14. Structure of M2 in a Technical Space (Language Families and Composition of Tools)

Prof. Dr. U. Aßmann

Technische Universität Dresden

Institut für Software- und Multimediatechnik

http://st.inf.tu-dresden.de/teaching/most

Version 19-0.3, 19.11.19

1) Problem of Tool Composition
2) Data definition languages
3) Query languages
4) Constraint languages
5) Reuse languages
6) Transformation and Restructuring languages
7) Behavior specification languages
8) Language families in several technical spaces
9) .. and all together now...

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Obligatory Literature

- http://en.wikipedia.org/wiki/List_of_UML_tools
- [HRC-MM] The SPEEDS project. Deliverable D.2.1.5. SPEEDS L-1 Meta-Model, Revision: 1.0.1, 2009
  - http://speeds.eu.com/downloads/SPEEDS_Meta-Model.pdf
- [HRC-Kit] The SPEEDS project. SPEEDS Training Kit.
  - http://www.speeds.eu.com/downloads/Training_Kit_and_Report.zip
  - Training_Kit_and_Report.pdf: Overview
  - Contract-based System Design.pdf: Overview slide set
  - ADT Services Top level Users view.pdf: Slide set about different relationships between contracts

References

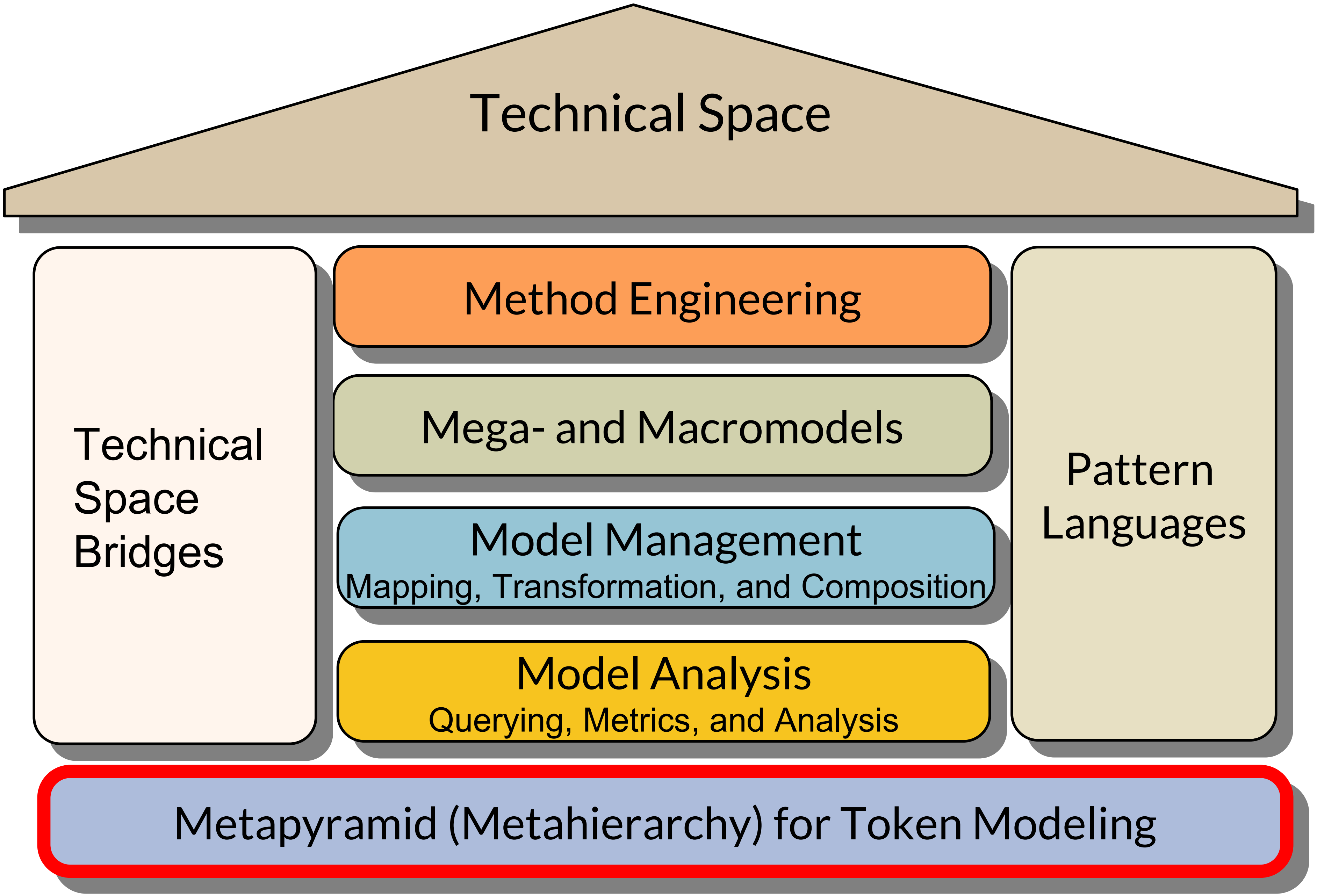
- [Vered Gafni] Presentation Slides about the Heterogeneous Rich Component Model (HRC).
- [CSL] The SPEEDS Project. Contract Specification Language (CSL)
    - http://www.speeds.eu.com/downloads/D_2_5_4_RE_Contract_Specification_Language.pdf
- G.Gößler and J.Sifakis. Composition for component-based modeling. Science of Computer Programming, 55(1-3):161–183, 2005.

Other Literature

- Informatik Forum http://www.infforum.de/
- Data-Flow Diagrams:
  - De Marco, T.: Structured Analysis and System Specification; Yourdon Inc. 1978/1979. Siehe auch Vorlesung ST-2
  - McMenamin, S., Palmer, J.: Strukturierte Systemanalyse; Hanser Verlag 1988
- Workflow languages:
  - ARIS tool (IDS Scheer, now Software AG)
    - http://en.wikipedia.org/wiki/Architecture_of_Integrated_Information_Systems
- Big CASE IDE
  - MID Innovator (insbesondere für Informationssysteme)
    - http://www.modellerfolg.de/
  - MagicDraw http://www.nomagic.com/

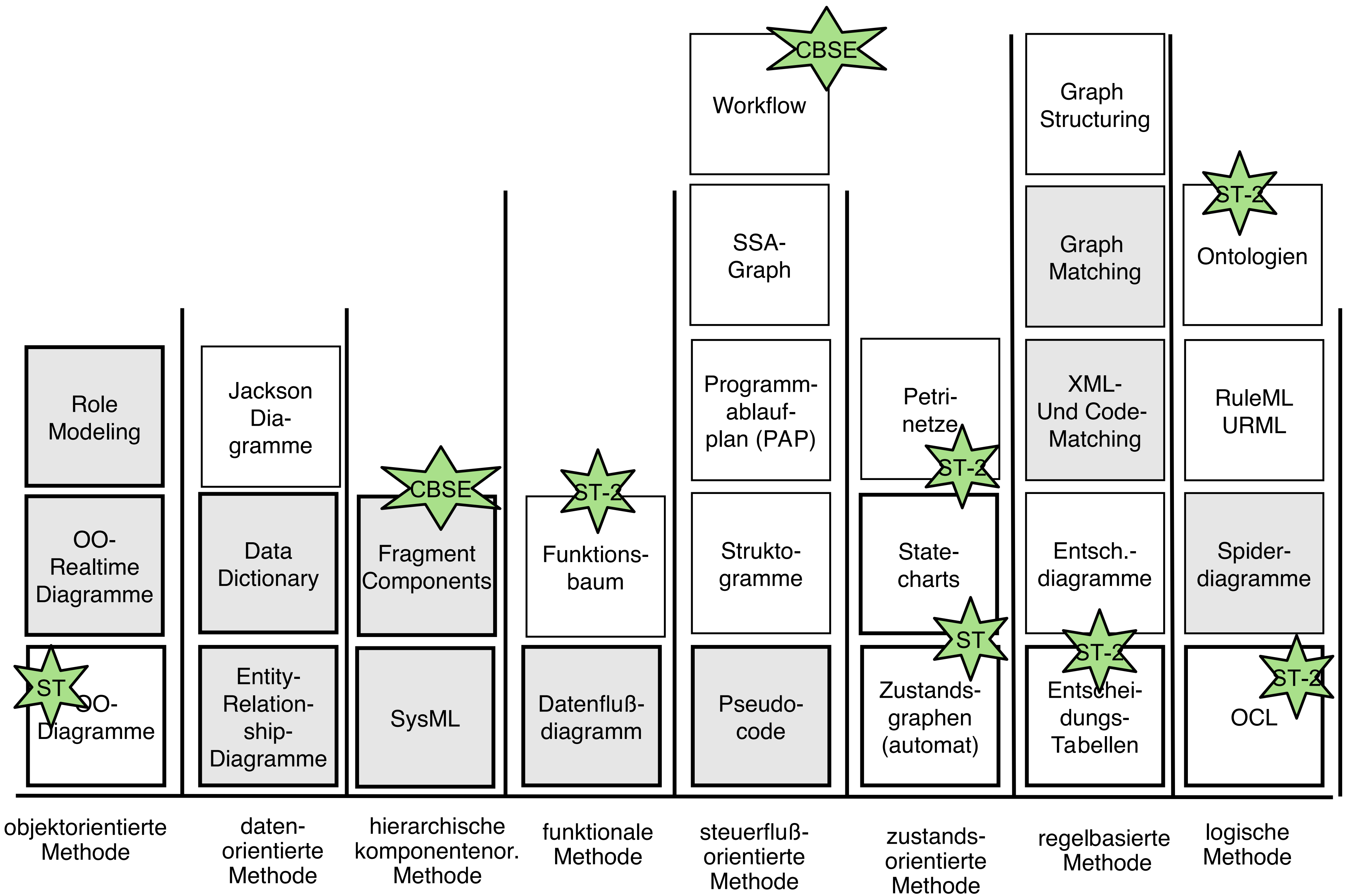
14.1 Basic Techniques of Software Engineering, Language Families, and Tool Composition

Q10: The House of a Technical Space



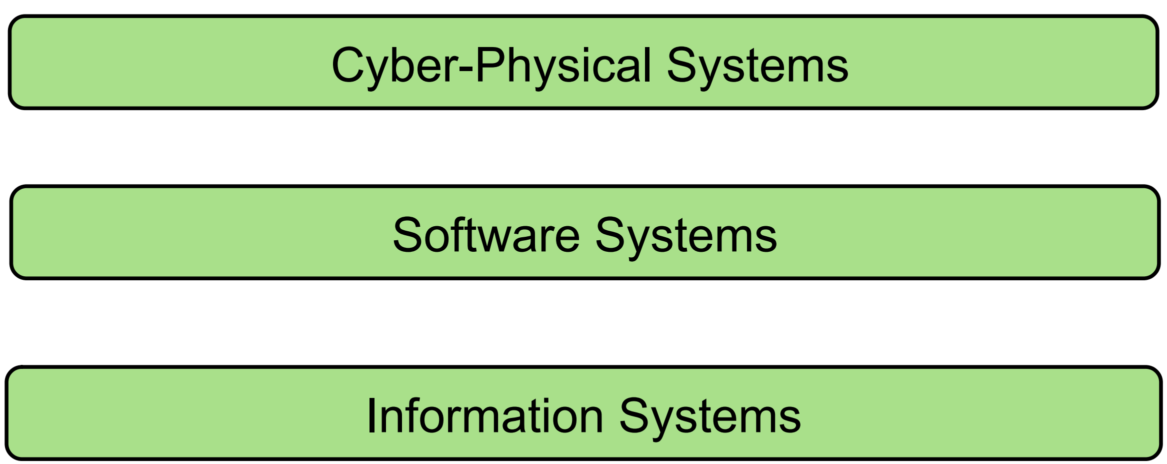
Basic Techniques and Languages for Modeling

Derived from [Balzert]



Metamodel Layering for Language Families

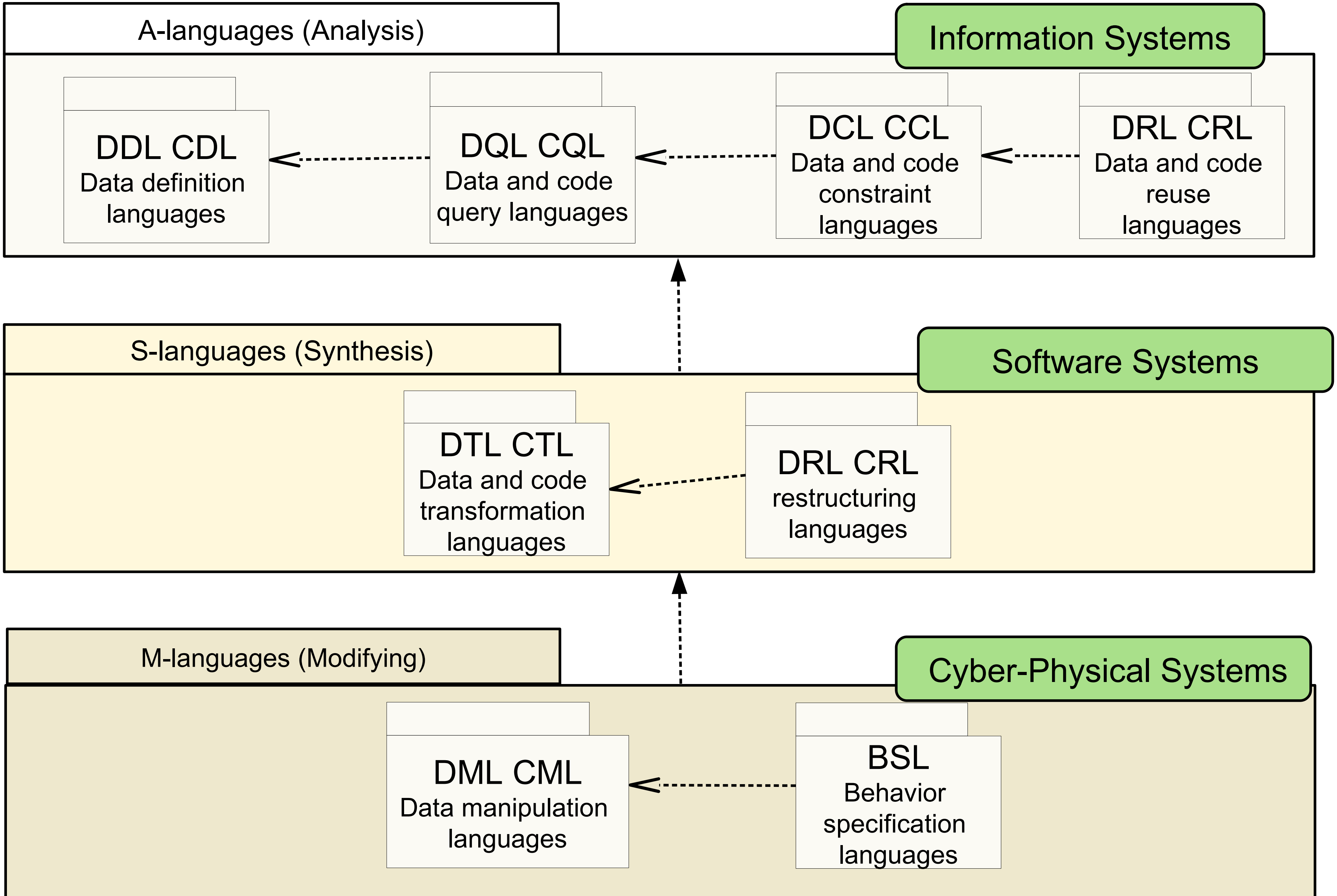
- M2 can systematically be divided into **M2 layers**
  - with metamodel packages in **language families** result:
  - Language engineering by composition
  - Tool construction by composition
  - Basic technique composition from several languages
  - Method engineering by method composition of basic techniques
- Productivity of Process
- Reliability of Software
- Different forms of Systems need different M2 layers:



CPS and Simulation

Public domain
https://cdn.pixabay.com/photo/2017/05/14/20/11/simulator-2312973_960_720.jpg

Basic Language Families (Layer Structure of M2)



Basic Language Families (Structure of M2)

- In the metahierarchy, code covers M3-M0, because M0 is populated by objects of the dynamic semantics
  - Data does not have dynamic semantics, so it only covers M3-M1 (or M2-M0); however, when data is loaded as code, it changes its nature.
- **Data and code modeling with definition languages (DDL, CDL)**
  - DDL form the basic packages of M2 to be imported by all other packages
  - Ex.: MOF → UML-CD → UML-Statecharts
  - Ex: lifted metamodels, such as EBNF-Grammars, Relational Schema (RS), Entity-Relationship-Diagrams (ERD), UML-CD, SysML-Component diagrams
- **Analysis languages (A-languages):**

Information Systems

  - Queries with **query languages** (DQL, CQL)
  - Consistency checking with data and code **constraint languages** (DCL, CCL) on wellformedness of data and code
  - **Reuse languages: Contract languages and composition languages**
    - Architectural description languages (ADL)
    - Template-Sprachen (template languages, TL) → course CBSE

Basic Language Families (Structure of M2) (ctd.)

Software Systems

- **Synthesis languages** (S-languages)
  - **Declarative Transformation Languages** (DTL, CTL)
    - Data flow diagrams (DFD)
    - Term- und graph rewrite systems
    - XML transformation languages
  - **Restructuring** (data and code restructuring languages, DRL, CRL)
    - **Wide Spectrum Languages** for refinement **(broadband languages, Breitbandsprachen)**
    - **Data exchange languages** (data exchange languages)

Cyber-Physical Systems

- Data and State **Manipulation Languages** (M-languages)
  - (non-declarative) Data manipulation languages (DML)
    - Workflow Languages, Petri Nets
    - Imperative languages
- Languages for **behavior specification language** (BSL)
  - Action-based state transition systems (finite automata and transducers)
  - Condition-Action-languages, Event-Condition-Action-languages (ECA)
  - → course Softwaretechnologie-2

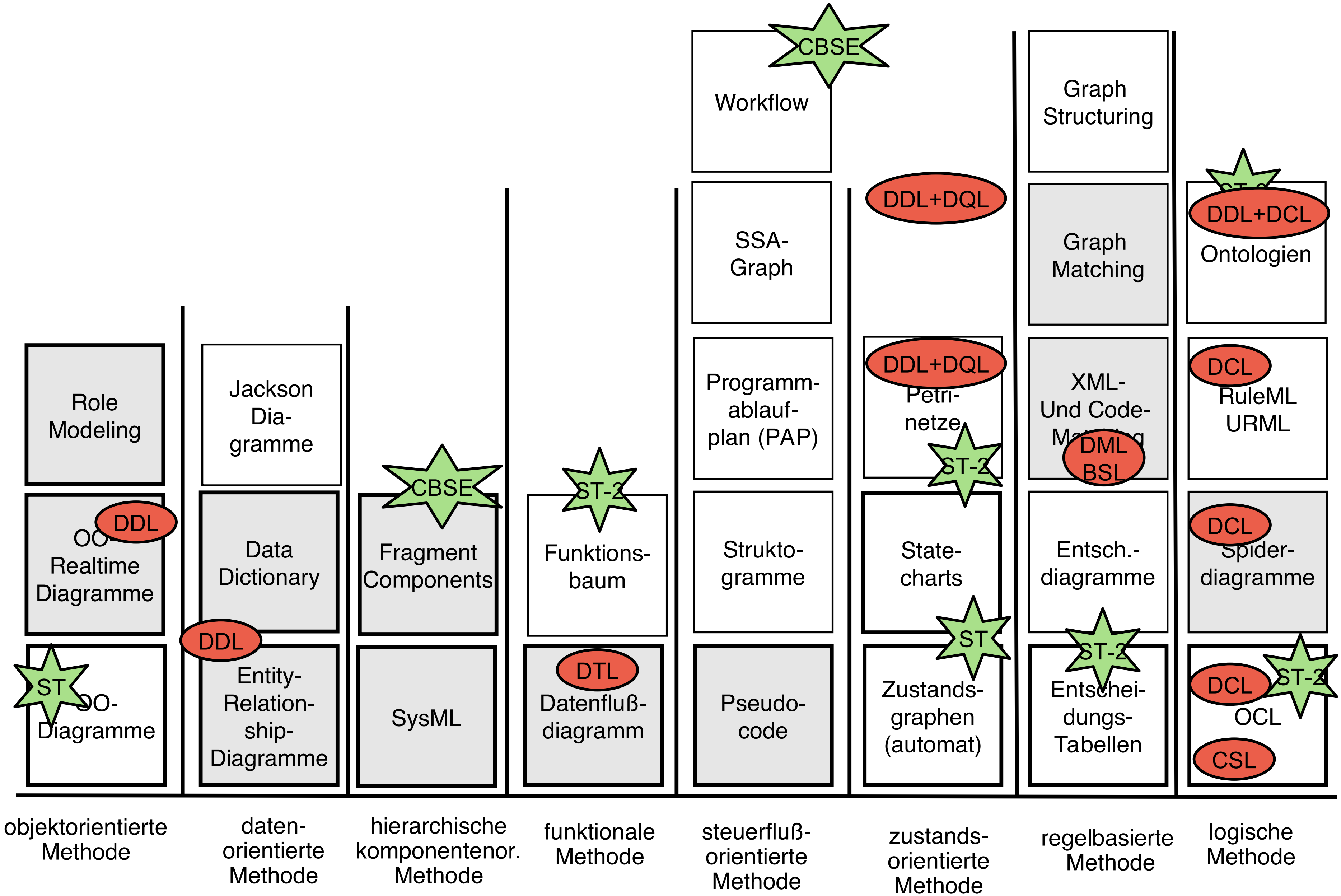
Software Engineering of Heterogeneous Systems

- A complex MDSD tool chain or Software IDE uses many base techniques and languages
- There is no homogeneous software construction
- Example: New electric car platform of VW and its design tool

How to compose tools for base techniques
to a large MDSD tool chain
or CASE IDE?

Basic Techniques and Languages for Modeling in this Course and Elsewhere

Derived from [Balzert]



14.2 Data Definition Languages (DDL) and Code Definition Languages (CDL)

The basic layer of M2

Usually lifted to M3 (i.e., self-descriptive)

All materials are shaped by a DDL or CDL

Data Dictionaries (Data Catalogues) as Basis for all Tools and IDE

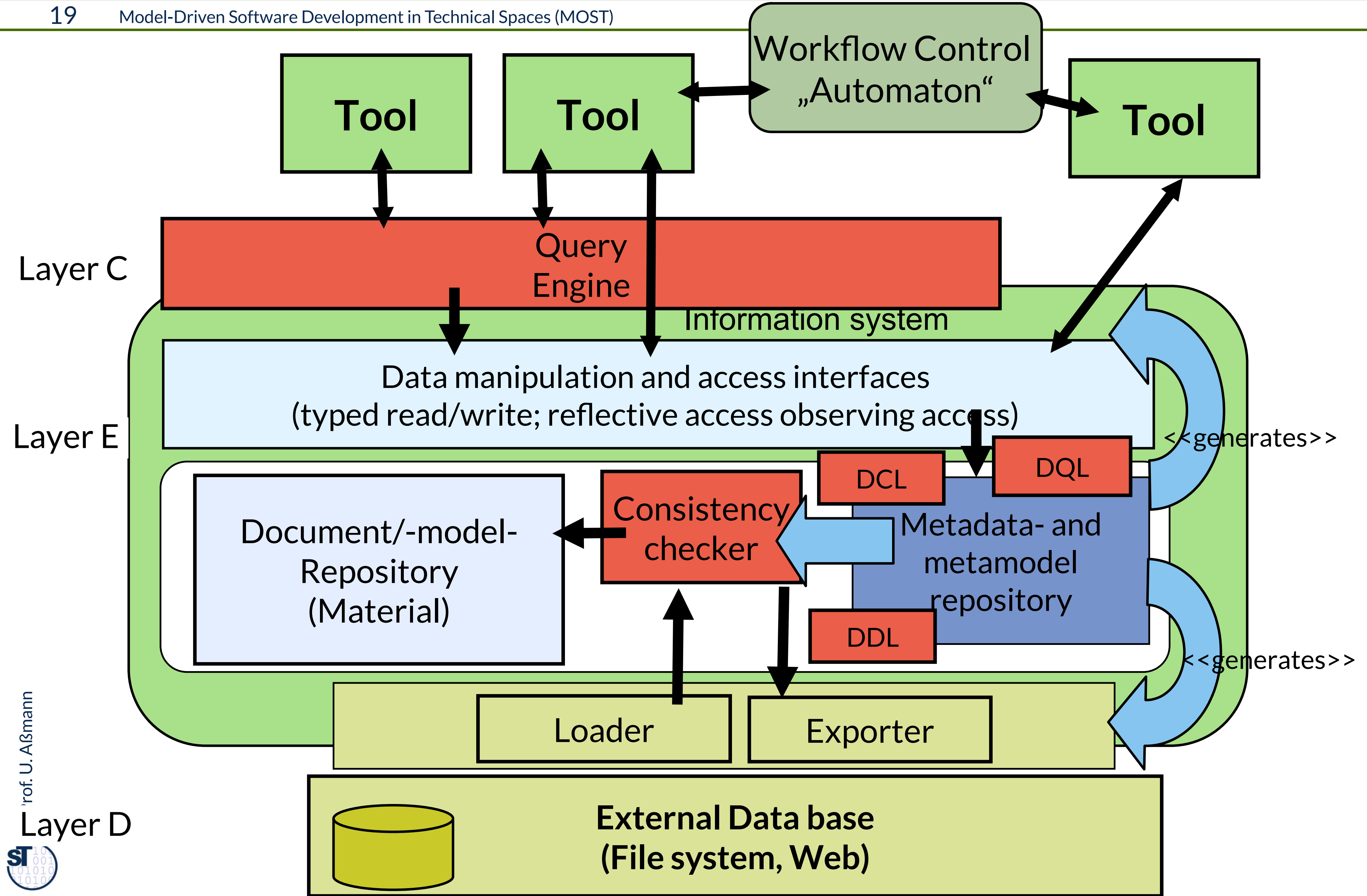
- A **data dictionary (data schema)** contains all types of data flowing through a system, including those stored in a repository (on M1)
  - Scope: local for an application, for several applications, for an entire company or even for a supply chain
  - A data dictionary is a special kind of model repository
  - If the data are models, it is called **metamodel repository**
- A **homogeneous data dictionary** is specified in a DDL
  - EBNF definies text languages (sets of text types)
  - Relational Schema (RS) defines relations and tabels
  - XML Schema (XSD) defines tree languages
  - ERD or UML-CD define graph languages
- A **heterogeneous data dictionary** is specified in several DDL
  - Usually, MDSD tool chains and Software IDE maintain heterogeneous metamodel repositories

Information Systems are based on DDL

- An **information system** is a software system conducting data analysis about a repository
    - Data warehouses, business intelligence, data analytics
- A **stream-based information system** is a software system conducting data analysis on a set of data streams

- Every software tool, every IDE relies on an information system
    - maintaining artefacts (data, programs, models, documents)
    - giving information about them
    - typed by the types in a data dictionary
- The data dictionary is described in a **data definition language (DDL)** or **code definition language (CDL)**
- The repository and the data streams are queried and analyzed by A-languages

Q7: Tool Architecture with Data Sharing in a Metamodel-Driven Repository



14.3 Query Languages (QL)

DQL – Data Query Languages

CQL – Code Query Languages

All materials are queried by technical tools shaped by a DQL or CQL.

DQL and CQL

- Querying
  - Pattern matching of structural patterns
  - Joining information
  - Reachability queries
- Metrics : counting of patterns
- Analysis: Deeper knowledge (implicit knowledge)
  - Program and model analyses on value and type flow

14.4 Constraint Languages (DCL,CCL) for Consistency Checking

All materials are constraint-checked by techncial tools shaped by a DCL or CCL.

Well-formedness of Models

A model is **well-formed (consistent)**, if it fulfils the context-sensitive constraints (integrity rules, consistency rules) of its metamodel.

- Wellformedness is checked by **semantic analysis (context analysis)**:
  - **Name analysis (Namensanalyse)** finds the meaning of a name
    - **Type analysis (Typananalyse)** finds the meaning of a type
  - **Type checking** checks the use of types with their definition
  - **Invariant checks**
    - **Range checks** (Bereichsprüfungen) test the validity of variables in ranges
    - **Structuring** of data structures: Acyclicity, layering, connected components, reducibility
  - **Forbidden combinations**
  - **Replicated definitions**

Well-formedness of Metamodels and Data Dictionaries

A **model** is **well-formed (consistent)**, if it fulfils the context-sensitive constraints (integrity rules, consistency rules) of its metamodel.

A **metamodel** is **wellformed**, if it fulfils the context-sensitive constraints of its metametamodel.

A **data dictionary** is **wellformed**, if all contained models fulfil its context-sensitive constraints.

A **metamodel repository** is **wellformed**, if it fulfils all its context-sensitive constraints.

A **multimodel** is **wellformed**, if it fulfils all its context-sensitive constraints. Then it is called a **macromodel.**

Reuse Languages and Contract Languages

A **reuse language** is a (sub-)language) controlling the reuse of program or model elements.

Examples:

- **Contract languages** check whether components, modules, classes, procedures and methods are applied correctly
- **Component model definition languages** define reuse languages and contract languages [Johannes-PhD]

14.5 Data Transformation Languages (DTL)

Text, XML, Term, and Graph Rewriting

see separate Chapter

DTL and DML

- A DML (data manipulation language, Datenmanipulationssprachen) is used to transform data
- **Declarative DTL (Datentransformationssprachen, DTL)** consist of declarative rule systems transforming a repository
  - Term rewriting for trees, terms, link trees, and XML trees
  - Graph rewriting for graphs
- **Imperative DML (general DML)** know states and side effects.

Restructuring Languages (DRL)

- **Restructuring** means to transform while to retain invariants.
- A **restructuring language** is a DTL giving guarantees about the transformed materials.
- A **refactoring language** restructures code and retains some of its invariants
- Languages for **Refinement**:
  - Refinement means that a transformed program *implies the semantics* of the original
  - A **wide spectrum language** transforms programs by refinement, generating more and more versions *implying* the requirements specification (the original)

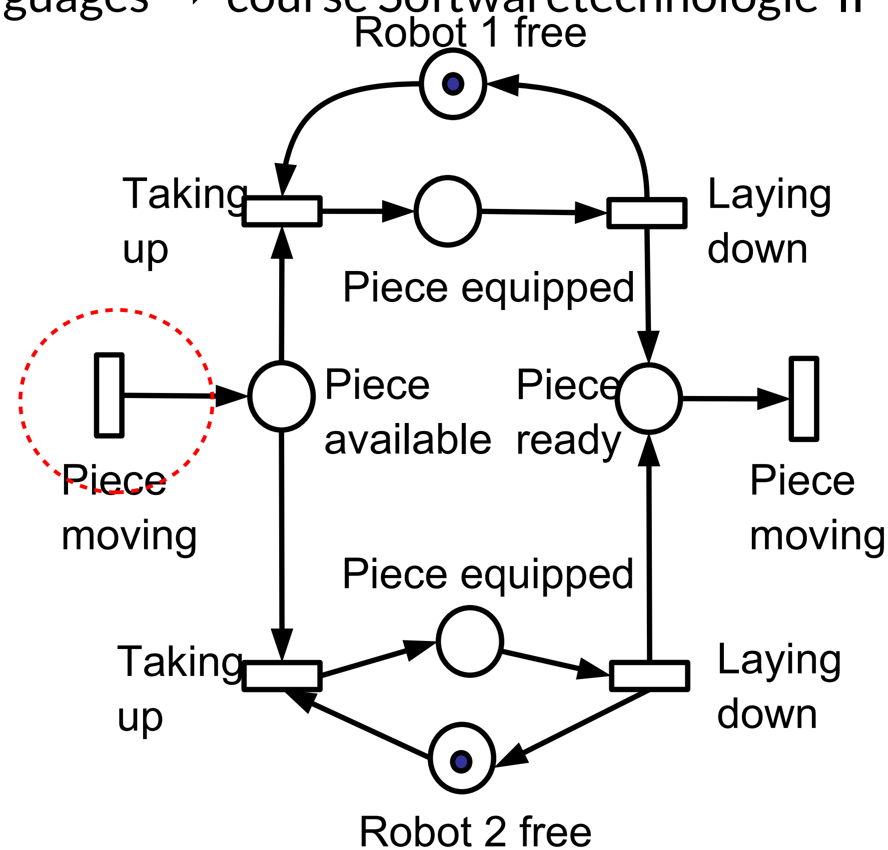
14.6. Behavior Specification Languages (BSL)

All automata (workflow engines) in a TS execute workflows written in a BSL.

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

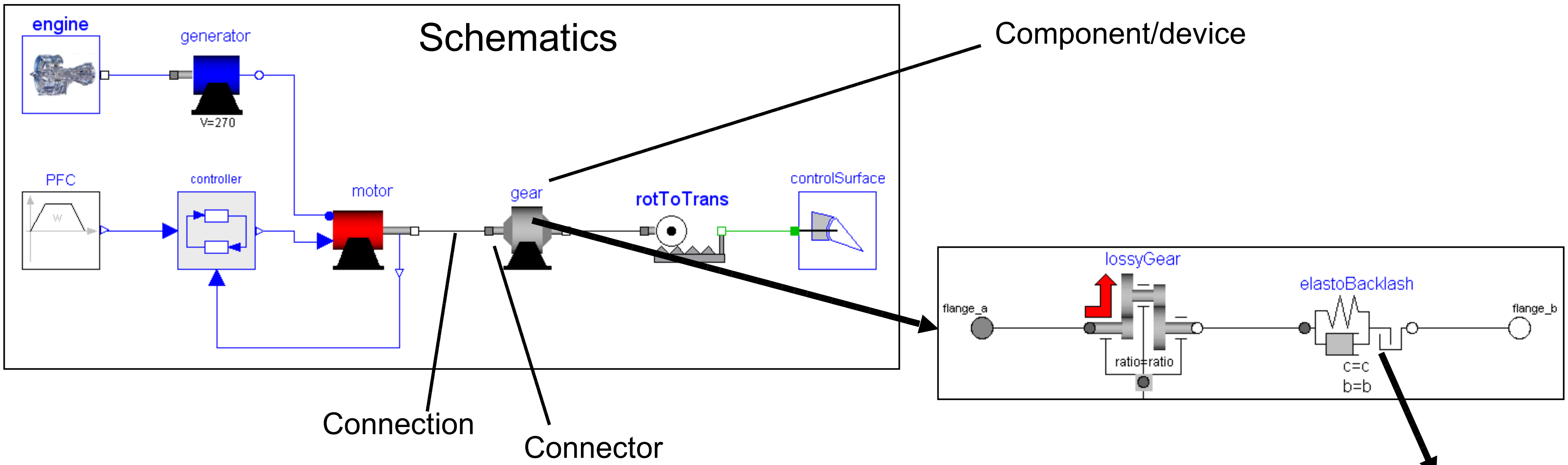
Automaten, Petri-Nets, DFD and Workflow Languages

- **State-oriented Behavior specification languages** enable the specification of **interpreters (operational dynamic semantics)** and **simulators** (interpreters with a specific platform)
- Automata, Transducers, Statecharts → course Softwaretechnologie-I
- DFD, Petri-Nets and Workflow languages → course Softwaretechnologie-II



[Balzert]

Modelica Users View




```
equation
        0 = flange_a.tau + flange_b.tau;
  phi_rel = flange_b.phi - flange_a.phi;
    w_rel = der(phi_rel);
   ...
```

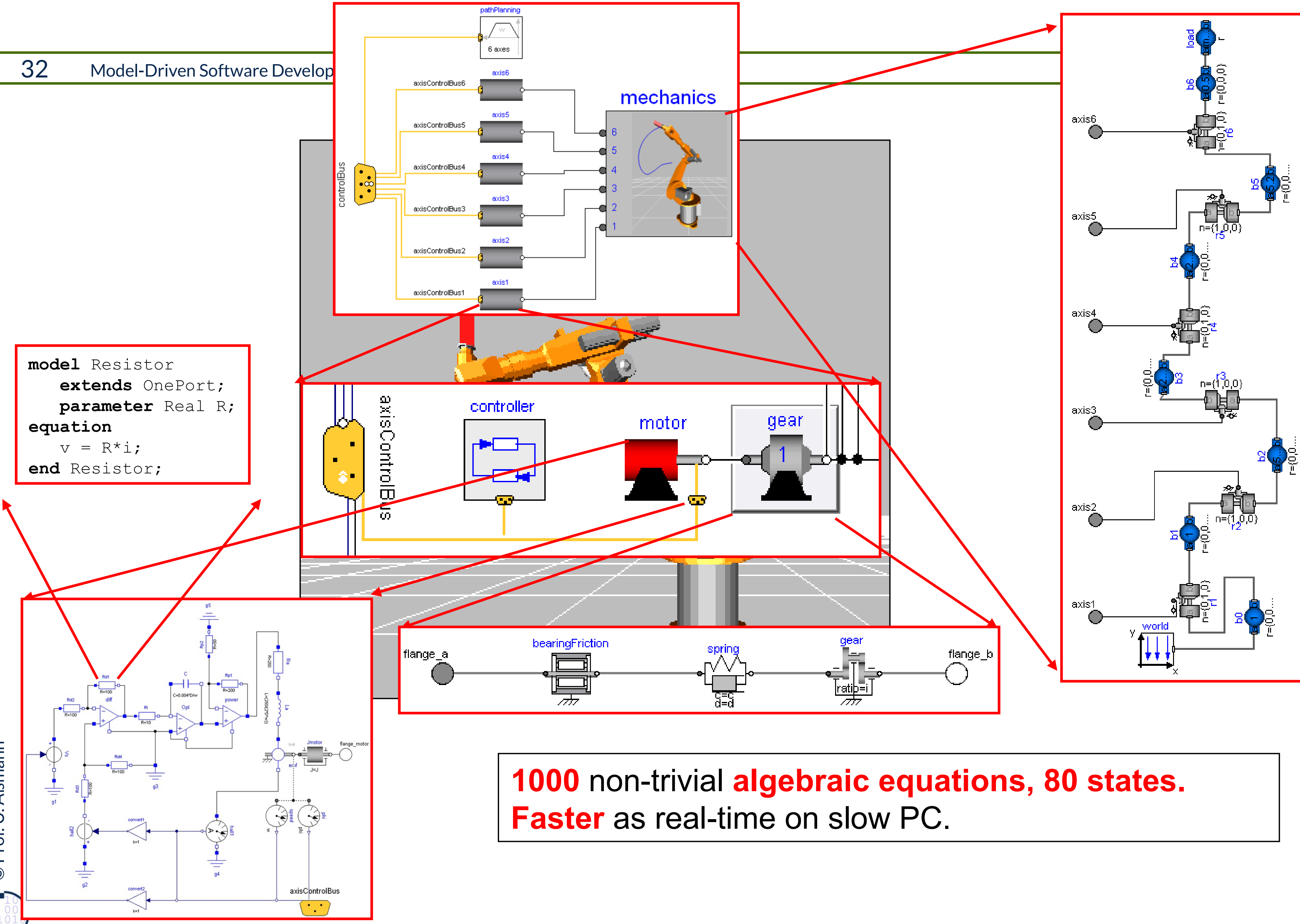
- Each **Icon** represents a **physical component**. (electrical resistance, mechanical device, pump, ...)
- A **connection line** represents the actual physical **coupling** (wire, fluid flow, heat flow, ...)
- A component consists of **connected** sub-components (= hierarchical structure) and/or is described by **equations**.
- By **symbolic** algorithms, the high level Modelica description is transformed into a set of explicit differential equations:

$$0 = \mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), t)$$

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

Example: Industrial Robots (from Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot)

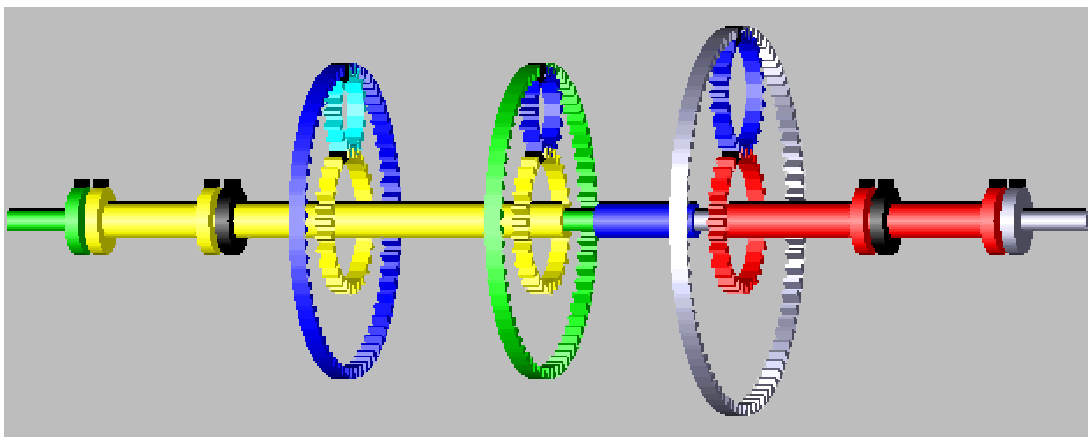


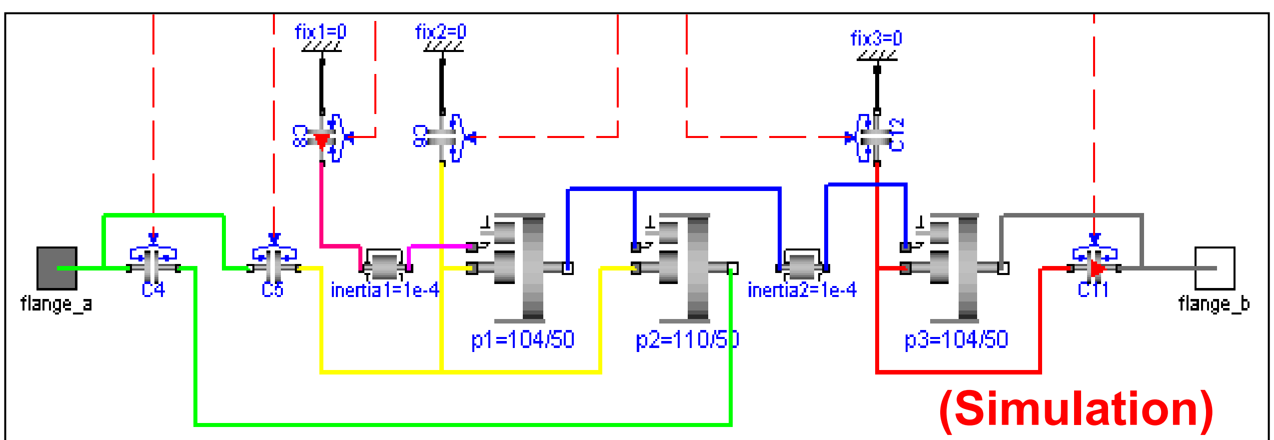

```
model Resistor
   extends OnePort;
   parameter Real R;
equation
   v = R*i;
end Resistor;
```

1000 non-trivial **algebraic equations, 80 states.**
Faster as real-time on slow PC.

Example: Hardware-in-the-Loop Simulation of automatic gear boxes (different vehicle manufacturers)

Electronic Control Unit
(Hardware)





+ driver + engine
+ 1D vehicle dynamics

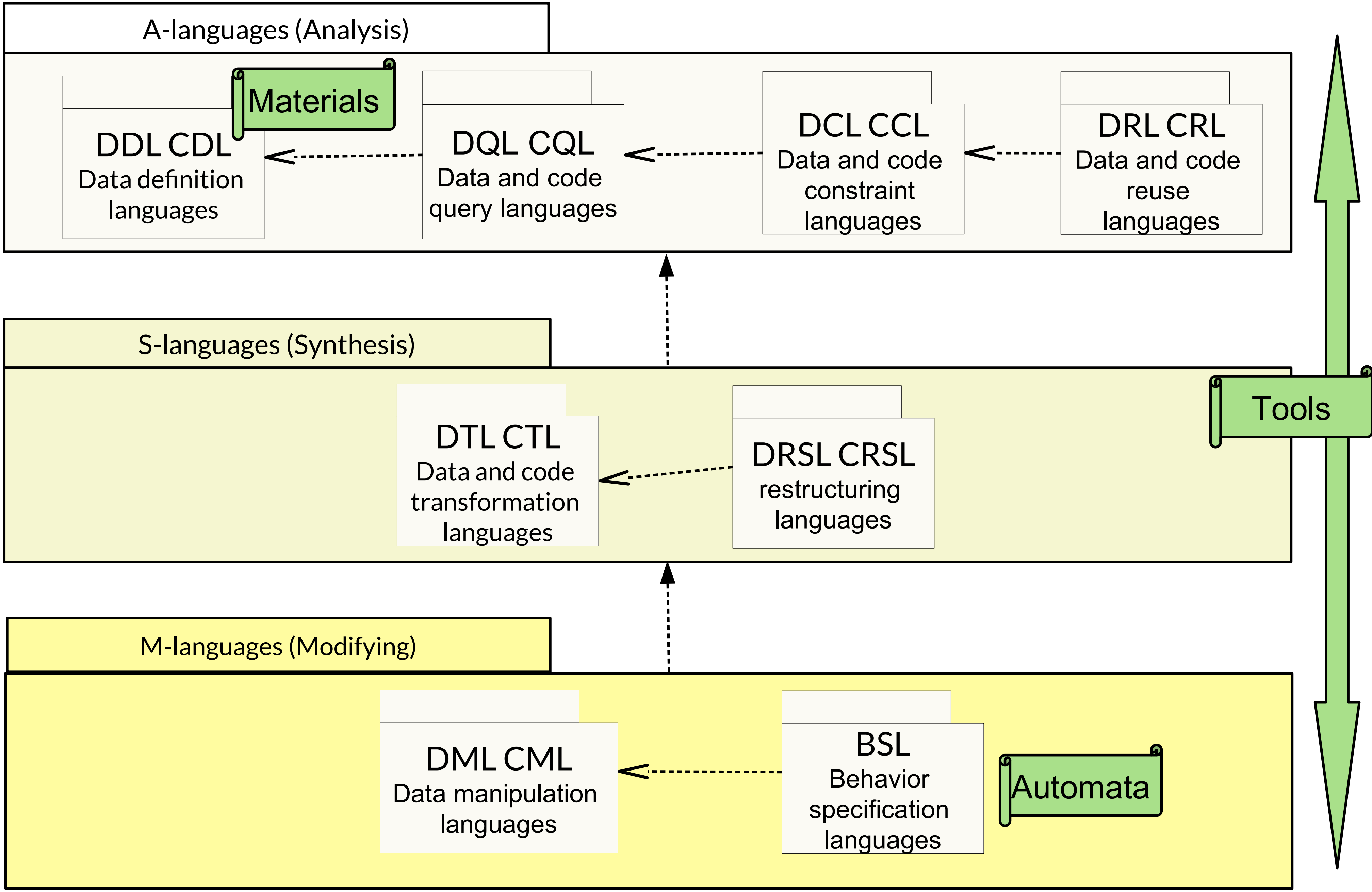
14.7 Examples of Language Families on the M2 Layers

Every technical space has a language hierarchy on M2 with a similar, layered structure.

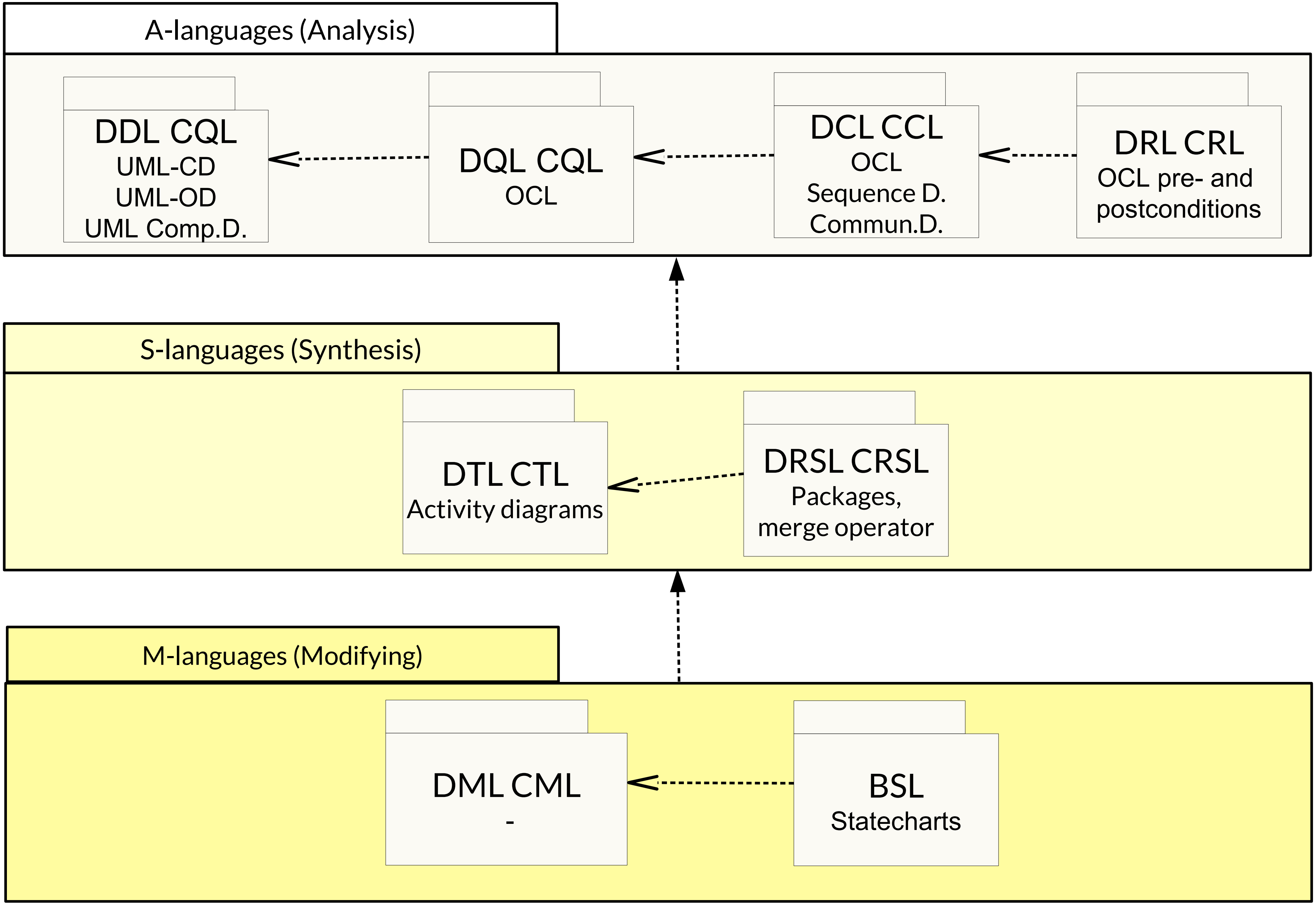
All tools have an underlying language family.

Every IDE has an underlying language family.

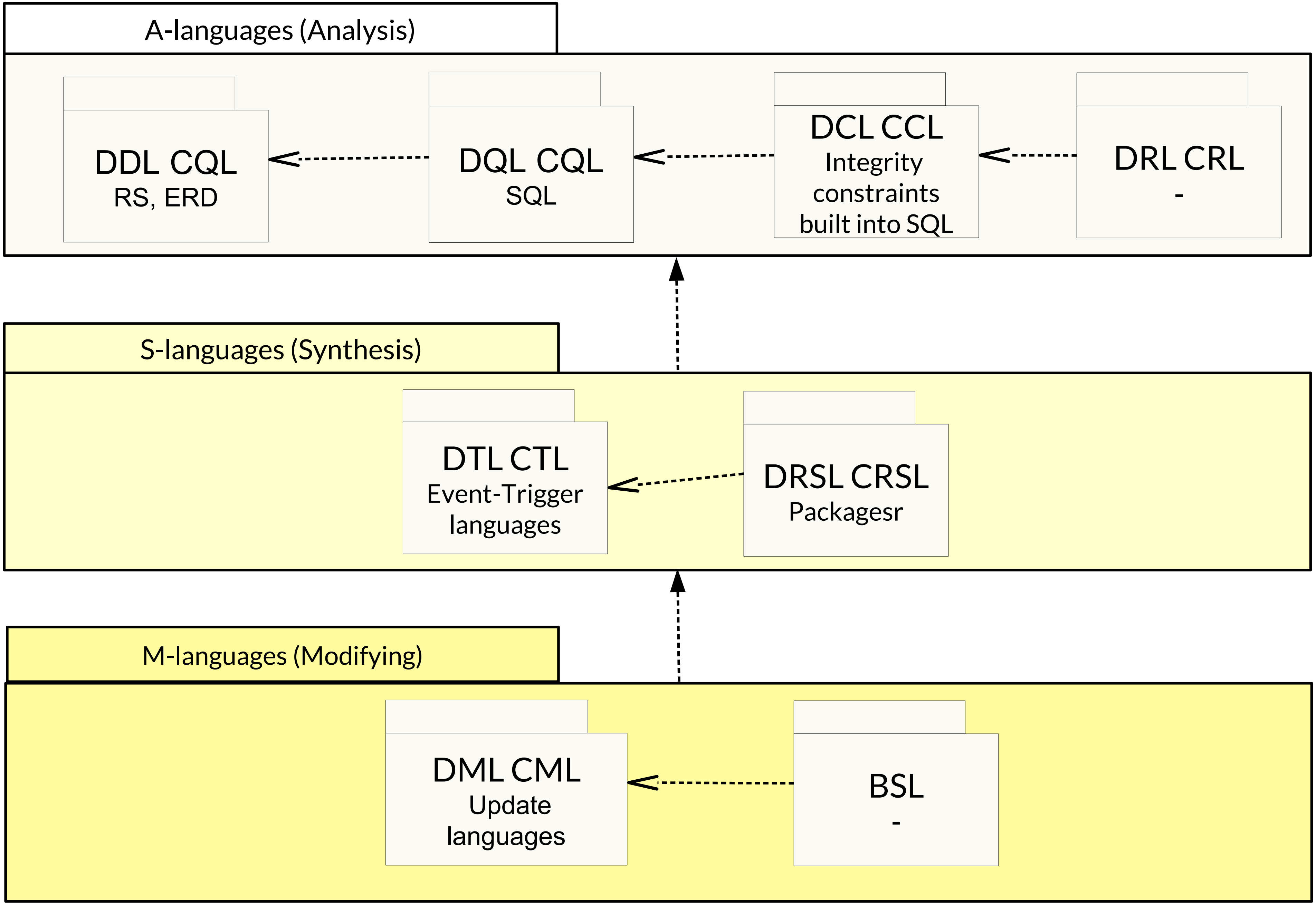
Basic Language Families (Layer Structure of M2)



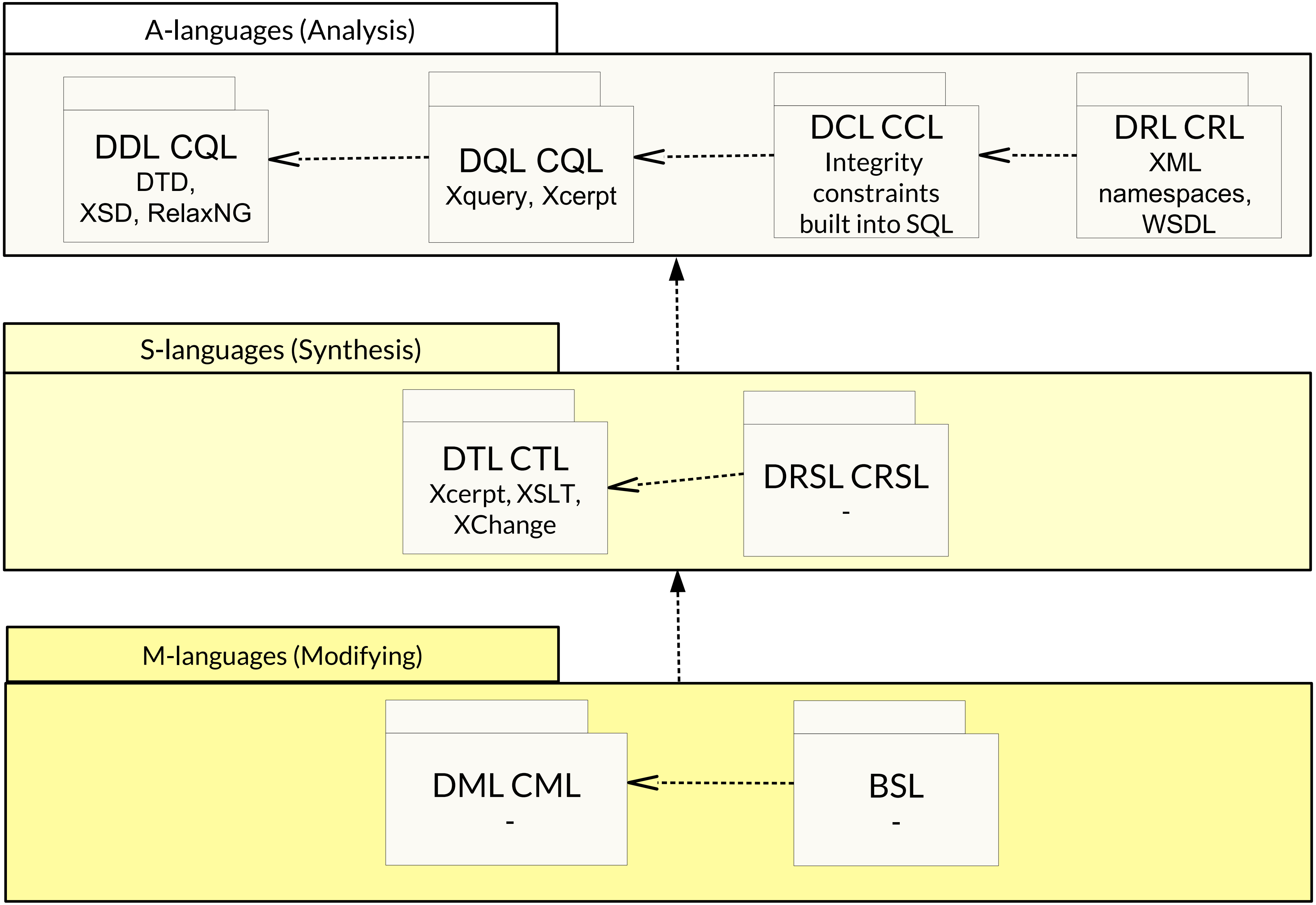
UML Language Family in the ModelWare TS



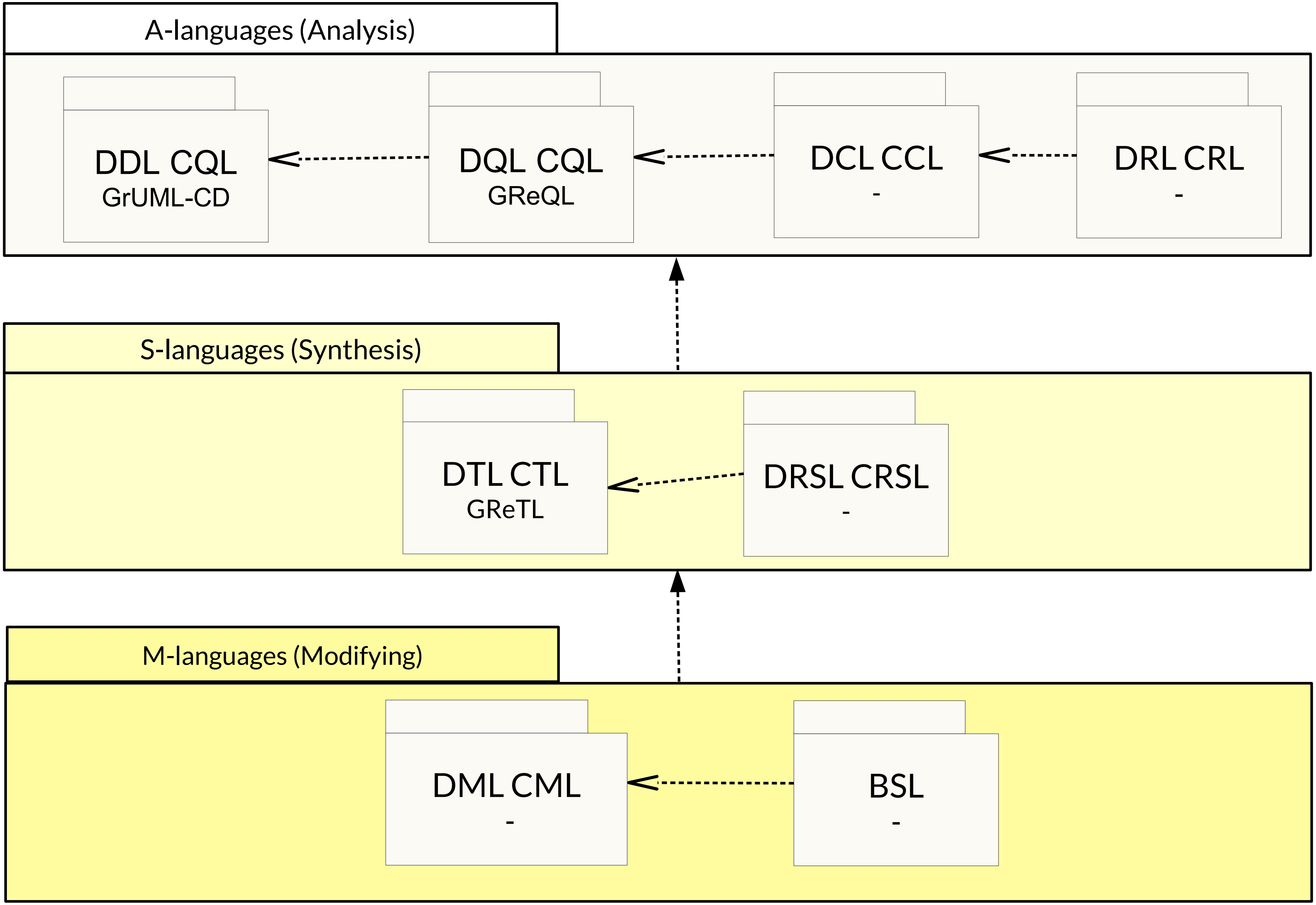
ERD/RS Language Family in the Relational TS



XML Language Family in the Link Tree TS



GrUML Language Family [Ebert]

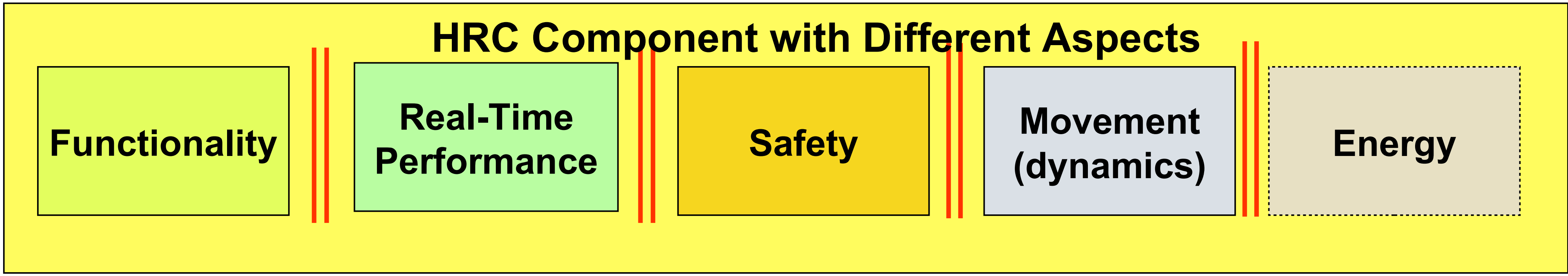


14.8. ... and all together now...

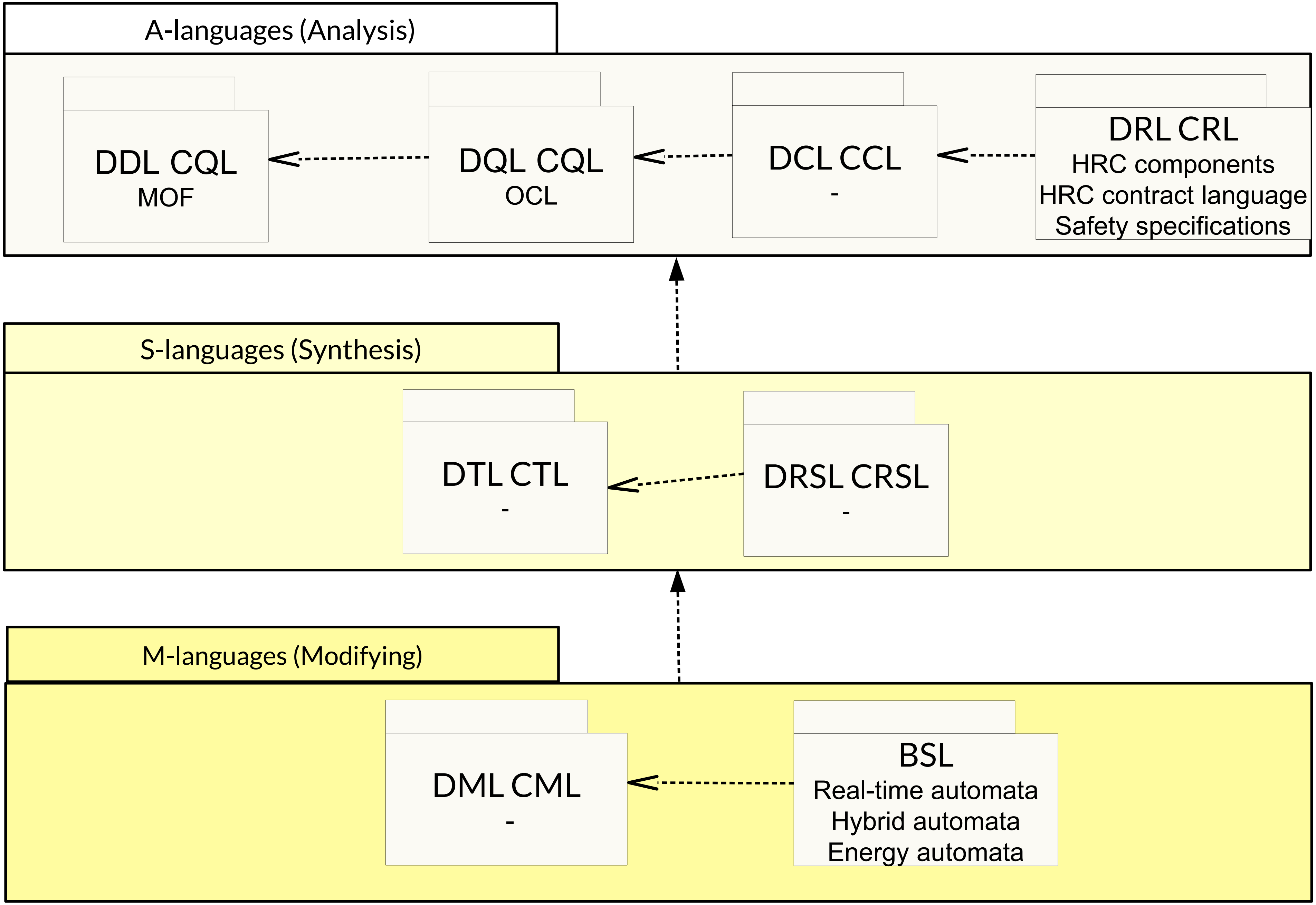
Example Megamodel: Composition of Contracts in the HRC (Heterogeneous Rich Components) MDSD Tool Chain for Complex Embedded Systems

[Vered Gafni]

- Within a HRC component, contracts *in different views* can be synchronized (synchronized token-based modeling)
  - The real-time assertions can be coupled with functional, real-time, safety, physical movement (dynamics), and energy view
  - Every contract has a different contract language
- Between different components, the contracts of a certain viewpoint can be composed and checked (viewpoint-specific modeling)



Example Megamodel: HRC Language Family for Safety-Critical Embedded Software



Why is it Important to Know about the M2-Layers?

The megamodels managed by MDSD Tool Chains and Software IDE very often combine different languages from several layers of M2 (**M2-Mix**)

- ERD – MOF – XSD – UML-CD
- Xquery – XSLT – SQL – SPARQL
- OCL – SpiderDiagrams – OntologyLanguages
- Java – C++ – C#
- Petrinets – DFD – WorkflowNets – BPMN

Domain-specific languages always consist of an M2-Mix

Basic techniques (Basistechniken) also

Methods also

Why is it Important to Know about the M2-Structure?

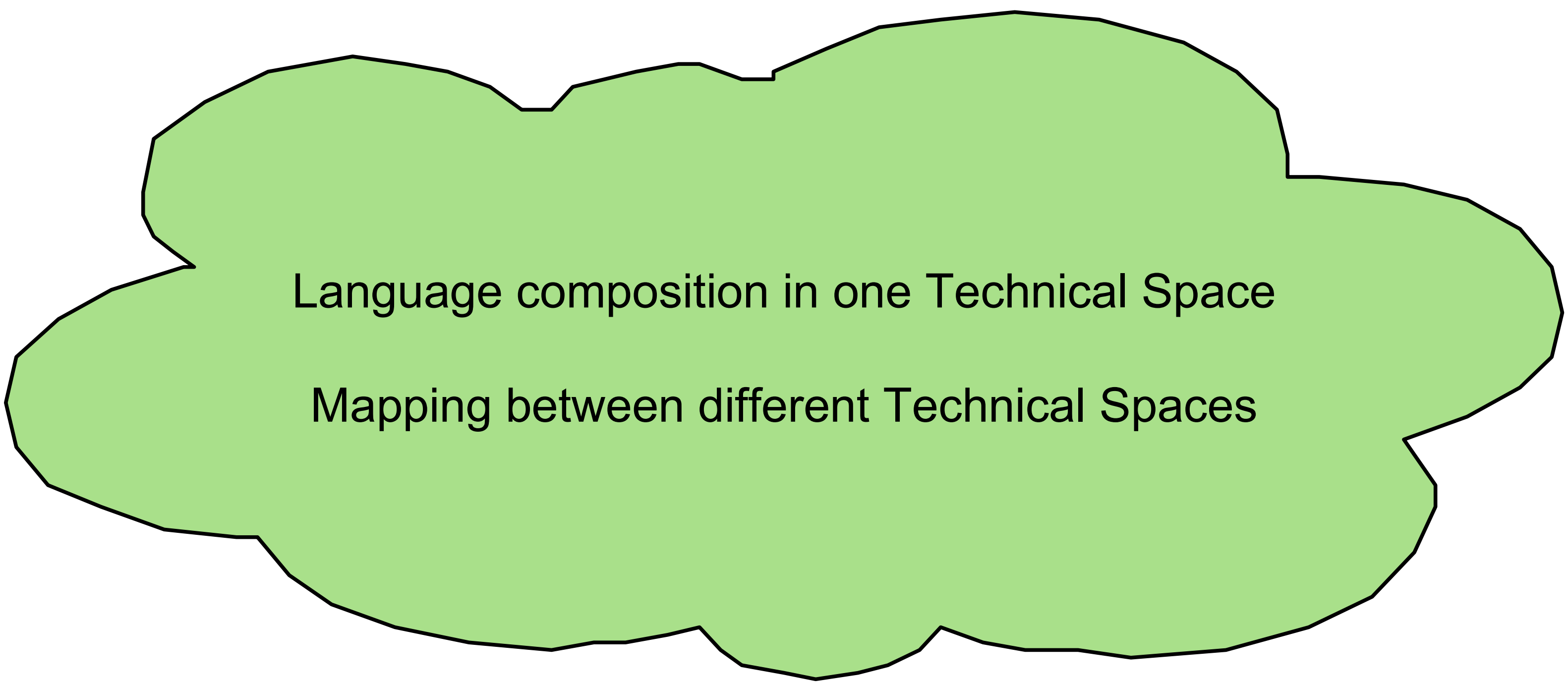
How can we compose metamodels for tool composition?

- Language families can be arranged in M2 layers
  - Many languages on upper layers can be composed with languages on lower layers
- If everything is in one Technical Space, composition of tools relies on the composition of languages
  - For that we need Model Composition Systems (forthcoming, → course CBSE)
    - Example: UML-Package Merge-Operator

Language composition: Compose new language constructs from layers further down

How Can We Compose Tools for Base Techniques for MDSD Tool Chains and Software IDE?

- If we have to treat several Technical Spaces, Bridges between TS have to be built



The End – Exam Questions

- Where would you position a query tool on M1 – in the tools layer or in the materials layer? Does the tool's metamodel in its query language belong then to the tools metamodel layer of M2 or to the materials metamodel layer?
- Why can we compose different DQL with a given DDL?
- How is it possible to apply a graph query language on XML trees?
- Why is UML such a complex language?
- A MDSD tool chain such as the HRC IDE for embedded systems works with many languages in different technical spaces. Explain some ingredients of such a complex IDE.