An introduction to Conceptual Modeling

Dipartimento di Matematica e Informatica, Universita' di Firenze

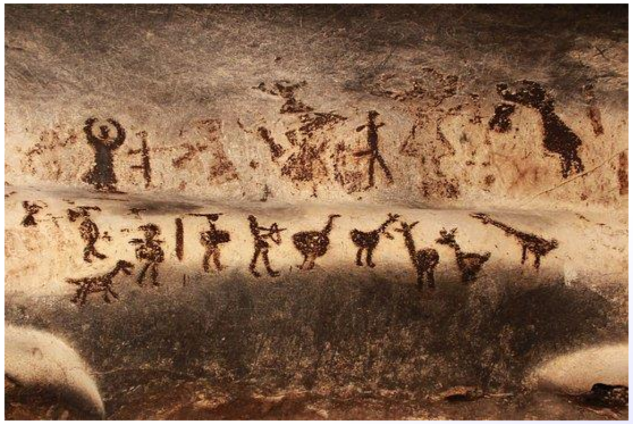
mohamad.gharib@unifi.it

Outline

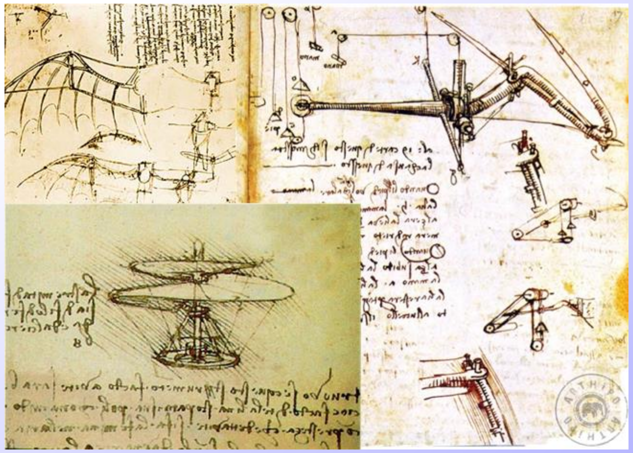
- Short history of modeling
- Modeling in Computer Science
- Types of Modeling in Computer Science
- What is Conceptual Modeling?
- Breif history of modeling languages
- Conceptual modeling languages
- What is a metamodel?
- Meta-Modeling and the OMG Meta Object Facility (MOF)

Short history of modeling

- **Pre – information age**
  - Humans used symbols to model their environment since thousands of years.
  - Then, they start to model in science and engineering areas.
  - But their models were limited by the size of the medium on which they were represented.



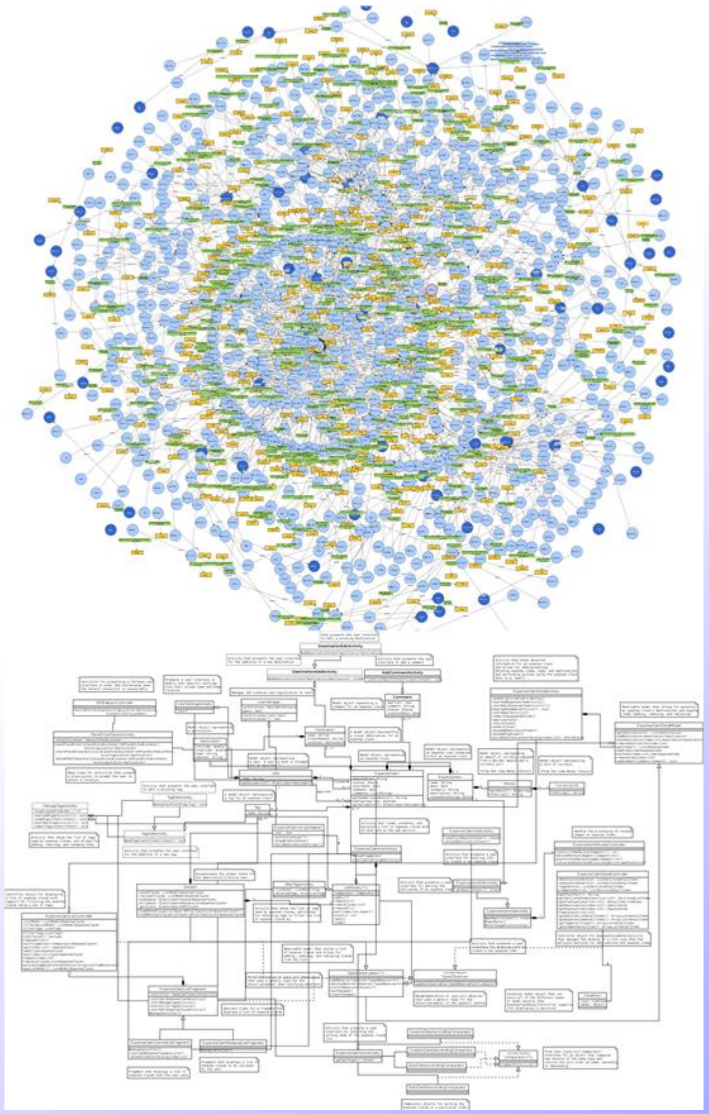
Magura cave – Bulgaria 6th – 8th century BC



Leonardo da Vinci 15th – 16th century

Short history of modeling

- Pre – information age
  - Humans used symbols to model their environment since thousands of years.
  - Then, they start to model in science and engineering areas.
  - But their models were limited by the size of the medium on which they were represented.
- Information age
  - Almost no limits anymore. More specifically, the limits of models and modeling were defined by the capabilities of the machines on which they were developed and presented.



Modeling in Computer Science

- Modeling in the main areas of Computer Science:
  - *Databases:* semantic [data] models (e.g., ER, EER) to design databases;
  - *Artificial Intelligence (AI):* knowledge representation depending on Description Logics (DL), semantic networks, ontologies, etc. to build knowledge bases;
  - *Software Engineering:* system modeling (model-driven architectures (MDA) and model-driven engineering (MDE)):
    - Architecture and requirements models (e.g., Object Oriented diagrams, Goal models, UML, SysML, etc.);
    - Software/business process modes (e.g., Business Process Model and Notation (BPMN), Petri-nets, statecharts, etc.).

Modeling in Computer Science

- Modeling in the main areas of Computer Science:
  - *Databases:* semantic [data] models (e.g., ER, EER) to design databases;
  - *Artificial Intelligence (AI):* knowledge representation depending on Description Logics (DL), semantic networks, ontologies, etc. to build knowledge bases;
  - *Software Engineering:* system modeling (model-driven architectures (MDA) and model-driven engineering (MDE)):
    - Architecture and requirements models (e.g., Object Oriented diagrams, Goa models, UML, SysML, etc.);
    - Software/business process modes (e.g., Business Process Model and Notatior (BPMN), Petri-nets, statecharts, etc.).

Modeling in Computer Science

- Modeling **use** in main areas of Computer Science:
  - *Databases:* people use semantic [data] models to facilitate the structuring of large amounts of data;
  - *Artificial Intelligence (AI):* [expert] systems use **knowledge bases** to infer new knowledge, and/or perform complex [intelligent] tasks;
  - *Software Engineering:* people (usually, system stockholders) use the resulting diagrams/models for communicating and exchanging knowledge among one another.

Modeling in Computer Science

- Is there any implications of the different uses of models?

Modeling in Computer Science

- Is there any implications of the different uses of models? Yes

Modeling in Computer Science

- Is there any implications of the different uses of models? Yes
  - *Level of formality of language,* formal, semi-formal, informal e.g., if people use the language can be semi-formal or even informal; for expert systems (AI) the language should be formal.
  - *Types of knowledge to be captured,*
    - for SE and Databases knowledge related to the domain;
    - for AI, knowledge related to the task.
  - *Level of completeness of the models*
    - for SE and Databases coverage can be incomplete;
    - for AI, coverage has to be, more or less, complete.

Modeling in Computer Science

- Is there any implications of the different uses of models? Yes
  - *Level of formality of language,* formal, semi-formal, informal e.g., if people use the language can be semi-formal or even informal; for expert systems (AI) the language should be formal.
  - *Types of knowledge to be captured,*
    - for SE and Databases knowledge related to the domain;
    - for AI, knowledge related to the task.
  - *Level of completeness of the models*
    - for SE and Databases coverage can be incomplete;
    - for AI, coverage has to be, more or less, complete.

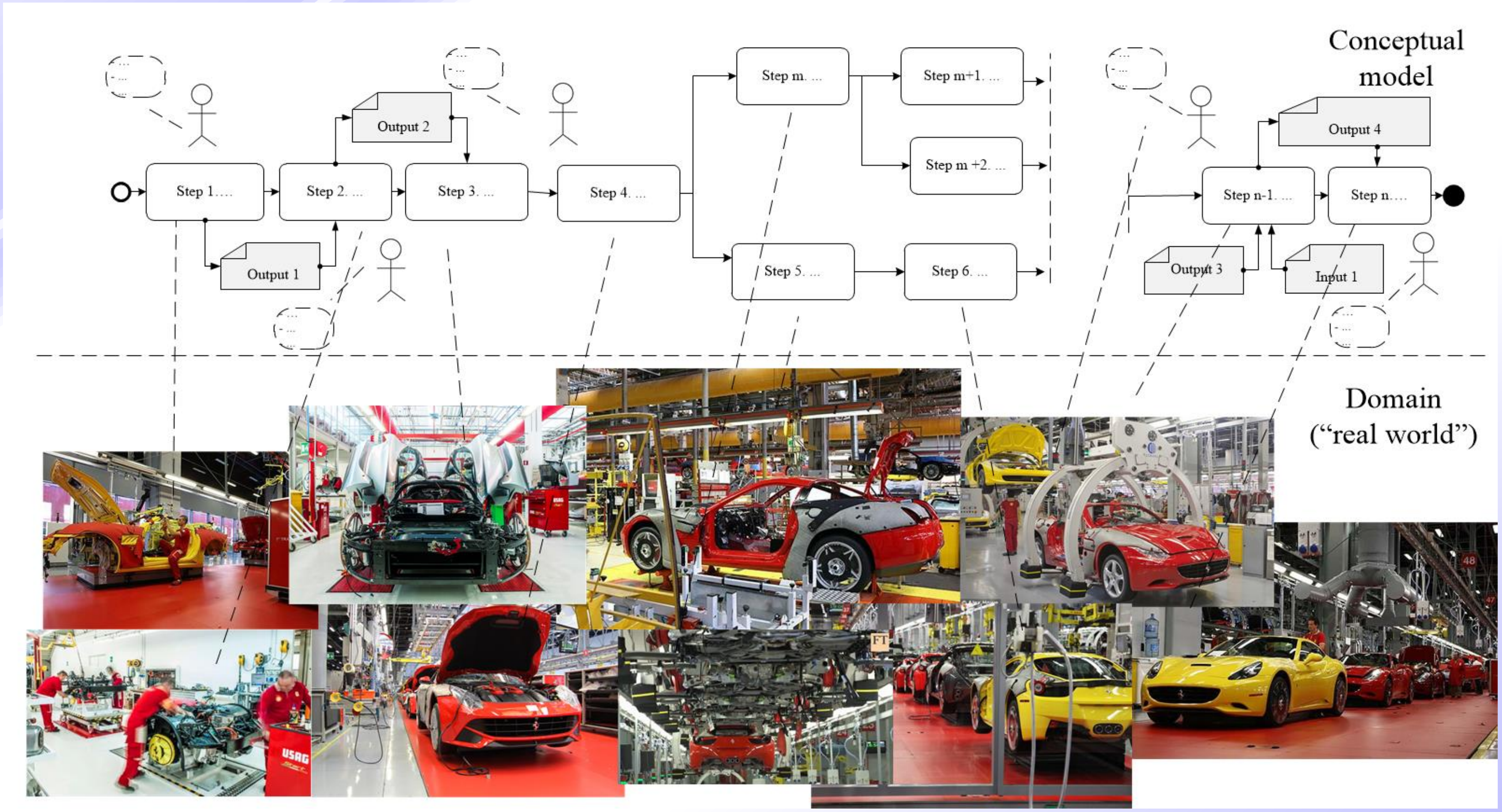
Types of Models in Computer Science

- **Physical models** use specific and less generic machine-oriented terms/concepts (e.g., columns, keys, data types, validation rules, database triggers, procedures, access constraints)
- **Logical models** use specific [business-oriented] terms/concepts (e.g., entities (tables), attributes (columns/fields) and relationships (keys)).
- **Conceptual models** use high-level non-technical terms/concepts (e.g., almost every thing you can imagine).

Types of Models in Computer Science

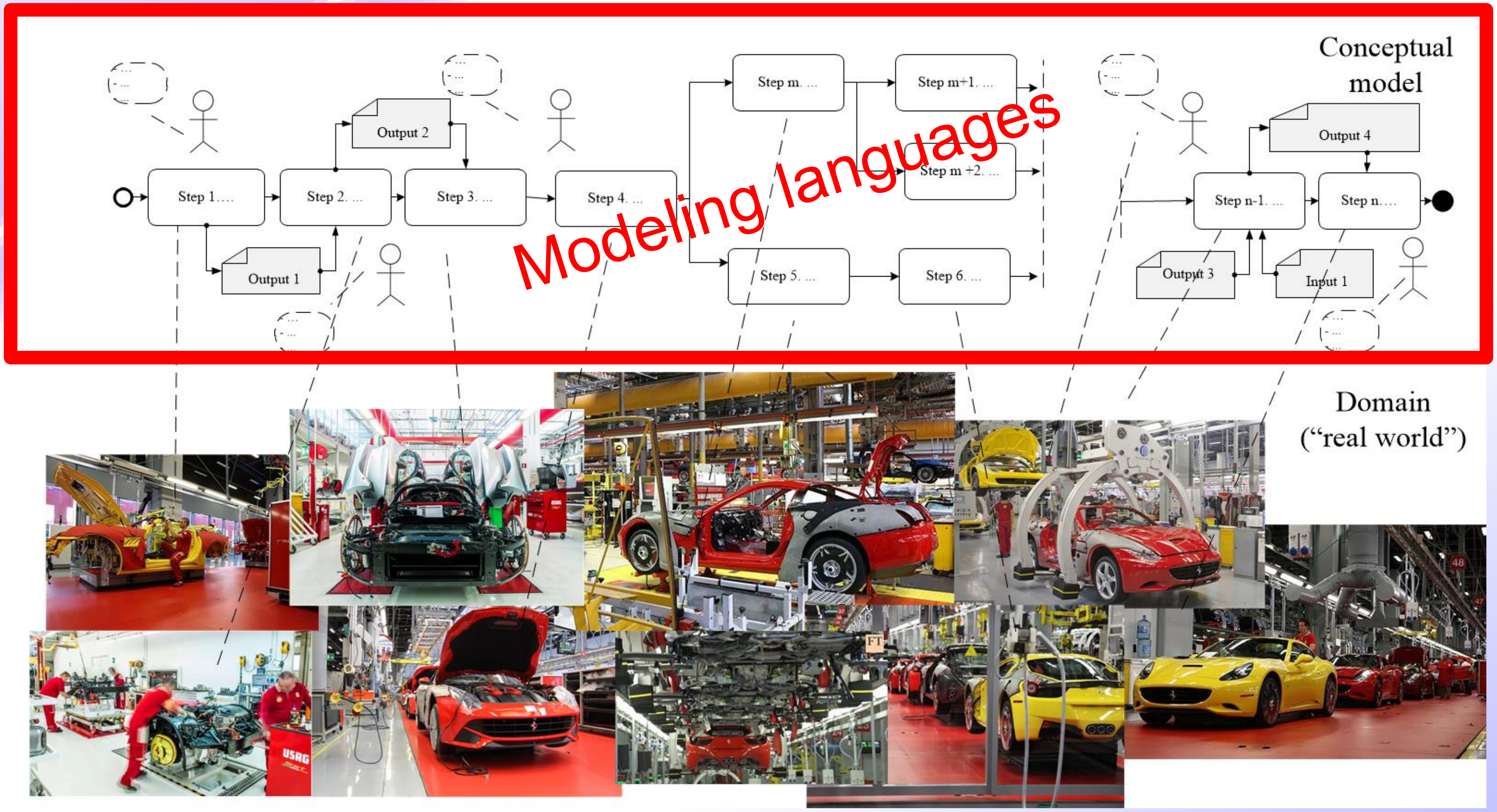
- **Physical models** use specific and less generic machine-oriented terms/concepts (e.g., columns, keys, data types, validation rules, database triggers, procedures, access constraints)
- **Logical models** use specific [business-oriented] terms/concepts (e.g., entities (tables), attributes (columns/fields) and relationships (keys)).
- **Conceptual models** use high-level non-technical terms/concepts (e.g., almost every thing you can imagine).

What is Conceptual Modeling?



- Conceptual model should represent (model) specific aspects of a specific domain.

What is Conceptual Modeling?



- Conceptual model should represent (model) specific aspects of a specific domain.

Brief history of modeling languages

➢ Brief history of modeling languages in Computer Science

- In the 60s, limited attempts for modeling the “real world” to extend some programming language.
- In the 70s , the Entity-Relationship (E-R) model was developed.
- In the 80s, the attempts to extend the software/hardware limited view of system modeling to consider the environment where such system will be implemented have started.
- In the 90s, the Unified Modeling Language (UML) was developed and latter adopted as a standard modeling language by Object Management Group (OMG).
- In 2001, the Systems Modeling Language (SysML) was developed as an extension of a subset of the UML depending on the same UML's profile mechanism.

Brief history of modeling languages

➢ Brief history of modeling languages in Computer Science

- In the 60s, limited attempts for modeling the “real world” to extend some programming language.
- In the 70s , the Entity-Relationship (E-R) model was developed.
- In the 80s, the attempts to extend the software/hardware limited view of system modeling to consider the environment where such system will be implemented have started.
- In the 90s, the Unified Modeling Language (UML) was developed and latter adopted as a standard modeling language by Object Management Group (OMG).
- In 2001, the Systems Modeling Language (SysML) was developed as an extension of a subset of the UML depending on the same UML's profile mechanism.

"Conceptual" modeling languages

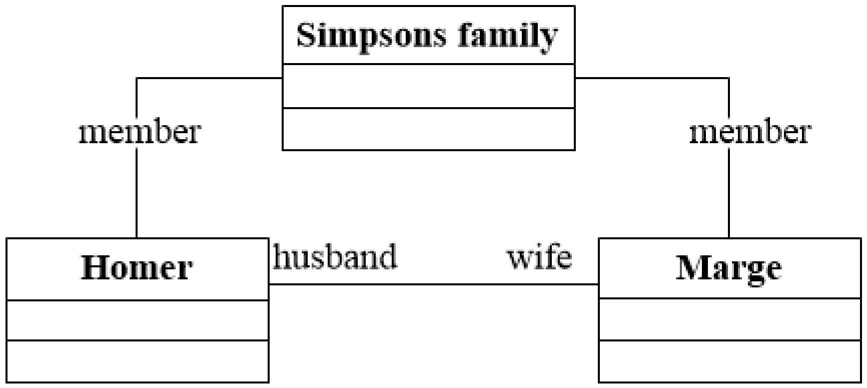
- A modeling language is used to express (represent) information/knowledge about a system, domain, etc. in a structured and consistent way relying on a set of rules (language semantics).
- A conceptual modeling language includes:
  - Building blocks (constructs): 1- Primitive Terms, e.g., classes, stereotypes, association, etc. and 2- Abstraction Mechanisms, e.g., Generalization, Aggregation.
  - Semantics: constraints on the use of building blocks of the model (e.g., OCL).

"Conceptual" modeling languages

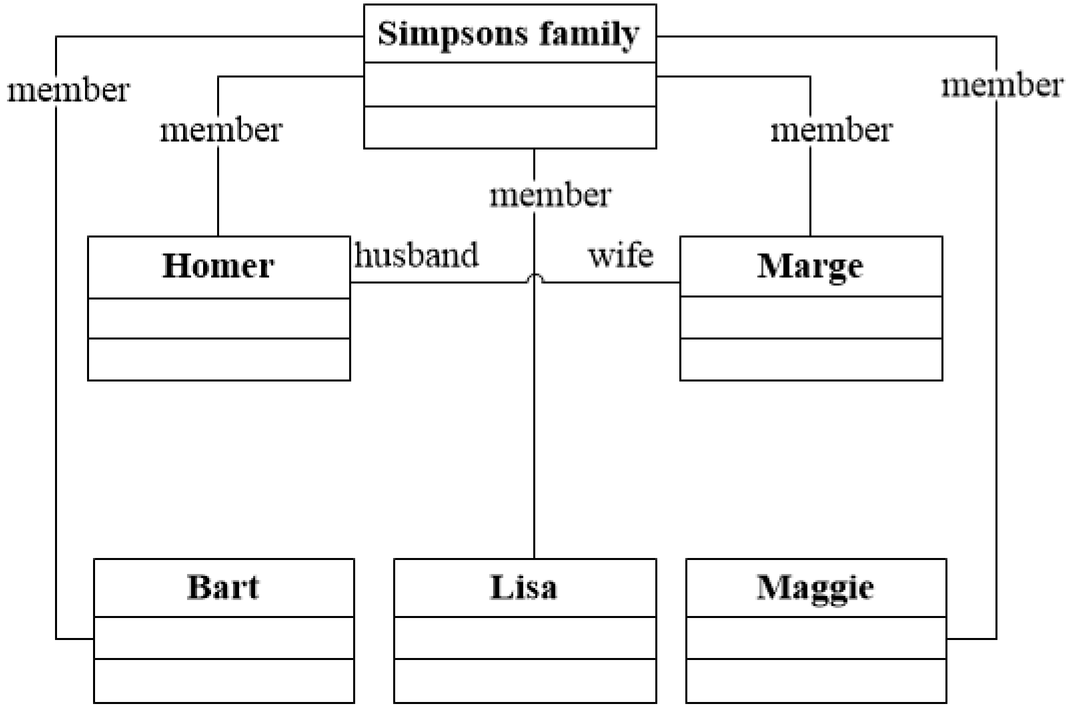
- A modeling language is used to express (represent) information/knowledge about a system, domain, etc. in a structured and consistent way relying on a set of rules (language semantics).
- A conceptual modeling language includes:
  - Building blocks (constructs): 1- Primitive Terms, e.g., classes, stereotypes, association, etc. and 2- Abstraction Mechanisms, e.g., Generalization, Aggregation.
  - Semantics: constraints on the use of building blocks of the model (e.g., OCL).
- Tools can be used for creating, managing, and "validating" a model.

Conceptualization the Simpsons family

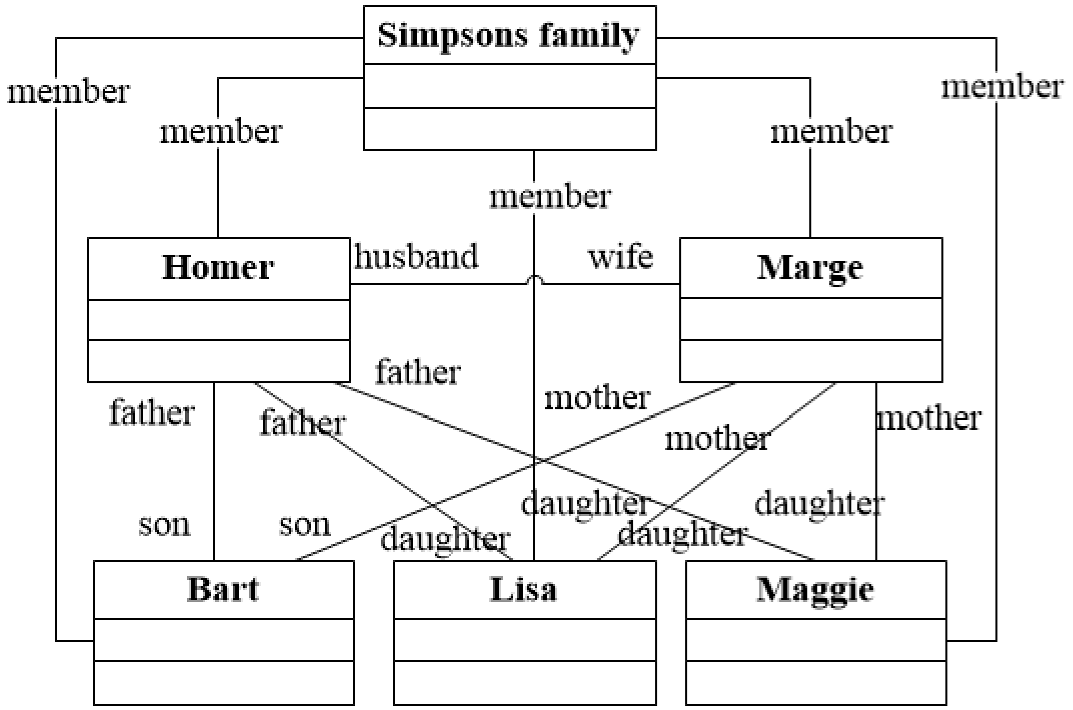
Conceptualization the Simpsons family



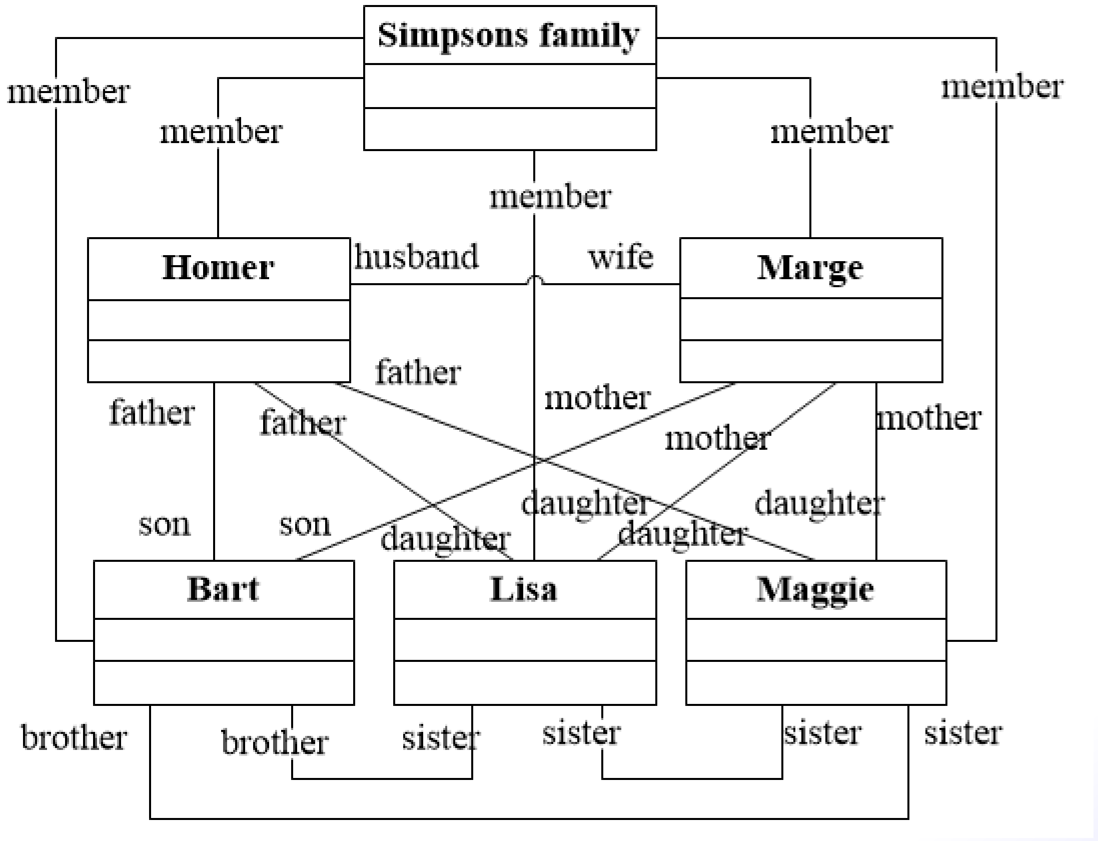
Conceptualization the Simpsons family



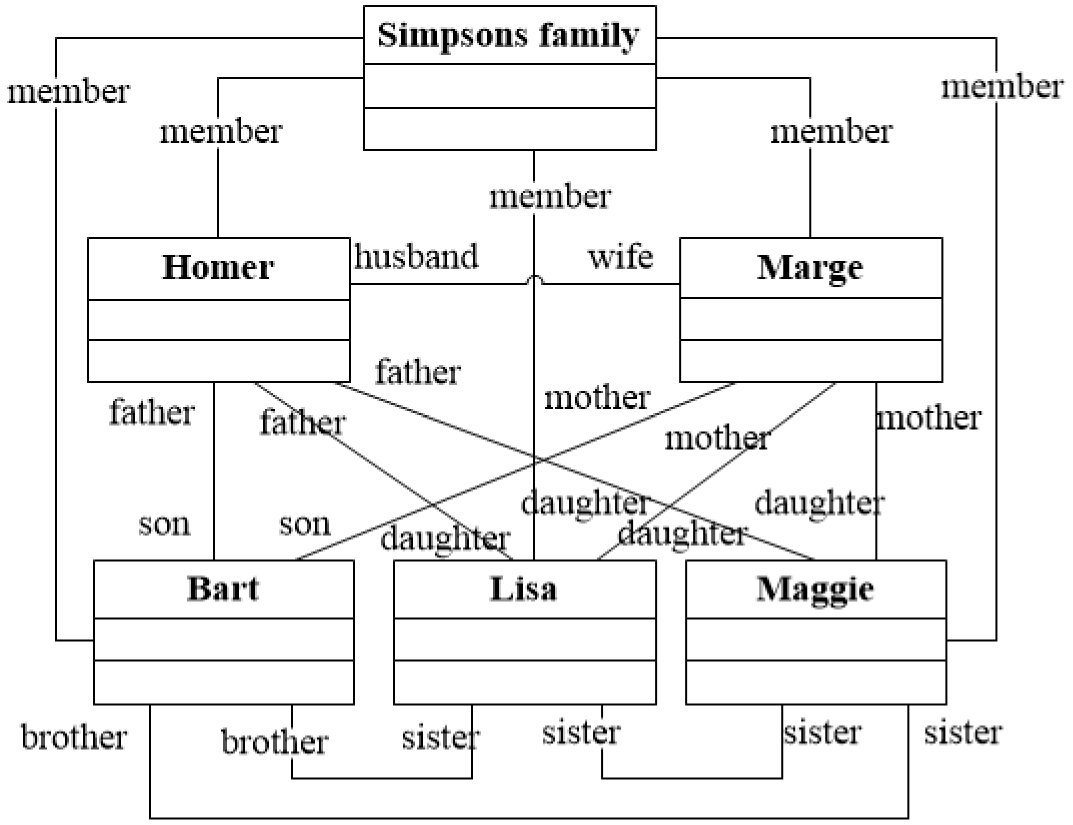
Conceptualization the Simpsons family



Conceptualization the Simpsons family



Conceptualization the Simpsons family



- ➢ But how can we use these “concepts” and “relationships” to model other families?

What is a metamodel?

- In Computer Science, the term is used heavily and with several different meanings:
  - In Databases, a metadata means "data about data";
  - In Conceptual Modeling, a metamodel means a "model of a data model".
- *One of the most fundamental task for developing a modeling language is defining its metamodel .*

What is a metamodel?

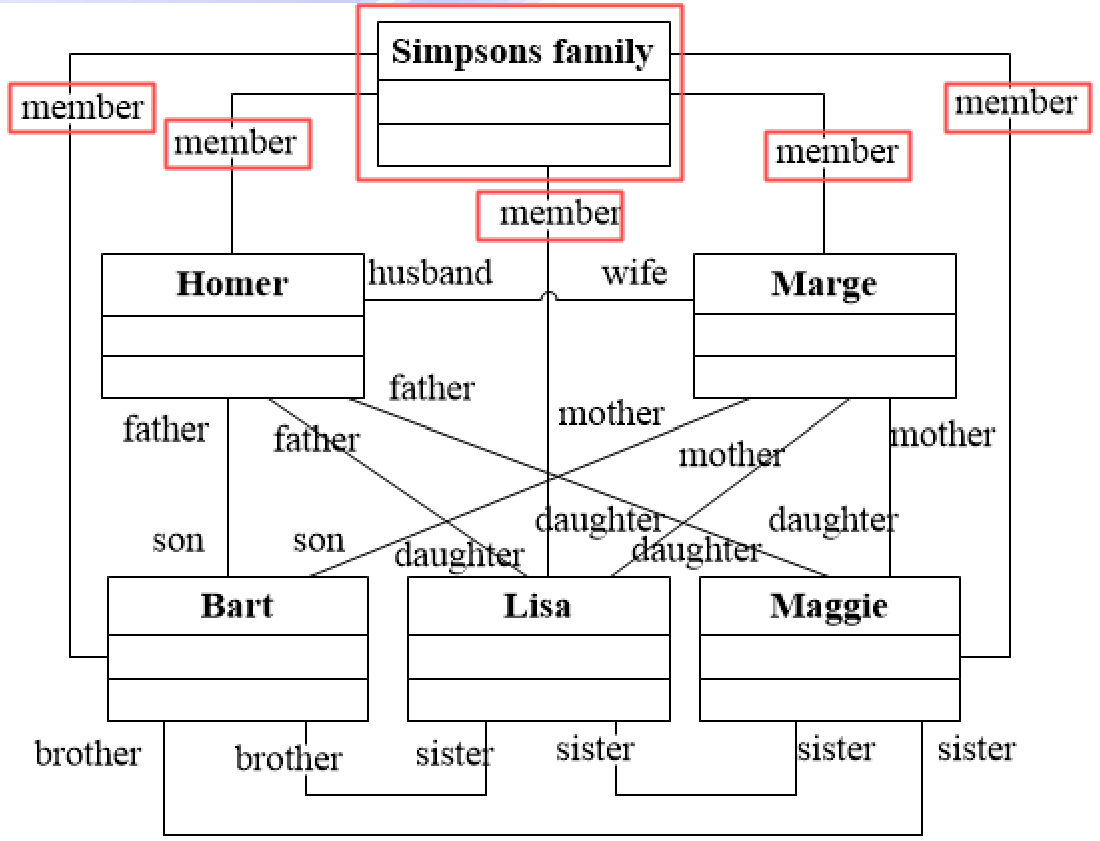
- In Computer Science, the term is used heavily and with several different meanings:
  - In Databases, a metadata means "data about data";
  - In Conceptual Modeling, a metamodel means a "model of a data model".
- *One of the most fundamental task for developing a modeling language is defining its metamodel. WHY*

What is a metamodel?

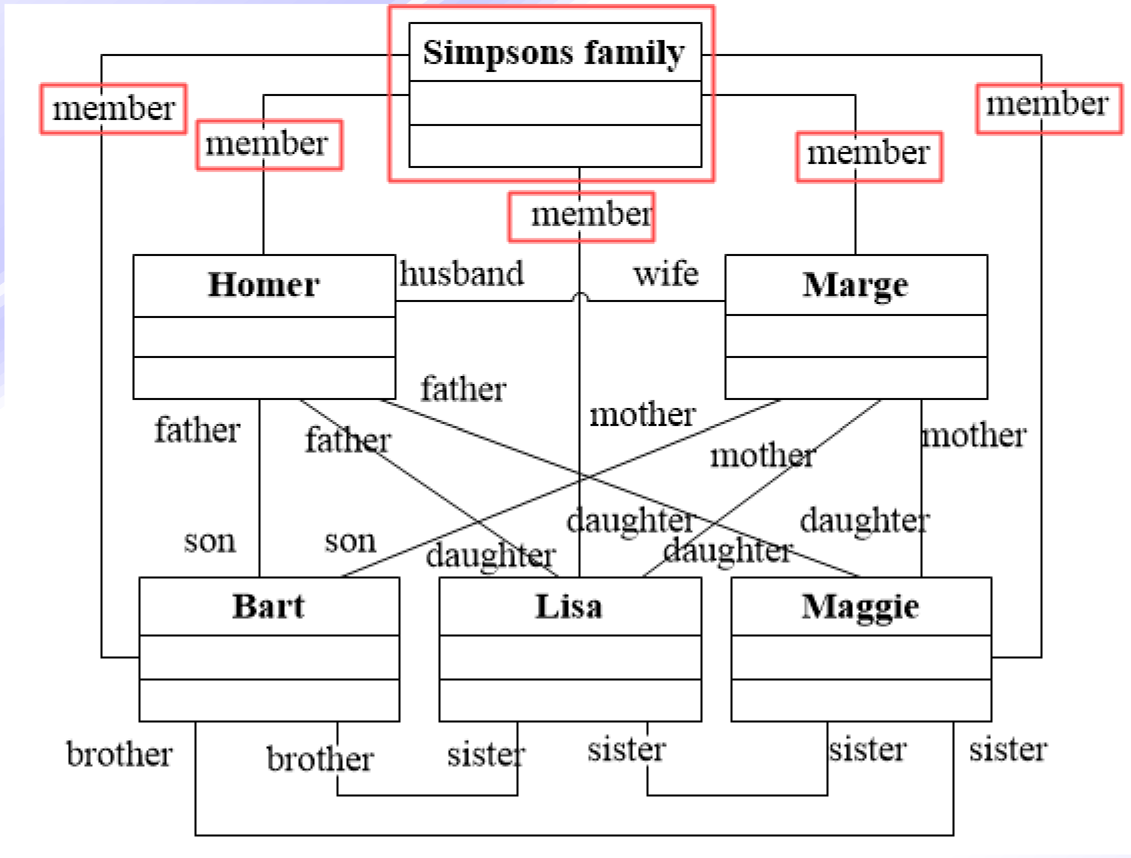
- In Computer Science, the term is used heavily and with several different meanings:
  - In Databases, a metadata means "data about data";
  - In Conceptual Modeling, a metamodel means a "model of a data model".
- *One of the most fundamental task for developing a modeling language is defining its metamodel. WHY*

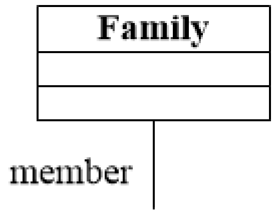
- ***A metamodel*** defines the ***key*** concepts of a ***modeling language*** as well as various relationships among these concepts.

Designing a metamodel for a language to describe a family

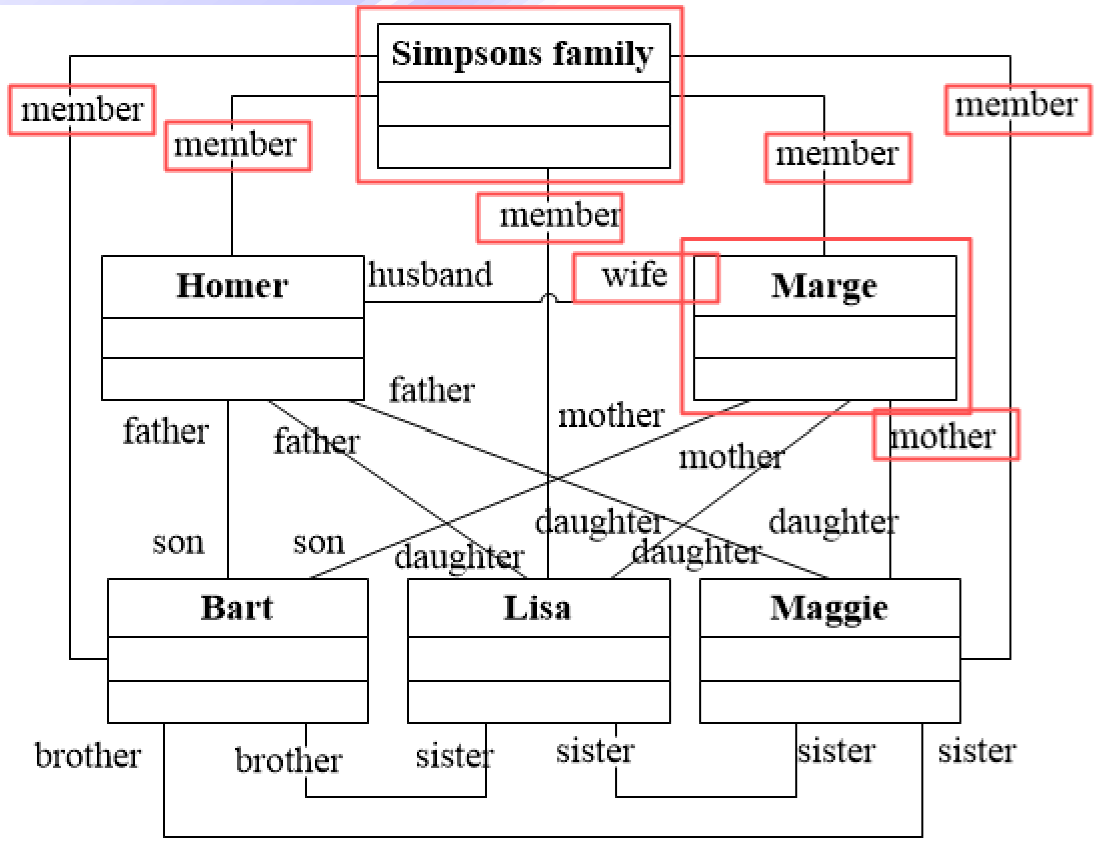


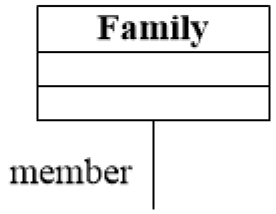
Designing a metamodel for a language to describe a family

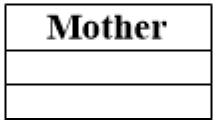


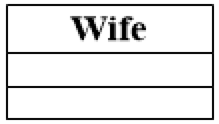


Designing a metamodel for a language to describe a family

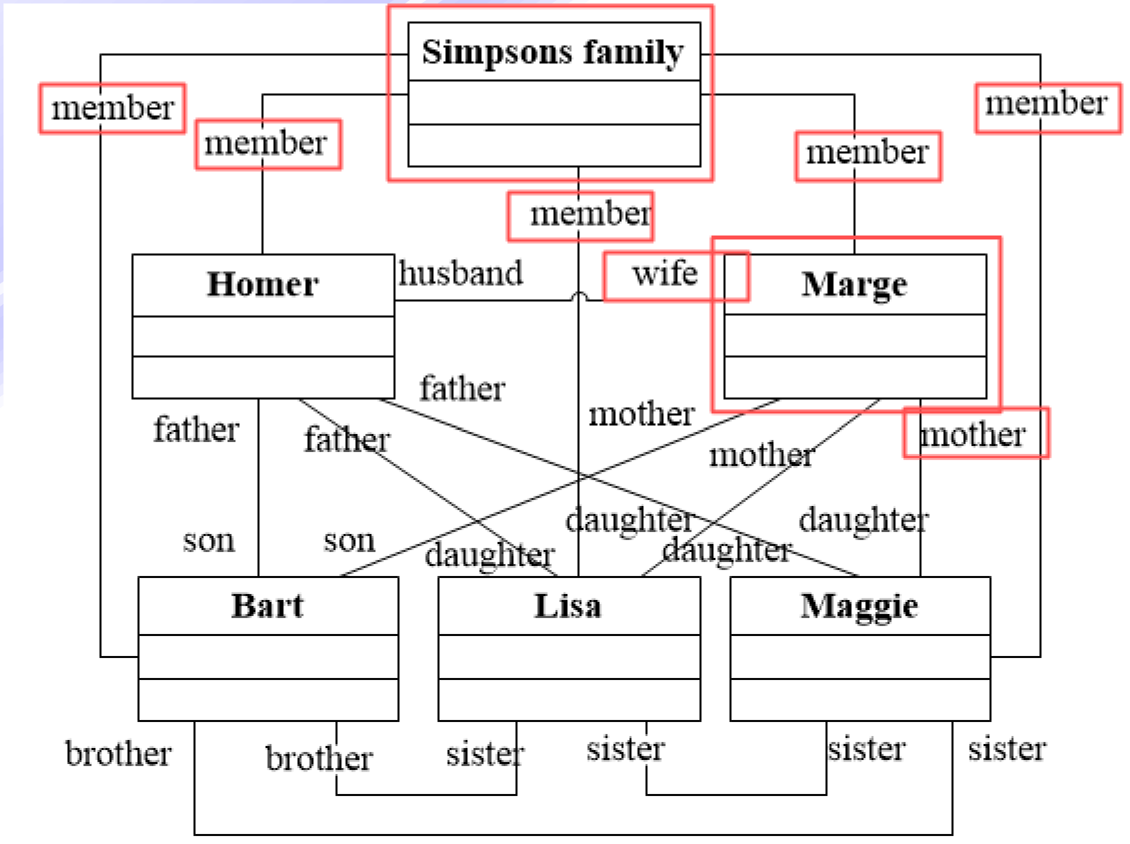


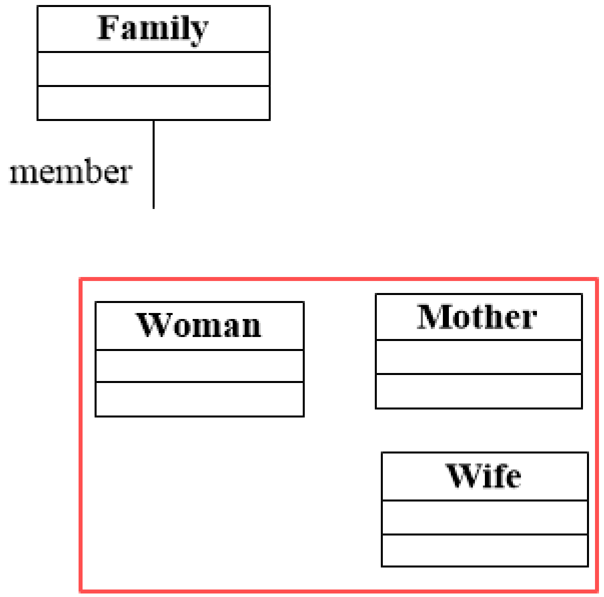




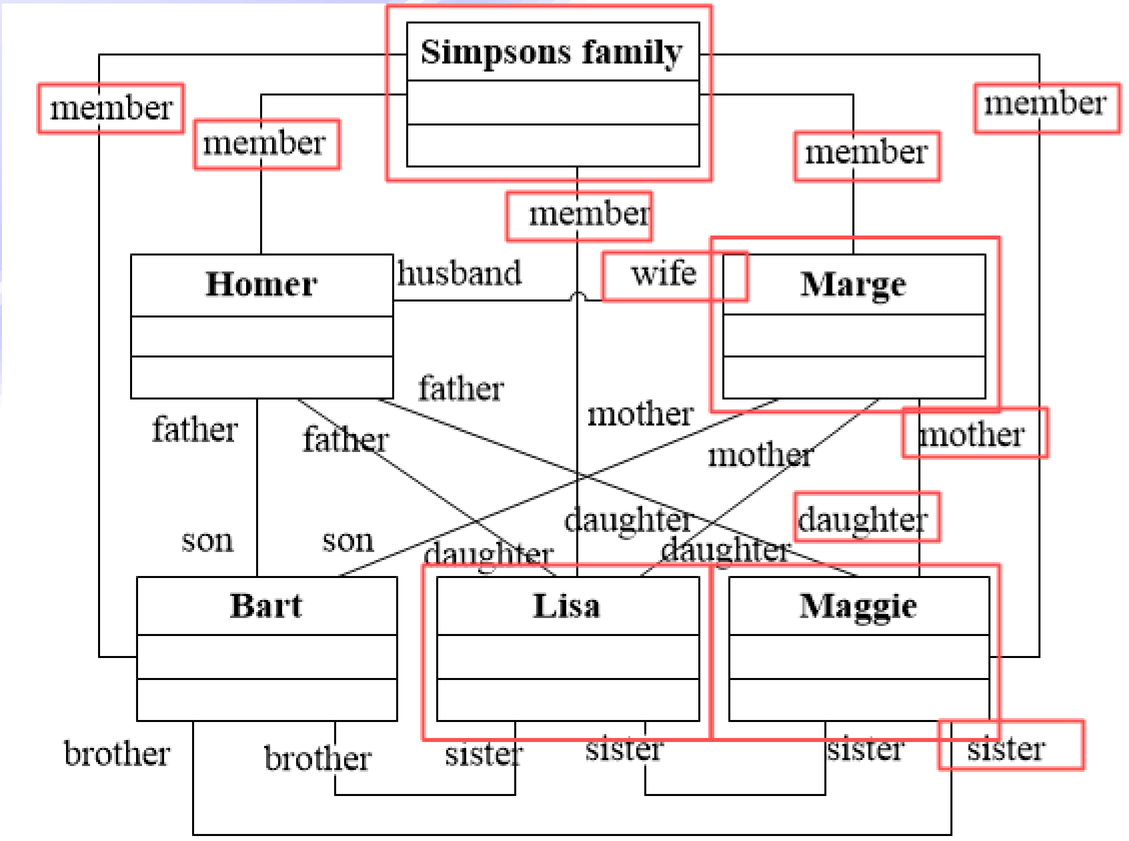


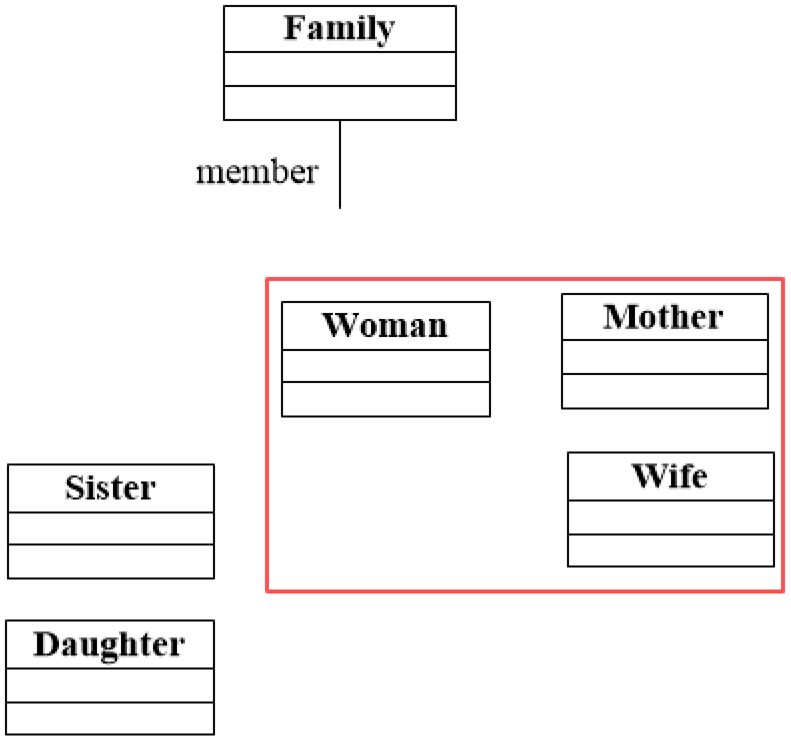
Designing a metamodel for a language to describe a family



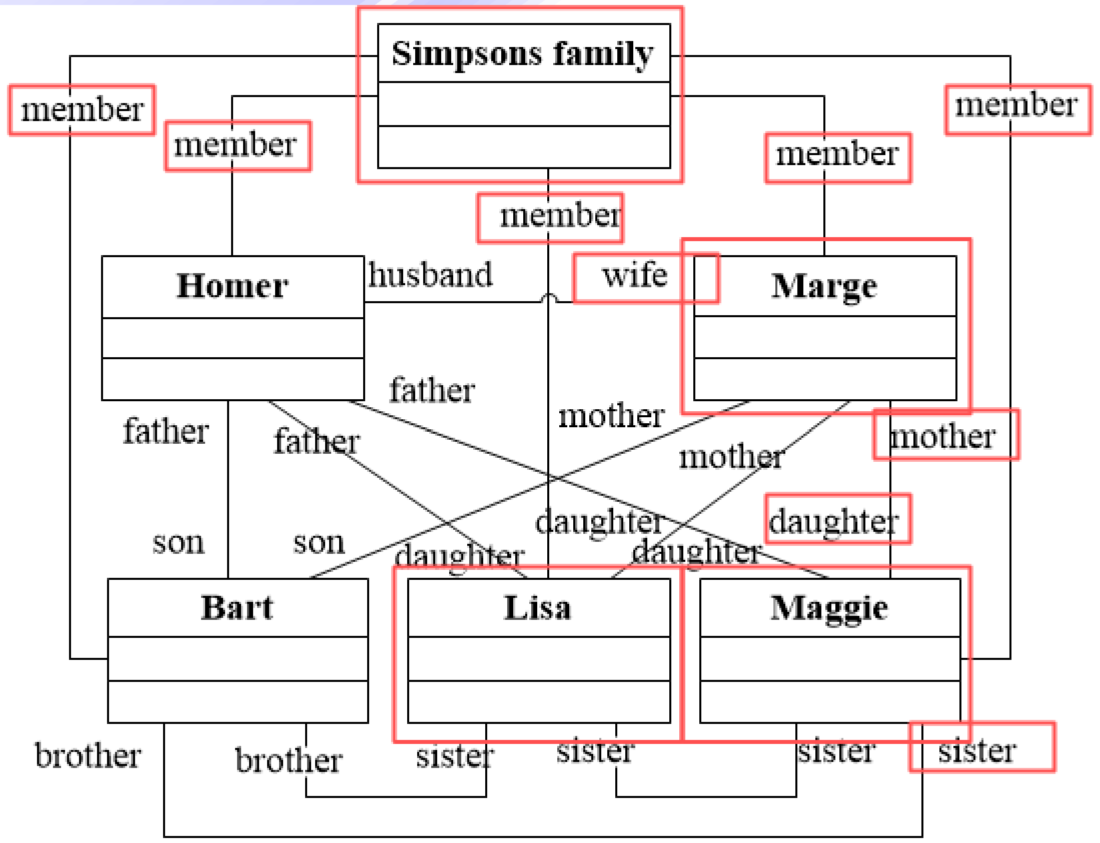


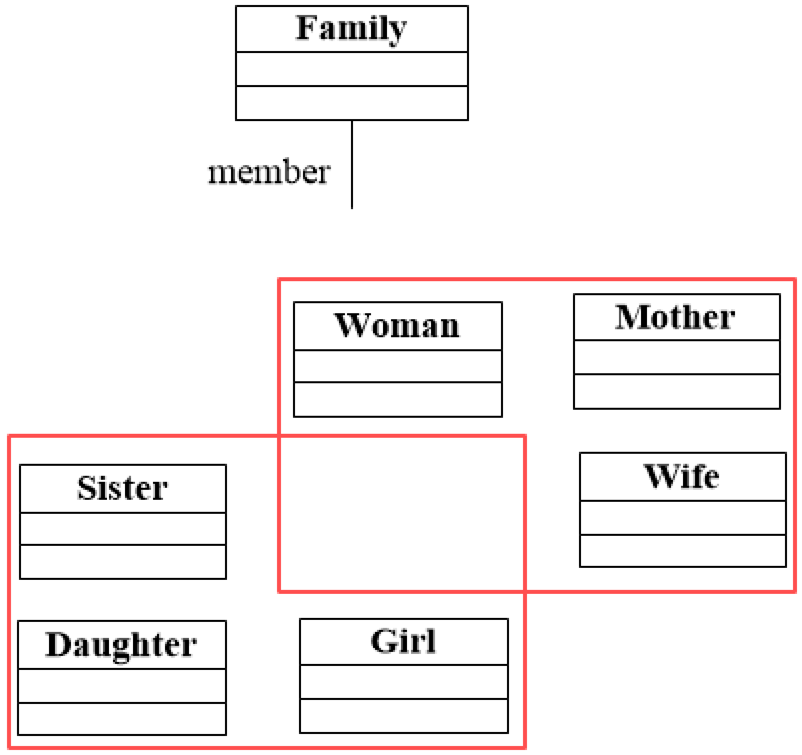
Designing a metamodel for a language to describe a family



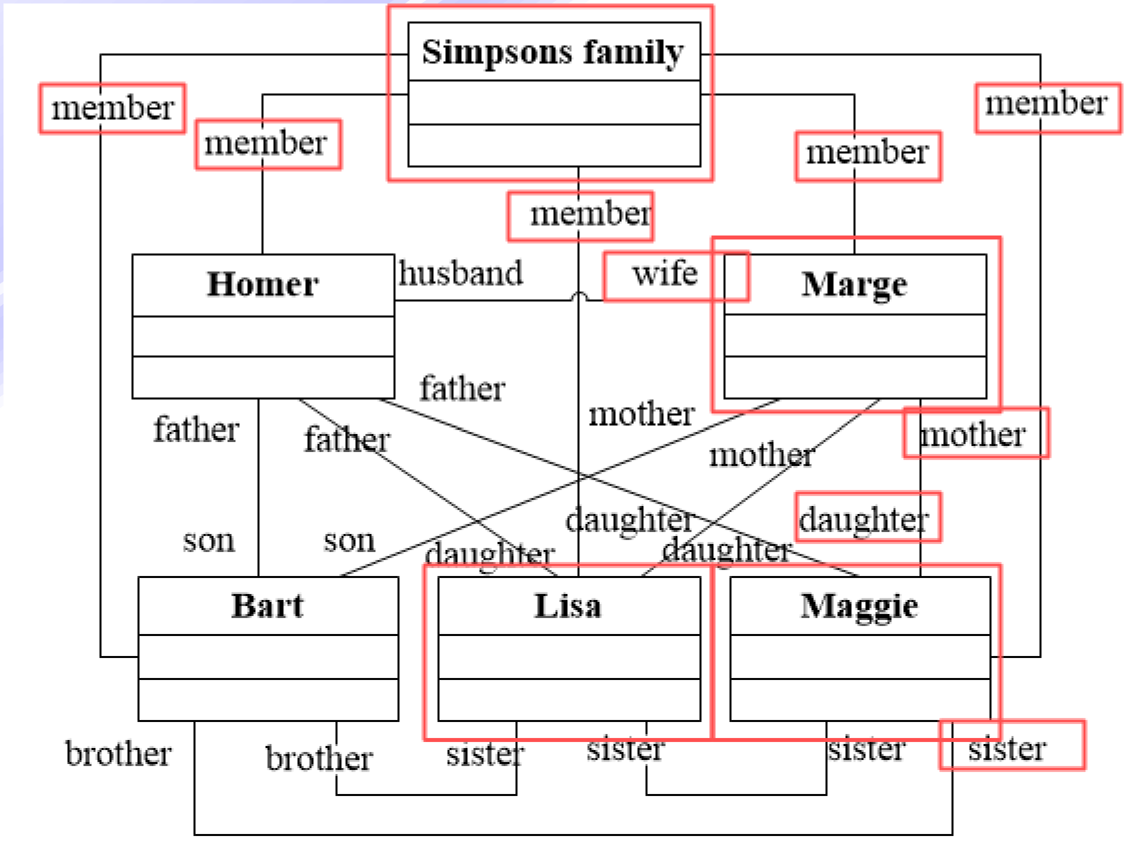


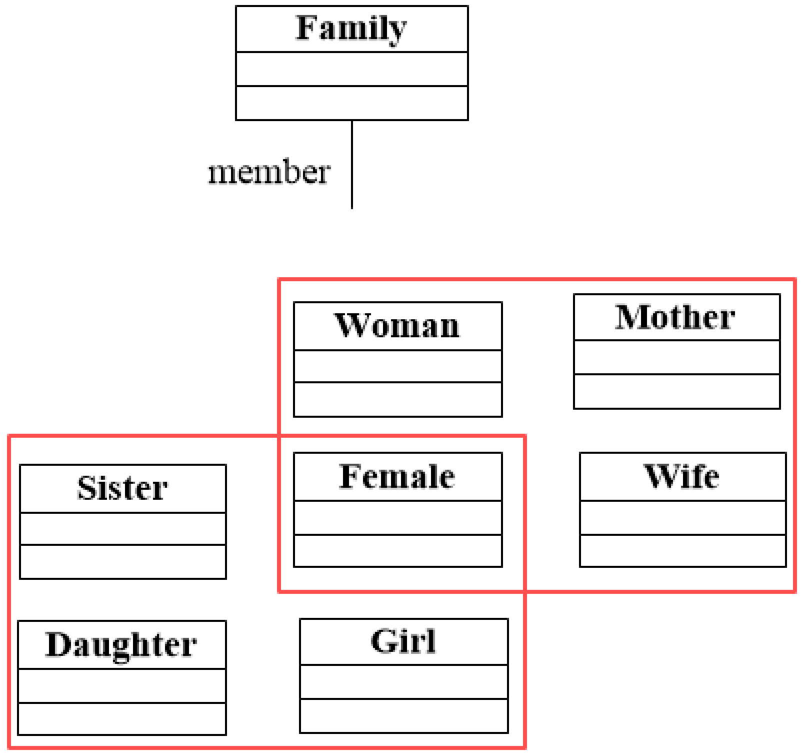
Designing a metamodel for a language to describe a family



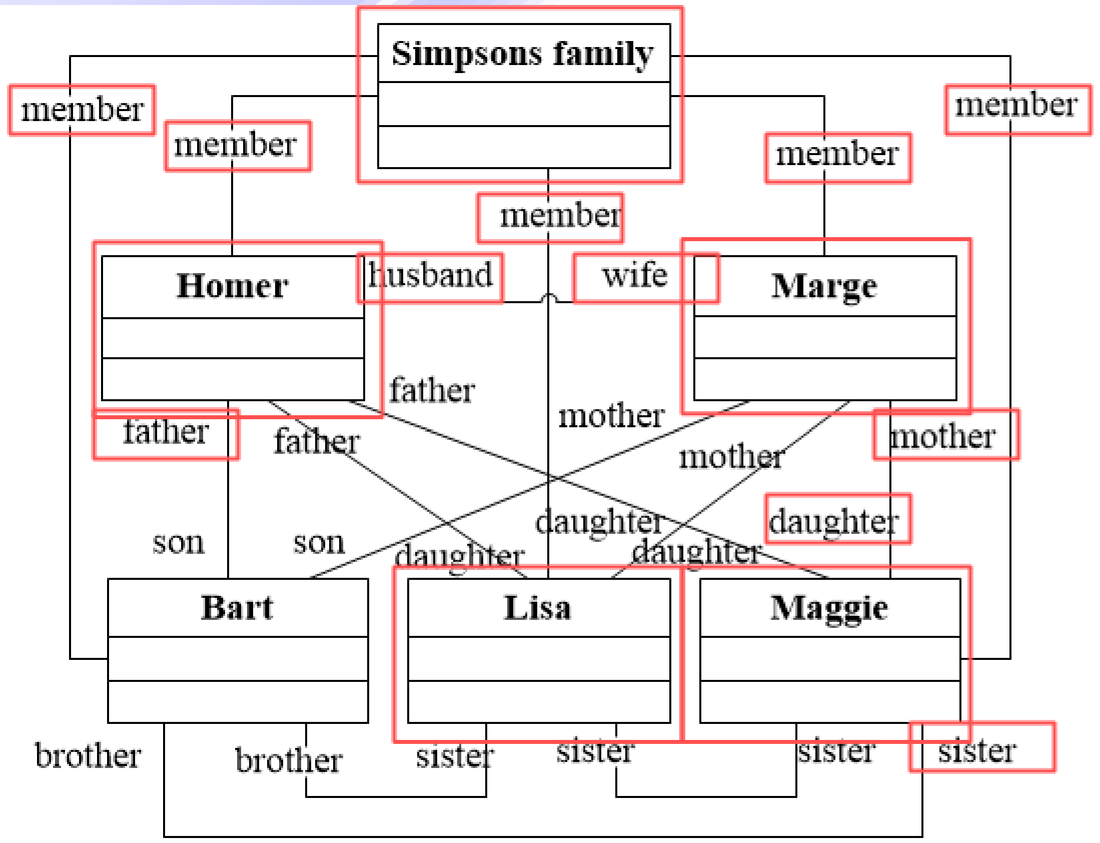


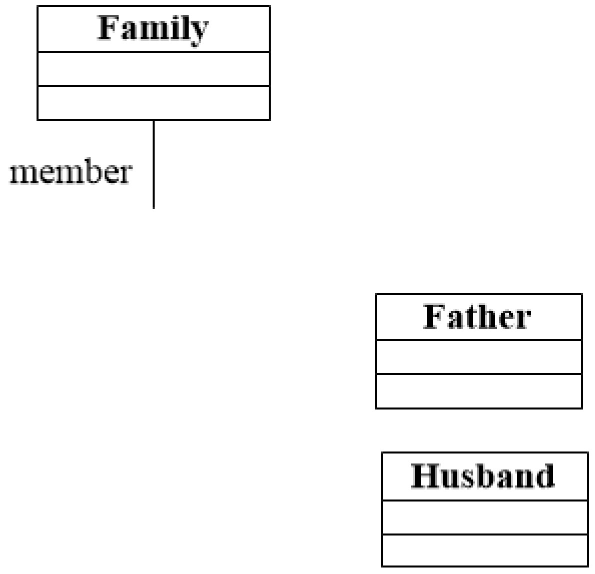
Designing a metamodel for a language to describe a family



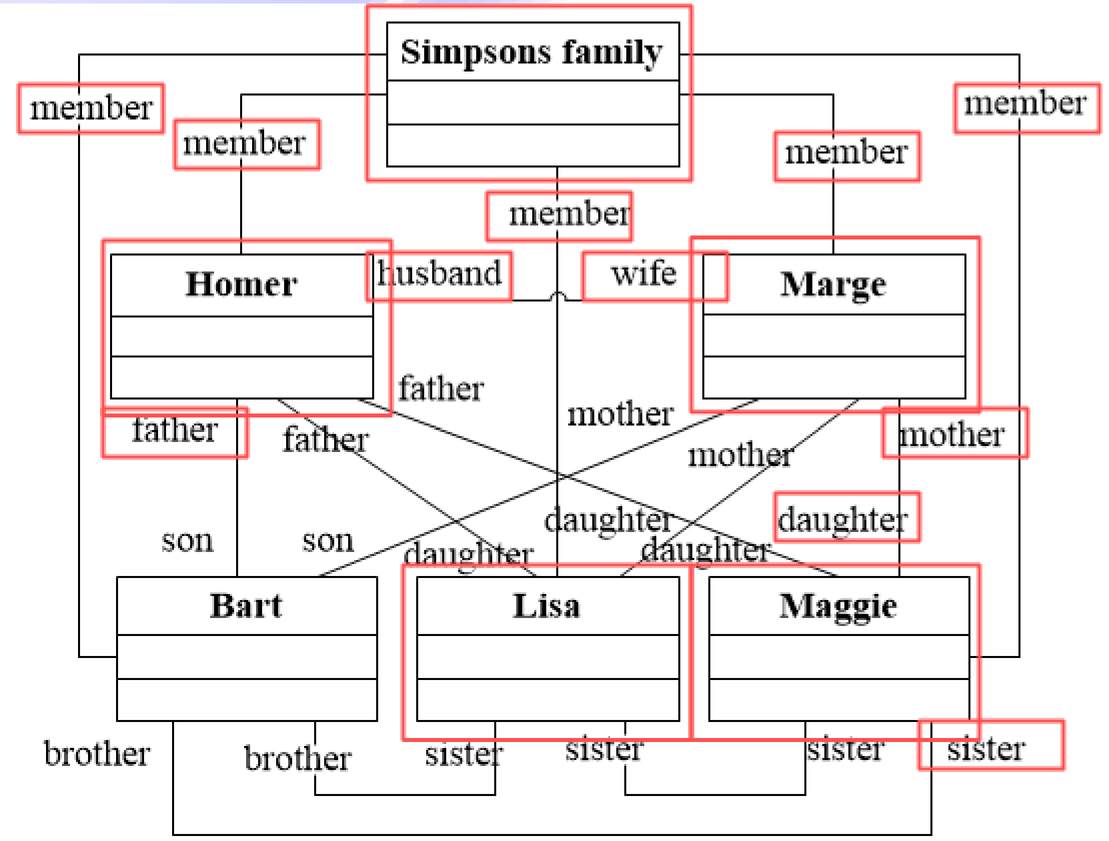


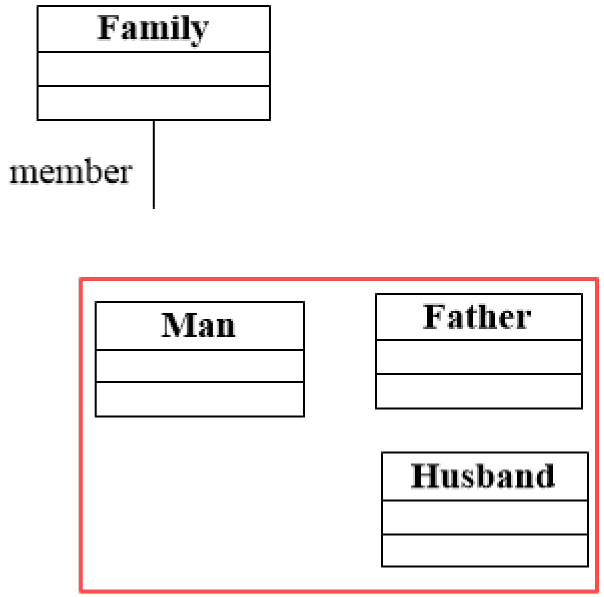
Designing a metamodel for a language to describe a family



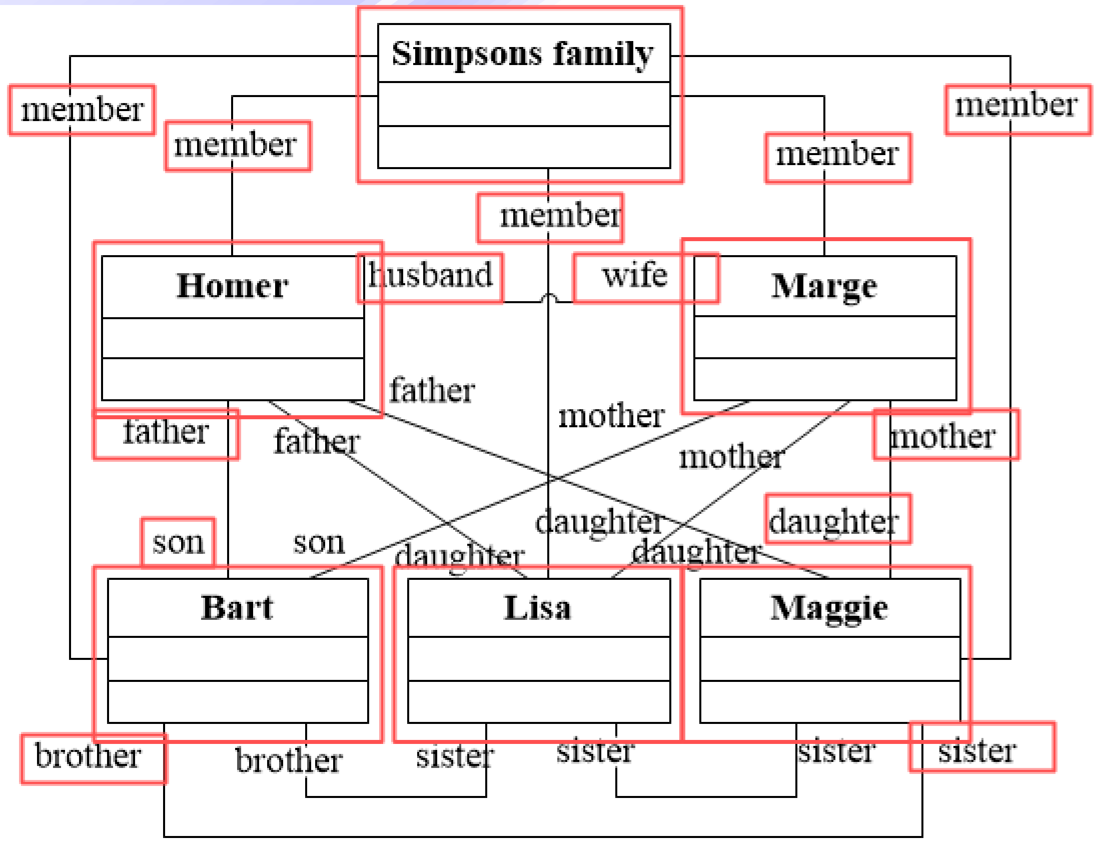


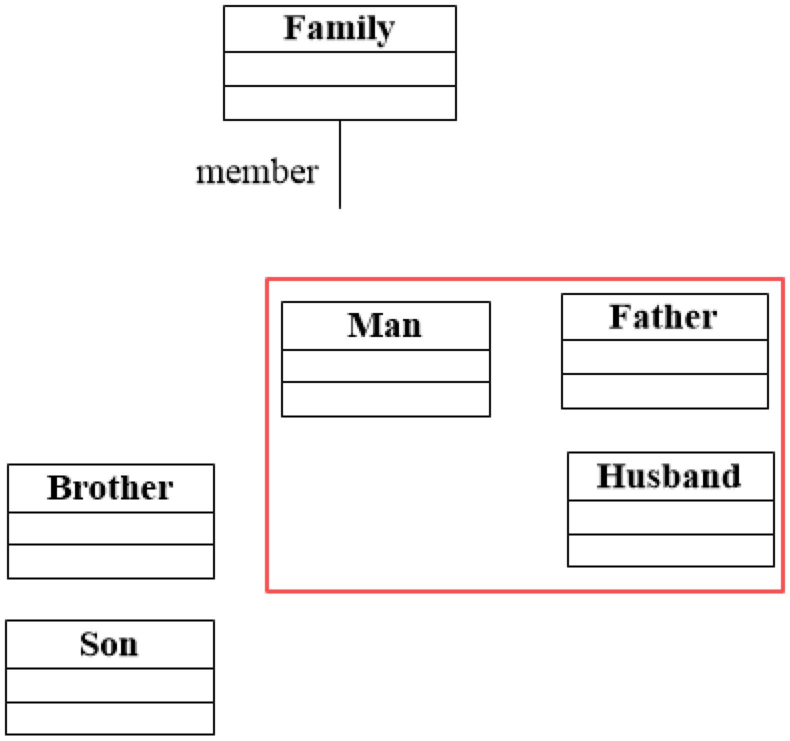
Designing a metamodel for a language to describe a family



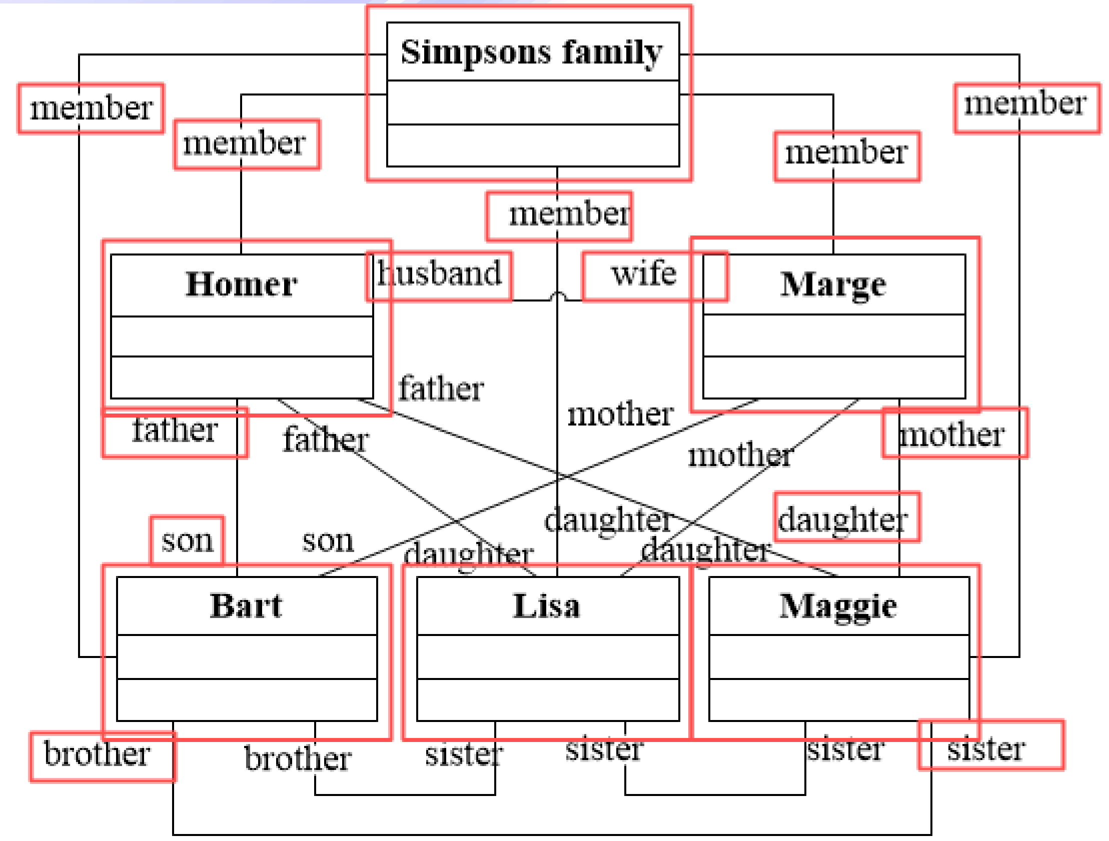


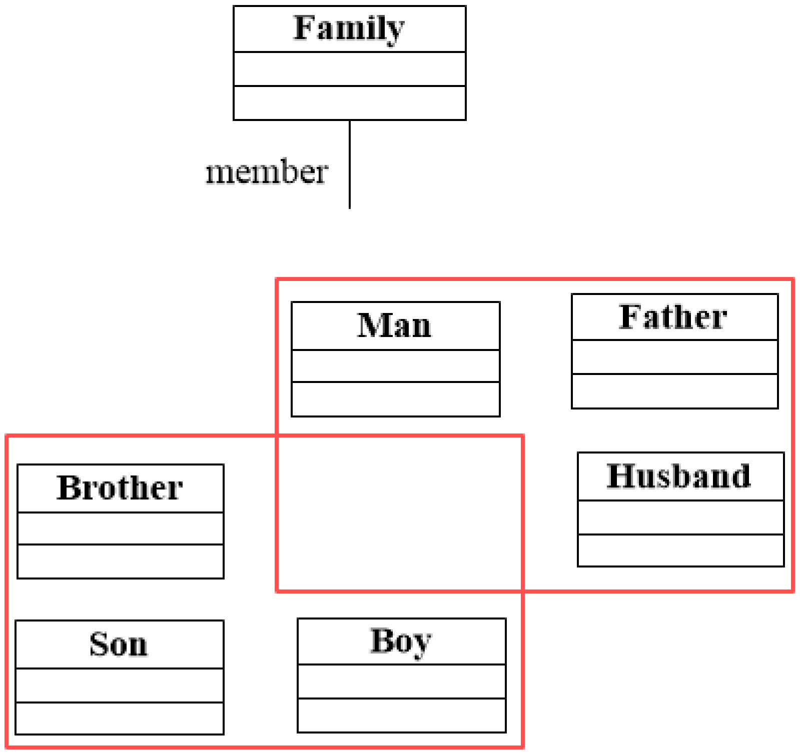
Designing a metamodel for a language to describe a family



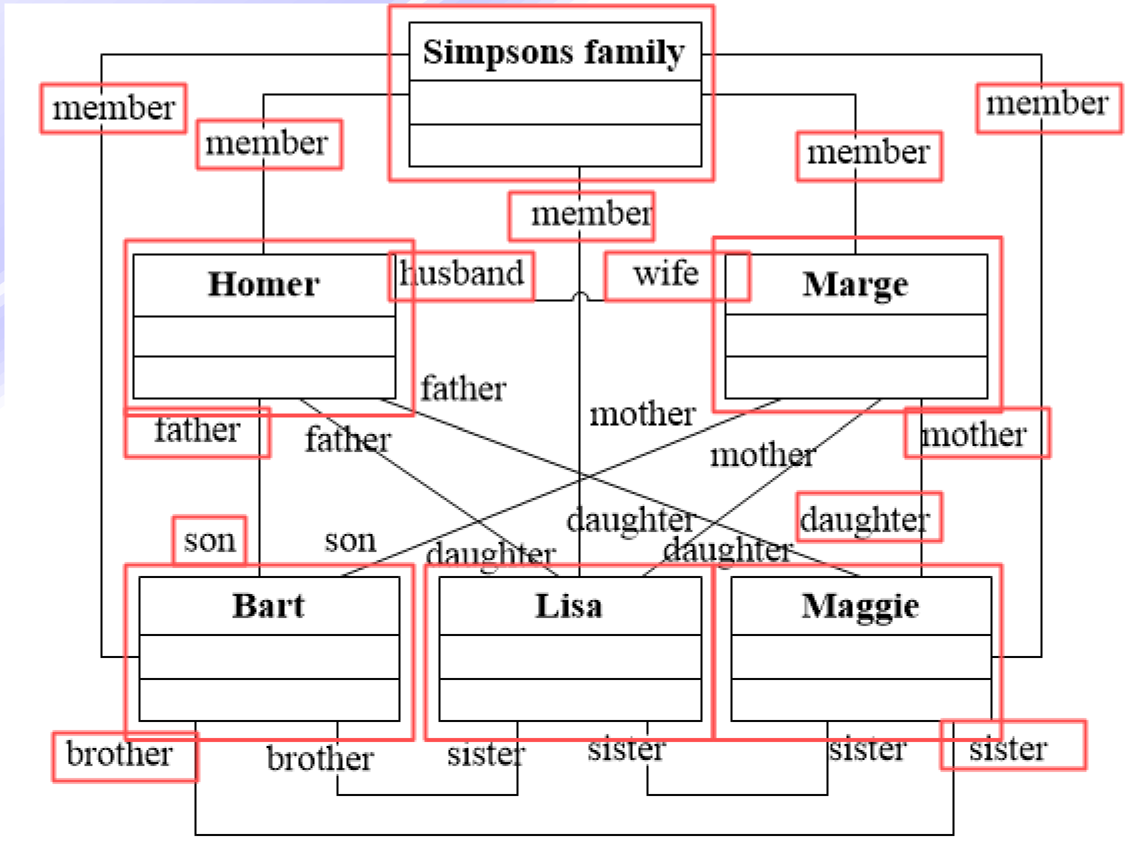


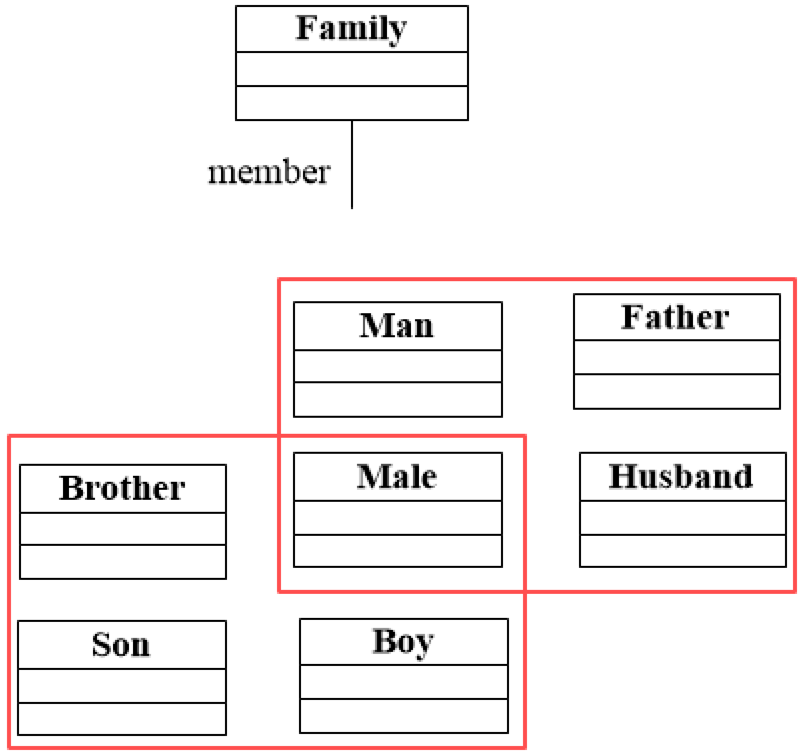
Designing a metamodel for a language to describe a family



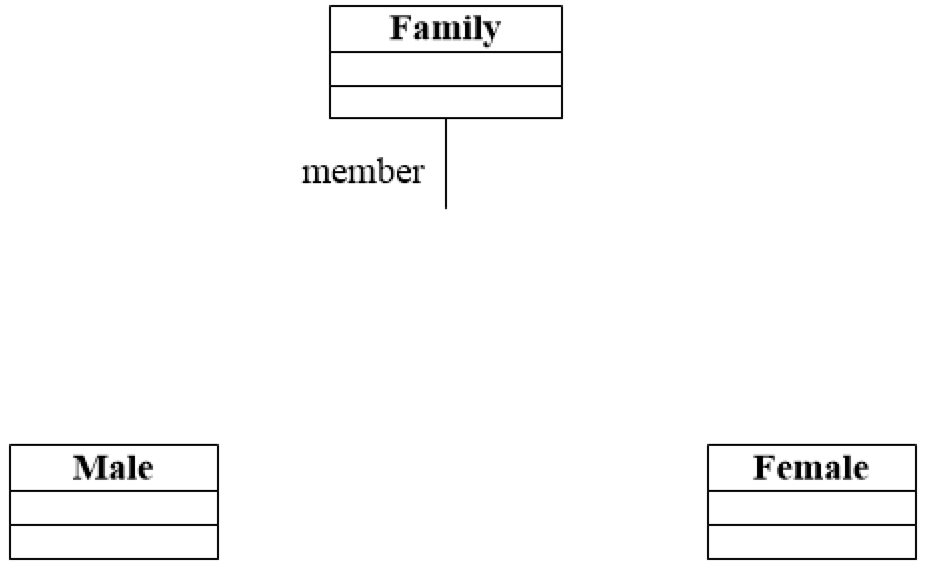


Designing a metamodel for a language to describe a family

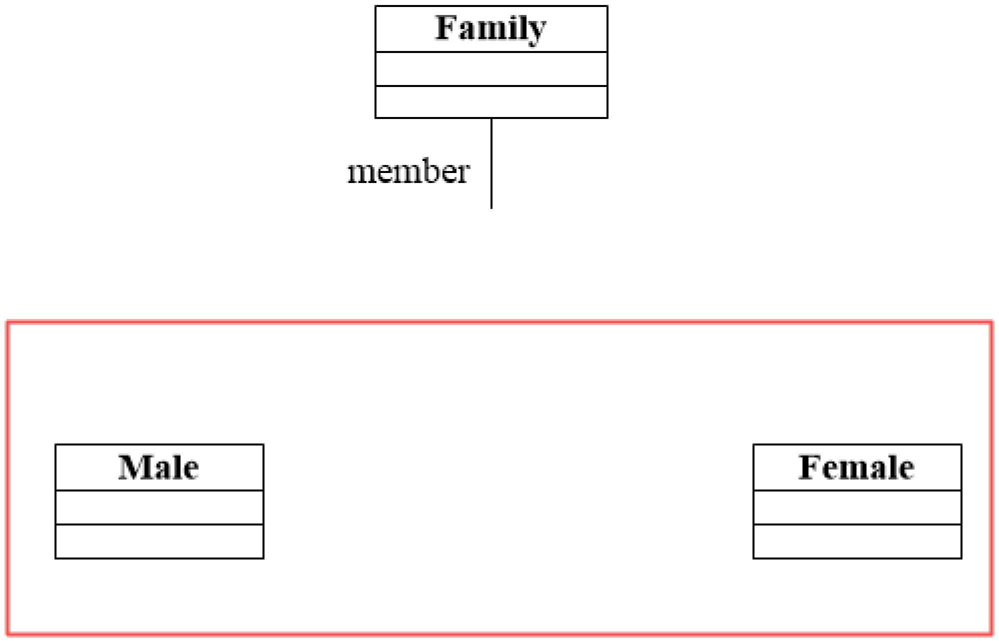




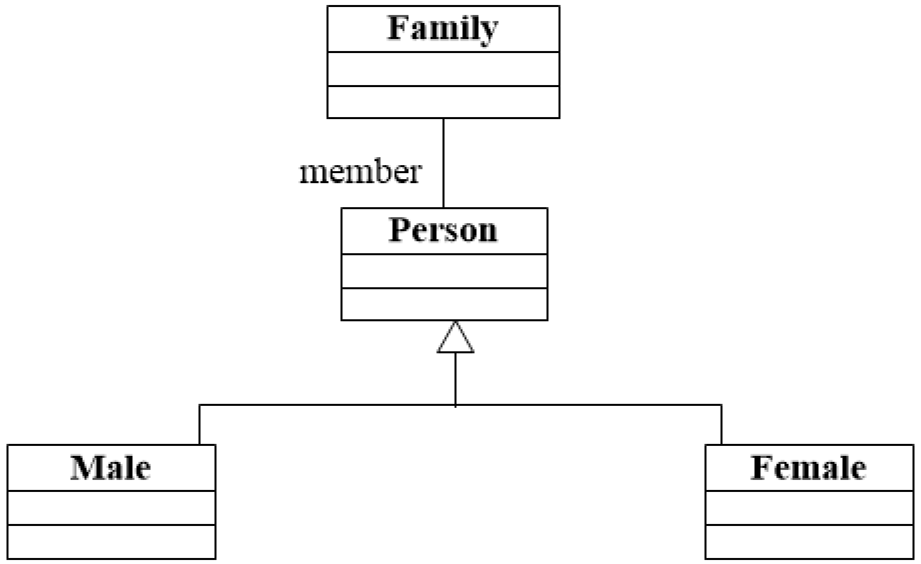
Designing a metamodel for a language to describe a family



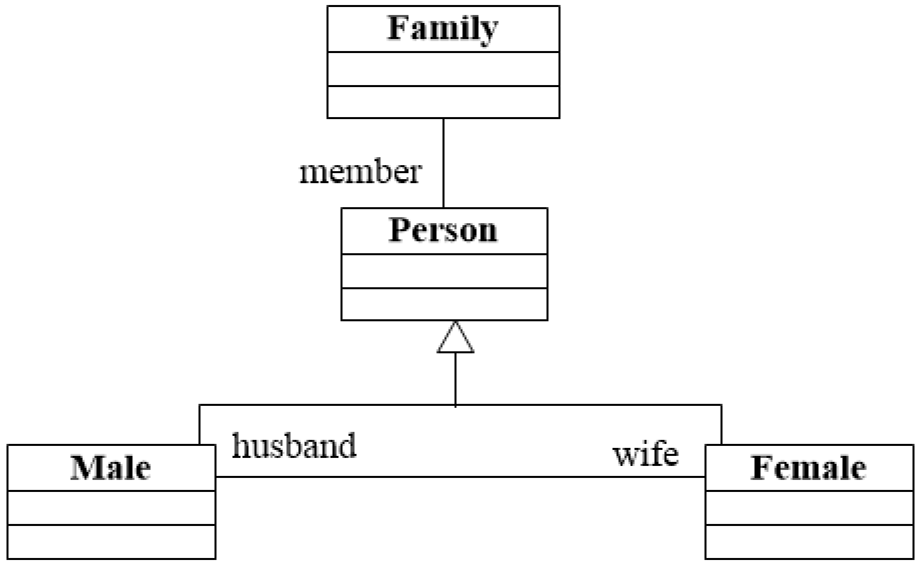
Designing a metamodel for a language to describe a family



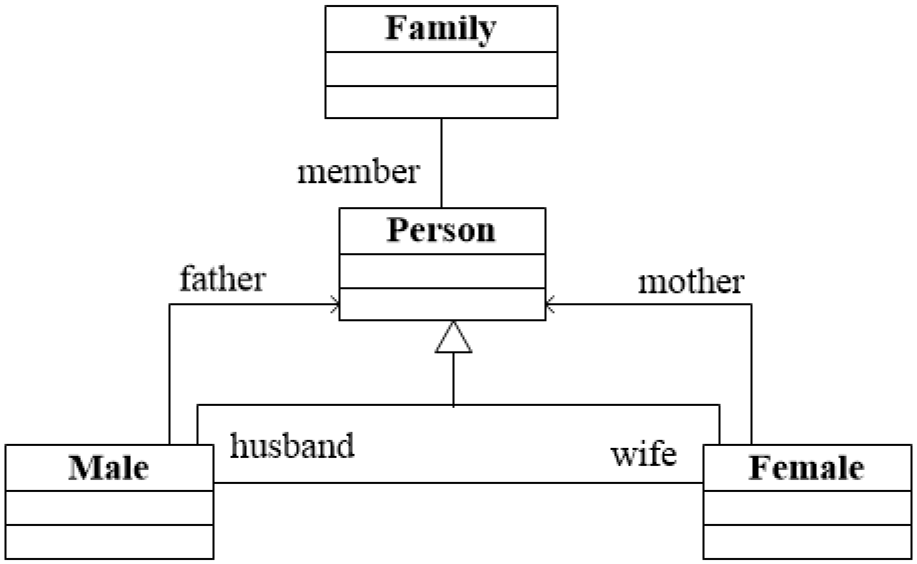
Designing a metamodel for a language to describe a family



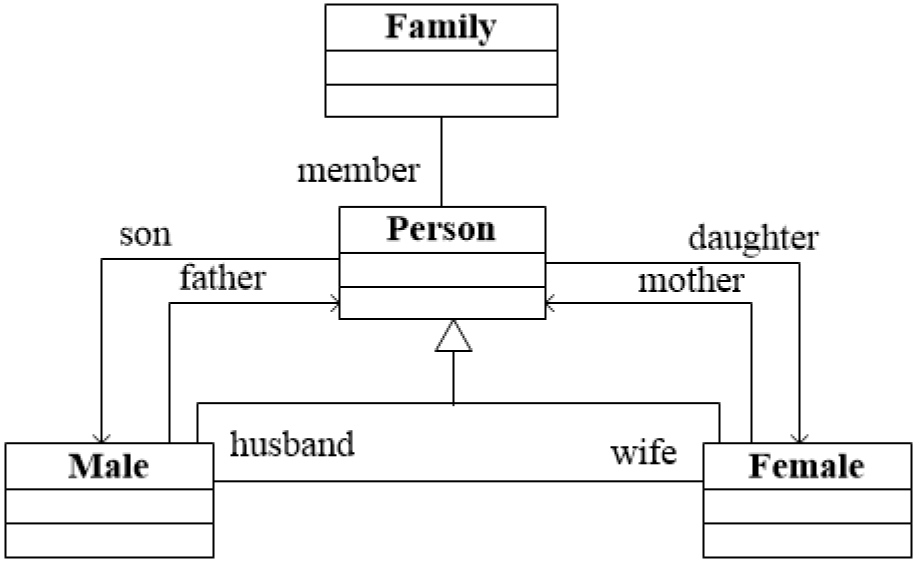
Designing a metamodel for a language to describe a family



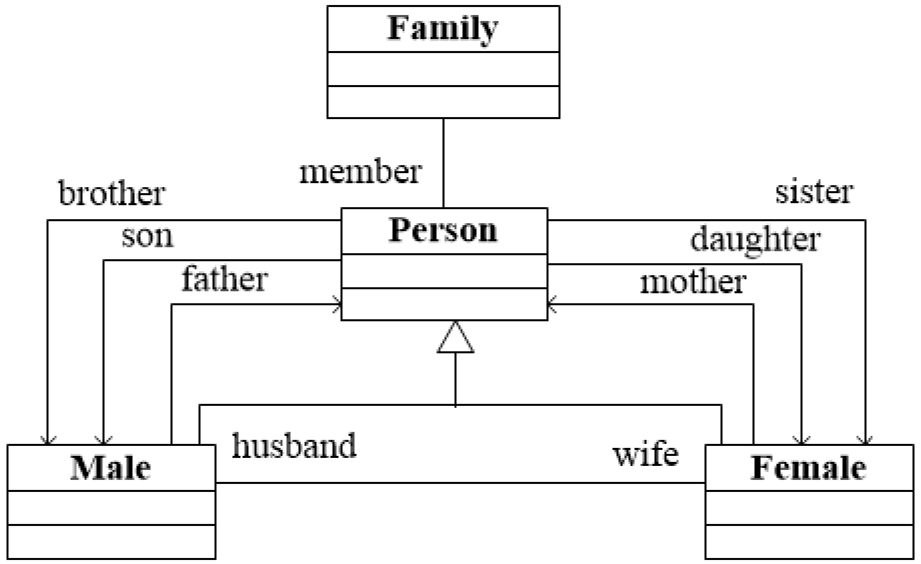
Designing a metamodel for a language to describe a family



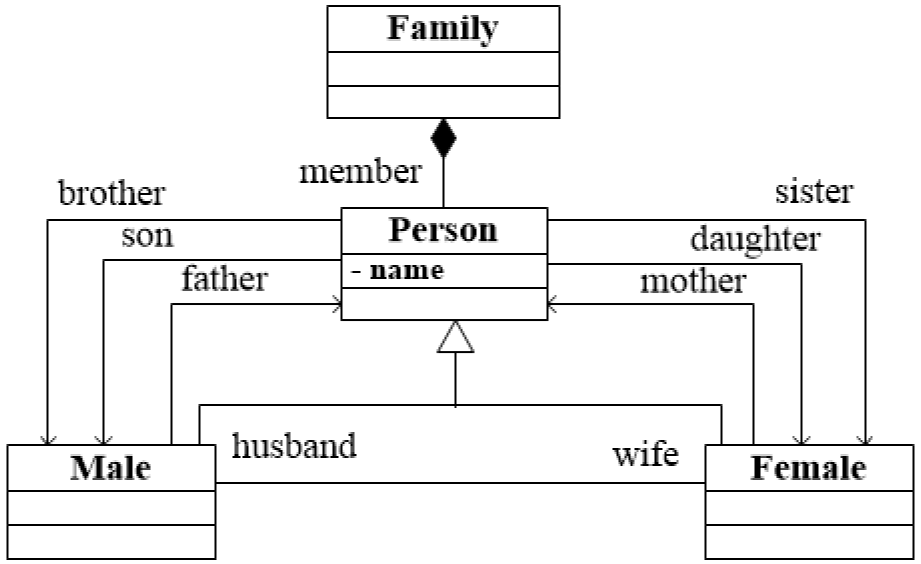
Designing a metamodel for a language to describe a family



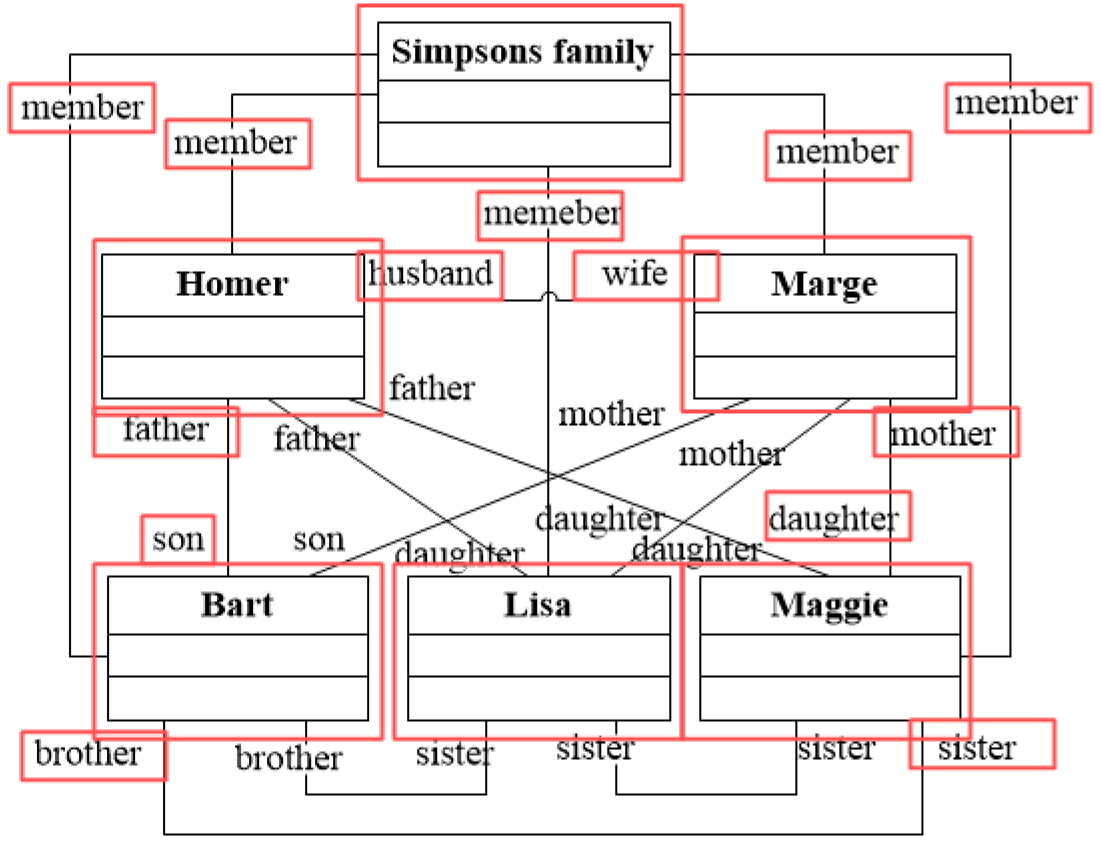
Designing a metamodel for a language to describe a family

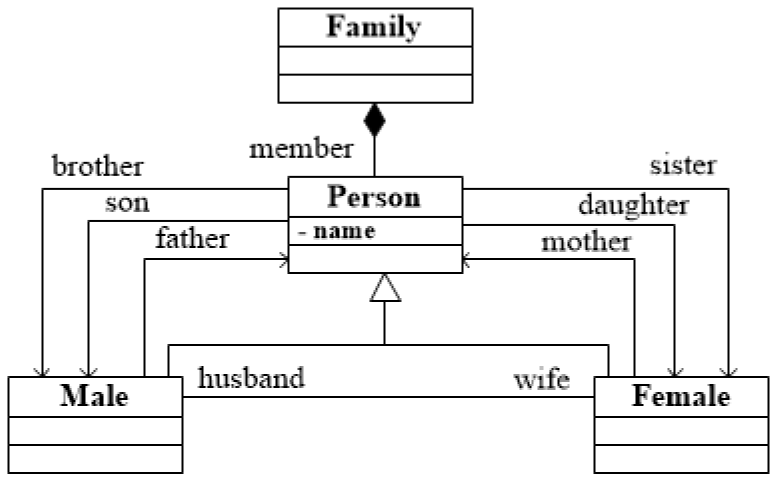


Designing a metamodel for a language to describe a family

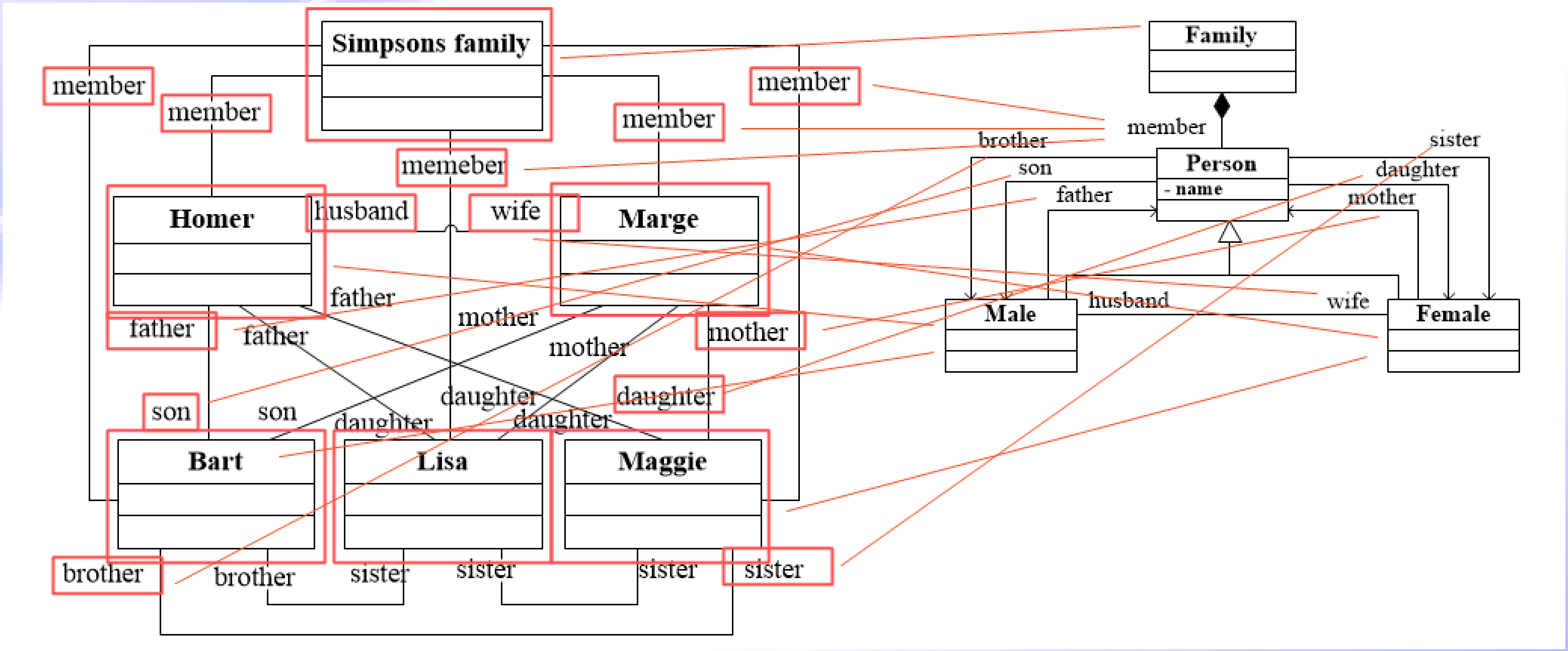


Designing a metamodel for a language to describe a family

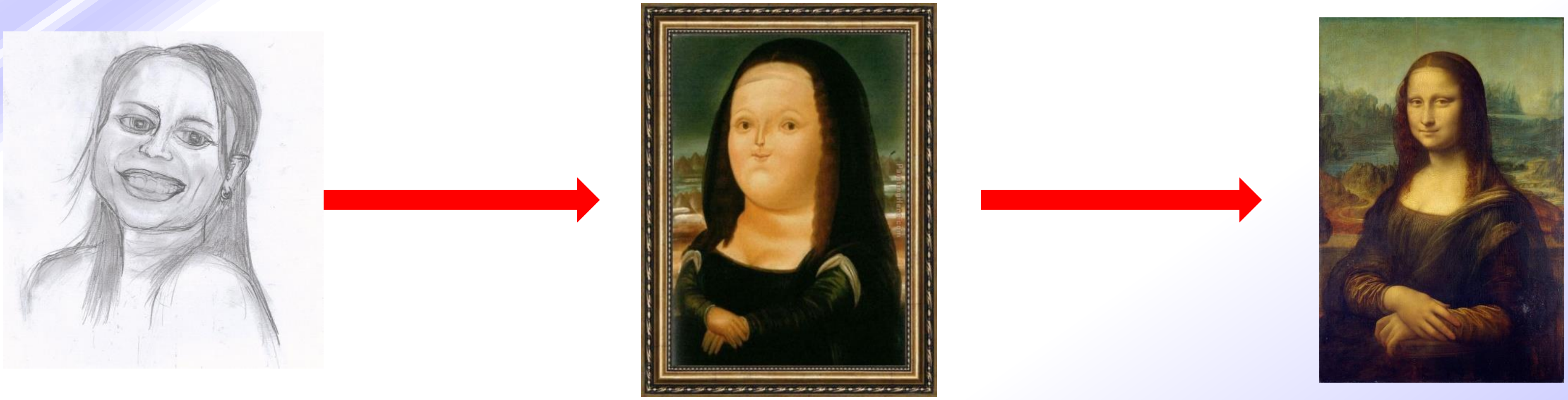




Designing a metamodel for a language to describe a family



The quality of a metamodel



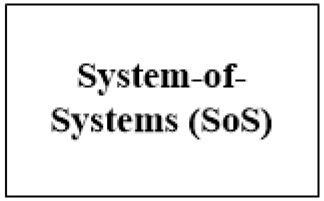
Designing a metamodel for CPSoS

- **System-of-Systems (SoS)** is an **integration** of a finite number of **Constituent Systems (CS)** which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.

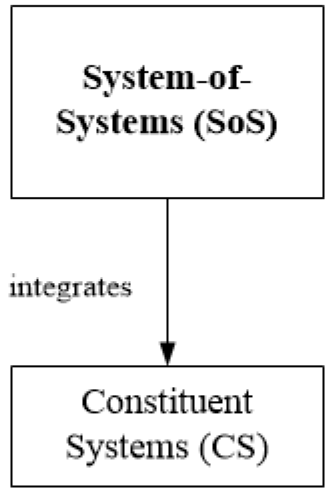
Designing a metamodel for CPSoS

- **System-of-Systems (SoS)** is an **integration** of a finite number of **Constituent Systems (CS)** which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.
- **A SoS** *integrates* **CSs.**

Designing a metamodel for CPSoS



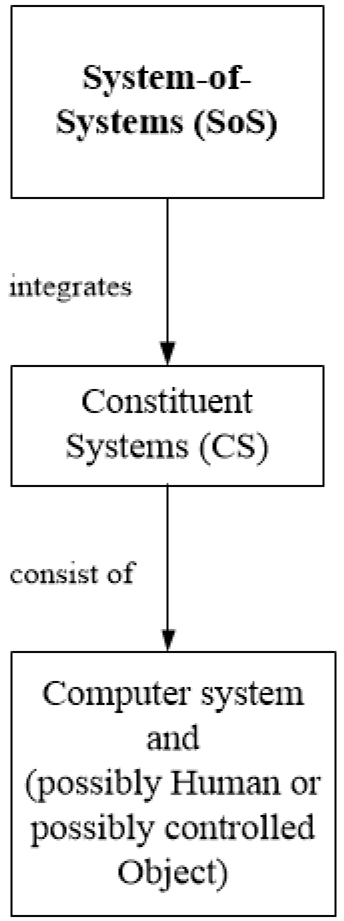
Designing a metamodel for CPSoS



Designing a metamodel for CPSoS

- **System-of-Systems (SoS)** is an **integration** of a finite number of **Constituent Systems (CS)** which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.
- **A SoS** *integrates* **CSs.**
- A **Constituent System (CS):** A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans

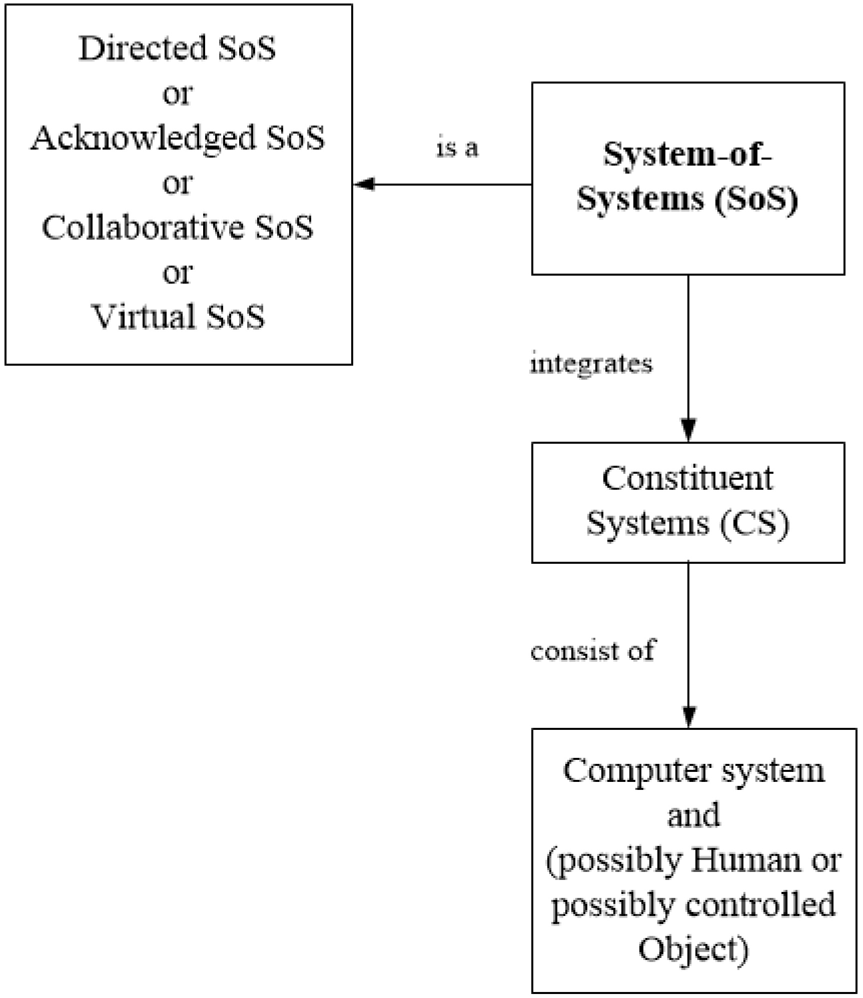
Designing a metamodel for CPSoS



Designing a metamodel for CPSoS

- **System-of-Systems (SoS)** is an **integration** of a finite number of **Constituent Systems (CS)** which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.
- **A SoS** *integrates* **CSs.**
- A **Constituent System (CS):** A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans
- A **SoS** can be:
  - **Directed SoS:** An SoS with a central managed purpose and central ownership of all CSs. An example would be the set of control systems in an unmanned rocket.
  - **Acknowledged SoS:** Independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.
  - **Collaborative SoS:** Voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS.
  - **Virtual SoS:** Lack of central purpose and central alignment.

Designing a metamodel for CPSoS



Designing a metamodel for CPSoS

- Each **CS** has an **interface** , where the services are offered to other CSs, namely:

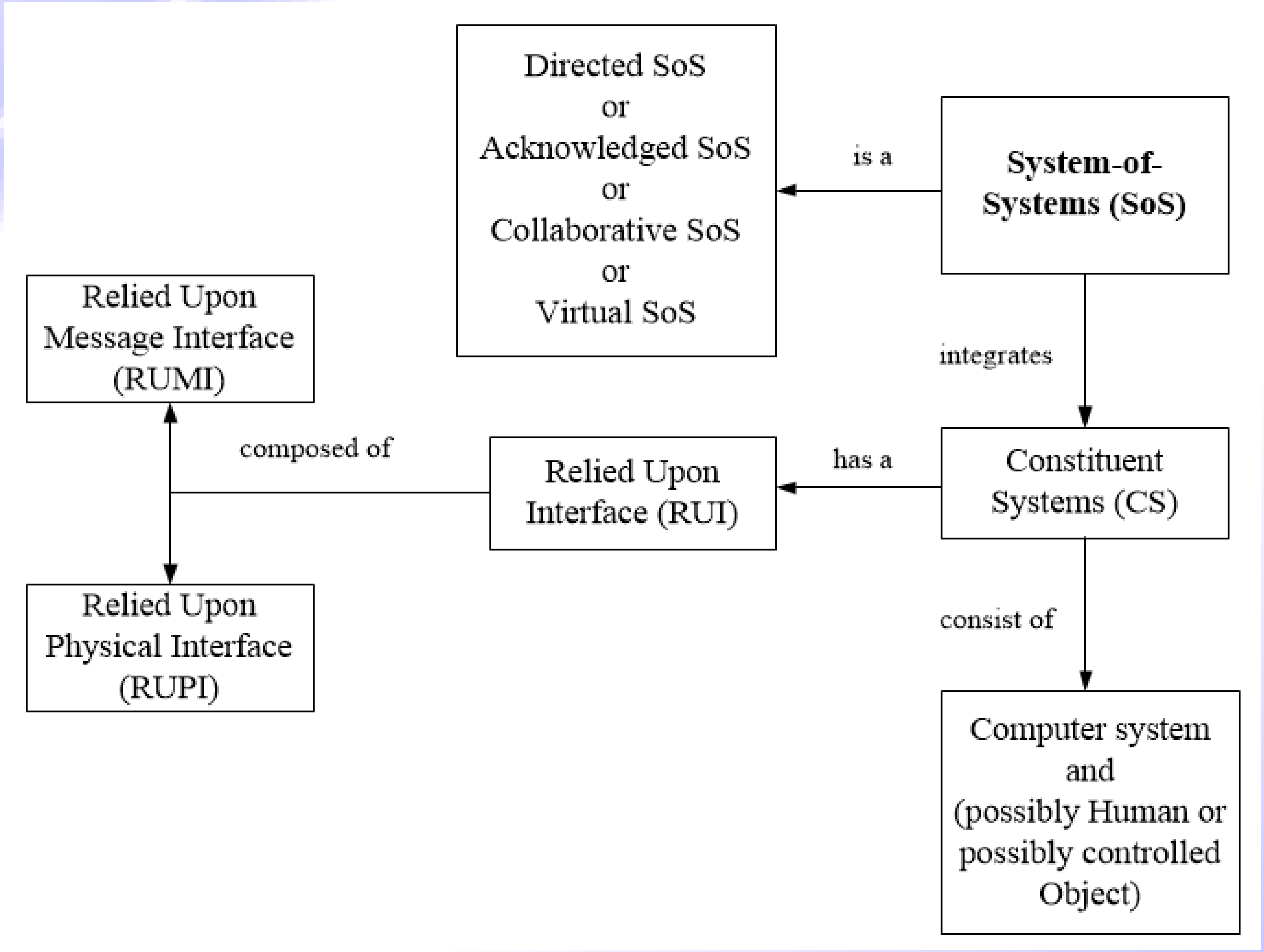
Relied upon Interface (RUI): An interface of a CS where the services of the CS are offered to other CSs.

RUI is *composed of :*

- **Relied upon Message Interface (RUMI):**

- **Relied upon Physical Interface (RUPI):**

Designing a metamodel for CPSoS


Directed SoS or Acknowledged SoS or Collaborative SoS or Virtual SoS
is a
System-of-Systems (SoS)
integrates
Constituent Systems (CS)
has a
Relied Upon Interface (RUI)
composed of
Relied Upon Message Interface (RUMI)
Relied Upon Physical Interface (RUPI)
consist of
Computer system and (possibly Human or possibly controlled Object)

Designing a metamodel for CPSoS

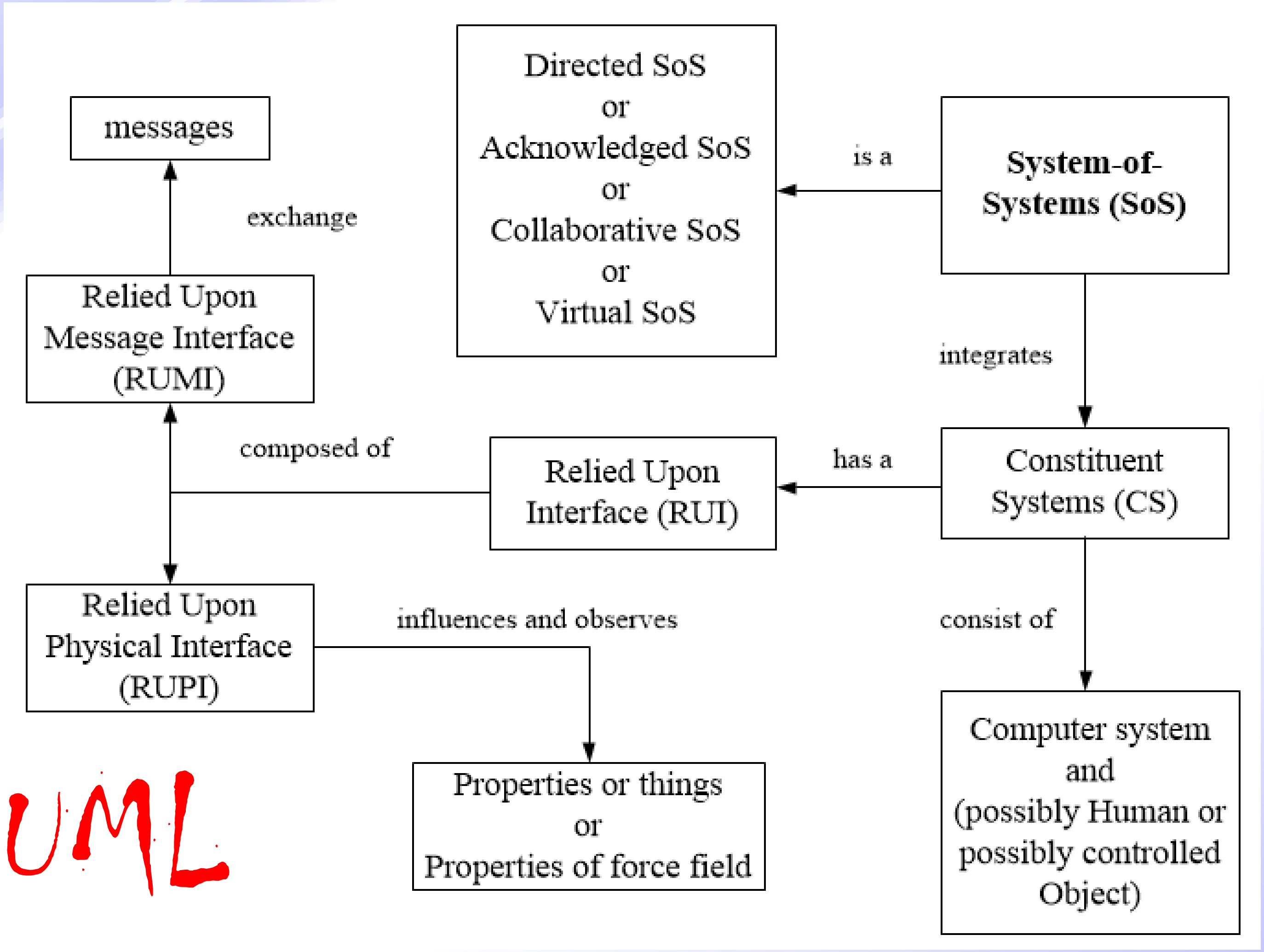
- Each **CS** has an **interface** , where the services are offered to other CSs, namely:

Relied upon Interface (RUI): An interface of a CS where the services of the CS are offered to other CSs.

RUI is *composed of :*

- **Relied upon Message Interface (RUMI):** A message interface where the services of a CS are offered to the other CSs of an SoS.
- **Relied upon Physical Interface (RUPI):** A physical interface where things or energy are exchanged among the CSs of an SoS.

Designing a metamodel for CPSoS


messages
exchange
Relied Upon Message Interface (RUMI)
composed of
Relied Upon Physical Interface (RUPI)
influences and observes
Properties or things or Properties of force field
Directed SoS or Acknowledged SoS or Collaborative SoS or Virtual SoS
is a
System-of-Systems (SoS)
integrates
Constituent Systems (CS)
has a
Relied Upon Interface (RUI)
consist of
Computer system and (possibly Human or possibly controlled Object)

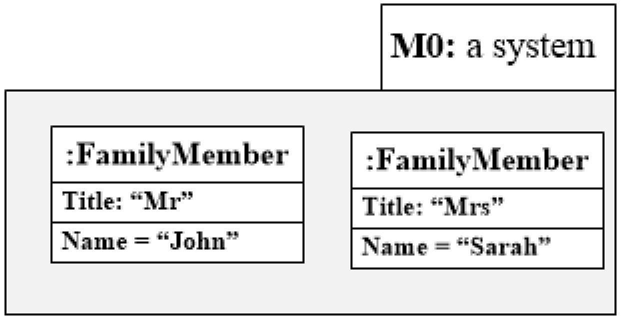
Not UML

Meta-Modeling and the OMG Meta Object Facility (MOF)

- MOF is an OMG standard for modeling languages
  - It is a model of metamodels (a meta-metamodel).
  - All UML modelling concepts can be represented within the MOF.
  - UML modelling concepts are defined as "metaclasses", i.e., metaclasses themselves are instance objects of MOF classes.
- The MOF has a 4-layer architecture: M0, M1, M2, and M3.

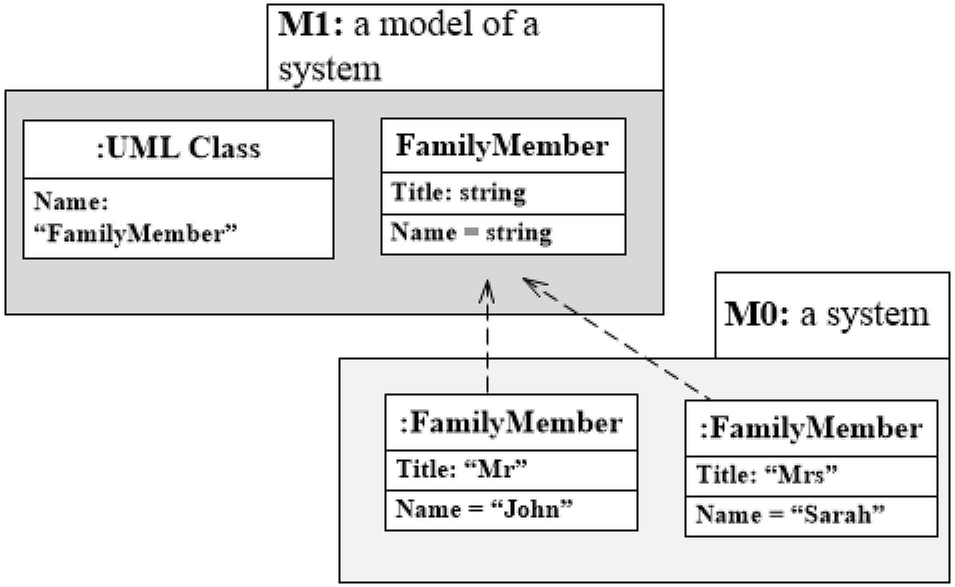
Meta-Modeling and the OMG Meta Object Facility (MOF) - M0

- Layer M0 defines an **actual** system.
  - Instances and/or executing instances
  - E.g., component instances, customer objects.



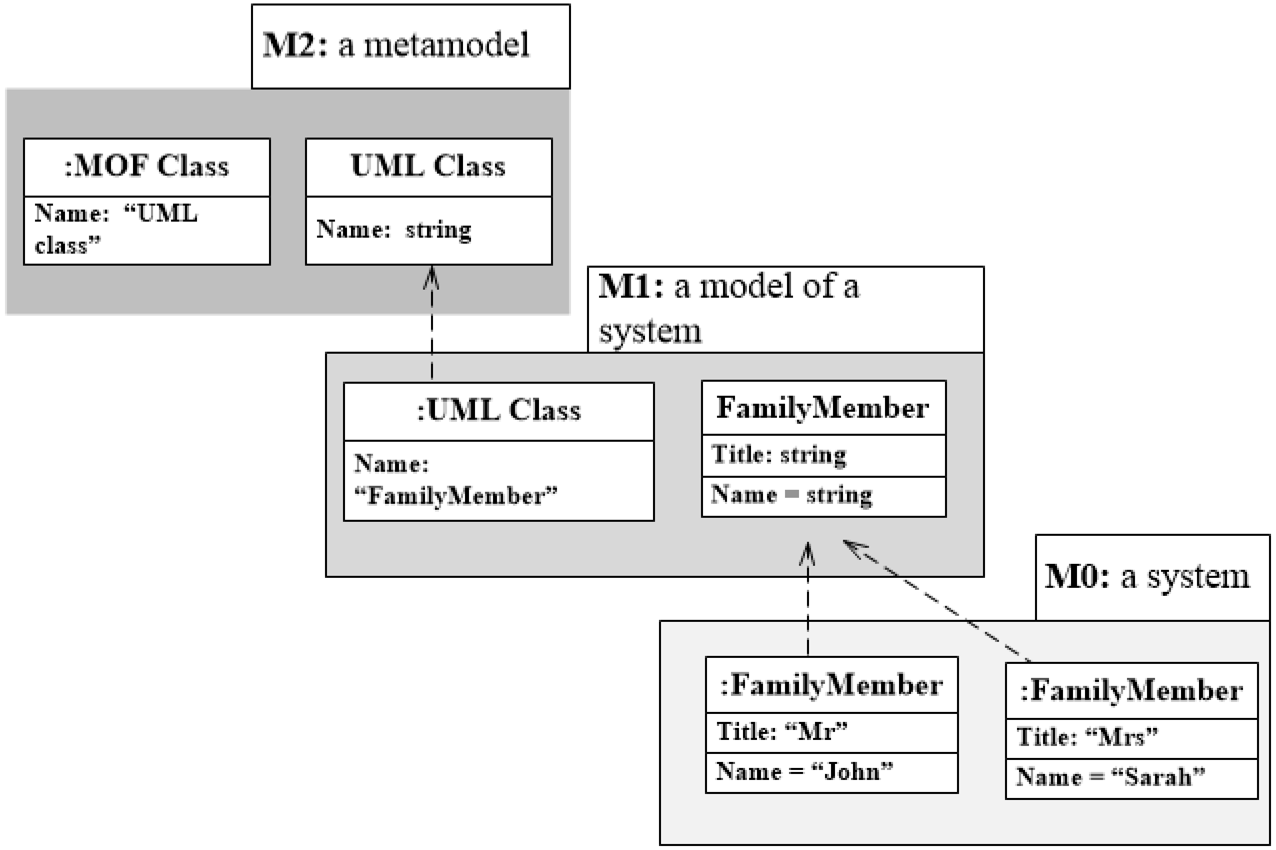
Meta-Modeling and the OMG Meta Object Facility (MOF) - M1

- Layer M1 is a system model.
  - Defines the types of entities and relationships that make up a system
  - E.g., a UML class model.
- Every element of M0 is an instance of M1



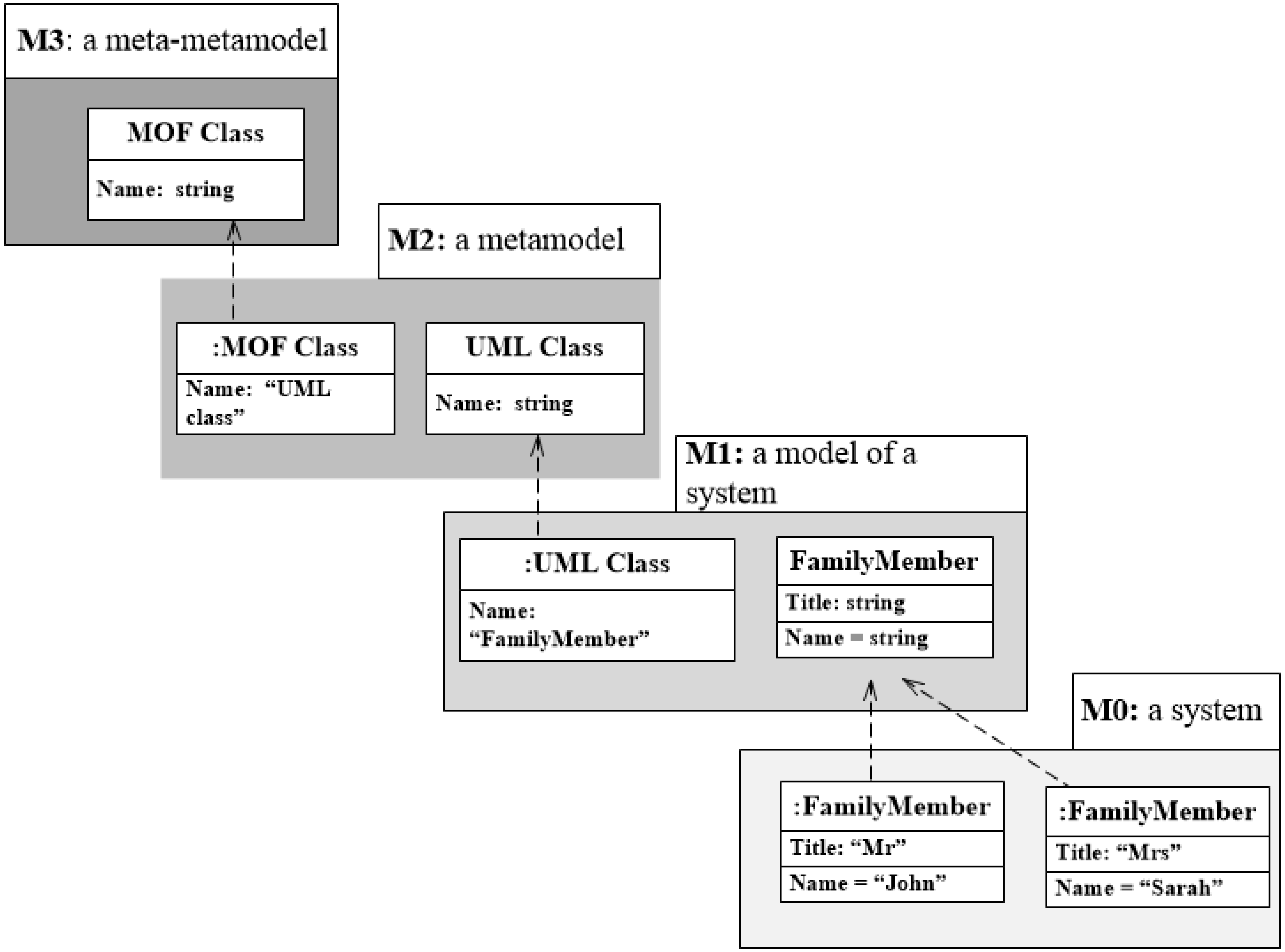
Meta-Modeling and the OMG Meta Object Facility (MOF) – M2

- Layer M2 defines the metamodel in M1
  - E.g., language used to make models in M1 defined by a model in M2.
- Every element of M1 is an instance of M2



Meta-Modeling and the OMG Meta Object Facility (MOF) – M3

- Layer M3 defines the model of metamodels in M2 (the meta-metamodel).
- The metaclasses of M2 are instances of M3 classes.
- M3 classes are called MOF classes.



Selected References for Reading

[1] Sprinkle, Jonathan, et al. 3. Metamodelling." Model-Based Engineering of Embedded Real-Time Systems. Springer, Berlin, Heidelberg, 2010. 57-76.

[2] Schichl, Hermann. "Models and the history of modeling." *Modeling languages in mathematical optimization*. Springer, Boston, MA, 2004. 25-36.

[3] OMG. 2001. OMG Unified Modeling Language specification, version 1.4. OMG document ad/00-11-01.