Part II

The MDM Tool

with the cooperation of

Pasquale Racca and **Luca Sabolo**

Basic Features of MDM

MDM is a tool that enables the users to:

- **define schemes** of <u>different</u> data models;
- **perform translations** of schemes from one model to another.

The tool can be at the basis of a customizable and integrated CASE environment supporting the analysis and design of information systems.

Technical issues of MDM

A prototype of MDM has been developed at *Università di Roma Tre*.

The current version is implemented:

- in **Visual C++**;
- under **Windows95**.

A graphical interface is currently under development. It is based on *Diagram Server*, a tool developed at the University of Rome "La Sapienza" that allows the editing and the automatic layout of complex diagrams.

A porting to Motif under Unix has been planned.

Main Components of MDM

- **Model Manager**: supports a specialized user, which we call *model engineer*, in the definition of a variety of models, on the basis of a limited set of metaconstructs covering almost all known conceptual models.
- **Schema Manager**: allows designers to create and modify schemes over the defined models, and to generate the translation of a scheme into any of the data models currently available.

Main functionalities of MDM

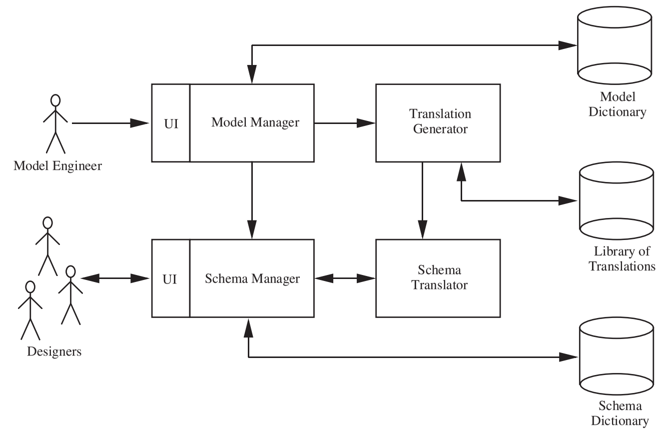
- Definition of a new model by means of a menu-driven **Model Definition Language**.

  The system is able to generate (according to a request done by the model engineer) a standard translation for it: from the "Supermodel" to the new model.

- Specification of a scheme for one of the defined model by means of a menu-driven **Schema Definition Language**.

- Request for a translation of a scheme into a specific target model.

The Architecture of MDM



Components of the Architecture (1)

- **User Interface**. This part allows the interaction with the system by means of menus and forms.

- **Model Manager**. It takes as input data model specifications and stores them in a *Model Dictionary*.

  The Model Dictionary contains the models defined by the model engineer together with the *Supermodel*, a model which subsumes every other model.

Components of the Architecture (2)

- **Schema Manager**. It takes as input scheme specification over models occurring in the Model Dictionary, checks its correctness and stores it in a *Schema Dictionary*.

- **Translation Generator**. Generates translations between pairs of models, on the basis of a predefined set of basic translations and stores them in the *Library of Translations*.

- **Schema Translator**. It executes translations of schemes between models, according to users requests.

Checks done by MDM

- **Consistency** of a model (based on the ability of verifying the coherence of constructs used and their combination).
- **Translatability** of a model (based on the ability of generating complex translations from basic ones).
- **Correctness of schemes** (based on the ability of matching specification of schemes to specifications of models).
- **Correctness of translations** (as above).

MDM **as an extensible system**

MDM can be extended in a number of ways:

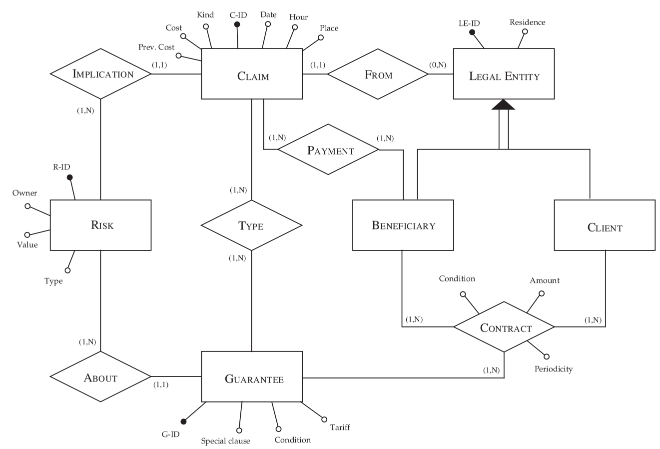
- in the **management of models**:
  - by adding new metaconstructs;
  - by refining the existing ones.
- in the **translation of schemes**:
  - by adding new basic translations,
  - by refining the existing ones.

Example of Application

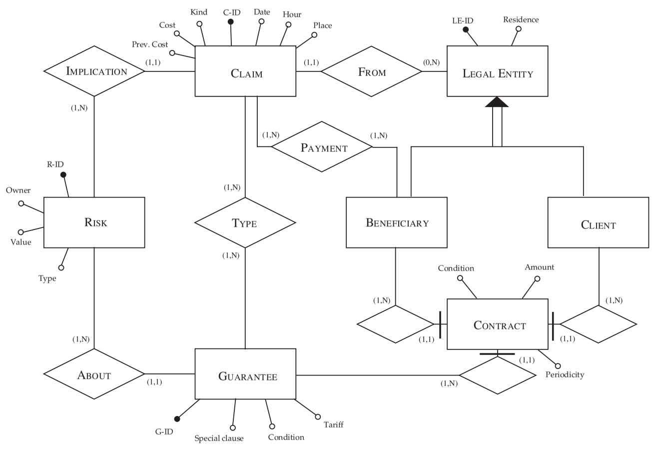
Models involved:

- **ER-Complete Model**: a version of the ER model with n-ary relationships, simple or composite attributes on entities and relationships, and generalizations.
- **Demo Model**: a version of the ER model with binary relationships, simple attributes on entities, and without generalizations.
- **Functional Model**: with monovalued and multivalued functions between entities, attributes on entities, and subset generalizations.
- **Relational Model**.

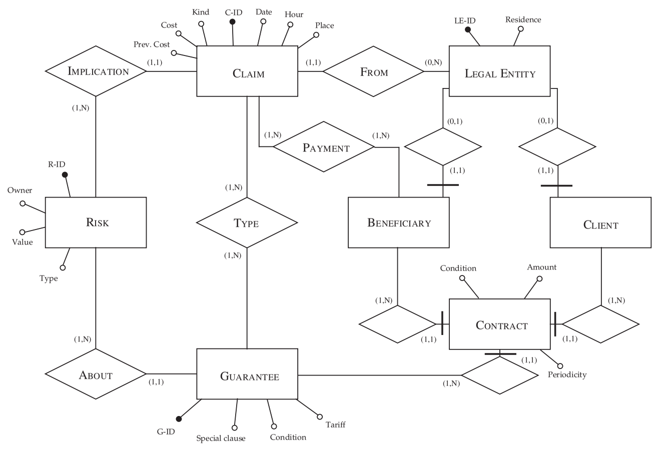
Example Scheme in the ER-Complete Model



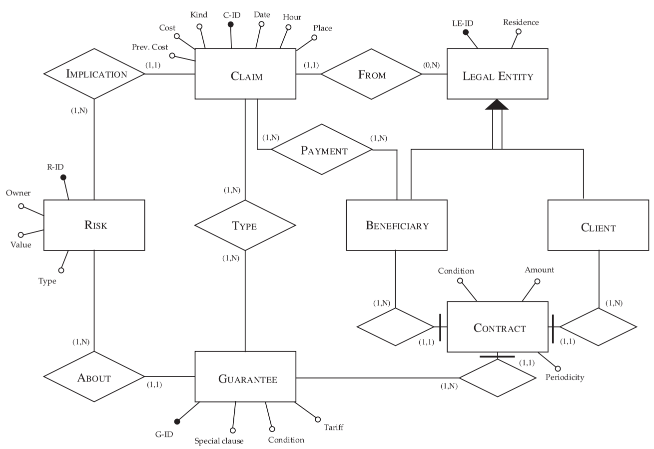
First step of translation into the Demo Model



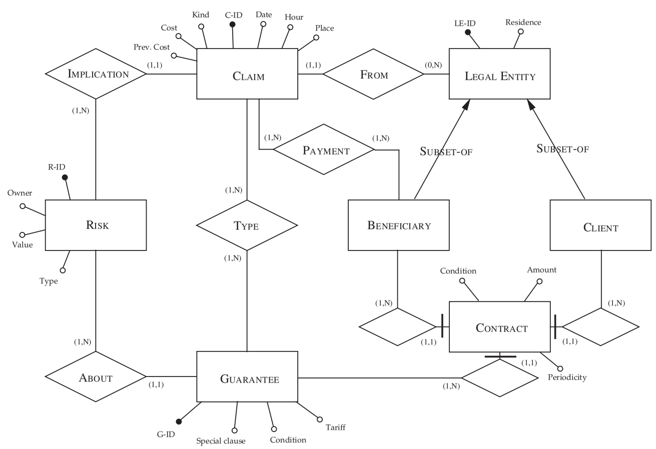
Second step of translation into the Demo Model



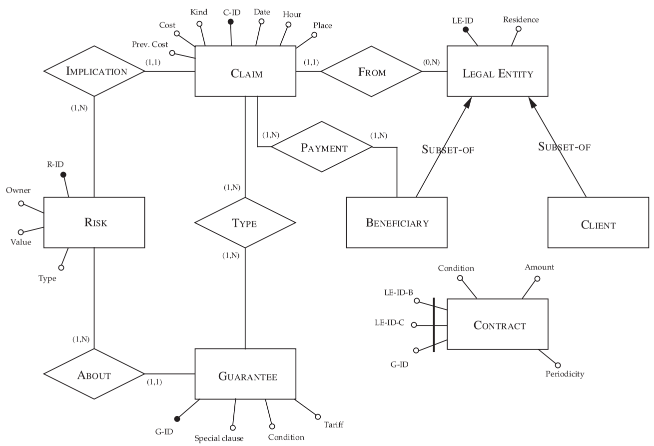
First step of translation into the Functional Model



Second step of translation into the Functional Model



Third step of translation into the Functional Model



Fourth step of translation into the Functional Model