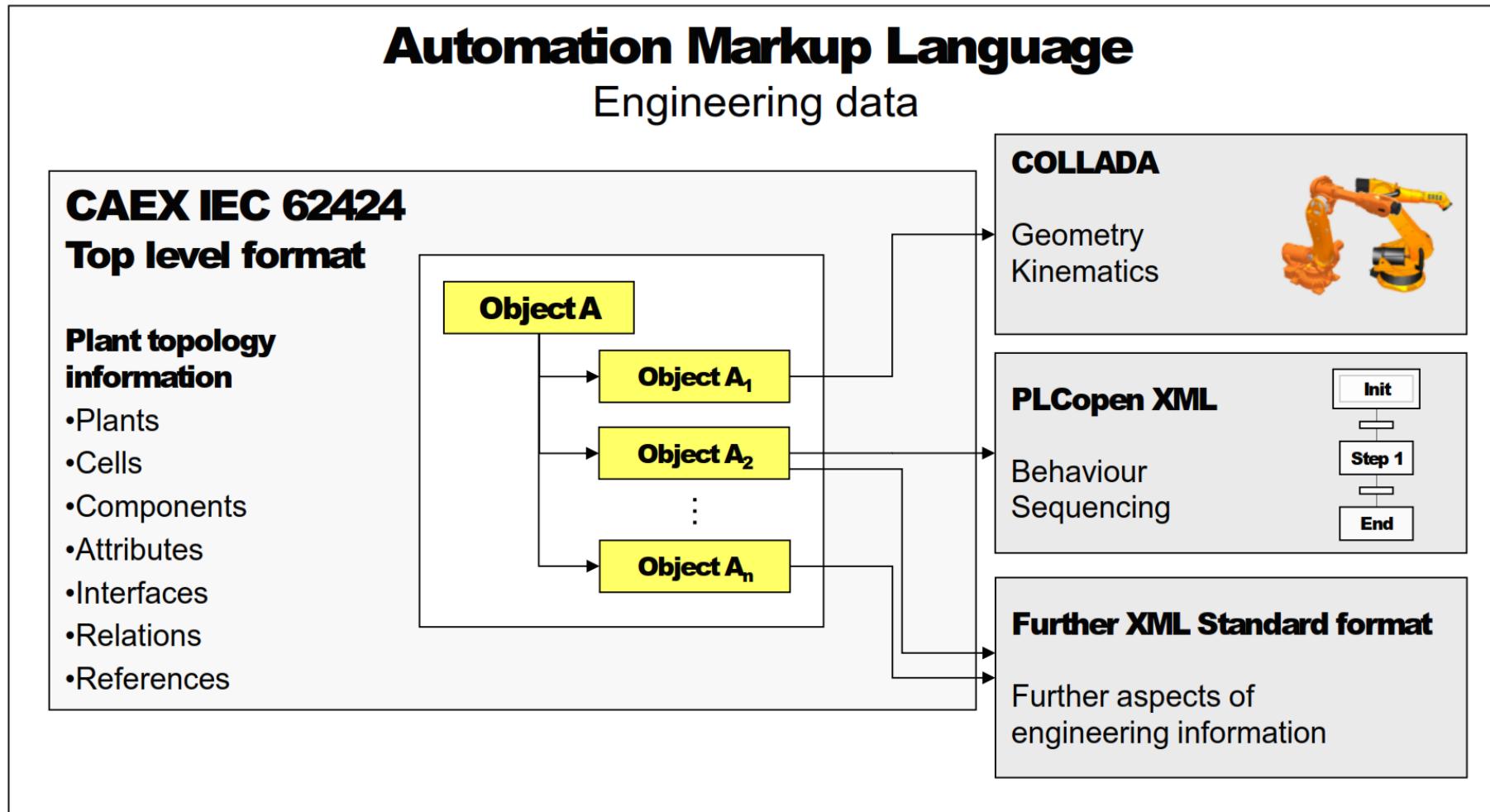
Überblick

AUTOMATION ML

- Standard (IEC 62714) für die semantische Beschreibung von Automatisierungsgeräten
- Entwickelt von einem Konsortium von Industrieunternehmen (ab 2006: Daimler, ABB, KUKA, Rockwell Automation, Siemens) und Forschungseinrichtungen (Fraunhofer IOSB, Universität Magdeburg)
- Seit 2009 eigenständiger eingetragener Verein (AutomationML e.V.), welcher mit der Weiterentwicklung und Verbreitung von AutomationML beauftragt wurde
- Mit der auf XML basierenden „**Automation Markup Language**“ können topologische, geometrische, kinematische und logische (programmatische) Aspekte einer Industrieanlage beschrieben werden
- Auch können beliebige weitere XML-Formate verknüpft werden um neue Informationstypen in die bestehende Datenbasis zu integrieren

<AutomationML/>

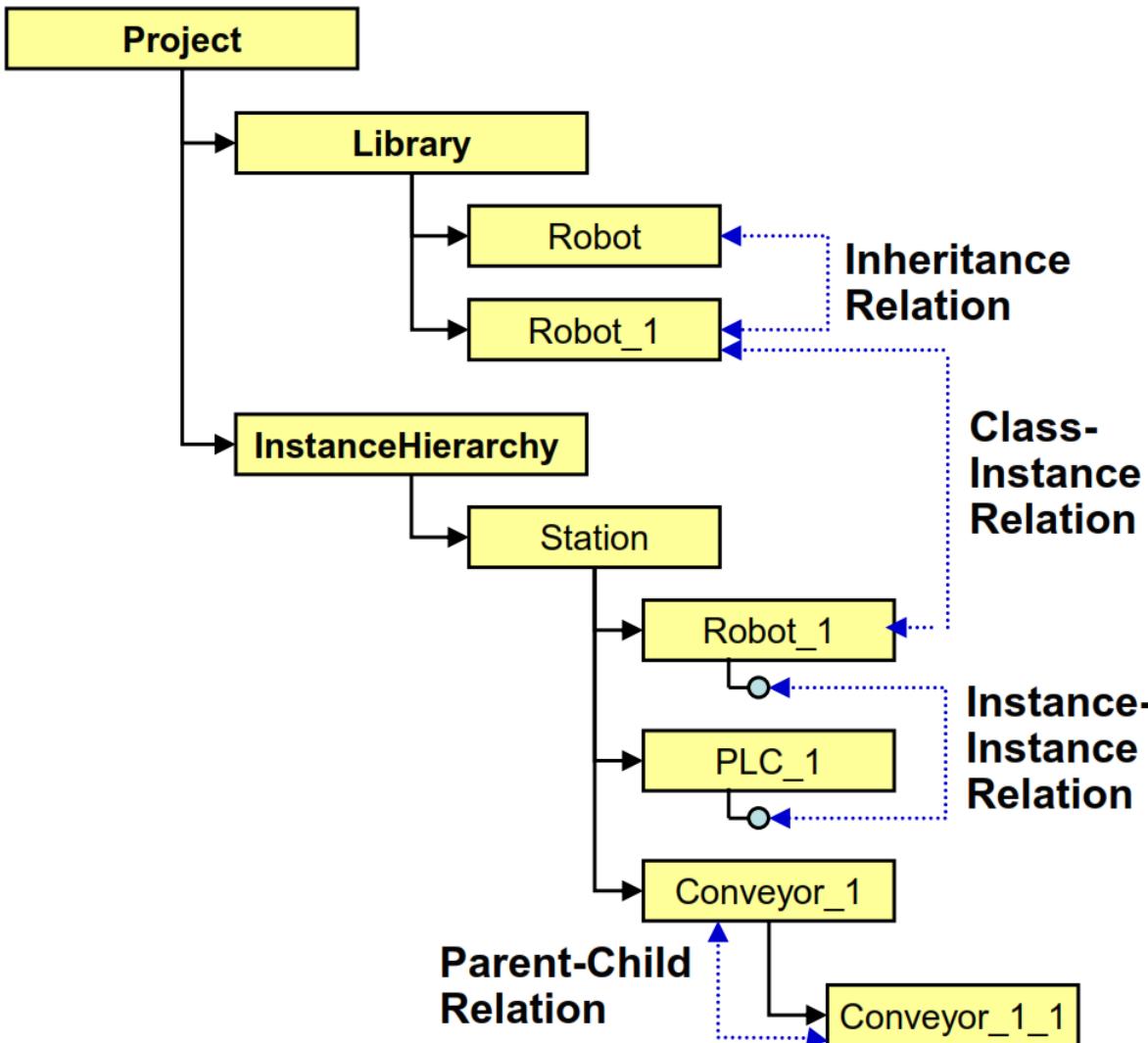
Quelle: iosb.fraunhofer.de



Quelle: [AML 2014-1]

- Für die Beschreibung der Topologie der Industrieanlage wird das Datenformat **CAEX** (Computer Aided Engineering Exchange), ein ebenfalls auf XML basierender Standard (IEC 62424), verwendet
- In einem Forschungsprojekt ab 2002 von der RWTH Aachen in Kooperation mit ABB entwickelt
- Ziel dieses Formats ist es, ein einheitliches Datenformat für den Austausch von und Kollaboration an zentralen, hierarchischen Strukturdaten für die Produktionslinie für unterschiedliche Softwaretools und alle am Planungsprozess beteiligten Personen zu ermöglichen

AutomationML – Topologie – CAEX-Beispiel



Quelle: [AML 2009]

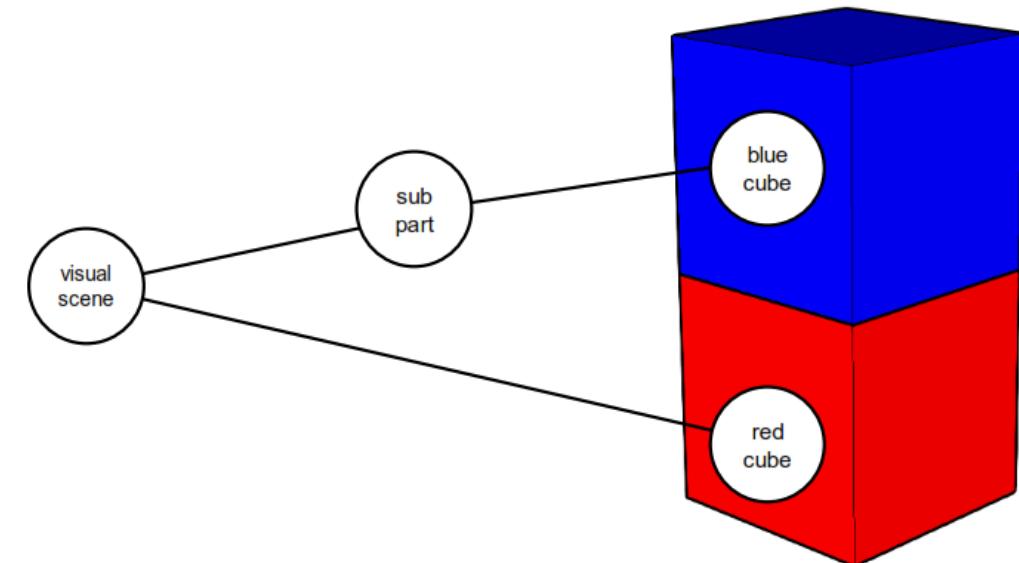
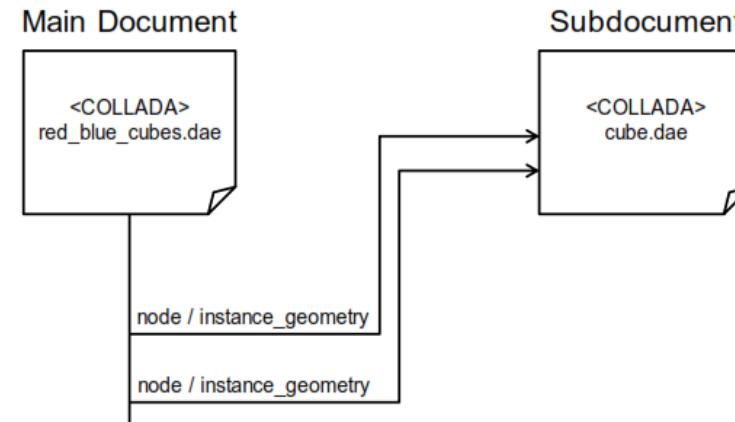
Hauptkonzepte:

- **Library**
enthält Vorlagen für konkrete Objekte
- **InstanceHierarchy**
beschreibt die tatsächliche Topologie und Attribute von / Beziehungen zwischen konkreten Objekten
- **Roles**
abstrakte Beschreibungen von Objekten, z.B. „Roboter“, „Fließband“
- **Interfaces & Relations**
semantische Verbindungen zwischen Objekten und/oder deren geometrischen, kinematischen oder logischen Informationen

- In AutomationML werden Daten zum geometrischen Aufbau in **COLLADA** (*COLLAborative Design Activity*), einem wiederum auf XML basierenden Format, spezifiziert
- Dies ist ein Datenformat zur Definition und zum Austausch von 3D-Modellen zwischen verschiedenen Programmen
- Ursprünglich von **Sony Computer Entertainment** entwickelt, aber inzwischen ein OpenSource-Format, welches von der **Khronos Group** weiterentwickelt und verwaltet wird
- COLLADA unterstützt außerdem die Definition von kinematischen Zusammenhängen und physikalischen Eigenschaften von Objekten

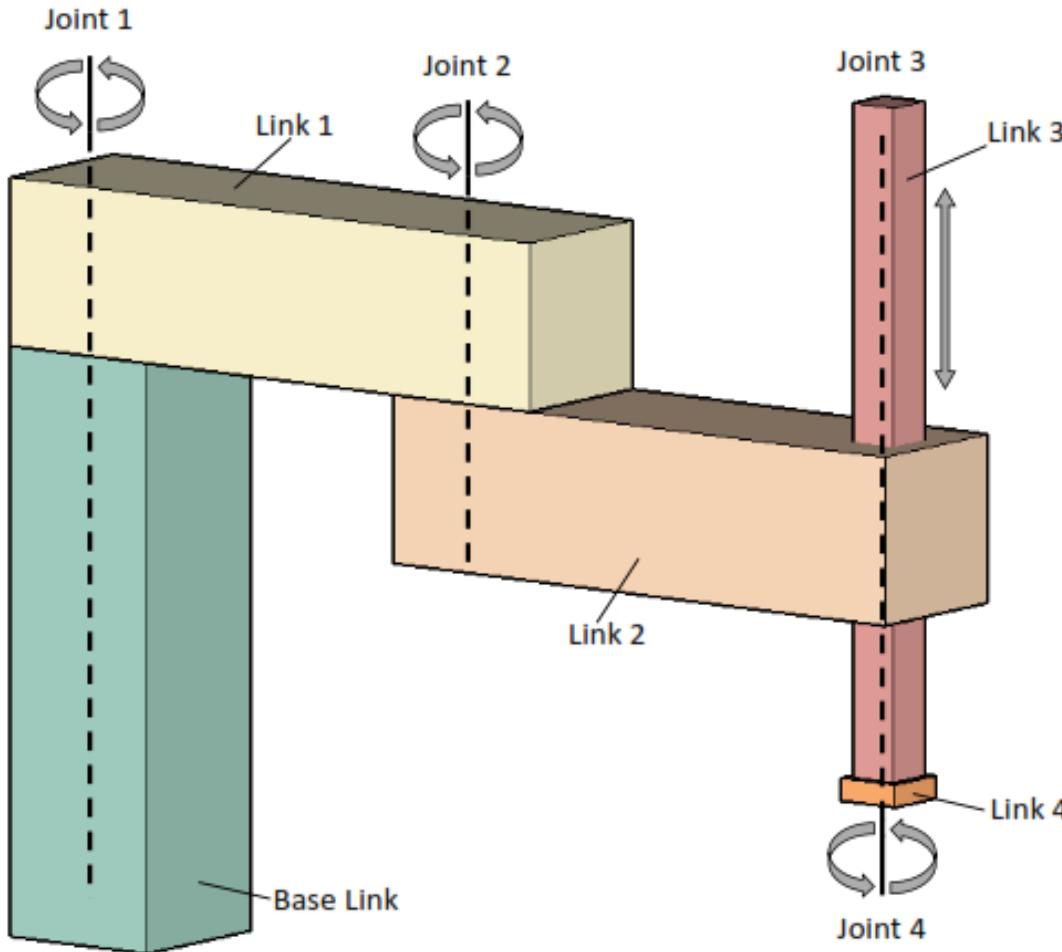
AutomationML – COLLADA – Aufbau

- Auch COLLADA verwendet wiederum einen hierarchischen und objektorientierten Ansatz zur Erstellung von 3D-Szenen
- So kann z.B. eine Szene angelegt werden, welche zwei Würfel enthält, hierzu beschreibt man in einer COLLADA-Datei den Aufbau eines Würfels und importiert diese in der Hauptszene zwei Mal
- Dabei können Attribute, wie die Farbe des Würfels, von der Hauptszene beliebig angepasst werden



Quelle: [AML 2015-2]

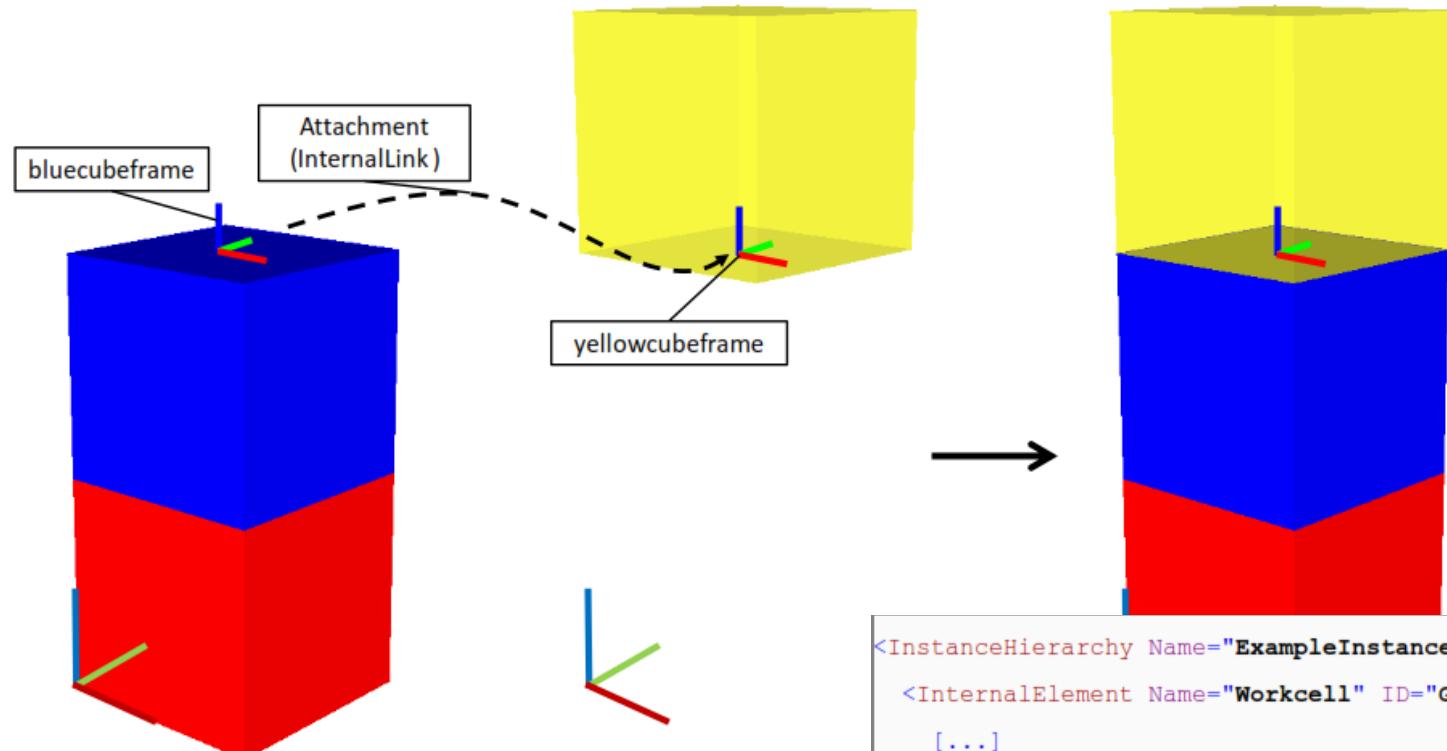
AutomationML – COLLADA – Kinematik



Quelle: [AML 2015-2]

- Für die Kinematik eines Roboters lassen sich in COLLADA alle relevanten Parameter festhalten
 - Max. Geschwindigkeit der Achse
 - Max. Beschleunigung
 - Max. Verzögerung
 - Max. Ruck
 - Achsgrenzen
- Somit sind statische geometrische Informationen über den Aufbau des Roboters und dynamische über die Freiheitsgrade des Roboters einander logisch zugeordnet und gemeinsam zentral abgelegt

AutomationML – COLLADA vs CAEX – Beziehungen



- Semantische Verknüpfungen zwischen geometrischen Objekten werden wiederum in CAEX angegeben, sofern diese nicht unveränderlich sind, wie z.B. im Falle der Achsen eines Roboters

```
<InstanceHierarchy Name="ExampleInstanceHierarchy">
  <InternalElement Name="Workcell" ID="GUID0">
    [...]
    <InternalLink Name="AttachmentBlueRedMovesYellowCube"
      RefPartnerSideA="GUID3:Attachment"
      RefPartnerSideB="GUID8:Attachment" />
  </InternalElement>
</InstanceHierarchy>
```

Quelle: [AML 2015-2]

- Das tatsächliche Verhalten der einzelnen Objekte und der gesamten Produktionsanlage im Allgemeinen wird in PLCopen XML (Standard nach IEC 61131-3) definiert
- Alle standardisierten Programmiersprachen für die SPS-Programmierung können in das PLCopen XML-Format übersetzt werden
- Damit ist es idealerweise auch möglich diese allgemein Beschreibung von logischen Abläufen und Programmen in die konkrete Variante der verwendeten SPS zurückzuübersetzen
- Damit kann dann z.B. die Implementierung einer Garagentorsteuerung in XML definiert und vor der konkreten Auslieferung in ein B&R-Programm übersetzt werden
- Programmiert werden kann dabei in Varianten der bekannten Sprachen, gespeichert wird das Programm aber in XML

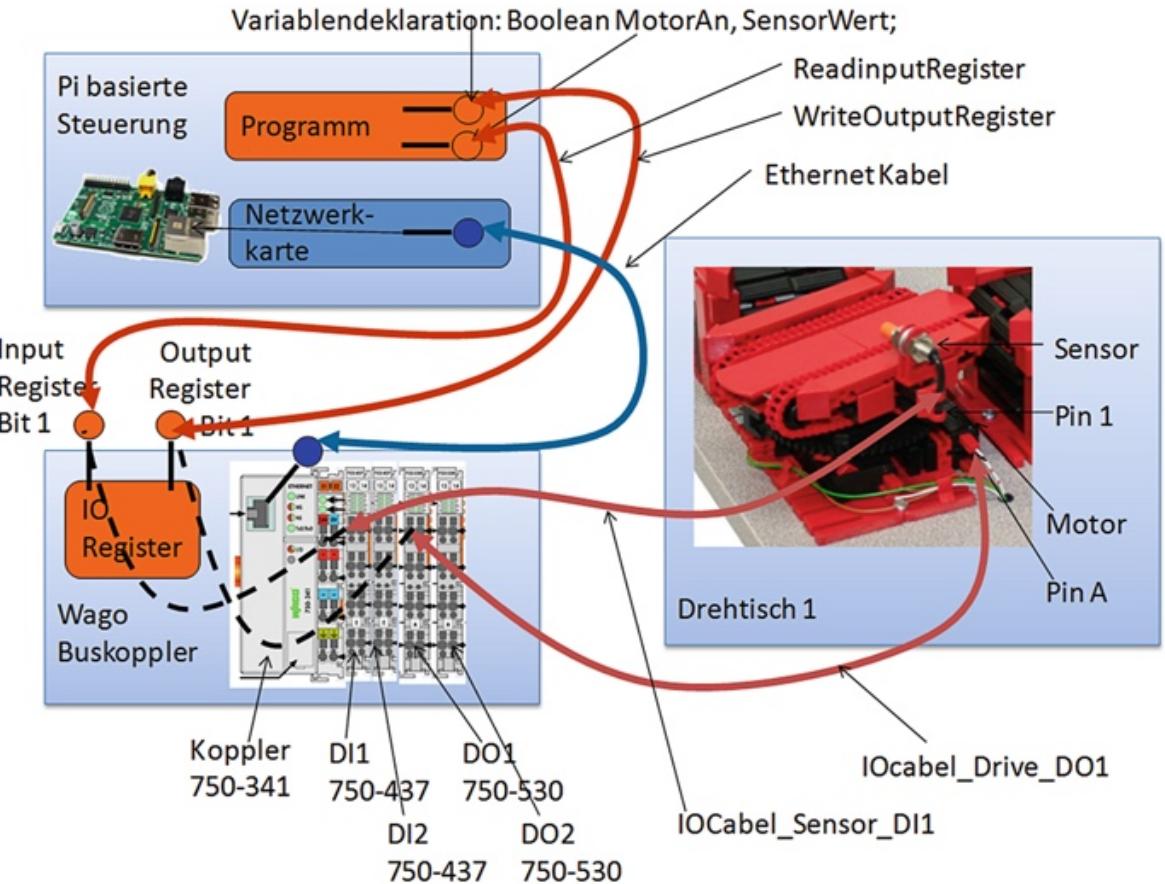
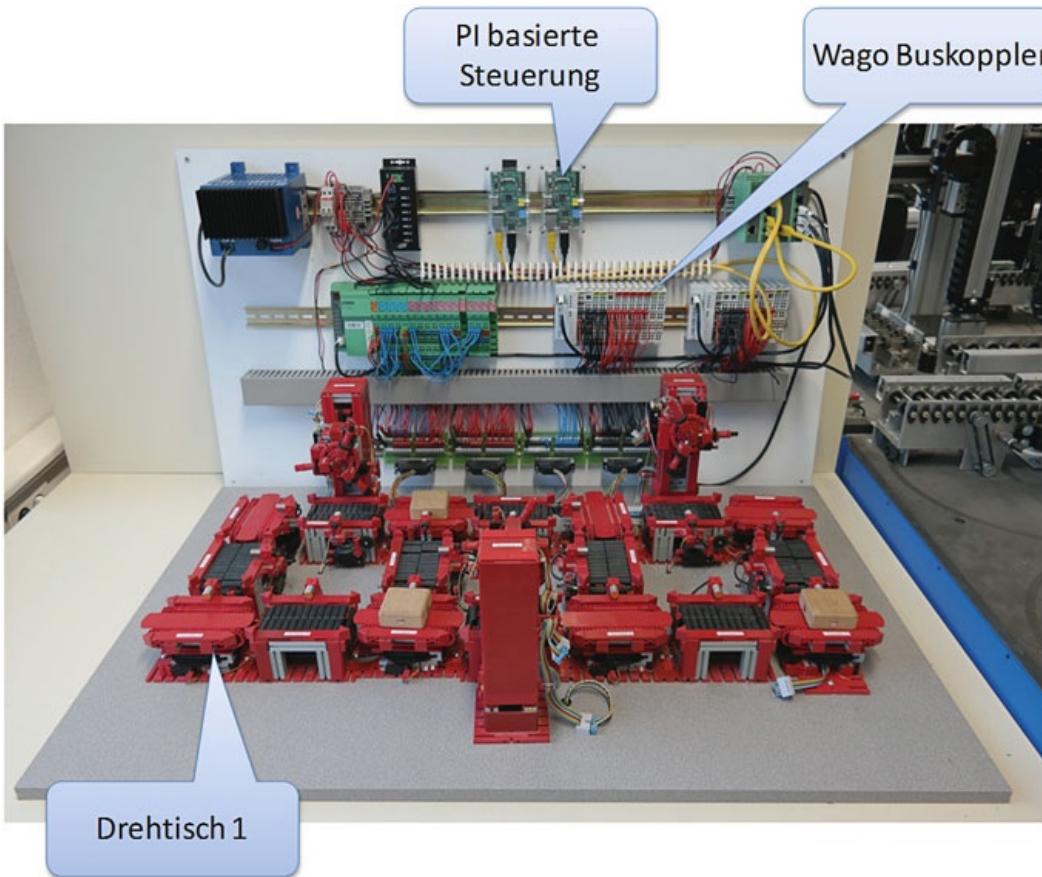


Quelle: plcopen.org

Beispiel	Beschreibung
	<ul style="list-style-type: none"> Gantt Charts <ul style="list-style-type: none"> Beschreiben einfache Abläufe Einsatz in frühen Engineering Phasen Erster Input für Simulationen und SPS-Code
	<ul style="list-style-type: none"> Impulse Diagramme <ul style="list-style-type: none"> Beschreiben einfache Abläufe Ergänzt um Rampen, interne und externe Signale Guter Input for Simulationen und SPS-Code
	<ul style="list-style-type: none"> Pert Charts <ul style="list-style-type: none"> Komplexe Ablaufbeschreibungen Detailliertes Zeitverhalten
	<ul style="list-style-type: none"> Sequence Function Charts (SFCs) <ul style="list-style-type: none"> Komplexe Abläufe und Verhalten Voll ausführbare Beschreibungen Sehr nahe an SPS-Code
	<ul style="list-style-type: none"> Zustandsdiagramme <ul style="list-style-type: none"> Komplexes Verhalten Häufig Event-getrieben Abbildung sehr intuitiv aber in der Regel nicht ausführbar

Quelle: [AML 2010]

AutomationML – Beispiel

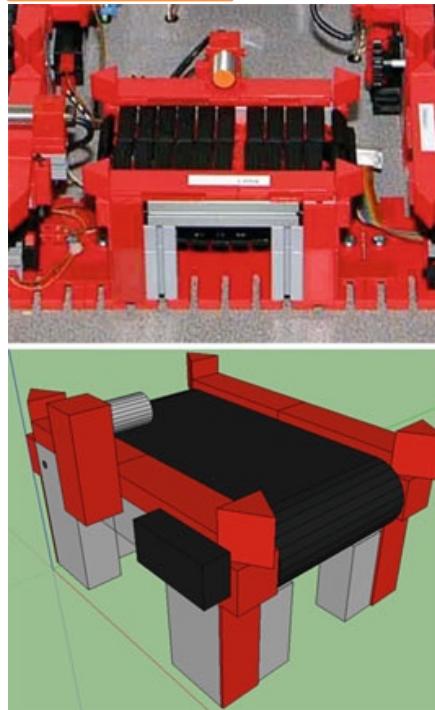


AutomationML – Beispiel (CAEX, COLLADA)

CAEX

```
IAFDemonstrationPlant
  FlexibleManufacturingSystem {Class: Role: Resource}
    TransportationLine {Class: Role: Cell}
      Drehtisch1 {Class: Drehtisch Role:}
        myMotor {Class: Motor Roles:}
          SupportedRoleClass: GeckoExampleEClassRoleClassLib/EClassClassification/27 Elektro-, Automatisierungs- und Prozesse
          SupportedRoleClass: AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/Device
          Interfaces
            MotorAn_PinA {Class: SignalInterface}
        myInduktivsensor {Class: Induktivsensor Roles:}
          SupportedRoleClass: GeckoExampleEClassRoleClassLib/EClassClassification/27 Elektro-, Automatisierungs- und Prozesse
          SupportedRoleClass: AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/Device
          Interfaces
            SensorWert_Pin1 {Class: SignalInterface}
        SupportedRoleClass: AutomationMLBaseRoleClassLib/AutomationMLBaseRole/Structure/ResourceStructure/MechatronicAsse
  ControlCabinet {Class: Role:}
    WagoIOA {Class: WagoIOSystem Role:}
      IOMapperApplikation {Class: IOMapperApplikation Roles:}
      Koppler750-341 {Class: WagoIO750-341 Role:}
        SupportedRoleClass: GeckoExampleEClassRoleClassLib/EClassClassification/27 Elektro-, Automatisierungs- und Prozesse
        SupportedRoleClass: ModbusTCPRoleClassLib/ModbusTCPPPhysicalDevice
        Interfaces
          EthernetSocket {Class: ModbusTCPSocket}
    D11 750-437 {Class: WagoIOinput750-437 Role:}
      SupportedRoleClass: GeckoExampleEClassRoleClassLib/EClassClassification/27 Elektro-, Automatisierungs- und Prozesse
      SupportedRoleClass: CommunicationRoleClassLib/PhysicalDevice
      Interfaces
        D12 750-437 {Class: WagoIOinput750-437 Role:}
        D01 750-530 {Class: WagoIOOutput750-530 Role:}
        D02 750-530 {Class: WagoIOOutput750-530 Role:}
      SupportedRoleClass: CommunicationRoleClassLib/PhysicalDevice
    PIBasedController1 {Class: PIBasedController Role:}
      MyPIProgram {Class: PIProgram Roles:}
      MyPINetworkCard {Class: PINetworkCard Roles:}
        SupportedRoleClass: GeckoExampleEClassRoleClassLib/EClassClassification/27 Elektro-, Automatisierungs- und Prozesse
        SupportedRoleClass: CommunicationRoleClassLib/PhysicalDevice
```

COLLADA

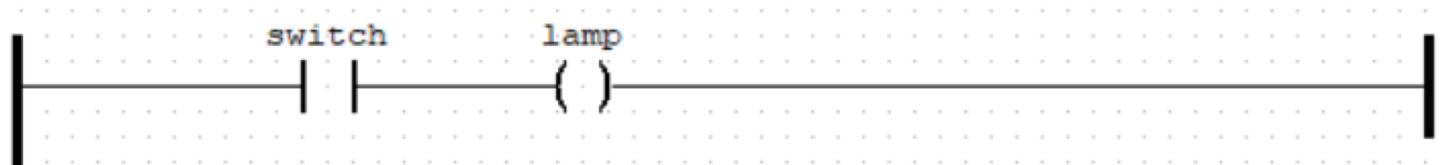


```
<COLLADA xmlns="http://www.collada.org/2005/11/COLLADASchema" version="1.4.1">
  <asset>
    <library_visual_scenes>
      <visual_scene id="ID1">
        ...
      </visual_scene>
    </library_visual_scenes>
    <library_nodes>
      <node id="ID3" name="Conveyer">
        ...
      </node>
      <node id="ID5" name="Sensor_gesamt">
        ...
      </node>
      <node id="ID7" name="Sensor">
        ...
      </node>
      <node id="ID17" name="Sensor_Ständer">
        ...
      </node>
      <node id="ID27" name="Motor">
        ...
      </node>
      <node id="ID37" name="ganze_Gestell">
        ...
      </node>
      <node id="ID39" name="Gestell">
        ...
      </node>
      <node id="ID41" name="Komponente_1">
        ...
      </node>
      <node id="ID49" name="Komponente_2">
        ...
      </node>
      <node id="ID63" name="Komponente_2">
        ...
      </node>
      <node id="ID78" name="Komponente_1">
        ...
      </node>
      <node id="ID86" name="Komponente_3">
        ...
      </node>
      <node id="ID94" name="Komponente_3">
        ...
      </node>
      <node id="ID103" name="Dreieck">
        ...
      </node>
      <node id="ID114" name="Band_Conveyer2">
        ...
      </node>
      <node id="ID116" name="Band_2">
        ...
      </node>
      <node id="ID130" name="Rolle">
        ...
      </node>
    </library_nodes>
    <library_geometries>
    <library_materials>
    <library_effects>
    <scene>
      <instance_visual_scene url="#ID1" />
    </scene>
  </COLLADA>
```

AutomationML – Beispiel (PLCOpen)

```

-<LD>
-<leftPowerRail localId="1" height="140" width="20">
  <position x="80" y="20"/>
  -<connectionPointOut formalParameter="">
    <relPosition x="20" y="20"/>
  </connectionPointOut>
</leftPowerRail>
-<contact localId="2" height="20" width="21" negated="false">
  <position x="210" y="30"/>
  -<connectionPointIn>
    <relPosition x="0" y="10"/>
    -<connection refLocalId="1">
      <position x="210" y="40"/>
      <position x="100" y="40"/>
    </connection>
  </connectionPointIn>
  -<connectionPointOut>
    <relPosition x="21" y="10"/>
  </connectionPointOut>
  <variable>switch</variable>
</contact>
-<coil localId="3" height="20" width="21" negated="false">
  <position x="310" y="30"/>
  -<connectionPointIn>
    <relPosition x="0" y="10"/>
    -<connection refLocalId="2">
      <position x="310" y="40"/>
      <position x="231" y="40"/>
    </connection>
  </connectionPointIn>
  -<connectionPointOut>
    <relPosition x="21" y="10"/>
  </connectionPointOut>
  <variable>lamp</variable>
</coil>
-<rightPowerRail localId="6" height="40" width="10">
  <position x="650" y="20"/>
  -<connectionPointIn>
    <relPosition x="0" y="20"/>
    -<connection refLocalId="3">
      <position x="650" y="40"/>
      <position x="331" y="40"/>
    </connection>
  </connectionPointIn>
</rightPowerRail>
</LD>
  
```



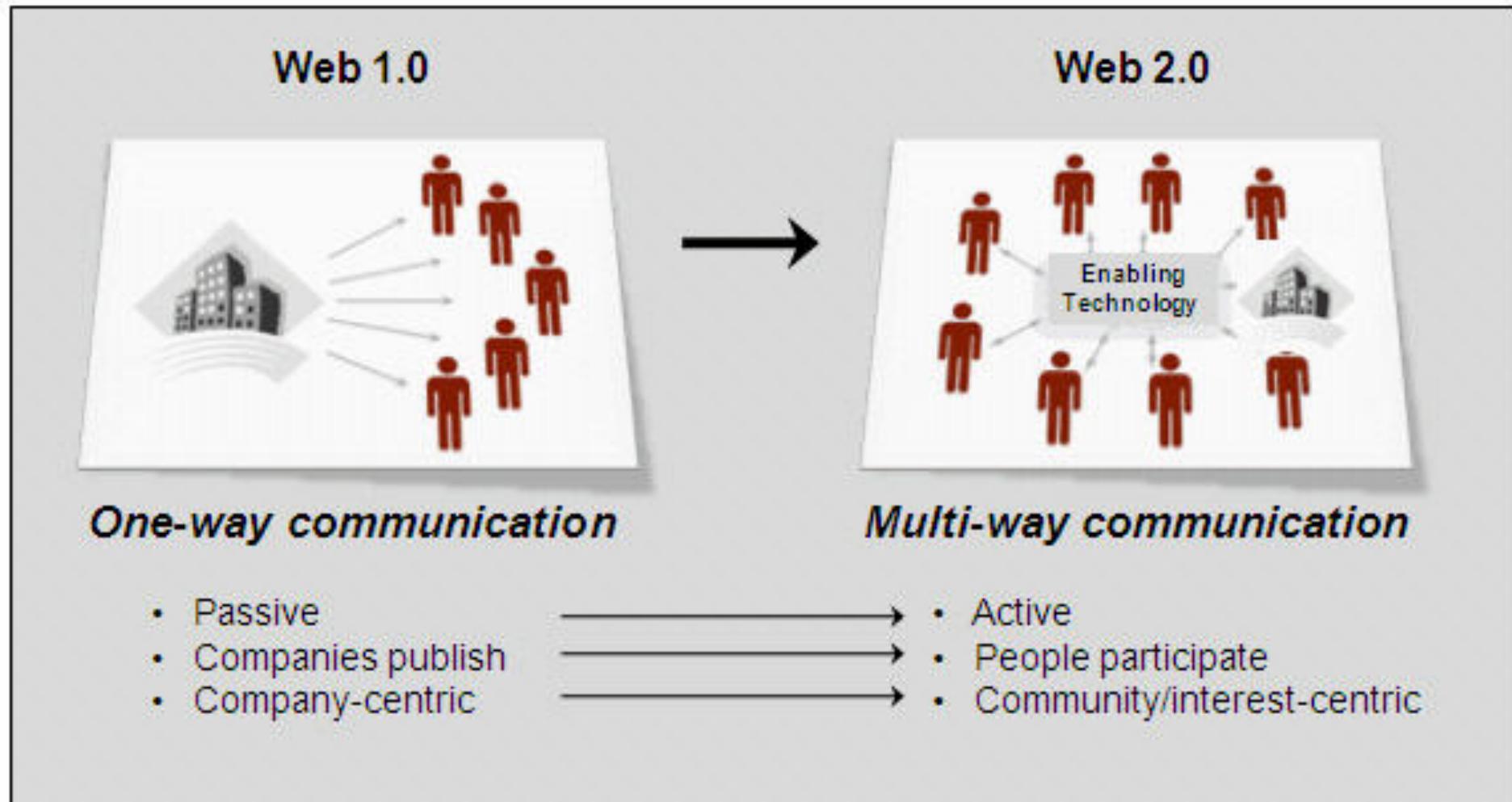
- Für die Realisierung der im Kontext von Industrie 4.0 angestrebten flexiblen, vernetzten und intelligenten Produktion, sind semantische Beschreibungen von Objekten, Prozessen und Beziehungen von elementarer Bedeutung
- AutomationML bietet hierfür einen standardisierten und beliebig erweiterbaren Standard an
- Daher wird AutomationML auch als „Digital Enabler“ und damit als Wegbereiter für die Industrie 4.0 angesehen

Empfohlene und weiterführende Literatur

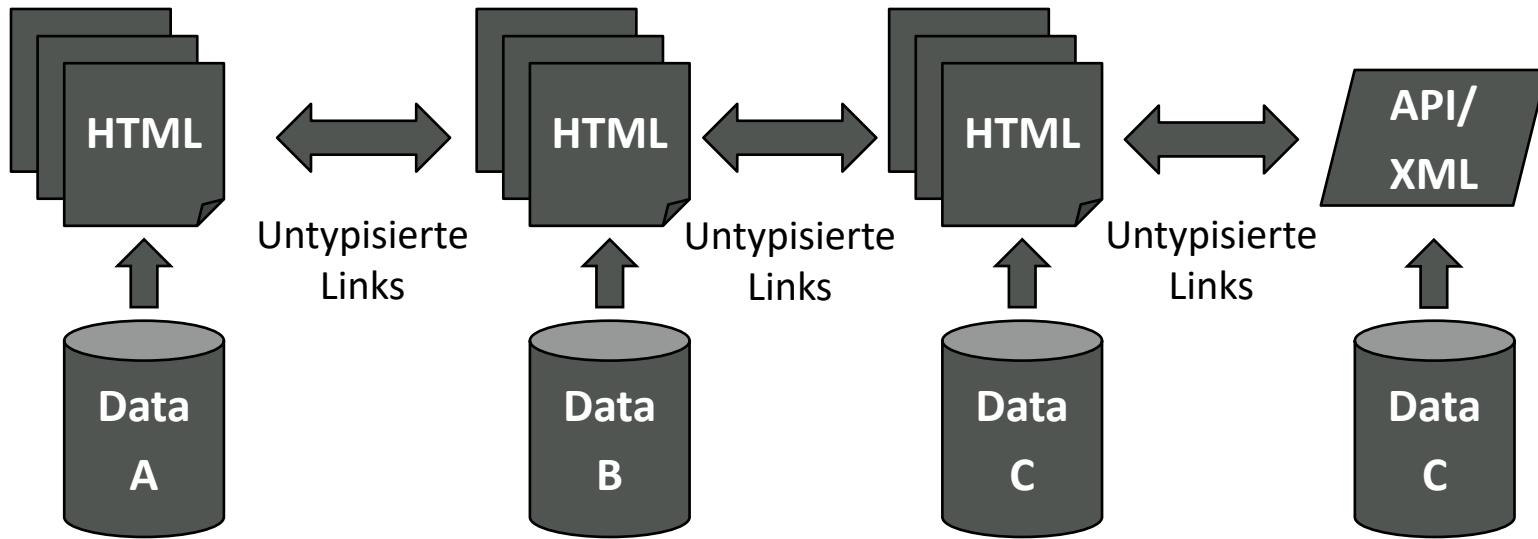
- [Adolphs 2015] Peter Adolphs et al.: „Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)“; Statusreport; April 2015; VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik und ZVEI e.V., https://www.vdi.de/fileadmin/user_upload/VDI-GMA_Statusreport_Refenenzarchitekturmodell-Industrie40.pdf
- [AML 2009] AutomationML e.V. „AutomationML Core Presentation at HMI2009“, https://www.automationml.org/o.red/uploads/dateien/1317722297-AutomationML_Core_HMI2009.pdf
- [AML 2015-1] AutomationML e.V. „AutomationML in a Nutshell“, https://www.automationml.org/o.red/uploads/dateien/1447420977-AutomationML%20in%20a%20Nutshell_151104.pdf
- [AML 2014-1] AutomationML e.V. „Whitepaper AutomationML Part 1 - Architecture and general requirements“, https://www.automationml.org/o.red/uploads/dateien/1417686950-AutomationML%20Whitepaper%20Part%201%20-%20AutomationML%20Architecture%20v2_Oct2014.pdf
- [AML 2014-2] AutomationML e.V. „Whitepaper AutomationML Part 2 – Role class libraries“, https://www.automationml.org/o.red/uploads/dateien/1417686992-AutomationML%20Whitepaper%20Part%202%20-%20AutomationML%20Role%20Class%20Libraries%20v2_Oct2014.pdf
- [AML 2015-2] AutomationML e.V. „Whitepaper AutomationML Part 3 – Geometry and Kinematics“, https://www.automationml.org/o.red/uploads/dateien/1446724412-AutomationML%20Whitepaper%20Part%203%20-%20AutomationML%20Geometry%20v2_Aug2015.pdf
- [AML 2013] AutomationML e.V. „Whitepaper AutomationML Part 4 – Logic Description“, https://www.automationml.org/o.red/uploads/dateien/1417686916-AutomationML%20Whitepaper%20Part%204%20-%20AutomationML%20Logic%20Description%20v2_Aug2013.pdf
- [Bock 2016] Jürgen Bock et al.: „Weiterentwicklung des Interaktionsmodells für Industrie 4.0-Komponenten“; Diskussionspapier; Nov. 2016; Plattform Industrie 4.0; <http://www.plattform-i40.de/i40/Redaktion/DE/Downloads/Publikation/interaktionsmodell-i40-komponenten-it-gipfel.pdf?blob=publicationFile&v=10>
- [Draht 2009] Drath, Rainer, ed. *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Springer-Verlag, 2009.

Überblick

SEMANTIC WEB

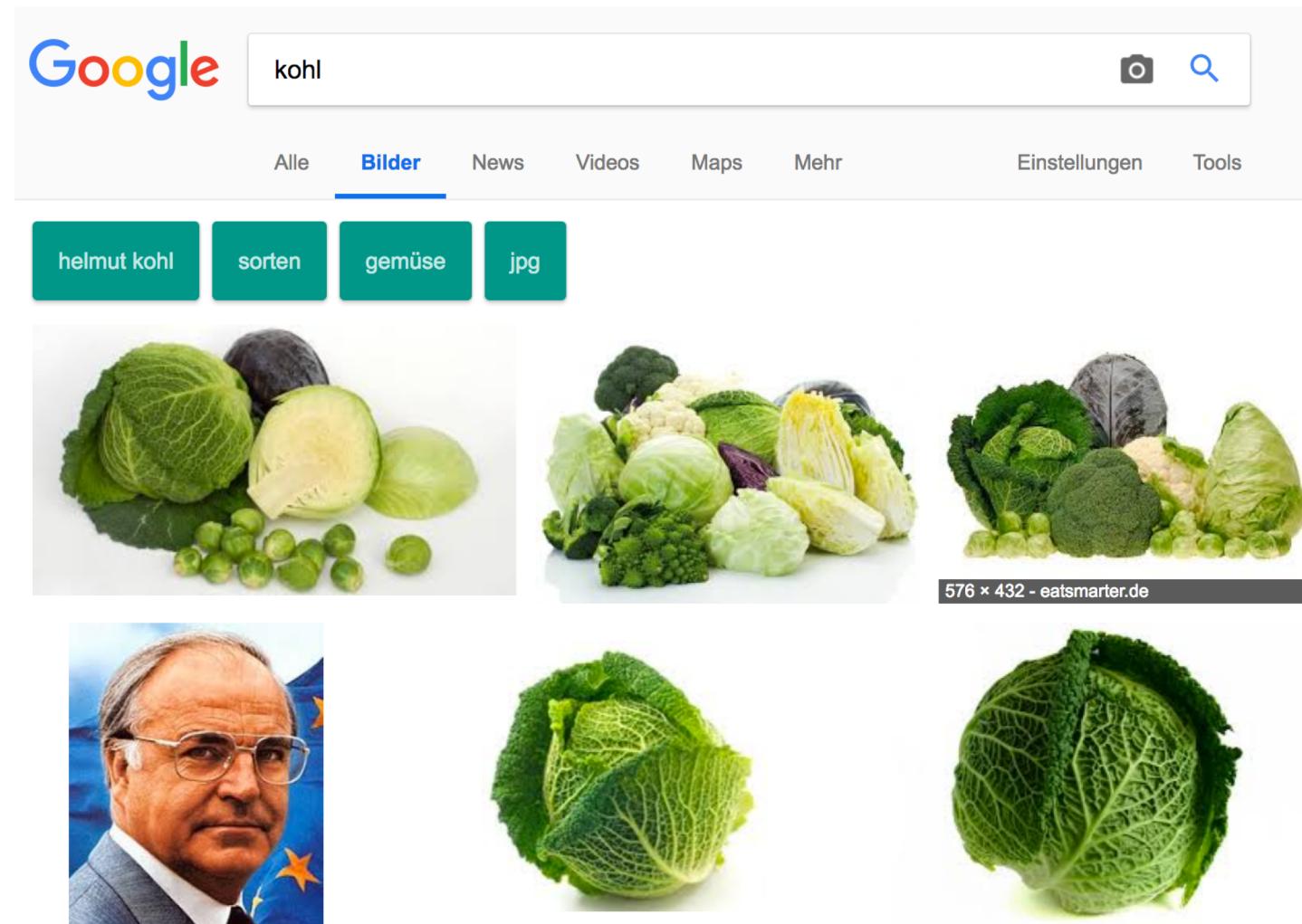


Das „klassische“ Web



- Hauptinhalte: **Dokumente** (Bilder, Text, Videos ...)
- **Verbindungen** zwischen Dokumente durch **Hyperlinks** im Dokument
- **Niedrige Strukturierung** der Daten
- Entworfen für die **menschliche Nutzung**

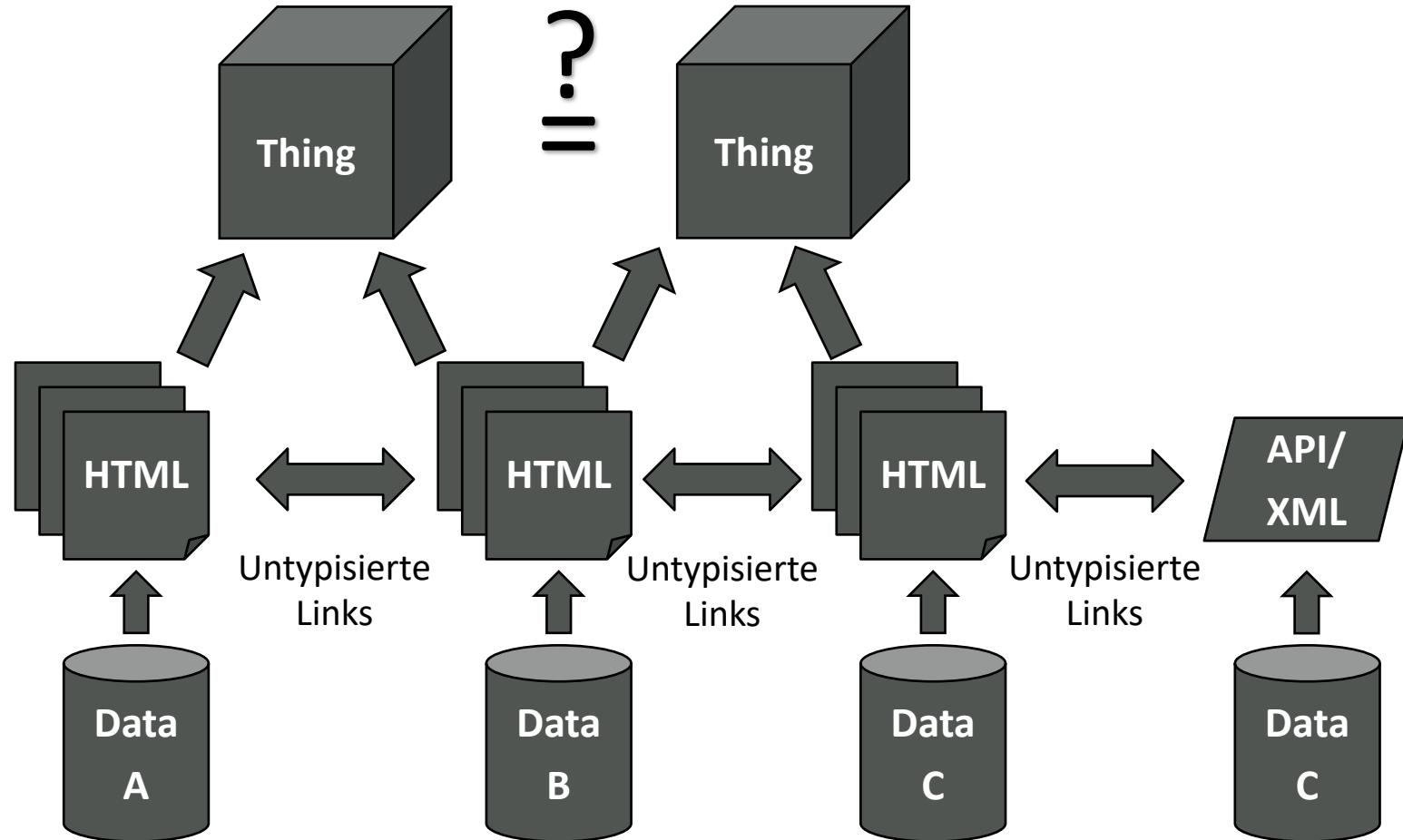
Das „klassische“ Web: Problem



Quelle: www.google.de

Das „klassische“ Web: Problem

Reden beide Dokumente über die gleiche Sache?



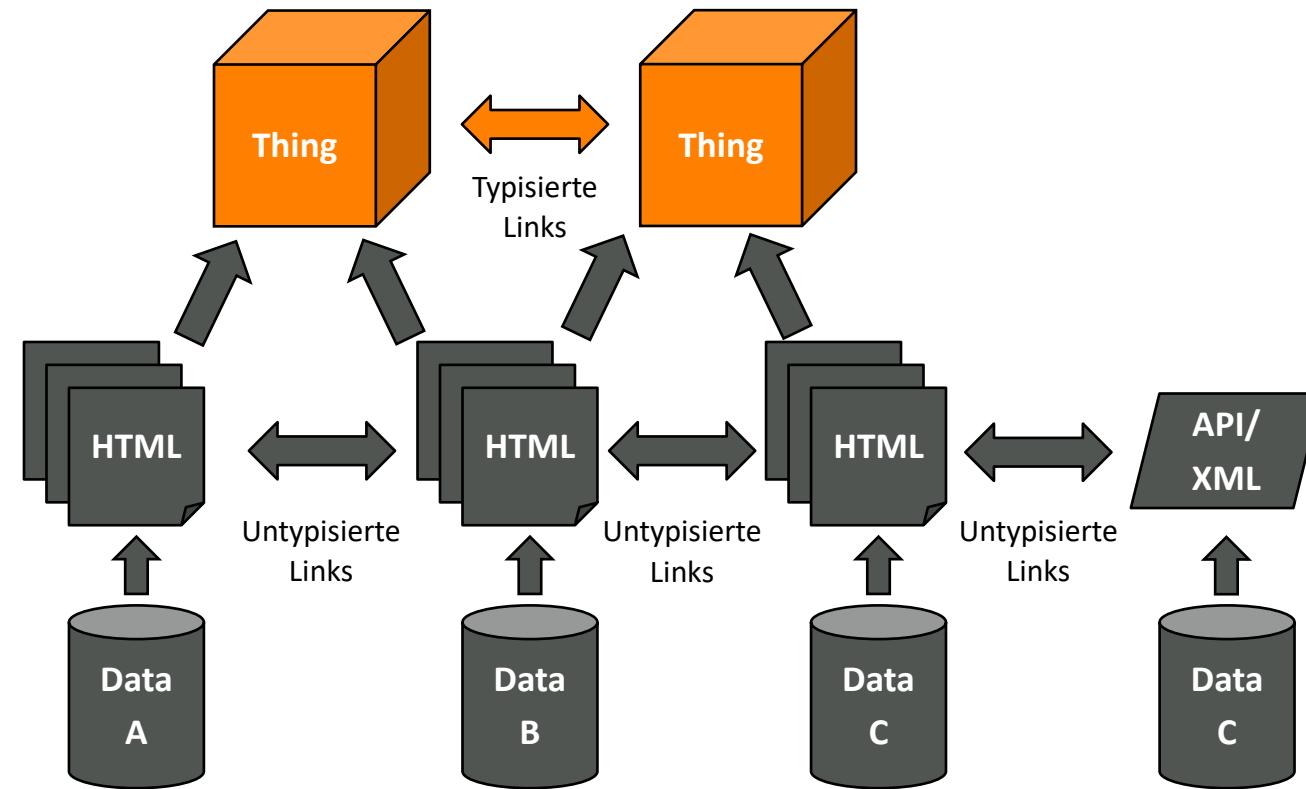
Komplexe Suchanfragen nicht möglich:

- Wie viele Hollywood Filme gibt es, die von einer Person gedreht wurde, welche in einer Stadt geboren wurden, wo die Durchschnittstemperatur über 15 Grad liegt?

Lösung:

- Suchen nach Quellen aller Hollywood Filme mit den Regisseure
- Suchen von Geburtsorten der Regisseure
- Suchen nach Durchschnittstemperatur der Geburtsstädte
- Auswerten der Ergebnisse

Lösung: Semantic Web

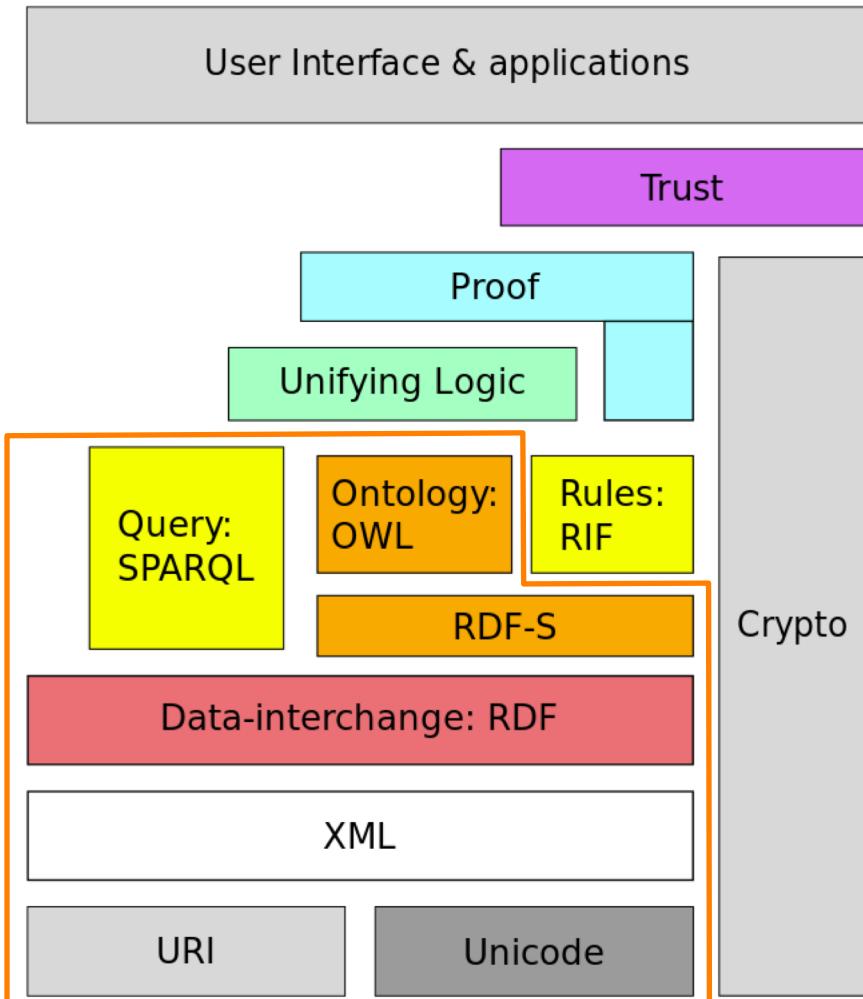


- Hauptinhalte: **Dinge (Things)**
- **Verbindungen** zwischen Dinge durch typisierte Links
- **Hohe Strukturierung** der Daten
- Entworfen für die **Nutzung für Mensch und Maschine**

„The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation“

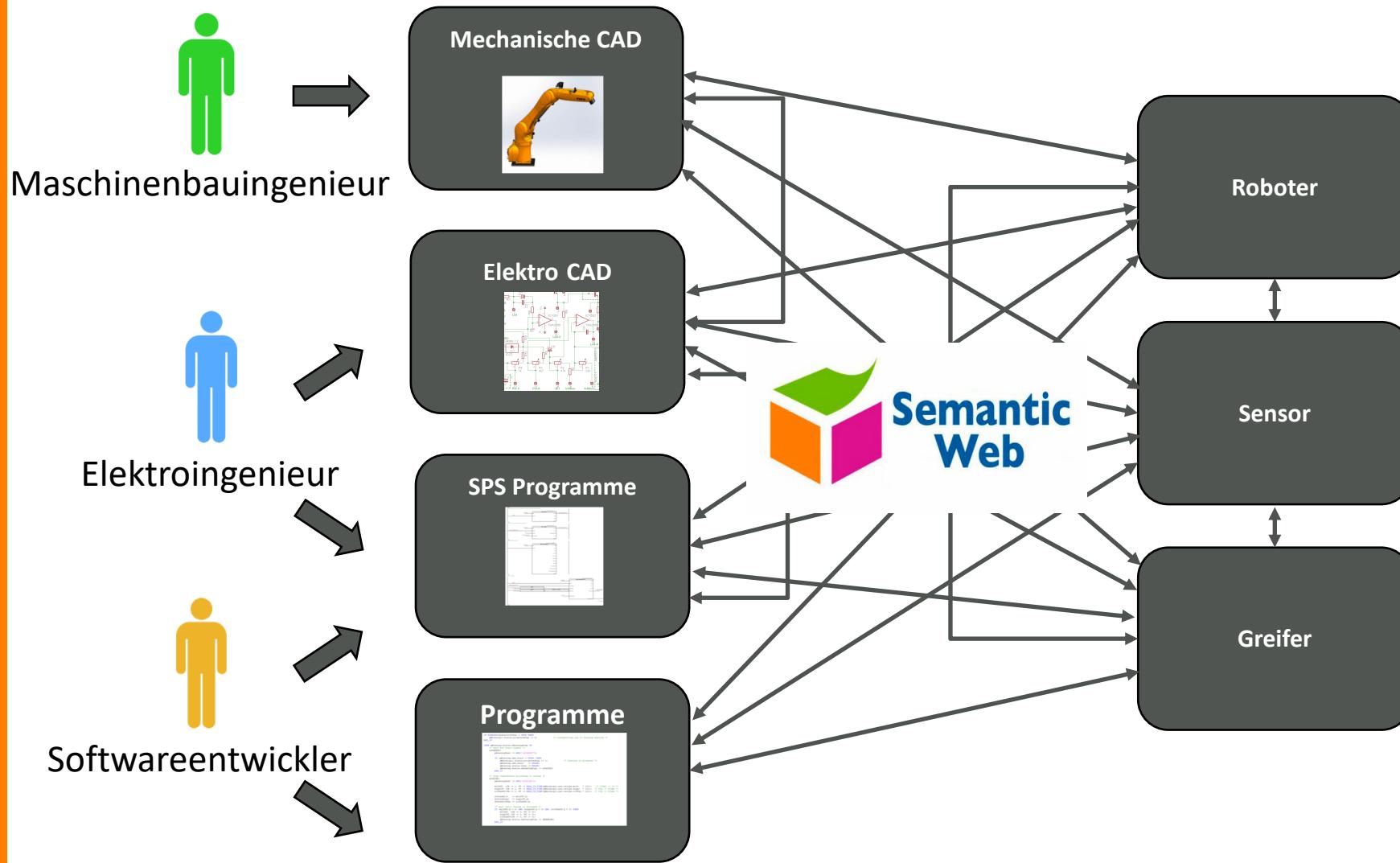
Tim Berners-Lee

- Resource Description Framework (RDF)
- RDF Schema (RDF-S)
- Web Ontology Language (OWL)
- SPARQL Protocol And RDF Query Language (SPARQL)



Quelle: www.w3c.org

Warum Semantic Web in der Industrie 4.0?



- **Ressourcen (Resource)**

- Alles was durch eine URI referenziert werden kann, wird als Ressource bezeichnet
- Beispiel: Webseiten, Fotos, Personen

- **Beschreibung (Description)**

- Informationen über eine Ressource, wie Attribute oder Relationen

- **Framework**

- Sammlung von Modellen, Sprachen und einer einheitlichen Syntax

RDF ist der **Hauptbestandteil des Semantic Webs** und dient als **Syntax** für das **Verknüpfen von Daten**

Resource Description Framework (RDF)

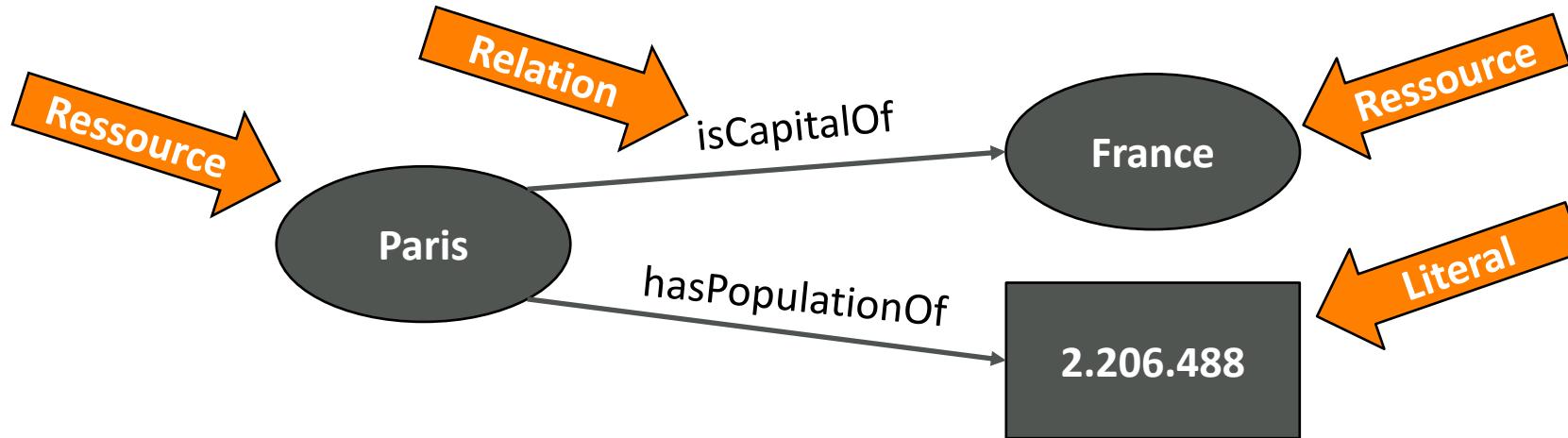
- RDF dient der **Beschreibung von Dingen (Thing)** und ihren **Verbindungen**
- **Verbindungen** zwischen den Dingen werden in **Tripel** abgebildet, welche aus **Subjekt – Prädikat – Objekt** bestehen
 - Subjekt ist eine **Ressource (Thing)**
 - Prädikat ist eine **Relation**
 - Objekt kann eine **Ressource (Thing)** oder ein **Literal (Value)** sein



Paris	ist Hauptstadt von	Frankreich
Paris	hat eine Einwohnerzahl von	2.206.488

Resource Description Framework (RDF)

- Aus **mehreren RDF-Tripel** können **RDF-Graphen** gebildet werden
 - Knoten sind **Ressourcen** und **Literale**
 - Kanten bilden die **Relationen**

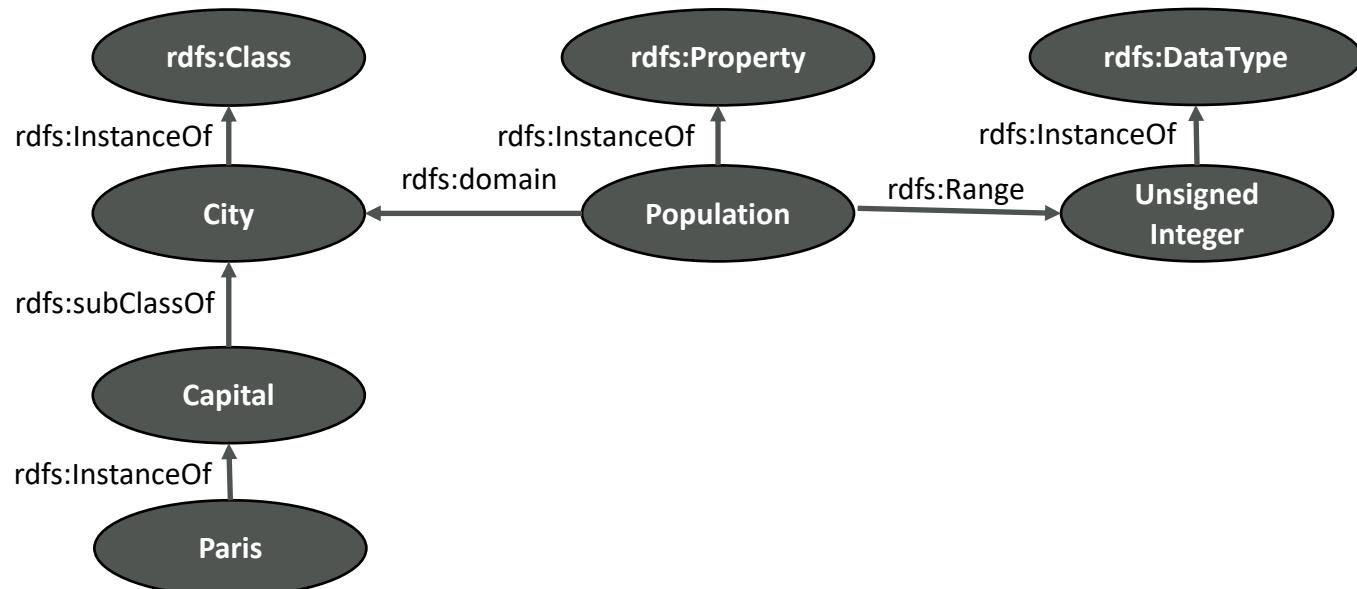


- Identifizierung von Ressourcen durch URLs

Resource Description Framework (RDF)

- Eindeutige Identifizierung von allen Ressourcen durch URIs
 - z. B. Paris: <http://dbpedia.org/resource/Paris>
- Repräsentation der Tripel durch RDF/XML, N-Triples oder JSON
 - RDF/XML (Standard):
 - <rdf:Description rdf:about="http://dbpedia.org/resource/Paris"> ← Subjekt
<db:capital> ← Prädikat
<rdf:Description rdf:about="http://dbpedia.org/page/France"> ← Objekt
</db:capital>
</rdf:Description>
 - N-Triples
 - <Subjekt> <Prädikat> <Objekt>.
 - JSON
 - { "Subjekt" : { "Prädikat": [Objekt] } }

- Erweiterung von RDF um ein **einheitliches Vokabular** zur **Interpretation von Zusammenhängen** zwischen Daten
- Vokabular für **Gruppen von Ressourcen** (Klassen) und **Relationen zwischen Ressourcen** (Eigenschaften)
 - Klassen
 - rdfs:Class
 - rdfs:DataType
 - ...
 - Eigenschaften
 - rdfs:subClassOf
 - rdfs:InstanceOf
 - rdfs:range
(Menge der möglichen Objekte)
 - rdfs:domain
(Menge der möglichen Subjekte)
 - ...



- Spezifikation des World Wide Web Consortiums (W3C) zur formalen Beschreibung von Ontologien
- Dabei werden Beziehungen formal beschrieben, dass auch Software die Bedeutung verarbeiten kann (Reasoning)
- OWL erlaubt es Ausdrücke ähnlich der Prädikatenlogik zu formulieren
 - Beispiel
 - $\forall x \text{Hauptstadt}(x) \Rightarrow \text{Stadt}(x)$
 - Für alle x gilt: Wenn x eine Hauptstadt ist, dann ist x eine Stadt
 - $\neg \forall x: \text{Städte}(\text{Hauptstadt}(x))$
 - „Nicht jede Stadt ist eine Hauptstadt“

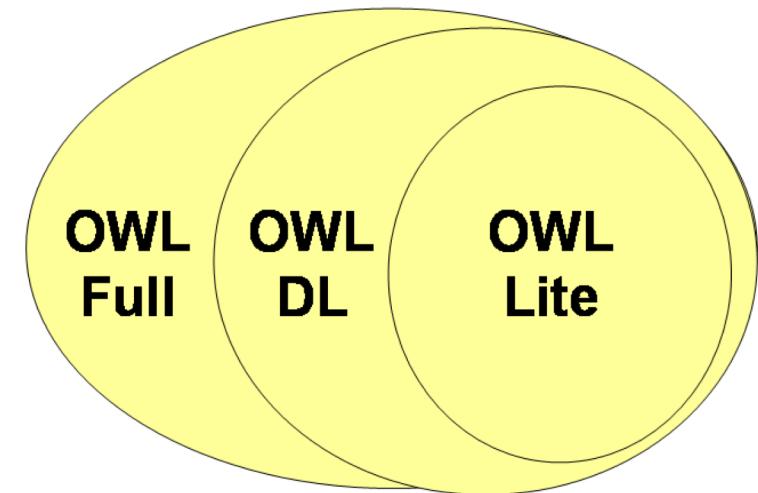
- OWL Ontologien bestehen aus **Klassen, Eigenschaften** (Properties) und **Individuals** (Instanzen von Klassen)
- OWL verwendet **Open World Assumption** (= Die Existenz von weiteren Individuen ist möglich, sofern sie nicht explizit ausgeschlossen wird)
 - Beispiel: Sind alle Kinder von Bill männlich

Aussage	Open World Assumption	Closed World Assumption (=Wissenbasis enthält alle Individuen und Fakten)
Bill hat das Kind Bob Bob ist männlich	Unbekannt	Ja
Bill hat das Kind Bob Bob ist männlich Bill hat höchstens ein Kind	Ja	Ja

OWL Lite \subseteq OWL DL \subseteq OWL Full

Unterscheidung in drei Varianten

- OWL Lite *SHIF(D)*
 - entscheidbar
 - Wenig ausdrucksstark
- OWL DL *SHOIN(D)*
 - Entscheidbar
 - Wird von aktuellen Softwarewerkzeugen unterstützt
- OWL Full
 - Unentscheidbar



Quelle: www.w3c.org

- **SHIF(D)** steht für

- **S** ist die Beschreibungslogik ALC (Attributive Concept Language with Complements) mit transitiven Rollen

- Konzeptname ist ein Konzept
 - Thing (T) und Nothing (\perp) sind Konzepte
 - Für C und D Konzepte, $\neg C$, $C \cap D$, und $C \cup D$ sind Konzepte
 - Für R eine Rolle und C ein Konzept, $\forall R:C$ und $\exists R:C$ sind Konzepte
 - Transitive Rollen (Beispiel: hatVorfahre)



- **H** = Rollenhierarchien
 - Beispiel: hatMutter \sqsubseteq hatEltern, hatVater \sqsubseteq hatEltern
 - **I** = inverse Rollen
 - Beispiel: hatKind $^{-}$ \equiv hatEltern
 - **F** = funktionale Rollen
 - Beispiel: hatMutter (hat höchstens eine Mutter)
 - **(D)** = Datentypen

- Für $SHOIN(D)$ werden folgen Beschreibungen der Beschreibungslogik hinzugefügt
 - **O** = Nominals (Definition durch Aufzählung)
 - Beispiel: {männlich, weiblich}
 - **N** = Zahlenrestriktionen
 - Beispiel: ≥ 3 hatKind (Klasse aller mit mindestens 3 Kinder)

OWL vs. Beschreibungslogik

OWL Befehl	Beschreibungslogik Syntax	Beispiel
owl:intersectionOf	$C \cap D$	Mensch \cap Weiblich
owl:unionOf	$C \cup D$	Mann \cup Frau
owl:complementOf	$\neg C$	\neg Mann
owl:allValuesFrom	$\forall P.C$	\forall hatKind. Weiblich
owl:someValuesFrom	$\exists P.C$	\exists hatKind. Männlich
owl:subClassOf	$C \sqsubseteq D$	Mann \sqsubseteq Mensch
owl:subPropertyOf	$P \sqsubseteq Q$	hatTochter \sqsubseteq hatKind
owl:transitiveProperty	$P^+ \sqsubseteq P$	hatNachfahren ⁺ \sqsubseteq hatNachfahren
owl:oneOf	$\{x\} \cup \{y\}$	{Max} \cup {Moriz}
owl:maxCardinality	$\leq n P$	≤ 1 hatKind
owl:minCardinality	$\geq n P$	≥ 2 hatKind

C, D sind Konzepte (Klassen); P, Q sind Rollen (Eigenschaften); x, y sind individuelle Namen

Beispiel: OWL vs. Beschreibungslogik

Beispiel: Klasse aller Personen bei denen alle Kinder Studenten sind, oder die Enkel Studenten sind.

Person $\cap \forall \text{hasChild}.\text{Student} \cup \exists \text{hasChild.}\underline{\text{Student}}$

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Student"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Student"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

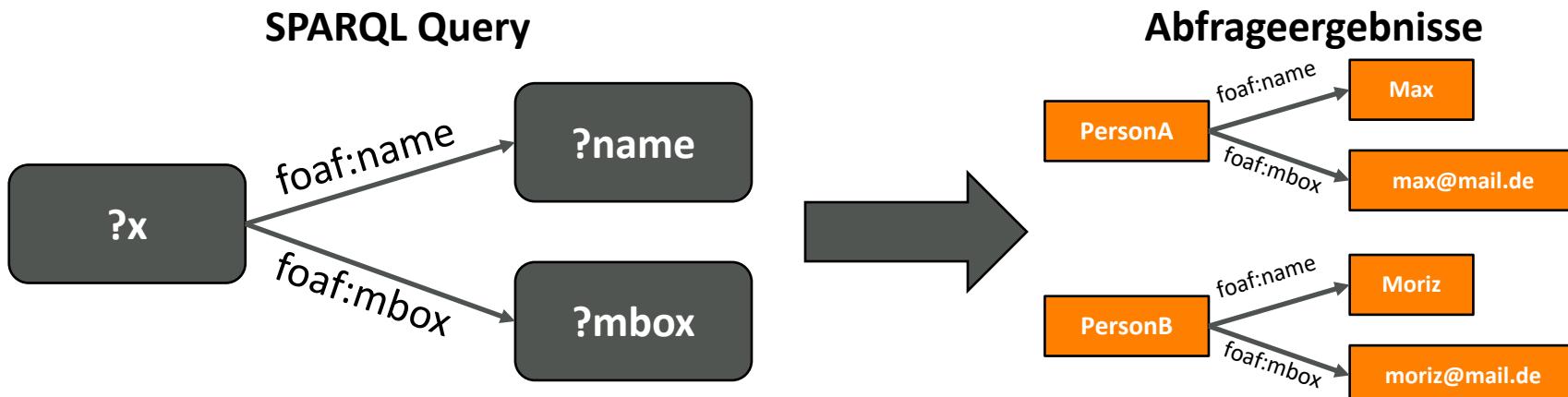
- **Standard Abfragesprache** zur Abfrage von Graphdaten die als RDF Tripel repräsentiert sind
- Einer der drei **Grundstandards des Semantic Web**, neben RDF und OWL
- Seit 2008 W3C Spezifikation
- Möglichkeit der Abfrage oder **Modifikation von RDF Graphen** (Hinzufügen, Löschen)
- Ähnliche Syntax zu SQL

SPARQL: Einfache Anfragen

- Einfache Beispielanfrage:

```
PREFIX foaf:<http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE { ?x foaf:name ?name .
        ?x foaf:mbox ?mbox }
```

- PREFIX** dient zum Abkürzen von URIs (Beispiel: `foaf` (Friend of a Friend - Ontology))
- SELECT** gibt die selektierten Variablen das Abfrageergebnis an
- WHERE** gibt das Abfragemuster an und werden als Basic Graph Pattern (BGP) angegeben
 - BGPs werden in Turtle Syntax angegeben
 - BGPs können Variablen enthalten (Beispiel: `?name`) die bei der Abfrage durch Ressourcen ersetzt werden



- **FROM**
 - Auswahl von Graphen der Ontologie, dadurch kann das Dataset ausgewählt werden
 - Klausel nicht notwendig, wenn nichts angegeben wird, wird der Default Graph verwendet
- **OPTIONAL**
 - Erlaubt optionale Teile des Abfragemusters (**WHERE**)
- **UNION**
 - Angabe von alternativen Teilen des Abfragemusters
- **{...}**
 - Gruppierung von Graph-Muster
- **FILTER**
 - Ermöglicht Filterung von Abfragen
 - Verwendung von Vergleichsoperatoren (<, =, >, <=, >=, !=), arithmetischen Operatoren (+, -, *, /), booleschen Operatoren (&&, ||, !), regulären Ausdrücken (REGEX(A,B)), ... möglich

```
SELECT ?book ?price ?titel
WHERE{
    { ?book ex:Price ?price .  

      FILTER (?price < 15 || ?price >= 20)  

      OPTIONAL { ?book ex:Titel ?titel }  

      { ?book ex:Author ex:Shakespeare } UNION  

      { ?book ex:Author ex:Marlowe }  

    }
}
```

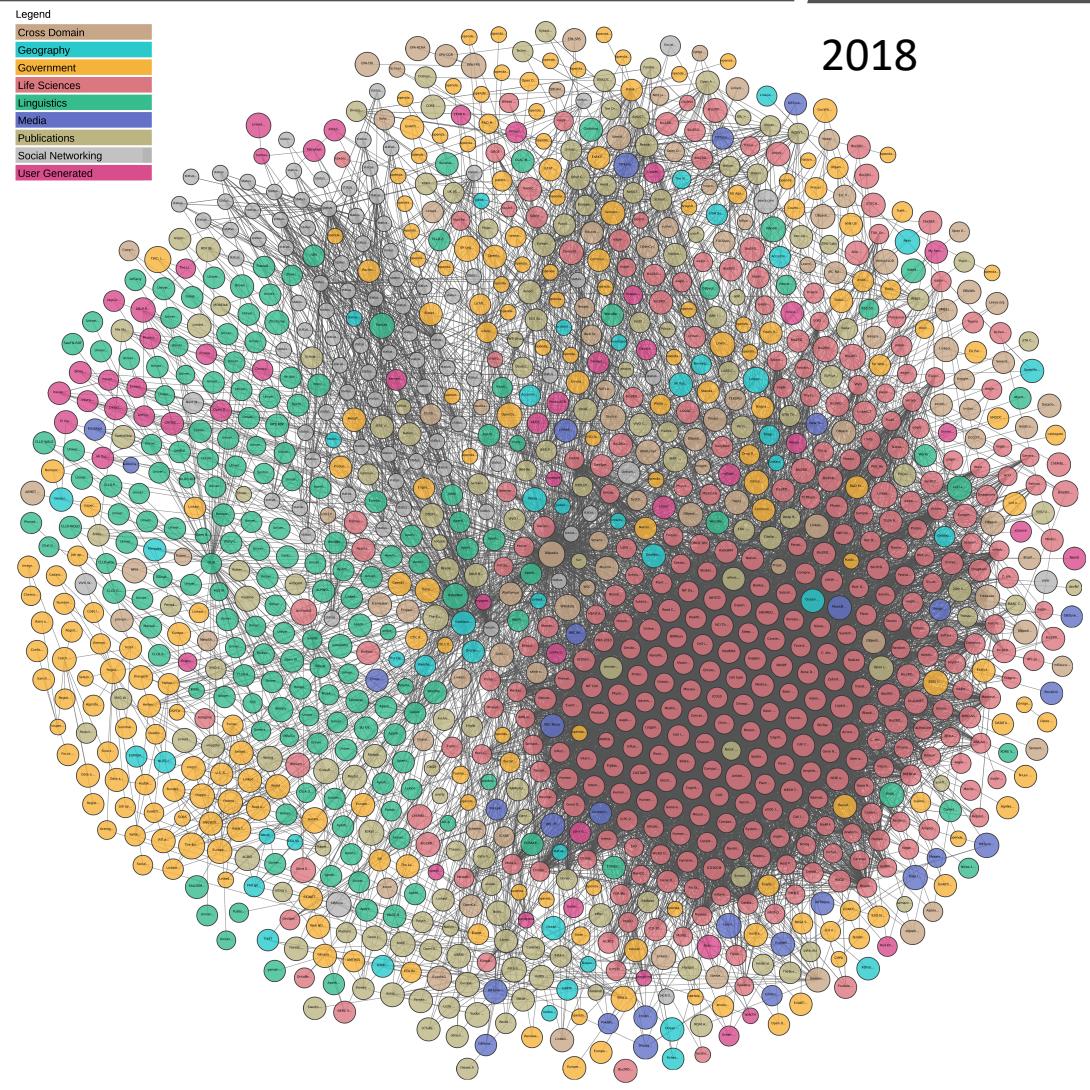
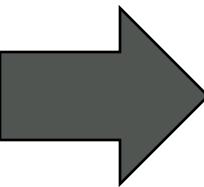
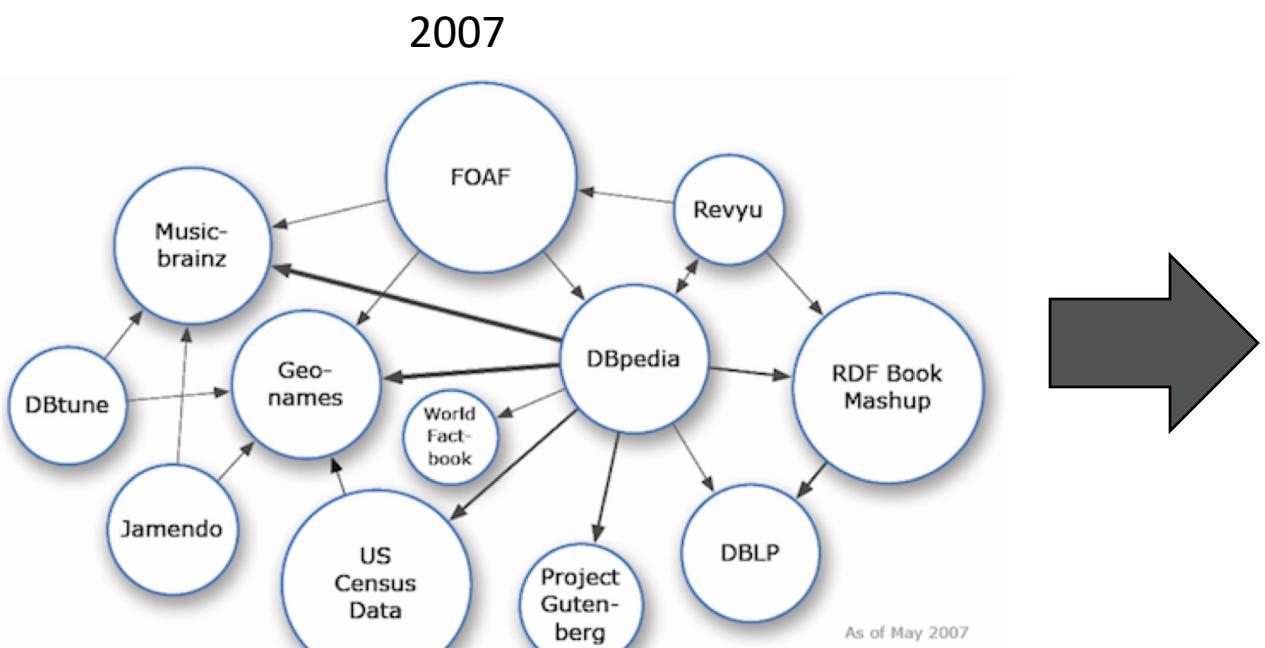
Ausgabe:

- Alle Bücher mit Preis und Titel
 - Preis muss kleiner als 15 oder größer gleich 20 sein
 - Es muss kein Titel vorhanden sein
 - Alle Bücher müssen von Shakespeare oder Marlowe als Author haben

- SPARQL bietet die Möglichkeit Graphen zu modifizieren
 - `INSERT {<http://example/egbook3> ex:Price 20}`
 - Fügt ein Tripel ein
 - `DELETE {<http://example/egbook3> ex:Price 21}`
 - Löscht das angegebene Tripel
 - `DELETE { ?book ?p ?o }`
`WHERE { ?book dc:date ?date .`
 `FILTER (?date < "2000-01-01T00:00:00"^^xsd:dateTime)`
 `?book ?p ?o`
`}`
 - Löscht alle Bücher die vor 2000 erschienen sind
 - Es gibt keine `UPDATE`-Funktion, wird aus `DELETE` und `INSERT` zusammengesetzt

- **Linked** = **Verknüpfungen** zwischen Daten
- **Open** = **freie Nutzbarkeit** der Daten
- Das Konzept von Linked Open Data geht im Wesentlichen auf Tim Berners Lee zurück
- Er stellte auch die **Regeln für Linked Data** auf
 - Use URIs as names for things
 - Use HTTP URIs so that people can look up those names.
 - When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
 - Include links to other URIs, so that they can discover more things

Linked Open Data Cloud



Empfohlene und weiterführende Literatur

- [Hitzler 2008] Pascal Hitzler et al.: „Semantic Web: Grundlagen“; Springer-Verlag, 2008.
- [Domingue 2011] John Domingue et al.: „Handbook of semantic web technologies“; Springer-Verlag, 2011
- [Gradmann 2016] Stefan Gradmann et al.: „Semantic Web und Linked Data“. In: Grundlagen der praktischen Information und Dokumentation – Handbuch zur Einführung in die Informationswissenschaft und – praxis; 2016, https://lirias2repo.kuleuven.be/bitstream/handle/123456789/485162/B7_05.01.2013_revSGN.pdf

- [RDF 2014] World Wide Web Consortium u. a. „RDF 1.1 concepts and abstract syntax“, 2014, <https://www.w3.org/TR/rdf11-concepts/>
- [RDFS 2014] World Wide Web Consortium u. a. „RDF Schema“, 2014, <https://www.w3.org/TR/rdf-schema/>
- [OWL 2014] World Wide Web Consortium u. a. „OWL Web Ontology Language Overview“, 2014, <https://www.w3.org/TR/owl2-overview/>
- [SPARQL 2008] World Wide Web Consortium u. a. „SPARQL Query Language for RDF“, 2008, <https://www.w3.org/TR/rdf-sparql-query/>