Atom Free IT

SOFTWARE AND SUSTAINABILITY
S2 RESEARCH GROUP
VRIJE UNIVERSITEIT AMSTERDAM

Methodical conversion from text to model

Bootstrapping modeling with MuDForM

robert.deckers@AtomFreeIT.com
+31646882428

20221119 v1

Your speaker in a nutshell

- Expertise, focus:
  (Domain) Modelling, Model Driven Development,
  (Non-functional) Requirements,
  Method engineering,
  Architecture:
    Aspect oriented design (patterns)
    Research, innovation
    System, software, information, enterprise
    Coaching, consultancy, training

- Some customers:
  Canon, HTI, ASML ,UT, TUE, UvA,
  BMW group, COA, KvK, Ohra, Rabobank, NXP, UVIT,
  CRV, SNS, Reaal, Delta Lloyd, Interpolis, APG, NS, VBI,
  Shell, ECT, Fortis, Vlisco

- Jobs:
  Software engineer: Dutch Army -1994,
  Software architect, method engineer: KISS b.v. –1999,
  Senior scientist architecture: Philips Research -2006,
  Modeling architect: Philips Healthcare -2007,
  Principal architect: Sogeti -2013,
  Consultant, coach, trainer, method engineer:
  Atom Free IT, PhD VU Amsterdam -today

- Education:
  TUE informatica -1991,
  EngD Software Technology -1993

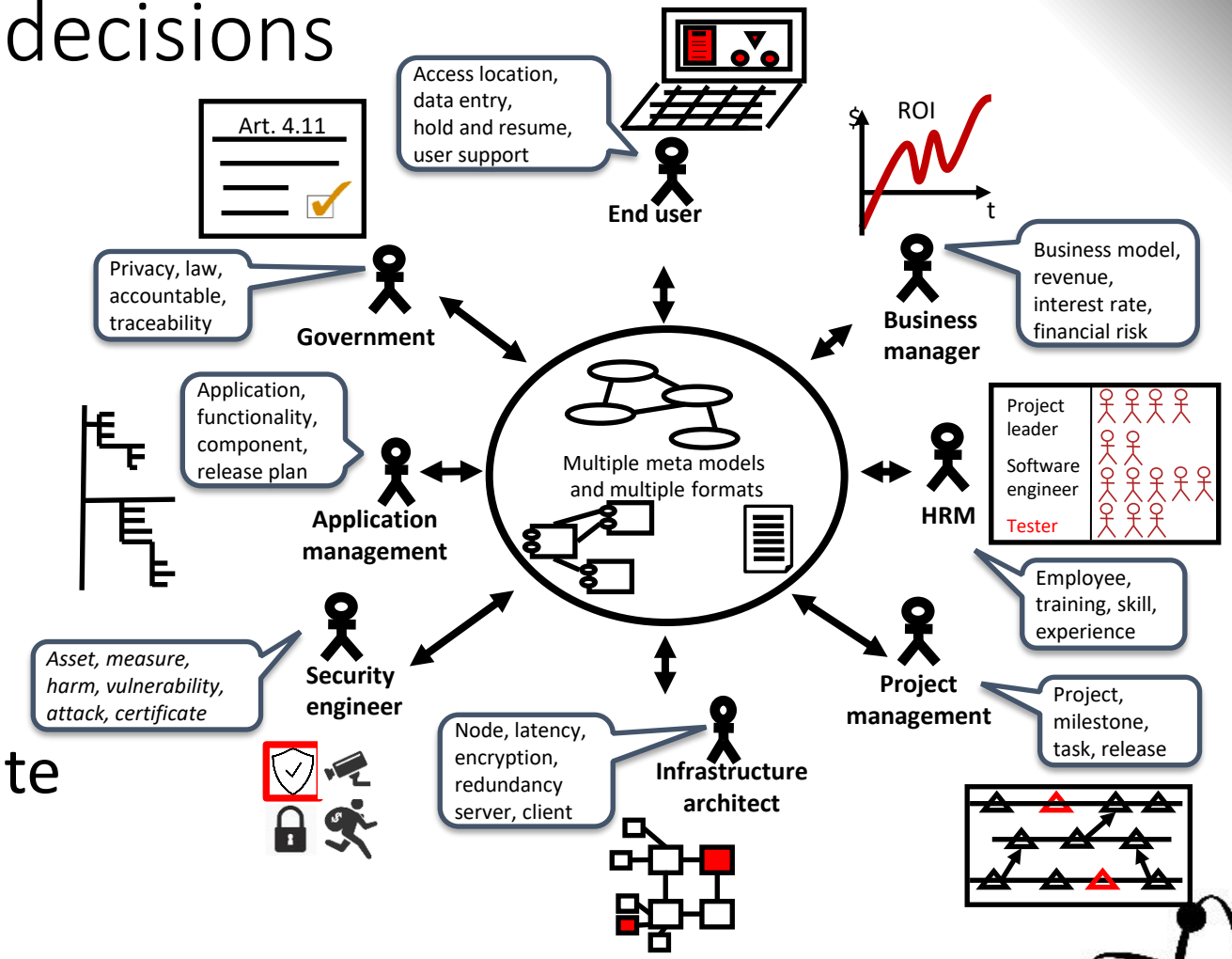
Whatever...
Just create

robert.deckers@AtomFreeIT.com
+31 6 46882428
https://www.linkedin.com/in/robertdeckers/

Software development is integration and transformation of human knowledge and decisions

- Support sharing and reusing decisions and knowledge
  (instead of source code and heads)

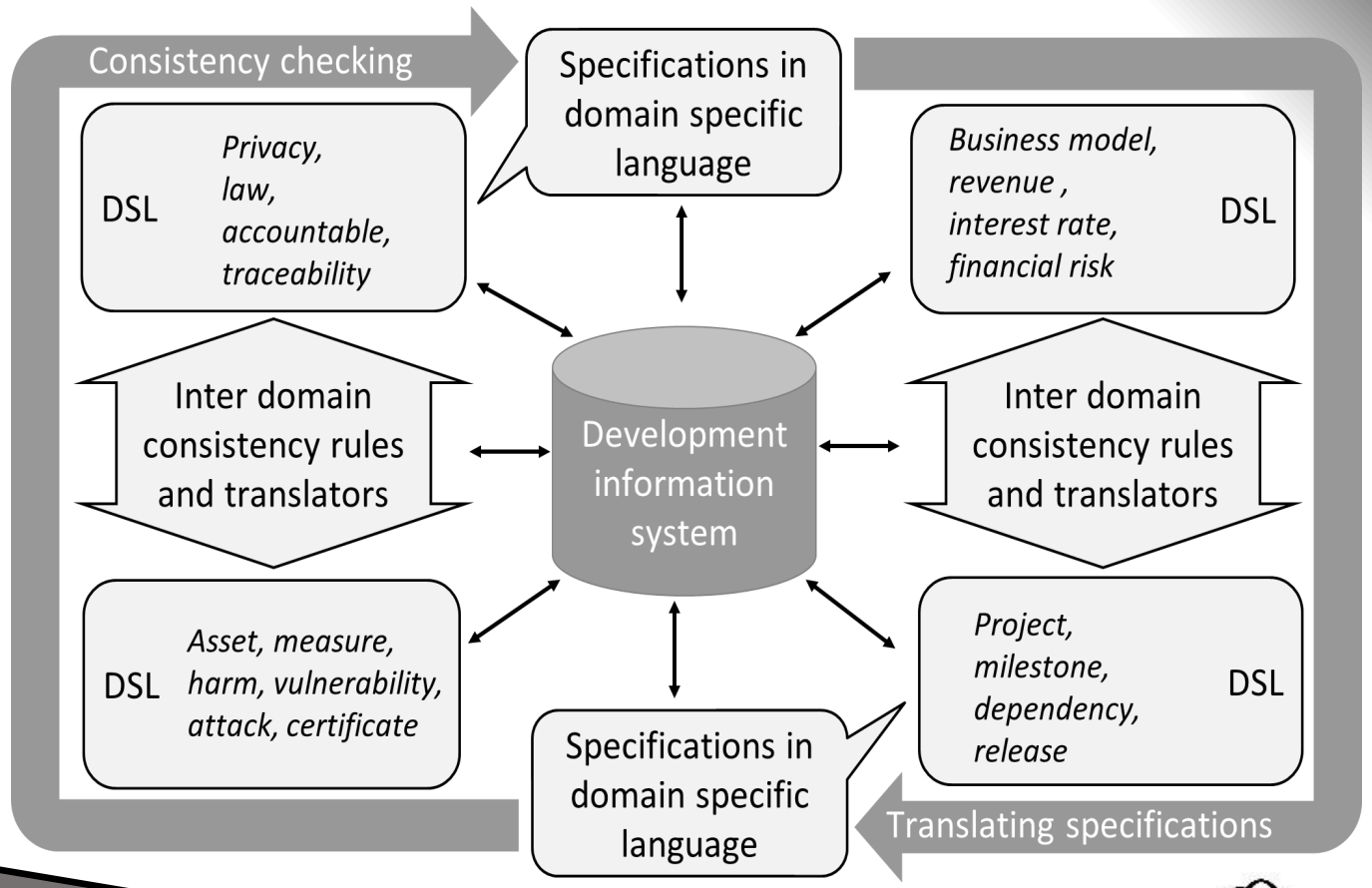
- People reason and communicate in their own language
  (instead of machine-based language)



Vision: MuDForM (Multi-Domain Formalization Method)

Challenges:

- Support for making specifications in terms of domain-specific languages
- Application and integration of multiple domains
- Integration of knowledge elicitation and model engineering
- Smart use and configuration of existing methods and tools



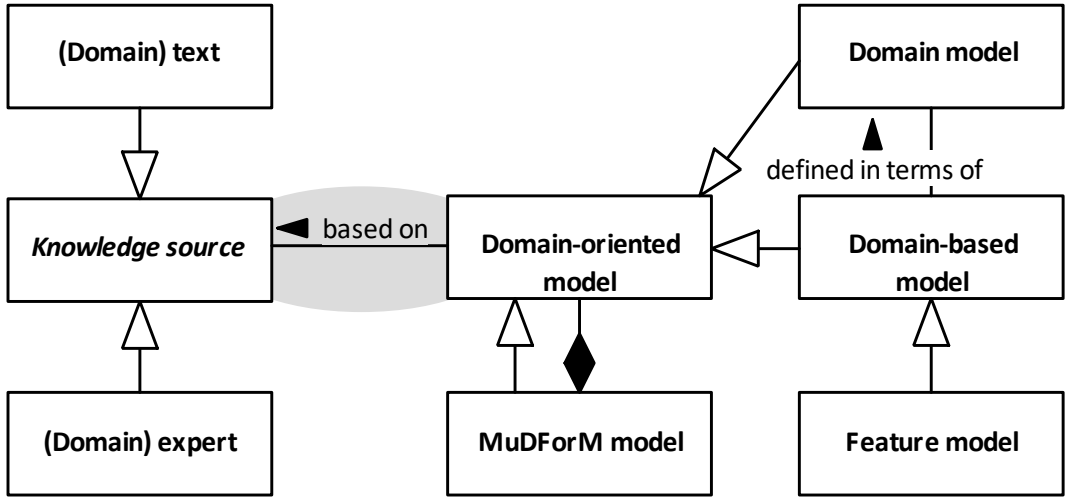
This story: Text to Model

Problem: Existing conversion methods are*

- Mostly just a language
  → lacking metamodel, steps, guidelines, and their integration.

- Limited to class diagrams
  → lacking support for (integrated) behavior, and for separation of domain, system, and context specifications.

- Have no explicit integration between text-to-model conversion and model engineering
  → loosing semantics in the conversion.



*Source: *Deckers R, Lago P. Systematic Literature Review of Domain-oriented Specification Techniques. Journal of Systems and Software 2022: 1–23. doi: 10.1016/j.jss.2022.111415*

Solution: Integrated and engineered method*

- Metamodel:
  - Integrate grammatical analysis concepts with modelling concepts

- Method steps:
  - For organizing and managing specification work
  - For the traceability of the model back to the input

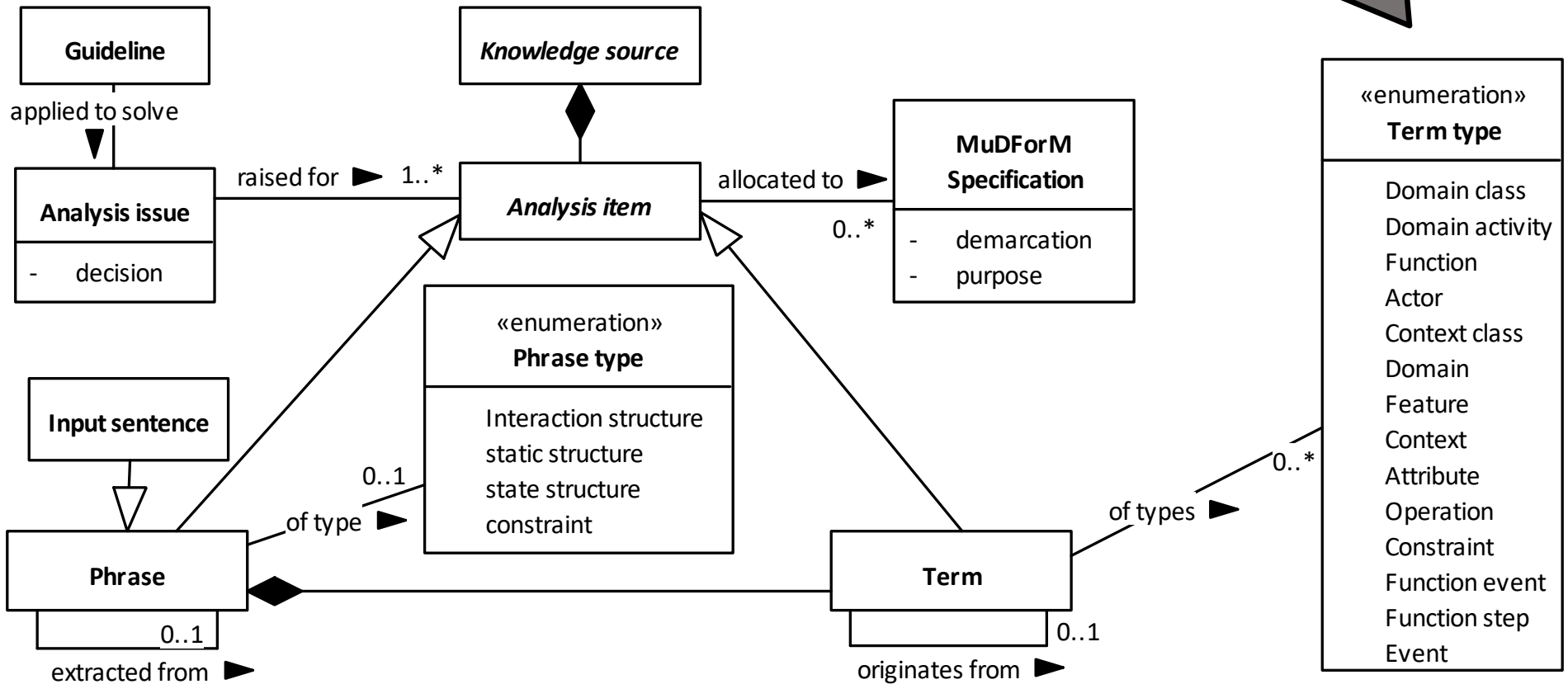
- Guidelines:
  - covering text to model conversion,
  - providing the foundation for model engineering

*MuDForM definition is available: https://github.com/robertdeckers/MuDForM

Metamodel:

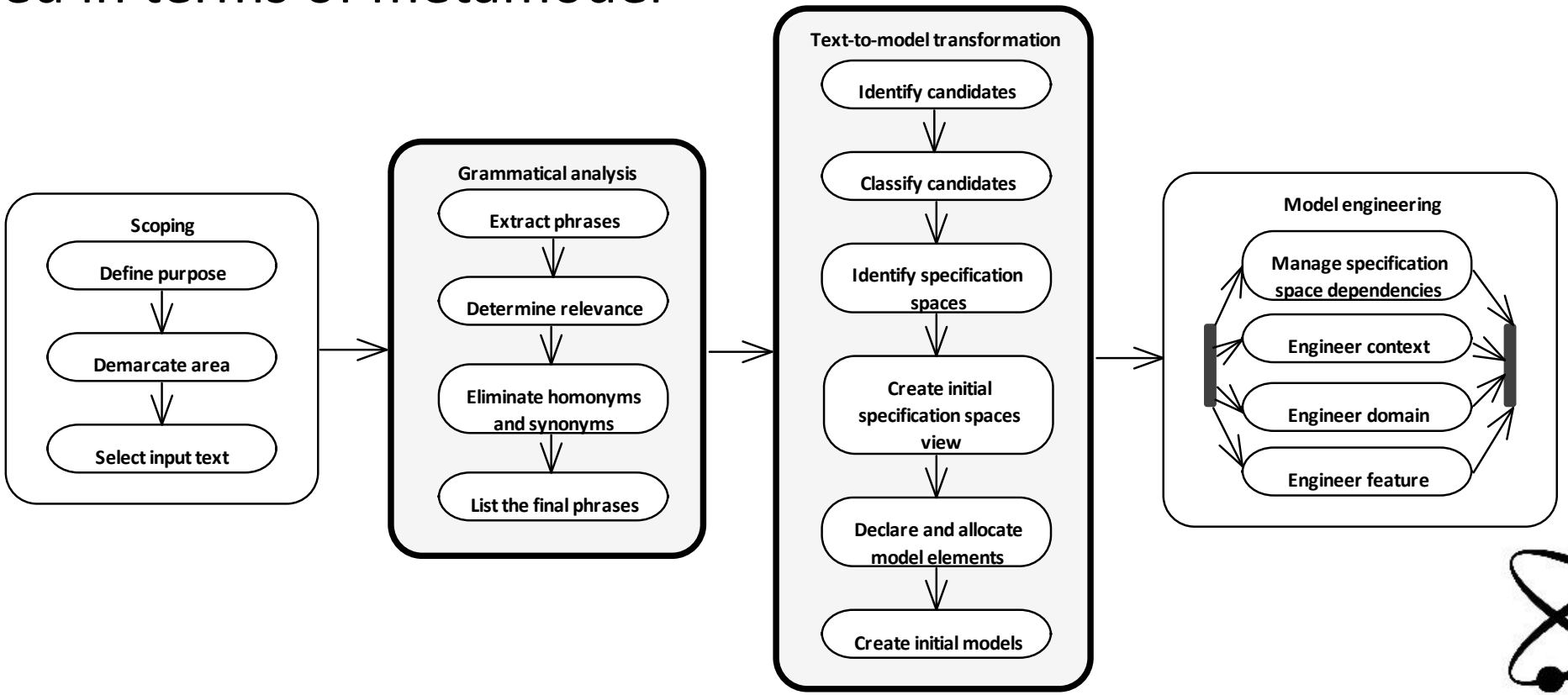
Specific Grammatical analysis concepts

Mapped onto modelling concepts



Method steps:

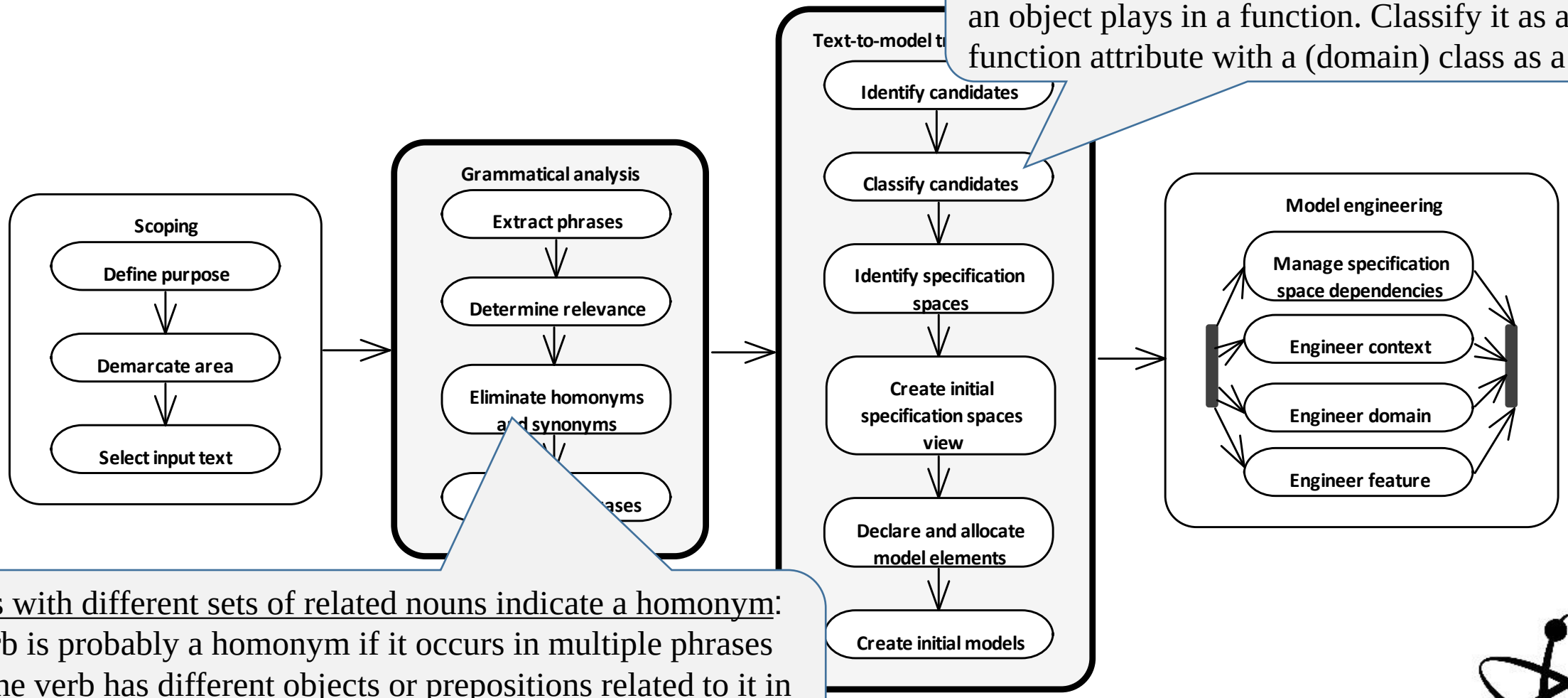
- Anchor for organization/management of specification work
- Covering phase from text to model, integrated in total method flow
- Expressed in terms of metamodel



Guidelines

- Allocated to method steps
- Some existing literature included
- Currently 35 text-to-model guidelines (of in total 130)
- Leading to an initial model,
  which is the entry point for the model engineering phase

Guideline examples

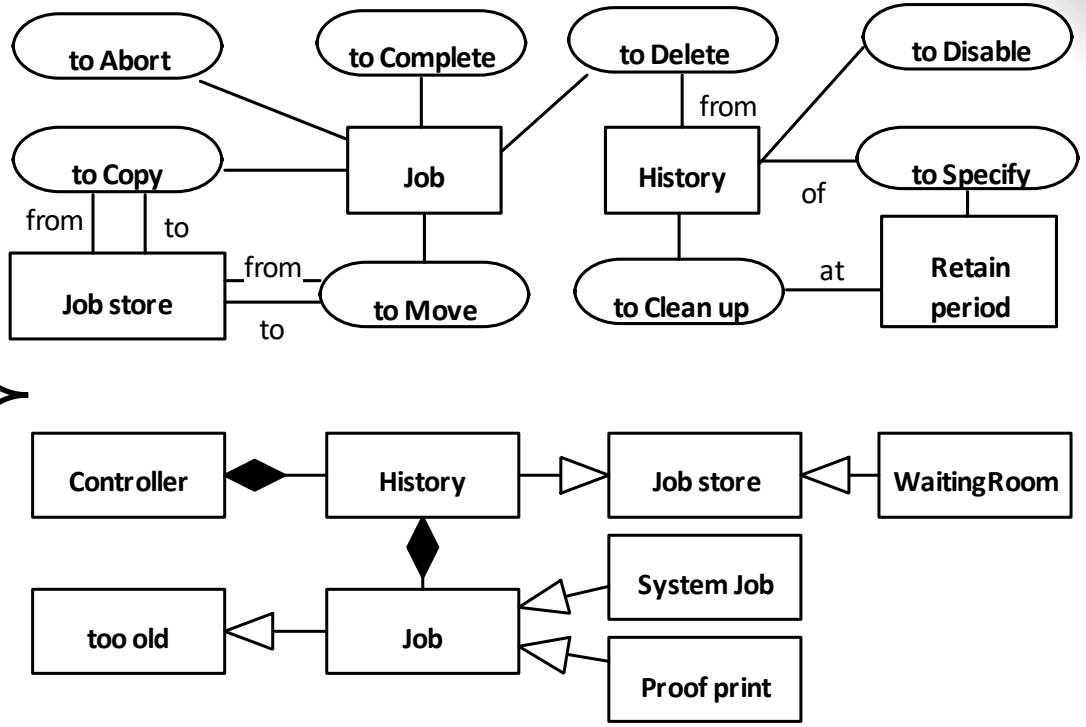


Definite articles indicate a function attribute:
A definite article, i.e., "the", might point to a role an object plays in a function. Classify it as a function attribute with a (domain) class as a type.

Verbs with different sets of related nouns indicate a homonym:
A verb is probably a homonym if it occurs in multiple phrases and the verb has different objects or prepositions related to it in those phrases. For example, in the phrases "I pay attention to the waiter" and "I pay the bill", "pay" is probably a homonym.

Excerpt from case study

Input sentence	Extracted phrases *(including final phrases)*	Decisions *(including new final phrases)*
When a print job is completed, it will be archived in the so-called "History".	*TO complete job* TO archive job in history	To archive and to move are synonyms. Chosen: to Move.
The History is a job store that will be used as a local temporary job store and is not intended for long term archiving purposes.	*History ISA job store* TO use history as local temporary job store TO intend History for purpose	To intend and to use are ignored because of guideline "Ignore intention phrases".
Only jobs that have been completed will end up in the History.	TO complete job Job TO end up in history	To end up is not a domain activity. "job is in History" is a state after "to archive". Chosen: *to move job from job store to job store*
Proof prints initiated from the waiting room and system jobs will not end up in the history when completed.	TO initiate proof print from waiting room *System job ISA job*	To initiate is considered out of scope. (It is in the scope of Job scheduling.), but *Proof print ISA job.*
Also jobs that have been aborted or deleted will not end up in the History.	*To abort job* *To delete job*	
The Settings editor provides functionality to clean up the History at specified time periods. The following time periods can be specified: One day, One week, One month, Forever.	To clean up history at time period. TO specify time period. One day, one week, one month, forever ISA time period.	Use retain period instead of time period. Furthermore, it is the retain period of the History which is specified, giving *TO specify retain period of history*, and *TO clean up History at retain Period.* One day, one week, one month, forever are possible values of retain period.
Jobs that have been longer in the History than the specified time period for the automatic cleanup are removed from the History	*History HAS jobs* To specify time period	
Therefore, jobs that are too old will automatically be removed from the history.	TO remove job from history *Job IS too old*	To Remove and to Delete are synonyms. Chosen: to Delete. Following the guideline: "Detect type of adjectives and adverbs", we asked what kind of thing "too old" is. We did not get a clear answer. So, we kept it.
If the history is disabled new completed jobs will be removed from the system, so they will not end-up in the history.	*TO disable history* TO complete job TO remove job from system	System and controller are synonyms. Chosen: controller. Giving: *Controller HAS history*
A job can be reprinted from the History by copying them from history to waiting room.	TO reprint job from history *TO copy job from history to waiting room*	Is "reprint" the activity or "copy"? Answer: To copy. Reprint is the intention. And, what is a waiting room? Answer: *Waiting room ISA job store. Giving: TO copy job from job store to job store.*



Conclusions

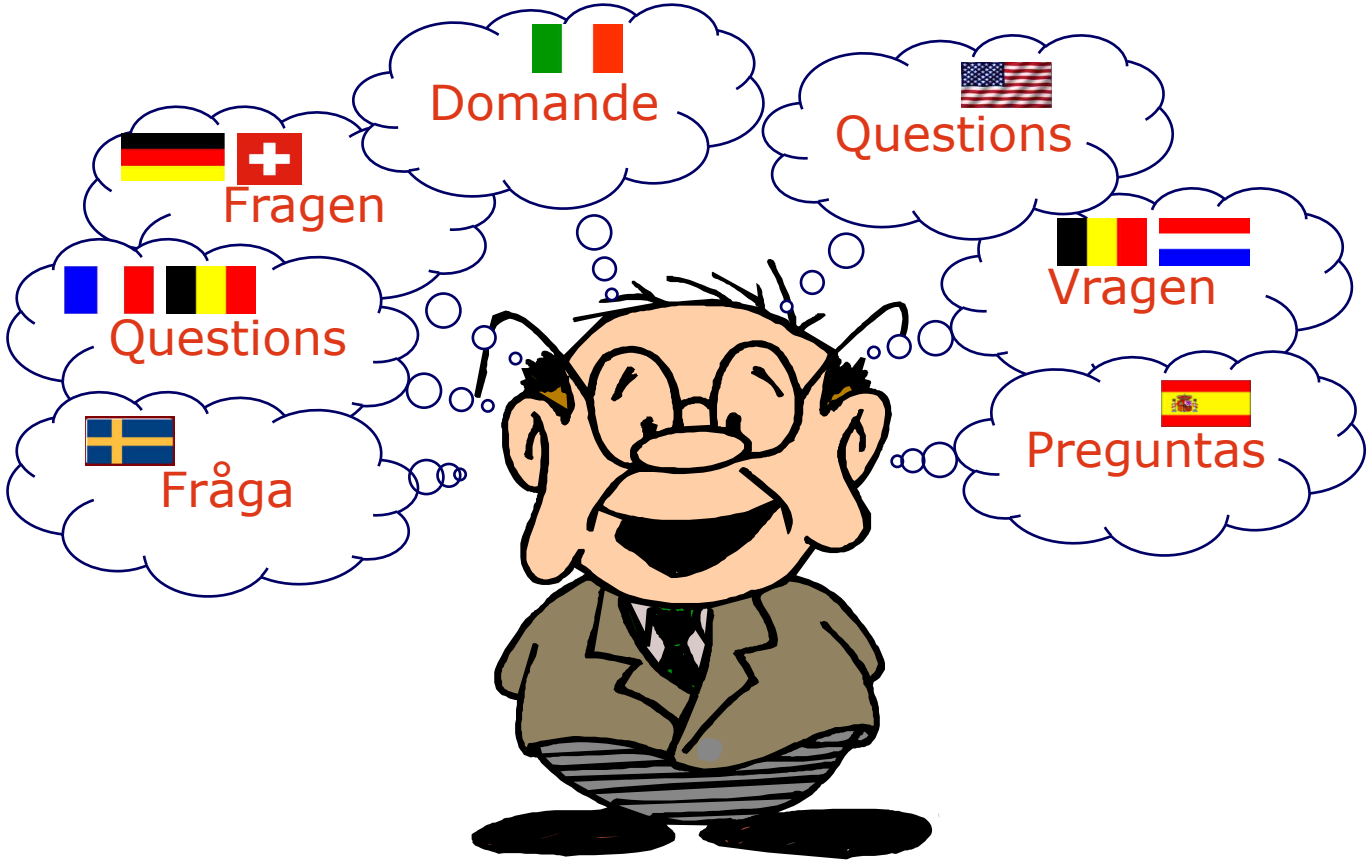
- Metamodel and steps are quite mature:
  - Result of 25 years of modelling and method engineering experience

- There are still enough guidelines to discover and capture:
  - Just started to document them. (https://github.com/robertdeckers/MuDForM)
  - Retrieve more guidelines from literatue → Literature study needed.
  - Build a community:
    - MuDForM is available with UML notation, an initial course exists.
    - MPS-based tool is under development.

- Method definition is too academic: handbook for practitioners needed.

- Supporting the transition from document-based enterprise to model-based enterprise.
  → PoEM

→Wanna exchange ideas and/or cooperate?



robert.deckers@AtomFreeIT.com
+31 6 46882428

Back up slides

Today’s starting points

Research Questions

- What methodical support can be given for the conversion of text into ingredients of a domain-oriented model?

- How should methodical support for extracting knowledge from text be integrated in a method to produce domain-oriented models?

My research scope…

MuDForM: Multi-Domain Formalization Method

RQ1: How to formalize the specification of decisions (requirements/design) for different stakeholders?

- Rigorous method for creating domain models, and domain-model-based specs, based on NLP.
- Several cases present
→ More quality attribute cases should be studied.

- Different integration types defined.
- Consistency before transformation
→Formal method needed

RQ2: How to integrate specifications, in particular of qualities and design aspects?

RQ3: How to guarantee that a (software) system meets all specifications?

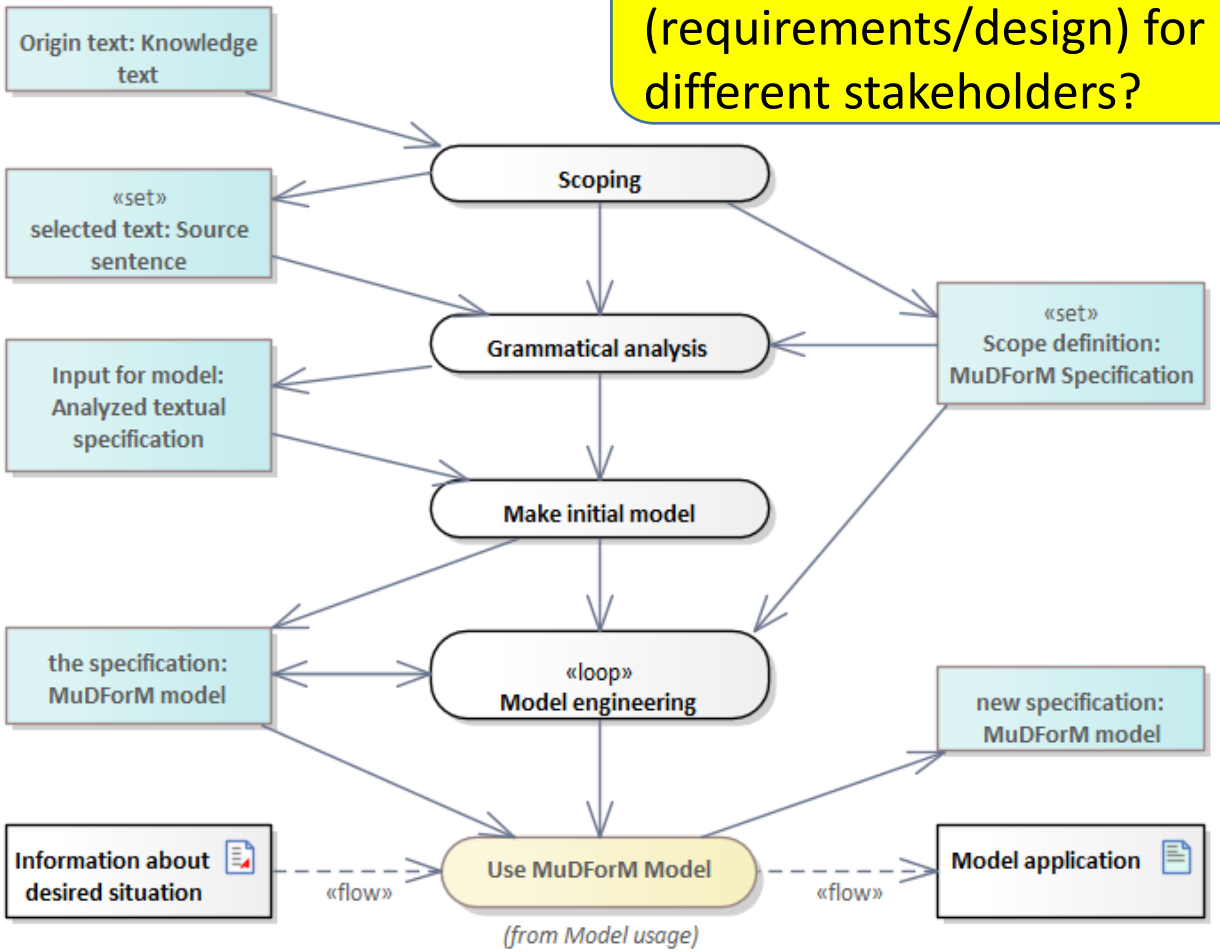
RQ4: What cognitive concepts should the set of specification primitives contain?

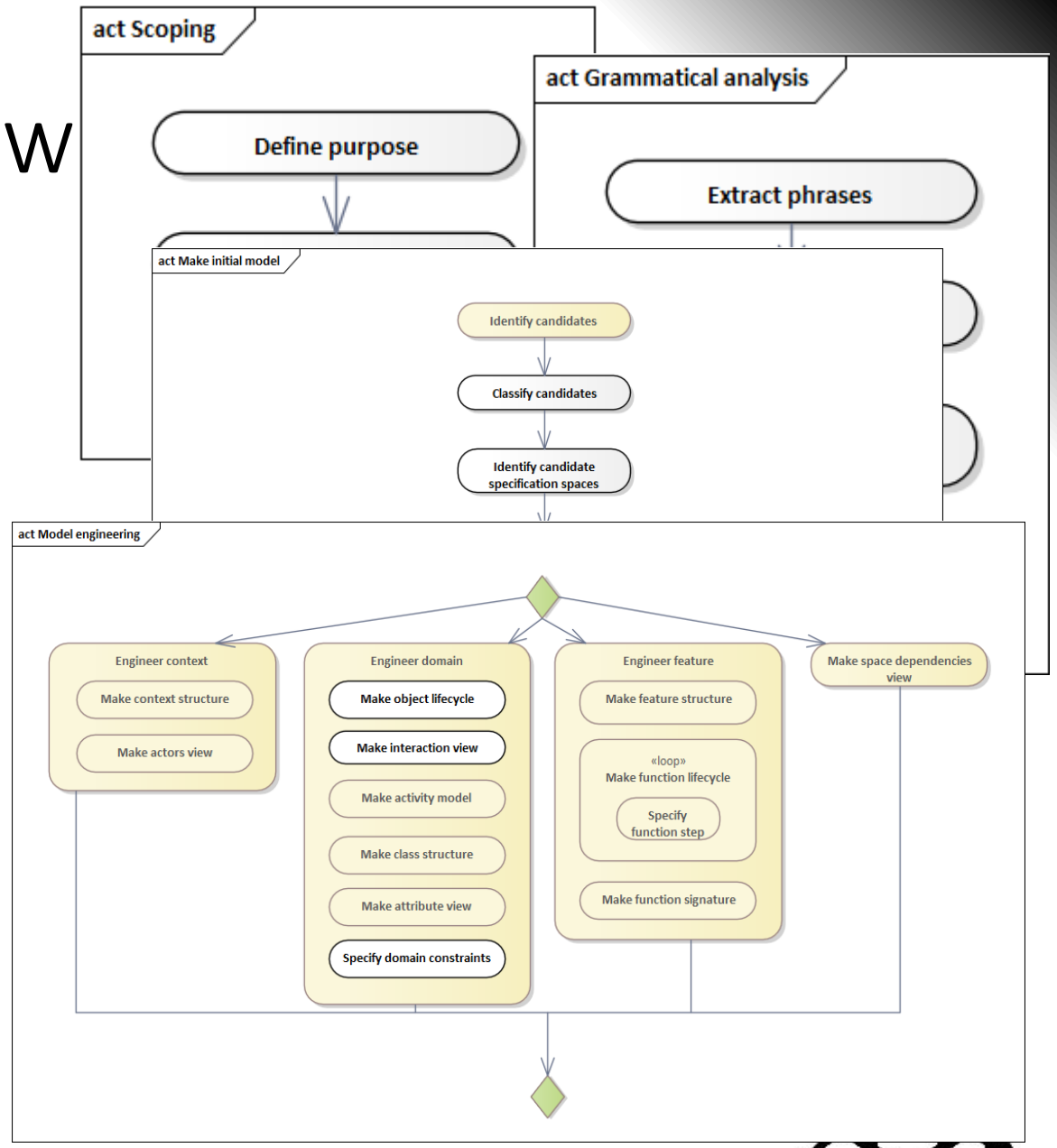
- Literature on formal languages, linguistics, language philosophy. Many sub answers.
- Initial meta model made.
→ Hints are welcome

- Reference architecture from personal experience
- Connect QA requirements with existing architectural patterns/tactics
→ Looking for cooperation in the future

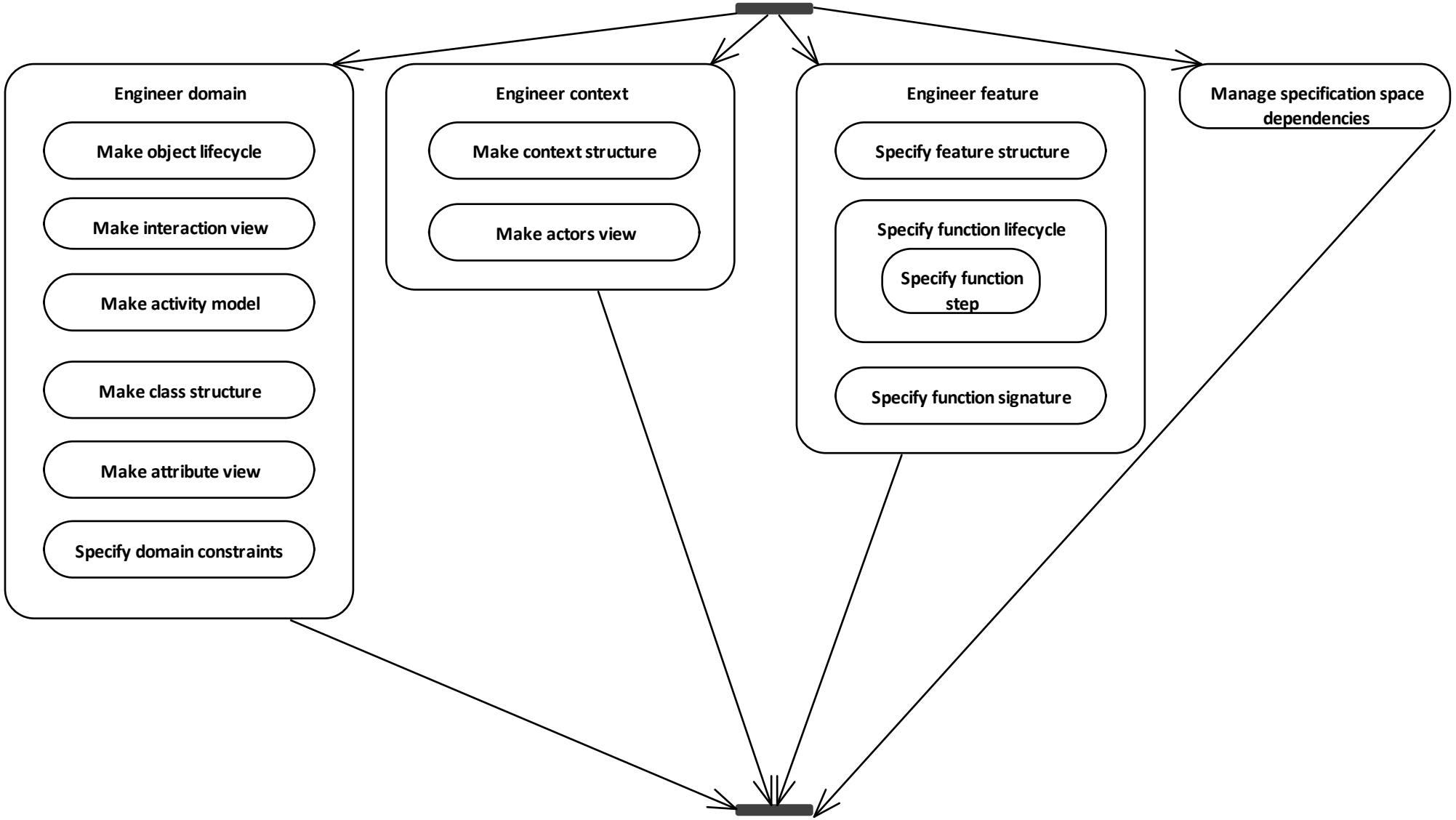
RQ1: MuDForM method flow

RQ1: How to formalize the specification of decisions (requirements/design) for different stakeholders?

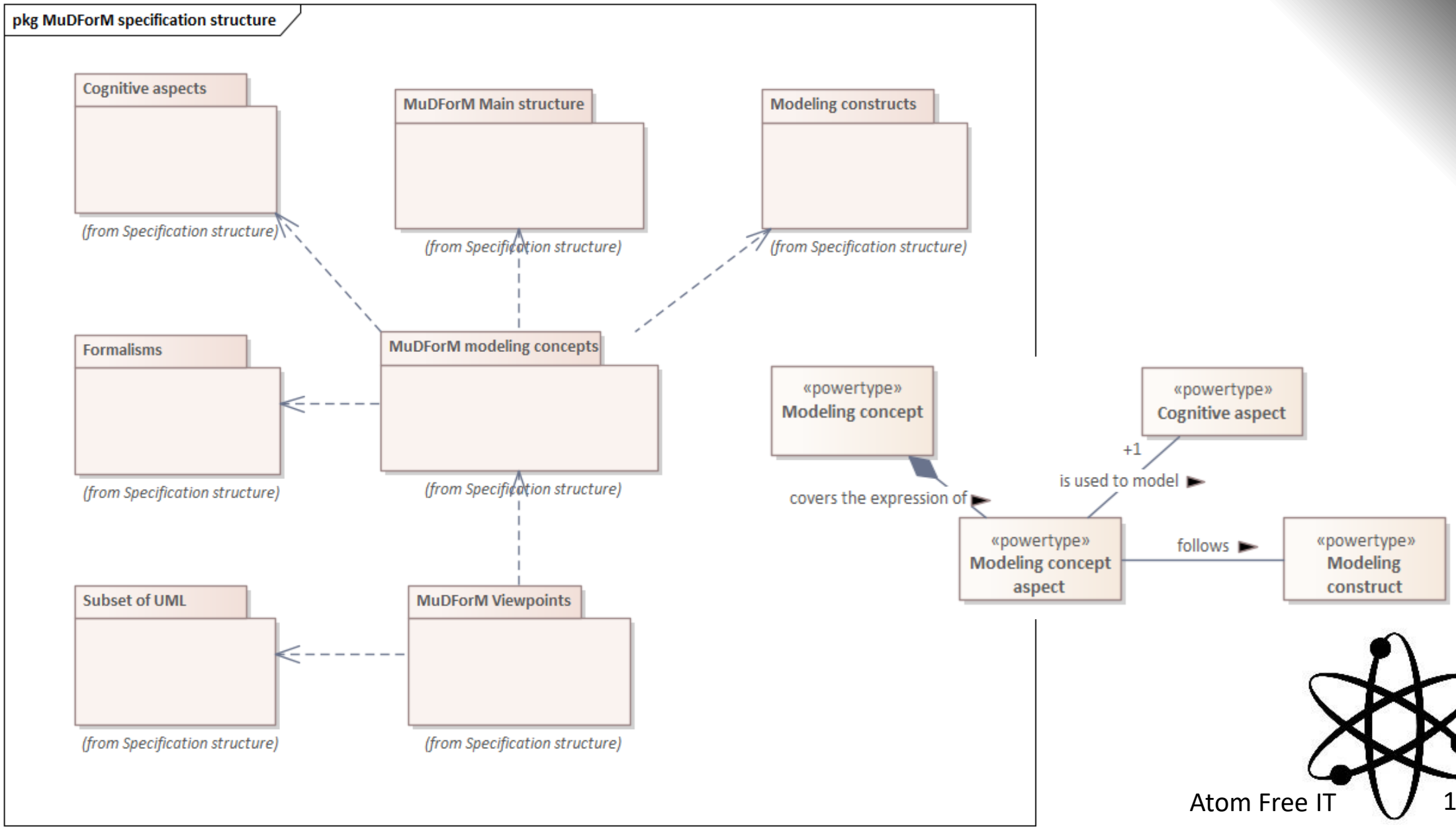




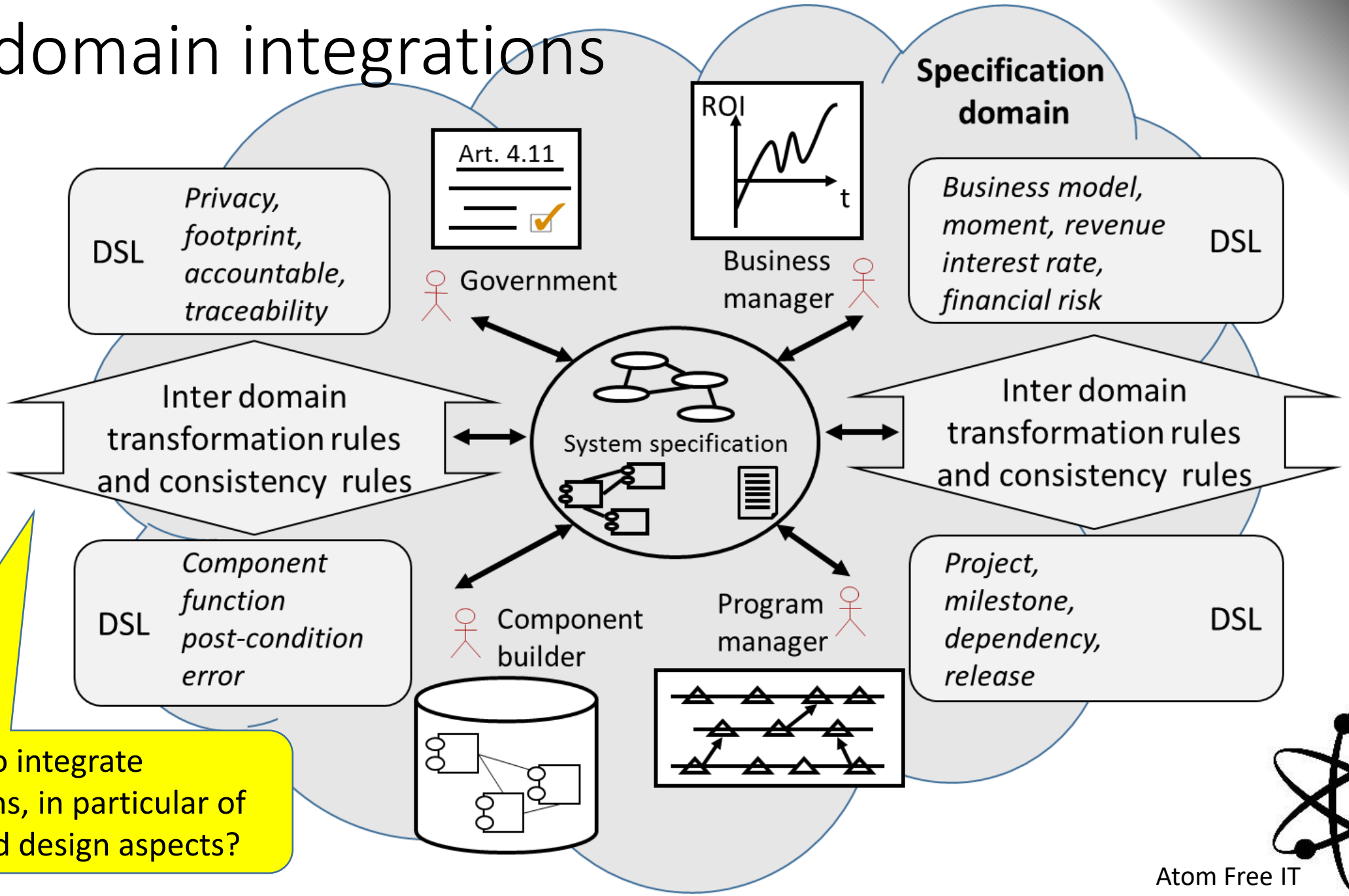
Method flow: model engineering



RQ1: MuDForM meta model

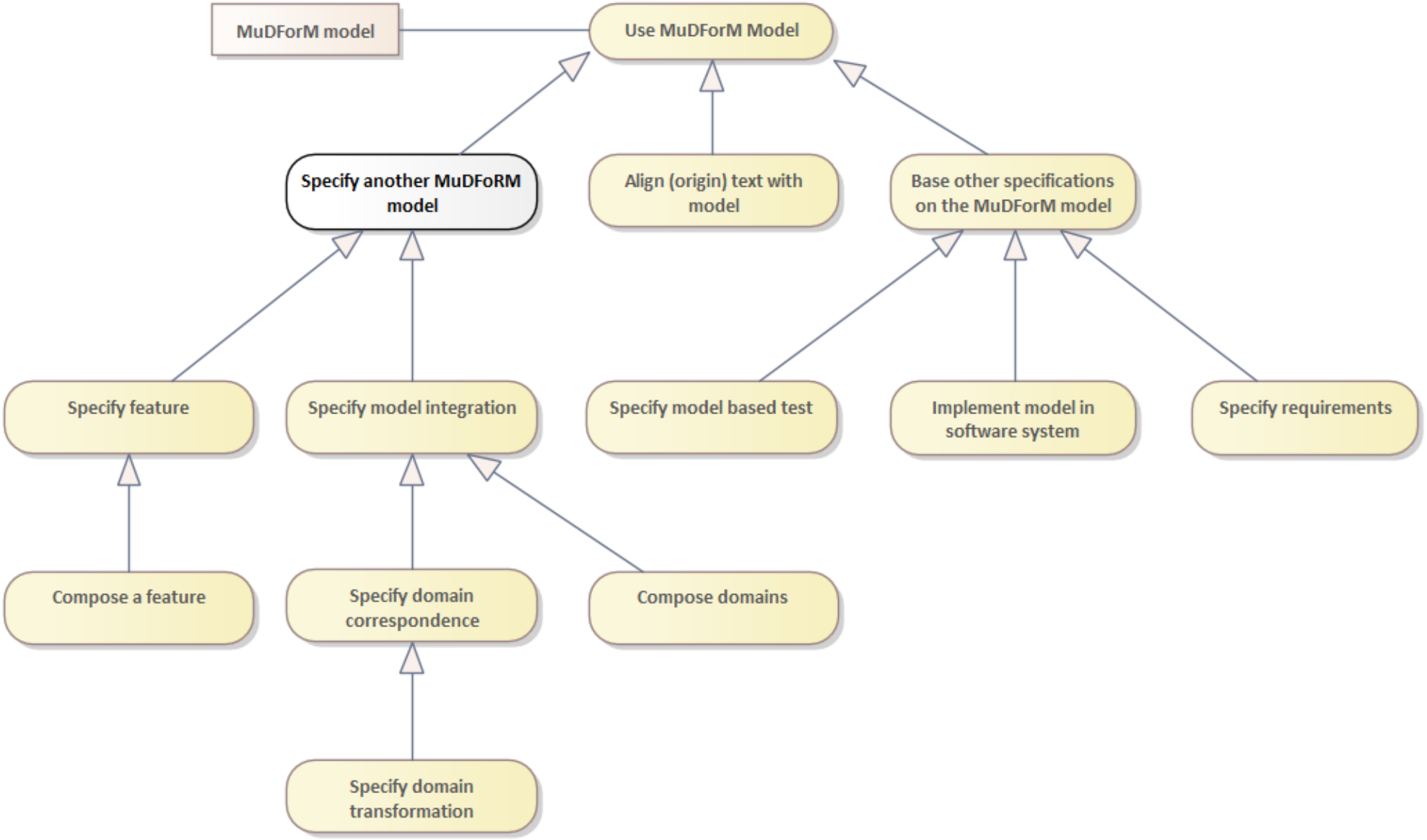


RQ2: domain integrations

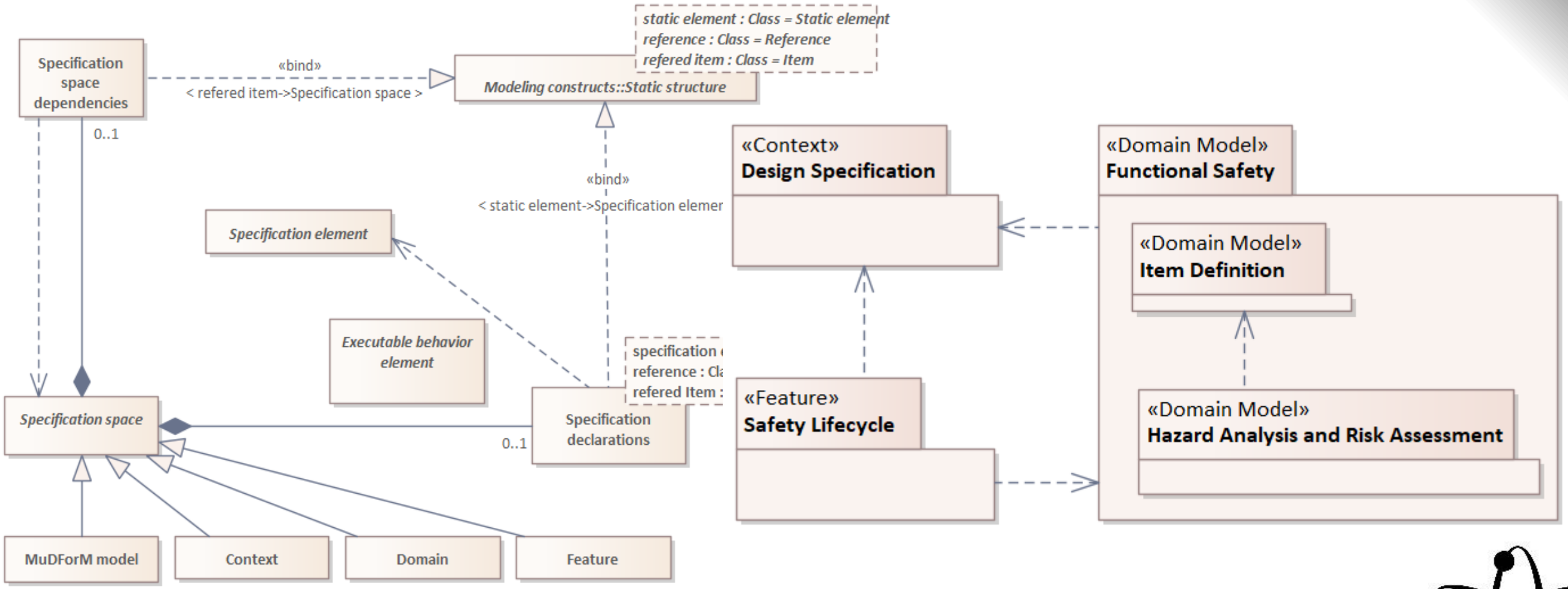


RQ2: How to integrate specifications, in particular of qualities and design aspects?

RQ2: different types of integration



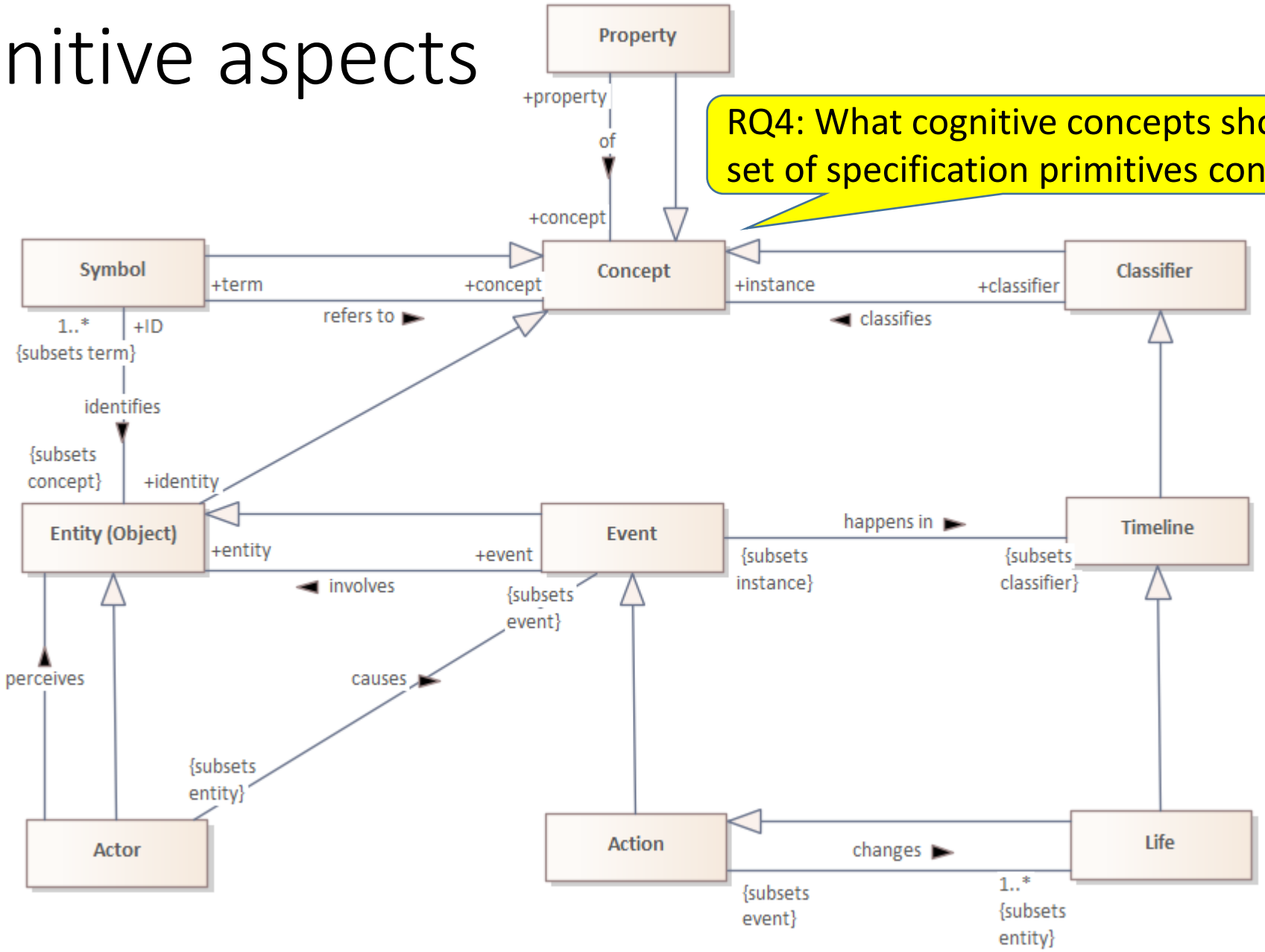
RQ2: different model types



RQ4: cognitive aspects

To be done:

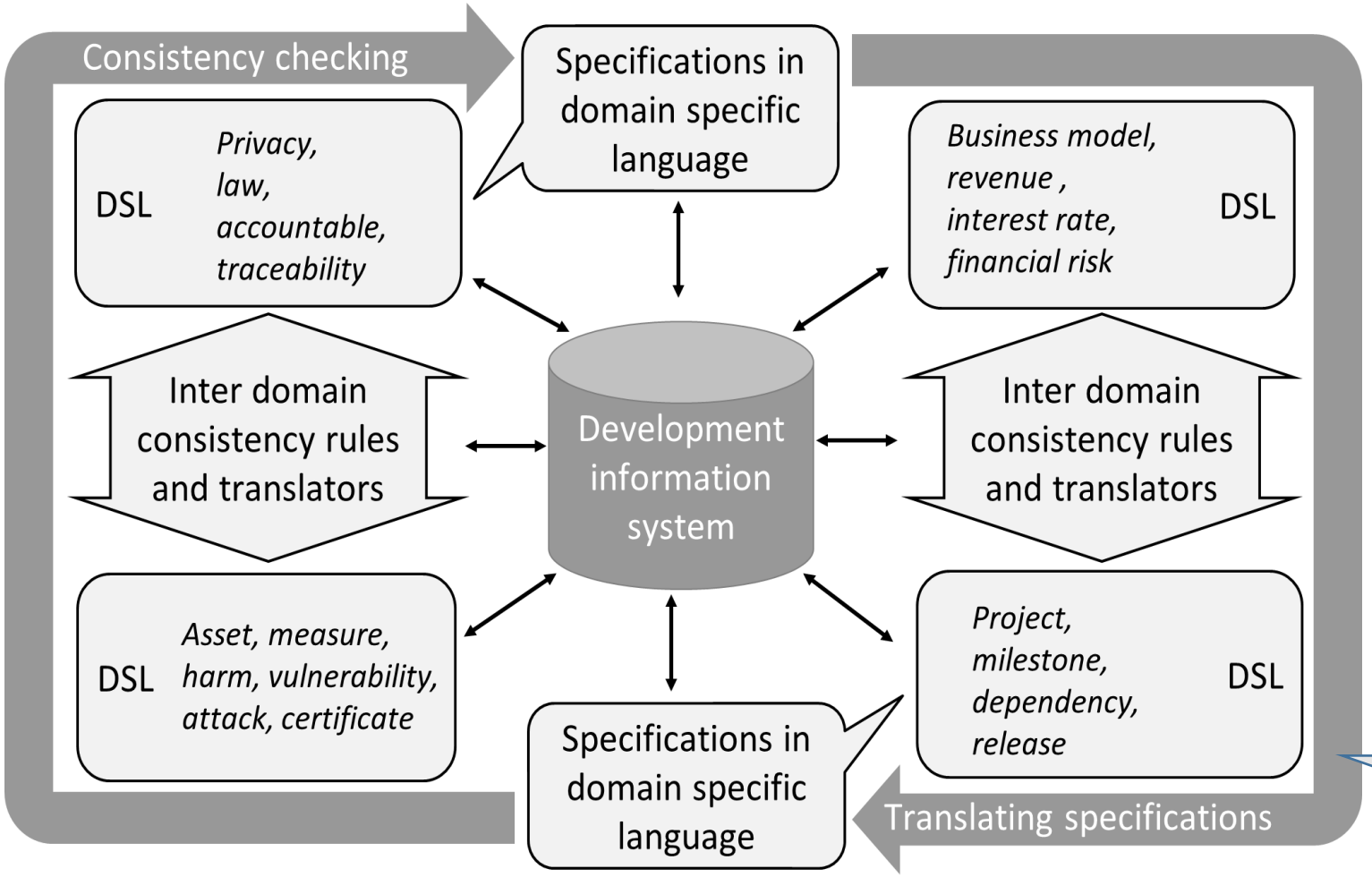
- Analogies
- Causality



RQ4: What cognitive concepts should the set of specification primitives contain?

RQ2: at runtime. (We did it 25 years ago)

RQ3: How to guarantee that a (software) system meets all specifications?



Message Window

A runtime engine that checks and translates