Rule Based Specification of Information Systems

Peter McBrien* and Anne Helga Seltveit†
The Norwegian Institute for Technology,
Trondheim,
Norway

Benkt Wangler‡
The Swedish Institute for
Systems Development,
Kista,
Sweden

Abstract

This paper describes the TEMPORA approach to capturing the functions and policies of a business as **business rules**, and the way in which those rules may be used to build a working information system. It discusses the nature of such rules, and presents some techniques for rule elicitation.

Keywords: Conceptual Modelling, Semantic Modelling, Rule-based systems, Business Rules, CASE-Tool environments

1 Introduction

An important problem in the areas of business analysis and systems development is question of how to capture the behaviour of the business as a model, and then use that model to build the information system which supports that business. A promising approach to the solution of this problem is to specify the behaviour of the business as rules, and to use those rules in an information system that supports the business. Such rules we will term **business rules**. The aim of this paper is to discuss the nature of these rules, to present how they may be represented, and to describe techniques for their elicitation.

The work reported in this paper was conducted within the ESPRIT project TEMPORA [LMS+91]. The TEMPORA project developed an information systems development environment covering the whole systems development cycle. The TEMPORA environment includes languages, methods and tools for specifying requirements and for designing an information system based on those requirements. A runtime system supports the execution of the specifications on a SYBASE/UNIX platform. The TEMPORA paradigm is founded on two important ideas:

- The explicit *specification of business rules* as part of a conceptual model which serves as the basis for information systems design.
- The inclusion of *support for the temporal dimension* in the specification.

Current approaches to the development of information systems tend to embed domain knowledge in the code which implements the system, which causes problems in later maintenance of the system. These problems would be avoided if we express explicitly, and then maintain throughout the system's life-cycle,

*email: petermcb@idt.unit.no

†email: ahs@idt.unit.no

‡email: benkt@sisu.se

the business rules of an organisation in the information system's specifications. The handling of time provides greater support for the temporal applications frequently found in business problems. A **valid-time** database model is provided [Jen94], where each fact has an associated set of **timepoints** at which it is valid [MSW92], and this temporal dimension may be accessed in rules.

However, the apparent advantage of being able to represent rules explicitly is not without costs. Experience using 'pure' rule-based approaches [Xia91] has revealed a number of problems:

- Maintenance is difficult due to lack of adequate control mechanisms.
- Testing is difficult and costly due to the difference between control flow and data flow not having been modelled in rules.
- Testability and maintainability are reduced since simplification of representation of a flat rule-base is obtained at the cost of sacrificing modularity and abstraction.

Reliability seems to be poor judging from the difficulties found in testing rule-based systems. Whether these are inherent problems of the rule-based paradigm is still a question of much debate. We will not contribute to a philosophical discussion about this, but rather show how the problems have been explicitly addressed by the TEMPORA approach. Specific advantages of the approach include the following key points:

- **Universality of approach**: Rules can be recorded at *all* stages of development, and links maintained between rules found at different stages. This allows the knowledge built during system development to be used in system maintenance, since changes in high-level policy are explicitly linked with their implementation in the information system.
- **Flexibility of presentation**: We support several views of the specification captured; namely rule-based, static (entity-relationship) and dynamic (process model) views. In the process and rule models we also distinguish between control and data, and links are provided between the different views.
- **Consistency of presentation**: The same modelling languages are used at different stages of development, removing the confusion introduced by having different languages model a particular fact at different stages.
- **Evolutionary**: The TEMPORA models are straight forward developments of proven modelling techniques, enhanced to support rules and the temporal dimension. This reduces the costs and disruption involved in migrating to the TEMPORA approach for systems development.
- **Methodological approach**: The is support provided for all stages of development, in the form of a methodology to be followed.

The importance of **enterprise modelling** or **business modelling** is recognised by business practitioners as well as researchers, and a number of approaches to enterprise modelling are proposed, each providing languages with associated methods and tools. Enterprise modelling is closely related to conceptual modelling, and most approaches to enterprise modelling are based on conceptual modelling results. Thus, in this context we will not make a distinction between them but rather focus on what approaches that exist, and what their basic features are. Two major classes of approaches exist: **information systems engineering** (**ISE**) and **knowledge based systems** (**KBS**). Within ISE, we may identify three broad groupings:

- Those with a **single-purpose orientation** or use a combination of such single-purpose orientations. The most common information engineering (modelling) approaches are data-oriented [Che76], object-oriented [Jac83] , process-oriented [deM78], rule-oriented [MNP$^+$91], or combinations of these [Mac86].

- **Requirements engineering** which combines concepts from both the ISE and KBS approaches [DHL+86].
- **Office information systems** approaches, where organisational concepts such as units, roles, documents, resources are emphasised [PV88, CAS88].

Tempora belongs to the ISE class of approaches, using a combination of single-purpose orientations: data modelling the ERT [MSW92], process modelling the PID [KMOS91] and rule modelling in the ERL [MNP+91]. In the KBS area, some of the work on **knowledge acquisition** can clearly be exploited for enterprise modelling [AWS92]. This has become more relevant recently, when we can observe a shift in focus, from the initial transfer view (e.g., basic knowledge acquisition techniques [Har86]) to the modelling view (e.g., more comprehensive knowledge acquisition approaches [BAJ92, Ste89]). In the transfer view, knowledge acquisition was considered as the problem of eliciting knowledge from an expert and represent this knowledge using an appropriate knowledge representation language. This view has now been modified and the knowledge acquisition problem is now considered a much more complicated task. The knowledge may be compiled or hidden, and the expert may become conscious about knowledge during the knowledge acquisition process. A model of the expert's knowledge is gradually constructed (i.e., the modelling view). This view corresponds very much to the view taken in enterprise modelling/conceptual modelling approaches in general, and Tempora in particular. Tempora differs however, in:

- providing a tailored business language that includes the temporal dimension.
- the capture support (i.e., knowledge acquisition) is provided by exploiting both the combination of diagrammatic and non- diagrammatic languages, and the links that relate information within and between models.

In the following we will first describe how we look upon business rules in general (Section 2), give a brief account of how they are represented in the Tempora conceptual schema (Section 3), and finally we outline the process for rule elicitation (Section 4).

2 Business Rules

Fundamental to Tempora is the notion of business, which we define as the continuously on-going process performed within an organisation in order to fulfill its objectives. It may be characterised by the fact that it deals with a specific set of phenomena and that this 'dealing with' is controlled by certain business rules, defining or constraining what actions to take in the various situations that may arise. Hence, we say that business rules are the rules that control the business. More precisely, business rules may be any of the following:

- **high level goals**: loosely defined business objectives and policy statements, which impact on the development of the information system

  e.g. *the business should maximise its profit*
- **low level goals**: more precise statements that describe the way that the business has chosen to achieve its goals and to implement its policies

  e.g. *invoices will be charged interest if not paid within 30 days*
- **external goals**: various rules imposed on the business from the outside world, in the form of laws and regulations

  e.g. *customers must be notified in advance of interest being charged*

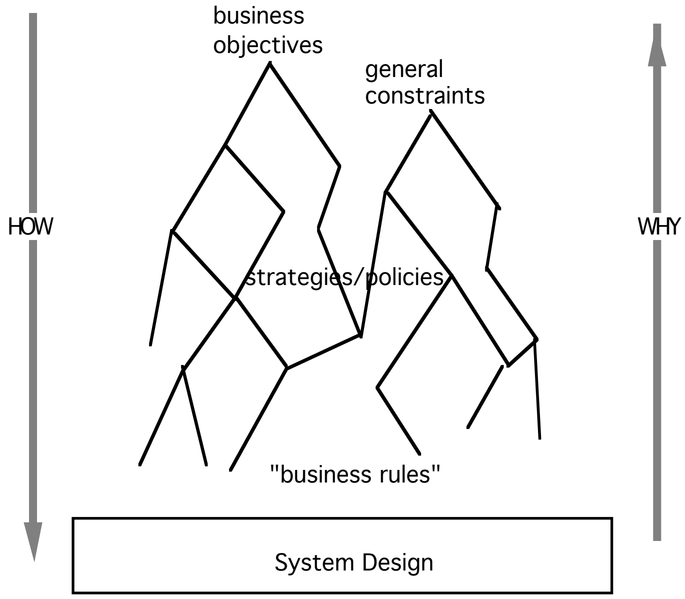


Figure 1: The business rule hierarchy

Identifying the high and low level goals, and the strategies to realise them, may be done by structural refinement, resulting in a hierarchy of goals and means for reaching them. In such an analysis, there is a gradual shift in perspective, in that what at one level is considered as part of a strategy to reach a higher level objective, is later considered a goal in itself, needing even more precise measures. Hence, a subgoal is seen as a means for bringing about a superior goal. The further we get in the analysis, and the lower level we reach, the easier it becomes to formulate rules.

The external goals can be treated in the same manner, since they are merely goals which we are forced to accept as part of our business. Treating them in this manner has the advantage that they will be connected to the internal goals of the business during the structural refinement process. Figure 1 illustrates the relationship between the various levels of abstraction. At the bottom level the business rules may be implemented and embedded in a computerised information system.

This is analogous to the hierarchical decomposition performed in traditional process modelling methods, such as structured analysis [deM78]. Where we differ from this approach is in (1) the use of rules and (2) the distinction drawn between control and data in the PID model. Whilst rules may be attached to processes to describe the behaviour, they may also exist as separate objects in the conceptual model. The rules may be related to each other, or to the other two Tempora languages. The use of rules thus acts as an enhancement of the process and data modelling languages.

Tempora views a business function as characterised by the facts that (1) it handles certain 'things' and that in doing this, (2) it follows certain rules. Hence, we use, in parallel to functional decomposition, 'thing' or **object modelling** by means of the ERT model, and rule specification by means of the ERL.

From a conceptual and logical point of view, every business rule can be regarded as a constraint on the business, restricting the possible ways to fulfilling its goals. We can thus adopt a uniform format for rules, no matter what purpose they serve. Only during the later stage of information system modelling we need specify the way that the rule is implemented in the runtime system. That business rules are explicitly formulated and that we keep a record of how low level rules relate to the higher levels it is important for later maintenance of the information system. We also need to keep track of where business rules are embedded in the programs of a system, so that we may analyse the impact that changes to the business objectives and rules may have on a system. The following sections describe how TEMPORA satisfies these goals.

3 The TEMPORA Conceptual Schema

In TEMPORA, the development of an information system should be viewed as a task of developing and augmenting the policy knowledge base of an organisation. This knowledge base is used throughout the software development process, from requirements specification through to a running system. Within TEMPORA, the corporate knowledge base is concerned with the definition of principal facts and operations within the organisation, together with the rules which guide these operations and ensure the integrity of these facts.

The objects in a TEMPORA conceptual schema can be divided into two classes: the static objects, which refer to the representation of structural relationships that exist between entities and/or values and the constraints upon them; and dynamic objects, which refer to the representation of how the static objects may evolve over time. The static description is contained in the ERT (Entity-Relationship-Time) augmented by ERL (External Rule Language) rules. The dynamic description in contained in the PID (Process Interaction Diagram) augmented by ERL rules. The ERL serves to link the static and dynamic aspects of the system, as indicated in Figure 2. The three modelling languages may be seen as three different windows or projections, through which the conceptual schema can be viewed. We briefly summerise the key features of the three modelling languages in the following sections.

3.1 The ERT Model

The ERT differs from the standard entity-relationship model in several ways, but for the purposes of our discussion there are two important aspects to highlight (Apendix A contains a more detailed description):

- The time for which an object is valid in the universe of discourse can be recorded in the database if it is marked with a T (for a detailed discussion of the temporal semantics of the ERT, refer to [MSW92]). Rules which use that object will need to be written to correctly access and maintain this valid-time record.
- Any object with is derived by a rule from occurrences of other objects is drawn using a dashed line. At least one rule must be provided which contains that object in its consequence, and which thus can be used in the latter information system stage as a derivation rule.

As an example of the ERT language in use, Figure 3 describes a simple application, used for calculating the costs of purchases and producing receipts in a super-market cash register. The customer entity class has been marked with a T since we want to keep track of the periods customers exist for. The customer has address relationship has been T-marked because we want to save the various addresses at which a customer has been living during its life. The total_cost value class is derived from the sum of the line_costs (themselves derived from price and qty), and hence is drawn using dashed lines.

3.2 The PID

The PID is an extension of standard dataflow diagrams [GS79]. It is based on the process language of the PPP Language [GLW91], but has been revised to fit with the TEMPORA paradigm. The PID describes the operation of an information system as a set of cooperating processes. In the PID, processes and their interaction are specified in a formal way. The interaction between business processes is modelled

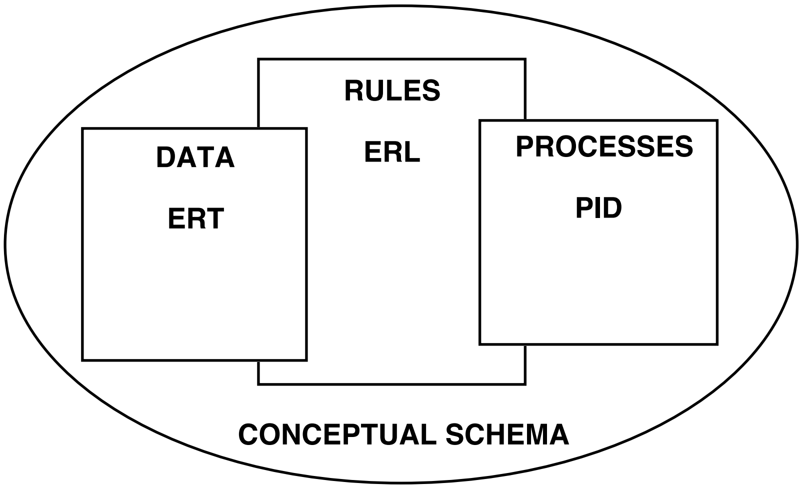


Figure 2: The TEMPORA conceptual schema

using the concept of flows. Every process may in turn be decomposed into a set of cooperating processes. The processes are successively decomposed into lower-level processes until an appropriate level of decomposition is achieved. The basic modelling concepts of a PID are:

- **Processes**, representing tasks and work procedures in the business. A process in one dataflow diagram may later be decomposed to reveal its inner structure in second diagram.
- **Dataflows**, representing the flow of information between processes that occur in the business. Flows are denoted as triggering or non-triggering.
- **ERT views**, representing persistent data, and described in the ERL as subsets of the ERT model.
- **External agents**, representing the agents lying outside the context of the business function being modelled.

A simple example of a PID diagram is shown in Figure 6. The PID modelling language of TEMPORA is used as a means for capturing rules needed to build a working information system, and for grouping those rules into transactions. As such, it will be brought into use after the initial business rule modelling has been conducted.

The basic structure of rules implementing a process is dictated by the structure of the PID model. The various parts of the rule which are dictated by the PID diagram for a non-decomposed process are as follows:

- All triggering input flows to the process must appear in the when (trigger) part of the rule.
- All non-triggering flows to the process must appear in the if (condition) part of the rule.
- All output flows appear in the then (consequence) part of the rule.
- The rule uses no other flows, but may use **predicates**, which do not have a representation in the PID model. A predicate can be viewed as a rule generating a flow, and takes the form if $\langle cond \rangle$ then $\langle flow \rangle$. Such predicates have the semantics of a Prolog language statement $\langle flow \rangle \vdash \langle cond \rangle$.

The anonymous flows from ERT views correspond to an ERL **ERT access expression**, which is an ERL expression contain the ERT objects that are associated with the views. The above four rules apply to the location of the expressions within the rule. The condition part of the rule will invariably have additional ERL expressions to 'process' the information contained on the input flows before putting on the output flows. The trigger and consequence may also have such additional expressions, though this will be must less common. An example of the coupling between process and rules is given in Figure 7.

3.3 The ERL

The ERL is used for three purposes; firstly to formally state business rules in a precise manner, secondly to describe additional static information about the data schema that can not be represented graphically in the ERT, and thirdly to describe the behaviour of each non-decomposed process in the PID model. For the purposes of our current discussion, we state that ERL rules are of the general form when *trigger* if *condition* then *consequence*, and that a rule is selected at any given moment in time if:

- the *trigger* has just become true at this **transaction-time** (i.e. the time of execution [Jen94]).
- the *condition* is true when asked as a query on the temporal database at the valid-time associated with the transaction-time.

If the rule is selected, then the *consequence* must also be true. It is a matter of information systems design to decide if this is done by making the rule a:

- **constraint**: a runtime error is raised if the consequence is not true.
- **derivation rule**: the consequence is taken as being true, and may be used in the conditions of other rules as a fact.
- **event-action rule**: the consequence is forced to be true by updating the database and creating new instances of flows.

Since the ERL syntax is designed to be simple to understand, we shall only illustrate the language here by giving a few examples. A brief description of ERL can be found in Appendix B, and a fuller description in [MNP+91].

- State that the quantity associated with a receipt line should never exceed the quantity of the article it is for. Note the use of square brackets allows the use of more than one relationship leaving the same entity.

  ```
  if receipt_line.X [has qty.Q1, refers_to article has qty.Q2]
  then Q1 <= Q2
  ```

- State that instances of the line_cost value class are a consequence on a simple calculation on the qty and price associated with the same receipt_line (hence use of .R after receipt_line).

  ```
  if receipt_line.R [has qty.Q, refers_to article has price.P]
  and C = Q *P
  then receipt_line.R has line_cost.C

  if T = sum{C for_which receipt.R contains receipt_line has line_cost.C}
  then receipt.R has total_cost.T
  ```

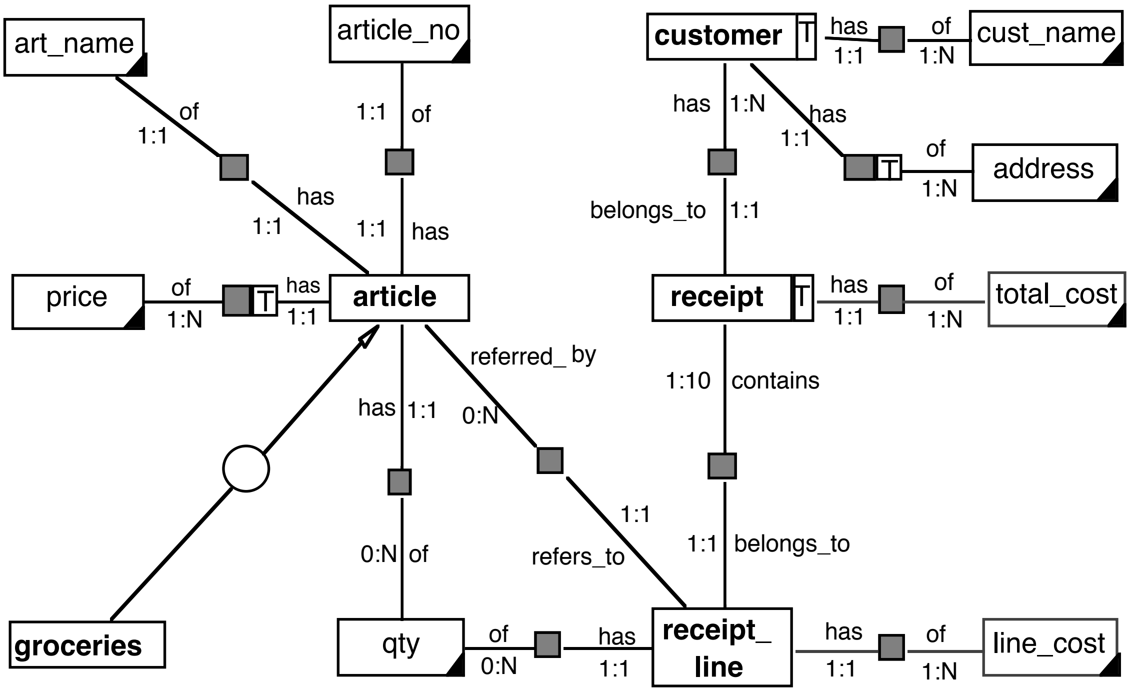


Figure 3: A conceptual schema for a supermarket application

- We state that the price of an article may never fall.

  if (just_before article.X has price.P1)
  and article.X has price.P2
  then P1 <= P2

- just_before is a temporal operator indicating that the query is made with respect to state immediately previous to the state in scope outside of the operator.

3.4 Rule Structuring

In Tempora two types of rule structuring are provided:

- Inter-language links between the ERT/PID diagrams, and groups of rules.
- Intra-language links between rules, called necessitates, and motivates, which record the design process where one rule requires the creation of (necessitates) or leads to the creation of (motivates) another rule.

Figure 4 summarises the rule links described above, together with the links provided between the PID and ERT. Figure 5 gives an example of a rule hierarchy (i.e., inter-rule links).

The inter-rule links enable browsing of the rule set according to their use in describing entities, relationships, values and processes. The intra-rule links allow a very flexible approach to rule structuring to be taken. For example, we can group all rules implementing a certain policy by introducing at a high-level a rule that expresses in natural language the need for having such policies, and then introducing motivates relationships from that rule to the existing rules implementing the policy.

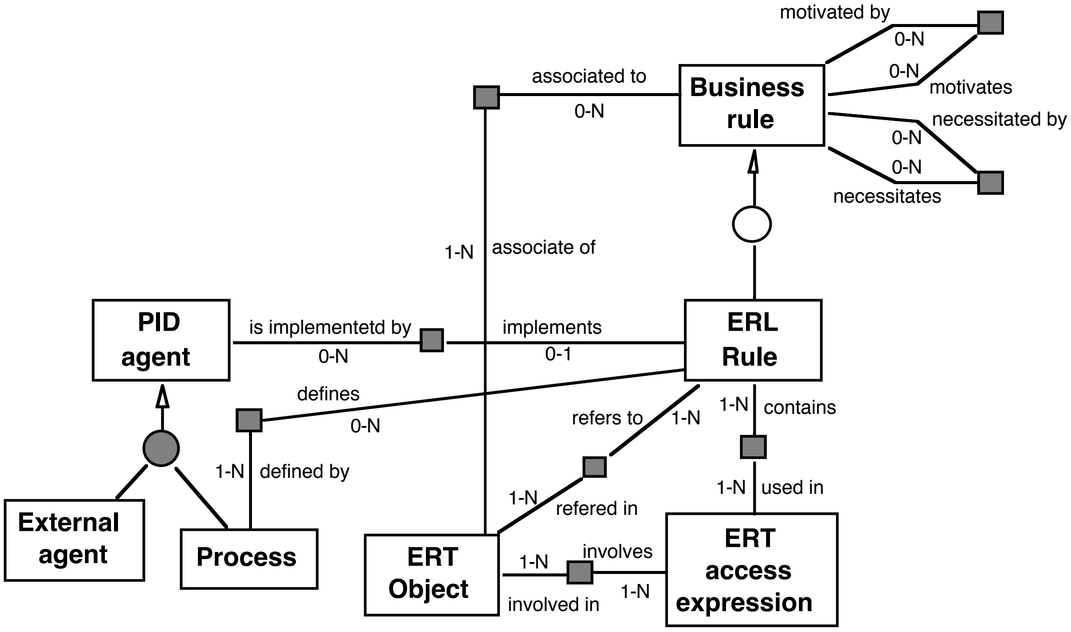


Figure 4: A meta-schema for the TEMPORA modelling languages

4 Business Rule Capture

At an early stage of the TEMPORA project, it was clear that comprehensive capture support for rules is necessary for successful systems development using the TEMPORA languages. Therefore, considerable effort has been targeted at developing means to assist the analyst in capturing rules. We aimed at extending and elaborating our modelling approach to improve the process of capturing rules. In particular, we have concentrated on providing methodological guidelines to help the analyst in this process of formulating rules correctly whenever it is logical and convenient for a rule to be specified.

First we outline the major activities involved in rule capture in TEMPORA. Then three particular methods of supporting some of these activities are elaborated.

4.1 Major Activities

Rule capture will involve the following activities (but necessarily in the order given):

1. *Elicitation of rules independent of ERT and PID.* Rule elicitation may take place in a number of ways. The methods used are **goal modelling** and **driving questions** (see below). In this way, a number of (typically informal) rules defining the goals can be captured. We may also envision more advanced methods to rule elicitation such as techniques/approaches from the field of knowledge acquisition.
2. *Elicitation of rules using the ERT model.* To support the elicitation process we may ask questions such as: What are the policies associated with this object, What are the constraints associated with this relationship, etc. There is a trade-off between the complexity of the ERT diagram and its understandability. Instead of introducing objects or relationships to represent a complex requirement (especially if they are not used much in the domain), it is often better to introduce one or more predicate rules, and associate these rules to the object, rather like **views** in RDBMS or **methods** in OODBMS.

3. *Elicitation of rules using the PID model.* There are at least three ways that the PID may be used to support the elicitation process. First, for each process one may identify the event-action rules that describe the behaviour of the process. Second, for each process one may identify the rules that govern the behaviour of the process, that is, the rules that describe the constraints of the process' behaviour. Rules associated with other components of the PID specification such as ERT views and timers may also be captured. Third, PID diagrams may be useful to give overview of the UoD in a user-friendly manner.

4. *Refinement of rules.* Rules may be refined in three ways. Firstly, by finding answers to the question how is this rule upheld? In this way the details of high level or abstract rules can be captured. Secondly, decision tables can be employed for finding the various cases in rules that need a combination of conditions to hold. This is often the case for rules that implements processes in the PID. Thirdly, when rules are formalised in ERL there is usually a need to have several formal rules for each informal one. This is due to the ambiguity of natural language and that informal rules tend to be more abstract.

5. *Formalisation of rules.* The results of the activities 1-3 are predominantly expressed using informal text. There is nothing that prevents the rules being defined directly in ERL, but this entails that the developer must be extremely experienced with the language. It is also likely that the formal rule will become very extensive making it hard to understand. Therefore it is recommended that the formalisation is performed in one of the manners described in activity 4. The result of activities 1-4 should be informal rules with a sufficiently detailed text that it can be formalised directly into a small number of formal rules.

6. *Conceptualisation of interrelationships between rules and between rules and other models.* Structural links to other models and links to other rules are specified.

4.2 Driving Questions

When conducting ERT and PID modelling we use a set of driving questions acquired during long experience of modelling with these types of languages [Wan93, WWW93]. Experience from case studies showed that a corresponding set of driving questions for rules was needed. In particular, it was important that these questions could guide the elicitation process with respect to domain issues as well as aspects of the rule language that was perceived difficult to use. For instance, temporal aspects were difficult to capture without guidelines to direct the modelling process. We have defined a set of driving questions based on experience gained in case studies [WWW93]. Examples of general questions are: '*What are the policies of the business?*', '*What are the restrictions on the business?*', and '*What external laws and regulations pertain to the business, in particular what relationship do they have with objects in the ERT, PID and ERL?*'. Examples of more specific questions related to a particular rule:

- What objectives does the rule contribute to fulfilling? How else is this objective realised?
- Is this rule a more narrow variant of some more general rule?
- Is the rule valid for all instances of this class or are there exceptions?
- If a rule is valid for a certain concept type, does it also apply to all specialisations or generalisations for this concept type?
- If a rule is valid for a certain concept which is a specialisation, does the rule apply also to other concepts based on the same specialisation criterion?

In addition to the types of inquiries given above, more specific questions can be formulated for particular classes of domains/applications. Such questions may have to do with types of rules and policies that normally appear in a certain area and a set of templates may be developed for each area.

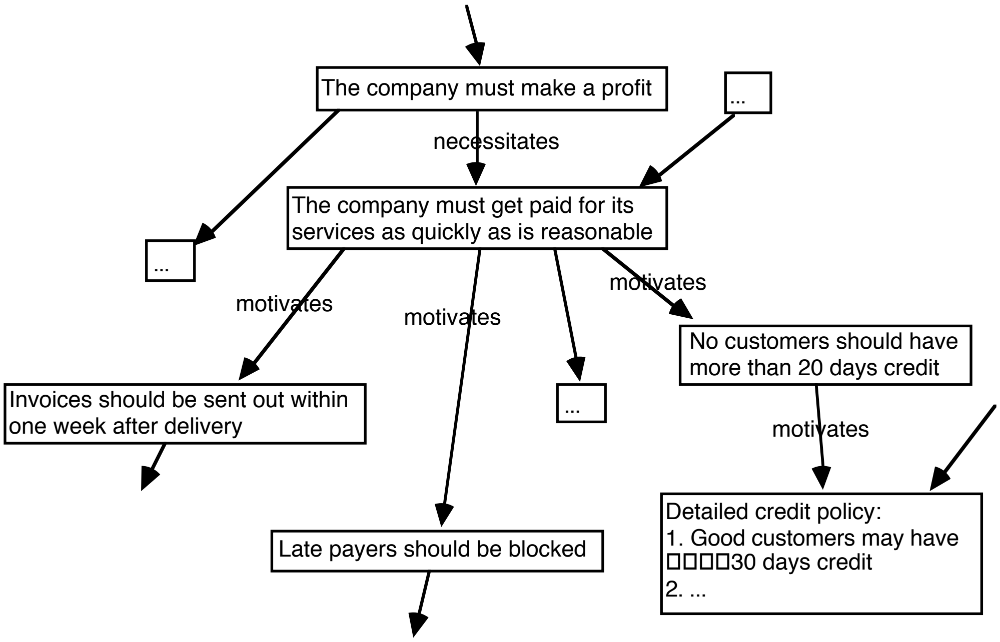


Figure 5: Business rule decomposition

4.3 Capturing Business Rules Using a Goal-oriented Rule Model

Tempora aims at viewing the development of an information system as the task of developing and augmenting the policy knowledge base and to model explicitly all business rules of the organisation in the information system. However, talking about rules especially at higher levels in the organisation, implies that we have means to express the vagueness characterising the information at these levels. Examples are an organisation's business concepts and long term goals. Experience from the Sweden Post case study [Wan93, WWW93] indicates that there is a need to elicit such rules and provide mechanisms to support the acquisition of such rules.

Goal modelling was introduced as a mechanism for capturing rules at high levels of abstraction. In performing business modelling, business rules are arrived at by repeatedly analysing business goals in order to specify how they should be accomplished (cf., Section 2). At higher levels, objectives are general and vague, but in the analysis they are refined to become more concrete in terms of business objects and business processes. The ERT and PID models also become more detailed. Rules defined on less abstract levels will by necessity be more specific since they deal with more specific things. Figure 5 shows a simple example of a breakdown of high-level goals to more precise subgoals/rules.

In the goal model you specify for instance, how to treat your customers, what the characteristics of your products should be in terms of quality, availability, etc. It also describes what should guide the business processes, for example what kind of policy the organisation should follow in relation to travelling expenses, tax regulations, etc. In the goal model it is possible to capture all those high level business rules that do not directly belong to a particular business process or a certain resource, but which are of more general importance for the whole enterprise. Goal modelling also allows for redundancy in the specifications.

However, due to the amount of information, the vagueness of high-level goals, redundancy, and the fact that not all rules derive from the primary business objectives but from restrictions forced upon the business by external sources, it may not be practicable to conduct such goal/rule analysis in a systematic way for the whole business area under consideration. Instead it may be used for parts of the business to support the rule capture. Goal/rule decomposition is also an important framework for understanding business rules.

At later stages of modelling, it will be necessary to either state the business rule in the ERL, or ensure that other rules exist (linked via motivates or necessitates links) which are described using the ERL. When translating a business rule stated using natural language into the ERL, we first must consider which parts of the rule fill the three sections of the basic rule structure when *trigger* if *condition* then *consequence*:

- In the *trigger* part of the rule, we describe what must have just become true for the rule to be true. The trigger should contain the events which must just have occurred if the rule is relevant.
- In the *condition* part of the rule, we describe what needs to be true for the rule to be considered. The condition should contain statements which must be true (and may have been true for some time) if the rule is relevant. If the condition is false, then the rule does not hold, and no action or effect will result.
- In the *consequence* part of the rule, we state what must be true if the condition is true. The execution of a TEMPORA specification will attempt to ensure that the consequence is made true if the condition is true. If for any reason this can not be achieved, an error will be raised in the runtime system.

The trigger and the condition of the rule may be empty, but there must always be a consequence since otherwise the rule would make no statement about what must be true in the specification, and so would serve no purpose.

Example 1 Separating natural language rules into three clauses of the ERL

All articles with a price less than 50 øre will have their price increased to 50 øre.

- trigger: empty

  (*since this rule is not related to a specific event, but always true*)
- condition: article has price less than 50 øre
- consequence: article has price equal to 50 øre

4.4 Rule Capture Using the ERT

The ERT model provides a data dictionary of objects in the UoD. If our information system is to handle these objects, where are a set of driving questions centred around each object in the ERT model:

- does every object get created by appearing in at least once outside the scope of a not operator but in the then part of an event-action rule?
- does every object get deleted by appearing in at least once inside the scope of a not connective in the then part of an event-action rule?
- does the object serve any purpose in the information system, by appearing at least once in a rule in positions other than the first two itemised above.
- is the life history of all objects fully described by rules?

- if the object is T-marked in the ERT, indicating that its valid time history is to be kept, are there any rules that use the temporal operators in conjunction with that object such that the valid time history is made use of?
- do all objects have rules describing constraints on the cardinality, value ranges and life history progression, other than those already modelled graphically in the ERT?

4.5 Rule Capture Using the PID

The PID is used as a means for decomposing the business into parts which are possible to grasp and to formally describe. There are at least three ways that the PID may be used to support the elicitation process: (1) for each process one may identify the event-action rules that describe the behaviour of the process, (2) for each process one may identify the rules that describe the constraints of the process' behaviour, and (3) PID diagrams are useful to give overview of the UoD in a user-friendly manner (i.e., by exploiting the PIDs graphical representation and structural properties). To support the elicitation process, view generation may be useful to provide relevant subsets of the PID model.

The results of using the PID for rule elicitation in case (1) and (2) above are rules expressed using informal text. This text may be represented directly in a stepwise manner using an intermediate representation. In the latter case, the informal rules may be mapped down to the basic structures of ERL (when ... if ... then ...) where the fields may be informal text, as illustrated by Example 1. These are successfully elaborated to adhere to the formal syntax and semantics of ERL.

By combining the process based and rule based approaches, a flexible representation and presentation of rules is achieved [KMOS91]. That is, both a form based representation (i.e., corresponds to the ERL) and a graphical representation of some aspects of rules are provided (i.e., by representing certain constructs of the rules in the PID). This coupling between the PID and ERL can be exploited to provide more comprehensive support to the process of rule conceptualisation (i.e., formalisation). If a PID specification of the relevant parts of the business exist, this diagram can be used as basis for rule generation. A skeleton of event-action rules from the process structure can be generated.

Consider, for instance, Figure 6 where a supermarket cashier system has been broken down to a level where one process (P1.2) is concerned with calculating the amount of money to pay for a specific item given its article number and the quantity.

Figure 7 illustrates how the process P1.2 is implemented by means of a single rule defining when and how the inputs of P1.2 are transformed to its output dataflows. It also shows the ERT-view defining the part of the database that is read.

As can be understood by Figure 7, the process P1.2 calculates the total cost of a purchased item to be entered on a single line of the receipt produced by the printer coupled to the cash register. It does this by reading from the database the price per quantity and name of the particular article. P1.2 also sends the cost to process P1.3 to be included in the final total of the receipt.

When performing PID modelling we could, in principle, specify all the relevant rules for objects on every level. However, we do not do this, since:

1. At high levels we do not have a detailed enough understanding of what goes on in all objects to be able to give a precise definition.
2. The amount of information in the rule at the higher level will often be little more than that expressed by the PID model.

Sometimes one may feel the need to specify functions involving elements of procedural code. If one tries to specify such a procedure on too high a level, there will be a risk of embedding a business rules in it.

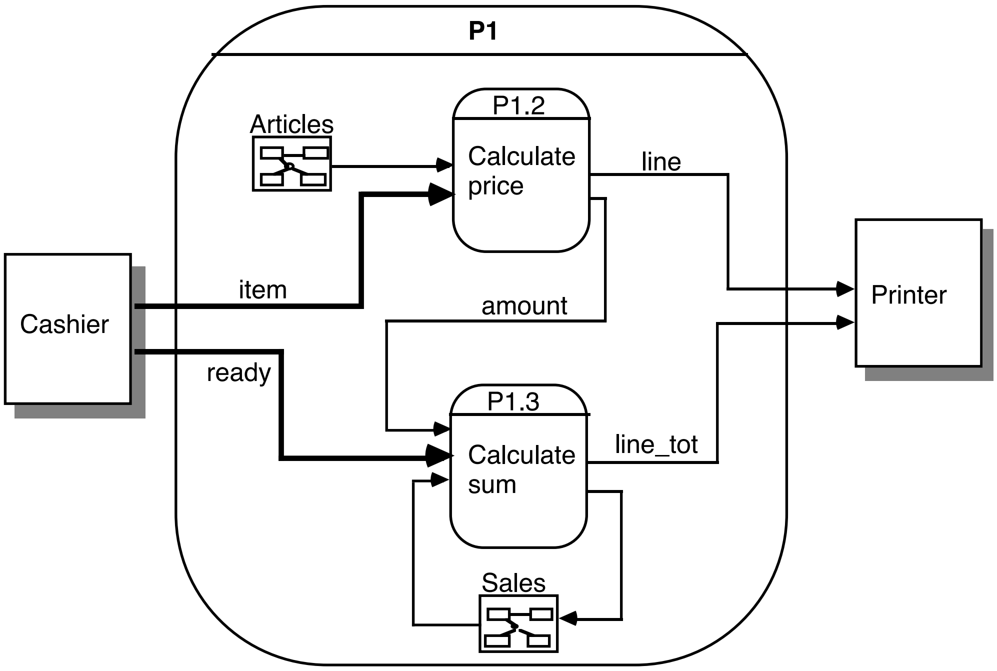


Figure 6: A cash register system

In order to avoid this, it is important to break down business functions beyond a point where business rules are hidden in such procedures, i.e. to a level where the processes can be described by a set of simple rules.

5 Concluding Remarks

The Tempora conceptual modelling languages, methods, and tools provide an environment suitable for rule capture and formalisation, using a combination of semantic data modelling in the ERT, rule specification in the ERL, and process modelling in the PID. A Tempora conceptual model consisting of the ERT, ERL and PID submodels constitutes a complete functional specification of an information system. Case studies [Wan93, WWW93] have been carried out that demonstrate the expressiveness and feasibility of the Tempora conceptual modelling languages and tools.

The Tempora systems development environment also comprises design and implementation tools that assist a systems designer in developing the target system from this functional specification, and a runtime platform for its execution.

The Tempora design discipline gives methodological advice as concerns rule elicitation. In particular, it provides a goal-oriented rule model, to supplement rules captured from the ERT and PID. The design discipline also provides specific rule elicitation techniques.

The Tempora system as a whole demonstrates the practically of using a rule-based approach for systems development.

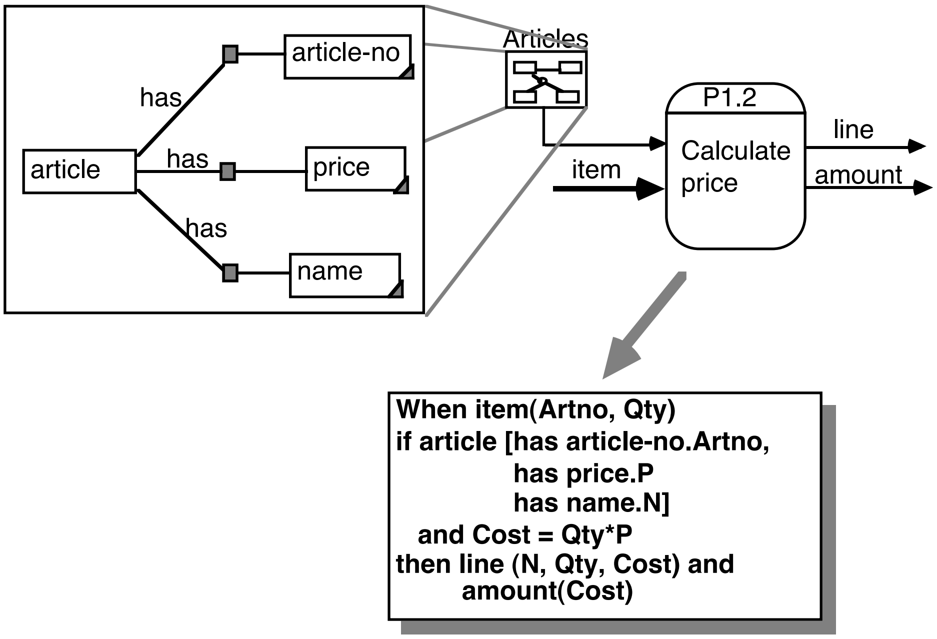


Figure 7: Part of a cash register system

Acknowledgements

The work reported in this paper has been partly funded by the Commission of the European Communities under the ESPRIT R&D programme. The Tempora project (number E2469) is a collaborative project between: BIM, Belgium; Hitec, Greece; Imperial College, UK; LPA, UK; SINTEF, Norway; SISU, Sweden; University of Lige, Belgium and UMIST, UK. SISU is sponsored by the National Swedish Board for Technical Development (STU-NUTEK), ERICSSON and Swedish Telecomm.

References

[AWS92] A. Anjewierden, B. Wielinga, and N. Shadbolt. Supporting knowledge acquisition: The ACKnowledge project. In *ESPRIT-92 Knowledge Engineering*, 1992.

[BAJ92] B.J.Wielinga, A.T.Schreiber, and J.A.Breuker. Kads: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 1992.

[CAS88] D. Conrath, V. De Antonellis, and C. Simone. A comprehensive approach to modeling office organization and support technology. In *IFIP WG8.4: Working Conference on Office Information Systems*, pages 73–92. North-Holland, 1988.

[Che76] P.P. Chen. The Entity-Relationship model — toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[deM78] T. deMarco. *Structured Analysis and System Specification*. Yourdon Press, 1978.

[DHL+86] E. Dubois, J. Hagelstein, E. Lahou, F. Ponsaert, and A. Rifaut. A knowledge representation language for requirements engineering. *Proc. of IEEE*, 74(10), 1986.

[GLW91] J.A. Gulla, O.I. Lindland, and G. Willumsen. PPP an integrated CASE environment. In *Proceedings of the Third Nordic Conference on Advanced Information Systems Engineering*, volume 498 of *LNCS*. Springer-Verlag, 1991.

[GS79] C. Gane and T. Sarson. *Structure Systems Analysis: Tools and Techniques*. Prentice Hall International, 1979.

[Har86] A. Hart. *Knowledge Acquisition for Expert Systems*. Kogan Page, 1986.

[Jac83] M. Jackson. *System Development*. Prentice-Hall, 1983.

[Jen94] C.S. Jensen *et al.* A consensus glossary of temporal database concepts. *SIGMOD Record*, 23(1):52–64, 1994.

[KMOS91] J. Krogstie, P.J. McBrien, R.P. Owens, and A.H. Seltveit. Information systems development using a combination of process and rule-based approaches. In *Proceedings of the Third Nordic Conference on Advanced Information Systems Engineering*, volume 498 of *LNCS*. Springer-Verlag, 1991.

[LMS+91] P. Loucopoulos, P.J. McBrien, F. Schumacker, B. Theodoulidis, V. Kopanas, and B. Wangler. Integrating database technology, rule-based systems and temporal reasoning for effective software: the TEMPORA paradigm. *Journal of Information Systems*, 1(2), 1991.

[Mac86] I. MacDonald. Information engineering: An improved, automatable methodology for designing data sharing systems. In *IFIP WG8.1 Working Conference on Comparative Review of Information Systems Design Methodologies: Improving the Practice*, 1986.

[MNP+91] P.J. McBrien, M. Niezette, S. Pantazis, B. Thedoulidis, G. Tziallas, A.H. Seltveit, U. Sundin, and R. Wohed. The TEMPORA external rule language. In *Proceedings of the Third Nordic Conference on Advanced Information Systems Engineering*, volume 498 of *LNCS*. Springer-Verlag, 1991.

[MSW92] P.J. McBrien, A.H. Seltveit, and B. Wangler. An entity-relationship model extended to describe historical information. In *Proceedings of CISMOD '92*, pages 244–260, Bangalore, India, 1992.

[PV88] B. Pernici and W. Vogel. A integrated approach to OIS development. In *ESPRIT '86: Results and Achievements*, pages 835–844. North-Holland, 1988.

[Ste89] L. Steels. COMMET: A componential methodology for knowledge engineering. Deliverable WP 2/3/4, Esprit Project COMMET, November 1989.

[Wan93] Benkt Wangler. Modelling the ISO-case with TEMPORA. Report, SISU, 1993.

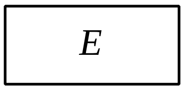
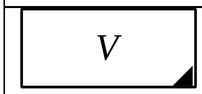
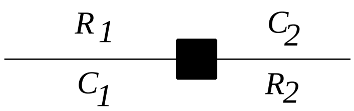
[WWW93] Benkt Wangler, Ulf Wingstedt, and Rolf Wohed. Requirements capture in practice: Experiences from a case-study at Sweden Post. Report, SISU, 1993.

[Xia91] L. Xiaofeng. What is so bad about rule-based programming? *IEEE Software*, 8(5):103–105, 1991.

A The Entity Relationship Time Language (ERT)

The basic elements of the ERT model are descibed in Table 1.

Note that value classes are treated as being as important in the model as entity classes, varying only in the meaning attached to the instances.

<table>
<tr>
<td>
</td>
<td>Entity Sets describe collections of similar entities, which we give an unique identifier to each of, called a surrogate. If E is an entity set in the ERT, and X is a variable, then
$E.X$ holds for $X \in E$.</td>
</tr>
<tr>
<td>
</td>
<td>Value Sets describe collections of values of a single type. If V is an value set in the ERT, and X a variable:
$V.X$ holds for $X \in V$.</td>
</tr>
<tr>
<td>
</td>
<td>Relationships describe sets of either (i) pairs of entities or (ii) pairs of one entity and one value. If E_1 and E_2 are entity sets[1], connected by R_1/R_2, and X, Y are variables, then:
$E_1.X R_1 E_2.Y$ hold for each $\langle X, Y\rangle \in R_1$
$\langle X, Y\rangle \in R_1 \equiv \langle Y, X\rangle \in R_2$
The cardinality of a relationship determines the minimum and maximum number of times any entity in the entity set may appear in the relationship. If C_1 is the cardinality attached to R_1:

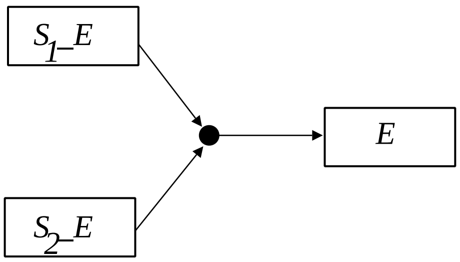
$C_1 = M_1 : N_1 \rightarrow$
$M_1 \leq \mathsf{count}(X \ \mathsf{for\_which} \ E_1.X \ R_1 \ E_2) \leq N_1$</td>
</tr>
<tr>
<td>
</td>
<td>Is-A Hierarchies are described by arrowed headed lines pointing from the subsets to the superset. Any instance of a subset must also be an instance of the superset:
$\forall x, y.((S_x\_E \cap S_y\_E = \emptyset) \vee (x = y)) \wedge S_x\_E \subseteq E$
The subset inherits the relationships of the superset:
$E_1.X \ R \ E_2.Y \wedge S_x\_E_1.X \rightarrow S_x\_E_1.X \ R \ E_2.Y$
In a total is-a hierarchy (denoted by a filled circle), each member of the superset must also be member of one subset. If $S_1\_E, \ldots, S_n\_E$ are total subsets of E then:
$E.X \rightarrow S_1\_E.X \vee \ldots \vee S_n\_E.X$
A partial is-a hierarchy (denoted by an open circle) allows certain members of the superset not to be members of the subset:
$E.X \not\rightarrow S_1\_E.X \vee \ldots \vee S_n\_E.X$</td>
</tr>
</table>

Table 1: The Non-temporal elements of the ERT modelling language

B Appendix B: The External Rule Language (ERL)

Four basic forms of rule are permitted, as listed in Table 2, where e_1, e_2 and e_3 are any valid expressions of the language:

ERL Rule	Semantics
e_1	The expression e_1 must hold at any moment or tick during the execution of the system.
if e_1 then e_2	If expression e_1 holds at any tick for some set of variable substitutions, then e_2 must also hold for the same variable substitutions.
when e_1 then e_2	If expression e_1 holds at any tick for some set of variable substitutions, but did not hold if it had been evaluated at the previous moment in real time, then e_2 must also hold for the same variable substitutions.
when e_1 if e_2 then e_3	If expressions e_1 and e_2 hold at any tick for some set of variable substitutions, and for which e_1 did not hold if it had been evaluated at the previous moment in real time, then e_3 must also hold for the same variable substitutions.

Table 2: ERL Rules

B.1 Access and Selection of ERT Sub-Model

We include the arithmetic relationships $=$, $= \backslash =$, $<$, $>$, $=<$ and $>=$ as notionally part of any ERT model, along with the data-types (values) of the model. Access if made to the ERT model by naming the sequence of entities, relationships and values in the order they appear in the model. The semantics of such expressions is described in Table 3.

ERL expression	Semantics
e_1	Holds for each instance of class e_1 in the ERT.
$e_1.v_1$	Holds for variable v_1 bound to each instance of class e_1 in the ERT.
$e_1 r_1 e_2$	Holds for each instance of class/value e_1 and ERT access e_2 for which there is a instance of relationship r_1 between the instance of e_1 and the leading instance of e_2.
$e_1.v_1 r_1 e_2$	As previous case, but also binds variable v_1 to the instance of e_1.
e_1 $[r_1 e_2, r_2 e_3]$	Equivalent to exists v_1 such_that $e_1.v_1 r_1 e_2$ and $e_1.v_1 r_2 e_3$
$e_1.v_1$ $[r_1 e_2, r_2 e_3]$	Equivalent to $e_1.v_1 r_1 e_2$ and $e_1.v_1 r_2 e_3$

Table 3: ERT access expressions