Towards DW Support for Formulating Policies

Deepika Prakash[1(✉)] and Naveen Prakash[2]

[1] Delhi Technological University, Delhi 110042, India
dpka.prakash@gmail.com
[2] ITM University, Gurgaon, India
praknav@hotmail.com

Abstract. Data warehousing is largely directed towards "what to do next" type of decisions that essentially address operational decision-making needs. We argue for developing data warehouse support for deciding on organizational policies: policies evolve and therefore need continual decision-making support. We propose an RE approach for discovering information contents of a data warehouse for policy decision-making. Each policy is represented in an extended first order logic that can be converted into a policy hierarchy. Each node in this hierarchy can be selected, rejected, or modified. In order to take this decision, the relevant information is determined by using Ends Information Elicitation and Critical Success Factor Elicitation techniques for information elicitation. The elicited information is converted into an ER diagram from which star schemas are obtained.

Keywords: Policy · Policy formulation · Directive · Requirements engineering

1 Introduction

A Data Warehouse, DW, provides information in an appropriately processed form to decision makers who then use it for decision-making. Thus, data warehousing provides the decisional perspective of an enterprise in contrast to transaction orientation of conventional enterprise models. Modeling of the information needs of a decision maker is now required and not of operational tasks/transactions.

Decision-making is required to bridge the gap between the expected and actual results produced by an organization. Thus, DW systems emphasize the 'operational' aspects of organizations, that is, what should be done next in order to have a beneficial influence on the operations of an organization. Imhoff and White propose to address multiple levels of decision makers in an organization in their DSS 2.0 [7]. They propose three kinds of business intelligence, BI, needs: Strategic BI is for meeting long term goals and its users are executives and business analysts; Tactical for taking initiatives to meet strategic goals having users as business analysts and managers; Operational for monitoring and optimizing business processes to be used by managers and operational users.

Whereas DSS 2.0 is based on an analysis of operational BI needs, OMG in its Business Motivation Model, BMM [10], and Prakash et al. [14] propose that organizations need to look at non-operational decision making as well.

J. Ralyté et al. (Eds.): PoEM 2015, LNBIP 235, pp. 374–388, 2015.
DOI: 10.1007/978-3-319-25897-3_24

1. BMM [10] proposes the notion of directives in its Means dimension. Directives may be policies and rules. BMM points out the need for formulating directives, that is on deciding what the policies and rules of an organization are. Thus decision making here is for formulating the environment within which operations of an organization shall be carried out.
2. Prakash et al. [14] see decision-making in an organization in three levels, formulation of policies, formulation of policy enforcement rules, and taking operation decisions that they call imperative decisions. The authors see these three levels as closely related to one another, policy level with policy enforcement level that, in turn, is related to the operational level.

Our analysis of policies brings out a number of properties of policies as follows:

a. **Policies evolve**: Policy evolution may occur due to (a) changing operational realities or (b) changing business realities like changes in the business environment or changes in regulations etc. This evolution implies periodic assessment and re-formulation of policies.
b. **Policies need specific information**: Policy formulation requires information so that both, the need for change is identified, and also the most appropriate policy is formulated. Thus, high attrition rates among doctors may indicate changes in Human Resource policies of a hospital. Notice also that a piece of information required for formulating a policy may not be of interest for operational decision making. Thus, for formulating policies, the attrition rates may be of interest whereas in the latter, the number of employees against the sanctioned employees only may be of interest. In general, we can say that policy formulation requires both operational as well as policy-specific data.

From the foregoing discussion we conclude that policy formulation is a good candidate for support through data warehouse technology because

1. Policy formulation is not a one-time task but is evolutionary in nature. It would therefore benefit from computer based decision-making support
2. A data warehouse is a good place to store both operational as well as policy specific information.

Since DW development starts off from the requirements engineering phase, **our research question is how can we do Requirements Engineering, RE, for a Policy Data Warehouse?**

We start by noting that a policy [10] governs strategies/tactics that are in turn, the means to obtain the desired results expressed as goals and objectives. This suggests goal orientation as a starting point. However, we believe that adopting goal-orientation data warehouse requirements engineering [1–4, 8, 12, 13] would make our requirements engineering process very heavy. This is because we would (a) determine goals, (b) arrive at the strategies/tactics for goals, (c) discover policies for strategies/tactics and, lastly, (d) elicit information relevant to each policy and structure it in multi-dimensional form. Evidently, the number of steps here is quite large and involved. On the other hand, we notice that a rich source of policies is often available:

- From regulatory bodies, organizations that lay down standards etc. For example restaurants and hotels are covered by municipal and hygienic norms, hospitals by health norms, educational institutions by accreditation standards and so on.
- Other organizations carrying out similar business have their own policies that constitute best practice.
- An organization may have its own legacy policies.

Since starting from policies directly, makes for a relatively lighter approach, we propose **a reuse-based requirements engineering technique** that exploits such 'given' policies. For each policy, we can accept, modify, or reject either the total policy itself or any component that constitutes it. Thus, we set up a choice set, {select policy component, modify policy component, delete policy component} and our decisional problem is the selection of an alternative from this set. As we have already seen, this selection requires information. Thus, **our problem is to elicit policy-specific information and then to structure it in multi-dimensional form**. We propose to address this problem in four steps as follows.

A. **Represent policies**: We propose to represent policies in first order logic. This is in accordance with the principle given in article 5.3 of the Business Rules Manifesto [15]: expressions in logic are fundamental to represent business rules as business people see them. From an operational point of view, the structuring of a statement in logic over simpler well defined formulae allows us to examine each sub-formula (recursively) for selection, deletion, modification as discussed above. As in SBVR [11] we can then develop a natural language capacity over the theoretical underpinning of the first order logic.
B. **Determine the nature of Information**: In order to elicit information, we need a model that identifies what information is to be elicited. Our model tells us that besides detailed, aggregated, and historical information kept in data warehouse, we additionally need 'categorization information.
C. **Information Elicitation**: We need to elicit information *relevant* to each component of every policy. To establish this 'relevance', recall that a policy [10] is a directive governing courses of action. Courses of action, in turn, are the means to achieve ends. Thus, relevant information is about achievement of ends by policies and policy components. There are two aspects of ends. Since the success of managers [16] depends on a portfolio of Critical Success Factors, managers formulate policies that are beneficial to these factors. Thus, we need to elicit CSF-relevant information for each policy/policy component. Additionally, we need to obtain information about achievement of organizational goals/objectives by policies. For this, we use an elicitation technique that provides Ends-relevant information.
D. **Information Structuring**: The elicited information is to be organized in multi-dimensional form. For this, we propose to convert the elicited information as an ER diagram and then use available techniques [5, 9] to convert the ER schema into star schema form.

The next four sections address these four steps respectively. In Sect. 6, we present the results of a case study. Section 7 is a discussion of our work in relation to existing

literature. In Sect. 8, we conclude by summarizing our work and pointing out directions for the future.

2 Representing Policies

We represent policies in the first order logic with extensions to allow variables for sets of values. The logic is defined as follows.

There are two kinds of variables, those that denote a single value, SV, and others that denote a set of values, CV. A *simple term, ST,* can either be a constant, an SV, or an *n*-adic function symbol applied to *n* SVs. A *collective term, CT,* is either a CV or an *n*-adic function symbol applied to *n* CVs.

An *atom* is an n-place predicate $P(x_1, x_2, ----, x_n)$ where any x_i is either ST or CT. There are standard predicates for the six relational operators named EQ (x, y), NEQ (x, y), GEQ (x, y), LEQ (x, y), GQ (x, y), LQ (x, y).

The formulae of the logic are defined as follows:

- Every atom is a formula
- If F1, F2 are formulae then F1 AND F2, F1 OR F2 and Not F1 are formulae
- If F1, F2 are formulae then F1 → F2 is also a formula
- If F1 is a formula then ∃sF1 and ∀sF1 are formulae. Here s is SV or CV.
- Parenthesis may be placed around formulae as needed
- Nothing else is a formula.

The precedence while evaluating the formulae is as follows:

- Universal and existential quantifiers, ∀, ∃
- Logical AND, OR, NOT

We give below two examples to illustrate the foregoing. The first example uses quantification over an SV whereas the second shows quantification over a CV.

Policy 1: Every Ayurvedic hospital must run an O.P.D.

$$\forall x[\text{Ayu}(x) \rightarrow \text{run}(x,\text{OPD})]$$

where Ayu(x) says that x is Ayurveda hospital and Run(x, OPD) says that x must run an OPD.

Policy 2: A Semi-private ward must have area of 200 Sq. ft. and 2 beds.

$$\forall x \exists B[\text{spw}(x) \rightarrow \text{EQ}(\text{area}(x), 200)\text{AND EQ}(\text{count}(B),2)]$$

where spw(x) says that x is a semi private ward, EQ(x, c1) says that x is equal to c1, and B is a set of beds.

We convert the well-formed formula that expresses a policy into a policy hierarchy. For this purpose, we view an expression of our logic in two parts, one on the LHS of the implication and the other on its RHS. These parts are decomposed till atoms are reached. The algorithm for arriving at the policy hierarchy is given below. We illustrate the algorithm by considering the policy "Every doctor must have a post graduate degree".

$$\forall x[doc(x) \rightarrow degree(x, MD)]$$

where doc(x) says x is a doctor and degree(x, MD) says that x has MD degree.

Algorithm: To generate policy hierarchy tree for a given policy
Input: Policy as formula
Output: Policy hierarchy
{
addroot(formula, root) //add policy as root of the tree
if formula contains quantifiers //(of the form ∀x or ∃x)
f_Q = formula minus quantifiers extracted from formula
addchild(f_Q, root) //add f_Q to root
if f_Q contains implication '→'
f_L = formula on left side of implication
f_R = formula on right side of implication
PT_L= makept(f_L) //make a postfix tree, makept, from f_L
Attach_pt_as_child(PT_L , f_Q) //attach the postfix tree, pt, to f_Q
PT_R = makept(f_R,) //make postfix tree from f_R
Attach_pt_as_child(PT_R , f_Q) //attach the postfix tree, pt, to f_Q
}

Step 1 of the algorithm gives us the root node, the full statement of the policy itself. In step two, we strip off the quantification from this statement and then attach the result to the root node as shown in Fig. 1. Thereafter, we make the postfix trees for doc(x) and degree(x, MD) respectively. This yields the nodes doc(x) and degree(x, MD). These are then attached as shown.

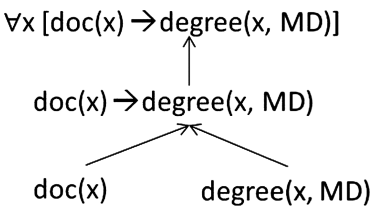


Fig. 1. Policy hierarchy for "Every doctor must have a post graduate degree"

Now, we associate **the choice set {select, modify, reject}** with each node of the policy hierarchy. If sufficient information is available in the Data warehouse, then an alternative from this choice set may be picked up to formulate the new policy using the process as follows:

Repeat until root of hierarchy is reached
Pick node in bottom up left right manner in policy hierarchy
Refer to Data Warehouse
Pick alternative from choice set

Applying this to our example of Fig. 1, the choice set {select, modify, reject} applies to the two atoms doc(x) and degree(x, MD) respectively. The node doc(x) identifies that the policy under formulation is about doctors and this continues to be the case. Therefore, doc(x) is selected. Now consider the second node degree(x, MD). This is an important node since a policy decision about the minimum qualifications of doctors is being formulated. This decision requires information about the types of cases in our hospital, mortality rates, cases with post treatment complications, successfully discharged patients, admitted patients, patients refused admission due to non-availability of specialized treatment and so on. Let it be available in the policy data warehouse.

After consulting the relevant information, let the node (degree, MD) be modified to (degree, PDCC) The resulting leaf level of the new policy is shown in Fig. 2. The next higher node in the hierarchy is associating doc(x) with degree(x, PDCC) and we do not require any additional information to make this association. The root node provides quantification and it needs to be decided whether we should use existential or universal quantification. Again, after consulting relevant information let the latter be selected. The resulting policy is shown in Fig. 2.

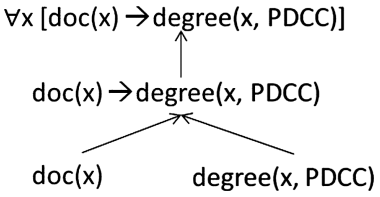


Fig. 2. The desired policy hierarchy

3 The Nature of Information

Figure 3 shows our meta-model, expressed in ER form, that elaborates the different types of information to be elicited for our warehouse. This model is an adaptation of the one of [14] and includes the 'categorized by' relationship.

As shown in the figure, Information is simple or aggregate. Simple Information corresponds to detailed information. Aggregate information is obtained as a summary by computing from simpler information. This is shown by the 'Is computed from' relationship between Aggregate and Information. Further, it is possible for information to be categorized as, for example, happens when the GROUP BY is used in SQL. This means that there is a categorization relationship between information and information as shown. Aggregations may be applied to categorized or non-categorized information.

Historical information is represented by the relationship 'History of' between Information and Temporal unit (for example, hours, minutes, seconds). When a temporal unit is associated with information then we must also know the number of years of history to be maintained. This is captured, as shown in the figure, by the attribute Period.

The cardinality of 'history of' shows that it is possible for information to have no temporal unit associated with it. In such a case, only current information is to be

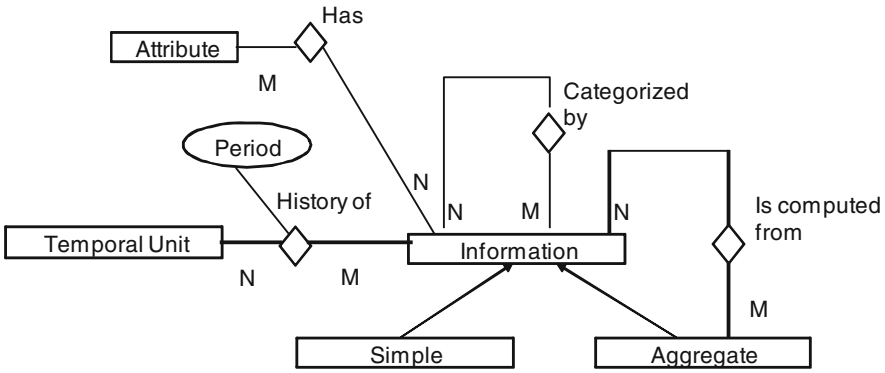


Fig. 3. Information meta-model

maintained. Further, it is possible for more than one temporal unit to be associated with Information. This allows the identification of multiple temporal units that can later be converted into temporal dimension hierarchies.

4 Eliciting Information

In this section we present the broad architecture of the requirements elicitation tool. Thereafter, we consider the two elicitation techniques of CSF Information and Ends Information elicitation.

4.1 Tool Architecture

All policies are processed by the Policy Hierarchy Maker and stored in a Policy Base of Fig. 4. The Policy Hierarchy Maker constructs policy hierarchies and stores them in textual format in the Policy Base after associating it with the name of the organization for which the policy is being formulated, the domain to which the organization belongs, and the policy type as follows.

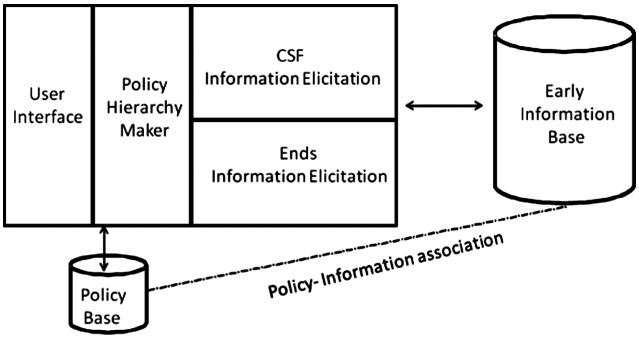


Fig. 4. The elicitation tool

The organization of the policy base is rooted in a two dimensional classification of a policy having policy type and domain as dimensions. The former dimension results from our treatment of an organization as a function. This allows us to partition policies as those governing input, output and the processing being performed, respectively. Input policies are concerned with the infrastructure, material and other inputs. Output policies deal with the amount and nature of output. Process policies provide the properties of the processing being carried out. In addition to input, output, and process there is a fourth kind that we call outcome policies. These specify the impact of the output on the organization. Along the domain dimension, policies are viewed in the context of domains, medical, life sciences, engineering etc.

Before starting a fresh policy formulation session, the requirements engineer must enter the domain and the name of the organization for which the policy is being formulated. This may be, for example, educational domain and DIT respectively. Upon getting this information, the tool can access domain specific policies available in the policy base. These can be retrieved by the requirements engineer in three ways as follows:

- Domain wise retrieval: all policies for the domain can be retrieved
- Policy Type wise: Policies of a domain but of input, output, process and outcome types are retrieved.
- Individual policies: Policies can also be retrieved based on specific terms in the policy. For example, information for all policies containing the term 'area' can be searched.

4.2 Elicitation Techniques

The model shown in Fig. 3 drives information elicitation. That is, the focus is to elicit for each node of a policy, simple information, aggregate information, category, history etc. We illustrate our elicitation techniques for the policy that for all in-patient departments, IPD, there must be at least one doctor, y, with MD degree, degree(y, MD), and the number of doctors, D, must be at least 2, GEQ(count(D), 2). This is

$$\forall x \exists y \exists D[IPD(x) \rightarrow doc(y) \text{ AND } GEQ(count(D),2) \text{ AND } degree(y, MD)]$$

The hierarchy of this policy is shown in Fig. 5.

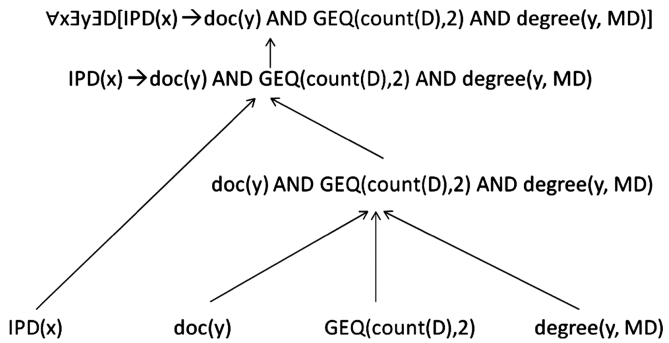
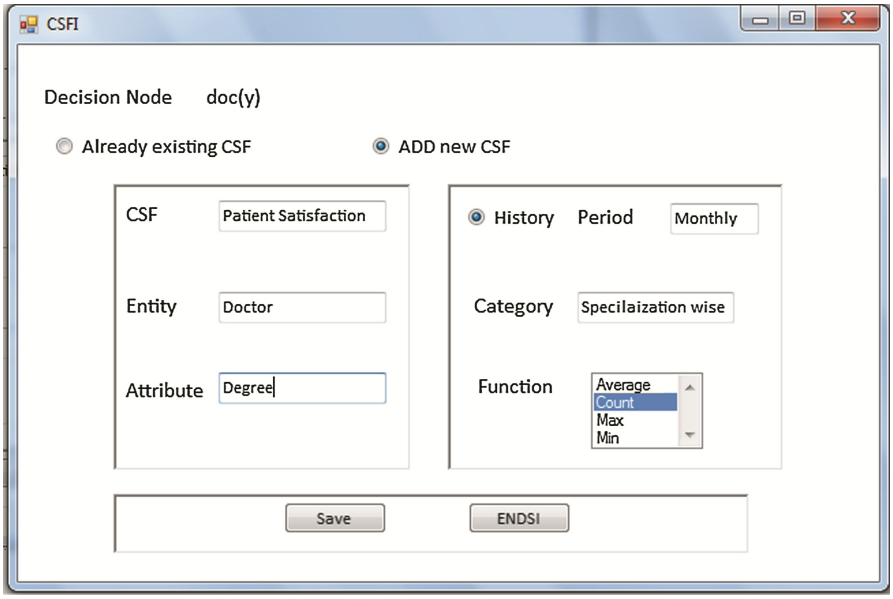


Fig. 5. Policy hierarchy

CSFI Elicitation. Critical Success Factor Information, CSFI, elicitation is a three step process consisting of (a) CSF [16] determination, (b) determination of information needed to assess CSF achievement and (c) determination of nature of the information in terms of the concepts of our meta model of Fig. 3. CSFI elicitation is terminated when all policies have been processed in this manner.



The top of the screen shows the node for which information is being elicited followed by the two options, to select an existing CSF or to add a new CSF. The panel on the left hand side of the screen shows an elicited CSF, Patient Satisfaction, and one piece of information required to assess it, Doctor, together with one attribute of this information, Degree. The panel on the right hand side allows statement of whether history is to be maintained and, if so, then the period for which it is to be maintained for one month in; the categorization of the information (specialization wise); and the function to be applied on the information,

Now, consider our example policy of Fig. 5. Let there be a CSF, 'patient satisfaction'. This determines the CSF (step (a) above). For step (b), we examine each node in the policy hierarchy to determine information needed to assess CSF achievement. For the node IPD(x), we obtain patient satisfaction measures for the in-patient department as a whole, (i) speed of patient admission, (ii) promptness of medical aid, (iii) delivery of para-medical services (iv) effectiveness of discharge procedures, (v) cases referred to other hospitals etc. Thereafter in step (c), the information model of Fig. 3 is populated with the elicited information and, consequently, aggregates, history requirements etc. are obtained.

Once IPD(x) is exhausted, we move to the next node, doc(y). Here, again, information relevant to satisfaction of 'patient satisfaction' is elicited (i) number of doctors in each specialization, (ii) experience of each doctor, (iii) roster of doctors, (iv) doctor

availability for patient etc. However, unlike with IPD, a historical record of this information is required. The process of CSFI elicitation contains in this way for each node of the policy hierarchy.

The information obtained through CSFI elicitation is kept in the early information base of Fig. 4.

Ends Information Elicitation. The steps in Ends Information elicitation, EI elicitation are (i) determining Ends, (ii) determining the effectiveness measures of the Ends, and (iii) determining the information pertaining to evaluating the effectiveness. The user interface is similar to the one for CSFI elicitation and is not shown here for reasons of space.

Consider our policy of Fig. 5 again. One of the ends of the hospital is to provide comprehensive patient service (step a). The next step is to determine the effectiveness measures of this end. Again, we move in a bottom up manner. Consider IPD(x). For the in-patient department we determine effectiveness measures, (i) referrals made to other hospitals, (ii) medical services hired. For each of these, we now elicit in step (c), the information for evaluating it. For the former, we get (i) daily number of operations over a one year period, (ii) daily referrals over the last year, (iii) disease/case for which referral was done, (iii) inward referrals by other hospitals, etc. Similarly, the second effectiveness measure, medical services hired, is examined to find information like consulting doctors on panel etc.

As before, the elicited information is used to populate the model of Fig. 3 and the ENDSI elicitation process is repeated for each node of the hierarchy. The elicited information is kept in the early information base of Fig. 4.

5 Building ER Schema

Once CSF Information and Ends Information elicitation have been carried out for all candidate policies, the early information base of Fig. 4 is fully populated. We can now move build an ER diagram. We assume that name conflicts have been resolved and a given name refers to the same concept, for example, Doctor of CSF Information elicitation and Doctor of Ends Information elicitation are the same.

Construction of the ER diagram is done in two steps as follows.

Step 1: Building individualized ER diagrams for each policy: When constructing ER diagrams for individual policies, in addition to the entities elicited above, we obtain entities from the statement of the policy itself using our guideline that *All quantified simple variables refer to entities*. This follows from the notion of the first order logic that variables denote real world entities. Thus, in our example policy of Fig. 5, the bound simple variables, SV of Sect. 2, are x and y and these range over IPD and doctor respectively. Our heuristic says that IPD and doctor are two entities of the ER diagram being built. Notice that D denotes a set of variables and is not simple. Therefore, it does not identify any entity.

Step 2: Integrating individual ER diagrams: Once all individual ER diagrams are obtained, these are to be integrated into one ER diagram. As mentioned in the Introduction to this paper, we take recourse to existing schema integration techniques for this

purpose. The integrated schema represents the required ER information to be kept in the DW To-Be.

For reasons of space, we do not present an example of ER construction.

6 Illustrative Case

We have applied our method to the medical domain. We considered a traditional Indian medical system which offers treatments in Ayurvedic medicine, Yoga, Unani and Naturopathy. The regulatory body is the AYUSH council. It defines policies [6] that all hospitals offering treatment using traditional medicine must comply with.

Our tool was used for the case study. The Policy Base was populated with the policies of AYUSH council. Our task was to formulate the policies of another hospital based on the AYUSH system of medicine. Since domain knowledge was required to carry out the analysis an expert in the AYUSH domain was involved in the case study. The expert used the tool and identified the information necessary to formulate policies. In eliciting information a team of doctors was consulted so that relevant information could be obtained.

The complete list of policies for AYUSH was represented in our logic. We constructed the policy hierarchy for each policy and elicited information. The statistics of the full case are as follows:

- Total number of Policies/Hierarchies = 151
- Number of nodes = 732
- Total number of decisions = 732 * 3 = 2196. This is because there are 3 alternatives, select, modify, reject for each node.

Our experience is that the structure of the policy hierarchy has a close bearing on the information elicited. The factors that influence this are as follows:

(a) Intermediate nodes that arise due to connectives like AND and OR
(b) Duplicate nodes on the left and right hand sides of the implication.

We consider each of these in turn.

Intermediate nodes have the potential to discover information in addition to that discovered from their component nodes. Therefore, wherever possible, a policy should be structured to yield intermediate nodes. We form the heuristic.

The intermediate node heuristic has the potential to increase the total number of nodes in a policy hierarchy. In contrast the common node heuristic considered below decreases the number of nodes to be considered.

Intermediate Node Heuristic: Merge policies using AND

We illustrate the application of this heuristic through an example. Consider the policy "A general ward must have 600 sq. ft. area, minimum 10 beds, at least 30 patients". We can represent this as three separate policies as follows. Each policy has its own policy hierarchy.

$$\forall x[G(x) \rightarrow EQ(area(x), 600)]$$

$$\forall x \exists B[G(x) \rightarrow GEQ(count(B), 10)]$$

$$\forall x \exists P[G(x) \rightarrow GEQ(count(P), 30)]$$

However, applying our heuristic we get

$\forall x \exists B \exists P[G(x) \rightarrow EQ(area(x), 600)]$ and GEQ(count(B), 10) and GEQ(count(P), 30)]

This yields one extra node for ANDing all the individual components of the policy. The policy hierarchy is shown in Fig. 6. The intermediate node yields information additional to that discovered from the individual nodes, for example, number of patients with communicable diseases, number of cases of patient-to-patient disease transfer.

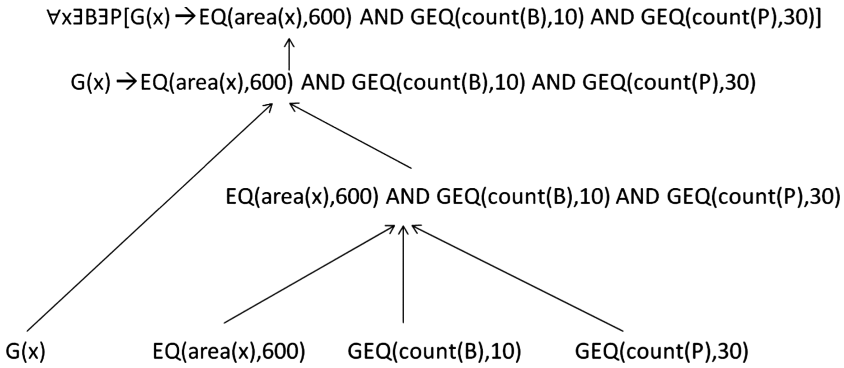


Fig. 6. The policy hierarchy

From the point of view of obtaining information, it is profitable to include intermediate nodes rather than otherwise because one needs to discover all information and not miss out on some.

Our experience is that merging policies using OR does not have an effect on information.

The intermediate node heuristic has the potential to increase the total number of nodes in a policy hierarchy. In contrast the common node heuristic considered below decreases the number of nodes to be considered.

Common Node Heuristic: For common LHS nodes between policies, elicit information only once.

Nodes can be common between policies. When nodes on the LHS of the implication are common then the information is also found to be common. For example consider two policies:

Every AYUSH hospital must have Total Patient beds ratio not higher than 1:6

$$\forall x \exists B \exists P[AYUSH(x) \rightarrow condratio(count(P), count(B), \leq, 1, 6)]$$

Bed occupancy rate for every AYUSH hospital is 50 percent

$$\forall x \exists B \exists BOCC[AYUSH(x) \rightarrow percent(count(BOCC), count(B), 50)]$$

The node AYUSH(x) occurs on the LHS of both these policies. Evidently, the same information shall be discovered for this node in both the cases. Thus, we form our heuristic that information elicitation needs to be done only once.

The implication of the Common Node Heuristic is that the number of nodes to be considered decreases. Notice however, that common nodes on the RHS do not reduce the number of decisions to be examined. This is because RHS nodes are dependent on the LHS nodes for their information elicitation context.

A comparison before and after the application of our heuristics is given below in Table 1:

Table 1. Result of heuristic application

	Before	After
Number of nodes in policy hierarchies	732	530
Number of common nodes on LHS	105	Nil
Number of decisions	2196	1590

While it is possible to establish a relationship between information and component of a policy in our elicitation techniques, it is difficult to establish inter-relationship between information through the user interfaces of Sect. 4. This requires domain knowledge and is to be separately determined.

7 Discussion

If we consider the area of Requirements Engineering holistically then we find a paucity of techniques based on reuse of requirements products. For example, in goal-oriented that carries a lot of momentum, we are unaware of any proposal for reuse of goals. This is true not only for goal-orientation in data warehousing but also in traditional, transactional systems. Perhaps, this is because policies are available in the public domain whereas goals are more personal to organizations and businesses.

Even though a policy hierarchy may seem similar to a goal reduction hierarchy three points must be noted

(a) There is a conceptual difference between goals and policies: whereas goals define what the system should do, policies are directives that govern behavior. As a result interest in goals is to determine whether they are functional or non-functional, soft or hard etc. In contrast, policies can be necessary or obligatory [11].
(b) The policy hierarchy is a different **form** of a policy expressed in our logic. No new component of a policy is discovered in building the policy hierarchy. In contrast, construction of a goal hierarchy is done to discover sub-goals. The goal hierarchy is thus not merely a different form of a goal.
(c) The policy hierarchy can be directly used for eliciting information relevant to the retention, modification, or rejection of a node. In contrast, in traditional transactional systems, the goal hierarchy forms the basis of making goals operational in

real systems. When goal-orientation is applied to data warehousing, new notions like decision [12, 13], decisional goals, information goals [8] need to be introduced.

Ends analysis has been used in goal-orientation for goal reduction. We need to interpret Ends differently, what parameters are relevant to estimate Ends achievement and what is the specific information along each of these parameters. This is to arrive at the needed information to be kept in the data warehouse.

Further, we are unaware of the use of Critical Success Factors in goal-oriented data warehouse requirements engineering techniques [2–4, 8, 12, 13]. Indeed, we are not aware of these being used in goal–orientation in transactional systems as well.

Finally, we have separated the task of eliciting the needed information from that of structuring it. For the latter, we structure the elicited information, first as an ER diagram and the, in the second step, use existing techniques for conversion of ER diagrams into star schemas. This separation makes for a better focus on the elicitation task. In contrast to this, data warehouse requirements engineering has information structuring as its major concern. The proposal of Mazon et al. [8] does not propose information elicitation techniques but determines facts and measures form their intentional, decisional, and information goals. The goal-service-measure approach of [1] determines service measures from goals, organizes these in a Structured ER Model, SERM, and then arrives at the multi-dimensional structure. It is similar to our approach. However, moving from goals to services is ad-hoc, experience based and not based on guided elicitation like ours is.

8 Conclusion

Since policy assessment and re-formulation is periodically done in an organization, computer based support in the form of a data warehouse would be beneficial. The first step in development of such warehouses is that of requirements engineering. Our proposal is to reuse policies. We represent policies in first order logic that is converted into a hierarchy. The choice set, {select, modify, reject} is associated with each node in this hierarchy. Thereafter, we use Critical Success Factor Information Elicitation and Ends Information Elicitation to discover the information relevant to each alternative of the choice set. This information is converted to an ER diagram for which we have proposed some guidelines. We assume that existing techniques to are thereafter used to convert the ER into multi-dimensional form and have not addressed this issue here.

Experience with our case study leads to the formulation of the Intermediate Node and Common Node heuristics.

Our future work is motivated by our observation that policies are formulated by taking not only policy-specific but also operational, OLTP, information into account. The integration of policy-specific and operational information into one unified data warehouse is therefore an important question. What happens if this integration is not done, what are the problems that arise? Should we not develop techniques that promote integration at the requirements stage itself? These are some of the questions that we expect to address in the future.

References

1. Boehnlein, M., Ulbrich vom Ende, A.: Deriving initial data warehouse structures from the conceptual data models of the underlying operational information systems. In: Proceedings of Workshop on Data Warehousing and OLAP (DOPLAP), pp. 15–21, USA (1999)
2. Boehnlein, M., Ulbrich vom Ende, A.: Business process oriented development of data warehouse structures. In: Proceedings of Data Warehousing 2000. Physica Verlag (2000)
3. Bonifati, A., Cattaneo, F., Ceri, S., Fuggetta, A., Paraboschi, S.: Designing data marts for data warehouses. ACM Trans. Software. Eng. Methodol. **10**(4), 452–483 (2001)
4. Giorgini, P., Rizzi, S., Garzetti, M.: GRAnD: a goal-oriented approach to requirement analysis in data warehouses. Decis. Support Syst. **45**(1), 4–21 (2008)
5. Golfarelli, M., Maio, D., Rizzi, S.: Conceptual design of data warehouses from E/R schemes. In: Proceedings of HICSS-31, vol. VII, pp. 334–343 (1998)
6. GOI. Department of AYUSH, Ministry of Health and Family Welfare, Government of India, No. Z.20018/4/2000-ISM (Tech)/HD Cell, National Competitive Bidding (2000)
7. Imhoff C., White C.: DSS 2.0 -http://www.b-eye-network.com/view/8385
8. Mazón, J.-N., Pardillo, J., Trujillo, J.: A model-driven goal-oriented requirement engineering approach for data warehouses. In: Hainaut, J.-L., Rundensteiner, E.A., Kirchberg, M., Bertolotto, M., Brochhausen, M., Chen, Y.-P.P., Cherfi, S.S.-S., Doerr, M., Han, H., Hartmann, S., Parsons, J., Poels, G., Rolland, C., Trujillo, J., Yu, E., Zimányie, E. (eds.) ER Workshops 2007. LNCS, vol. 4802, pp. 255–264. Springer, Heidelberg (2007)
9. Moody, L.D., Kortink, M.A.R.: From enterprise models to dimensional models: a methodology for data warehouses and data mart design. In: Proceedings of the International Workshop on Design and Management of Data Warehouses, pp. 5.1–5.12, Stockholm, Sweden (2000)
10. OMG 14. Object Modeling Group, Business Motivation Model, release 1.3, OMG document number, dtc/14-11-07. http://www.omg.org/spec/BMM/1.3/
11. OMG-Semantics of Business Vocabulary and Business Rules (SBVR), vol. 1.0. http://www.omg.org/spec/SBVR/1.0/PDF
12. Prakash, N., Gosain, A.: Requirements driven data warehouse development. In: CAiSE Short Paper Proceedings, pp. 13–17 (2003)
13. Prakash, N., Gosain, A.: An approach to engineering the requirements of data warehouses. REJ **13**(1), 49–72 (2008)
14. Prakash, N., Prakash, D., Gupta, D.: Decisions and decision requirements for data warehouse systems. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 92–107. Springer, Heidelberg (2011)
15. Ross R.G.: Business Rules Manifesto, Business Rules Group, version 2.0 (2003)
16. Wetherbe, J.C.: Executive information requirements: getting it right. MIS Q. **15**(1), 51–65 (1991)