In order to achieve a consistent database, information models first have to be validated and evaluated before they are finally coded in the Data Dictionary. The need to provide the validation of information models increases by the use of dynamically reconfigurable Object Management Systems such as they are a part of a modern CAD Framework.

Validation and evaluation of a information model using STEP means to check EXPRESS language models (ISO 10303) against product models specified by the STEP exchange structure (ISO 10303-21).

A first approach was done by NIST implementing the National PDES Testbed in 1988 using C. They generate an in-memory data dictionary from the EXPRESS language and an in-memory representation of the product models. The product models are interpreted based on the data dictionary. In 1990 the implementation of the NIST STEP Class Library (SCL) using C++ was initiated. In its final version the SCL will provide functions to validate the in-memory product models against the C++ classes that are generated from EXPRESS language models.

In our approach we translate both the EXPRESS language model and the product models to Prolog. We found that a lot of problems occurring during the validation are solved best by the use of a logic programming language. This is, because most of the properties that have to be checked relate to the checking of set inclusions and logical relationships. Using Prolog for implementation purpose we developed an elegant way to map EXPRESS models and the related STEP exchange structure to Prolog. The mapping to Prolog in most parts is nearly a one to one mapping, so that the produced code is even readable by a user who has no knowledge about Prolog. This means in addition, that a cross compiler for each of the two languages to Prolog is most easy to implement.

The model checker, a separate Prolog module, processes the mapped code and gives a detailed error report for each consistency fail. The choice of Prolog as an implementation language facilitates the implementation of a reporting component to explain inconsistencies in a most detailed way. This is simply to monitor the path in the proof tree if the reduction fails.

The simplicity of the translation will be shown by the following two examples translating the EXPRESS language to Prolog and the STEP exchange structure to Prolog.

Mapping the EXPRESS language, first each type definition is translated to a Prolog fact. The mapping of entities is twofold. In a first list each entity is enumerated by naming the identifiers of its attributes, called list_of_ids. In a second list, called list_of_types, the types of the attribute definitions are enumerated (lists are denoted by [ ] in Prolog). Each entity gets an additional parameter id by default to store the identifier of the particular instance.

```prolog
ENTITY person;
  name :STRING;
  birthdate:ARRAY[1:3] OF INTEGER;
  hair :hair_type;
  children :set [0:?] of person;
END_ENTITY;
```

Internal mappings of inherited objects are translated to instances with the same identifier.

Our approach has shown that it is most natural to translate STEP to a logical programming language and that validation using Prolog can even be handled by non-Prolog experts.

\[\text{Fundmed by ECIP2, ESPRIT Project 2072}\]