10

Modelling domain knowledge with $\mathcal{EDDL}_{DP}$

Ernesto Compatangelo
Istituto di Informatica - Università di Ancona
via Brecce Bianche, 30 - 60125 ANCONA (Italy)
e-mail compatan@anvax1.unian.it

Giovanni Rumolo
Dipartimento di Informatica e Sistemistica - 1ª Università di Roma
via Salaria, 113 - 00198 ROMA (Italy)
e-mail rumolo@assi.dis.uniroma1.it

Abstract

This paper describes a new kind of logic-based approach to domain knowledge modelling in information systems engineering. The approach is centred around $\mathcal{EDDL}_{DP}$, a formal but user-friendly language used for an integrated description of structural and behavioural concepts at the domain level. These concepts, called information meta-concepts, represent the external outlook of the essential components of real-world application domains under a minimalist viewpoint. $\mathcal{EDDL}_{DP}$ is an analyst-oriented, epistemological concept language which allows automatic reasoning about a domain knowledge base composed of information meta-concepts. The $\mathcal{EDDL}_{DP}$ language is founded on an extensible framework explicitly conceived for the development of a new generation of computer-based analysis tools endowed with automatic deductive capabilities. The syntax of the language as well as some of the automatic reasoning capabilities bound to $\mathcal{EDDL}_{DP}$ descriptions are shown by means of a real-world example. The main features of the $\mathcal{EDDL}_{DP}$-centred approach are compared with the corresponding features of two popular structural and behavioural modelling approaches.

Keywords

domain, conceptual modelling, meta-concepts, concept languages, automatic reasoning

A. Sutcliffe et al. (eds.), *Domain Knowledge for Interactive System Design*

1 INTRODUCTION

Domain knowledge is the main source of information used in disciplined approaches addressing the elicitation and the specification of the requirements of an information system [Jarke et al., 1993]. Therefore, domain knowledge modelling (DKM) has been always considered, at least implicitly, as a critical preliminary stage supporting system requirements analysis and specification [Borgida et al., 1985, Mylopoulos et al., 1990, Grosz, 1992, Hsia et al., 1993]. In fact requirements, i.e. relevant knowledge about problem issues arising from either the introduction or the modification of an information system, are mainly constraints imposed over a corresponding domain model. In order to be of more effective use, at least the most relevant part of the huge amount of domain knowledge elicited in real-world situations should be captured inside computers. In fact, by performing automatic reasoning over the considered domain, non-trivial hidden properties can be derived and inconsistencies can be detected, thus providing analysts with further valuable information.

Whenever dealing with computer-based knowledge representation and reasoning in information systems engineering (as well as in any other area), analysts are faced by three fundamental questions [Borgida and Jarke, 1992]: (i) what kind of knowledge is captured and thus modelled; (ii) what kind of language(s) are employed for this task; (iii) what reasoning processes run on these languages. The $\mathcal{EDDL}_{DP}$-centered approach to domain knowledge modelling provides an overall answer to the above questions within the framework of concept languages [Mylopoulos et al., 1990, Brachman et al., 1991]. The presentation is organised as follows. Section two outlines the $\mathcal{EDDL}_{DP}$-centred approach to domain knowledge modelling. Section three introduces the epistemological domain description language $\mathcal{EDDL}_{DP}$, showing its main characteristics by means of a real-world example. Section four describes the features of $\mathcal{EDDL}_{DP}$ with respect to other domain knowledge modelling approaches. Finally, section five outlines some major conceptual results as well as future work.

2 THE $\mathcal{EDDL}_{DP}$-CENTRED APPROACH TO DOMAIN KNOWLEDGE MODELLING

Domain knowledge is currently modelled as a collection of concepts, attributes and relationships organising the relevant entities of a given set of real-world organisational phenomena [Barstow, 1993]. Although distinct approaches to the essence of domain knowledge modelling seem to coexist [Kramer, 1993], a domain is generally regarded as a topic for an indicative description in its own, independently of any optative description it may be eventually made of a system to be embedded in that domain [Jackson and Zave, 1993]. Despite of a tight sequential link to requirements-oriented activities, domain knowledge modelling (and analysis) must be considered as an independent phase leading to to the description of real-world situations. Such a phase must be thus based on a specifical set of ontological, methodological and operational assumption giving rise to a framework for domain knowledge modelling and analysis.

The $\mathcal{EDDL}_{DP}$ epistemological domain description language is founded on an innovative, ontologically and methodologically extensible, conceptual framework based on two different description levels, namely:

1. An epistemological, analyst-oriented, external description level characterised by a description scheme defining a set of real-world concepts strongly bound to the specific nature of the considered domain. Each epistemological concept is built using an analyst-oriented syntax which is easily understandable by users as well. Each syntactical declaration of a given concept constructor at the epistemological level is mapped into a corresponding finite set of declarations at the underlying terminological level by way of a particular set of rewrite rules.

2. A terminological, reasoning-oriented, internal description level. Syntax and semantics of concept constructors of the corresponding description scheme define a decidable language belonging to the KL-ONE family [Woods and Schmolze, 1992]. Being endowed with both a formal syntax and a set-theoretic semantics, this latter language naturally supports automatic reasoning within the framework of terminological knowledge representation systems such as CLASSIC [Brachman et al., 1991].

This paper, which focuses on the epistemological level of domain knowledge modelling, is centred around the $\mathcal{EDDL}_{DP}$ language, while the corresponding terminological level as well as its counterpart language are described elsewhere [Compatangelo and Rumolo,]. It is worth noting that the introduction of these two intertwinned levels is an innovative feature of the $\mathcal{EDDL}_{DP}$-centered approach avoiding, among other things, direct user interaction with the reasoning-oriented terminological level. In this way, analysts can concentrate on the very nature of the domain which is modelled using a simple, almost natural language instead of the rigid, formal terminological syntax. The introduction of an epistemological level is not a matter of syntactical beauty culture, but a way of explicitly discriminating between distinct real-world concepts deserving different description, classification and management rules according to their specifical nature. In fact, the terminological symbolic level, roughly corresponding to the underlying mathematical formalism of a physical description, is not endowed with embedded criteria allowing a discrimination among different kinds of structured domain concepts. Therefore, this latter level is more properly used as a representation algebra based on terminological concept constructors, which can be manipulated by automatic reasoners without any reference to the epistemological meaning of the overall information content of each concept.

The $\mathcal{EDDL}_{DP}$ language gives rise to a particular conceptual description (i.e. to a specifical information-oriented image) of the potential environment of a hypothetical system not explicitly taken into account. An $\mathcal{EDDL}_{DP}$-centered domain knowledge model is represented in a user-friendly form in order to facilitate its development as well as its interpretation by different specialists. Being a formal approach to knowledge representation, the $\mathcal{EDDL}_{DP}$-centered approach is neither concerned with nor it does explicitly support most of those elicitation, cooperation, communication and agreement activities which are considered as the essence of requirements engineering [Hofmann, 1993, Pohl, 1994]. Being a formal language endowed with a well-defined semantics, $\mathcal{EDDL}_{DP}$ cannot deal with semantically ill-defined information, thus limiting its scope to those concepts for which

an unambiguous interpretation in terms of set theory can be given. The introduction of each new kind of concept in $\mathcal{EDDL}_{DP}$ must be thus thoroughly analysed in order to verify its compatibility with the overall decidability, computational tractability and semantical meaningfulness of the language.

The $\mathcal{EDDL}_{DP}$-centered, analyst-oriented approach has been explicitly conceived as a theoretical background for the development of a new generation of computer-based analysis support tools endowed with deductive capabilities. At present, this approach is limited to the description of two categories of real-world domain concepts, namely meta-data and meta-processes. These two fundamental categories of concepts, which are considered by the authors as the relevant kernel of domain knowledge, provide sufficient input to the subsequent requirements elicitation, analysis and specification phases.

The description of relevant domain knowledge about a specifical environment is performed in terms of structural and behavioural concepts and links at the highest abstraction level with respect to the considered granularity. Concepts and links introduced at this level give rise to a coherent and finite-complexity image of what composes the considered real-world picture, but not of how each component is internally structured. Such an image, representing environmental knowledge, does not contain information involving structural instance ordering or behavioural procedurality (i.e. control) in both its temporal or selection forms. In fact, the introduction of ordering or procedural information corresponds to the imposition of a set of constraints over a domain model, thus giving rise to information systems requirements specification at a lower abstraction level.

3 THE $\mathcal{EDDL}_{DP}$ LANGUAGE

Following some well-known proposals [Mylopoulos et al., 1990, Loucopolos et al., 1991, Greenspan et al., 1994], the domain knowledge modelling approach described in this paper is based on logic. Like other recent proposals [Kangassalo, 9293], knowledge representation is centred around the definition of concepts belonging to the universe of discourse (UoD). The ontology of $\mathcal{EDDL}_{DP}$ gives meaning to each concept introduced at the epistemological (external) level of the proposed formal description scheme.

At present, $\mathcal{EDDL}_{DP}$ gives the analyst two different real-world epistemological categories of concepts in order to develop a description of the UoD, namely: meta-data concepts and meta-process concepts. These two basic categories of structural and behavioural concepts are considered by the authors as the fundamental components of a domain description. It is worth noting that the term concept as used in this context has the same interpretation as the term class used in object-oriented approaches [Coad and Yourdon, 1991, Rumbaugh and Blama, 1991]. Using such a minimalist ontology of concepts, a reasonable trade-off between the desired characteristics of a hypothetical domain description language and its practical feasibility in term of decidability, semantics consistency and computational complexity is reached.

Each meta-data concept is an abstraction of a collection of autonomous individuals populating the UoD. Meta-data are characterised by a set of structural properties called attributes. An $\mathcal{EDDL}_{DP}$ attribute is a binary relationship between meta-data individuals. Each attribute links the meta-data under definition to another meta-data called domain

of the attribute. Upper and lower number restrictions (cardinalities) on attributes, as well as functional attribute chaining can be also introduced.

Each meta-process is an abstraction of a collection of transformations between UoD meta-data individuals. Meta-processes are characterised by their contexts, each representing the structural description of the corresponding external meta-process behaviour in terms of its input and output meta-data components. Meta-processes are characterised by a set of structural properties (meta-process attributes) called channels. An $\mathcal{EDDL}_{DP}$ channel is a binary relationship between meta-process and meta-data individuals. Each channel, invariably belonging to one of the two input or output groupings, links the meta-process under definition to a meta-data concept called domain of the channel. Upper and lower number restrictions (cardinalities) on channels can be also introduced. As $\mathcal{EDDL}_{DP}$ explicitly avoids supporting procedural details representing the how of a process description (i.e. the internal structure of the corresponding transformation function), a meta-process concept is regarded as a full black box. No information other than its input and output meta-data and channels is contained in a meta-process context declaration. Sequential chaining of two meta-processes can be declared by a flow assertion, representing a conceptual description of process behaviour imposing a constraint between two meta-processes.

3.1 Exploring the $\mathcal{EDDL}_{DP}$ Syntax

In order to depict the essential features of the $\mathcal{EDDL}_{DP}$-based approach to domain knowledge modelling, a presentation by means of an example was adopted in this paper. The full syntax of the language is described in [Compatangelo and Rumolo,]. Following this style of presentation, the following exemplary textual description of a system-oriented patient monitoring domain (PMD) will be considered for discussion.

A set of measured raw physiological data about a patient (blood pressure, temperature, pulse frequency) is periodically read from different devices, formatted and successively registered in a chronologically ordered list. Every time a data set is read, the value of each element of the set is compared with a fixed range of corresponding normal physiological value. If at least one element is outside its normal range, an alarm condition is signalled. A patient monitoring report containing a chronologically ordered subset of the last N data taken from the previously cited list is issued whenever requested by the medical staff.

The corresponding $\mathcal{EDDL}_{DP}$ description of the patient monitoring domain is given in Figures 1, 2 and 3. The following syntactical conventions were adopted: terms in bold denote reserved words of the language (i.e. concept constructors), terms in small capital style with first letter in capital denote concept names (i.e. meta-data or meta-process names), while terms all in small capital style denote attributes. An (optional) N : M declaration denotes the cardinality of an attribute. Meta-data and meta-process concepts are respectively declared in $\mathcal{EDDL}_{DP}$ using the reserved words **data** and **process** for sake of simplicity. However, it is worth noting that the meaning of these meta-concepts encompasses both a more restricted data processing interpretation as well as a more general interpretation considering them as structural or behavioural information about real-world elements. This means that an $\mathcal{EDDL}_{DP}$ domain description can be used at

data CONTROL
included in GENERIC-DATA
data REPORT-REQUEST
included in CONTROL
data ALARM-CONDITION
included in CONTROL
data PHYS-VALUE
included in GENERIC-DATA
data BLOOD-PRES
included in PHYS-VALUE
data PULSE-FREQ
included in PHYS-VALUE
data MEAN-TEMP
included in PHYS-VALUE
data PHYS-DATA **equals**
structure with attributes
PRES **as** 1 : 1
of type BLOOD-PRESS ;
TEMP **as** 1 : 1
of type MEAN-TEMP ;
FREQ **as** 1 : 1
of type PULSE-FREQ ;
data RAW-VALUE
included in GENERIC-DATA
data FMT-VALUE
included in GENERIC-DATA
data RAW-BLOOD-PRES
included in
RAW-VALUE, BLOOD-PRES
data RAW-PULSE-FREQ
included in
RAW-VALUE, PULSE-FREQ
data RAW-MEAN-TEMP
included in
RAW-VALUE, MEAN-TEMP
data RAW-PHYS-DATA **equals**
structure with attributes
PRES **as** 1 : 1
of type RAW-BLOOD-PRES ;
TEMP **as** 1 : 1
of type RAW-MEAN-TEMP ;
FREQ **as** 1 : 1
of type RAW-PULSE-FREQ ;

data FMT-BLOOD-PRES
included in
FMT-VALUE, BLOOD-PRES
data FMT-PULSE-FREQ
included in
FMT-VALUE, PULSE-FREQ
data FMT-MEAN-TEMP
included in
FMT-VALUE, MEAN-TEMP
data FMT-PHYS-DATA **equals**
structure with attributes
PRES **as** 1 : 1
of type FMT-BLOOD-PRES ;
TEMP **as** 1 : 1
of type FMT-MEAN-TEMP ;
FREQ **as** 1 : 1
of type FMT-PULSE-FREQ ;
data MOST-RECENT-PHYS-DATA
included in FMT-PHYS-DATA
data CHRONO-DATA-STORE **equals**
structure with attributes
ELEMENT **of type**
FMT-PHYS-DATA ;
data RECENT-PHYS-DATA
included in FMT-PHYS-DATA
data PATIENT-REPORT
included in
CHRONO-DATA-STORE **and**
structure with attributes
ELEMENT **of type**
RECENT-PHYS-DATA ;
data NORMAL-PHYS-RANGES **equals**
structure with attributes
NORMAL-BLOOD-PRES
of type FMT-BLOOD-PRES ;
NORMAL-MEAN-TEMP
of type FMT-MEAN-TEMP ;
NORMAL-PULSE-FREQ
of typeFMT-PULSE-FREQ ;
disjoint
REPORT-REQUEST,
ALARM-CONDITION

Figure 1: data descriptions in the PMD example

different abstraction levels, i.e. an epistemological knowledge base can contain details about the essential information concepts which describe the world surrounding a data-processing system, it can contain information about data structures and processes or both.

```
process  TRANSFORMATION
         has input
         RAW-DATA as 1 : 1
         of type RAW-PHYS-DATA ;
         has output
         FMT-DATA as 1 : 1
         of type FMT-PHYS-DATA ;
process  COMPARISON
         has input
         CURRENT-VALUES as 1 : 1
         of type FMT-PHYS-DATA ;
         REFERENCE-VALUES as 1 : 1
         of type NORMAL-PHYS-RANGES ;
         has output
         STATUS as 1 : 1
         of type ALARM-CONDITION ;

process  REPORT-GENERATION
         has input
         REQUEST as 1 : 1
         of type REPORT-REQUEST ;
         REPORT-DATA as 1 : N
         of type RECENT-PHYS-DATA ;
         has output
         REPORT as 1 : 1
         of type PATIENT-REPORT ;
process  UPDATE
         has input
         MOST-RECENT-DATA as 1 : 1
         of type FMT-PHYS-DATA ;
         has output
         UPDATED-DATA-STORE as 1 : 1
         of type CHRONO-DATA-STORE ;
```

Figure 2: process descriptions in the PMD example

```
flow  MONITORING
      from TRANSFORMATION
      using FMT-DATA
      to COMPARISON
      using CURRENT-VALUES

flow  REGISTRATION
      from TRANSFORMATION
      using FMT-DATA
      to UPDATE
      using MOST-RECENT-DATA
```

Figure 3: Flow descriptions in the PMD example

3.2 The $\mathcal{EDDL}_{DP}$ semantics: an outline

$\mathcal{EDDL}_{DP}$ set-theoretic semantics is fully expressed in terms of its mapping into the underlying terminological language $\mathcal{TDDL}_{DP}$, as described in [Compatangelo and Rumolo,]. Each epistemological description is translated into a finite set of terminological expressions using a specifical set of rewrite rules. In this paper, the $\mathcal{EDDL}_{DP}$ semantics will be outlined by means of the above PMD example.

For instance, let us consider the $\mathcal{EDDL}_{DP}$ description of PHYS-DATA shown in figure 1. This meta-data concept has the meaning of that set of UoD elements always having the attribute pressure of type BLOOD-PRES, the attribute frequency of type PULSE-FREQ and the attribute temperature of type MEAN-TEMP. Under an epistemological viewpoint,

an attribute (i.e. PRES) is a general property linking an object (i.e. PHYS-DATA) to the value of the corresponding attribute domain (i.e. BLOOD-PRESS).

Following a well-known semantical distinction originally introduced in the KL-ONE family of concept languages [Woods and Schmolze, 1992], each new meta-data concept can be defined as being either **exactly corresponding to** or **included within** its description structure composed of already-existing meta-data concepts. Meta-data concepts of the first kind are declared using the **equals** clause, while meta-data concepts of the second kind are declared using the **included in** clause and may belong to an IS-A generalisation. In order to specify that the meta-data PATIENT-REPORT is a subset of CHRONO-DATA-STORE, it is sufficient to declare

data PATIENT-REPORT **included in** CHRONO-DATA-STORE

In this way, the structure of the latter meta-data is automatically inherited by the former one. The IS-A relationship has the meaning of set containment: by means of inheritance, the PATIENT-REPORT meta-data type has the same attributes as the CHRONO-DATA-STORE meta-data type. Moreover, for each meta-data definition, the *most specific meta-data concept* (i.e. the data concept immediately above the given one in the IS-A hierarchy) can be deduced by a classification algorithm [Borgida and Patel-Schneider, 1994]. Each meta-data is defined using a meta-data expression. $\mathcal{EDDL}_{DP}$ has a suitable set of meta-data constructors, including multiple inheritance, attribute number restrictions and attribute inheritance. Each one contributes to the definition of intensional knowledge assertions about meta-data concepts belonging to the considered UoD.

A meta-process is considered as a transformation from some set of input meta-data to some other set of output meta-data. Every input/output meta-data inflows/outflows the meta-process through a specific role named channel. The context of a meta-process represents the captured structural description of process behaviour. A context is defined through the description of all of its channels and meta-data partitioned into the two input and output groupings. Each meta-process is an individual concept interpreted as a singleton set differing from a meta-data class. Under an epistemological viewpoint, a meta-process is a single transformation between input and output meta-data sets, while its context is the collection of all transformations between the same meta-data sets. Each channel represents a link between the meta-process and a meta-data concept. Channels are suitable abstractions for a uniform description of control data, input/output procedure parameters and object methods.

While each single meta-process concept is declared by means of its context, sequential chaining of two meta-processes can be declared by a flow assertion. The latter is a conceptual description of process behaviour representing two constraints between meta-processes. First, all data linked to the first meta-process by means of its output channel flow into the second meta-process by means of its corresponding input channel. Second, the specific meta-data linked to the first meta-process by means of its output channel must belong to the same type as the specific meta-data linked to the second meta-process by means of its corresponding input channel. For instance, the MONITORING flow declaration shown in figure 3has the following interpretation. All meta-data linked to the TRANSFORMATION meta-process by means of the FMT-DATA channel flow into the COMPARISON meta-process by means of the CURRENT-VALUES channel. The domain of the

linked meta-data concept must be of the same type as the one of CURRENT-VALUES, while the second constraint is enforced by using the same FMT-PHYS-DATA name.

It is worth noting that two potentially distinct classifications for meta-process and meta-process contexts, differing because of the distinct classification rules of meta-processes with respect to their corresponding contexts, may coexist [Compatangelo and Rumolo,]. For instance, let us consider the meta-concept descriptions shown in Figure 4. Following

process	DIV **has input** DIVIDEND **as** 1 : 1 **of type** INTEGER; DIVISOR **as** 1 : 1 **of type** INTEGER; **has output** RESULT **as** 1 : 1 **of type** INTEGER;	**process**	INTEGER-PART-OF-DIVISION **has input** DIVIDEND **as** 1 : 1 **of type** REAL; DIVISOR **as** 1 : 1 **of type** REAL; **has output** RESULT **as** 1 : 1 **of type** INTEGER;
data	REAL **included in** GENERIC-DATA	**data**	INTEGER **included in** REAL

Figure 4: A fragment of a mathematical domain

the terminological rule of concept subsumption[Borgida and Patel-Schneider, 1994], the context of the DIV meta-process will be classified under the context of the INTEGER-PART-OF-DIVISION meta-process. Conversely, following the general rule of function subtyping [Cardelli, 1984], separately applied to both input and output context components, the INTEGER-PART-OF-DIVISION meta-process will be classified under the DIV meta-process. Analysts can use a process typing hierarchy in order to define a new process from an existing one. In fact, $\mathcal{EDDL}_{DP}$ has a **shares context with** clause allowing the declaration of a new meta-process by reusing the context of a previously defined meta-process. In the general case, a new meta-process concept will be classified with respect to the conjunction of its shared context with new additional assertions extending or modifying it. For instance, a meta-process declared as

process NATURAL-DIV **shares context with** DIV **and**
has output NATURAL-RESULT **as** 1 : 1 **of type** NATURAL **redefines** RESULT;

will not be classified under DIV (according to the previously cited rule of function subtyping) until when the following meta-data concept description is added:

data NATURAL **included in** INTEGER

This way of dealing with inheritance allows analysts to focus on the modelling activity only, leaving scheme checking to computers.

4 $\mathcal{EDDL}_{DP}$ VERSUS OTHER MODELLING APPROACHES

The entity-relationship model [Chen, 1976] is a good starting point in representing real-world application domains, although structural concepts only can be explicitly considered

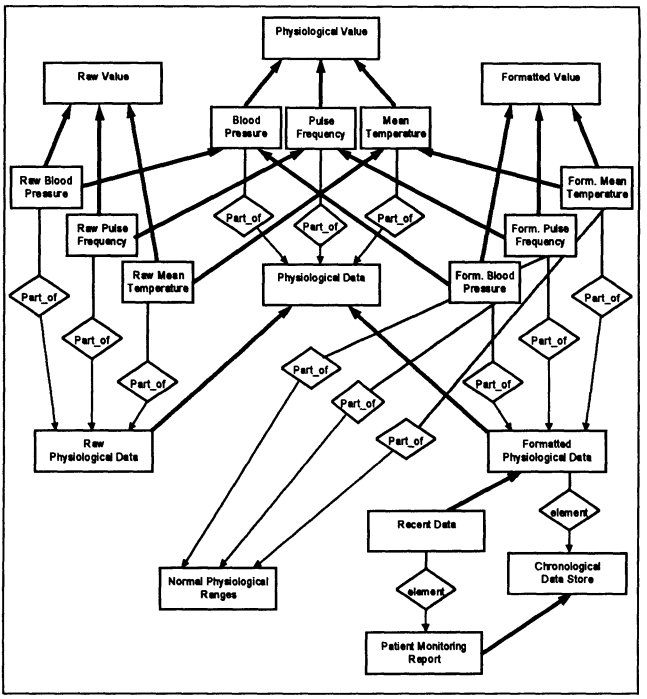

Figure 5: An E-R diagram detailing the structural part of the PMD example

and modeled. An entity represents a set of homogeneous individuals, while a relationship represents a relation (at least a binary one) between entities. Figure 5 shows a complete description of the structural part of the PMS domain captured by the E-R approach. Structured data flow modelling [Yourdon, 1989] is regarded as a conceptual tool even at the problem-oriented abstraction level. The patient monitoring domain according to the SDF modelling approach is shown in Figure 6. It is worth noting that although SDF modelling was born as a functional approach to software design, nevertheless it can be used at the domain level too. However, a SDF domain model may retain a certain system-oriented flavour.

We claim that $\mathcal{EDDL}_{DP}$ subsumes most modelling concepts proposed in semiformal schemes. The main correspondences between $\mathcal{EDDL}_{DP}$, E-R and SDF are briefly reported in order to show how $\mathcal{EDDL}_{DP}$ can express and model most concepts originally introduced in semiformal approaches. A formal analysis of $\mathcal{EDDL}_{DP}$ versus the latter two approaches shows that both E-R and SDF do not even consider some fundamental description language properties such as soundness or completeness. Therefore, the following comparison will be limited to their apparent expressive power.

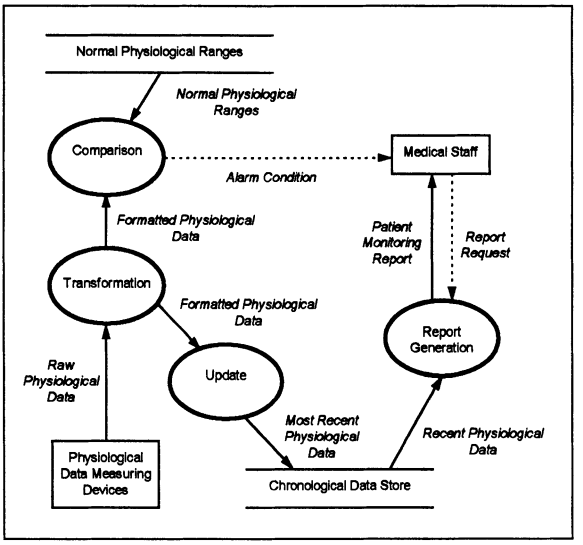

Figure 6: The SDF diagram for the PMD example

The E-R model has the following classes of descriptive concepts: (i) entities, corresponding to $\mathcal{EDDL}_{DP}$ meta-data concepts; (ii) attributes, which are close to $\mathcal{EDDL}_{DP}$ attributes and (iii) relationships, which can be expressed in $\mathcal{EDDL}_{DP}$ by either attributes or meta-data depending on the particular E-R interpretation of the relationship itself. Moreover, the $\mathcal{EDDL}_{DP}$ model has multiple IS-A hierarchies and numerical restrictions. The SDF model has the following classes of concepts: (i) terminators, which can be skipped from the $\mathcal{EDDL}_{DP}$ scheme as useless, being already integrated in the process concept definition; (ii) data stores, which can be easily defined as $\mathcal{EDDL}_{DP}$ meta-data concepts; (iii) processes, which are close (but not equal to) $\mathcal{EDDL}_{DP}$ meta-processes and (iv) flows, represented in $\mathcal{EDDL}_{DP}$ by flow declarations.

An $\mathcal{EDDL}_{DP}$ description is more expressive than the corresponding E-R one, because of explicit treatment of processes. At the same time, it overcomes several limitations of a SDF description, such as semantical ambiguity, lack of cardinality of data flows, incomplete definition of processes. Moreover, a $\mathcal{EDDL}_{DP}$ description is much shorter and more compact than the corresponding SDF one. $\mathcal{EDDL}_{DP}$ explicitly distinguishes between data (existing on their own) and input/output channels connecting them to processes. In $\mathcal{EDDL}_{DP}$, meta-processes cannot be described without reference to (at least one) input and one output meta-data concept. Therefore, data stores as considered in SDF (i.e. *Chronological Data Store*, *Normal Physiological Ranges*) are no more needed as special entities distinguished from other data concepts bound to process descriptions. The uniform meta-data description in $\mathcal{EDDL}_{DP}$ disambiguates the role of data contained in SDF

stores with respect to those contained in SDF flows. Moreover, eliminating terminators and considering no more data stores as special entities makes the context diagram in a SDF model quite useless, thus lowering the number of refinement levels.

Hierarchical subsumption relationships among data are ambiguously treated in the SDF modelling approach. Particularly, the data dictionary section is intrinsically unable to show existing structural subsumption relations between data (i.e. *Recent Physiological Data* as well as *Most Recent Physiological Data*, *Chronological Data Store* and *Patient Monitoring Report* in the PMD example, see Figure 6). Moreover, the case of data stores connected to flows carrying the same data class with different input and output names (i.e. *Most Recent Physiological Data* and *Recent Physiological Data* with respect to *Chronological Data Store*) is rather ambiguous. These shortcomings are not present in the corresponding $\mathcal{EDDL}_{DP}$ meta-data descriptions, where *Most Recent Physiological Data* is no more needed. $\mathcal{EDDL}_{DP}$ allows automated inference of structural hierarchies. For example, the IS-A relationship linking both *Physiological Data* and *Raw Physiological Data* to *Physiological Data* is inferred by a subsumption algorithm. Therefore, explicit IS-A assertions among meta-data concepts are used by analysts as domain constraints whose verification is left to the automatic reasoning system. In fact, the subsumption relationship linking both *Formatted Physiological Data* and *Raw Physiological Data* to *Physiological Data* is derived from the semantics of the concept defining *Physiological Data* instead of being an arbitrary assertion about concepts.

The flow concept in $\mathcal{EDDL}_{DP}$ is quite different from the corresponding (overloaded) one in SDF. In fact, a SDF flow declaration represents both the data flowing in it (an entity) and the name of the link connecting two entities (a relationship). In this way, a fundamental ambiguity is introduced by mixing objects corresponding to a one-place logical predicate (entities) with objects correspondig to a two-places predicate (flows). Moreover, two flows with the same name (thus containing the same data, i.e. *Formatted Physiological Data*) can link different couples of processes (*Transformation - Comparison*, *Transformation - Update*) introducing another source of ambiguity in reasoning. No default link between meta-processes is implicit in $\mathcal{EDDL}_{DP}$; links are explicitly declared through flow declarations whenever needed. Conversely, each meta-process in $\mathcal{EDDL}_{DP}$ is linked (by channels) to its input and output meta-data respectively. In this way, the SDF data store concept becomes redundant, and the flow concept becomes just a link chaining the output channel of the first meta-process to the input channel of the second one. This excludes any source of ambiguity between flows directly connecting two meta-processes and meta-data flowing into them. Moreover, while SDF modelling explicitly distinguishes control from data by introducing two different kinds of processes and flows [Yourdon, 1989], $\mathcal{EDDL}_{DP}$ modelling considers controls as meta-data with a particular (different and somehow elementary) information content, thus allowing meta-data to play a control role if needed.

It is worth noting that $\mathcal{EDDL}_{DP}$ is not just the merged version of the E-R and SDF approaches for several fundamental reasons. First, SDF is not semantically defined at all (i.e. it has no unambiguous interpretation, as shown above), so a hypothetical merged language would neither be endowed with an unambiguous semantics not it could be supported by any existing computer-based knowledge Representation & management system. Second, $\mathcal{EDDL}_{DP}$ meta-processes and flow links, although similar, are not SDF processes

and flows, as shown above. Third, $\mathcal{EDDL}_{DP}$ is a real multiparadigm language in the sense that it supports an integrated description of both structural and behavioural concepts using a uniform, simple language instead of a set of different, specialised languages modelling only a single aspect of domain knowledge.

5 CONCLUSIONS

$\mathcal{EDDL}_{DP}$ is based on a formal scheme allowing an integrated description of both structural and behavioural domain concepts. It provides the epistemological, analyst-oriented, external description level of an overall framework accommodating domain knowledge at different abstraction levels. Such a framework includes an underlying terminological, reasoning-oriented internal description level centred around a formal logic language. The introduction of the two intertwined description and abstraction levels, as well as the explicit differentiation between ontological, methodological and operational issues throughout the framework are among the novel features of the proposed approach.

$\mathcal{EDDL}_{DP}$ offers a rich set of meta-data and meta-process description operators for the development of the essential domain model of an information system environment. At first sight, modelling concepts explicitly offered by $\mathcal{EDDL}_{DP}$ might seem very limited (and limiting) if compared with other modelling approaches as well as with recent research issues in related areas [Grosz, 1992, Maiden and Tyndale, 1994]. However, two major points must be remarked. First, an approach in the KL-ONE style is explicitly intended to support automatic reasoning over domain schemes, emphasising at the same time the role of deductive services offered by terminological knowledge representation systems. This choice implies the existence of a well-defined semantics for $\mathcal{EDDL}_{DP}$, automatically limiting the number of possible domain concepts as well as their expressive power. This limitation is necessary in order to keep the language meaningful, decidable and computationally tractable with respect to a complete freedom of choice. In fact, no kind of automatic reasoning other than syntactical correctness checking is possible about a domain model based on either an ill-defined semantics or no formal semantics at all. Second, $\mathcal{EDDL}_{DP}$ was developed in a bottom-up way, starting with meta-data and meta-processes as fundamental domain modelling concepts common to the whole information systems area. These concepts are considered by the authors as the core of any real-world domain model. However, additional concepts other than meta-data and meta-process ones will be uniformly introduced in future, extended $\mathcal{EDDL}_{DP}$ versions if needed. Each new candidate concept will be endowed with a particular set-theoretic semantics explicitly depending on its specific role at the epistemological level.

The ultimate goal of the $\mathcal{EDDL}_{DP}$-centered approach is the development of a new generation of computer-based analysis support tools endowed with automatic reasoning capabilities. This approach regards such a tool as a powerful assistant in domain knowledge modelling, with the role of helping the analyst during the continuous revision process ending with the completion of the domain specification. Under this viewpoint, $\mathcal{EDDL}_{DP}$ represents a basic language with enough expressive power, but without loss of decidability and completeness. Future work will have to deal with the effectiveness of the proposed approach in dealing with real-world domains of very large dimensions.

REFERENCES

[Barstow, 1993] Barstow, D. (1993). Should we specify Systems or Domains ? In *Proc. of the 1st IEEE Int. Sym. on Requirements Engineering (RE-93)*, page 79.

[Borgida et al., 1985] Borgida, A., Greenspan, S., and Mylopoulos, J. (1985). Knowledge Representation as the basis for Requirements Specification. *IEEE Computer*, pages 82–91.

[Borgida and Jarke, 1992] Borgida, A. and Jarke, M. (1992). Special Issue on Knowledge Representation and Reasoning in Software Engineering. *IEEE Transactions on Software Engineering*, 18(6).

[Borgida and Patel-Schneider, 1994] Borgida, A. and Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308.

[Brachman et al., 1991] Brachman, R. J. et al. (1991). Living with CLASSIC: when and how to use a KL-ONE-like language. In Sowa, J. F., editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann.

[Cardelli, 1984] Cardelli, L. (1984). A Semantics of Multiple Inheritance. In Kahn, G. et al., editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 52–67.

[Chen, 1976] Chen, P. P. (1976). The Entity-Relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

[Coad and Yourdon, 1991] Coad, P. and Yourdon, E. (1991). *Object Oriented Analysis.* Prentice-Hall, 2nd edition.

[Compatangelo and Rumolo,] Compatangelo, E. and Rumolo, G. Formal Domain Knowledge Modelling with Epistemological Concept Languages. Submitted for publication.

[Greenspan et al., 1994] Greenspan, S., Mylopoulos, J., and Borgida, A. (1994). On Formal requirements Modelling Languages: RML Revisited. In *Proc. of the 16th Int. Conf. on Software Engineering (ICSE-16).*

[Grosz, 1992] Grosz, G. (1992). Building Information Systems Using Generic Structures. In *International Computer Software and Applications Conference (ICSAC-92), Chicago (USA).*

[Hofmann, 1993] Hofmann, H. F. (1993). Requirements Engineering: A Survey of Methods and Tools. Technical Report 93.05, Institut für Informatik der Universität Zürich (IFI).

[Hsia et al., 1993] Hsia, P., Davis, A., and Kung, D. (1993). Status Report: Requirements Engineering. *IEEE Software*, pages 75–79.

[Jackson and Zave, 1993] Jackson, M. A. and Zave, P. (1993). Domain Descriptions. In *Proc. of the 1st IEEE Int. Sym. on Requirements Engineering (RE-93)*, pages 56–64. IEEE Computer Society Press.

[Jarke et al., 1993] Jarke, M. et al. (1993). Requirements Engineering: An Integrated View of Representation, Process and Domain. In *4th Eurpoean Conference on Software Engineering.*

[Kangassalo, 9293] Kangassalo, H. (1992/93). COMIC: A system and methodology for conceptual modelling and information construction. *Data and Knowledge Engineering*, 9:287–319.

[Kramer, 1993] Kramer, J. (1993). "Generalisations are False" ? In *Proc. of the 1st IEEE Int. Sym. on Requirements Engineering (RE-93)*, page 79. IEEE Computer Society Press.

[Loucopolos et al., 1991] Loucopolos, P. et al. (1991). Design and Execution of Event/Action Database Applications. In *2nd Int. Conf. on Deductive Approaches to Information Systems and Databases.*

[Maiden and Tyndale, 1994] Maiden, N. and Tyndale, D. (1994). Reuse of Domain Abstractions during Requirements Engineering. Technical Report 94-5, ESPRIT Basic Research Project 6353, Novel Approaches to Theories Underlying Requirements Engineering (NATURE).

[Mylopoulos et al., 1990] Mylopoulos, J. et al. (1990). Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362.

[Pohl, 1994] Pohl, K. (1994). The Three Dimensions of Requirements Engineering: A Framework and its Applications. *Information Systems*, 19(3):243–258.

[Rumbaugh and Blama, 1991] Rumbaugh, J. and Blama, M. (1991). *Object-Oriented Modelling and Design.* Prentice-Hall.

[Woods and Schmolze, 1992] Woods, W. A. and Schmolze, J. G. (1992). The KL-ONE family. *Computers and Mathematics with Applications*, 23(2-9):1–50.

[Yourdon, 1989] Yourdon, E. (1989). *Modern Structured Analysis.* Prentice-Hall.

BIOGRAPHY

Ernesto Compatangelo received his "Laurea" degree in Physics from University of Rome "La Sapienza" in 1985. He was at Scuola Normale Superiore in Pisa from 1986 to 1989. Later, he worked as system analyst and information systems consultant. He received Dottorato (PhD) in Artificial Intelligent Systems from University of Ancona in 1986. His main interests include information and computer-based systems engineering, conceptual modelling, knowledge representation.

Giovanni Rumolo received his "Laurea" degree in Electronical Engineering from University of Rome "La Sapienza" in 1993. He works as a consultant for the State Authority for Governmental Informatics (AIPA). He is at present PhD student in Medical Informatics at University of Rome. His main interests include medical and governmental information systems, formal requirements languages, knowledge representation.