A
Dissertation
On

Eliciting Information Requirements For Data Warehouses

Submitted in fulfilment of the requirements of the degree of

Doctor of Philosophy

(Computer Engineering)

By

DEEPIKA PRAKASH

(**University Roll No. 01/Ph.D./CoE/2009**)

Under the supervision of:

Dr. Daya Gupta,

Professor, Dept. of Computer Engineering, Delhi Technological University (DTU)

To the

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

NEW DELHI – 110042

2016

DECLARATION

I, Deepika Prakash, student of Ph.D. (Roll No: 01/PhD/CoE/2009), hereby declare that the thesis entitled "*Eliciting Information Requirements for Data Warehouses*" which is being submitted for the award of the degree of Doctor of Philosophy in Computer Engineering, is a record of bonafide research work carried out by me in the Department of Computer Engineering at Delhi Technological University.

I further declare that the work presented in the thesis has not been submitted to any University or Institution for any degree or diploma.

Date: _______________ ____________________________

Place: New Delhi

Deepika Prakash
(**Candidate**)
01/Ph.D/CoE/2009
Department of Computer Engineering
Delhi Technological University (DTU)
New Delhi- 110042

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

CERTIFICATE

Date: ______________

This is to certify that the work embodied in the synopsis entitled "***Eliciting Information Requirements for Data Warehouses***" submitted by ***Deepika Prakash*** with ***Roll No. 01/PhD/CoE/2009*** as a *part-time* research scholar in the Department of Computer Engineering, Delhi Technological University, is an authentic work carried out by her under my guidance and is submitted to Delhi technological University for the award of the Degree of **Doctor of Philosophy**.

The work is original and has not been submitted, in part or full, to any other University or Institution for the award of any other degree.

Supervisor

____

Dr. Daya Gupta,

Professor,

Department of Computer Engineering,

Delhi Technological University (DTU)

New Delhi - 110042.

ACKNOWLEDGEMENT

First and foremost, I am indebted to three most important and amazing people in my life. To my parents, a very big THANK YOU. They supported me emotionally through this journey and provided me tremendous support. I believe I learn from the BEST. To my sister, Kelika, yahoo, I made it this far. THANK YOU for being my friend in the many, many moments of crisis. **I dedicate this thesis to all three of them** for their endless love, support and encouragement and for giving me the strength to chase my dreams.

I would like to extend thanks to the many people who so generously contributed to the work presented in this thesis. Special mention goes to my supervisor, **Prof. Daya Gupta**. My Ph.D. has been an amazing experience and I thank her wholeheartedly, not only for her academic support, but also for giving me so many wonderful opportunities. She has been a truly dedicated mentor and had constant faith in my work.

Similar, profound gratitude goes to **CBIP group** for helping me in developing confidence through various presentations and for sharing their experience and providing valuable feedback.

I would also like to thank **Dean (IRD)** and Dr. Anil Kumar **AR (IRD).** I extend my gratitude to **Prof. O.P.Verma (**DRC Chairman) and to **Prof. Vishal Verma (**Ph.D. Coordinator).

Finally, but not the least, thank you to all my friends. Your friendship made this journey even more enjoyable. I cannot list all the names, but you are always on my mind.

Deepika Prakash

November, 2015

ABSTRACT

Data Warehouse support in terms of Requirements Engineering models and techniques has been extensively provided for operational level of decision making. However, it is increasingly being recognized that there are other forms of decision making that exist in an organization. These are strategic in nature and ‘above’ operational decision making.

This thesis addresses the issue of providing decision making support to both strategic and operational decision making in the same DW system. The solution starts by defining two broad categories of decisions for which decision support is needed, one for policy enforcement rule (PER) formulation decisions and the other for operational decisions. Both kinds of decisions are structured based on a *generic decision meta-model* developed here.

The process starts by developing two Data Warehouses, one for policy enforcement rules and the other for operational decisions. In order to identify the needed information for supporting decision making, a set of *generic techniques for eliciting information* is proposed. This information is stored in the DW. Again, the structure of information for the two DWs is based on a *generic information meta-model* developed here.

The two DWs are *integrated upstream* in the requirements engineering phase using an *integration life cycle* proposed in this thesis. It is argued that there is a need for integration following the problems of inconsistency and loss of business control that can occur. This is due to common information and differing refresh times between the two Data Warehouses.

Further, three tools were developed to provide computer support for arriving at information for (a) PER, (b) operational decisions and (c) integrating information. This process was validated using AYUSH policies.

PUBLICATIONS FROM THE THESIS

INTERNATIONAL JOURNAL PAPERS:

1. **Prakash, D**. and Gupta, D. (2015) 'Data Warehouse requirements engineering: an emerging discipline', *International Journal Business Information Systems* (*in press*).
   [Scopus Indexed]
2. **Prakash, D**., and Gupta, D. (2014). Eliciting Data Warehouse Contents for Policy Enforcement Rules. *International Journal of Information System Modeling and Design (IJISMD), 5*(2), 41-69.
   [Scopus Indexed]
3. **Prakash, D**., and Gupta, D. (2014). Data Warehouse Integration at the Requirements Engineering Stage: A semi-automated Approach. *Business and Information Systems Engineering. Springer.* (communicated on 04/01/2015)
   [SCI-Expanded, Scopus Indexed]

INTERNATIONAL CONFERENCE PAPERS:

1. Prakash, N., **Prakash, D.**, and Gupta, D. (2011). Decisions and Decision Requirements for Data Warehouse Systems. In *Information Systems Evolution* (pp. 92-107). Springer Berlin Heidelberg.
   [Scopus Indexed]

LIST OF ABBREVIATIONS

BMM	Business Motivation Model
BRG	Business Rules Group
CADEI	Computer Aid For Decision Early Information elicitation
CC	Complex-Complex
CG	Connectivity Graph
CIM	Computation Independent Model
CREWS	Co-operative Requirements Engineering With Scenarios
CRUD	Create Retrieve Update Delete
CS	Complex-Simple
CSF	Critical Success Factor
CSFI	Critical Success Factor Information
CSO	Choice Set Objective
CT	Collective Term
CV	Set Variable
DB2	IBM DataBase 2
DM	Data Mart
DR	Decision Requirement
DSS	Decision Support System
DW	Data Warehouse
DWARF	Data WArehouse Requirements deFinition technique
DW-ENF	Data Warehouse-Extended NFR Framework

DWop	Operational Data Warehouse
DWper	Policy Enforcement Rule Data Warehouse
DWRE	Data Warehouse Requirements Engineering
EI	Early Information
EI_{op}	Operational level Early Information
EI_{per}	PER level Early Information
ELISPE	ELiciting Information Support for Policy Enforcement
ELISO	ELiciting Information Support for Operations
ENDSI	ENDS Information
EQ	Equal to
ER	Entity Relationship
ETL	Extraction Transformation Load
FR	Functional Requirement
GBRAM	Goal-Based Requirements Analysis Method
GDI	Goal – Decision – Information
GEQ	Greater than Equal to
GORE	Goal-Oriented Requirements Engineering
GQ	Greater than
GQM	Goal-Question-Metric
GRAnD	Goal-oriented Requirement Analysis for Data warehouses
IS	Information System
JAD	Joint Application development
KAOS	Knowledge acquisition in automated specification of software
LEQ	Less than Equal to
LHS	Left Hand Side

LQ	Less than
MD	Multi-Dimensional
MDA	Model Driven Development
MEANSI	MEANS Information
MOLAP	Multi-Dimensional Online Analytical Processing
NEQ	Not Equal to
NFR	Non-Functional Requirement
NLP	Natural Language Processing
OLTP	Online Transaction Processing
PER	Policy Enforcement Rule
PIM	Platform Independent Model
QVT	Query-View-Transformation
RAD	Rapid Application Development
RE	Requirements Engineering
RHS	Right Hand Side
ROLAP	Relational Online Analytical Processing
SBRE	Scenario Based Requirements Engineering
SC	Simple-Complex
SD	Strategic Dependency
SDM	Strategic Dependency Model
SE	Software Engineering
SOM	Check in paper
SQL	Structured Query Language
SR	Strategic Rationale

SRM	Strategic Rationale Model
SS	Simple-Simple
ST	Simple Term
SV	Single Value variable
TRE	Transactional Requirements Engineering
Tropos	Towards requirements-driven information systems engineering
UML	Unified Modeling Language
V&V	Verification and Validation
VMOST	Vision, Mission, Objectives, Strategies, Tactics

Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Data Warehouse Requirements Engineering

This chapter starts off by discussing the different definitions of Requirements and Requirements Engineering, RE followed by a short discussion on the phases of Requirements Engineering. The chapter reviews the techniques for eliciting requirements for functional systems and the reasons for developing a separate set of techniques for Data Warehouse Requirements Engineering (DWRE). Finally, the state of the art in DWRE is reviewed. Using this discussion, the problem of the thesis is arrived at and subsequently the approach to solving the problem is discussed.

1.1 Importance of Requirements Engineering: Failure Statistics

Origins of Requirements and Requirements Engineering, RE, lie in software engineering (SE). SE aims to deliver the needed functionality in the hands of the user. A functional system is for transactional activity. Here requirements are elicited from users, collected, prioritized and a system specification is made.

Over the last almost twenty five years, the importance of requirements engineering, RE, in the systems development life cycle has been well recognized. Information/software systems developers realized that one major factor that goes into systems failure is the inadequate attention paid to requirements formulation (Coman and Ronen, 2010; Flyvbjerg and Budzier, 2011). Indeed, the effect of poorly engineered requirements ranges from outright systems rejection by the customer to major reworking of the developed system.

The *Software Hall of Shame* (Charette, 2005) listed about 30 large software-development projects that failed between 1992 and 2005. These failures arise because projects go beyond the actual needs or because of expansion of the scope of the original project (Charette, 2005; Coman and Ronen, 2010; Flyvbjerg and Budzier, 2011). According to studies conducted at Bell labs and IBM (Hooks and Farry, 2001), of all defects encountered in software development, 80% are in the requirements phase. Boehm and Papaccio (Boehm and Papaccio , 1988) said that the cost of correcting requirements errors is 5 times when done during the design phase, 10 times during implementation phase, 20 times during testing and 200 times after the system has been delivered. The result is expensive products at best and total rejection of software at worst. This is corroborated by the Standish group (Standish, 2003) that reports "incomplete requirements" was one of the reasons that "Challenged" and "Failed" projects had in common.

1.2 Definition of Requirement and Requirements Engineering

A widely accepted definition of a requirement is the one given by (IEEE).

Definition 1: A requirement as defined in (IEEE Standard, IEEE-Std '610, 1990) is "*(1) a condition or capability needed by a user to solve a problem or achieve an objective, (2) A condition or capability that must be met or possessed by a system or system components to satisfy a contract, standard, specification or other formally imposed documents, (3) A document representation of a condition as in (1) or in (2)".*

Requirements thus arise from user, general organization, standards, and government bodies. These requirements are then documented.

Requirement has also been defined for a product by Robertson, and also by Kotonya.

Definition 2: "Something that the product must do or a quality that the product must have" (Robertson and Robertson, 2012)

Definition 3: "A description of how the system shall behave, and information about the application domain, constraints on operations, a system property etc." (Kotonya and Sommerville, 1998)

As goal oriented techniques developed, requirement was defined with respect to a goal.

Definition 4: "A requirement specifies how a goal should be accomplished by a proposed system" (Anton, 1996)

By (Sommerville, 1995) as

Definition 5: "Requirements are high level abstractions of the services the system shall provide and the constraints imposed on the system".

Requirements have been classified as functional, FR and non-functional requirements, NFR. A number of definitions exist as follows:

Definition 6: Functional requirements are "statements about what a system should do, how it should behave, what it should contain, or what components it should have" and Non-functional requirements are "statements of quality, performance and environment issues with which the system should conform". (Sutcliffe, 2002)

Definition 7: "Non-functional requirements (or quality attributes, qualities, or more colloquially "-ilities") are global qualities of a software system, such as flexibility, maintainability, usability, and so forth" (Mylopoulos *et al.*, 1999)

Requirements Engineering

Once the concept of requirements is defined, the next question is about understanding how requirements can be obtained and modelled. RE deals with both obtaining and modelling of requirements. Indeed, a number of definitions of RE exist in literature.

Definition 8: Requirements Engineering (RE) is defined (IEEE Standard, IEEE-Std '610, 1990) as "*the systemic process of developing requirements through an iterative cooperative process of analyzing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of understanding gained*".

The process is cooperative because different stakeholders have different needs and therefore varying viewpoints. RE must take into account conflicting views and interests of users and stakeholders. Capturing different viewpoints allow conflicts to surface at an early stage in the requirements process. Further, the resulting requirements are the ones that are agreeable to both customers and developers.

A number of definitions for the goal-oriented perspective exist. A selection is presented below.

Definition 9: According to (Zave, 1997), "*Requirements engineering is the branch of software engineering concerned with the real-world goals for functions of and constraints on software system. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software.*"

This definition incorporates "real world goals" in its definition. In other words, this definition hopes to capture requirements that answer the "why" of software systems. Here, the author is referring to “functional requirements”. Further, the definition also gives emphasis to “precise requirements”. Thus quality of requirements captured is also important.

Definition 10: “RE (Lamsweerde, 2000) is concerned with the identification of goals to be achieved by the envisioned system, the operationalization of such goals into services and constraints…."

Definition 11: For (Nuseibeh and Easterbrook, 2000) “*software systems requirements engineering (RE) is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation*”.

Here the emphasis is on *identifying stakeholders* and capturing the requirements of the stakeholders. “Communication” is also given importance along with analysis and implementation.

1.3 Requirements Engineering as a process

The input and output of the RE process of (DeMarco and Plauger, 1979) is shown below in Fig 1.1. Stakeholders are the problem owners. They can be users, designers, system analysts, business analysts, technical authors, customers. In the RE Process, requirements are elicited from these sources. Output of the process is generally a set of agreed requirements, system specifications and system models. The first two of these are in the form of use cases, goals, agents or NFRs. System models can be object models, goal models, domain descriptions, behavioural models, problem frames etc.

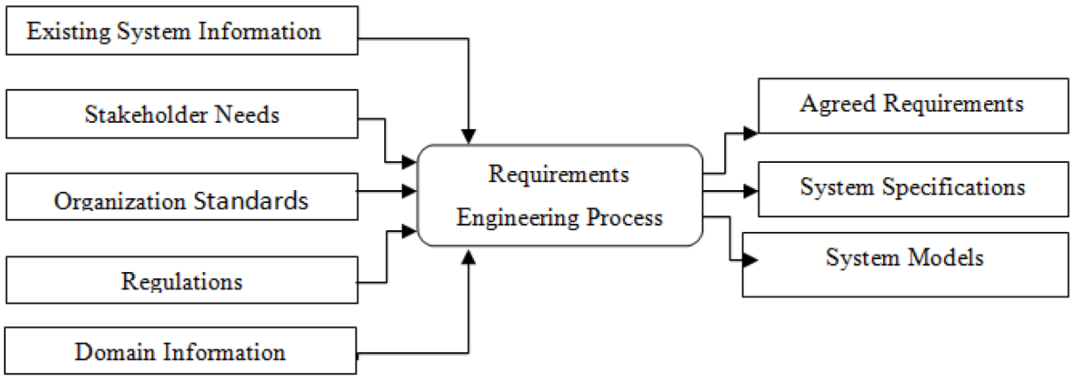


Fig 1.1: Input and Output to requirements engineering process (DeMarco and Plauger, 1979)

There are three fundamental concerns of RE namely, understanding the problem, describing the problem and attaining an agreement on the nature of the problem. The process involves several actors for the various activities.

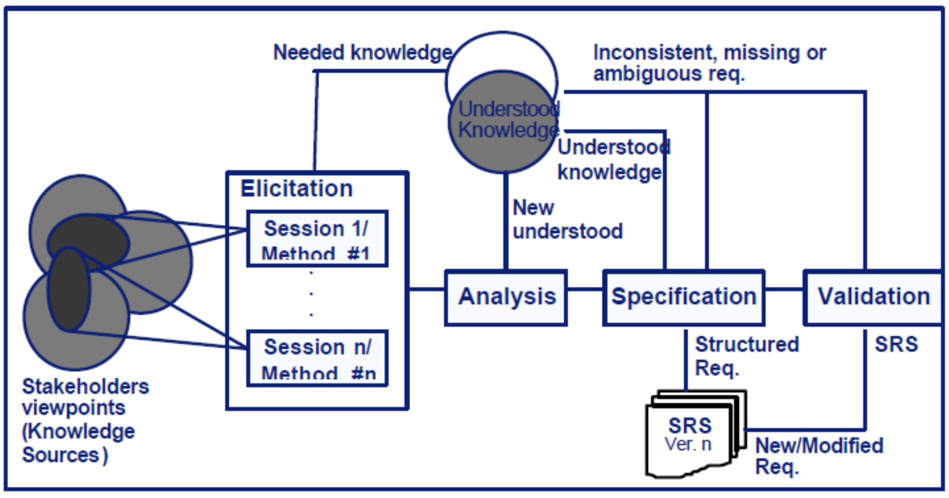


Fig 1.2: The Requirements Engineering process (Zang, 2007)

The Requirements Engineering process of (Zang, 2007) is shown in Fig 1.2. The activities of requirements development include:

- '**Elicitation**': Requirements are elicited from users, domain experts, literature, existing software that is similar to the one to be built, stakeholders etc. The requirements engineer is involved in identifying different sources, as well as acquiring and finding the relevance and significance of the requirements. As can be seen from Fig 1.2, several sessions are conducted and further different sessions can employ different methods to elicit requirements. The requirements are then presented in the form of models. This is a labour intensive step and usually takes a large amount of time and resources.

- **‘Analysis/ Negotiation’**: An agreement between the various stakeholders on the requirements of the system to-be, is established. Conflicts arise due to different views and goals of the stakeholders and are resolved by a facilitator.
- ‘**Specification** and **Documentation’**: Requirements are documented for use in subsequent system development. Formal specification languages, knowledge representation languages etc. are used to document requirements. Any inconsistencies, missing requirements found during the validation phase can also be fed back into this task. From the Specification and documentation task, one can go to the negotiation task (e.g. if conflicting requirements are found) or elicitation task (if more information is required). System analyst and domain experts are involved in this task.
- ‘**Verification** and **Validation’ (V&V)**: The main goal is to check if the document meets the customers’/clients’ needs. Consistency and completeness are some aspects of the document that are checked. The project manager and the client are involved in this task.

There are two phases of RE, an early phase and late RE phase. Early RE phase focuses on whether the interest of stakeholders is being addressed or compromised. Elicitation and negotiation form early RE phase. Late RE phase focuses on consistency, completeness and verification of requirements (Yu, 1997).

In recent years, much emphasis is placed on agile development. Agile methodologies are oriented to solving the problem of dealing with large number of requirements. Here, methods that involve natural language processing (NLP), machine learning (clustering techniques like k-means, hierarchical) are employed to obtain ‘similar’ requirements. Once ‘similar’ requirements are obtained, they are then prioritized and higher priority ones are converted to user stories and elaborated. Iterations proceed with the feedback of the stakeholder. This is different from traditional methods where requirements obtained were modelled and directly prioritized as a one-shot activity.

1.4 Requirements Elicitation techniques

Elicitation techniques fall mainly in the following categories namely traditional techniques, model driven techniques and scenario oriented techniques. The first category is useful in the early RE phase of elicitation and negotiation. Model driven techniques have been used for both early as well as late RE phases. Scenario oriented techniques have widely been used for verification and validation (V&V) RE phase. This section describes some techniques of each category.

1.4.1 Traditional elicitation techniques

These include interviews (Goguen and Linde, 1993), analyzing existing documentation which form a rich source of requirements (Sutcliffe, 2002), questionnaires, group elicitation techniques (Leffingwell and Widrig, 2000), brainstorming, workshops, prototyping, contextual, cognitive and ethnographic studies.

These techniques have certain disadvantages. The interaction between the requirements engineer and the stakeholder can be based on assumptions (Goguen and Linde, 1993) with the possibility of ambiguity in understanding the question (Suchman and Jordan, 1990), a tendency to over-analyze leading to the system to be too constrained (Hickey and Davis, 2003) and improper sampling leading to bias (Goguen and Linde, 1993) among others.

The disadvantages seen with traditional techniques are noticed particularly when one moves to systems with increasing complexity (Lapouchnian, 2005). This led to the development of model driven techniques.

1.4.2 Model driven RE

Modelling requirements has today become a core process in requirements elicitation. Generally, the system and possible alternate configurations of the system are modelled. These techniques shift the focus from "what" feature of the system to "why" of the system (Anton 1996). While the former focuses on activities of the system, the latter focuses on the rationale for setting the system up. There are two techniques goals and agent oriented modelling both of which are interrelated.

a) Goal oriented Requirements Engineering (GORE)

"Goal-oriented requirements engineering (GORE) is concerned with the use of goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting, and modifying requirements"(Lamsweerde, 2004). This indicates that goals can be used in almost every activity of the requirements process. Goals have been looked upon in a number of ways some of which are described below:

i. (Dardenne *et al.*, 1993) states that goals are high-level objectives of the business, organization or system; they capture the reasons why a system is needed and guide decisions at various levels within the enterprise.

ii. According to (Anton, 1996), “Goals are targets for achievement which provide a framework for the desired system. Goals are high level objectives of the business, organization, or system”

iii. According to (Lamsweerde, 2000), "Goal is an objective the composite system should meet.”

iv. (Pohl, 2010) observes that goals are prescriptive statements as against descriptive statements in that they state what is expected from the system and not statements describing the domain of the system.

Goals have been used in RE for eliciting functional requirements (Dardenne *et al.*, 1993) as well as non-functional requirements ((Mylopoulos *et al.*, 1999). Hard goals can be by the system and used for modelling and analysing FRs (Sutcliffe, 2002). Satisfaction and information goals are examples of hard goals. Softgoals are goals that do not have a clear-cut criterion for their satisfaction (Mylopoulos *et al.*, 1999) and are “satisficed”. They are used to model and analyse NFRs.

Goal models are motivated by the AND/OR graphs of problem solving in Artificial Intelligence where a given problem is decomposed into sets of sub-problems. The AND decomposition indicates ALL sub-problems must be tackled to solve the problem while OR decomposition states that any one set of sub-problems can be tackled to solve the problem.

In GORE, Goals are decomposed into sub-goals. This refinement is applied recursively till lowest level goals are reached. Thus, one obtains directed acyclic graphs with the nodes of the graphs representing goals (Haumer *et al.*, 1998) and achievement as edges. An AND association means that all the sub-goals, g1,....,gn, must be satisfied to satisfy the parent goal,

g. An OR association means that satisfying at least one sub-goal, g1,...gn, is enough to satisfy the parent goal, g.

The remaining question is of how the sub-goal is obtained. Means-Ends analysis, which is a reduction technique, is applied to arrive at the graph. Notice that each level of the goal hierarchy describes the same system. However, the level of abstraction differs with the root level goal at the highest level of abstraction and the leaf level goal at the lowest level. The lowest level leaf goal can be operationalized.

Enhancements to this basic goal model have been made. Apart from the AND/OR link, a third link, “conflict”, was also added to the refinement tree to capture the case when satisfying one goal causes another goal to not be satisfied. Further links were introduced to reflect the fact that goals positively or negatively “support” other goals (Lamsweerde, 2001). Pre-conditions, post-conditions and trigger conditions were added by (Lamsweerde, 2000). Link between goals and operations was also introduced by (Dardenne *et al.*, 1993).

Before one can start goal modelling, goals need to be identified. One source of goals is by reverse engineering from current systems and documents like ER diagrams, flowcharts. Another source is by forward engineering from stakeholder interviews. Stakeholders own goals, though, requirements are expressed by them not in terms of goals but as actions and operations (Anton, 1996). Goals can be extracted from actions by selecting appropriate “action words”.

GORE became a popular method of modelling requirements. A number of Goal Oriented Requirements Engineering have been proposed (Castro *et al.*, 2002; Kavakli and

Loucopoulos, 1998; Antón, 1996; Bubenko *et al.*,1994; Dardenne *et al.*,1993). KAOS and GBRAM techniques are briefly described below:

- KAOS (Dardenne, 1993) defines a formal specification language for goal specification consisting of objects, operations, agent, and goal. Objects can be entities, relationships or events. The elicitation process is in two parts. Initially, a set of system goals and objects and an initial set of agents and actions are defined. In the second part refining goals using AND/OR decomposition, identifying obstacles to goals, operationalizing goals into constraints and refining and formalizing definitions of objects and actions is done iteratively. Goal refinement ends when every sub-goal is realizable by some agent.
- Goal-Based Requirements Analysis Method, GBRAM, (Anton, 1996) identifies, elaborates, and refines the goals as requirements. It deals with two issues. How can goals be identified and what happens to requirements when goals change? The first part of the question is answered by Goal Analysis and the second part by Goal Evolution. In the former goals, stakeholders, actors and constraints are identified. This gives a preliminary set of goals. Once validated by the stakeholders, this initial set can be refined.

Recently though, certain difficulties in using GORE have been pointed out. It has been observed by (Antón and Potts, 1998) that identifying goals of the system is not the easiest task. Further, GORE is subjective, dependent on the requirements engineer view of the real world from where goals are identified (Haumer *et al.*, 1998). (Horkoff and Yu, 2010) points out that such models are "informal and incomplete" and "difficult to precisely define". (Horkoff and Yu, 2012) observe "goal modeling is not yet widely used in practice"

(Matulevičius and Heymans, 2007) notices that constructs used in KAOS are not used in practice.

b) Agent Oriented Requirements Engineering

"Agents" have been defined in software engineering as objects that can change state and behaviour. They can be humans, machines or be of any other type.

Let us look at the relationship between an agent and goals. In GORE, agents are not central concepts like goals are and do not have a say in the RE process. They are simply assigned to goal and fulfil goals. So during goal modelling, goals are identified and then during operationalization, agents are allocated to goals.

Agent oriented methods treat agents as first class concepts. The central concept is that "goals belong to agents". This is in contrast to GORE where "agents fulfil goals". Notice that even though it is possible to have goals without agents (Eric, 1997) and agents without goals as in their NFR framework, goals and agents complement each other.

Agents have the following properties (Castro *et al.* 2002; Yu, 1997; Lapouchnian, 2005):

i. Agents are intentional in that they have properties like goals, beliefs, abilities etc. associated with them. These goals are local to the agent. It is important to note that there is no global intention that is captured.
ii. Agents have autonomy. However they can influence and constrain one another. This means that they are related of each other at the intentional level.
iii. Agents are in a strategic relationship with each other. They are dependent on each other and are also vulnerable w.r.t. other agents' behaviour.

Agents help in defining the rationale and intensions of building the system. This enables them to ask and answer the “why” question. Agent oriented RE focuses on early RE. Some techniques include (Neto and Morais, 2013; Yu and Mylopoulos, 1994). (Yu and Mylopoulos, 1994) developed the i* framework for modelling and reasoning about the organizational environment and its information system. The central concept of i* is that of the *intentional actor*. This model has two main concepts, the *Strategic Dependency* model (SDM) and the Strategic *Rationale* model (SRM). The SDM component of the model describes actors in their organizational environments and captures the intentional dependencies between them. The freedom and the constraints of the actors are shown in terms of different dependencies like goal, task, soft goal and resource dependencies. SRM is at a much lower level of abstraction than SDM. It captures the intentional relationships that are internal and inside actors. Intentional properties are modelled as external dependencies, using means-ends relationships as well as task decomposition. Means-ends relationship helps us understand “why an actor would engage in some task”. This can also assist in the discovery of new soft goals and therefore provide more alternate solutions. During modelling one can travel from means to ends or vice versa. Task decomposition results in hierarchy of intentional elements part of a routine.

Using ontological studies (Matulevičius and Heymans, 2007) found similarities between i* and KAOS. According to them, constructs like i* goal and soft goal of KAOS, and means-end link of i* and contribution relation of KAOS are conceptually the same. Notice, KAOS does model agents as having wishes and they do participate in the RE process.

1.4.3 Scenario Oriented Requirements Engineering

Scenarios have been used for requirements engineering (Sutcliffe *et al.*, 1998) particularly for elicitation, refining and validating requirements, that is, in the late RE phase. Scenarios have also been used to support goals formulated in the early requirements phase. They show whether the system satisfies (fulfilment) or does not satisfy (non-fulfilment) a goal. In other words, scenarios ‘concretise’ goals.

(Holbrook, 1990) states that "Scenarios can be thought of as stories that illustrate how a perceived system will satisfy a user's needs". This indicates that scenarios describe the system from the viewpoint of the user. They have a temporal component as seen in the definition given by (Lamsveerde and Willemet, 1998): “a scenario is a temporal sequence of interaction events between the software-to-be and its environment in the restricted context of achieving some implicit purpose(s)”. Scenarios have also been defined with respect to agents. (Plihon *et al.*, 1998) says that scenario is "…possible behaviours limited to a subset of purposeful…communications taking place among two or several agents".

Scenario based approaches describe requirements and system behaviour through examples, scenes, narrative descriptions of contexts, use cases and illustrations of agent behaviours. Scenarios represent typical interactions between the system To-Be and its stakeholders. An interaction is a series of message-response pairs: a message is sent by the stakeholder to the system To-Be and the response is the reaction of the system to the message. The entire interaction specifies a typical functional requirement.

A meta schema was proposed by (Sutcliffe *et al.*, 1998) looks at the relationship between goals, scenarios and agents (see Fig. 1.3 below). Scenarios are a single instance of a use case.

Use cases are composed of actions that help in fulfilment of goals. One use case fulfils one goal. A single action 'involves' one or more agents.

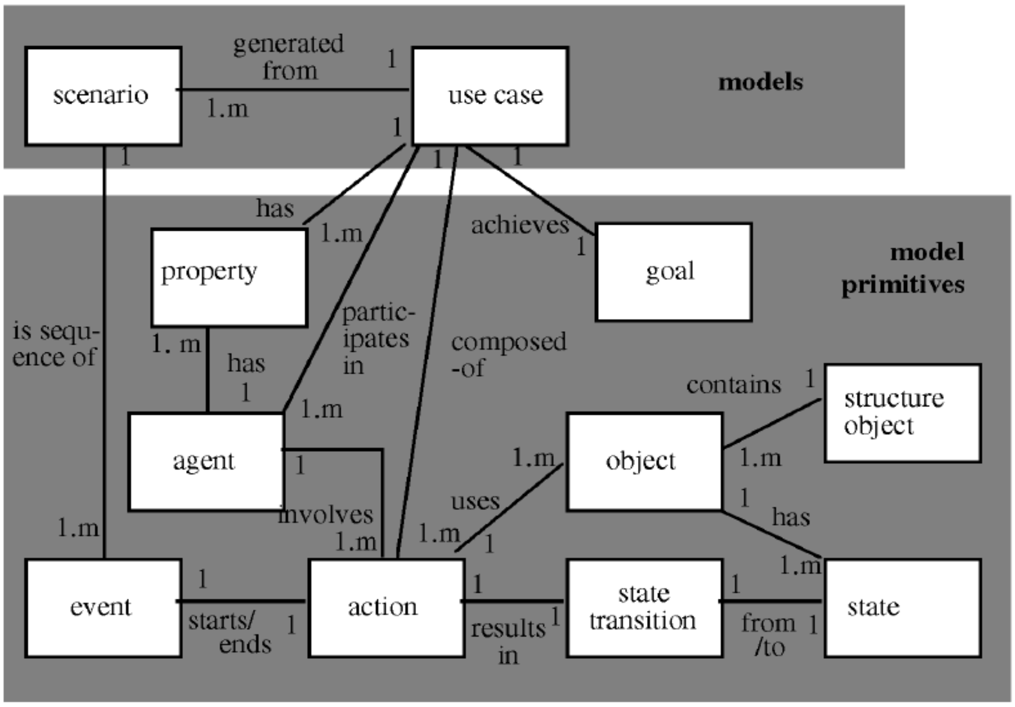


Fig. 1.3.: Meta schema as proposed by Sutcliffe *et al.*, 1998

Scenarios can be expressed using natural languages (Haumer, 1998), tables (Potts *et al.*,1994), scenario scripts (Ruben and Goldberg, 1992), regular grammars (Glinz,1995) or state-charts (Harel,1987). Scenarios can be descriptive (Potts *et al.*,1994), explanatory (Wright,1992) or exploratory (Holbrook,1990). In addition to this they may also contain behavioural, organisational and stakeholder information.

Several scenario based elicitation techniques exist two of which are SBRE (Holbrook, 1990) CREWS (Sutcliffe *et al.*, 1998).

- SBRE (Holbrook, 1990): There are two worlds, users' and designers' world. The goal set is defined in the user's world. It contains information regarding goals and constraints of the system. The goals are represented as sub-goals. The design set is in the designer's world. This set consists of design models that represent the system. The goal set and the design set communicate with each other with the help of scenarios that is in the scenario set. This set shows how a specific design meets a goal. Scenarios have a one to one relationship with the design models. A specific scenario may satisfy many goals. Any issue that may arise is captured in the issue set. A feedback cycle captures the user’s response to Issue and Design. Scenarios form part of the specification of the required system.

- CREWS (Sutcliffe *et al*, 1998): This technique is integrated with OO development and employs use cases to model functionality of the system-to-be. Here, scenario is represented by one instance of an event which is defined by a pathway of a use case. Thus, many scenarios can be generated from one use case and one scenario is composed of one or more events. Use cases are elicited from users and formatting guidelines. The use cases are compared with generic requirements and finally normal and exception flows are modeled. From the former normal scenarios and from the latter exception scenarios are generated. They are validated using validation frames. Scenarios originated from system design and those captured from actual experience are captured by this technique.

Proposals for Goal-Scenario coupling also exist in literature (Liu and Yu, 2004; CREWS, 1998; Pohl and Haumer, 1997; Cockburn, 1997). This can be unidirectional from goals to scenarios or bidirectional coupling of goals and scenarios. Unidirectional coupling says that goals are realized by scenarios and this reveals how goals can be achieved. Bi-directional coupling considers going from scenario to goals in addition to going from goals to scenarios. It says that scenarios can be sources of sub-goals of the goal for which the scenario is written.

It is clear that RE for transactional systems (TRE) is well established. The work in this thesis is focussed in the area of RE for Data Warehouse systems. Thus, it is first important to understand the influence that the above mentioned methods have had on RE for DW systems. Subsequent sections examine the existing techniques to arrive at the problem statement of the thesis.

1.5 Data Warehouse Systems

In order to understand DWRE, let us first review the definition of DWs.

Definition 1: (Inmon, 2005) defines Data Warehouse (DW) as "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's **decision-making process**".

Let us examine each property of a DW mentioned in this definition:

Subject- Oriented: Functional systems are application driven. For example, for a university system applications developed for a functional system are: library system, admission system, HR system etc. DW subjects for this university system are students, faculty, placement etc.

Integrated: Data of a DW comes from various sources like databases, flat files etc. The formats in which data is stored in these different sources may therefore differ. For example, in one source say gender is represented as male and female. In another source 1 may represent male and 0 female. Similarly, there can be differences in units of say height, weight etc between the different sources. There are often fields that essentially contain the same data but named differently like description or remarks or comments. Thus, there is a need to convert, reformat, summarize data before integration so that after integration data has one single corporate image.

Time-variant: Time variant property states that at data in the DW represents information at some moment in time but this data varies from moment to moment. Therefore, data is time stamped to know moment for which it is valid.

Non-Volatile: DW data is read-only and no update in place is allowed. As a result, it is possible to have current data as well as data at earlier moments. That is, a DW collects historical information. Since a DW is non-volatile, it is refreshed periodically, and at refresh time, a fresh snapshot of the data is loaded.

Definition 2: (Dayal *et al.*, 2009) says that a DW "…consolidates data from several operational databases, and serves a variety of front-end querying, reporting and analytic tools".

The foregoing definitions show that DW systems are different from traditional functional systems. A functional system is for transactional activity. Data Warehouses on the other hand are set up to cater to the decisional needs of decision makers or knowledge makers. According to (Gam and Salinesi, 2006) goals of a traditional functional system are different from those of a DW. The former "implement" business rules. The users expect the system to support achievement of business processes. The users' of a DW are decision makers'. They use the DW to "monitor" business processes.

There are also differences in the way data is gathered in the two kinds of systems (Gam and Salinesi, 2006). For, a functional system, data comes from transactional activity and this data follows CRUD life cycle. DW data is collected from various data sources and data here is preserved as historical data.

1.5.1 Data Warehouse Development Strategies

There are two broad DW development strategies, those of Inmon (Inmon, 2005) and Kimball (Kimball, 1996). In Kimball's approach individual data marts, DM, are built independently with each data mart representing a subject specific view. Thus, a typical DW system consists of a collection of multiple data marts connected together.

The two development strategies are shown in Fig. 1.4. Consider the top half of the figure first. Using Kimball's approach, data specific to a data mart is extracted from OLTP by the ETL process. This is then loaded to form the specific data mart, data mart1. Again, data specific to the next data mart, data mart2, is extracted and ETL performed. By following this process individual data marts are built.

However, as the number of DMs increase, the problem of inconsistency between some DMs is noticed. Such DMs are integrated, which typically involves identifying conformed dimensions and facts. It is interesting to note that integration is done even as other DMs are developed independently. If inconsistency arises again, between the newly built DMs and an integrated DW, integration is performed again.

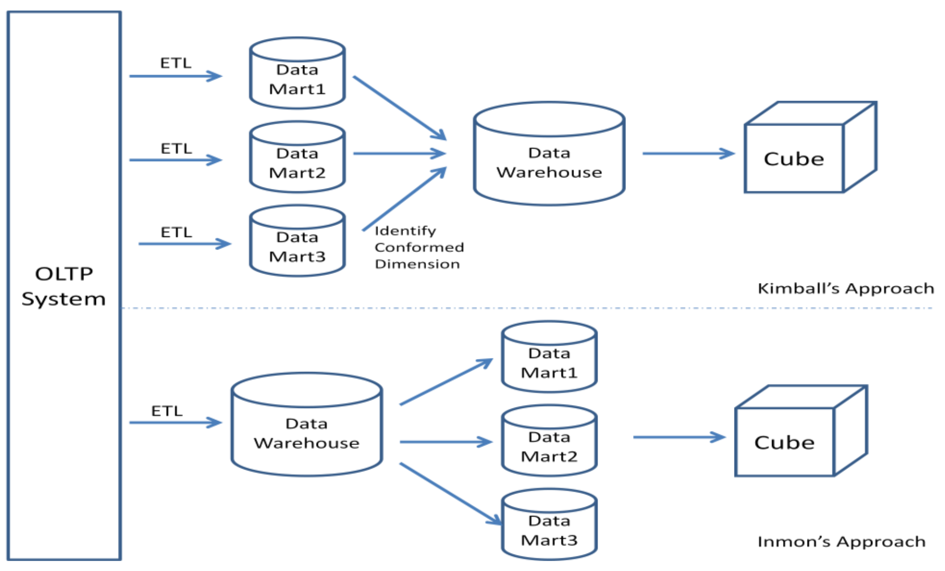


Fig. 1.4: Difference between Kimball's and Inmon's DW development strategy

This process is conceptualised in Fig. 1.5 by the independently developed data marts DM1, DM2 to DMn. When the inconsistency problem arises, then the data marts are integrated (the edges in Fig. 1.5) to develop DW1 even as other data marts DMp to DMr are developed. Now, the inconsistency problem arises again leading to yet another integration. The underlying principle is that of inconsistency **detection and correction**. The worst case situation is when the very first two data marts display invalid/inconsistent results.

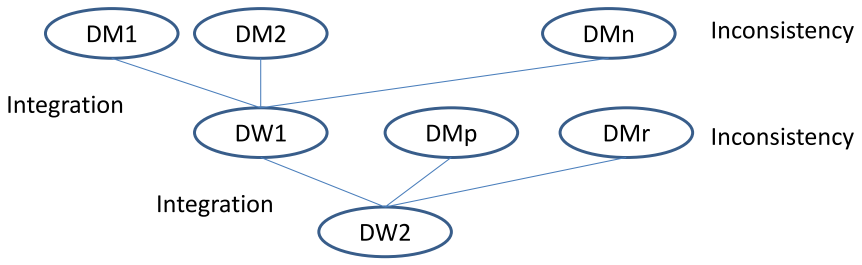


Fig. 1.5: Integration of individual data marts

Now consider Inmon's development strategy shown in the bottom half of Fig 1.4. In Inmon's approach a full DW system is built and DMs are derived from it. Thus all the data that is required is Extracted from the OLTP systems, Transformed and Loaded to make the Data Warehouse. This is similar to the schema-subschema relationship found in data base technology, data marts are analogous to subschema whereas the Data Warehouse is the schema.

1.5.2 Operational Nature of Decision making

Underlying the DW movement is the assumption that the DW is needed for taking decisions for running an organization. These decisions can therefore be said to be operational in nature, or that, a Data Warehouse supports operational decision making.

Consider the second generation of Data Warehouse systems. In Inmon's DW 2.0 (Inmon *et al.*, 2010) architecture, three components are addressed, Data architecture, Infrastructure and Unstructured data. Data is divided into four sectors: Interactive sector with 'very current' transactional data; Integrated sector with current data on hourly, daily or real time basis; Near line sector with historical data; Archival sector with data older than near line data. Unstructured Data makes it possible to store text, images and other such forms of data. However, notice in terms of decision making, it is still of the operational kind.

In Imhoff and White's DSS 2.0 (Imhoff and White, 2008), there is compartmentalization of operational, business analytics and content analytics in separate modules. The authors recognize multiple levels of decision makers for long term goals. In terms of decision making, the nature continues to be operational.

1.6 Data Warehouse Failure Statistics and need for RE

Notice there is no mention of RE while building DW systems. This was indeed the case with Inmon (Inmon, 1996) observing that, in fact, requirements are the last thing to be discovered while building DW systems. However, this has undergone a change. Whereas from a theoretical point of view, researchers extended the gains made in transactional systems development to Data Warehouse systems development, results from real Data Warehouse projects were also coming in. Data Warehouse failure statistics highlighted the crucial role of RE in mitigating system failure (Alshboul, 2012). Hayen (Hayen *et al.*, 2007) refers to studies that indicate the typical cost of a Data Warehouse project to be one million dollars in the very first year. However, these projects are very risky: one-half to two-third of most Data Warehouse projects fail. One of the causes of this failure (Alshboul, 2012) is inadequate determination of the relationship of the DW with strategic business requirements. These statistics reinforce the need for RE for decisional systems.

1.7 Influence of Transactional RE on Data Warehouse RE

There is enough evidence of the influence of transactional RE (TRE) on Data Warehouse RE (DWRE). Concepts like goals, scenarios and goal-scenario coupling have been adapted to fit in the DW context. While TRE is for transactional activity, DWRE is used to support decision making. Thus, in TRE, the focus is on discovering system functionality and associated non-functional requirements, NFRs, in the case of DWRE, information to support decision making needs to be discovered.

While comparing goals of TRE with that of DWRE approaches, one finds the introduction of some additional concepts. This was in order to make goal models elicit information rather than functionality of transactional systems. Thus, along with AND/OR goal hierarchy of

TRE, additional concepts like decision and information nodes along with goal-decision edges and decision-information edges was developed in (Prakash and Gosain 2003; Prakash and Gosain, 2008). Similarly, information requirement, goal-business process link among others was seen in (Mazón *et al.*, 2007). There was also a movement to develop a view of goals which was different from TRE goal. In this regard, goals were viewed as describing nature/purpose of a service (Boehnlein and Ulbrich, 1999), as a quality measure (Bonifati *et al.*, 2001) among others. The prominence of GORE in DWRE can be clearly seen by the number of goal oriented DWRE techniques.

Bidirectional Goal-Scenario coupling has been adapted to Decision-Information scenario coupling by Prakash and Gosain (Prakash and Gosain, 2008). When comparing scenario of TRE and information scenario here, one finds that while the former specifies goal achievement, the latter specifies information. Further, a scenario illustrates main task and its variations and exceptions while an information scenario reveals information of main decisions and variations of decisions. As for the differences between Goal-Scenario coupling and Decision-Information scenario, the former helps discover new goals while the latter helps discover new decisions. Other than the above mentioned technique of (Prakash and Gosain, 2008), other DWRE approaches that use goal-scenario coupling have not been found.

1.8 Data Warehouse Requirements Engineering Techniques

DWRE techniques_identify DW structures like facts, dimensions and finally arrive at star schema. DW structures are identified from existing systems, information gathered from users' of the DW or a combination of the two. Based on this, DW RE techniques are classified into two broad categories: Supply driven and Demand driven.

a) **Supply driven techniques**: the basic approach is bottom-up where existing transactional systems are used to arrive at DW structure (Winter and Strauch, 2004). Two such techniques are reviewed here, (i) that starts from existing databases and (ii) that start from ER schema of operational systems.

(i) **Database Driven Approach** (Golfarelli *et al*., 1998; Golfarelli and Rizzi, 1999): This approach starts off with analysis of existing database systems, which is followed by determination of requirements of the Data Warehouse. The DW designer in collaboration with the managers of information systems collect documentation of the existing system and produce database scheme of either the entire information system or the part that is of interest. The database scheme created is used by the DW designers and the end users of the DW to produce a choice of facts and the preliminary workload. The authors propose rules to help in the selection of facts. The preliminary workload helps in the identification of dimensions and measures. Taking the facts and the preliminary workload defined, the dimensional schema is produced which consists of fact schemes. (Prakash *et al.*, 2009) states that since, information contents of DW are defined by data present in the operational data sources, information is limited to existing databases. Identification of external sources and other internal sources is difficult and thus information related to them is not captured.

(ii) **ER schema driven approach**: DW structure is also obtained by starting with ER schema. This approach produces measures and dimensional attributes of the Data Warehouse after starting from ER schemas of databases. Described below are techniques by Hüsemann (Hüsemann *et al*., 2000) and by Moody (Moody and Kortink, 2000).

- (Hüsemann *et al*., 2000): The process of arriving at DW structures starts by the DW designer, in collaboration with the managers of information systems, collecting documentation of the existing system. From this database scheme of either the entire information system or the part that is of interest is created. In the next step, the database scheme created is used by the DW designers and the end users of the DW to produce a choice of facts and the preliminary workload. Rules to help in the selection of facts are proposed by the authors. The preliminary workload helps in the identification of dimensions and measures. In the conceptual design step, using facts and preliminary workload defined in the previous step, a dimensional schema is produced. This schema consists of fact schemes obtained by applying algorithms defined for the same.

- (Moody and Kortink, 2000): This method also derives Data Warehouse structures from enterprise data models like ER schemas. They even insist on developing a model if one does not already exist. After obtaining ER schema, in the next step entities are classified as transaction, component or classification entities. Transaction entities form fact tables. Component entities form dimension tables and answer the "who", "what", "when", "where", "how" and "why" of business events. Similarly, classification entities form dimension tables. All hierarchies that exist in the data model are identified.

(Kimball, 1997) notes that "Entity relation models cannot be used for enterprise Data Warehouses". Information is limited to what has been captured by the ER diagram. In the case of Hüsemann‘s model, ER schemas do not cater to temporal information. No guidance is available on whether such information needs to be captured. There is limited guidance in identification of facts, dimensions and measures. Determining aggregates was also a manual process with no guidance available. (Moody and Kortink, 2000) offers no guidance in selecting which transaction entities are important for decision making and therefore become facts. A precedence hierarchy for resolving ambiguities that arise during classifying entities has been defined. But again no guidance in terms of an algorithm has been provided. Further, these techniques do not give primary importance to users' perspective (Giorgini *et al.*, 2005). This means that the DW designer ends up deciding on the relevance of data instead of the user (Schaefer *et al.*, 2011).

b) **Demand/Requirement driven**- These techniques start by determining information requirements of DW users and then arrive at DW structures. Some literature divides demand driven approaches into user driven and goal driven. Whereas the former elicits information requirements from business users of the system, the latter elicits them from top level management of the organization. Each of these approaches is considered below.

(i) USER DRIVEN APPROACHES

This is a bottom-up approach where from users information requirements are elicited. The focus here is on developing requirements analysis techniques and improving participation of business users (Golfarelli, 2010). In the case of user driven approach, users are involved and have a clear understanding of the system being built. Discussed below are some techniques followed by their drawbacks.

- (Paim and Castro, 2003) proposed the DWARF technique where they adapted traditional requirements elicitation techniques like interviews, workshops, prototyping to elicit requirements from users' perspective. They designed DW functionalities (ETL) using the use case model. They developed EW-ENF for non-functional DW requirements like indexing, schema loading etc.

- (Winter and Strauch, 2003) proposes a cyclic process which maps the information demand made by middle level managers and knowledge workers with information supplied in operational databases, reports etc. They have an ‘initial’ phase, an ‘as is’ phase and a ‘to be’ phase. In the first phase they argue that since different users can result in different data models, the dominant users must be identified. This helps target a specific business process. In the ‘as is’ phase, an information map is created by analyzing (a) existing information systems and (b) reports that the users commonly use. According to the authors, analyzing the latter helps identify more sources of information that one is not commonly aware of. In the ‘to be’ phase, information demand is elicited from the user by asking business questions. The information supply and information demand is compared and inconsistencies analyzed. Finally, using semantic models information requirements are modelled.

- (Bruckner *et al.*, 2001) DW requirements are defined at different abstraction levels by the authors: Use Cases are used for communication between stakeholders, domain experts and DW designers. Use cases provide the input at the abstraction level ‘detailed system requirement’. The authors propose an incremental method to develop use cases. Facade iteration is the first iteration where outline and high level descriptions are captured. Its purpose is to identify actors, create placeholders for other major iterations. The information gathered is minimal as it captures names and short descriptions of actor interactions with DW system. The next iteration is Filled iteration where ideas of use cases, generated during the Facade iteration, are broadened and deepened. They generally include ‘functional’, information requirements plus requirement attributes. Since the requirements gathered can be too large, in the Analysed iteration, use cases are first individually evaluated for errors and omissions, then prioritized and pruned. This is done so that at the end only the use cases that provide sufficient information to build DW system are left. The next iteration is Optimized iteration. Here conflicting/inconsistent use cases are identified and reassessed. Unlike the previous iteration where each use case is considered individually, here all the use cases are considered together to understand their effect on each other. The last iteration is Finished which includes touching up and fine pruning use cases so that they are complete and can be used for design of the DW system.

- (Prakash and Bhardwaj, 2012) The authors divide their requirements engineering task into an early information part and a late information part. In the former, unstructured information for decision making is determined and in the latter unstructured early information is converted into more structured form like facts, dimensions.

Early information phase starts with the notion of a target. A target is defined as <aspect, indicator> where aspect represents quality, work area, work unit and indicator is a "metric whose value identifies 'what is to be achieved'". There are two reductions defined by the authors (a) aspect driven where aspect of the top level target is reduced into sub aspects and sub indicators are determined to form sub-targets; (b) indicator driven where indicators of top level targets are reduced into sub indicators, their aspects determined and sub-targets formed. A target hierarchy is thus formed by the two reduction processes.

For targets to be achieved, choice sets are associated with targets (see Fig 1.6). This is where decision making is done. The authors define choice set as "a pair <CSO, O> where CSO is the choice set objective and O is the set of alternatives for meeting the CSO". Relationship 'fulfils' captures the relationship between target and choice set. Similar to aspect and indicator reduction, CSO and O can also be reduced to build hierarchies. These hierarchies let us know "how it is to be achieved".

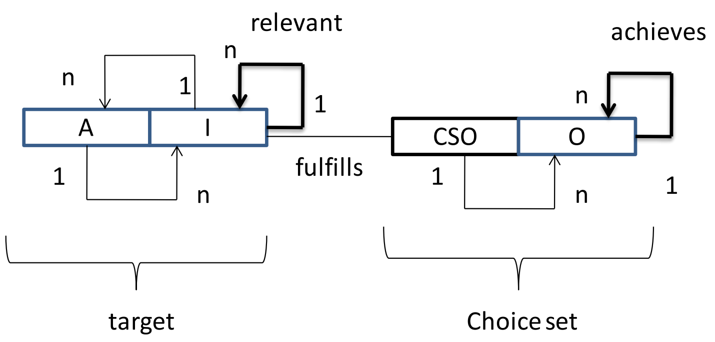


Fig. 1.6: Meta model for target with a choice set (Prakash and Bhardwaj, 2012)

In order to select an alternative from the choice set, pertinent information is required. This is elicited using the techniques of (Prakash *et al.*, 2009). Their entire RE process therefore has the following steps: Determine top level aspects, business indicators for each aspect, formulate top level targets, decompose targets, identify CSO, build achieves hierarchy, identify information. In order to perform these steps, brainstorming sessions, management reports can be used.

Summary of User driven approaches

Broadly, user driven techniques suffer from the following disadvantages:

- Users' requirements are difficult to elicit (Golfarelli, 2010). Some may not be able to describe their requirements (Boehnlein and Ulbrich, 2000)
- Requirements of users' keep changing through the project (List *et al.*,2002; Golfarelli, 2010)
- Users' do not see organization from a "broad angle" and so the requirements are "narrow" (List *et al.*,2002)
- There is lack of traceability between requirements model and final DW model (Mazón *et al.*, 2007)

(ii) GOAL DRIVEN APPROACHES

Goal driven techniques follow a top-down approach where top level managers define their goals and further refinement of these goals is done through goal decomposition techniques. Unlike user driven approach, users are not involved the process of system design. Some techniques of this approach have been described below followed by their drawbacks.

- (Gam and Salinesi, 2006) CADWA is a goal based approach that identifies 'early requirements', refines requirements and concretizes them into DW structures. There are three stages to this process a) eliciting requirements b) designing DW fragment model and c) integrating DW fragment models. Here, requirements are captured by '**anticipating'** decision makers' requests. This is done by studying overall business objectives and opportunities, decision makers' macro and micro business plans. The Action plans that are obtained are finally converted into DW models.

- (Prakash and Gosain, 2008) The requirements engineering process discovers the decisions of interest as well as the relevant information. It is assumed that these decisions are themselves based on a discovery of organizational goals. The set of goals, decisions, and information revealed by the requirements engineering process are all organized in a Goal – Decision – Information (GDI) schema (see Fig 1.7).

The authors view a goal "as an aim or objective that is to be met". A goal is non-operational, passive which means that it cannot perform or cause any action. A decision is "a specification of an active component that causes goal fulfilment". Similar to a goal, a decision can be simple or complex. Whereas the former cannot be decomposed into simpler ones, the latter consists of simple or complex decisions. Relationship between goals and decisions is through the association '*is influenced by'*. Information is required for decision making and this is captured by '*is required for'* association.

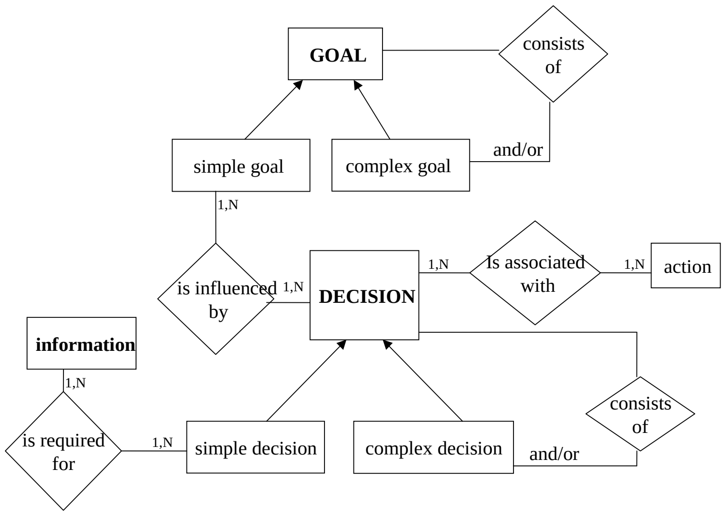


Fig. 1.7: GDI schema showing Goal-Decision and Information (Prakash and Gosain, 2008)

The GDI schema forms the basis of development of the ER diagram from which the properties of the star schemas to be developed are determined. The complete set of properties including aggregates and history are postulated. The design includes information additional to mere facts and dimensions by including aggregates and history.

- (Mazón *et al.*, 2007) The authors propose a GORE technique for DWRE based on i* methodology. They relate goals supported by DW with information requirements. Facts and dimensions are discovered from information requirements. Their approach is integrated with *model driven development* (MDA) which defines a computation independent model (CIM) and platform independent model (PIM) at the conceptual level. CIM models goals and information requirements and PIM models the multi-dimensional structures. *Query-View-Transformation* (QVT) is used for transformation between these models.

CIM is specified by the i* framework which is modified (see Fig 1.8) as follows. An intentional actor refers to a decision maker involved in decisions making process. For each intentional actor, intentional elements are *goals*, *tasks* and *resources* and intentional relationships are *means-ends*, *decomposition* link. To the resources stereotype, concepts of *Business Process*, *Measure* and *Context* have been added. *Goals* can be of three kinds, *strategic*: highest level of abstraction, *decision*: next level of abstraction, *information*: lowest level of abstraction. This hierarchy is used for discovering goals using refinement.

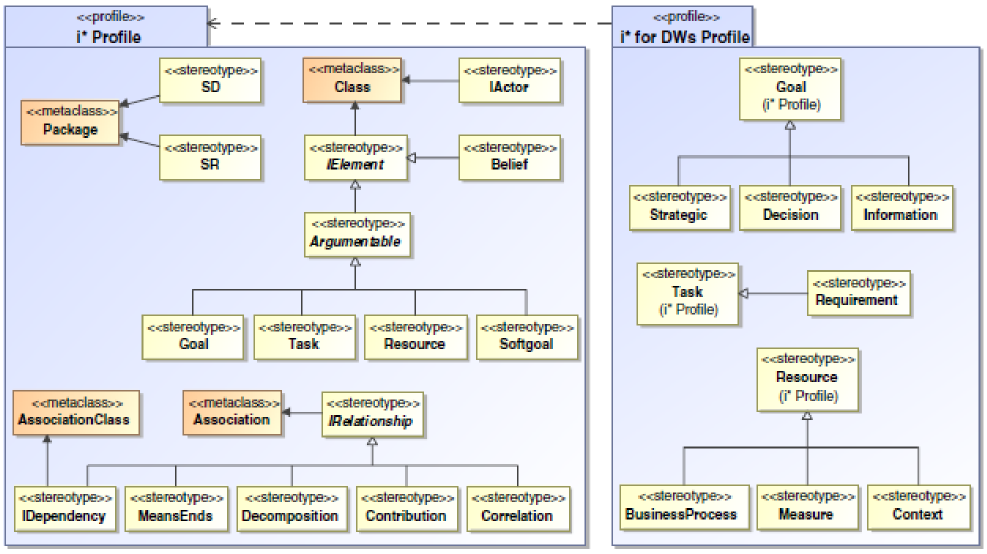
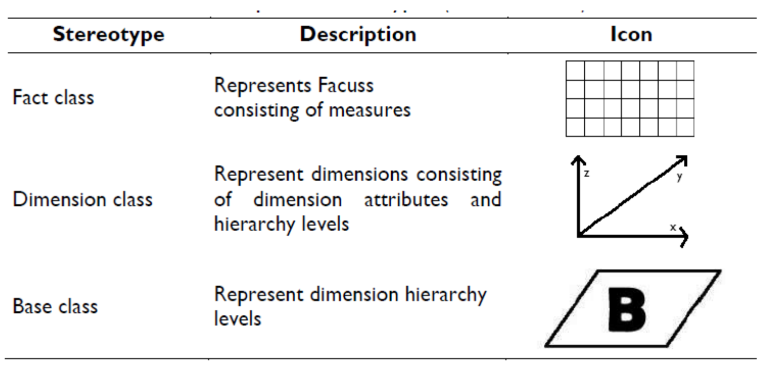

<<profile>>
i* Profile
<<profile>>
i* for DWs Profile
<<stereotype>> SD
<<metaclass>> Class
<<stereotype>> IActor
<<metaclass>> Package
<<stereotype>> SR
<<stereotype>> IElement
<<stereotype>> Belief
<<stereotype>> Argumentable
<<stereotype>> Goal
<<stereotype>> Task
<<stereotype>> Resource
<<stereotype>> Softgoal
<<metaclass>> AssociationClass
<<metaclass>> Association
<<stereotype>> IRelationship
<<stereotype>> IDependency
<<stereotype>> MeansEnds
<<stereotype>> Decomposition
<<stereotype>> Contribution
<<stereotype>> Correlation
<<stereotype>> Goal (i* Profile)
<<stereotype>> Strategic
<<stereotype>> Decision
<<stereotype>> Information
<<stereotype>> Task (i* Profile)
<<stereotype>> Requirement
<<stereotype>> Resource (i* Profile)
<<stereotype>> BusinessProcess
<<stereotype>> Measure
<<stereotype>> Context

Fig. 1.8: Modified i* framework for CIM (Mazón *et al.*, 2007)

The process first starts with identification of decision makers. Strategic goals are elicited for each decision. SD model is built. SR model is then built for each decision maker. Strategic goals are decomposed into decision and decision into information. Information requirements are identified from information goals. QVT transformation is applied to the CIM to obtain PIM which is the conceptual model. Fact, fact attribute, dimensions and dimension hierarchies are identified in PIM. They are represented using UML class stereotypes as shown below in Fig 1.9.



Stereotype	Description	Icon
Fact class	Represents Facuss consisting of measures	▦
Dimension class	Represent dimensions consisting of dimension attributes and hierarchy levels	z y x
Base class	Represent dimension hierarchy levels	B

Fig. 1.9: UML class stereotypes (Mazón *et al.*, 2007)

This technique has been used by (Leal *et al*, 2013) to develop a business strategy based approach.

- (Leal *et al.*, 2013) proposes arriving at DW structures by using a business oriented approach. There are four phases as seen in Fig 1.10: business strategy analysis using VMOST, aligning business strategy with DW using BMM model components, building conceptual model based on i*, arriving at MD model. The last two phases use the method described above by (Mazón *et al.*, 2007).

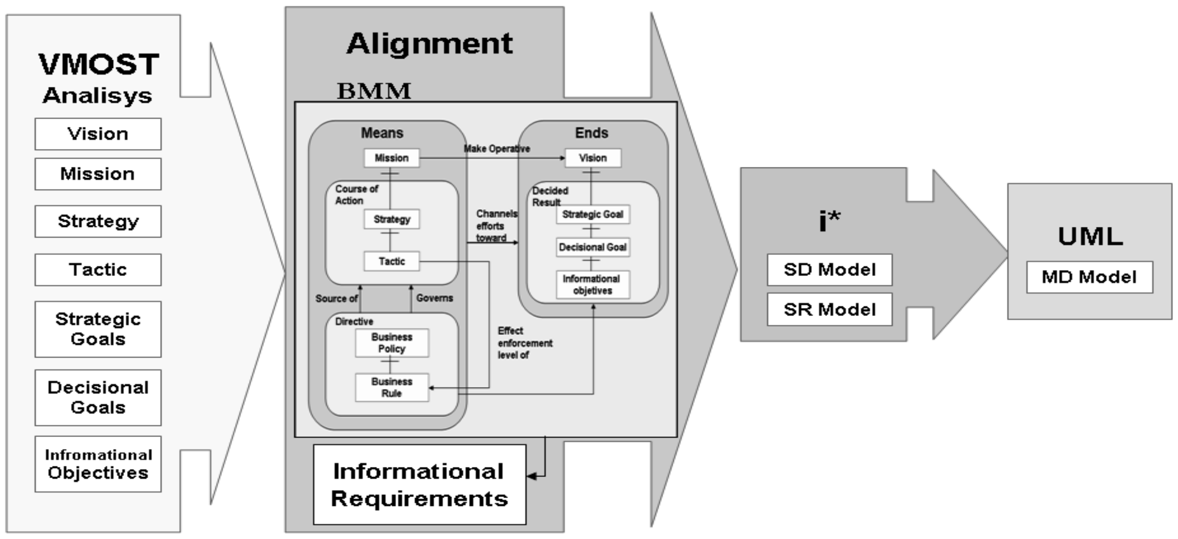


Fig. 1.10: Business Oriented approach using VMOST (Leal *et al.*, 2013)

Business Strategy is analysed by VMOST analysis. For this, first business actors (decision makers) are identified. DW is also considered as an actor. Components namely, vision, mission, strategies and strategic goals are obtained from the actors. Intentional elements like objectives, tasks, tactics and relationships represented as means-ends links are obtained next. Decisional goals are obtained for strategic goals and Information goals from decisional goals. Once the strategy elements are elicited, alignment is verified using BMM like whether strategy is a component of Mission, strategic goal is amplifying vision, mission is making vision operational etc.

- (Boehnlein, and Ulbrich, 1999; Boehnlein, and Ulbrich, 2000) This approach derives the Data Warehouse structure from business process models. There are four stages in the process. The first three are based largely on the SOM methodology while the last stage deals with identifying Data Warehouse structures. The link goal-service-business process-metric is followed to discover goals and services of the system.

In the first stage of the derivation process goals of the system are identified. These goals are then decomposed into sub-goals. The business process is then analyzed. Here a distinction is made between a main process and a service process. Decomposition rules and feedback loop are applied several times to obtain a detailed interaction schema. In the third stage business objects are identified. The dependencies between the objects, attributes are assigned to the objects. In the last stage initial Data Warehouse structures namely dimensions and measures are added to the object schema.

Summary of GORE in DWRE

The first drawback is the inherent limitation of goal orientation itself. Recall that (Horkoff and Yu, 2012) has observed that while goal modelling has been used for a number of case studies, it is not yet widely used in practice. Goal reduction is also not a straight forward process. The approach of (Boehnlein, and Ulbrich, 1999; Boehnlein, and Ulbrich, 2000), gives us no guidelines about movement from goals to business process and thereafter to Data Warehouse structures. In the goal stage, it does partly rely on goal reduction but does not adopt the decision perspective. (Prakash and Gosain, 2008) move away from goal decomposition/reduction as a way of processing goals and define goal-decision association as a way to do DWRE. However, decisions are defined from the narrow perspective of goals of the system.

c) <u>Mixed Driven</u>

Mixed driven techniques were developed to overcome the disadvantages of supply driven and data driven techniques when used separately. As the name suggests, they use a combination of these techniques. Three such techniques have been discussed below.

- (Bonifati *et al.*, 2001) They obtain DW structures from users' goals and operational databases. There are three levels of analysis done (i) top down for the users' goals (ii) bottom up for operational databases, and (iii) integrated analysis which integrates DW structures got from top down and bottom up analysis methods.

Top down analysis - Users' requirements are collected through traditional techniques like interviewing and brainstorming. Their goals and needs are expressed via the Goal-Question-Metric approach. These goals are further analyzed, decomposed into simpler sub-goals. More detailed analysis of these goals is usually needed to elicit information in order to produce the star schemas. Additional information regarding the nature of these goals is collected on *GQM Abstraction sheets. Ideal star schemas* are extracted from these abstraction sheets. The requirements captured here are functional in nature and are independent of the underlying data in the operational data sources.

Bottom up analysis – The entity relationship diagram at the conceptual level of an existing operational database is analyzed. ER schema is converted to a Connectivity graph (CG) using certain rules defined in the paper. All n-ary (n>2) and many-to-many relationships are converted to one-to-many relationships. Entities having additive attributes are treated as a *potential fact entity*. They form the centre node and dimensions reachable by one-to-many or one-to-one relationships are all possible dimensions for this fact. Thus, from this graph one obtains a snowflake graph. *Candidate star schemas* are derived from the snowflake graphs.

Integration analysis – The star schemas got from the previous two steps namely the ideal and the candidate star schemas are matched. A metrics for selection is applied and the star schemas are ranked. The designer then chooses the best fit for system design.

- (Giorgini *et al.*, 2005) The GRAND approach is an extension of the Tropos methodology (Bresciani *et al.*, 2004). It divides the requirements elicitation process into two perspectives, the organizational perspective and the decisional perspective. The former models the requirements of the stakeholders while the later is from the perspective of decision makers. Both perspectives produce models having the following two steps in common: goal analysis and fact analysis. However, the actors for both the perspectives are different, stakeholder in the case of organizational model and decision maker for decisional model.

In the goal analysis phase, goals for the actor are represented using an actor diagram. Each goal is decomposed by AND/OR decomposition and the rationale diagram built. Facts are associated with goals in the rationale diagram and added to the diagram in the fact analysis phase. The organizational model extends this diagram adding attributes to the facts. These attributes can be dimensions or measures.

The decisional model has dimension and measure analysis as additional steps. The goals of the actor, who is the decision maker involved in the decisional process, are analyzed and decomposed. There could be some goals that were not discovered in the organizational modelling that could surface here. Similar to organization modelling, facts are associated with the goals of actors. Finally a set of dimensions and measures are associated with the facts.

This technique can be used in a mixed driven (supply and demand driven) as well as demand driven framework. The supply driven part is used to identify attribute hierarchies with the help of data source schema. In the absence of source schema, this technique is used within the demand driven framework in which case attribute hierarchy identification is left for the requirements engineer to identify by interacting with domain experts. Within the mixed driven framework, decisional model facts are mapped onto entities or n-ary associations of the operational database schema. Dimensions and measures are mapped using the attributes of organizational model as a bridge. For each successful mapping, the many-to-one associations are used to generate the attribute hierarchies. Algorithm of (Golfarelli, 1998) is then used to generate the fact schema.

Summary of Mixed driven approaches

According to the approach of (Giorgini *et al*., 2005), there is a change of perspective required that views nodes of a goal hierarchy as goals in the first perspective and as decisional alternatives later. This treats all alternatives uniformly and deals with 'what is to be achieved' dimension. In the approach of (Bonifati *et al.*, 2001), the set of decisions is implied by the goal-question framework that is developed. Further, there is little guidance on what questions to ask. Yet, the metrics determined are critically dependent on the questions associated with goals.

1.9 Arriving at the Problem Statement

The discussion above brings forth the following drawbacks currently facing DWRE:

1 <u>Lack of Data Warehouse support for upstream decision-making in an organization:</u>

As mentioned earlier, DW systems have been built, extensively, to provide information for the needs of operational decision makers. Thus, information support for daily decision making is provided to all levels in an organization. IBM mentions two interesting points in (IBM, 2013). Firstly, "*Decision making happens at every level, in every function, in every region of your organization"*. Secondly, "*Every one of those decisions is based on the information people have on hand".* As already mentioned (Imhoff and White, 2008) has already addressed this issue but only for operational decision-making.

However, the policies of an organization and rules of business also need to be decided upon (BRG, 2010) for a business. Decision making in an operational business is thus done in the context of policy and rules decisions. These latter are thus **upstream** to decisions about business operations and are not supported by DW technology. Thus, there is a need to develop specific techniques for such strategic decision making.

2 Limited understanding of the Decision-Information Link :

Decisional and Information perspectives have been introduced by Giorgini (Giorgini *et al.*, 2005) and Prakash and Gosain (Prakash and Gosain, 2008) respectively. However, the nature of the link between a decision and information relevant to it, has not been studied. There is a need to explicitly model this relationship and treat both decision and information as first class concepts.

3 Limited techniques specific to Information Elicitation:

DWRE techniques are highly oriented towards arriving at information in the form of Facts, Dimensions and Measures. This is either done directly without analyzing information and without sufficiently exploring information before structuring it. Techniques like (Giorgini *et al*, 2005; Gam and Salinesi, 2006; Mazon, *et al.,* 2007) belong to the former class and techniques like (Boehnlein and Ulbrich, 1999; Bonifati *et al*., 2001; Prakash and Gosain, 2008; Prakash and Bhardwaj, 2014) belong to the latter. Even though some investigation of information was done with Information scenarios of (Prakash and Gosain, 2008) there is no guidance provided by the authors on postulating Information scenarios.

While arriving at MD structures is essential, it is equally important to elicit, examine and analyze information that is unstructured. Evidently, one needs to find ways by which information can be elicited in a guided manner.

4 Integration of Upstream and Operational DW:

Information across DW systems can be common. The solutions that exist involve identifying conformed dimensions and facts and integrate. In other words, all existing techniques talk about star schemas/data mart integration. However, this means that too much time is taken up in first arriving at the star schema and then in integrating them. Since the point of integration is downstream at the conceptual design step, it is likely that, requirements specification of the integrated system is out of step with the real system. Lastly, in terms of availability of an operational DW, a logical and physical DW is available only after the entire integration process is completed.

Thus, the problem statement of the thesis is:

To develop an RE approach for upstream or strategic DW systems and integrating, at the requirements level, strategic with operational DWs.

1.10 Proposed Solution

The solution to the above problems can be broken down into the following sub-goals:

1. Defining upstream and operational decisions: The thesis defines two broad categories of decisions: Imperative decisions and Managerial decisions. Managerial decision making deals with strategic decision making. One kind of Managerial decisions' is those that deal with formulation of norms and standards that are to be followed in organizations. These decisions are Policy Decisions. The other kind of managerial decisions are those that are concerned with the enforcement of given policies. **The decision problem here is that of defining an appropriate set of rules that the organization will follow during its operations.** These decisions are Policy Enforcement Rule (PER) Decisions**.** This leads us to

Imperative decisions. These decisions are derived from policy enforcement rules and consist of operational actions. **The imperative decision making problem is that of selecting the most appropriate action in a given situation and one which at the same time does not violate policy enforcement rules.**

2. Developing generic decision, information meta models along with information elicitation techniques: There are two kinds of decision support needed, one for policy enforcement rule formulation decisions and the other for operational decisions. In order to identify the needed information for supporting this decision making, **the thesis proposes a set of generic techniques for eliciting information that shall be stored in the DW**. In other words, these information elicitation techniques apply to both kinds of DWs. The thesis also proposes a **generic meta-model for decision and information** to define the structure of decisions and information in both DW. Further, the thesis proposes to **conceptualize the decision-information link as a decision-requirement** and proposes a meta-model for the same.

3. Providing decision Support for policy enforcement rule formulation decisions and operational decisions: The left hand side of Fig 1.11 shows the two DWs, Policy Enforcement Rule DW and operational DW. Notice that they are independent of each other. This independence can be a source of problems if there is some common information between them. This commonality leads to replication of information. This thesis shows that information replication leads to the problems of (a) Inconsistency in decision making, and (b) Loss of business control.

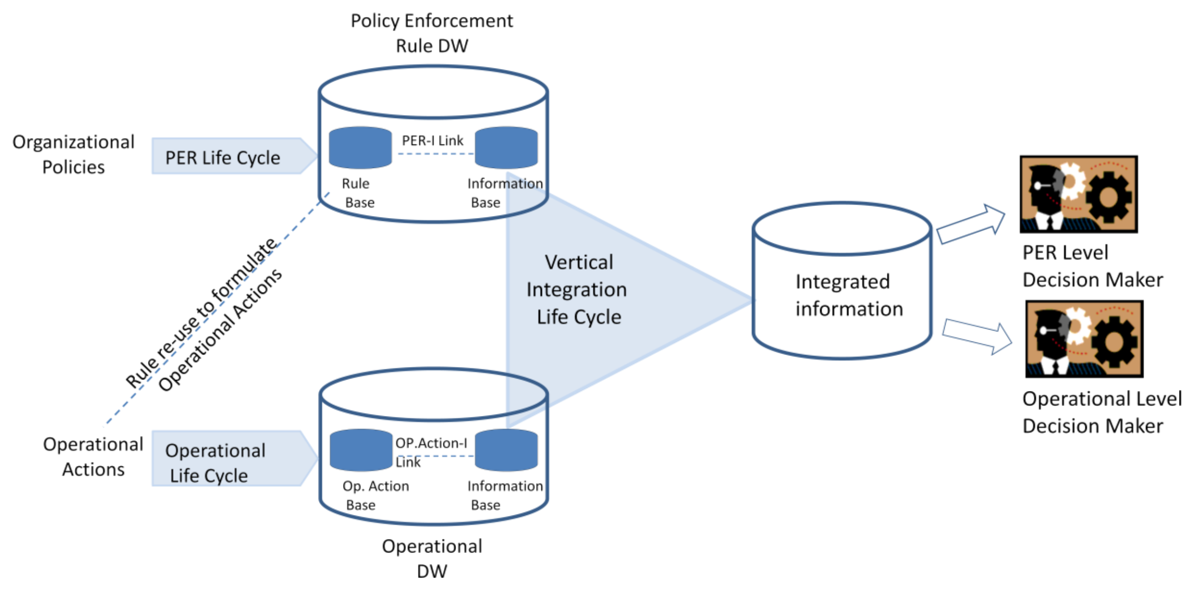


Fig. 1.11: The overall process of arriving at an integrated enterprise wide Data Warehouse

4. Integrating Data Warehouses: The thesis proposes an integration technique that integrates at the Requirements level. This is in contrast to the data mart approach discussed in section 1.5.1 in which integration is done once data marts become operational. The proposal in this thesis is that DW integration is performed the moment the set of required information is obtained for a pair of DWs. Thus, integration is moved from the star schema level, where facts and conformed dimensions are integrated together, right upstream to the requirements engineering phase. By this, effort in design and construction of DW to-be is reduced. This is shown by the broad arrow in Fig. 1.11. The figure also shows that the same integrated DW can be used by both policy enforcement rule decision makers as well as operational decision makers.

1.11 Outline of the Thesis

Chapter 2 defines typology of decisions in the 'Decision Environment' namely, Managerial set of decisions which can be Policy level or PER level decisions and Imperative decisions which are operational in nature. The generic decision requirement model, information model and decision models are presented. This is followed by describing three generic information elicitation techniques that are applicable to every level of decision making to elicit information.

Chapter 3 discusses PER life cycle to formulate PER and arrive at the Data Warehouse. The chapter proposes representing organizational policies in an extended first order logic. Thereafter, the chapter proposes guidelines that are applied to each of the four types of policies to arrive at policy enforcement rules. Also, early information is elicited using the generic techniques ENDSI, MEANSI and CSFI analysis. Finally, the chapter proposes guidelines to convert early information into ER diagrams by first generating individual and then integrating them. ER schema can then be used to identify facts etc by existing methodologies.

Chapter 4 discusses operational life cycle. The chapter discusses expressing actions of PER into operational level actions, eliciting early information elicitation using ENDSI, MEANSI, CSFI analysis techniques. Finally, the chapter discusses the conversion process of early information into ER diagrams and then into star schema by existing methodologies.

Chapter 5 presents a Vertical Integration life cycle. The chapter highlights the problems in keeping PER and Operational DW systems as separate systems. After establishing the need for integration, the chapter justifies the proposed approach to integrate upstream. Subsequently, a four step semi-automated integration approach is proposed having MetaData Reader, Correspondence Drafter, Information Mapper and finally the Conflict Resolver.

Chapter 6 illustrates the process of arriving at EI_{per}, EI_{op}, and $EI_{integrated}$ using AYUSH medical domain. The elicitation process was applied to structural policies of AYUSH. The chapter presents statistics of the study, lessons learnt and the result of applying these learnings.

Chapter 7 discusses the implementation using ELISPE, ELISO and CADEI tools to arrive at EI_{per}, EI_{op}, and. The approach is based on the models and techniques discussed in chapters 2, 3, 4 and 5. The architecture of the tools along with an explanation on the working of the different components of the tools is described. To get a feel of the tools various screen shots have also been included.

Finally, Chapter 8 summarizes the contribution of the thesis, present the conclusion and discuss future scope of the work.

Chapter 2

The Decision Requirement and Information Elicitation

This chapter considers the decisional environment and shows that there are two kinds of decisions, imperative and managerial. In order to take decisions, information is required. This association is modelled as a tuple <decision, information> and refer to it as **Decision requirement**. This chapter also develops a meta model for decision requirements and also model the notion of a decision and information from the Data Warehouse perspective. Thereafter, the next section discusses the manner in which information is elicited for given decisions.

2.1 The Decisional Environment

There is a close relationship between the information systems and Data Warehouse of an organization. The former are used to populate the latter through the ETL process. In the opposite direction, the decision taken by using the Data Warehouse has the effect of changing information system contents. This means that information systems operate in a **decisional environment**. In other words, the decisional environment provides the context in which an information system (IS) operates. This is shown in Fig. 2.1. When the information system is sent a stimulus from the decisional environment then the functionality that responds to this stimulus is invoked.

Stimuli can be sent by two different kinds of actors, IS administrators and IS operators. These stimuli correspond to two kinds of decisions, managerial and imperative. Managerial decisions are used to 'initialize' the IS where as the latter work within the initialized IS to operate the system. For example, in a railway reservation system IS administrators initialize train data whereas IS operators invoke functionality to make reservations and cancellations using information set up by the IS administrator.

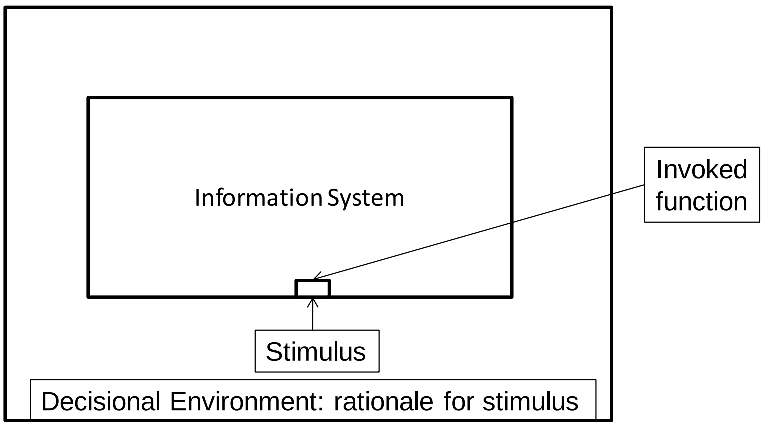


Fig. 2.1. Embedded IS in a Decisional Environment.

2.2. Nature of Decisions

Decision theory focuses on managerial decision-making and how organizations process and use information in making decisions (Mullins, 2010; Jones and George, 2008). Turban (Turban, 1998) considers decision making as the activity of manufacturing a new piece of knowledge expressing commitment to some course of action. Makarov (Makarov, 1987) formulates the decision-making problem as the pair $<\Omega, OP>$ where Ω is a set of alternatives and OP is an optimality principle. The solution to $<\Omega, OP>$ is the Ω_{OP} set selected by the optimality principle OP. According to Simon (Simon, 1977) the decision-making process consists of three phases, (a) intelligence, that involves searching for conditions that call for decisions, (b) design, which involves inventing, developing, and analysing possible courses of actions, and (c) choice, which implies the selection of a course of action from those available.

Regarding selection of alternatives, one approach is to use a single criterion. However, according to Roy (Roy, 2005), this is not sufficient when the consequences of the alternatives to be analyzed are important or in the presence of multiple viewpoints and contradictory criteria. Multicriteria analysis allows a more in-depth treatment in these situations. The goal of multicriteria DM (MCDM) methods is to define priorities among alternatives according to multiple criteria. The main five families of MCDM methods are: MAUT (Multiattribute Utility Theory) (Keeney and Raiffa, 1993), AHP (Analytic Hierarchy Process) (Saaty, 1980), outranking methods (Roy, 1996), weighting methods (Keeney, 1999), and fuzzy methods (Fuller and Carlsson, 1996).

With regard to the kinds of decisions, (Gbande and Akuhwa, 2015) state that executives take either of the two major types of decisions: programmed (structured) and non-programmed (unstructured) decisions. McLeod and Schell (McLeod and Schell, 2001) classify decisions along three dimensions, namely, the level of the organization at which a decision is taken, the structuredness of decisions, and whether or not negotiation is required. Thus, they propose the following types of decisions

- Management Level: Strategic, management control, operational control
- Structuredness: Structured, Semistructured, Unstructured
- Negotiation: Negotiated decisions, Unilatteral decisions

The Object Modeling group, OMG, in its Business Motivation Model (BRG, 2010) conceptualizes a business in terms of policies and directives that govern their enforcement. This suggests to us yet another classification of decisions that is based on the nature of the task to be carried out, namely, policy formulation, determination of policy enforcement rules, operational decisions.

The foregoing implies that there are two broad kinds of decisions in the decision environment,

- ***Managerial:*** Managerial decision making deals with setting up the business. It is strategic kind of decision making where the organization decides its policies and how these shall be enforced.
- ***Imperative***: Operational kind of decision making. This corresponds to our operational decisions to manage the business processes and, in general, monitor that the operational organization is working in accordance with the managerial decisions that have been taken. Deviations, if any, call for corrective decisions.

It follows that imperative decisions can only be taken after managerial decisions have been taken. Thus, Managerial decisions provide the context for the imperative decisions.

Managerial decisions are of two kinds, the policies to be followed and the enforcement of these policies. The former is referred to as Policy formulation decisions and the latter as Policy Enforcement Rule (PER) formulation decisions. Thus, managerial decisions can be seen to be in two layers, Policy formulation decisions are first taken and then PER formulation decisions are taken. Policy formulation can be done by following policies defined by regulatory bodies or by best practices in the domain. Policy enforcement rules deal with corrective actions that need to be taken when policy violations occur. Thus, policies are the input to this layer. Only once policies are formulated can they be enforced. This makes the Policy layer provide the context for Policy enforcement rule layer with the latter nested in the former as can be seen in Fig 2.2.

Fig 2.2 also shows that Imperative decisions are nested in the policy enforcement level. The set of corrective actions provided by the policy enforcement level are input to the tactical level. In the tactical level a decision of which action is to be executed is taken.

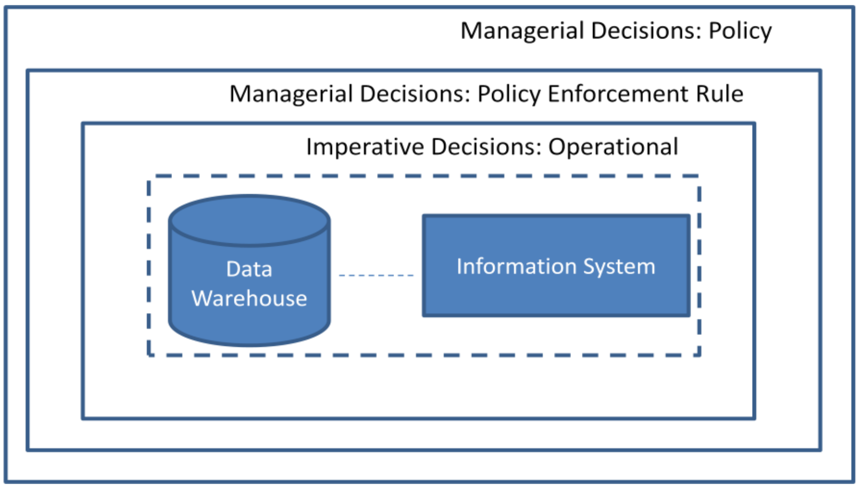

Managerial Decisions: Policy
Managerial Decisions: Policy Enforcement Rule
Imperative Decisions: Operational
Data Warehouse
Information System

Fig 2.2: The decision continuum

Since each layer addresses a different level of management in an organization, the decision maker concerned with each layer also varies. So, the policy layer has upper management, senior functionaries as decision maker, PER layer has a lower level of management and the imperative layer has the lowest level of management as its decision maker.

Recall that while the Data Warehouse provides information for taking decisions, the information system in turn populates the warehouse. Changes in the information system are reflected in the Data Warehouse when the latter is refreshed. This interaction between the IS and the DW is shown by the dotted line in Fig 2.2.

To further illustrate the difference between the three layers consider the following example. Different hospitals can adopt different doctor: patient ratios, 30:1, 20:1 or 15:1. The decision of which ratio to adopt is taken by managers and other senior functionaries of the hospital.

This decision is a Policy level decision. Fig 2.3 shows the choice set as {1:30, 1:20, 1:15}. The decision maker selects one from this choice set. Once a particular ratio is adopted as a policy, the next question is of enforcement of the policy for which rules have to be defined.

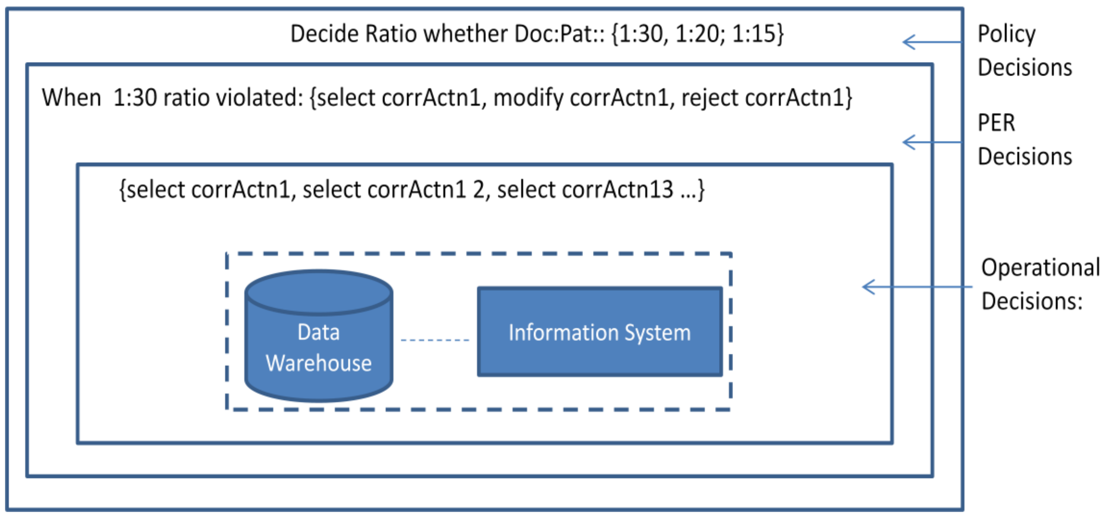

Decide Ratio whether Doc:Pat:: {1:30, 1:20; 1:15}
Policy Decisions
When 1:30 ratio violated: {select corrActn1, modify corrActn1, reject corrActn1}
PER Decisions
{select corrActn1, select corrActn1 2, select corrActn13 …}
Operational Decisions:
Data Warehouse
Information System

Fig 2.3: The difference between the different layers of the decision continuum

Now, consider the next inner layer. Assume that the policy ratio of 1:30 is selected. This will be input to the PER decision layer. Events like buying/discarding equipment, introducing a new specialty etc may increase/decrease the number of patients and thus alter the ratio. Corrective actions like hire new doctor, transfer doctor from another wing may have to be taken to increase the number of doctors and remain compliant with the policy. Now, which corrective action is to be part of the enforcement rule is the decision problem for a PER decision maker. Thus, the choice set presented is {select correctiveAction1, modify correctiveAction1, reject correctiveAction1} and depending on the choice exercised a particular corrective action either directly becomes part of the PER, or a modified version is adopted. If the choice is to reject the corrective action, then naturally the action will not be part of any PER.

For the imperative layer, the PER formulated in the PER layer are input and the decisional problem is to select which corrective action is to be taken. For example, the decision maker can decide to transfer a doctor.

For decision making at each level the Data Warehouse is consulted. Relevant information for each level must be present in the Data Warehouse. Each layer can have independent Data Warehouses. However, recall the decision continuum establishes that any decision at the imperative level must not violate policy enforcement rule and decisions at policy enforcement rule level must not violate related Policy. This means separate, independent solutions for strategic and operational decision making cannot be provided. An integrated enterprise wide resource, the integrated DW is needed that can address strategic as well as operational decision making.

2.3 Meta-Model of Decisions

The decision meta-model, expressed in UML notation, shows three kinds of Decisions: Atomic, Abstract and Complex. An atomic Decision is the simplest decision that cannot be decomposed further into its parts. An abstract Decision is arrived at by using generalization/specialization principles. This gives rise to ISA relationships between decisions. Finally, a complex Decision is composed of other simpler decision requirements. Complex decision requirements form an AND/OR hierarchy. (The notions of this hierarchy have already been introduced in Chapter 1.) Notice that each decision belongs to either the Managerial or Imperative level in the Decision Environment.

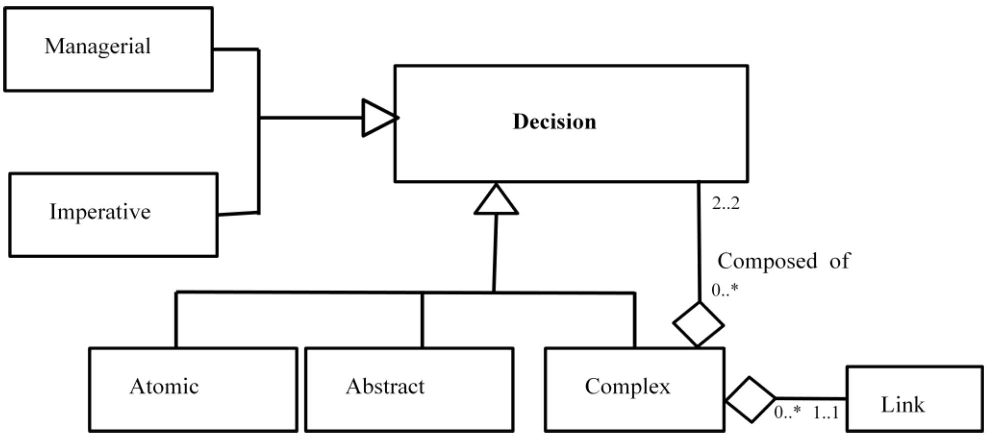


Fig. 2.4: Decision Meta Model

To illustrate an abstract Decision, consider an automobile plant that makes 1-tonne and 13-tonne trucks. Let the decision of interest be *Set up New Assembly Line.* This decision can be specialized into two decisions *Start New 1-tonne Line* and *Start New 13-tonne Line* respectively. Each of these is an ISA relationship with *Set up New Assembly Line* as shown below in Fig 2.5. in shared target style of UML.

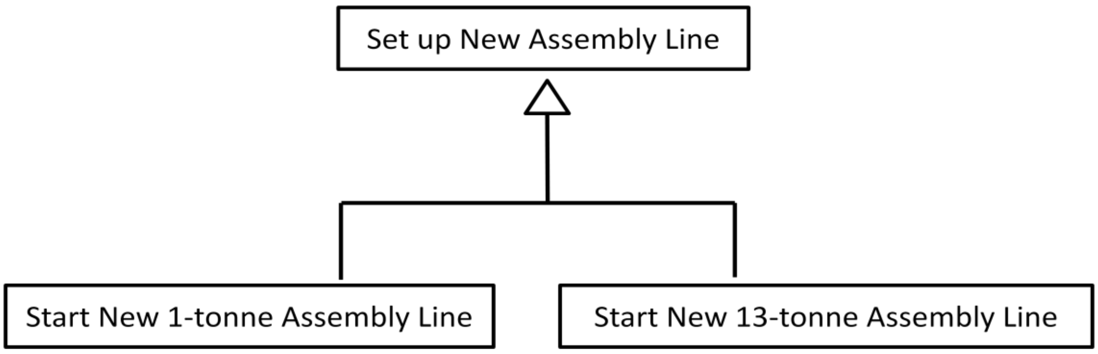

Set up New Assembly Line
Start New 1-tonne Assembly Line
Start New 13-tonne Assembly Line

Fig 2.5: An Abstract Decision

The introduction of an abstract decision is motivated by the consideration as follows. It is possible to treat the specialized classes of an ISA hierarchy as flat structures by introducing attributes in atomic/complex classes, and, vice versa, it is possible to convert classes with common attributes into an ISA hierarchy. Thus, whether hierarchical abstraction is used or not in a particular situation is only determined by the clarity that it brings to a schema. An

ISA hierarchy brings out commonality explicitly, which remains hidden if an attribute is used instead of it.

Our inclusion of an abstract decision provides for explicit representation of ISA structures. Thus, in our example, the complex decision, *Set up New Assembly Line* is a generalization of its two specialized decisions, *Start New 1-tonne Line* and *Start New 13-tonne Line* respectively. These latter two decisions inherit the properties *Set up New Assembly Line*, that is, the AND/OR structure of Fig. 2.6b.

Now, the Decision *Set up New Assembly Line* is a complex one having two component decision requirements, *Create separate profit centre* and *Merge with existing profit centre* (see Fig. 2.6 (a)). An OR link connects these two components so as to define the complex decision.

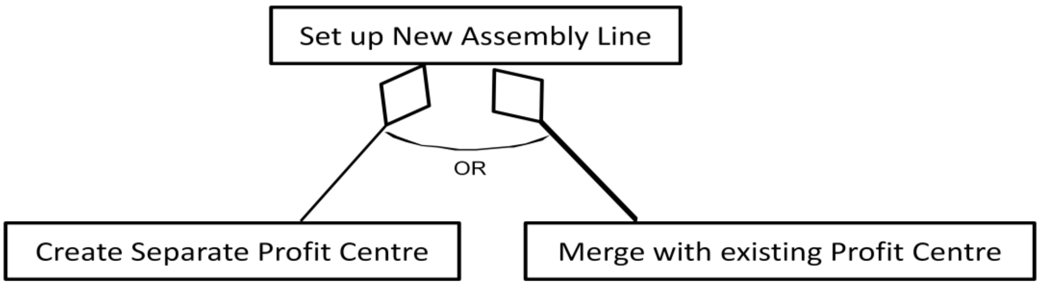

Set up New Assembly Line
OR
Create Separate Profit Centre
Merge with existing Profit Centre

Fig 2.6 (a): A Complex Decision with OR link

Fig. 2.6 (b) shows an example of a complex decision with a combination of AND/OR links. Two more components are shown, *Decide Capacity* and *Choose Location* with an AND link to the existing OR link between *Create separate profit centre* and *Merge with existing profit centre*.

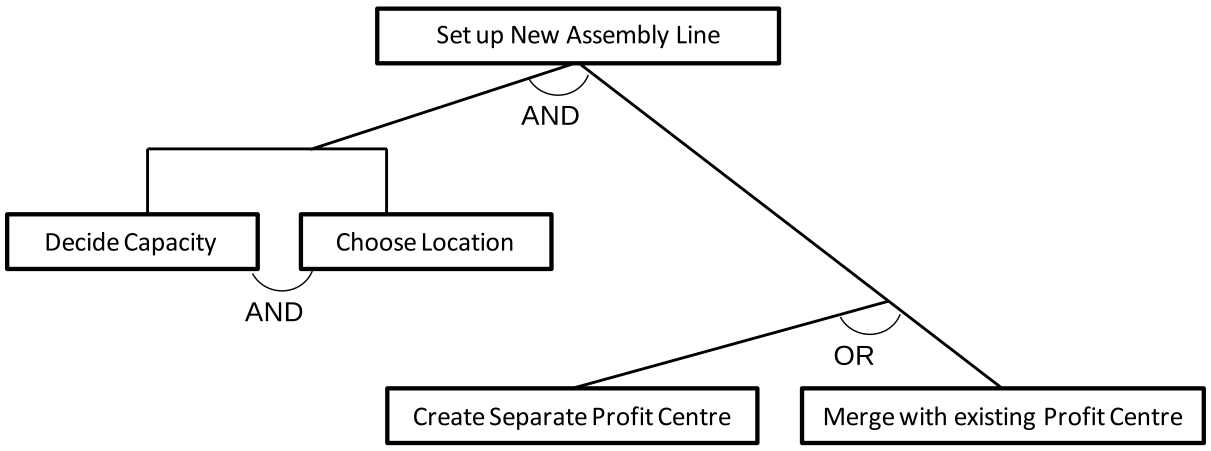

Set up New Assembly Line
AND
Decide Capacity
Choose Location
AND
OR
Create Separate Profit Centre
Merge with existing Profit Centre

Fig 2.6 (b): A Complex Decision

2.4 Modeling Information

Since information is required to take decisions, we introduce our information model here. Let there be a set of decisions D = {D_1, D_2…. D_n}. Let I_1, I_2…. I_n be the sets of information relevant to the corresponding decisions of D where the set I_i = {I_{i1} , I_{i2} , $I_{i3….}$ I_{ik}}, i between 1 and n. We shall refer to I_{ik} as an instance, member or element of Information interchangeably. Then, the set of relevant information to D, represented as Information in Fig. 2.7 is defined as the union of I_1, I_2…. I_n. In other words,

Information = I_1 **U** I_2 …. **U** $I_{n\,=}$ {I such that I belongs to I_k, k between 1 and n}

Now three kinds of information are relevant to data warehousing (Kimball 2002; Inmon, 2005), detailed information which is at the lowest level of granularity, summarized or aggregated information, that is obtained from other detailed/aggregated information, and historical information that may be the history of detailed information or of aggregated information. This information has its own dimensions. For example,

- Detailed information: sales transaction
- Aggregated information: weekly summary of sales

- Historical information: sales for the last three months, weekly sales for the last three months

Fig. 2.7 introduces the corresponding typology, Detailed for detailed information and Aggregate for aggregated information. The figure shows that an aggregate is obtained by the 'Is computed from' relationship between Aggregate and Information.

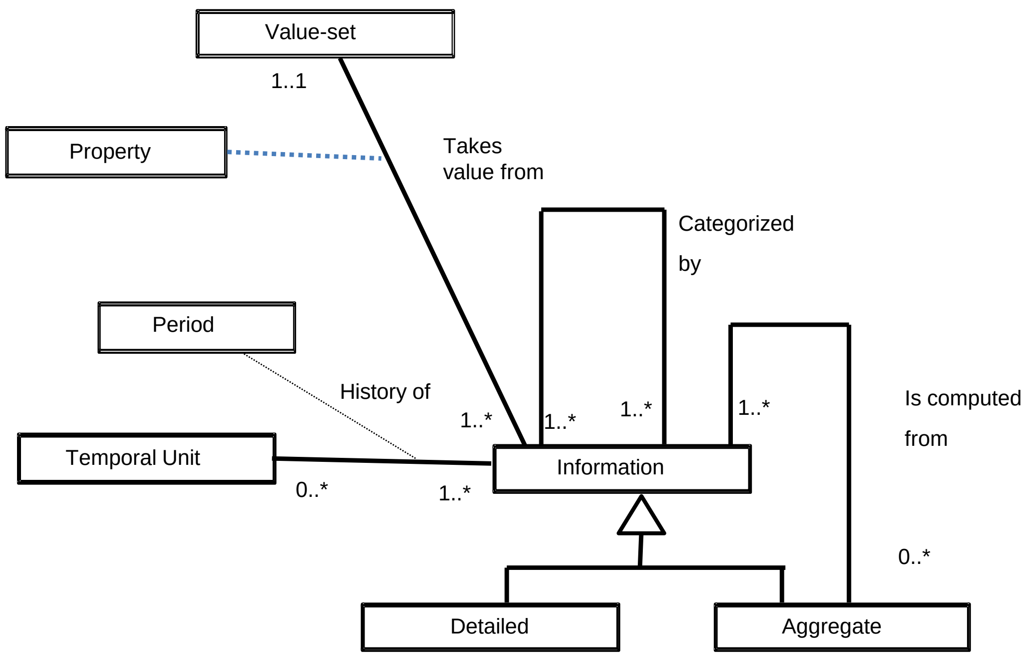


Fig. 2.7: Information Model in Data Warehouses

Historical information is represented by the relationship 'History of' between Information and Temporal unit. The cardinality of this relationship 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 maintained. However, when a temporal unit is associated with information then the number of years of history to be maintained may also be of interest. This is captured, as shown in the figure, by the attribute Period.

Data warehouse schemas are multi-dimensional (Kimball, 2002; Inmon, 2005) with fact data and dimension data. This is represented in Fig. 2.7 by the relationship, categorized by, between information and information. An example of a fact and its dimension is sales by salesperson. Here, sales is a fact and salesperson is its dimension. In terms of Fig. 2.7, fact is categorized by salesperson.

Information is also associated with a value-set and takes on values from it. In Fig. 2.7 this association is called “Takes value from”.

In the subsequent chapters and sections of this thesis, the word information and entity is used interchangeably. This can be seen particularly while arriving at the information base in chapters 3, 4 and in the implementation chapter, namely, chapter 7.

2.5 Decision Requirement

In order to make a decision, reference to the information in the Data Warehouse needs to be made. This is represented as a pair <decision, information> and referred to as a decision requirement. The notion of decision requirement is elaborated below.

2.5.1. The Decision Requirement Meta-Model

The Decision Requirement, DR, meta-model is shown in Fig. 2.8. As shown it is modelled as an aggregate of information and decision.

Fig. 2.8 shows that there are three kinds of decision requirements, atomic, abstract and complex. An atomic DR is the smallest decision requirement. It cannot be decomposed into its parts. An abstract DR is a decision requirement that is arrived using generalization/specialization principles. This gives rise to ISA relationships between decision requirements. Finally, a complex DR is composed of other simpler decision requirements.

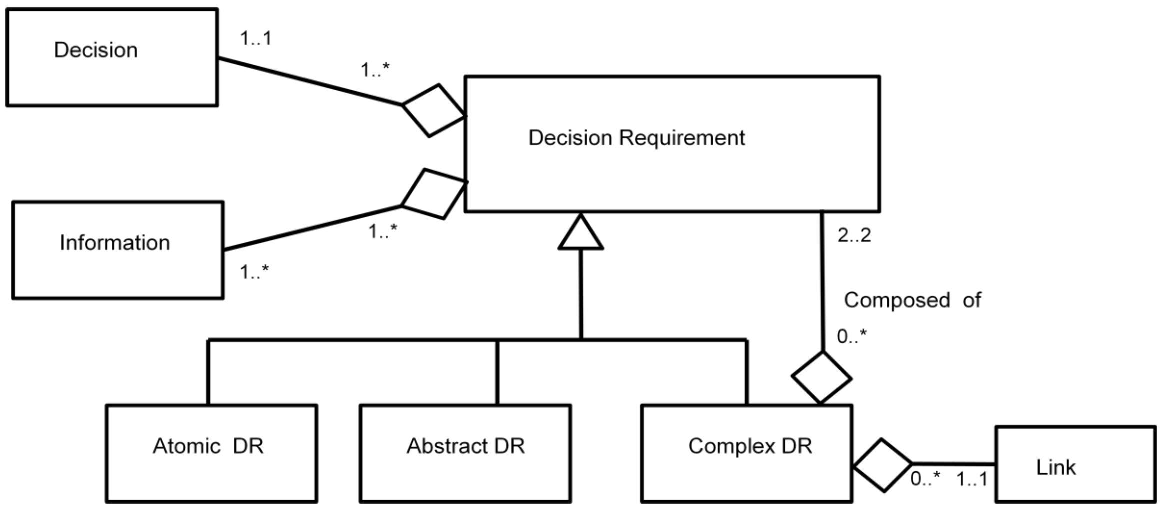


Fig. 2.8: Decision Requirement Meta Model.

Notice that the typology of a decision requirement follows that of a decision closely. Indeed, since a decision can be reduced to an AND/OR graph, so also, a decision requirement can be decomposed into an AND/OR graph. The only difference is in the association of the notion of information with a decision requirement.

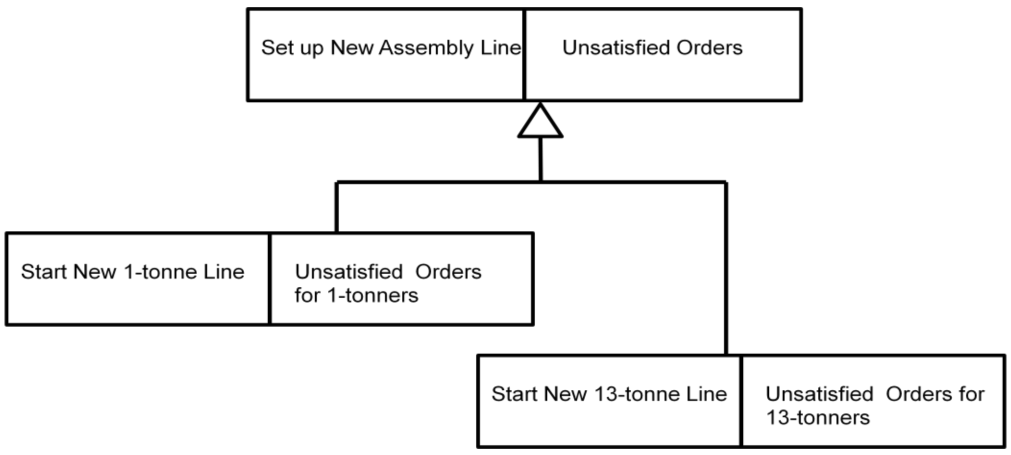


Fig. 2.9: Abstract Decision Requirement with an IS A hierarchy

To illustrate an abstract DR, consider, see Fig. 2.9, the decision *Set up New Assembly Line*. Let the required information be *Unsatisfied Orders*. This DR can be specialized into two DRs with decisions *Start New 1-tonne Line* and *Start New 13-tonne Line* respectively and required information, *Unsatisfied Orders for 1-tonners* and *Unsatisfied Orders for 13-tonners*. Each DR is an ISA relationship with *Set up New Assembly Line*.

Now let us consider composition. The Decision Requirement <*Set up New Assembly Line, Unsatisfied Orders*> is a complex one having two component decision requirements, <*Decide Capacity, Resources Available*> and <*Choose Location, Land Availability*>. An AND link connects these two components so as to define the complex decision requirement, <*Set up New Assembly Line, Unsatisfied Orders*> (see Fig. 2.10).

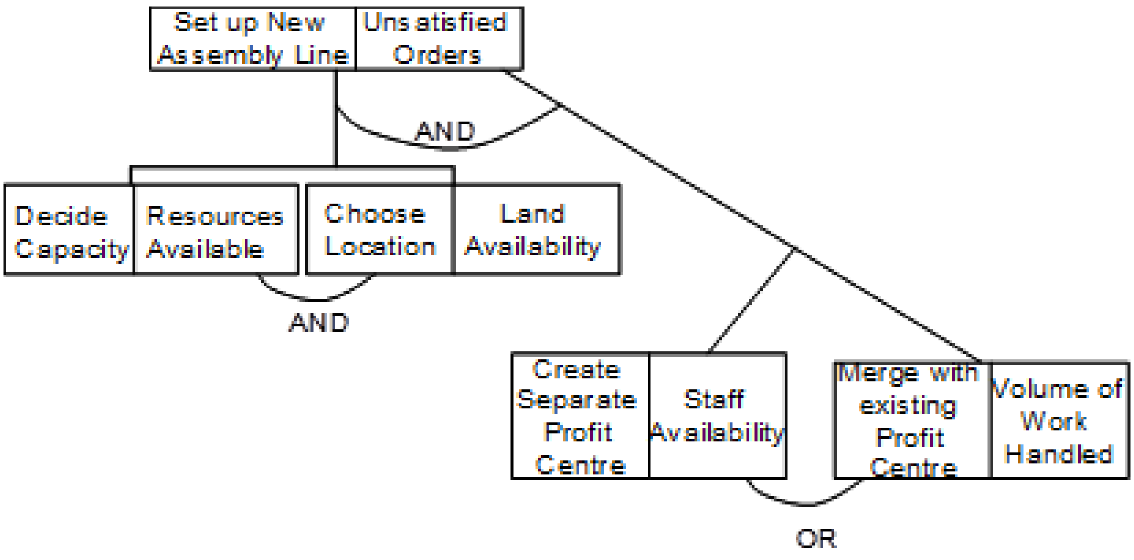


Fig. 2.10: Composition of Decision Requirements with AND and OR link

The foregoing shows that a DR can be decomposed to reflect the decomposition of its decision component. It is also possible to do DR decomposition through information decomposition. In this case, the decision part is held constant whereas information components are elaborated. The Choose Location decision of Fig. 2.10 is shown as associated with the information, Land Availability. Land availability can be decomposed into two pieces of information, Land site and Land size Then the complex DR <Choose location, Land availability> can be decomposed into <Choose Location, Land site> and <Choose Location, Land size> respectively.

2.6. Information Elicitation Techniques

The thesis proposes three generic information elicitation techniques as described below. These can be applied to any layer of the decision continuum.

Critical Success Factors

Bullen and Rockart (Bullen and Rockart, 1981) consider a CSF as a key area of work in which success is essential for a manager to meet his goals. A manager should have full information to determine if work is proceeding well in the area. It has been pointed out that most managers have only a few critical success factors, typically 4-8 (Wetherbe, 1991). Bullen and Rockart lay down an interviewing technique for eliciting CSFs.

Our interest is not in defining the CSFs of a manager. This task can be performed by using the technique of (Bullen and Rockart, 1981). Instead, given already defined CSFs, we are interested in obtaining information for estimating CSF satisfaction, and therefore in defining an elicitation technique for this information.

The CSF Information elicitation technique, CSFI Elicitation for short, obtains information required to assess progress in critical work areas. The essential question here is to identify the variables/measures that must be monitored to ensure that these factors remain in control. This control is carried out by appropriate decision making.

Table 2.1 shows the essence of the CSFI technique. In the first two columns, the CSF and the decision with which it is associated is tabulated. The third column contains the variables that go into assessing the CSF. Finally, the last column contains the information relevant to the variables.

Table 2.1: CSF Information

Decision	**CSF**	**CSF Measure**	**Information**
Add New Pharmacy	Medicine delivery	• Waiting time of patient	**Aggregate:** average waiting time **Category :** patient type **History:** *time unit:* week *Duration* 10 weeks

The example presented in Table 2.1 is for the decision of adding a new pharmacy in the health service. The CSF associated with it is Medicine delivery since it is a critical work area in the service. One variable that helps in assessing the CSF is the waiting time of patients at the pharmacy. The information needed for this variable is the average waiting time categorized by patient type and a weekly record of this informtion needs to be kept for a 10 week duration.

Note that, in general, there may be more than one measure for a given CSF. However, we have exemplified out technique in Table 2.1 with one variable.

Ends Achievement

Ends achievement can be considered in two different ways, depending upon the way one conceptualizes the notion of Ends. These are as follows:

1) An end is a statement about what is to be achieved a goal. Notice that an End is different from a CSF in that the latter is a work area where success is critical to the manager. In this view, one can do Ends analysis by asking which ends contribute to the achievement of which other ends. When this is applied recursively then we obtain an Ends hierarchy. One technique used is **means-ends analysis**. In this, the problem solver begins by envisioning the end, or ultimate goal, and then determines the best strategy for attaining the goal in his current situation. Here, a means-ends hierarchy is built in which nodes at a certain level are goals and those at the next lower level are means of achieving it. Means-ends analysis is recursively applied till the leaves of the hierarchy are reached.

2) The second view of Ends achievement views an End as the result achieved by performing a task or as the intended result of a decision. When compared with view 1) above, one does not ask which end achieves a given end. Instead, one asks what information is needed to ensure the effectiveness of the end. In other words, Ends analysis here is the identification of information needed to evaluate the effectiveness of the end.

There is a difference between the notion of a CSF and this view of Ends. Whereas a CSF is about success in a critical work area, an End is the expected result of a decision. A CSF is a more ‘macro’ issue whereas an End is relatively more focused and is at a ‘micro’ level’.

Since our interest is in determining information, we adopt the second view. In our context, ‘Ends’ refers to the result achieved by a decision. It identifies a concrete change in the organization that is a consequence of the decision. The decision-maker/requirements engineer interaction is centred round determining the information for the effectiveness of the result.

Ends Information Elicitation, EI elicitation, is the identification of information needed to evaluate the effectiveness of the end to be achieved. The requirements engineering task is that of determining the variables and information of interest in estimating this effectiveness.

Table 2.2 shows the four aspects of EI elicitation. In the first two columns, the End and the decision with which it is associated is tabulated. The third column contains the measure that go into assessing the effectiveness of the End. Finally, the last column contains the information relevant to the variables.

Table 2.2: Ends Information

Decision	**End**	**End Effectiveness Measure**	**Information**
Add New Pharmacy	Profitability	Service provided	**Aggregate:** total sales **Category :** medicine-wise
			Aggregate: Number of transactions **Category:** Shift-wise

We continue in Table 2.2 with the example for the decision of adding a new pharmacy. The End associated with it is Full Utilization. An effectiveness variable that helps in assessing the effectiveness of the End is the service provided. The information needed for this variable is the total sales medicie-wise. The second row shows additional informaiton, the number of transactions during each shift.

As for CSFI, there can be more than oneeffectiveness variable per End and there can be many Ends for a decision.

Means Efficiency

Broadly speaking, a means is a way of achieving the ends. When considering Ends achievement, we have mentioned the use of Means-Ends for developing the ends hierarchy. A lower level in the hierarchy is the means of achieving the immediately higher level. Both levels describe the same system but in different terms.

There is yet another way of looking at Means. This view treats a Means as a first class concept of the business world. A means is of direct interest in the business world, just as an Ends is or a CSF is. It is the instrument, the process, the activity or task deployed to achieve an End. The interesting question for a manager is the efficiency of the deployed means. Thus, Means Efficiency deals with identification of information for evaluating the efficiency of the means. The requirements engineer/stakeholder interaction is now centered round eliciting variables that provide information on the efficiency of the means adopted for each decision.

We can again understand Means Information elicitation through the 4-column Table 2.3. As before, the first two columns associate the Means with a decision. Thereafter, the efficiency of the Means is captured in a measure, and finally, the information is obtained.

Table 2.3: Means Information

Decision	**Means**	**Means Efficiency Measure**	**Information**
Add New Pharmacy	Establish afresh	Resources Used	Estimated cost Category: Resource wise
		Time	Setting up time

The example in Table 2.3 is for the same decision, Add New Pharmacy. The means is to start completely afresh and not reuse an existing building. The efficiency variables are the resources, civil, electrical, fixtures and furniture etc. that shall be used. The information needed is the cost for each resource. The second row of the table shows that efficiency can be estimated as the time to set up the new pharmacy, and the total start up time is the information to be maintained.

Summary

This chapter has developed a model for decision requirements in the tradition of model driven requirements engineering. The consequence has been that the notions of decision and information have also been modelled. As a result of these models, the structuring of information (in the last step of the three elicitation processes) has been separated from the task of eliciting it as in the third step of the three elicitation processes. In other words, the third step is concentrated on the mere obtaining of information without attempting to structure it in any way. It is because of this that the thesis postulated, in Chapter 1, that the elicited information is early information.

Having obtained early information, the task thereafter is to structure it in multi-dimensional form. To do this, the thesis proposes to build an ER diagram for the early information and thereafter convert it to the star schema using Golfarelli's algorithm (Golfarelli *et al.*, 1998). The manner in which this is done shall be taken up in the next chapter, in the context of PER decision formulation.

Note: The ideas of this chapter have been published in Information Systems Evolution (2010). Springer.

Chapter 3

Policy Enforcement Level Decision making

Chapter 2 considered the notion of a decision and information as forming the decision requirement. The notions developed there are neutral to the nature of decisions, managerial or imperative. That is, the proposals of Chapter 2 are generic and can be applied to any kind of decision, managerial or imperative.

Let us consider managerial decisions. Policy formulation is done in a number of contexts, in government/public policy formulation (Lindbloom, 1993), in the corporate/business world (Hillman and Hitt, 1999), in specific areas like manufacturing (Park, 2000), and accounting (Newton, 1980). The stakeholders are also varied and comprise (Lindbloom, 1993; Ritchie, 1988) general public, opinion makers, service providers, service users, activists etc. Some other factors going into policy formulation are

- The role of other corporates (Cooke and Morgan, 1993) in the formulation of policy by a business house
- Dependence (Park, 2000) on other related policies
- The role of consensus building (Ritchie, 1988)
- The strategy (information, financial incentive, and constituency building), level of participation (individual, collective) and action (transactional and relational) to be adopted (Hillman and Hitt, 1999)

From the foregoing, we see that policy formulation is a many-facetted and complex task. A full treatment requires a focused effort that is left for a separate investigation. Therefore, in this thesis, we assume a policy representation system and focus on policy enforcement rule decisions and operational decisions.

This chapter presents the first life cycle, one for policy enforcement rule (PER). The life cycle shows discovery of early information in the RE stage from organizational policies using a multi-step process. As stated in the introduction, this early information can be used to arrive at the star schema for DWper using guidelines proposed that first convert early to late information in the form of ER diagram. Subsequently, existing techniques are used to convert ER schema into star schema. The use of early information as an input to the vertical integration life cycle will be discussed later in chapter 5.

3.1. PER Life Cycle

Table 3.1 shows the various stages of the PER life cycle namely, Requirements Engineering, Conceptual and Logical Design phase. As can be seen, the input to the life cycle is formulated organizational policies that have to be enforced. Therefore, it is assumed here that formulation of policies is already performed in the policy level of decision making.

Requirements Engineering stage: has three sub stages. In the Rule Formulation Substage, policy enforcement rules (PER) are formulated to enforce organizational policies. The PERs are represented in the 'WHEN triggering action IF condition THEN correcting action' form. Elicitation of both triggering and correcting actions are done by applying proposed guidelines and policy enforcement rules are formulated. The decisional problem here is to decide on the correcting actions that are to be part of the PER. In order to take this decision, information is needed and this is elicited using the generic information elicitation techniques in the Early Information Substage. This information is 'early'.

Table 3.1: PER life cycle

Stage		Input	Output
Requirements Engineering	Rule formulation Substage	Organizational Policies	Policy Enforcement Rules
	Early Information Substage	Policy Enforcement Rules	Early information
	Late Information Substage	Early information	ER Schema
Conceptual Design		ER schema	Multi-dimensional model
Logical Design		Multi-dimensional model	MOLAP/ROLAP

Early information needs to be converted into a more structured form and this is done in the late information substage. As can be seen in Fig 3.1, information, which is 'early', is converted into ER schema. For this the thesis defines guidelines.

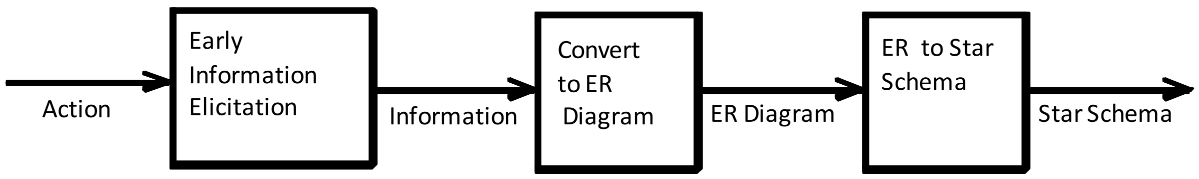


Fig 3.1: Overall process of arriving at star schema from PER actions

Conceptual Design Phase: ER schema obtained in the previous stage is converted into multi-dimensional structures. For this the thesis relies on existing techniques of Golfarelli (Golfarelli *et al*., 1998) and Moody (Moody and Kortink, 2000).
Each step is looked at in detail in the subsequent sections of the chapter.

3.2. Policy Representation

Again notice, formulated organizational policies are input. This section presents a representation system for policies. This is because PERs are derived from policies, which implies that their own structure will be derived from the structure of the policy they are enforcing.

We have taken recourse to representation of a policy in a logic. The first order logic (Boulos, 2002; Jeffrey, 1991; Shoenfield, 1967) has been used extensively in computer science, for example, in database technology to formulate the tuple and domain relational calculus respectively that form the basis of query languages SQL and QBE; in artificial intelligence for knowledge representation, and by Object Modeling Group, OMG, in its Semantics of Business Vocabulary and Rules, SBVR (OMG, 2008) for representing business rules.

There are a number of limitations of the logic, its expressiveness and of the fragments of natural languages that it can describe. From the former stand point, (Boulos, 2002), (Jeffrey, 1991), (Shoenfield, 1967) the first-order logic is undecidable. Further, extensions like infinitary logics and higher-order logics are more expressive and are needed to permit categorical axiomatizations of the natural numbers or real numbers.

A number of features of natural language cannot be represented (Gamut, 1991) in the first order logic for example, quantification over predicates, predicate adverbials, relative adjective, prepositions etc. Additionally, counting quantifiers, like at most N or at least M, needed in many natural language sentences, are missing. For natural language analysis the logic system to be used needs (Gamut, 1991) a much richer structure than first-order predicate logic.

Now policies of organizations are likely to be expressed in natural language. We are not interested in performing natural language analysis and our interest is limited to showing how we can move from policies to policy enforcement rules. Thus, from our point of view, the first order logic, even with its limitations, provides the basic framework that we can use for converting policies to policy enforcement rules.

Now, we represent policies in a logic system, Extended First Order Logic, 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 that refers to a variable
  - an *n*-adic function symbol applied to *n* SVs
- A *collective term, CT,* is
  - a CV that denote a set of values
  - an *n*-adic function symbol applied to *n* CVs

- An *atom* is an n-ary predicate P defned on 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)
- 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 $\rightarrow$ F2 is also a formula
- If F1 is a formula then $\exists$sF1 and $\forall$sF1 are formulae. Here s is a variable, 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, $\forall, \exists$
- Logical AND, OR, NOT

Notice that besides usual first order features (constants, variables that denote individuals, predicates and functions) the formulation also has set variables (CV) that can be quantified.

Shown below are two examples. The first example uses quantification over an SV whereas the second shows quantification over a CV.

Example 1: Consider a policy “every Ayurvedic hospital must run an O.P.D”.

- Ayu(x): x is a Ayurveda hospital
- Run(x, OPD): x must run an OPD
- OPD is a constant

Its representation is:

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

Example 2: "A Semi-private Ward must have area of 200 Sq. ft. for 2 beds". Its representation is as follows:

- spw(x): x is a semi private ward
- EQ(x,c1): x is equal to c1
- B is a set of beds
- Belongs(x, y): y belongs to x

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

3.3. Rule Formulation Sub-Stage

In order to develop guidelines for formulating policy enforcement rules, policies are classified into four categories based on the type of formulae used for their representation. It can be seen that formulae can be classified into two groups Simple and Complex. Formulae with the following are said to be Complex:

- n-adic functions
- conjunctions and disjunctions

All other formulae are simple.

Further, recall policies are of the form Quantifier(If F1 THEN F2). Depending on the nature of F1 and F2, there are four types of policies.

Simple-Simple (SS): Both F1 and F2 are simple. For example the policy, all doctors must have a degree in M.D. represented as ∀x(doc(x)→ degree(x, MD)), is a simple policy. Both formulae on the LHS and RHS of the implication are simple.

Table 3.2: Types of Policies

S.No	**F1**	**F2**	**Policy Type**
1	Simple	Simple	Simple, SS
2	Simple	Complex	Complex, SC
3	Complex	Simple	Complex CS
4	Complex	Complex	Complex CC, subsumed in 2 and 3

A complex policy has at least one formula as complex. When the complex formula is on the RHS, the policy is of **simple-complex (SC) type**. Consider the policy, ∀y∃B∃N(GB(y)→ EQ(ratio(count(B), count(N)), ratio(8,1)) AND Belongs (y,B) AND Belongs(y, N)). Formula on RHS of the implication involves function count and conjunction operator AND. Therefore it is complex. The LHS is a simple formula. Thus, this policy is simple-complex or SC. Another example is ∀x∃B(S(x)→ LEQ(count(B),3) AND GT(count(B),1) AND Belongs(x, B)) where the RHS formula has both function, count(b) as well as AND as conjunction and thus complex.

When the LHS has a complex formula and the RHS is simple the policy becomes **complex-simple (CS)**. For example, ∀x (nurse(x) AND GEQ(salary(x), 15000)→ provide (x, ProvidentFund)) is of type complex-simple because the LHS of the implication has conjunction AND, and so is complex. The RHS is a simple formula. A **complex-complex (CC)** policy has both formulae as complex. The policy ∃wtabSet∃ftabSet (woodTable(wtabset) AND fibreTable(ftabset)→ EQ(Sum((count(wtabSet), count(ftabSet)),2)) is of the CC type. As can be seen the LHS of the implication has a conjunction OR and RHS has function count, both of which are complex.

There are two conditions when policy violation can occur in the general form Quantifier(IF F1 THEN F2). If there is an action A that causes F2 to become False when F1 is True. In this case, action B needs to be taken to make F2 True. The second condition may be when action A causes F1 to be False. Then action C needs to be taken to disallow action A.

Now, the problem is (a) representing policy enforcement rules and (b) elicitation of actions A, B, and C above for given policies. Each of these is considered in turn.

3.3.1. Representing PERs

Policy enforcement rules belong to the class of business rules. There are two broad approaches to business rules representation, natural language based and logic based. In (Leite and Leonardi, 1998) one finds a baseline of business rules expressed as statements in natural language following specified patterns. Another variant of the natural language representation is the use of templates (Sosunovas and Vasilecas, 2006). A template consists of template expressions, for example determiner expression, subject expression, characteristic expression and so on. Semantics of Business Vocabulary and business Rules, SBVR, (OMG, 2008) proposes its own SBVR Structured English for expressing business rules.

The semantics of SBVR Structured English expressions are rooted in its formal, logic-based system. The SBVR formalism has no notion of activity fact type (OMG, 2008; Steen *et al.*, 2010). (Fu, 2001) proposed a predicate logic based Business Rules Language, BRL. BRL is limited in scope and only a small number of built-in predicates are captured.

As is well known, logic based representation can be expressed in the IF-THEN form. This form has been used by (Auechaikul and Vatanawood, 2007); a variant IF-THEN-ELSE- by (Muehlen and Kamp, 2007) and WHEN-IF-DO by (Rosca, 1997).

Now, as discussed in the previous section, the representation of policy enforcement rules must allow us to express the basic notion of an action. Actions are interesting from two points of view as follows:

- **Triggering**: This type of action triggers a policy violation. This action could on the Then side of the implication and cause the IF side to be false. It can also be on the IF side causing the Then side to be false. Action A above is a triggering action.
- **Correcting**: As stated once there is a policy violation, suitable corrective action has to be taken. Actions B and C above are correcting actions.

The absence of an activity fact type in SBVR (OMG, 2008) makes one look for a rather more direct way to represent triggers and correcting actions. . Indeed, a representation in logic shall not yield a direct representation of triggers and actions which will need to be derived from the functions/predicates comprising well formed formulas of the logic. Therefore, the work here proposes to represent the triggering aspect of an action in the WHEN part of a rule; the condition to be checked upon the trigger occurring in the IF part; and the correcting action in the THEN part of the rule. Therefore, here representation of a policy-enforcing rule is

WHEN triggering action IF condition THEN correcting action

Notice the similarity of the policy-enforcement rule with that of the notion of a trigger in SQL. A trigger (Elmasri, 2004) is a stored program, a pl/sql block structure that is fired when INSERT/UPDATE/DELETE operations are performed and certain conditions are satisfied. There are thus three components to a trigger, an event, a condition and an action corresponding to the WHEN, IF, and THEN part respectively.

In SQL, a trigger is seen as an executable component. However, a policy enforcement rule is a directive that governs/guides [BRG, 2010] a future course of action. Seeing this similarity with SQL, we use here the basic idea behind a range variable of SQL.

The remaining question is about the representation of an action. Actions, both triggering and correcting, are of the form <verb> <range variable>. To see this let us first consider the notion of a range variable.

The notion of range variable here is similar to that in SQL. Whereas in SQL the range variable ranges over a tuple, the range variable here represents a single instance of a noun. Once defined, a range variable can be repeatedly used. Before using it, a range variable must be declared using the form:

<noun> < range variable >

In this declaration, noun can be a simple noun or constructed by noun-noun modification. As an example of noun-noun modification consider, *Ayurvedic Hospital* built over two nouns Ayurvedic and Hospital, the former is a noun that modifies the latter. We will italicize range variables for easy identification.

As examples of declaration of range variables, consider

<OPD> <*x*>

<Ayurvedic Hospital> <y>

In the first example, OPD is a noun and x is its range variable. This says that x is an instance of OPD. Similarly, in the second example, y is an instance of Ayurvedic Hospital.

Now we can construct actions which, as mentioned above are of the form <verb> <range variable>. Using the range variable x and y declared above we can define actions, create *x* and operate *y* respectively.

The policy enforcement rules take the form

WHEN <verb> <range variable> IF condition THEN <verb> <range variable>

3.3.2. Enforcing Policies

In order for the requirements engineer to formulate PER for a policy P, s/he has to decide on the possible correcting actions for a triggering action. Let this a the set {corrAction1, corrAction2, corrAction3.... }.

On examining this set closely, one finds that in fact with every action there is a choice the requirements engineer has to make, whether to select, modify or reject the action. In other words, the choice set presented to the requirements engineer is:

{**select** corrAction1, **modify** corrAction1, **reject** corrAction1, **select** corrAction2, **modify** corrAction2.........}

The actions selected become part of the PER, rejected actions are not part of any PER and the modified actions become part of the PER. For example, if corrAction1 and corrAction2 are selected and corrAction3 is rejected then the requirements engineer arrives at two PER

WHEN trigAction1 IF violatingCondition THEN corrAction1

WHEN trigAction1 IF violatingCondition THEN corrAction2

Note, the same action can be a correcting action for more than one kind of triggering action. Also, a triggering action in one PER can be a correcting action in another PER and vice versa.

The next task is to develop guidelines to elicit actions and further to formulate policy enforcement rules.

3.3.3. Guidelines for Eliciting Actions

This section develops macro guidelines to elicit actions. Both sides of the implication are examined. Actions are elicited in the form described above, <verb> <range variable>. Once this is done the WHEN IF THEN form is filled in with suitable triggering and correcting actions.

- **Guideline I:** This guideline suggests to the requirements engineer to define one or more triggering actions to make LHS true. Now, policy violation occurs when the RHS becomes false. Therefore, correcting actions that cause RHS to become **true** have to be elicited.
- **Guideline II:** this guideline suggests to the requirements engineer to define one or more triggering actions to make RHS false. Policy violation can occur if the LHS of the policy is true. To avoid this, correcting action to make the LHS false has to be elicited.

These guidelines are applied to SS, SC, CS and CC types of policies.

Simple Type Policy

Recall a simple type policy has both formulae as simple.

Example I: Every Ayurvedic hospital must run an Out Patients Department.

$$\forall x[\text{Ayurvedic}(x) \rightarrow \text{Run}(x, \text{OPD})]$$

The first step is to define range variables:

<Ayurvedic hospital> <*x*>

<OPD> <*y*>

Now the two guidelines are applied.

<u>Guideline I</u>

The requirements engineer defines one or more triggering actions to cause Ayurvedic(x) to become true. Let the requirements engineer specify that the action ‘create Ayurvedic hospital’ does this. By guideline I above, correcting actions must be elicited so that Run(x, OPD) is true. Let the requirements engineer specify these as:

a) construct OPD
b) use existing OPD

The policy enforcement rules are:

- WHEN create *x* IF !Run(*x*, *y*) THEN start *y*
- WHEN create *x* IF !Run(*x*, *y*) THEN construct *y*
- WHEN create *x* IF !Run(*x*, *y*) THEN reuse *y*

As can be seen action ‘create x’ is in the required form with create as verb and ‘*x’* as range variable. Here x is an instance of noun phrase Ayurvedic Hospital. Similarly, correcting actions ‘start *y*’, ‘construct *y*’ and ‘reuse *y*’ use verbs, start, construct and reuse respectively. Range variable ‘*y*’ is an instance of OPD.

Guideline II

Now, different actions that can cause the RHS to become false are elicited. Let the action elicited be ‘stop OPD’. In this case, guideline II says that an action is needed to make the LHS of the policy false. The requirements engineer is presented with this and corrective action elicited could be:

a) stop being an Ayurvedic hospital

b) re-designate hospital

Thus, the rules are:

- WHEN stop *y* IF A*yurvedic*(*x*) THEN stop *x*
- WHEN stop *y* IF A*yurvedic*(*x*) THEN re-designate *x*

Complex SC Type Policy

As discussed earlier, in complex SC type policies, the LHS is simple but the RHS is complex (contains conjunctions, disjunctions, n-adic functions). Eliciting actions for LHS as in the case of simple policy types continues to be applicable. However, for each complex predicate on the RHS, elicitation strategies are to be formed.

Fu (Fu *et al.*, 2001) points out that special purpose language cannot have the full expressive power of general languages and recourse to predefined, standard predicates must be taken. Consequently, in (Fu *et al.*, 2001) standard predicates have been introduced that can be connected using conjunction and disjunction operators. Following this, standard predicates are shown in Table 3.3. The elicitation strategy is also shown in this table.

Table 3.3: Elicitation strategies for given RHS predicate

S.No	**RHS Predicate**	**Elicitation Strategy**
1	EQ(F(*x*),c)	If F(*x*) is less than c then elicit operation to Increase F(*x*) If F(*x*) is greater than c then elicit operation to reduce F(*x*)
2	LEQ(F(*x*), c)	If F(*x*) is greater than c then elicit operation to Reduce F(*x*)
3	LT(F(*x*),c)	If F(*x*) is equal to c then elicit operation to reduce F(*x*) If F(*x*) is greater than c then elicit operation to Reduce F(*x*)
4	GT(F(*x*), c)	If F(*x*) is less than c then elicit operation to Increase F(*x*) If F(*x*) is equal to c then elicit operation to Increase F(*x*)
5	GEQ(F(*x*), c)	If F(*x*) is less than c then elicit operation to Increase F(*x*)
6	NEQ(F(*x*),c)	If F(*x*) is equal to c then elicit operation to Increase F(*x*) or elicit operation to Reduce F(*x*)

The elicitation strategies of Table 3.3 aim to make the predicate true. For example, in row 1, if F(x) is less than c, then the attempt is to increase its value so that it satisfies the predicate. On the other hand, if it is greater, then the correcting action must decrease its value. This approach is systematically followed in all elicitation strategies of the table.

Now, the manner in which the two guidelines can be used for the SC type of policies is illustrated below.

Example I: Each private room must have an area of 200 sq ft.

$\forall x\ [pvtR(x) \rightarrow EQ(area(x),200)]$

The range variables are defined as:

<Private Room> <*pr*>

Guideline I

Let the elicited LHS triggering action be ‘create private room’. Since RHS is a complex predicate, the first row of Table 3.3 is applicable. It suggests the following

1. Elicit correcting action to Increase F(x). The choice set may be
    a. Rebuild private room
    b. Expand private room
2. Elicit correcting action to reduce F(x). The elicited action may be
    a. Partition private room

From the foregoing, the following three policy enforcement rules are formulated

- WHEN create *pr* IF LT(area(*pr*),200) THEN Rebuild *pr*
- WHEN create *pr*, IF LT(area(*pr*),200) THEN Expand *pr*
- WHEN create *pr*, IF GT(area(*pr*),200) THEN Partition *pr*

Guideline II

Now, consider triggering action ‘share room’ that causes the available area of a room to reduce. A correcting action is needed to make LHS false. Let this be elicited as ‘relocate private room’.

The policy enforcement rule is:

- WHEN share *pr* IF ! EQ(area(*pr*),200) THEN relocate *pr*

Example II: As a more complex example, consider the semi-private ward policy dealt with before:

$\forall x \exists B[S(x) \rightarrow$ LEQ(count(B),3) AND GT(count(B),1) AND belongs(x, B)]

Range variables are:

<Semi-private ward> <*spw*>

<bed> <*b*>

<u>Guideline I</u>

Notice that triggering action to be elicited is based on the combined action of LEQ and GT. Table 3.4 suggests the following for the two predicates LEQ and GT.

When a new semi private ward is created, then since Count(b) is zero, the second row of Table 3.4 is applicable and elicit actions are as follows:

a. Purchase bed
b. Transfer bed

Table 3.4: Elicitation strategy for LEQ and GT predicates

LEQ	GT
True: elicit nothing	True: elicit nothing
True: elicit nothing	False: elicit operation to increase F(*x*)

False: elicit operation to Reduce F(*x*)	False: elicit operation to increase F(*x*)
False: elicit operation to Reduce F(*x*)	True: elicit nothing

So the policy enforcement rule is:

- WHEN create *spw* IF !GT(count(*b*),1) THEN Purchase *b*
- WHEN create *spw* IF !GT(count(*b*),1) THEN Transfer *b*

<u>Guideline II</u>

If bed is removed from the ward then it may happen that there is no bed left in the ward, i.e., GT(count(*b*),1) is false. In this case correcting action to falsify the LHS is to be elicited. The elicited action may be 'Relocate semi-private ward'. This gives the business rule

- WHEN remove *b* IF !(GT(count(*b*),1)) THEN Relocate *spw*

Example III: For a general ward the bed: nurse ratio must be 8:1

∀x∃B∃N [GW(x) →EQ (ratio(count(B),count(N)), ratio(8, 1)) AND belongs(x, B) AND belongs(x, N)]

Range variables are:

<General ward> <*gw*>

<nurse> <*n*>

And '*b'* as range variable for bed has already been defined in the previous example and so it need not be re-defined here.

Guideline I

When a new general ward is created then the ratio of bed to nurse must be satisfied. Row one of Table 3.3 is applicable. The suggested actions are as follows:

1. Elicit correcting action to increase count(*b*)
    a. Purchase bed
    b. Transfer bed
2. Elicit correcting action to reduce count(*n*)
    a. Transfer nurse
    b. Fire nurse
3. Elicit correcting action to increase count(*n*)
    a. Recruit nurse
    b. Transfer nurse
4. Elicit correcting action to reduce count(*b*)
    a. Discard bed

The seven policy enforcement rules obtained are:

- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Purchase *b*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Transfer *b*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Transfer *n*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Fire *n*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN recruit *n*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Transfer *n*
- WHEN create *gw* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Discard *b*

Guideline II

Let the elicited triggering action be to remove a bed. The ratio of beds to nurse may not be 8:1, then an operation is needed to be elicited to falsify the LHS.

a) Use ward for some other purpose

The enforcement rule formed is:

- WHEN remove *b* IF !EQ (ratio(count(B),count(N)), ratio(8, 1)) THEN Reuse *gw*

Complex CS Type Policy

Here LHS is complex while RHS is simple. The elicitation mechanism used for treating the RHS in the case of simple policy types continues to be applicable. However, for each complex predicate on the LHS, elicitation strategies have to be formed. For each of the standard predicates, these are given in the last column of Table 3.5.

Example I: Provide provident fund to all nurses with a salary of Rs. 15000.

$$\forall x\ [\text{nurse}(x)\ \text{AND}\ \text{GEQ}(\text{salary}(x), 15000) \rightarrow \text{provide}\ (x, \text{ProvidentFund})]$$

The following range variables are defined:

<nurse> <*n*>

<ProvidentFund> <*pf*>

Table 3.5: Elicitation strategies for given LHS predicate

S.No	Predicate	Elicitation Strategy
1	EQ(F(*x*),c)	If F(*x*) is equal to c then elicit operation to make RHS true
2	LEQ(F(*x*), c)	If F(*x*) is less than or equal to c then elicit operation to

		make RHS true
3	LT(F(*x*),c)	If F(*x*) is less than c then elicit operation to make RHS true
4	GT(F(*x*), c)	If F(*x*) is greater than c then elicit operation to make RHS true
5	GEQ(F(*x*), c)	If F(*x*) is greater than or equal to c then elicit operation to make RHS true
6	NEQ(F(*x*),c)	If F(*x*) is not equal to c then elicit operation to make RHS true

<u>Guideline I</u>

When action 'recruit nurse' is taken and salary is fixed at Rs.16000. Row number 5 of Table 3.5 is applicable which says that action is to be elicited to make RHS true. Let this action be 'include in PF list'. The policy enforcement rule is:

- WHEN recruit *n* IF !provide (*n*, *pf*) THEN Allot *pf*

<u>Guideline II</u>

Suppose PF is stopped for some employee. A suitable correcting action is to be determined.

1. Elicit action to falsify the LHS
    a. Fire nurse
    b. Transfer nurse
    c. Lower Salary

The policy enforcement rules are:

- WHEN stop *pf* IF nurse(*n*) THEN Fire *n*

- WHEN stop *pf* IF nurse(*n*) THEN Transfer *n*
- WHEN stop *pf* IF housekeeper(*hk*) THEN Lower Salary

Complex CC Type Policy

This can be looked at as a combination of CS and SC types of policies. Tables 3.3 and 3.5 apply. Consider the following example.

Example: The total number of Wooden or Fibre Panchakarma Tables must be 2.

$\exists \mathrm{wtabSet} \exists \mathrm{ftabSet}\ (\mathrm{woodTable(wtabset)\ AND\ fibreTable(ftabset)} \rightarrow$

$\mathrm{EQ(Sum(count(wtabSet), count(ftabSet)),2))}$

where,

wtabSet: is a set of wooden tables

ftabSet: is a set of fibre tables

Range variables are:

<Wooden Table> <*wt*>

<Fibre Table> <*ft*>

<u>Guideline I</u>

When a new table of wood or fibre is bought then sum of wooden and fibre tables may not be equal to 2. Since this a complex predicate row 9 of Table 3.3 is applicable.

1. Elicit action to reduce the sum
    a. Discard fibre table
    b. Discard wooden table

2. Elicit action to increase the sum
    a. Purchase wooden table
    b. Purchase fibre table

So the enforcement rules are:

- WHEN purchase *wt* IF !EQ(Sum((count(wtabSet),count(ftabSet)),2)) Then Discard *wt*
- WHEN purchase *ft* IF !EQ(Sum((count(wtabSet), count(ftabSet)),2)) Then Discard *ft*
- WHEN purchase *wt* IF ! EQ(Sum((count(wtabSet), count(ftabSet)),2)) Then Purchase *wt*
- WHEN purchase *ft* IF ! EQ(Sum((count(wtabSet), count(ftabSet)),2)) Then Purchase *ft*

<u>Guideline II</u>

Let there be an elicited action that causes the sum to be unequal to 2. Then action must be elicited to make the LHS false

Elicit action to falsify the LHS

a. Discard wooden table
b. Stop purchasing wooden table
c. Discard fibre table
d. Stop purchasing fibre table

The enforcement rules are:

- WHEN add *wt* IF woodTable(*wt*) THEN Discard *wt*
- WHEN add *wt* IF woodTable(*wt*) THEN Stop Purchasing *wt*

- WHEN add *wt* IF fibreTable(*ft*) THEN Discard *ft*
- WHEN add *wt* IF fibreTable(*ft*) THEN Stop Purchasing *ft*

3.4. Early Information SubStage

The requirements engineer needs information in order to select one from the choice set. The three generic Information elicitation techniques of Chapter 2 are applied to elicit the needed information.

1. **CSFI Analysis:** Recall, CSFI analysis is a three step process of (a) determining CSF, (b) determining information and (c) determining properties of information to Actions. To illustrate, consider the action “Re-designate x” where x is an instance of AYUSH hospital. One CSF is to provide patient satisfaction. To assess this factor information needed is total yearly count of patients. Notice, the decision requirement is <Re-designate x, number of patients>. Applying the information model, Patient becomes Entity. With the help of this information the requirements engineer is able to decide whether it is worthwhile to take the action “Re-designate x”.

2. **ENDSI Analysis:** Ends analysis is a three step process involving determining a) Ends, b) Effectiveness of the End, and c) Information to evaluate the effectiveness of the End. This process continues till information for all the Ends has been determined. Revisit the action “Re-designate x”, x is an instance of AYUSH hospital. The objective or result of this action can be to Maximize economic return. The effectiveness of this end can be assessed by Revenue generated and information needed for the assessment may be cost per lab test, number of tests, service fees of nurses, consultancy fees of doctors. Applying the information

model, Lab Test, Doctor, Nurse become entities with service fees and consultancy fees as attributes for nurse and doctor entity respectively.

3. **MEANSI Analysis**: Means analysis is a three step process of a) determining means, b) determining efficiency measures for the means, and c) determining information to measure the efficiency of the means. Again consider the action "Re-designate x", x is an instance of AYUSH hospital. One means to perform this action is to select another speciality. Efficiency is expertise needed. Entity is Doctor, Patient, Disease, Nurses and Equipment. Early information needed is about number of patients with specialized disease, equipment needed, number of doctors having qualification among others. Disease is to be maintained type wise.

The results of performing steps 1 to 3 above are summarized in Table 3.6. In Table 3.6, CSFI, ENDSI and MEANSI analysis are performed for the second policy enforcement rule action re-designate x. The Table has three columns, the first for the action from which information is being elicited, the second column for the three information elicitation techniques being applied to the action, and the last column describes the information base. Recall that once CSFI, ENDSI and MEANSI for an action have been identified, relevant information to measure the CSF, effectiveness of the end and efficiency of the means is needed. Thus, for each measure, entity, attribute are identified as part of the Information base. Additional information is captured as History, Category and Function.

Table 3.6: Early information for action 'Re-designate *x*'; *x* is an instance of AYUSH hospital

Action	**Elicitation Method**		**Information Base**				
			Entity	***Attribute***	***History***	***Category***	***Function***
Re-	**CSFI**	Patient Satisfaction	Patient		Yearly		Count

designate	**ENDSI**	*Ends*	*Effectiveness*	Lab test	Cost			Sum
x		Maximize economic return	Revenue generated	Doctor	Salary			Sum
					Consultancy fee			Sum
				Nurse	Salary			Sum
					Service fee			Sum
	MEANSI	*Means*	*Efficiency*	Hospital	Specialty		Type	Count
		Select another specialty	Expertise needed	Patients	Name			Count
				Disease	Specialty		Type wise	Count
				Doctors	Qualification			Count
				Nurses				Count
				Equipment				Count
		Become general hospital	Expertise needed	Patients	Name		Type	Count
				Disease	Name		Type wise	Count
				Equipment				Count

3.5. Late Information SubStage

'Early' information of the form shown in Table 3.6 needs to be converted into a more structured form. For this, a two step process to transform early information into an ER diagram is proposed. The steps are:

- Building individual ER diagrams for each policy enforcement rule
- Integrating individual ER diagrams into a consolidated diagram.

Now that the integrated ER diagram is obtained, a star schema can be constructed by applying existing techniques. Golfarelli's algorithm is applied for star schema formation.

Individual ER Diagrams

The following two steps are applied to build individual ER diagrams

(1). All nouns of actions in the policy enforcement rule are identified and treated as entity sets of the ER diagram. This gives us an initial set.

(2). The initial set is augmented with entities and attributes obtained as part of the information elicitation process for the participating actions

(3). The requirements engineer defines relationships between entities.

The following policy enforcement rule is used illustrate the building of individual ER diagrams:

<Ayurvedic hospital> <*x*>

<OPD> <*y*>

WHEN create *x* IF !Run(*x*, *y*) THEN start *y*

Let us now apply the foregoing 3 steps to this policy enforcement rule. Applying (1) there are two actions, create and start. The participating nouns are Ayurvedic Hospital and OPD respectively. Therefore, we get two entities corresponding to these. Now, let us apply the second step. The information elicited for create and start is shown in Table 3.7. The column Entity under the major column Information Base shows the entities elicited. These become entities of the ER schema by step (2) above. Now, in the third step, the requirements engineer defines relationships, through stakeholdes interaction, between these entity types. The resulting ER schema is shown in Fig. 3.2.

Table 3.7: Information elicitation for WHEN create x, IF !Run(x, y) THEN start y

Action	**Elicitation Method**		**Information Base**				
			Entity	***Attribute***	***History***	***Category***	***Function***
Create *x*	**CSFI**	Provide Quality Care	Patient	Income	Yearly		Count

	ENDSI	*Ends*	*Effectiveness*					
		Provide treatment to patients	Facilities provided	Equipment Laboratory	Name		Ward	Count Count
	MEANSI	*Means*	*Efficiency*					
		Construct new	Resources needed	Ayurvedic hospital	Build cost Space			Sum Sum
		Hire existing	Resources saved	Ayurvedic hospital	Rental cost			Sum
Start *y*	**CSFI**	Provide Quality Care		Disease Doctor Patient	Name Speciality Degree	Month Yearly	Type	 Count Count
	ENDSI	*Ends*	*Effectiveness*					
		Treat patients using Ayurveda		Disease Patient	Name	Month Yearly	Type	Count Count
	MEANSI	*Means*	*Efficiency*					
		Construct new	Land required Infrastructure needed Personnel needed	OPD Furniture Equipment Doctor Attendant Staff	Build cost, build time, Space Name Name		Type	Count Count Count Count Count

Entity OPD is got from (1) and as information from MEANSI analysis. It is assumed that they are the same entity OPD. For action, create *x*, entity is Patient by CSFI analysis, Equipment, Laboratory by ENDSI and Ayurvedic Hospital by MEANSI analysis. The ER diagram is shown in Fig 3.2.

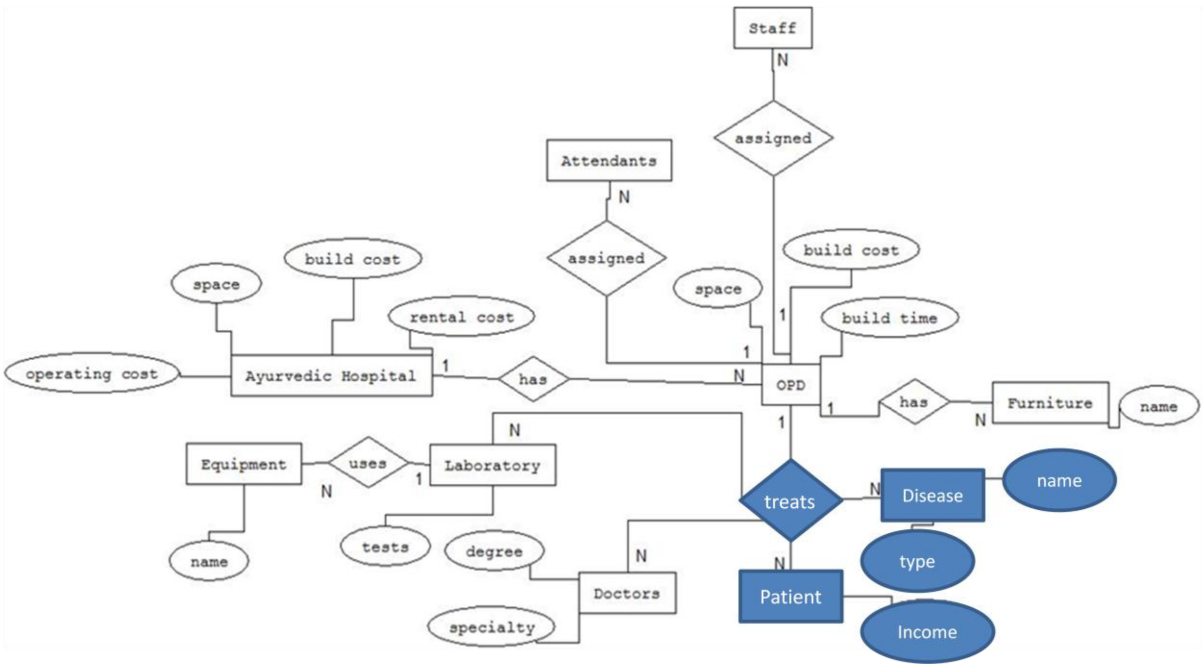


Fig 3.2: ER diagram for the policy enforcement rule WHEN create x IF !Run(x, y) THEN start y

Integrating ER Diagrams

The view integration technique (Batini, 1986) can be classified into two main streams: syntactic (Bernstein, 1976), (Raver, 1977), (Casanova, 1983), (Biskup, 1986) and semantic (Batini, 1984), (Navathe, 1986).

The syntactic approach employs functional dependencies of different database thereby obviating the need for a full understanding of the database. Its disadvantages are that it is NP hard and its inability to differentiate between dependencies over the same attributes but having different meanings.

The semantic approach, adopted in this thesis, uses the meanings of the elements in database views to perform view integration. Since the semantic approach operates at the entity and relationship level, and not at the attribute level, the technique is less complicated. The resultant global schema is also more "natural" and understandable to users and designers. The disadvantage of the semantic approach is that it requires more users' and designers' interactions to interpret and analyze conflicts.

During integration common entities of individual ER diagrams are combined and are based on the recognition that attributes of an entity is the collection of all attributes for the entity found in individual ERD. Similarly, the set of relationships between a pair of entities is union of the relationships for these entities found across ER diagrams.

To illustrate consider two policy enforcement rules:

- <Ayurvedic hospital> <*x*>

  <OPD> <*y*>

  WHEN create *x* IF !Run(*x*, *y*) THEN start *y*
- <private room> <*pr*>

  WHEN create *pr* IF LT(area(*pr*),200) THEN Expand *pr*

Early information for PER WHEN create *pr* IF LT(area(*pr*),200) THEN Expand *pr* is in Table 3.8 and the individual ER diagram is shown in Fig 3.3. Notice, both triggering and correcting actions gives us the same entity private room.

Table 3.8: Information elicitation for WHEN create *pr* IF LT(area(*pr*),200) THEN expand *pr*

Action	**Elicitation Method**			**Information Base**				
				Entity	***Attribute***	***History***	***Category***	***Function***
Create *pr*	**CSFI**							
	ENDSI	*Ends*	*Effectiveness*	Patient	Income			Count
		Revenue Generated	Service provided					
	MEANSI	*Means*	*Efficiency*	Private Room	Build time Build cost Space			Sum Sum Sum
		Construct new	Resources needed					

		Hire existing	Resources saved	Private Room	Rental cost			Sum
Expan d *pr*	**CSFI**							
	ENDSI	*Ends*	*Effectivenes s*	Disease Patient	Name Income	Month	Type	Count Count
		Service more patients	Revenue generated					
	MEANSI	*Means*	*Efficiency*	Private Room	Labour cost Space			Sum
		Remodel room	Resources needed					

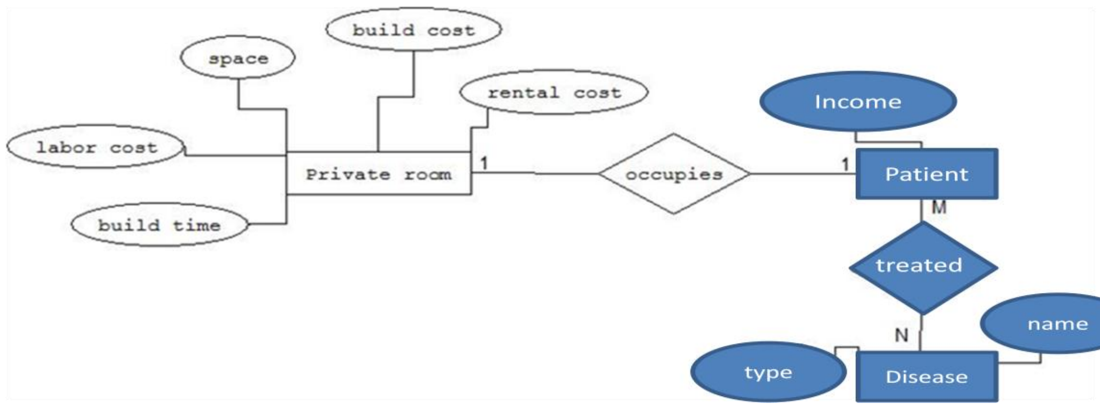


Fig 3.3: ER diagram for the policy enforcement rule WHEN create *pr* IF LT(area(*pr*),200) THEN Expand *pr*

The integrated diagram is shown in Fig 3.4. Notice the relationship ‘treated’ and ‘treats’ are assumed to be the same relationship ‘treats’.

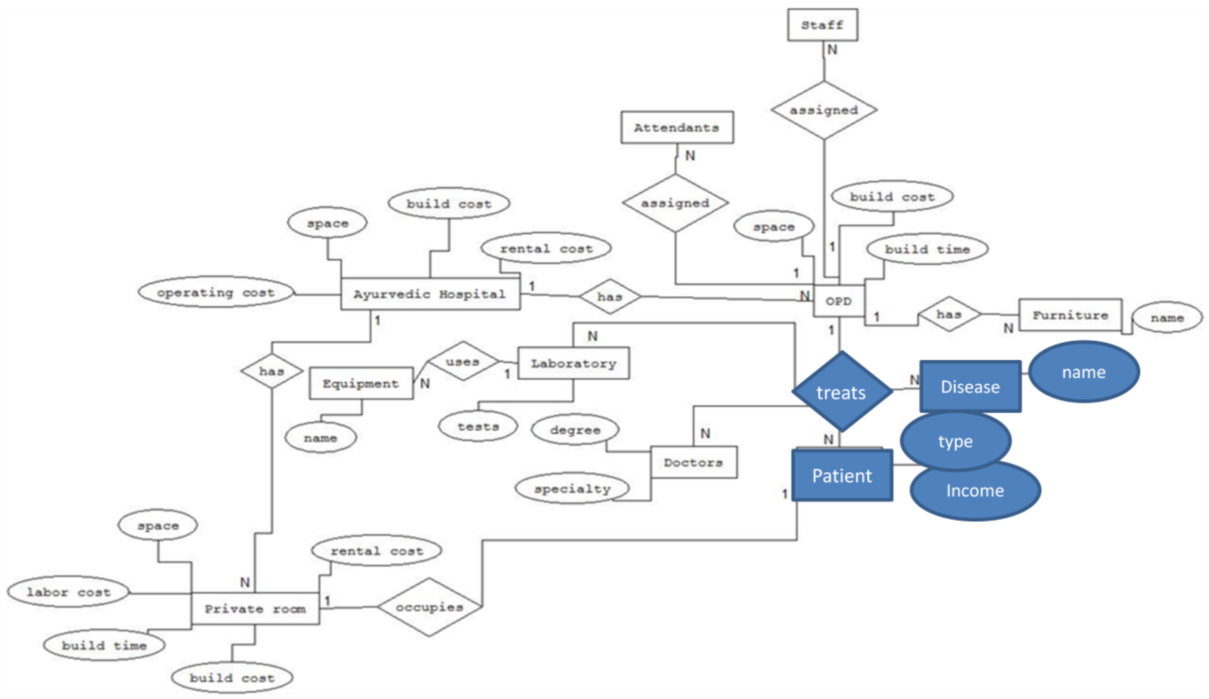


Fig 3.4: Integrated ER diagram

3.6. Conceptual Design Substage

Once the integrated ER schema is ready, the next step is to convert it to a star schema. As stated earlier this can be done by applying existing techniques. This section applies Golfarelli's (Golfarelli *et al*., 1998) algorithm.

A brief overview of the algorithm is as follows. It is a semi-automated methodology to build a conceptual model from existing ER schemas. The conceptual schema is referred to as a fact schema with facts, dimensions and hierarchies. Five steps are described to convert ER schema into fact schema. A fact forms the root of the schema which is identified first. An entity or an n-ary relationship that are frequently updated can become facts. Those entities or relationships that represent structural properties or are static do not become facts. For each fact identified an attribute tree is built. A recursive algorithm has been defined for this.

Attributes directly connected to the entity or relationship identified as fact, are added as children to the attribute tree. Similarly relationships that are many to one are also added as children. Once the process is complete and the complete attribute tree is built, the tree is *pruned* and *grafted*. Pruning involves dropping a non-relevant subtree. If a vertex, v, does not represent useful information while its descendants, v', do, then the descendants are preserved by moving the subtree, v', from v to the vertex preceding v. Thus, grafting is achieved. Finally, dimensions, fact attributes and hierarchies are defined.

According to Golfarelli's algorithm outlined above, any frequently updated entity or n-ary relationship can be treated as a fact. Since the relationship, treats, (see Fig. 3.4) is one such in a hospital environment, we select it to be a fact named 'Treatment' in Fig. 3.5. Following the algorithm, an attribute tree is built that contains the

1. Entities participating in 1: N relationships with treat. From Fig. 3.4 these are Patient, Disease, OPD, Doctor, and Laboratory.
2. Attributes of the entities are added to the tree, and

Now, after applying pruning and grafting we get the attributes of Treatment from (1) above as shown in Fig. 3.5. Further, with an unpruned attribute tree, the five entities from (2) form the dimensions for the fact Treatment as shown in Fig. 3.5. However, the requirements engineer finds that analysis by patient, disease, doctor and OPD are useful but analysis by laboratory is not. This is because, a laboratory conducts tests only and therefore laboratory-test wise treatment does not carry meaning. Laboratory is therefore pruned from the tree to yield the star schema of Fig. 3.6

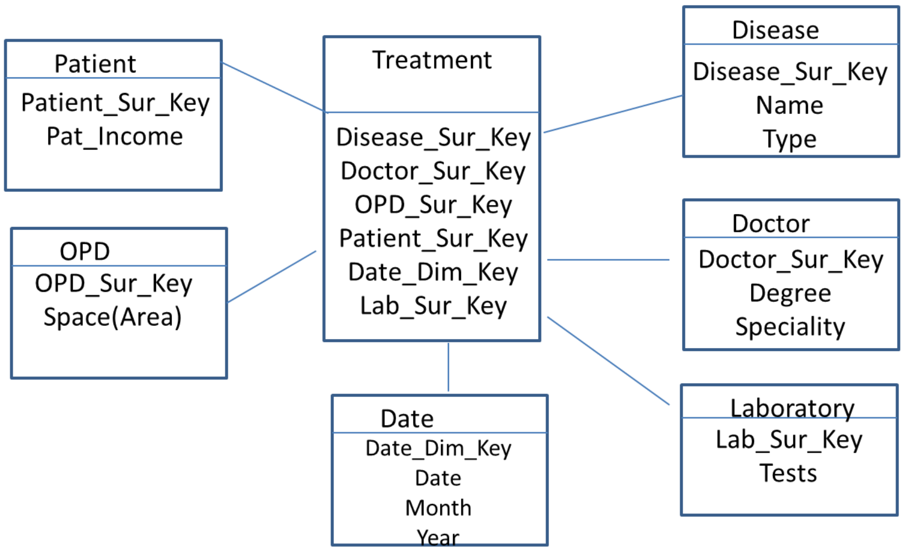


Fig 3.5: Unpruned Star Schema for Fact Treatment

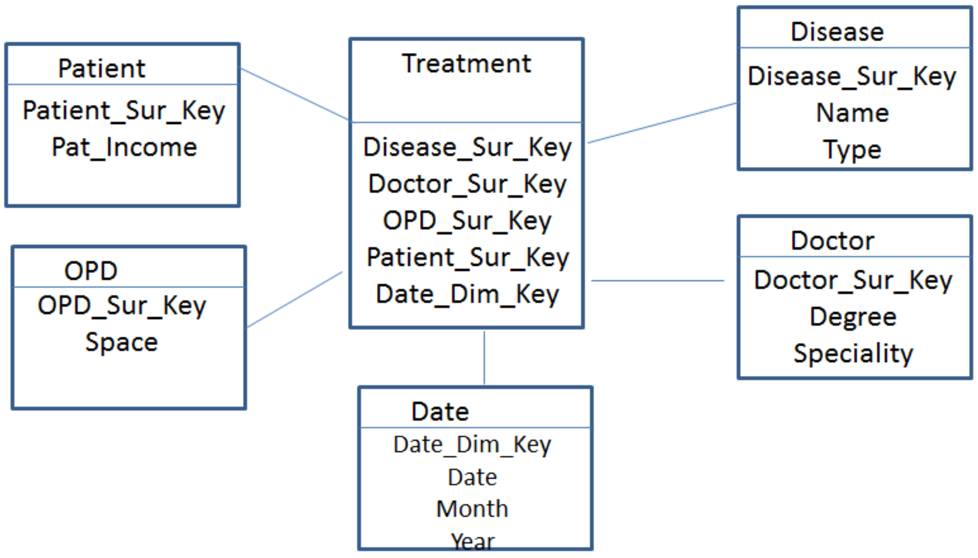


Fig 3.6: Star Schema for Fact Treatment

Summary

This chapter has shown that the basic framework for decision, information, decision requirement and associated elicitation techniques present in Chapter 2 can be followed in PER formulation. In doing so, this chapter defines a representation system for a policy.

Thereafter, guidelines for conversion of policies into possible enforcement rules are provided. The decisional problem is that of selecting the most appropriate rule for the business at hand. The chapter has also shown the application of the elicitation techniques presented in Chapter 2 to PER formulation. The early information thus obtained was converted into an ER diagram which, in turn, was converted into star schemas using Golfarelli's algorithm.

Note: The ideas in this chapter have been published in International Journal of Information System Modeling and Design (IJISMD) 2015.

Chapter 4

Operational Life Cycle

Chapter 3 of the thesis proposed a life cycle for PER decision layer of the continuum. Continuum implies that once PERs have been formulated, operational level actions have to be defined as part of the imperative decision layer. Since the models and techniques proposed in Chapter 2 are generic and are independent to the level of decision-making, these are applied to operational decisions.

As mentioned in Chapter 1, determination of the set of decisions needed to manage operational business processes has been done in a number of ways, for example by using goal modelling and business indicators respectively. However, goal analysis and business indicator analysis does not guarantee that the set of decisions that these techniques yield includes the set of decisions yielded by the PER layer. In other words, if the formulated rules have to be enforced in a business then enforcement decisions need to be taken. These decisions must be from choice sets that include the actions that participate in the rules. Therefore, information relevant to these must also be subjected to the information elicitation process.

This chapter starts off by considering how operational decision-making is related to the formulated policy enforcement rules. Thereafter, the underlying assumptions of what constitute operational decisions are articulated and the broad approach to eliciting relevant information for the operational level is outlined. This discussion lays the basis for the detailed explanation of the operational life cycle that follows.

The purpose of the operational life cycle is to support decision making for business operations by arriving at DWop. The input to this life cycle is actions of policy enforcement rules. In a manner similar to the PER life cycle, operational life cycle uses a multi-step process to discover early information in the RE stage. This early information is converted to ER diagram which can be used to arrive at the star schema for DWop.

For the first part, a two-step process is proposed. For the latter, existing techniques are applied to convert ER schema into star schema. The use of early information as an input to integrating the Data Warehouse for formulating policy enforcement rules and that fro operational decision-making will be discussed later in chapter 5.

4.1. Decision making at operational level

When a policy enforcement rule has been formulated, then the set of correcting actions to be taken in the organization is known. At the operational level, decision maker decides on which action is to be implemented in a given situation to produce the best results. In a given situation, more than one action may be applicable. Let this set be {action1, action2...} and thus the choice set formed is {select action1, select action2…….}. In other words, this is a set of alternatives available to the operational decision maker to address a given situation.

In the metadata of DWop newly discovered actions are linked to the parent action's PER. So for action *start 2-bed-pw* the PER will be that of *start pw.* This becomes relevant during the integration life cycle considered in the next Chapter, as there is a decision-rule correspondence that forms the basis of integration.

4.2. PER Actions to Decisions

Recall, decision model of Chapter 2 shows three kinds of decisions, atomic, abstract and complex. Actions of PER layer can be atomic, abstract and complex. Atomic actions are those that cannot be decomposed further, abstract actions follow IS/A specialization principles, and complex actions are composed of other actions which can be atomic, abstract or complex.

For example, action *start pw*, is an abstract decision with *start 2-bed-pw* and *start 3-bed-pw* in a specialization relationship with *start pw* (see Fig 4.1 below)

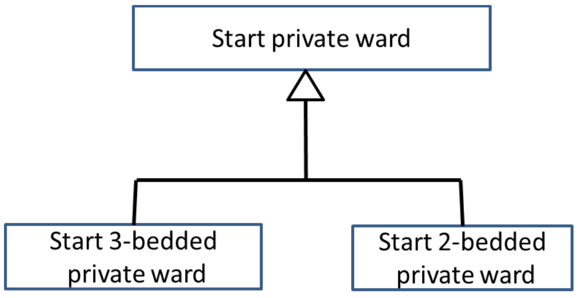


Fig 4.1: Specialization for action start private ward

start pw is also a complex decision with components choose department, choose location (see Fig 4.2 below). These atomic actions form Decisions of Table 4.1.

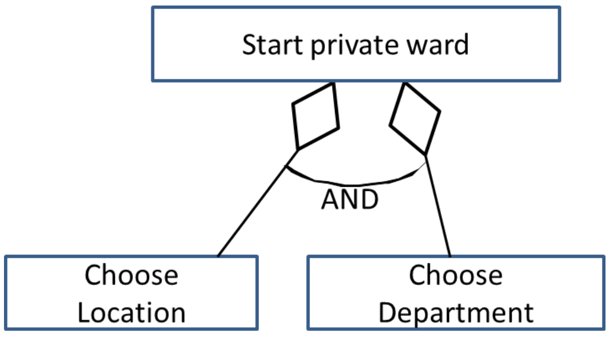


Fig 4.2: Decomposition tree for action start private ward

Now, the remaining question is, for what kind of action does information need to be elicited at the operational level? Recall that complex decision and ISA decision structures are hierarchies with atomic decisions as leaves. Intermediate nodes of such hierarchies, while describing the system at intermediate levels of abstraction, shall only contribute to the determining of the atomic actions. This is in accordance with Means-Ends analysis technique used to do decision reduction and already discussed in earlier chapters. Thus, the assumption made here is that business operations shall be carried out at the lowest levels of these hierarchies, i.e. for atomic decisions only.

Following from this the guidelines adopted in this thesis are:

1. If PER action is atomic, directly elicit information using CSFI, ENDSI and MEANSI techniques. For example, action *expand pr* is an atomic action.
2. If PER action is abstract, construct the IS/A hierarchy and arrive at atomic actions. Elicit information for the atomic actions using CSFI, ENDSI and MEANSI techniques. For our action *start pw,* the specialization tree is constructed, like the one in Fig 4.1. Information is elicited for actions *start pw, start 2 bedded pw, start 3 bedded pw.*
3. If PER action is complex, construct the AND/OR tree to arrive at atomic actions. Elicit information for the atomic actions using CSFI, ENDSI and MEANSI techniques. Again, consider action *start pw.*. The AND/OR decomposition tree is constructed as in Fig 4.2 and information is elicited for actions *start pw, choose location, choose department.*

Notice, information elicitation is for atomic actions that are arrived at from abstract and complex actions and from those directly input as atomic actions. As stated in chapter 2, decision maker at the PER layer is different from that at the operational level. At the PER layer, higher level of management is involved in the decision making process and at the operational level, lower level of management is involved. This means that CSF, ENDS and their effectiveness measures, as well as MEANS and their efficiency measures will vary as one moves from one level of the continuum to another. For the same decision (action), therefore, different early information is obtained at the two levels. This is why; early information is elicited in this level even for those actions that are input as atomic actions. This point is illustrated in section 4.4.

4.3 Operational Life Cycle

Table 4.1 shows Operational life cycle as having three main stages namely, Requirements Engineering, Conceptual Design and Logical Design stages. The Requirements Engineering stage itself consists of Decision Formulation Substage, Early information Substage and Late information Substage. In the conceptual design stage, the ER schema produced by the Requirements Engineering stage is converted into multi-dimensional schemas. Finally, in the Logical Design stage the schemas of Data Warehouses are produced.

The three main stages of Table 4.1 are now considered in detail.

Table 4.1: Operational Life Cycle

Stage		**Input**	**Output**
Requirements Engineering	Decision formulation Substage	PER Actions	Decisions
	Early Information Substage	Decisions	Early information
	Late Information Substage	Early information	ER Schema

Conceptual Design	ER schema	Multi-dimensional model
Logical Design	Multi-dimensional model	MOLAP/ROLAP

Requirements Engineering: Table 4.1 shows that the Requirements Engineering phase is divided into three Substages, Decision Formulation Substage, Early information and Late information Substage.

The basis of the Requirement Engineering stage considered is that the source of actions is the PER layer. This illustrated in Fig. 4.3. As seen, the actions of each rule are extracted and each action, a, of the PER layer is treated as a decision, d, to be taken at the operational level. This yields the initial set of decisions. Each decision in this set is then subjected to the AND/OR decomposition process and to the generalization-specialization process, see Fig. 4.3. This gives rise to decomposition and ISA hierarchies of decisions. For each hierarchy thus obtained, the leaf nodes are determined. These leaf nodes are atomic and thus, the set of all decisions at the operational level is obtained. The next task is to elicit information for every decision. For this, the techniques of CSFI, ENDSI, and MEANSI are deployed. The reasoning behind why only leaf nodes are considered for information elicitation is explained later.

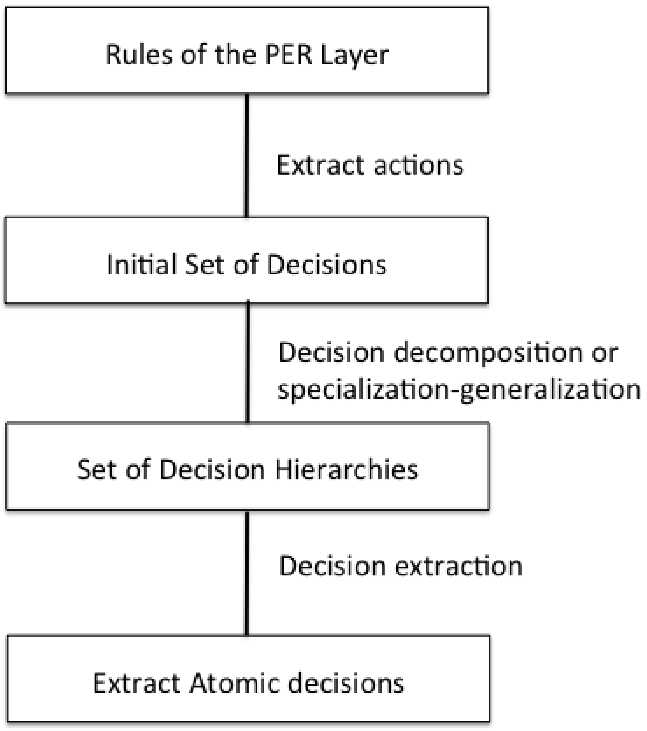


Fig. 4.3: The Overall process

The Decision Formulation Substage is complete when the set of atomic decision is determined. Attention now turns to eliciting information relevant to these. Each of these decisions is now input into the Early information Substage and Early Information EI_{op} is elicited using the generic information elicitation techniques of Chapter 2.

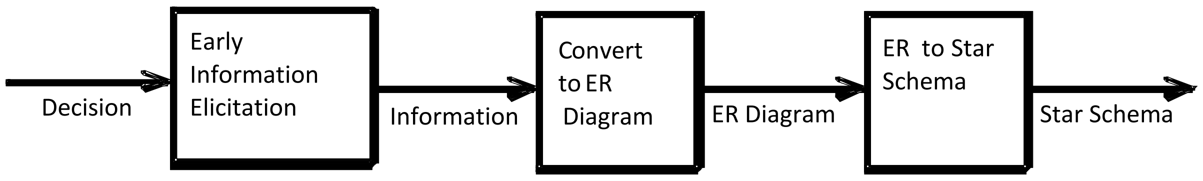


Fig 4.4: Overall process of arriving at star schema from Decisions

Since EI_{op} is early and unstructured, it is converted into an ER diagram. This overall process is shown below in Fig 4.4.

Conceptual Design: Once ER diagram is obtained, it is converted into a star schema. As already mentioned, techniques like those of Golfarelli and Moody exist for this purpose. However, conversion from ER to star schema is done using Golfarelli's algorithm.

The Early Information Substage is described below.

4.4. Early Information Elicitation

Three early information techniques, **CSFI Analysis, ENDSI Analysis and MEANSI Analysis** are applied to decisions obtained in the Action formulation Substage.

To illustrate the difference between the information elicited due to difference in the level of the decision maker, consider an action "Use Existing private ward". ENDSI and MEANSI done by the operational level decision maker and PER level decision maker are shown in Table 4.2 and 4.3 respectively.

Table 4.2: Information elicitation for expand *pr* at the PER level

Action	**Elicitation Method**			**Information Base**				
				Entity	***Attribute***	***History***	***Category***	***Function***
Expand *pr*	**CSFI**							
	ENDS I	*Ends*	*Effectiveness*					
		Service more patients	Revenue generated	Patient	Income			Count
				Disease	Name	Month	Type wise	
				Nurse				Count

	MEANSI	*Means*	*Efficiency*					
		Remodel room	Resources Needed	Private room	Space, Labour cost			

In Tables 4.2, MEANS at the PER level is 'Remodel room' with efficiency measure as resources needed while in Table 4.3 MEANS are 'Construct barrier', 'Break barrier'. Notice, MEANS is more operational in Table 4.3 than the PER. Thus, early information elicited is different in both the cases. Now, consider ENDSI analysis. The END for both the tables is the same, 'Service more patients'. However, the effectiveness measure is different with 'revenue generated' at the PER level and 'capacity of patients' at the operational level. This gives us different early information.

Table 4.3: Information elicitation for expand *pr* at the operational level

Action	**Elicitation Method**			**Information Base**				
				Entity	***Attribute***	***History***	***Category***	***Function***
Expand *pr*	**CSFI**							
	ENDS I	*Ends*	*Effectiveness*					
		Service more patients	Capacity of patients	Patients		Daily		Count
	ME	*Means*	*Efficiency*					

		Construct extension Break barrier	Cost needed Cost needed	Private room	Space Construction cost per sq. foot Breaking cost per sq. foot			Sum

4.5. Late Information Elicitation

As with PER early information, early information of Tables 4.2 and 4.3, need to be converted into a more structured form. Using a two-step process this early information is transformed into an ER diagram.

- Building individual ER diagrams for each action
- Integrating individual ER diagrams into a consolidated diagram.

<u>Individual ER Diagrams</u>

The following two steps are applied to build individual ER diagrams

(1). Identify noun part of actions as entity sets thus, obtaining an initial set. For action expand pr, pr represents noun private room. Thus, private room is an entity.

(2). The initial set is augmented with entity and attributes obtained from the information elicitation process. Again consider action expand pr. Table 4.3 shows the elicitation process gives entities Patient and Private room. For entity patient, no attributes were elicited. For Private room attributes elicited are space, Construction cost per sq. foot and Breaking cost per sq. foot. Notice that private room was also obtained during the previous step. It is assumed that these two entities refer to the same entity private room.

(3). The requirements engineer defines relationships between entities. For entities Patient and Private room relationship “occupies” is defined with cardinality of one patient occupying one private room.

Integrating ER Diagrams

Common entities of individual ER diagrams are combined. Attributes of an entity is the collection of all attributes for the entity found in individual ERD. Similarly, the set of relationships between a pair of entities is union of the relationships for these entities found across ER diagrams.

Now that the integrated ER diagram is obtained, a star schema can be constructed by applying existing techniques. Golfarelli’s algorithm is applied for star schema formation.

Summary

This chapter considers decision-making for managing business processes not just from the perspective of what is to be achieved (goal modelling) or from that of what is to be measured (business indicators). Rather, it says that the aspect of decision-making to ensure compliance with policy enforcement rules needs to be taken into account. Thus, the set of decisions needed to manage operational business processes is the union of the decisions from goal modelling, business indicator determination and policy enforcement.

The decisional problem is common to all the three kinds of decisions. Faced with a situation requiring decision-making, the decision-maker is to select the most appropriate action to be taken in a given situation.

This chapter shows that the basic framework presented in Chapter 2 can be followed at the operational decisional level. The application of decision model, information model, elicitation techniques and decision requirement model led to

1. Discovery of new decisions
2. Discovery of new early information

The work reported here throws up one interesting open problem. This problem pertains to the sources of operational decisions. The problem is to determine if all three kinds of analysis, goal, business indicator, and our policy compliance analysis need to be carried out to determine the set of decisions. Our reasons for believing that this is an open problem are as follows.

In the context of this thesis, interest is in showing the manner in which requirements engineering for upstream data warehousing could be done and how upstream and operational data warehouses could be integrated together. During this, we found that PERs lead us to operational decisions and we investigated the associated requirements engineering issue that lies here. However, the problem of this thesis is not to establish a relationship between the set of decisions derived from the various analysis techniques reported in the literature. Thus, we believe, this is an open problem and outside the scope of this thesis.

Note: The ideas of this chapter have been published in Information Systems Evolution (2010). Springer.

Chapter 5

Integrating the Warehouses

This chapter first considers the problem of integration. Thereafter, it goes on to show the two problems of inconsistency and loss of business control that occur. Finally, it proposes to move the integration problem upstream to integration of Data Warehouse requirements.

5.1 The Integration Problem

The possibility of Data Warehouse/mart integration arises when there are several Data Warehouses/data marts. According to the CMP white paper (CMP), organizations have reported between 30 and 100 data marts coexisting after a period of time. These have been developed using the approach of Kimball (Kimball and Ross, 2002) that envisages a bus architecture of data marts. Thus, all these data marts exist to solve specific subject oriented decision issues that arise in managing business operations.

Multiple data marts may exist in a number of other situations (Cabibbo and Torlone, 2004):

- when organizations merge or acquire others: Each organization participating in the merger/acquisition has built its own Data Warehouse/data mart for managing its operations. The consolidated organization resulting from merger/acquisition needs to unify these to manage the integrated operations
- when there is a requirement to combine proprietary with public Data Warehouses: again, the public and proprietary Data Warehouses are to be integrated.

Multiplicity of data marts leads to inconsistent results. There are two major reasons for this, schema and data differences. These differences result in users of the two data marts to work with data that is inconsistent with one another. Consequently, managers may end up taking different decisions because they see different data marts.

The Data Warehouse community has proposed data mart integration for unifying data marts into one consolidated Data Warehouse. As a result, different schemas are integrated and differences in data are sorted out. This leads to better decision making in **the operations** of the organization.

So far two Data Warehouses, one for PER formulation and the other for business operations, have been developed. As for the case of data marts, it is possible that there are schema and data differences between these. As for data marts, PER decision makers and managers of business operations see the same data as different or as having different logical properties. Consequently, they may take different decisions because they see different data. This calls for integration of these Data Warehouses.

Notice that this integration is needed, not for managing business operations, but to ensure compatibility between the PER and business operations level. The traditional data mart integration that facilitates decision-making at the same level of business operations is referred to hereinafter as horizontal integration. In contrast, when integration spans across decision-making levels, PER and business operations, it is referred to hereinafter as vertical integration.

Vertical integration is to be contrasted with horizontal integration. The latter refers to integrating at the same level. Consider the operational level (Fig. 5.1) consisting of data marts for supporting operational decision making. Since all the data marts are at the operational level, their integration is referred to as horizontal integration. The upper rectangle in the figure shows data marts for Policy Enforcement Rules, PER, developed. Again, there is a horizontal integration problem of these.

Since policy enforcement rules regulate operations, PER Data Warehouses support decision making at a 'higher' level than operational Data Warehouses. Again, if there is redundant information between the marts/warehouses of the two levels, then need for integrating these will arise. This integration across levels is vertical integration.

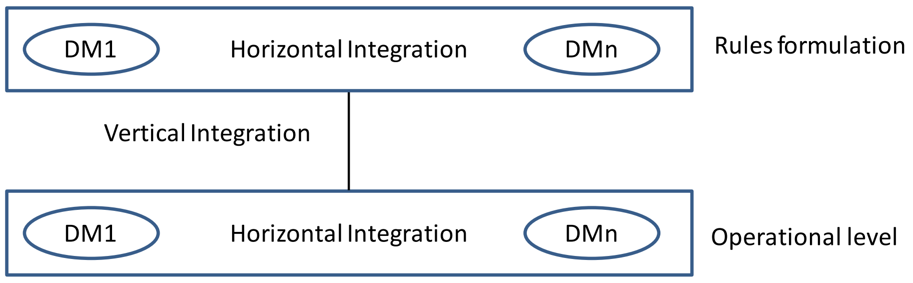


Fig 5.1: Horizontal and Vertical integration

The problem of vertical integration arises because of the continuum between PER and business operations. This continuum says that (a) the two Data Warehouses operate in the same decision environment and (b) there is a relationship between them: the former provides the rationale for the latter. The subsequent sections of this chapter show that inconsistency in these Data Warehouses leads to a special result: loss of business control in the organization. It is therefore essential, in order to retain business control, to integrate the two kinds of Data Warehouses.

5.1.1 The Integration Dichotomy in Databases

Experience with solving the **integration problem in databases** shows that there is a schema-data dichotomy. According to (Dayal and Hwang, 1984), there are two issues with integration in multi database systems, differences between the schemas of the several databases and differences between the data in the data bases. Data differences in the case of overlapping data require special attention. Specifically, the authors argue, differences between overlapping data may be due to obsolescence of data in one database, because the same data is really about two different entities of the real world, or because the data are homonyms. The authors give the example of employee data in two databases. In one database it is empid = 1, salary = 25 and in the other it is empid =1, salary = 35. Applying the argument, this may be because the first database contains obsolete information; the reference in the two databases to empid = 1 actually is to two different employees; salary are homonyms and come from two different jobs that empid=1 does. The total salary is the sum of the salaries.

Dawyndt (Dawyndt *et al.*, 2005) also pointed out the dichotomy between schema and data integration in the integration problem. This work was done in the context of microbial data systems. The authors point out that for database integration to be successful, one requires (a) development of common schemas so as to obtain a single logical point of access and (b) also data integration so as to remove duplications and other inconsistencies. Specifically, the problem of microbial database integration addresses was that of taking joins across databases by establishing equivalence between identifiers.

To sum up, inconsistencies in multiple database systems may arise due to inconsistencies in the logical schemas, for example, naming conflicts, data type conflicts or in the data itself, for example, duplication of data, implicit differences in units etc.

There are two solutions to database integration that are available. The first approach is to do logical integration but not physical integration. This has been done in Multibase but as pointed out by Dayal (Dayal and Hwang, 1984), this solution assumes that the local databases are disjoint. As a result, the assumption is that there are no data differences between the databases being integrated. The second approach is to do both logical and physical integration. This assumes that there are both logical and data differences in the databases.

There are two interesting implications of the database experience:

1. A point is reached where inconsistencies in multi database systems become so complex that the organization needs to consider building a Data Warehouse.

   Inmon (Inmon, 2005), says that as the number of databases proliferate in an organization a number of issues with data arise. First there may be independent databases from which other databases may be populated. Thus, there may not be a common source of data at all. As databases proliferate, it may get difficult to identify sources of data. Further, it may not be possible to exercise control on people entering data. Additionally, there may be an algorithm differential or time differential between the data in the different databases.

   Reed *et al* (Reed *et al*, 2010) encountered the inconsistency problem in managing domestic violence information that was present in four autonomous databases of different sources, They found that the databases contained inconsistent information that made management of domestic violence difficult. They go on to integrate the four databases into one data mart, and eliminating redundancy and inconsistency through the ETL process.
2. Just as with databases, one can expect that as Data Warehouses/data marts proliferate, inconsistencies arise and these may be due to logical or data differences. As before, if the data in the warehouses/marts is disjoint then only logical integration may be enough. However, when both differences exist then schema as well as data integration are necessary.

   In this regard, the white paper of CMP (CMP) says that the effective method to do data mart integration is to create a single physical and logical data model for the enterprise. This is done in a two-step process, first by bringing the logical data model on a common centralized system and then doing integration at both the data and schema levels.

5.1.2 Integrating Data Marts

In considering DW integration, name conflicts are assumed to be resolved following schema integration approaches. Since, multi-dimensional modelling basically deals with facts and dimensions, efforts have been made to match facts and dimensions of fact schema when performing DW integration:

- It was demonstrated (Cabibbo and Torlone, 2004) that drill across operations performed over non-conforming dimensions and facts are either not possible or produce invalid results. They assumed that data marts are available in a variety of forms, DB2, Oracle, SQL server, etc. and proposed an integration strategy of three steps consisting of a (a) semi-automated part (Cabibbo *et al.*, 2006) to identify dimension compatibility (b) verification of compatible dimensions and (c) making incompatible dimensions compatible. Thus, the integration problem is a semantic issue.
- The approach of (Riazati *et al.*, 2010) is based on the observation that in many practical situations, the assumption that in aggregation hierarchies, levels and their inter-relationships are given does not hold. They infer these levels and inter-relationships from their data and use them for integration.
- Golfarelli (Golfarelli *et al.*, 2011) position fact/dimension conformity in the larger context of the functional and physical architecture of the integrated DW and resolution of the trade-off between technical and user priorities.
- In ORE (Jovanovic *et al.*, 2014), information requirements of the integrated DW are

determined as a matrix of facts and dimensions. Each fact row is considered to be an information requirement and is to be realized in a single data mart. Thus, one gets as many data marts as the number of fact rows in the matrix. This collection of data marts is then integrated into the full DW by fact matching, dimension matching, exploring new multidimensional design and final integration.

The authors propose to use an ontology for available data sources to identify relationship between concepts.

Summary

The foregoing shows that a proliferation of independently built data marts leads to the problem of data mart integration. The requirement is to logically and physically unify the several data marts into one.

The underlying assumptions behind work on data mart integration (Cabibbo *et al*, 2004) are:

a) Data marts are structured in a uniform way; they use notions of facts and dimensions only
b) Data quality in a data mart is usually higher than in a database because of the ETL process

Therefore, the interesting issue is to integrate facts and dimensions across data marts for the purpose of providing a single logical schema for querying.

5.2 Need for PER-Operations DW Integration

Now consider the two Data Warehouses for policy enforcement rules and business operations. If the two are independently developed then the problem of inconsistency arises. As before, this arises because of schema and data differences. Clearly, the same problems as for multiple data marts arise, there is no single point of query due to schema differences and there are data differences. As a result, PER formulators see different data from business operations people and rules may be formulated that are not in consonance with operational realities. Alternatively, operations people may just ignore the formulated rules.

Assume for the moment that there are no schema and data differences between the two Data Warehouses. As shown below, inconsistency may still arise and may result in loss of business control.

1. Inconsistent data: The effect of inconsistent data is seen when the refresh times of the Data Warehouses are different. Let the PER Data Warehouse, DW_1 and the operations Data Warehouse, DW_2. Let these contain overlapping information I and to distinguish between I in DW_1 and I in DW_2, I_1 and I_2 shall be used respectively. Let it be that at ETL time of DW_1 and DW_2, $I_1 = I_2 = I$. Now, let the refresh times of DW_1 and DW_2 be T_1 and T_2 respectively. Then, at T_1, I_1 changes to I'_1 whereas it is only at T_2 that I_2 changes to I'_2 to make $I'_1 = I'_2$. Thus, in the time interval $(T_2 - T_1)$ the two warehouses show inconsistent values of I.

   This leads to data obtained from one warehouse to be different from that obtained in the other in the window $(T_2 - T_1)$. Thus, rule formulators and operational decision makers respectively end up taking decisions on different data in this temporal window. The larger this window, the longer this inconsistency exists.

   That this only happens in the window makes the inconsistency problem worse than if it were to happen uniformly. It is not possible to get to the source of the problem once the second Data Warehouse has been refreshed. Thus, the problem is unrepeatable outside this window and may go completely undetected or, if detected, the reason for the inconsistency may be difficult to find after the window is past.

There is no guarantee that the two Data Warehouses shall be refreshed at the same time. Further, the operational level Data Warehouse is likely to be refreshed more frequently than that dealing with formulation of policy enforcement rules. This is because operational decision-making occurs more frequently than decision making for rules. Therefore, given operational pressures, operational Data Warehouse is more likely to be kept refreshed than the rules formulation Data Warehouse.

2. Loss of business control: Inconsistency of data in PER and Operations Data Warehouses results in loss of business control. Consider the window ($T_2 – T_1$). Three cases arise
    a. The operations Data Warehouse, DW_2 is refreshed and contains latest data. This data is **not** in accordance with the policy enforcement rules governing it. Evidently, either business operations are not in conformity with the enforcement rules and there is need for better control on the operational business or the business environment has changed and enforcement rules need to be reformulated.
    b. As before, DW_2 is refreshed and contains the latest data. This data **is in** accordance with policy enforcement rules. Evidently, this is a benign case compared to (a): even though the data is different there is no conflict between PER and operations people.
    c. The operations Data Warehouse, DW_1 is refreshed and contains latest data. Therefore, PER formulators are ahead of business operations in so far as formulating rules are concerned. They may formulate new rules that

operations people believe are not needed. Again, this is not a serious problem and will sort itself out once the operations Data Warehouse is refreshed.

As mentioned above, the operations Data Warehouse is more likely to be refreshed than the PER Data Warehouse, making case (a) a live possibility. To make case (a) precise, loss of business control occurs when data of an operations DW calls for decision makers of the policy enforcement DW to take decisions, but the decisions are not taken because data in the latter do not suggest this need.

The two problems are illustrated using the following example. Let there be a policy "bed: nurse ratio must be 8:1". Consider a Unit in a hospital consisting of 400 beds. According to the policy, the number of nurses must be 50.

The policy enforcement rule for this policy says that if the ratio falls below 8:1 then an alert to recruit new nurses is sent out; if it is greater, then an alert about excess staffing is sent out. Operational information keeps track of the number of nurses, beds, patients registered, discharged etc.

Inconsistent Data

Let us examine the effect of differing refresh times and show the existence of inconsistency. Consider Table 5.1 that shows transactions and DW refreshing being carried out at different times.

Table 5.1: Transactions and refresh times for DWop and DWper

Time	**Transaction**	**Dwop**	**DWper**
T1	Delete nurse$_1$		
T2	Delete nurse$_2$		
T3	Delete nurse$_3$		
Tk		Refresh	
To	Add nurse4		
Tn			Refresh

Before the first transaction is executed, let the number of nurses in the hospital be 50. The first three rows of Table 5.1 show the deletion of three nurses from the transactional system. At Tk, the operational Data Warehouse, DWop, is refreshed and the number of nurses in this warehouse is 47. Between Tk and Tn, DWper continues to show that there are 50 nurses. At this moment the Data Warehouses are inconsistent.

Similarly, the table shows that there is inconsistency again at Tn. At To, a new nurse is added. At Tn, DWper is refreshed and it contains 48 nurses; DWop contains 47.

Loss of Business Control

Loss of business control occurs in the presence of Data Warehouse inconsistency. Recall that loss of business control occurs when DWop may ask for a change of policy enforcement rule but DWper may show no such need.

Table 5.2: Data at time T and t’ for DWop and DWper

Time	DWop	DWper
T	Number of nurses = 50	Number of nurses = 50
t’	Number of nurses= 35	Number of nurses = 50

Consider Table 5.2. DWop and DWper at time T show that the hospital has the required number of nurses that is 50. At time t', DWop shows 35 nurses whereas DWper continues to show 50.

Consider a possible reaction by decision makers charged with formulating rules, under the assumption that DWper shows 50 instead of 35. Clearly, there is failure to recruit enough new nurses as required by the rule. Therefore a new rule is required “if number of nurses is less than 80% of the required number then transfer nurses from other units”.

Table 5.2 shows that DWop is suggesting the need for a change in a policy enforcement rule but DWper does not. Thus, loss of business control occurs since appropriate decision making is inhibited.

5.3 The Approach to Integration

Having shown the problems that occur due to separate Data Warehouses, this section develops an approach to building a unified Data Warehouse. This approach has two salient features, (a) it is the ‘build by integrating’ approach and (b) it minimizes integration cost wherever possible. Each of these two aspects is considered below.

5.3.1 Build by Integrating

The approach of Kimball to Data Warehouse development, (see Fig. 1.4) encourages development of independent data marts. When inconsistencies arise, then integration is to be done. This can be said to be **detection and correction** approach: when a problem is detected it is fixed. Clearly, there is considerable cost in building one single, unified logical and physical Data Warehouse.

The alternative approach is that of Inmon (see Fig. 1.4) which proposes development of a full Data Warehouse from which data marts are derived. Since there is a common Data Warehouse, the inconsistency problem does not arise. In contrast to the approach of Kimball, this is the inconsistency **prevention** approach. However, this is a heavy weight approach with long lead times to deliver working systems.

So, the question is whether there is an approach that, while not being heavy weight is also relatively cheap? One way of reducing cost is to do integration earlier in the Data Warehouse life cycle. That is, the approach of Kimball is followed but independent data mart development is done only until the requirements engineering stage. These are now integrated to form the requirements of the new Data Warehouse which are then taken into logical and physical development.

The foregoing can be done pair wise. If there is indeed an operational data mart, then its requirements specification is obtained and integrated with the independently developed requirements specification of the new data mart. It can be seen that the cost to physically build the second data mart is eliminated.

That is, (see Fig. 5.2) when an organization starts on a new data mart DM2, then its requirements specification, RDM2 is integrated with the requirements specification, RDM1, of the existing DM1. The integrated requirements specification, RDW1, is then taken through the design and implementation stages of the development life cycle to yield a physically and logically unified Data Warehouse, DW1. Similarly, when the new data mart DM3 is to be developed, then its requirements specification RDM3 is integrated with RDW1 to yield RDW2 and DW2 is built.

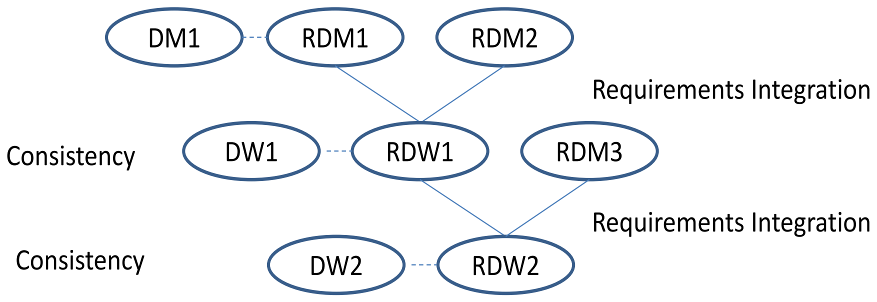


Fig. 5.2: Pair-wise integration at the requirements engineering stage

The underlying development principle of the approach is to 'build by integrating'. Whenever, a new DM is to be developed, the developer first looks for an existing DW/DM. Initially, the first DW/DM is logically and physically built since no other DW is found. When the second DM is to be developed, its requirements specification is integrated with that of the first, and the new integrated. Logical and physical DW is obtained. This process of integrating the new DM with an existing DW continues.

Thus, at no time does the approach call for development of intermediate DMs before a final DM is built. The build by integration approach can thus be seen to do pair-wise integration.

5.3.2 Integrate Early Information

The build by integrating approach moves integration to the requirements engineering phase thereby reducing wasted effort. In other words, this principle of reducing wasted effort can be applied to integration at the requirements stage itself. In the development life cycle of chapters 3 and 4, there are TWO possibilities for integration in the RE stage:

(a) integrate ER schemas. Existing ER integration techniques can be used to produce the integrated ER schema (Batini *et al*, 1986; Lee and Ling, 1995; Lee and Ling, 2003). This can then be converted into multidimensional form using existing algorithms of Golfarelli or Moody.

(b) integrate early information. Convert integrated early information into the ER form for subsequent development of the multidimensional view.

Shown below is a brief discussion on each approach. Notice that (b) is the most appropriate approach.

Consider two data marts, DM1 and DM2. Let DM1 be already operational. When constructing DM2 using strategy (a), determine the ER schema of DM2, obtain the ER schema of DM1, integrate these and produce the integrated warehouse. Existing ER schema integration techniques and algorithms for converting ER schemas to multidimensional form can be exploited. This is shown in Fig 5.3.

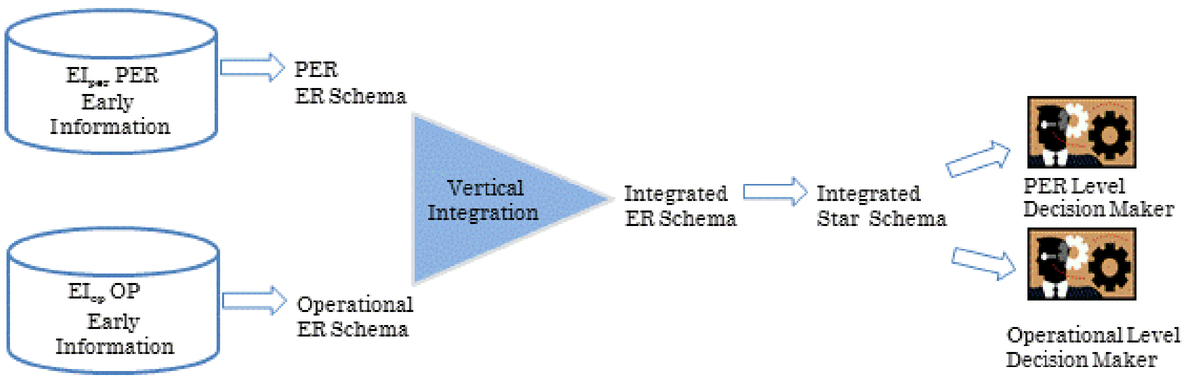


Fig 5.3: Strategy (a) integrating at the ER schema point

Using strategy (b) there is further reduction in the work to be done since even the ER schema is not to be produced. The integrated early information is then converted into the integrated

ER diagram for subsequent conversion into the integrated multi dimensional schema of the DW To-Be.

The proposal here is to take the early information obtained in the two life cycles, EI_{per} from DWper and EI_{op} from DWop, and produce integrated early information $EI_{integrated}$ from them. Thereafter, $EI_{integrated}$ is converted into the integrated ER diagram for subsequent generation of multi dimensional schema. This process is shown in Fig 5.4.

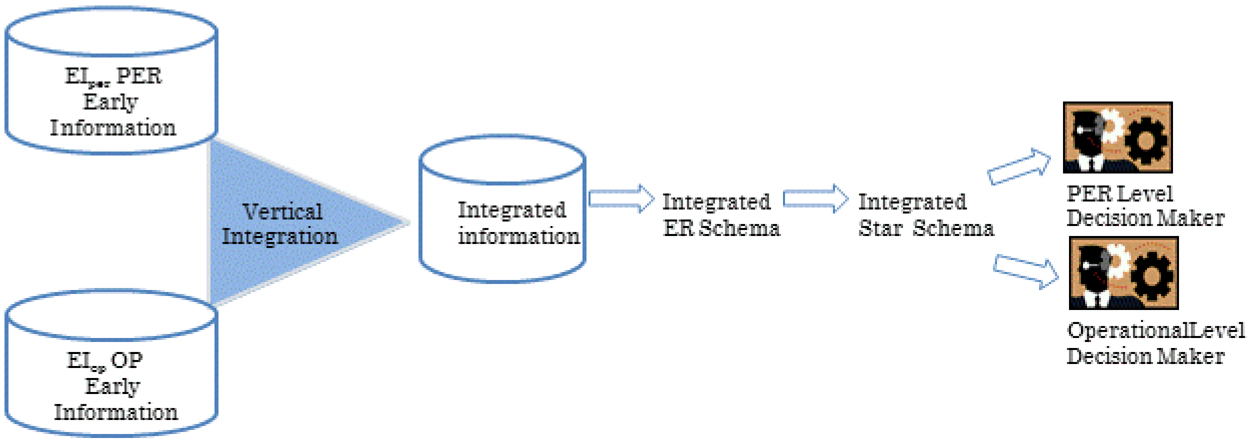


Fig 5.4: Strategy (b) integrating Early Information

Notice that whereas prevailing data marts integration approaches work with integration of facts and dimensions, it is possible, in this case to separate early information from late information. The former is unstructured information whereas the latter consists of facts and dimension. The proposal is to do integration of early information. The integrated early information is then converted into the integrated ER diagram for subsequent conversion into the integrated multi dimensional schema of the DW To-Be.

5.4 The Vertical Integration Life Cycle

The foregoing is applied in developing an integrated PER/Operations Data Warehouse.

The integration technique is organized in four steps. This section details the integration process using these steps.

5.4.1. Overview of four Components

The four broad steps of the integration approach itself are as follows:

1. Reading requirements specifications of the Data Warehouses. It is assumed that these are available as part of the metadata. The **MetaData Reader** reads these specifications.
2. Proposing correspondence between the requirements specifications. Four correspondence strategies are defined here from brute force to strong correspondence. The **Correspondence Drafter** proposes correspondences.
3. The proposed correspondences are examined to carry out detailed integration of information. This task is carried out by the **Information Mapper**.
4. When the Information Mapper throws up situations of conflict, then these are resolved by the **Conflict Resolver**.

5.4.2. MetaData Reader

The MetaData Reader assumes that a trace of the information obtained during requirements engineering is available. Metadata of DWper has:

- Rule: PE rule identifier for which EI has been elicited;
- Analysis Type: analysis method used for eliciting EI, ENDSI, MEANSI, CSFI;
- Analysis value: effectiveness measure for ENDSI, efficiency measure for MEANSI and CSFI factor for CSFI analysis; and finally
- Early Information: EI identifier.

The metadata for DWper is illustrated in Table 5.3. It can be seen that analysis of R1 using CSFI analysis yielded $EI_{R1,CSFI,A}$, $EI_{R1,CSFI,B}$ and using ENDSI analysis yielded $EI_{R1,ENDSI,C}$ whereas analysis of R2 yielded $EI_{R2,CSFI, D}$.

Table 5.3: Trace Information of PE Rules

Rule	**Analysis Type**	**Analysis Value**	**Early Information**
R1	CSFI	A	$EI_{R1,CSFI,A}$
R1	CSFI	B	$EI_{R1,CSFI,B}$
R1	ENDSI	C	$EI_{R1,ENDSI,C}$
R2	CSFI	D	$EI_{R2,CSFI, D}$

The structure of metadata in DWop is similar to the structure of metadata of DWper. Since decisions are traceable to rules there is an additional column called Decision which is the decision identifier. An example is shown in Table 5.4. Here two decisions D1 and D2 are traceable to R1 and R2 respectively. Again, the information obtained for each decision under different analysis types is available.

Table 5.4: Trace Information of Decisions

Decision	**Rule**	**Analysis Type**	**Analysis Value**	**Early Information**
D1	R1	CSFI	P	$EI_{D1,CSFI,P}$
D1	R1	ENDSI	Q	$EI_{D1,ENDSI,Q}$
D2	R2	CSFI	R	$EI_{D2,CSFI,R}$

The MetaData reader reads the metadata from the two Data Warehouses for use by the correspondence drafter.

5.4.3. Correspondence Drafter

The correspondence drafter can be based on a number of strategies, from the brute force strategy to strong correspondence strategy. These strategies are described below:

1. **Brute Force strategy**: In this strategy each row of one table is compared with every other row of the second table. Assuming the tables to have m and n rows respectively, the total number of comparisons will be m*n. For low values of m and n this strategy is suitable.

2. **Weak correspondence strategy (WCS)**: As the values of m and n of the brute force strategy become large, there is a need to deploy heuristics. Let there be a rule R; a decision D; and early information EI_R and EI_D. WCS says that EI_R and EI_D correspond to one another provided D is traceable to R. Thus, for Table 5.3 and 5.4, the weak correspondences shown in Table 5.5.

Table 5.5: Weak Correspondence strategy

DWper	DWop
$EI_{R1,CSFI,A}$ $EI_{R1,CSFI,B}$ $EI_{R1,ENDSI,C}$	$EI_{D1,CSFI,P}$ $EI_{D1,ENDSI,Q}$
$EI_{R2,CSFI, D}$	$EI_{D2,CSFI,R}$

Assuming that the amount of EI of a rule and that for a decision derived from it is not large, this strategy is suitable.

3. **Average correspondence strategy (ACS)**: As the early information to be considered in the WCS rises, there is need for a stronger heuristic. Average correspondence suggests that early information that has been obtained from the same analysis type for a rule and that obtained for decisions traceable to this rule is likely to exhibit correspondence. Formally, let there be a rule R; a decision D; analysis types AT1 as well as AT2; and early information $EI_{R,AT1}$ and $EI_{D,AT2}$. ACS says that $EI_{R,AT1}$ and $EI_{D,AT2}$ correspond to one another provided (i) D is traceable to R, and (ii) AT1 = AT2. Thus, for Table 5.3 and 5.4, the average correspondences shown in Table 5.6.

Table 5.6: Average Correspondence strategy

DWper	DWop
$EI_{R1,CSFI,A}$ $EI_{R1,CSFI,B}$	$EI_{D1,CSFI,P}$
$EI_{R1,ENDSI,C}$	$EI_{D1,ENDSI,Q}$
$EI_{R2,CSFI, D}$	$EI_{D2,CSFI,R}$

Assuming that the amount of EI of an analysis type is not very large, this strategy is suitable.

4. **Strong correspondence strategy (SCS)**: Again, as the amount of early information to be considered in ACS rises, there is need for an even stronger heuristic. Let there be a rule R; a decision D; analysis types and values AT1, V1 as well as AT2, V2 respectively; and early information $EI_{R,AT1,V1}$ and $EI_{D,AT2,V2}$. Now, a strong correspondence occurs between $EI_{R,AT1,V1}$ and $EI_{D,AT2,V2}$ provided (i) D is traceable to R (ii) AT1 = AT2 and (iii) EQV(V1,V2). EQV is defined as follows:

- For AT1 = AT2 = CSFI, EQV(V1,V2) if V1 is computed from V2 or V1=V2
- For AT1 = AT2 = ENDSI, EQV(V1,V2) if achievement of V1 contributes to achievement of V2 or V1=V2
- For AT1 = AT2 = MEANSI, EQV(V1,V2) if V1 is a MEANSI that contributes to the MEANSI used to achieve V2 or V1=V2

Assume that EQV(A,P), EQV(C,Q), and EQV(D,R). Thus, for Table 5.3 and 5.4, the strong correspondences are shown in Table 5.7. Notice that the second row shows no strong correspondence for $EI_{R1,CSFI,B.}$

Table 5.7: Strong correspondence strategy

DWper	**DWop**
$EI_{R1,CSFI,A}$	$EI_{D1,CSFI,P}$
$EI_{R1,CSFI,B}$	
$EI_{R1,ENDSI,C}$	$EI_{D1,ENDSI,Q}$
$EI_{R2,CSFI, D}$	$EI_{D2,CSFI,R}$

5.4.4. Information Mapper

Once the Correspondence Drafter reports the correspondences, attention shifts to a more detailed examination of early information. The notion of early information was elaborated in chapter 2, and it was shown there that early information is described in terms of the following

- Attribute
- History: Whether or not its history is to be maintained
- Categorization
- Functional: use of a function like Count, Max, Min etc.

To establish a mapping between correspondences generated by the Correspondence Drafter, there is a need to ensure that information of one can be mapped to that of the other. This is the job of the Information Mapper: it compares two pieces of early information, EI1 and EI2 and reports their integration, $EI_{integrated}$.

Suppose EI1 has I1, A1, H1, C1, F1 and EI2 has I2, A2, H2, C2, F2. While comparing EI1 and EI2, three possibilities can arise. EI1 and EI2 can be

- Fully mapped: This is the case when I1==I2 and A1==A2, H1==H2, C1==C2, T1==T2 and F1==F2. In this case EI1=EI2= $EI_{integrated}$. One copy of early information is included in $EI_{integrated.}$
- Partially mapped, if I1==I2 and at least one of the other properties are not equal. In this case there are conflicts that need to be examined and resolved.
- Not mapped: defined as I1<>I2. Here there is no overlap between the information and $EI_{integrated}$ = EI1 U EI2.

5.4.5. Conflict Resolver

EI which is partially mapped is sent to the Conflict Resolver. There are the following two kinds of conflicts.

1. Property present in EI1 and not present in EI2 or vice versa: When such a conflict arises then the proposed heuristic is to maintain property in $EI_{integrated}$. For example, EI1 shows that history is required and EI2 shows that it is not. Obviously, then, history in $EI_{integrated}$ has to be maintained. The requirement of DM2 shall be satisfied with current data and that of DM1 by current plus back data.
2. Property present in both EI1 and EI2 but with different property values: Table 5.8 shows the different scenarios that can arise. Notice in the case of Attribute, Categorization and Function, $EI_{integrated}$ contains A1 U A2, C1 U C2 and F1 U F2. In

the case of temporal unit, the value having the lower grain is chosen since roll-up operations can always be performed at the level of BI tools.

Table 5.8: Conflict resolution

Property	**EI1**	**EI2**	**$EI_{integrated}$**
Attribute	A1	A2	Both A1 and A2
History	H1	H2	Lower grain
Categorization	C1	C2	Both C1 and C2
Function	F1	F2	Both F1 and F2

In conclusion, Metadata Reader reads the metadata, early information, of the two data marts to be integrated. Correspondence Drafter proposes correspondences between rules and decisions traceable to their rules. This can be done using the proposed heuristics for either brute force, WCS, ACS or SCS strategies. Now, for each correspondence EI is examined by the Information Mapper to generate $EI_{integrated.}$ For this the Mapper applies heuristics to find if two given EI are fully mapped, partially mapped or not mapped. Partially mapped ones move to the conflict resolution stage.

5.4.6 An example showing Integration

The process of integration is shown for the following PER and Decision:

R1: WHEN start x IF !area(x, 200) THEN expand x

D1: Remodel x

where range variable is <private ward> <*x*>.

1. Metadata Reader

Consider the metadata for PER R1 as shown below in Table 5.9. Each row gives information about a PER, analysis type applied, analysis value obtained and EI identifier. Observe from Table 5.9 that during PER life cycle EI was elicited, using two CSFI factors, three ENDSI and three MEANSI analyses. Details of the early information in the last column of Table 5.9 are provided later when we consider Information Mapper because these details are needed then.

Table 5.9: Trace Information of PE Rules

S.No	Rule	Analysis Type	Analysis Value	Early Information
1	R1	CSFI	Patient Satisfaction	$EI_{R1,CSFI,PS}$
2	R1	CSFI	Quality Care	$EI_{R1,CSFI,QualC}$
3	R1	ENDSI	Service higher income group patients	$EI_{R1,ENDSI,IncGrp}$
4	R1	ENDSI	Service more patients	$EI_{R1,ENDSI,SPat}$
5	R1	ENDSI	Improve patient care	$EI_{R1,ENDSI,PC}$
6	R1	MEANSI	Construct new	$EI_{R1,MEANSI,NewRoom}$
7	R1	MEANSI	Hire room	$EI_{R1,MEANSI,HireRoom}$
8	R1	MEANSI	Remodel existing Room	$EI_{R1,MEANSI,RemodRoom}$

Table 5.10 shows metadata of DWop for decision D1. There is an additional column that shows the PER from which the decision is derived. In Table 5.10, second column contains the decision and third column shows the corresponding PER. Also, observe for D1, EI was

elicited using two CSFI factors, two ENDSI and four MEANSI analyses EI was elicited. Again, details of early information aare considered when dealing with Information Mapper.

Table 5.10: Trace Information of Decisions

S.No	**Decision**	**Rule**	**Analysis Type**	**Analysis Value**	**Early Information**
1	D1	R1	CSFI	Patient Satisfaction	$EI_{D1,CSFI,PatSat}$
2	D1	R1	CSFI	Quality Care	$EI_{D1,CSFI,QC}$
3	D1	R1	ENDSI	Attract higher income group patients	$EI_{D1,ENDSI,Income}$
4	D1	R1	ENDSI	Provide patient attention	$EI_{D1,ENDSI,PatAtt}$
5	D1	R1	MEANSI	Construct new	$EI_{D1,MEANSI,NewPvt}$
6	D1	R1	MEANSI	Hire existing	$EI_{D1,MEANSI,HirePvt}$
7	D1	R1	MEANSI	Splitting room	$EI_{D1,MEANSI,SplitPvt}$
8	D1	R1	MEANSI	Adding section	$EI_{D1,MEANSI,AddSec}$

2. **Correspondence Drafter**

The next step is to find correspondences between PERs and decisions for each row of Tables 5.9 and 5.10. The brute force strategy was not applied and so it is not shown here. This is because the EI to be integrated was large which made the total number of comparisons also large.

Applying WCS:

WCS says that EI_R and EI_D correspond to one another provided

(i) D is traceable to R.

Table 5.10 shows that D1 is traceable to R1. There is a weak correspondence between EI of Table 5.9 and Table 5.10. The result is shown in Table 5.11. Note, neither is the analysis type nor is the analysis value taken into consideration while drafting correspondence.

Table 5.11: Correspondence between EI_R and EI_D using WCS strategy

Dwper	**Dwop**
$EI_{R1,CSFI,PS}$,$EI_{R1,CSFI,QualC}$ $EI_{R1,ENDSI,IncGrp,}$$EI_{R1,ENDSI,Spat,}$ $EI_{R1,ENDSI,PC}$ $EI_{R1,MEANSI,NewRoom;}$ $EI_{R1,MEANSI,HireRoom;}$ $EI_{R1,MEANSI,RemodRoom}$	$EI_{D1,CSFI,PatSat,}$ $EI_{D1,CSFI,QC}$ $EI_{D1,ENDSI,Income,}$ $EI_{D1,ENDSI,PatAtt}$ $EI_{D1,MEANSI,NewPvt}$; $EI_{D1,MEANSI,HirePvt;}$ $EI_{D1,MEANSI,SplitPvt;}$ $EI_{D1,MEANSI,AddSec}$

Applying ACS:

ACS says that $EI_{R,AT1}$ and $EI_{D,AT2}$ correspond to one another provided

i. D is traceable to R, and

ii. AT1 = AT2

Consider the first and second rows of Table 5.10. Here D1 is traceable to R1. The analysis type is CSFI. In Table 5.9, row numbers 1 and 2 have rule as R1 and analysis type as CSFI. Thus, the correspondence is ACS. Similarly, row numbers 3, 4 of Table 5.10 and row numbers 3, 4, and 5 of Table 5.9 have ACS between their EIs. The result is shown in Table 5.12. Notice, the analysis value is not taken into consideration here.

Table 5.12: Correspondence between EI_R and EI_D using ACS strategy

Dwper	**Dwop**
$EI_{R1,CSFI,PS}$,$EI_{R1,CSFI,QualC}$	$EI_{D1,CSFI,PatSat,}$ $EI_{D1,CSFI,QC}$
$EI_{R1,ENDSI,IncGrp,}$$EI_{R1,ENDSI,Spat,}$	$EI_{D1,ENDSI,Income,}$ $EI_{D1,ENDSI,PatAtt}$

$EI_{R1,ENDSI,PC}$	
$EI_{R1,MEANSI,NewRoom}$, $EI_{R1,MEANSI,HireRoom}$, $EI_{R1,MEANSI,RemodRoom}$	$EI_{D1,MEANSI,NewPvt}$, $EI_{D1,MEANSI,HirePvt}$, $EI_{D1,MEANSI,SplitPvt}$, $EI_{D1,MEANSI,AddSec}$

Applying SCS:

Consider the first row of Table 5.9 and Table 5.10. Applying, the rules for SCS $EI_{R1,CSF,PS}$ and $EI_{D1,CSF,PatSat.}$

(i) Decision D1 is traceable to R1

(ii) Analysis type for both is CSFI.

(iii) $EI_{R1,CSF,PS}$ has the same analysis value "patient satisfaction" as $EI_{D1,CSFI,PatSat}$. Thus, according to the rules above, there is equivalence, EQV(PS, PatSat).

All the three conditions for strong correspondences between $EI_{R1,CSFI,PS}$ and $EI_{D1,CSFI,PatSat}$ are satisfied. Similarly for $EI_{R1,CSFI,QualC}$ and $EI_{D1,CSFI,QC}$, and for $EI_{R1,ENDSI,IncGrp}$ and $EI_{D1,ENDSI,Income}$ a strong correspondence is found and shown in the second and third rows of the table.

The fourth row of Table 5.13 shows no entry against $EI_{R1,ENDSI,SPat}$. This is because there is no equivalent analysis value found in Table 5.10.

Table 5.13: Correspondence between EI_R and EI_D using SCS strategy

S.No	Dwper	Dwop
1	$EI_{R1,CSFI,PS}$	$EI_{D1,CSFI,PatSat}$
2	$EI_{R1,CSFI,QualC}$	$EI_{D1,CSFI,QC}$
3	$EI_{R1,ENDSI,IncGrp}$	$EI_{D1,ENDSI,Income}$
4	$EI_{R1,ENDSI,Spat}$	

5	$EI_{R1,ENDSI,PC}$	$EI_{D1,ENDSI,PatAtt}$
6	$EI_{R1,MEANSI,NewRoom}$	$EI_{D1,MEANSI,NewPvt}$
7	$EI_{R1,MEANSI,HireRoom}$	$EI_{D1,MEANSI,HirePvt}$
8	$EI_{R1,MEANSI,RemodRoom}$	$EI_{D1,MEANSI,SplitPvt}$
9	$EI_{R1,MEANSI,RemodRoom}$	$EI_{D1,MEANSI,AddSec}$

To obtain the fifth row of Table 5.13, consider the fifth row of Table 5.9 and the fourth row of Table 5.10. Again, rules 1 and 2 are satisfied because D1 is traceable to R1 and analysis type is the same for both, that is ENDS. Notice that achievement of “Provide patient attention” contributes to achievement of “Improve patient care”. Thus, there is EQV(PC, PatAtt).

The last two entries of Table 5.13, rows 8 and 9, are obtained because the MEANS “splitting room” and “adding section” of Table 5.9 contributes to the MEANS “remodel room” of Table 5.10. Thus, there is EQV(RemodRoom, SplitPvt) and EQV(RemodRoom, AddSec).

3. Information Mapper

Information mapper checks to see if early information to be integrated is fully mapped, partially mapped, or not mapped.

Mapping Information from WCS:

The information mapper picks one EI from DWper and the other from DWop for integration at random. If they are fully mapped then only one set is maintained and integrated with the next EI picked at random by the information mapper. If at any point there is a conflict, then the conflict resolver resolves the conflicts and integrates EIs. If EIs are not mapped then both the copies are stored. This process is repeated till all the entries of Table 5.11 have been processed and integrated.

Mapping Information from ACS:

The first row of Table 5.12 has two entries from DWper and two from DWop. The information mapper picks one from each DWper and DWop at random, integrates and then picks the remaining two for integration. After it finishes with the first row of Table 5.12 it proceeds to the second row and follows the same process. Throughout, the rules for fully mapped, partially mapped and not mapped are followed.

Mapping Information from SCS:

Table 5.13 shows that $EI_{R1,CSFI,PS}$ and $EI_{D1,CSFI,PatSat}$ have a strong correspondence. The process of information mapping for CSFI analysis type is shown below. Consider information for $EI_{R1,CSFI,PS}$ and $EI_{D1,CSFI,PatSat}$ as shown in Table 5.14.

Table 5.14: Early information for $EI_{R1,CSFI,PS}$ and $EI_{D1,CSFI,PatSat}$

Early Information	**Information**	**Attribute**	**History**	**Category**	**Function**
$EI_{R1,CSFI,PS}$	Patient		Yearly		Count
$EI_{D1,CSFI,PatSat}$	Patient		Yearly	Unit-wise Ward-wise Department -wise	Count

Clearly, information "Patient" is mapped. Now, is the question of whether it is fully or partially mapped. Notice here that while patient of $EI_{R1,CSFI,PS}$ is not categorized, patient of

$EI_{D1,CSFI,PatSat}$ is categorized unit-wise, ward-wise and department-wise. Thus, they are partially mapped and this conflict is resolved by the conflict resolver. $EI_{integrated}$ obtained is shown in Table 5.15.

Table 5.15: Early information after integrating $EI_{R1,CSFI,PS}$ and $EI_{D1,CSFI,PatSat}$

Information	**Attribute**	**History**	**Category**	**Function**
Patient		Yearly	Unit-wise Ward-wise Department –wise	Count

Consider information for $EI_{R1,CSFI,QualC}$ and $EI_{D1,CSFI,QC}$ shown in Table 5.16. Information Disease has the same attribute, history and category and function values in both the rows. Thus, for information Disease the EI are fully mapped.

Table 5.16: Early information for $EI_{R1,CSFI,QualC}$ and $EI_{D1,CSFI,QC}$

Early Information	**Information**	**Attribute**	**History**	**Category**	**Function**
$EI_{R1,CSFI,QualC}$	Disease	Name	Monthly	Type-wise	
$EI_{D1,CSFI,QC}$	Disease Doctor Patient	Name Speciality Income	Monthly Monthly Monthly	Type-wise Daily	 Count

Doctor and Patient are unique to $EI_{R1,CSFI,J}$ and not mapped. Thus, $EI_{integrated}$ obtained is shown in Table 5.17.

Table 5.17: Early information after integrating $EI_{R1,CSFI,QualC}$ and $EI_{D1,CSFI,QC}$

Information	**Attribute**	**History**	**Category**	**Function**
Disease	Name	Monthly	Type-wise	
Doctor	Speciality	Monthly	Daily	
Patient	Income	Monthly		Count

In the next iteration, Tables 5.16 and 5.17 are integrated, conflicts resolved and the resulting $EI_{integrated}$ is shown in Table 5.18.

Table 5.18: $EI_{integrated}$ after integrating from Tables 5.15 and 5.17

Information	**Attribute**	**History**	**Category**	**Function**
Disease	Name	Monthly	Type-wise	
Doctor	Speciality	Monthly	Daily	
Patient	Income	Monthly	Unit-wise Ward-wise Department – wise	Count

This process is repeated for all the entries of Table 5.13. After $EI_{integrated}$ is obtained, this information is converted to ER diagram and then to a star schema. The integrated schema is presented in Appendix B.

Summary

In contrast to the prevailing 'fact and dimension conformity' approach, the proposal in this thesis is to integrate early information obtained in the requirements engineering stage.

Upstream integration has two advantages over the more traditional methods of data mart Integration described in Chapter 1.

1. In the latter methods, all data marts to be integrated have to first be independently developed. Thus, for each data mart a conceptual, logical and physical structure is

developed. When integration is to be performed, conformed dimensions are identified. In other words, the conceptual model is integrated.

Notice, there is considerable downstream activity effort involved in first developing the data marts and then integrating them. In the method proposed in this thesis, the new data mart is not built separately. Rather integration is done at the requirements stage itself. Thus, development effort is saved in producing the individual data marts.

2. Further, at every increment, a unified logical and physical Data Warehouse is available, which as pointed out earlier, is highly effective.

This focus on requirements integration coupled with pair-wise DW development minimizes development effort. It does so by never allowing completely operational data marts to be built before these are integrated, thereby reducing the wasted effort in throwing away DM designs and implementations. Further, this approach is a prevention approach. There is no detection and correction because the occurrence of a problem is pre-empted by not allowing an un-integrated data mart to be developed.

Note: The ideas of this chapter are in Business and Information Systems Engineering. Springer (communicated on 04/01/2015)

Chapter 6

Validation and Experience

The method proposed in this thesis was validated against a traditional Indian system of medicine which includes Ayurveda, Yoga and Naturopathy, Unani, Siddha and Homoeopathy. This system is regulated by AYUSH which is a government of India body under Ministry of Health and Family Welfare. AYUSH has departments of Ayurveda, Yoga and Naturopathy, Unani, Siddha and Homoeopathy (thus abbreviated AYUSH). Each department develops and regulates education and research in their respective areas. Further, AYUSH body regulates hospitals offering services in the above mentioned areas of medicine. It does so by defining policies that other hospitals must comply with.

AYUSH policies can be found on their website (AYUSH, 2000). Structural policies are considered here. There were a total of 151 policies across the various systems of medicine. These policies were represented in an extended first order logic form defined in Chapter 3. Using these as input into the Policy Enforcement Rule life cycle DWper was built. The PER actions formulated were input into the operational life cycle and DWop was built. Finally, using the integration life cycle, early information bases of DWper and DWop was integrated to form $EI_{integrated}$.

Three tools namely, ELISPE, ELISO and CADEI, whose implementation is described in Chapter 7, were used to elicit EI_{per}, EI_{op} and $EI_{integrated}$ respectively. Using ELISPE, the total effort required to move from policies to PER to DWper was three man months. This included one domain expert and the author of this thesis. Using ELISO, the total effort required to move from PER to operational actions and to construct DWop was two man months including the author of this thesis and the domain expert. Using CADEI, the total effort required to integrate EI_{per} and EI_{op} was 15 days. This was performed by the author of this thesis and the task was to define equivalences while drafting correspondences. At this stage only minimal interaction was required with the domain expert.

Sections 6.1 and 6.2 respectively of this chapter, consider the application of the elicitation and integration techniques. For the former, detailed examples for arriving at DWper have already been considered, in Chapter 3. Therefore, here only the statistics of the study, lessons learnt and the result of applying these learning are presented.

Section 6.2 takes up the validation and experiences gained from the integration part of the AYUSH example. First the manner in which integration can be done through the four step integration process is shown. This integration yields $EI_{integrated}$. Thereafter, experiences gained are outlined.

6.1 Lessons Learnt during Elicitation

As mentioned above, a sufficiently large example is needed to base the validation on. It was assumed that an example that dealt with 100 or more policies would bring out the main features of the approach proposed in this thesis. When the AYUSH example was looked at, a coherent set of 151 policies was found, coherent because all of these related to the infrastructural aspects of AYUSH hospital. Thus, the techniques proposed in the previous chapters and the three tools were used. The policies were expressed in our logic by the author of this thesis and validated for correctness through interaction with the domain expert. The effort required for this is included in the three man-months mentioned earlier in this chapter.

We found that all the 151 policies could be expressed in our logic. These policies and their corresponding expressions in the logic are presented in Appendix A.

As already explained in chapter 3, at the PER level, policy enforcement rules are identified following the two guidelines and each rule throws up a number of actions. A summary of the case is shown below:

Total number of policies = 151

Total number of Actions (triggering and correcting) = 1624

Total number of policies enforcement rules obtained from guideline 1 = 492

Total number of policies enforcement rules obtained from guideline 2 = 320

Total number of policy enforcement rules = 812

Now, let us consider the lessons learnt. There are two broad lessons, namely (b) the effect of common actions that might be determined on the elicitation process and (c) the applicability of the elicitation techniques. Both of these are considered in turn.

Effect of Common Actions: There were a number of situations where the same common actions were found. Identification of common actions has two effects:

- *Improvement in efficiency of the elicitation process*: Since the actions are common, the elicitation process can be done exactly once for these and does not have to be repeated several times.
- *Determination of Non-redundant early information*: Since the relevant information is the same for, early information is obtained exactly once.

The above is illustrated further. Actions can be common between two policy enforcement rules. This can happen when either triggering action of two policy enforcement rules is the same or when correcting action of two rules is the same. Observe the entries in Table 6.1 and 6.2.

Table 6.1: Information for create *x* for PER WHEN create *x*, IF !Run(*x*, *y*) THEN start *y*

Action	**Elicitation Method**			**Information Base**				
				Entity	***Attribute***	***History***	***Category***	***Function***
Create *x*	**CSFI**	Provide Quality Care		Patient		Yearly		Count
	ENDSI	*Ends*	*Effectiveness*	Bed Equipment Laboratory	Name Test		Ward	Count Count Count
		Provide treatment to patients	Facilities provided					
	MEANSI	*Means*	*Efficiency*	Ayurvedic hospital	Build cost Space			Sum Sum
		Construct new	Resources needed					
		Hire existing	Resources saved	Ayurvedic hospital	Rental cost			Sum

Table 6.1 shows information elicited for triggering action, create x, for PER *WHEN create x IF !Run(x, y) THEN start y*. Table 6.2 shows information for triggering action, create x, for PER *WHEN create x, IF !ratio(count(b), count(n),8, 1) THEN re-designate x*. The Range Variables for both of these rules are: <Ayurvedic hospital> <*x*> and <OPD> <*y*>.

Notice, action Create x is common between the two rules and so is the information elicited.

Table 6.2: Information for action *create x* for PER WHEN create *x* IF !ratio(count(*b*), count(*n*),8, 1) THEN re-designate *x*

Action	**Elicitation Method**			**Information Base**				
				Entity	***Attribute***	***History***	***Category***	***Function***
Create *x*	**CSFI**	Provide Quality Care		Patient		Yearly		Count
	ENDSI	*Ends*	*Effectiveness*	Bed Equipment Laboratory	Name Test		Ward	Count Count Count
		Provide treatment to patients	Facilities provided					
	MEANSI	*Means*	*Efficiency*	Ayurvedic hospital	Build cost Space			Sum Sum
		Construct new	Resources needed					
		Hire existing	Resources saved	Ayurvedic hospital	Rental cost			Sum

It can also happen that an action that acts as a trigger for one rule is the correcting action of another rule. For example consider the two policy enforcement rules:

1. WHEN remove *b* IF !(ratio(count(*p*),count(*b*),1,6) THEN re-designate *x*
2. WHEN start *pr* IF !EQ(count(attbed),1) THEN remove *b*

For the first policy enforcement rule, action remove *b*, where b is an instance of bed, is the triggering action. The same action is a correcting action for the second policy enforcement rule.

Irrespective of where the common action is, common actions yield same information. In the case of entries in Table 6.1 and 6.2, action, create *x*, where *x* is an instance of Ayurvedic Hospital, gives the same information. It was observed that action remove *b*, where *b* is an instance of bed, yields same information irrespective of whether it is a triggering action or correcting action. **Thus, information elicitation may be done only once**.

Identification of common actions can drastically reduce the number of actions to be considered for information elicitation. In the case of AYSUH, the following was found:

Number of common Actions = 1100

Total number of Actions= Actions-Common Actions =1624-1100=524

A**pplicability of Elicitation Techniques:** Experience here is that not all elicitation techniques are uniformly applicable to all actions:

- Means Analysis was applicable to almost all the actions. This can be seen in Chapter 3 Tables 3.6, 3.7 and 3.8. Efficiency measures and information could thus be identified.
- Almost all the Actions had an End. Actions in Tables 3.6, 3.7 and 3.8 have an End associated with them and the effectiveness of the end could be measured.
- CSF was more difficult to identify for all actions. Whereas for actions create *x* and start *y* (see Table 3.7) CSF could be defined, defining a CSF for expand *pr* was more difficult.

Whereas it seems possible to associate ENDs and MEANS with an action, the association of a CSF with an action is relatively more difficult. A fair amount of pre-work is required to be carried out that associates CSF with actions. If this pre-work is not done then elicitation becomes slow and tedious.

6.2 Lessons Learnt from Integration

Using pair-wise integration approach, the early information bases of DWper and DWop was integrated. Recall, the integration approach contains four steps (a) **Metadata Reader** that reads off the metadata from DWper and DWop, (b) **Correspondence Drafter** that finds correspondences between PER and Decision using Brute Force, WCS, ACS or SCS strategy as desired by the requirements engineer, (c) **Information Mapper** examines and checks if the pair of early information being integrated is fully, partially or not mapped. Naturally, if a conflict arises, as with partially mapped information, then the fourth component the **Conflict Resolver** resolves the conflicts.

Let us now present the main lessons learnt from the integration of DWper and DWop. These are as follows:

Time taken for drafting correspondences: In SCS, for defining EQV(V1,V2), if the case is that V1=V2, then the time taken to define equivalence is not high. No interaction with the requirements engineer is required as the correspondence drafter itself finds the equivalence. However, for the other cases namely, V1 is computed from V2; V1 contributes to achievement of V2; and V1 is a means that contributes to the means used to achieve V2; EQV has to be defined by the requirements engineer. This part is a manual process and time taking.

In ACS, since it's a direct text search, there is no intervention required by the requirements engineer and therefore the time taken to form correspondences is lower than SCS.

In WCS, again there is no intervention required by the requirements engineer and is much faster than SCS. Between ACS and WCS there is no significant time difference.

Number of times Conflict Resolver had to resolve issues: It was found that he number of times conflicts had to be resolved was the lowest while employing SCS and maximum during WCS. This was because of the random nature of picking up a pair of EI for integration. By forming correspondences in ACS and SCS, the random nature of forming an EI pair is reduced. Note, there is no randomness in SCS strategy. This meant the time taken to form $EI_{integrated}$ was maximum in WCS and minimum in SCS.

The Trade-off: There is **a trade-off between time taken by the correspondence drafter and number of conflicts to be resolved**. This is shown in Table 6.3.

Table 6.3: Trade-off between different correspondence strategies

Correspondence Strategy	Time taken for drafting correspondences	Number of times hit to Conflict Resolver
WSC	Lowest	Highest
ASC	Average	Average
SCS	Highest	Lowest

Summary

Using the policies of AYUSH regulatory authority, DWper with EI_{per} was built. The actions of PERs formulated were then used to build DWop with EI_{op}. Finally, EI_{per} and EI_{op} were integrated. The three correspondence strategies WSC, ACS and SCS were applied and using the information mapper and the conflict resolver, EI_{per} and EI_{op} was integrated to obtain $EI_{integrated}$.

Some of the experience with the example deals with the efficiency and non-redundancy of the elicitation process. Other experience is with the applicability of the different elicitation techniques. The reasons behind the partial applicability of the CSFI technique need investigation and is an open problem falling out of this thesis.

Note: The ideas in this chapter have been published in International Journal of Information System Modeling and Design (IJISMD) 2015 and in Business and Information Systems Engineering. Springer (communicated on 04/01/2015)

CHAPTER 7
Implementation

Chapter 2 of the thesis presented generic models for decision requirement, decision and information along with generic information elicitation techniques. These were applied to the PER and operational layer of the decision continuum to develop two DW systems, DWper and DWop. Early information of DWper and DWop was integrated to $EI_{integrated}$ and an integrated DW system was developed.

This chapter discusses the implementation to arrive at EI_{per}, EI_{op}, and $EI_{integrated}$. In this regard, three tools namely, ELiciting Information Support for Policy Enforcement (ELISPE) for the first, ELiciting Information Support for Operations (ELISO) for the second, and Computer Aid for Decision Early Information elicitation (CADEI) respectively were developed. They were written out in *Microsoft .net platform* with backend support provided by *Microsoft SQL Server*. The details of the code can be obtained from the author.

Chapter 7 is organized into three parts with Part I containing the architecture of ELISPE, Part II architecture of ELISO and finally, Part III for CADEI.

PART I: ELiciting Information Support for Policy Enforcement (ELISPE)

Recall from chapter 3, the input to PER life cycle is organizational policies for which PERs have to be formulated. Using guidelines of chapter 3, actions are elicited and PERs are formulated in the WHEN *triggering action* IF condition THEN *correcting action*. In order to decide which *correcting action* is to be part of a PER, one among the choice set {select, modify, reject} is to be selected. Early information required for this decision making is elicited by applying the three techniques, CSFI, ENDSI and MEANSI. Thus, three components are to be part of ELISPE, one for eliciting actions, second for formulating PER and third for eliciting early information.

Note, for the late RE phase an existing tool Dia was used for ER schema modelling.

The architecture of ELISPE is shown in Fig. 7.1. There are two parts:

- **front end** part that formulates enforcement rules and elicits Actions.
- **back end** part of the tool helps elicit information.

Organizational policies are present in the policy base of Fig. 7.1. These policies are presented, one by one, in textual form to the requirements engineer. The policy being presented is processed by the Action Elicitor. For each policy, the Action Elicitor applies the guidelines discussed in Chapter 3. The Action Elicitor stores each elicited action in the Action base.

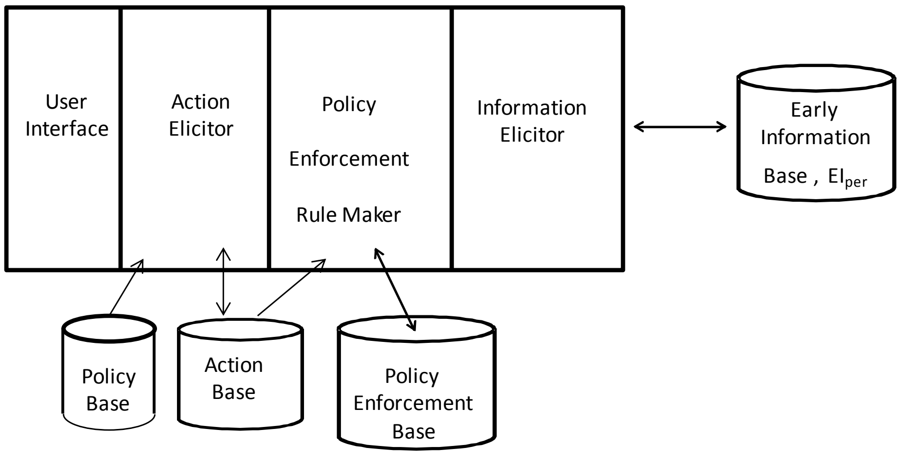


Fig 7.1: Architecture of ELISPE

These actions are used as input to the policy enforcement rule maker where actions are filled into the WHEN IF THEN form. These are then stored in the Policy enforcement base. This forms the front end part of ELISPE.

Notice there are two users of the system, the repository manager and the requirements engineer. The creation and maintenance of the four bases of Fig 7.1 is done by the repository manager. Using this, information is made available to the requirements engineer as a service. That is, if a requirements engineer needs to formulate rules then s/he registers for use of the system, accesses its contents and formulates them. Naturally, the four bases are created once. They are populated thereafter.

Detailed below is the role and responsibilities of the repository manager and the requirements engineer.

Repository Manager

The repository manager creates and maintains the Policy, PER, Action and Early Information bases. That is, when the repository is to be populated with a new domain, then it is the repository manager's task to create the partition in the repository. Now, the repository is ready to accept different organizations and their policies and rules. The repository manager is also responsible for registration and de-registration of requirements engineers. When a requirements engineer is de-registered then the repository ceases to contain information about the policies and rules for which s/he was responsible.

The repository manager interacts with the four bases through an interface where it is possible to create a new domain or to select an existing domain from the presented list. An organization can be created within the domain.

Notice that the repository manager has the responsibility to populate the repository with policies so as to start off use of ELISPE.

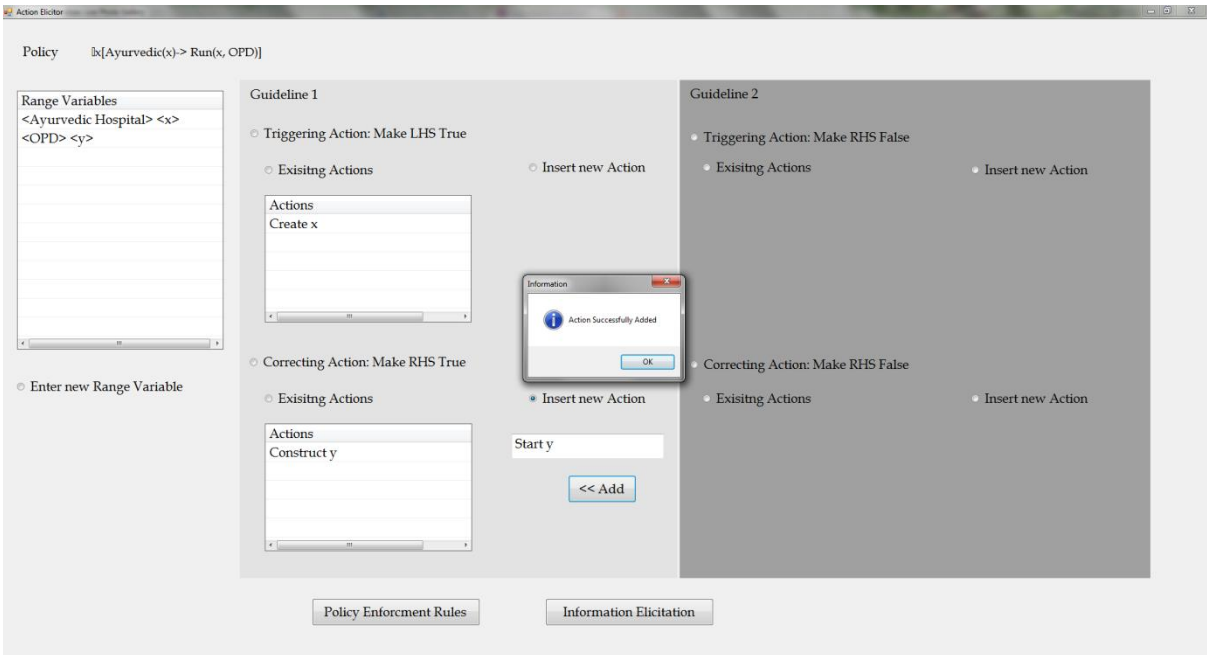
Requirements engineer

The requirements engineer's role is defined in a specific organization with the responsibility of formulating policy enforcement rules. To get access to the four bases, the requirements engineer first registers in the repository. Now it is possible to (a) create and (b) update rules. S/he browses/retrieves policies from the repository, applies guidelines, and formulates PERs of the organization. Notice that, the requirements engineer role requires familiarity with extended first order logic**.** This expertise may exist in the requirements engineer directly or it may be obtained through appropriately qualified experts.

Each component of ELISPE is discussed below.

Action Elicitor

The user interface for action elicitation is shown in Fig 7.2. The policy for which actions are being elicited is shown on the top left hand side of the screen. On the left hand side of the screen range variables already existing in the Action base are shown. A new range variable can be entered by clicking on the 'Enter new Range Variable' radio button. The centre panel shows guideline 1 being applied and right hand side shows the panel for guideline 2. In both panels triggering actions and correcting actions are elicited. For each type of action, triggering and correcting, there are two choices presented to the requirements engineer. Actions present in the Action base are viewed by selecting 'Existing actions' choice. To insert a new action, the 'Insert new Action' radio button is selected.



Policy ∀x[Ayurvedic(x)-> Run(x, OPD)]

Range Variables
<Ayurvedic Hospital> <x>
<OPD> <y>

Actions
Create x

Actions
Construct y

Start y

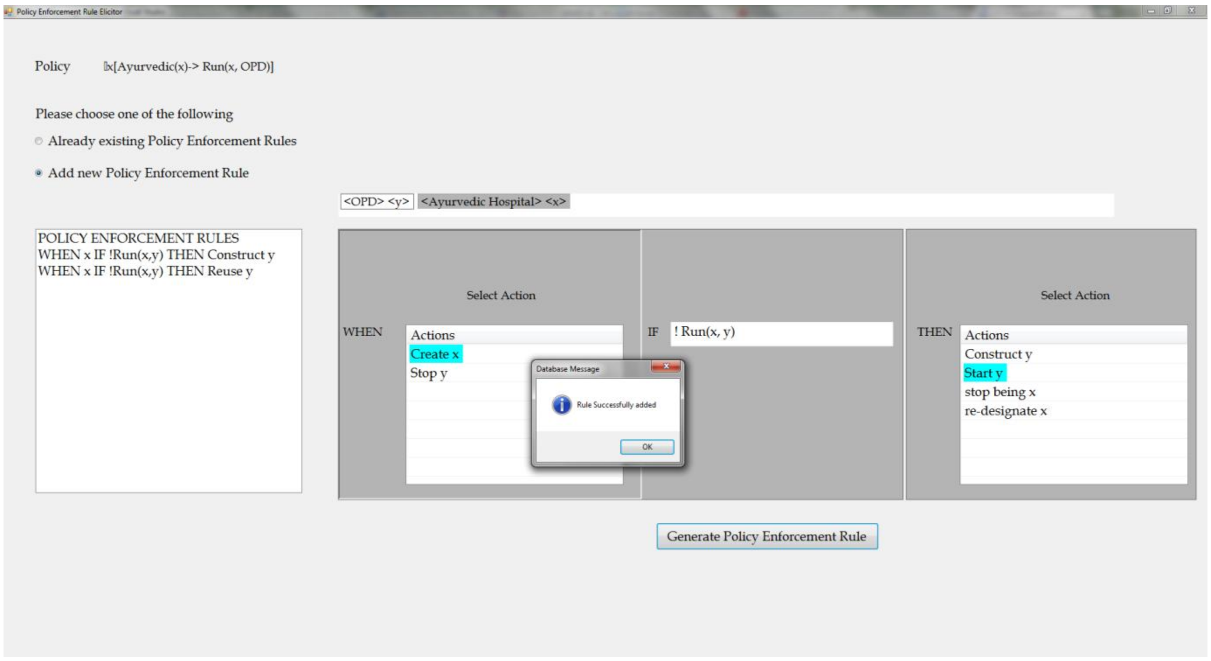
Action Successfully Added

Fig. 7.2: Eliciting Actions for policy "∀x[Ayurvedic(x)→ Run(x, OPD)]"

Fig 7.2 deals with the policy, "∀x[Ayurvedic(x)→ Run(x, OPD)]". Range variables x and y are displayed where x is an instance of Ayurvedic Hospital and y an instance of OPD. Create x is a triggering action that already exists in the Action base and is displayed. Similarly, Construct y is a correcting action that is also displayed as an existing action. When the requirements engineer selects Insert new Action, then the new action Start y is entered.

<u>Policy enforcement rule maker</u>

Once actions are elicited, policy enforcement rules are formulated in the policy enforcement rule maker. Actions in the Action base are used as input and policy enforcement rules are stored in the policy enforcement base. The user interface for policy enforcement rule maker is shown in Fig 7.3.



Policy Enforcement Rule Elicitor

Policy ▯x[Ayurvedic(x)-> Run(x, OPD)]

Please choose one of the following

○ Already existing Policy Enforcement Rules

● Add new Policy Enforcement Rule

POLICY ENFORCEMENT RULES
WHEN x IF !Run(x,y) THEN Construct y
WHEN x IF !Run(x,y) THEN Reuse y

<OPD> <y> <Ayurvedic Hospital> <x>

WHEN — Select Action

Actions
Create x
Stop y

IF: ! Run(x, y)

THEN — Select Action

Actions
Construct y
Start y
stop being x
re-designate x

Database Message: Rule Successfully added

Fig 7.3: Formulating Policy enforcement rules for policy "∀x[Ayurvedic(x)→ Run(x, OPD)]"

Again the policy for which the enforcement rules are being elicited is mentioned on the top left corner of the screen. Two options are given to the requirements engineer, to either view already existing rules in the policy enforcement rule base or to insert a new rule. When the former is selected, a list of the rules is displayed on the left hand side of the screen. The list of range variables is also displayed. For a new rule to be created, the panel on the right hand side is displayed. Actions elicited (in Fig 7.3) are presented to the requirements engineer. The actions are divided depending upon whether they play a triggering or correcting role with the triggering action on the left most sub-panel and correcting on the right most sub-panel. The middle sub-panel is where the IF condition is input. The requirements engineer selects the desired action. The selected actions are highlighted and the IF condition is keyed in. The tool constructs the rule in the WHEN IF THEN format upon clicking the Generate Policy Enforcement Rule button.

Information Elicitor

Fig 7.4 shows the back-end part of ELISPE tool. As seen the Information Elicitor of Fig 7.1 has been expanded here. Actions in the Action base are the input to the information elicitation section of the tool. Using three information elicitation techniques, Critical Success Factor (CSFI), ENDSI and MEANSI analysis, information relevant for an action is elicited. This information is stored in the early information base.

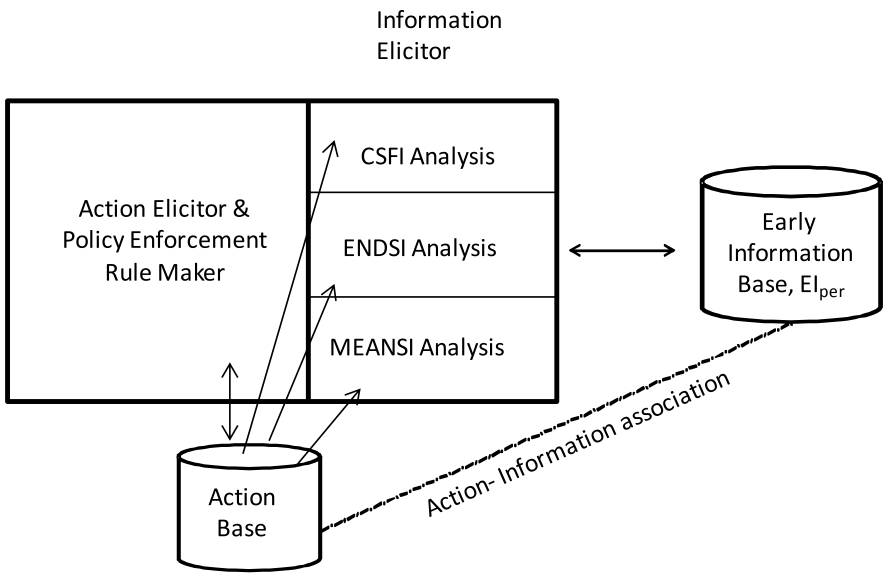
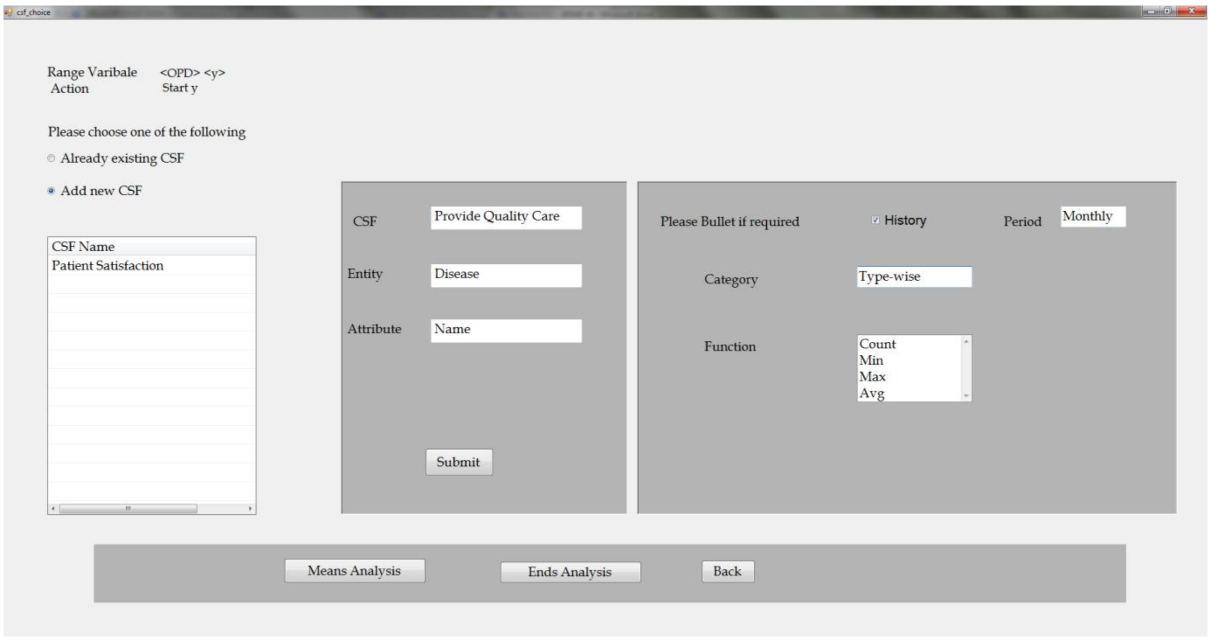

Information Elicitor
Action Elicitor & Policy Enforcement Rule Maker
CSFI Analysis
ENDSI Analysis
MEANSI Analysis
Early Information Base, EI_{per}
Action Base
Action- Information association

Fig 7.4: Back End Architecture of ELISPE

The user interfaces for the three information elicitation techniques is shown below.

1. **CSFI analysis:** Consider the action "start *y*" where y is an instance of OPD. One CSF is to provide quality care. To assess this factor information needed is number of doctor, specialty of doctor, number of patients, type of disease, name of disease. Doctor, Patient and Disease become entities. Specialty becomes an attribute of entity Doctor and for entity Disease attribute is name categorized type-wise. With the help of this information the requirements engineer is able to decide whether it is worthwhile to take the action "start *y*".

The user interface for CSFI analysis is shown in Fig 7.5. The top left hand side of the screen shows the action, along with the relevant range variables, for which information is being elicited. The requirements engineer is presented with two choices to either choose an existing CSF or to create a new one.



csf_choice

Range Varibale <OPD> <y>
Action Start y

Please choose one of the following
○ Already existing CSF
● Add new CSF

CSF Name
Patient Satisfaction

CSF: Provide Quality Care
Entity: Disease
Attribute: Name

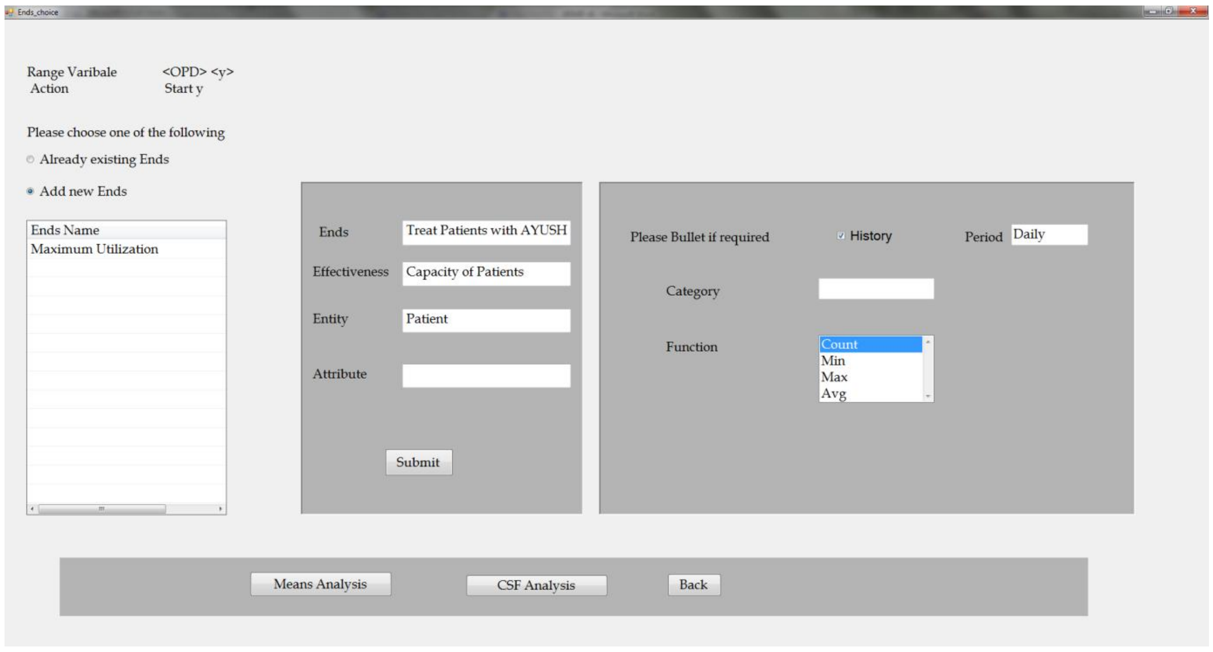
Please Bullet if required ☑ History
Period: Monthly
Category: Type-wise
Function: Count, Min, Max, Avg

Fig. 7.5: Eliciting Information by CSFI Analysis

When the former is chosen a list of the existing CSFs are displayed and the desired CSF can be chosen for modification. When the latter option of creating a new CSF is chosen the panel in the centre of the user interface is displayed. Here a new CSF is entered. Relevant entity with its attribute is further entered here. The right most panel is for entering additional information for the entity.

2. **ENDSI Analysis**: Revisit the action "start *y*", *y* is an instance of OPD. The objective or result of this action can be to treat patients using AYUSH. The effectiveness of this end can be assessed by Capacity of patients and information needed for the assessment may be daily count of patients.

The user interface for ENDSI analysis is similar to the screen used for CSFI analysis and is shown in Fig 7.6.


Ends_choice

Range Varibale <OPD> <y>
Action Start y

Please choose one of the following
Already existing Ends
[✓ Add new Ends]

Ends Name
Maximum Utilization

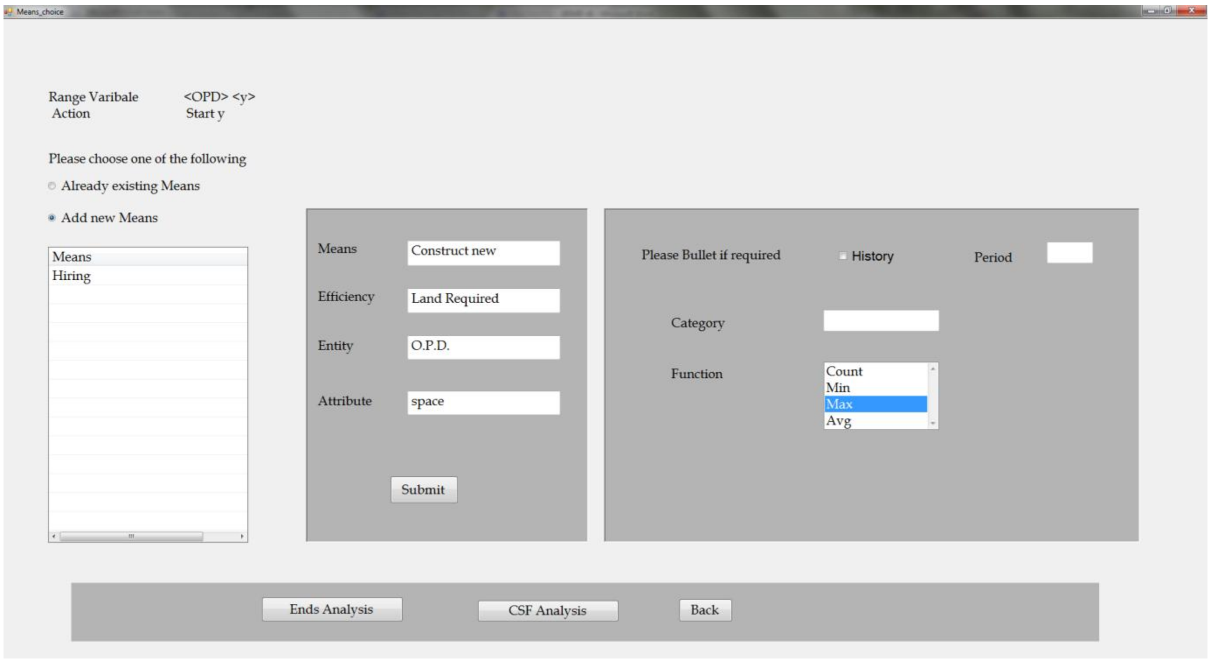
Ends: Treat Patients with AYUSH
Effectiveness: Capacity of Patients
Entity: Patient
Attribute:

Please Bullet if required [✓ History] Period: Daily
Category:
Function: [✓ Count] Min Max Avg

Fig 7.6: Eliciting Information by ENDSI Analysis

The right hand top side of the screen shows the action for which the analysis is being done. Again the two options of either selecting an already existing End or creating a new End are presented. The former shows a list of existing Ends. For the latter, Ends, effectiveness measure and relevant entity and attribute is entered in the centre panel that is displayed. Additional information for the entity is entered through the right most panel.

3. **MEANSI Analysis:** Again consider the action "start *y*", *y* is an instance of OPD. One means to perform this action is to construct a new OPD. Efficiency is land required. Entity is OPD. Information is maximum space needed.


Means_choice

Range Varibale <OPD> <y>
Action Start y

Please choose one of the following
○ Already existing Means
● Add new Means

Means
Hiring

Means: Construct new
Efficiency: Land Required
Entity: O.P.D.
Attribute: space

Please Bullet if required ☐ History Period
Category
Function: Count, Min, [✓ Max], Avg

Fig. 7.7: Eliciting Information by MEANSI Analysis

Fig 7.7 shows the user interface for MEANSI analysis. The screen is similar to CSF and Ends analysis. Means, Efficiency measures are input in the centre panel and entity, attribute and additional information is entered using the screen.

All the information elicited is stored in the early information base, EI_{per} of Fig 7.4. Since each piece of early information is linked to an action, there is an association between the two. This is shown in Fig 7.4 by the dotted line between the Action base and EI_{per}.

PART II: ELiciting Information Support for Operations (ELISO)

The architecture of ELISO, based on chapter 4, is shown below in Fig 7.8. The PER actions in the PER action base are presented to the requirements engineer. These are processed by the policy hierarchy maker. New actions are discovered from the actions presented and stored in the OP action base. So the OP action base contains the set of operational actions that are coming directly from the PER actions and the newly discovered actions.

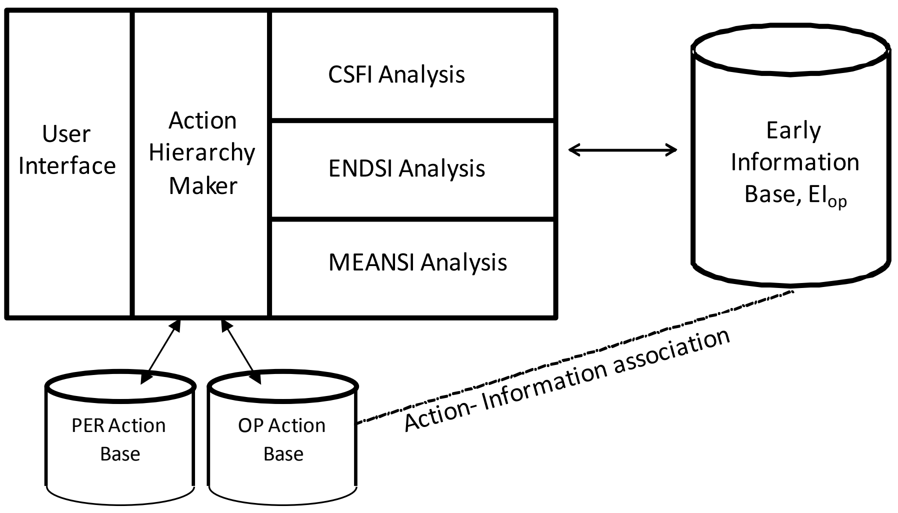


Fig. 7.8: Architecture of ELISO

Each action from the OP action base is input to the Information Elicitor where using CSFI, ENDSI and MEANSI analysis early information is elicited and stored in the early information base, EI_{op}. Again each piece of early information elicited is associated with an action from the OP action base.

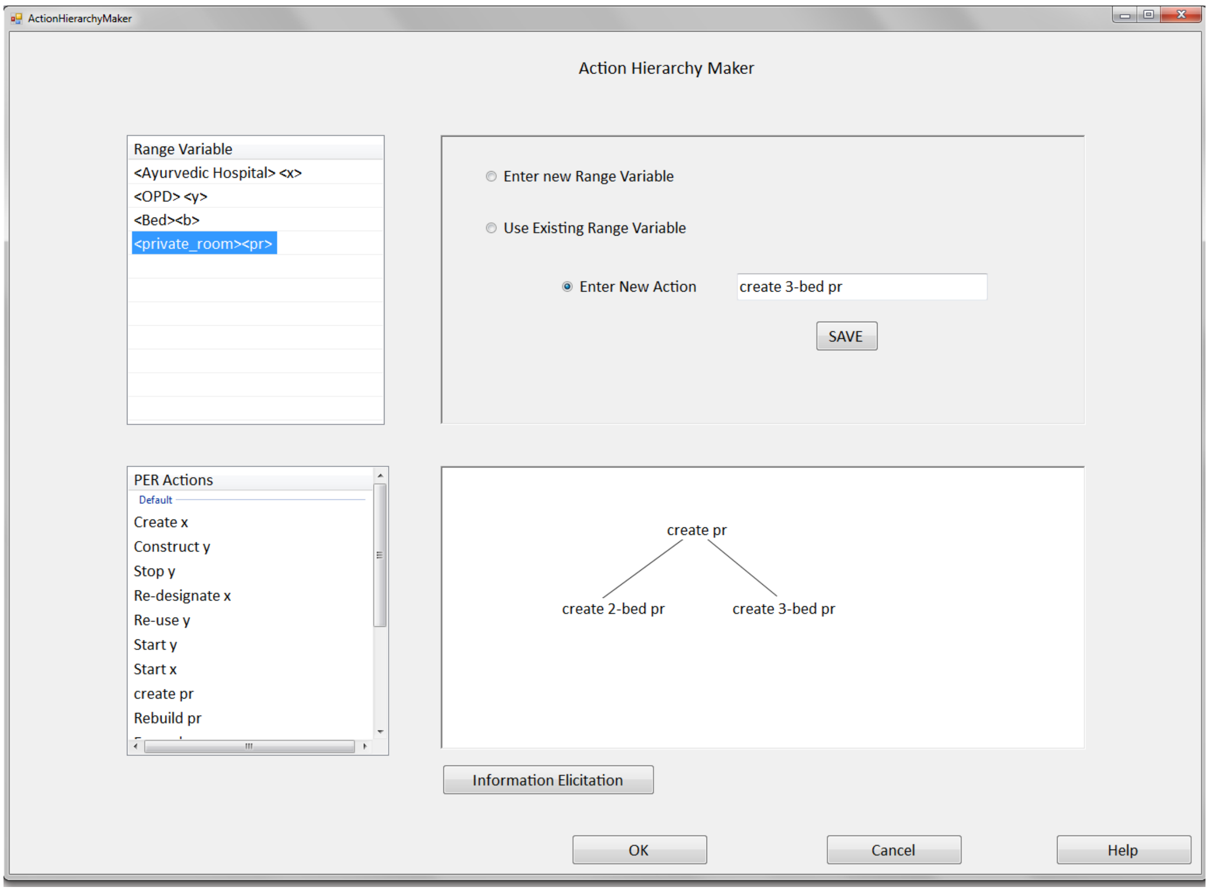
Action Hierarchy Maker part of ELISO is discussed below. The Information Elicitor is similar to the one in ELISPE and so is not discussed here again.

Action Hierarchy Maker

Fig 7.9 below shows the user interface. The screen is divided into three sections. The left hand side of the screen shows the range variables and the PER actions that are from the PER action base. The upper panel on the right hand side of the screen is where new range variable s and new actions are defined. The bottom panel of the screen is where the action hierarchy is constructed.

New actions may mean defining new range variables. For this, 'Enter new Range Variable' button has been provided. On checking the radio button, new range variables can be entered.

If an existing range variable can be used, ‘Use existing Range Variable’ button is clicked. From the list provided the necessary range variable can be selected. New actions are defined and are ready for use.



[✓ Enter New Action] create 3-bed pr

Fig. 7.9*:* Action Hierarchy Maker

The actions are dragged and dropped in the bottom panel of the screen and hierarchies generated. Fig 7.9 shows action create pr, where pr is an instance of private ward, as the action selected from the PER action base. Two new actions have been defined namely, create 2-bed pr and create 3-bed pr. Notice no new range variable has been defined. The IS/A hierarchy can also be seen in the figure.

PART III: Computer Aid For Decision Early Information elicitation (CADEI)

The third part of this chapter discusses CADEI that integrates EI_{op} and EI_{per}. This integration is based on the process discussed in chapter 5.

CADEI has four components, the metadata reader, correspondence drafter, information mapper and the conflict resolver. The architecture of CADEI is shown below in Fig 7.10 and 7.11. The first two components used in the process namely, metadata reader and correspondence drafter are shown in Fig 7.10. The architecture involving information mapper and conflict resolver is shown in Fig. 7.11

Fig 7.10 shows that metadata reader has two sources as input. The first source is PER-action base and EI_{per}, populated using ELISPE. The second source is OP-Action base and EI_{op} base populated using ELISO. The metadata reader reads the metadata of the two sources and sends the two to the correspondence drafter. The requirements engineer is presented with a list of four strategies to select from, based on which, the correspondence drafter finds correspondences between the metadata of DWper and DWop and stores the same.

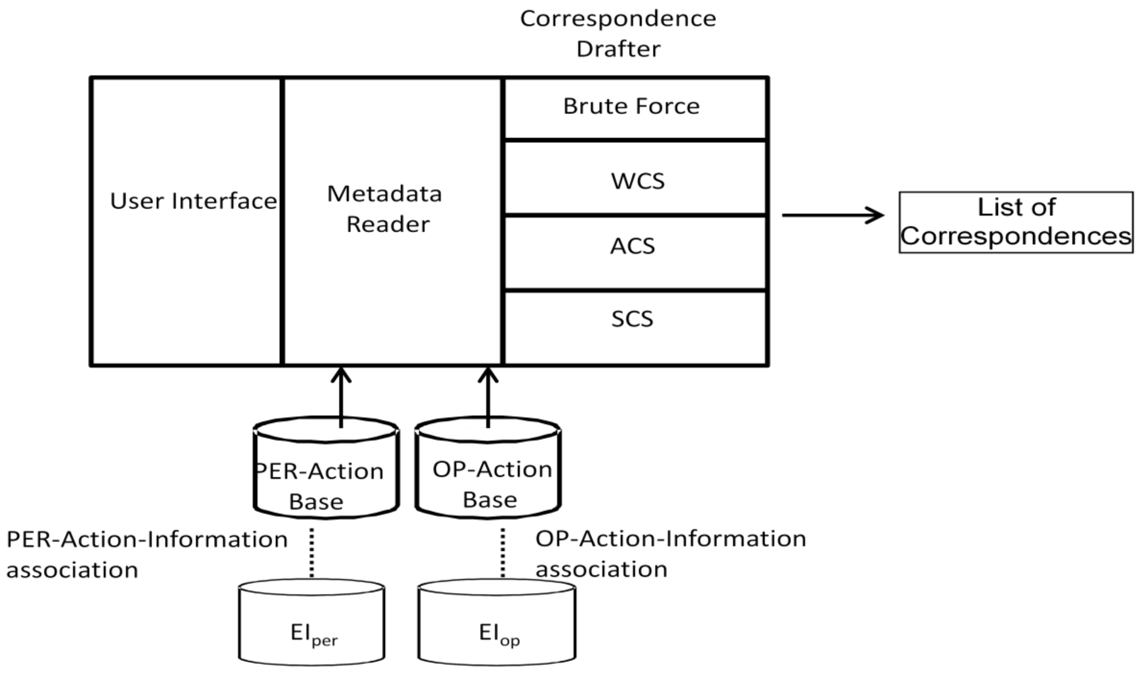


Fig. 7.10*:* Architecture of CADEI-I

The correspondences output from the correspondence drafter of Fig 7.10, along with the two early information bases to be integrated are the input to the information mapper of Fig 7.11. In other words, based on the correspondences, pair wise integration of information in EI_{op} and EI_{per} is performed by the information mapper. For each pair of information being integrated, the information mapper finds if it is fully, partially or not mapped. For fully mapped information, one copy is taken into the next iteration of integration. Partially mapped information is sent to the conflict resolver. Once the conflict is resolved, one copy of the resolved information is taken into the next iteration of integration. Naturally, for not mapped information both the pieces of information are taken into the next iteration of integration. The final set of integrated information is stored in $EI_{integrated}$ of Fig 7.11.

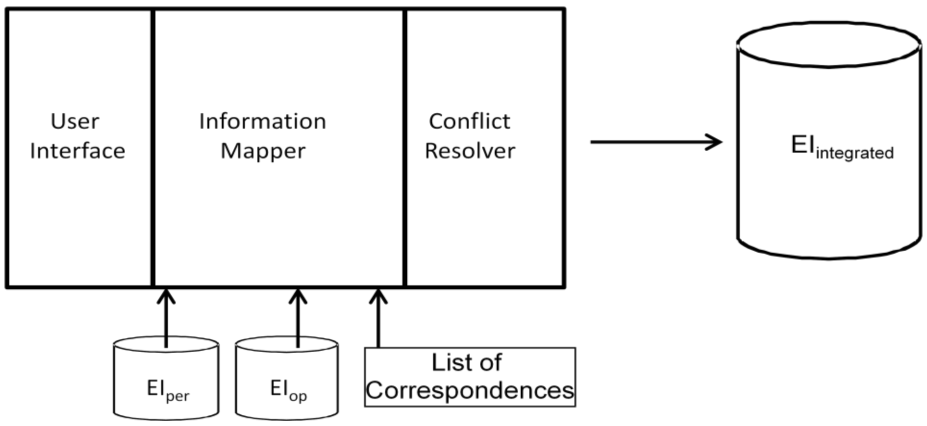
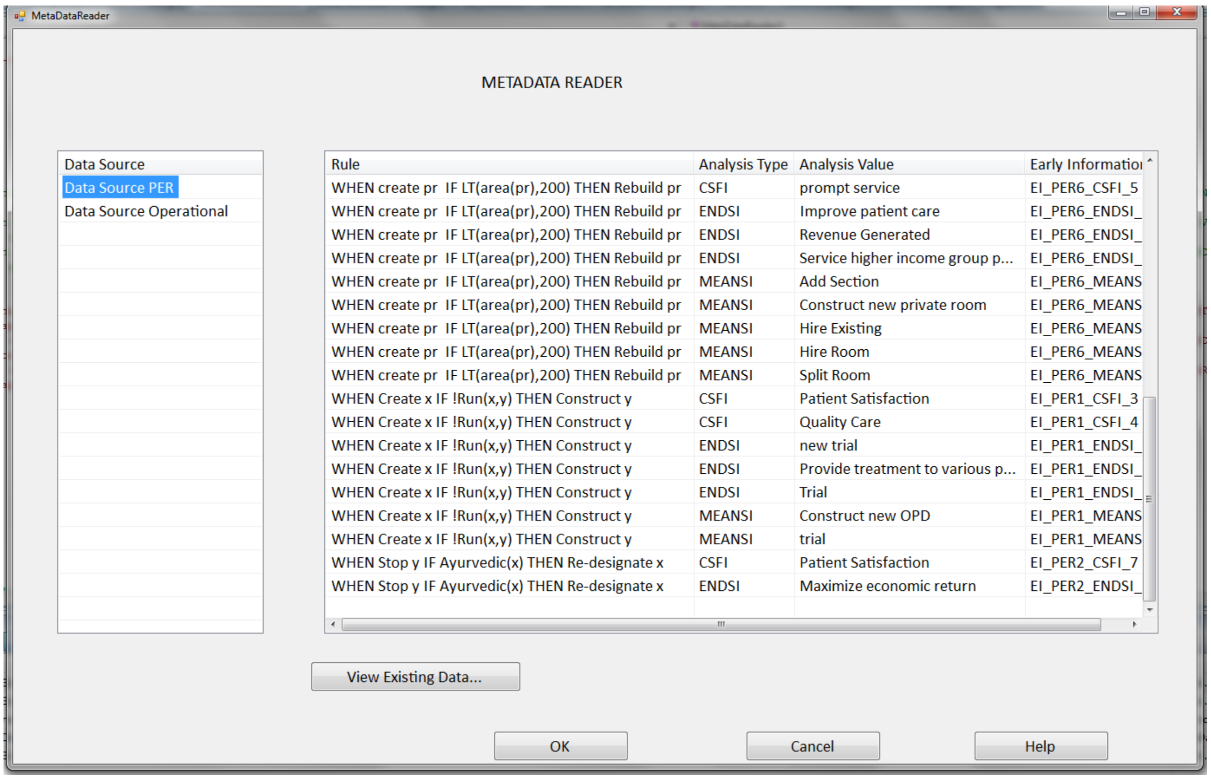


Fig. 7.11: Architecture of CADEI-II

As far as automation is concerned, no manual intervention is required in the metadata reader component of CADEI. With respect to the correspondence drafter, manual intervention is required to select a correspondence strategy. Once this is selected, correspondences are generated automatically, as discussed in chapter 6, for WCS and ACS strategies. For SCS manual intervention is required for finding equivalence (refer to chapter 6). No manual intervention is required in the information mapper component and the conflict resolver component. Thus, CADEI is a semi-automated tool to integrate early information with manual intervention required in the SCS strategy of the correspondence drafter.

The user interface for Metadata Reader is shown below in Fig 7.12. The left side panel shows all the data source from where the metadata is being read. The highlighted one shows the current data source. In Fig 7.12 it is the PER DW. The right hand side panel is the Metadata of DWper. The left most column shows the rule, the next column is for the analysis type, the third for analysis value and finally the fourth early information indicator. Since

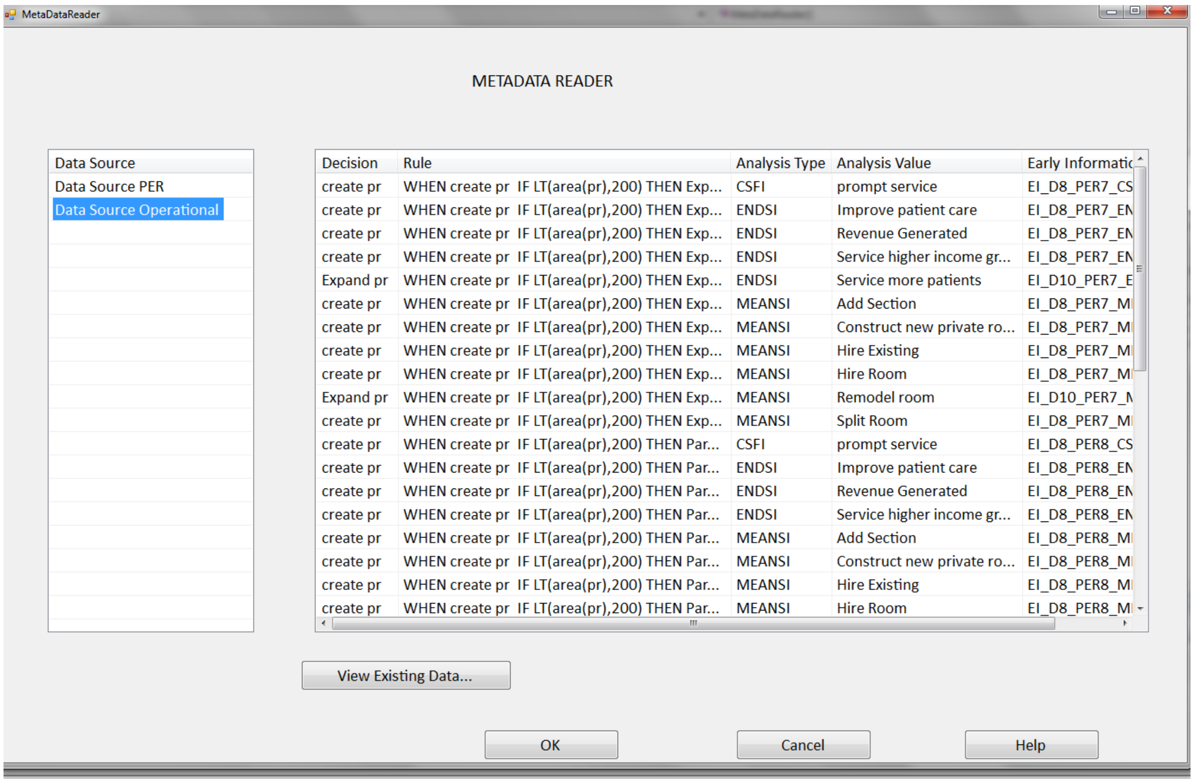


Data Source
Data Source PER
Data Source Operational

Rule	Analysis Type	Analysis Value	Early Informatio
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	CSFI	prompt service	EI_PER6_CSFI_5
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	ENDSI	Improve patient care	EI_PER6_ENDSI_
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	ENDSI	Revenue Generated	EI_PER6_ENDSI_
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	ENDSI	Service higher income group p...	EI_PER6_ENDSI_
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	MEANSI	Add Section	EI_PER6_MEANS
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	MEANSI	Construct new private room	EI_PER6_MEANS
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	MEANSI	Hire Existing	EI_PER6_MEANS
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	MEANSI	Hire Room	EI_PER6_MEANS
WHEN create pr IF LT(area(pr),200) THEN Rebuild pr	MEANSI	Split Room	EI_PER6_MEANS
WHEN Create x IF !Run(x,y) THEN Construct y	CSFI	Patient Satisfaction	EI_PER1_CSFI_3
WHEN Create x IF !Run(x,y) THEN Construct y	CSFI	Quality Care	EI_PER1_CSFI_4
WHEN Create x IF !Run(x,y) THEN Construct y	ENDSI	new trial	EI_PER1_ENDSI_
WHEN Create x IF !Run(x,y) THEN Construct y	ENDSI	Provide treatment to various p...	EI_PER1_ENDSI_
WHEN Create x IF !Run(x,y) THEN Construct y	ENDSI	Trial	EI_PER1_ENDSI_
WHEN Create x IF !Run(x,y) THEN Construct y	MEANSI	Construct new OPD	EI_PER1_MEANS
WHEN Create x IF !Run(x,y) THEN Construct y	MEANSI	trial	EI_PER1_MEANS
WHEN Stop y IF Ayurvedic(x) THEN Re-designate x	CSFI	Patient Satisfaction	EI_PER2_CSFI_7
WHEN Stop y IF Ayurvedic(x) THEN Re-designate x	ENDSI	Maximize economic return	EI_PER2_ENDSI_

Fig. 7.12*:* Metadata Reader for PER data source

The left panel of Fig 7.13 shows metadata being read for Operational data source. There is an additional column for Decision in the right hand side panel when compared with Fig 7.12.



Data Source
Data Source PER
Data Source Operational

Decision	Rule	Analysis Type	Analysis Value	Early Informatic
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	CSFI	prompt service	EI_D8_PER7_CS
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	ENDSI	Improve patient care	EI_D8_PER7_EN
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	ENDSI	Revenue Generated	EI_D8_PER7_EN
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	ENDSI	Service higher income gr...	EI_D8_PER7_EN
Expand pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	ENDSI	Service more patients	EI_D10_PER7_E
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Add Section	EI_D8_PER7_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Construct new private ro...	EI_D8_PER7_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Hire Existing	EI_D8_PER7_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Hire Room	EI_D8_PER7_M
Expand pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Remodel room	EI_D10_PER7_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Exp...	MEANSI	Split Room	EI_D8_PER7_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	CSFI	prompt service	EI_D8_PER8_CS
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	ENDSI	Improve patient care	EI_D8_PER8_EN
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	ENDSI	Revenue Generated	EI_D8_PER8_EN
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	ENDSI	Service higher income gr...	EI_D8_PER8_EN
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	MEANSI	Add Section	EI_D8_PER8_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	MEANSI	Construct new private ro...	EI_D8_PER8_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	MEANSI	Hire Existing	EI_D8_PER8_M
create pr	WHEN create pr IF LT(area(pr),200) THEN Par...	MEANSI	Hire Room	EI_D8_PER8_M

Fig. 7.13*:* Metadata Reader for Operational data source

On clicking the View Existing Data button on the bottom left hand side of the screen, Top 1000 rows of EI can be viewed.

The correspondences obtained from the correspondence drafter, are stored in a database table. The information mapper accesses the contents and processes each row and maps information. For this a table valued function was created. A snippet of the code is shown below.

```
DECLARE @ROW_ID      INT
SELECT * INTO dbo.#TEMP_SCS
FROM dbo.SCS

WHILE EXISTS (SELECT * FROM dbo.#TEMP_SCS)
BEGIN

       SELECT @ROW_ID = (SELECT TOP 1 ID
                         FROM dbo.#TEMP_SCS
                         ORDER BY ID ASC)

       SELECT * INTO #TEMP_MAPPEDINFO FROM fn_getMappedInformation(dbo.#TEMP_SCS.EI1,
dbo.#TEMP_SCS.EI2)

       DELETE FROM dbo.#TEMP_SCS
       WHERE @ROW_ID = ID
END
```

Summary

This chapter of the thesis discussed the architecture and the working of three tools one for eliciting EI_{per}, ELISPE, a second one for eliciting EI_{op}, ELISO and finally, CADEI, a tool that applies the process of Chapter 5 and integrates EI_{per} and EI_{op}.

Note: The ideas in this chapter have been published in International Journal of Information System Modeling and Design (IJISMD) 2015.

Chapter 8

Conclusion, Contribution and Future Scope

The thesis brought out certain limitations in current DWRE techniques. Firstly, it was observed that Data Warehouse support in terms of RE models and techniques has been extensively provided for operational level of decision making. However, it has been recently pointed out by BMM that there are other decisions that are taken in an organization, for example, deciding on policies and rules of business among others. Notice also, that while operational daily decision making is done by lower level management, decisions on arriving at organizations policies and rules of business are taken by relatively higher levels of management. A study of the available literature shows that DW technology *does not address issues around Data Warehouse support for these 'alternate' decisions*.

It was observed that DWRE has adopted both decisional and information perspectives but have not treated the notion of a decision or of information as first class concepts of DWRE. The consequence of this is that

a. The relationship between the notions of decision and information is not fully explored. Thus, the decision-information association is left un-articulated and remains implicit. This inhibits a full investigation into what information is needed for which decision and vice-versa.
b. Decision models have not been developed. It is therefore difficult to know the structure of a decision and the semantic notions that go into defining it. The former means that it is not possible to adopt model driven requirements engineering leading to relatively poor guidance in the elicitation task. The latter implies that the conceptual basis for adopting the notion of a decision itself remains weak.

c. Information models are assumed to be multi-dimensional in nature. This leads to an emphasis on determining facts and dimensions at the expense of determining information properties like required aggregations and historical information needs. As for decisions, this implies that only partial guidance can be provided in the information elicitation task.

Not only are the issues of guidance and semantics of concepts important but also important is the issue of the nature of information elicited. While arriving at multi-dimensional structures is essential, the thesis argues that considerable pre-work is required before committing to these. Therefore, the approach adopted is to postpone the structuring of information elicit, examine and analyse information that is unstructured. It is only once these steps are carried out to the satisfaction of the requirements engineer that the structuring issue is addressed.

To sum up, the thesis found that there is *need to treat decision and information as first class concepts of RE models, develop decision and information models for conceptual clarity and effective guidance, and to lay emphasis on eliciting early, unstructured information be before arriving at multi-dimensional structures.*

In addressing these limitations, the thesis offers a solution for addressing both strategic and operational in the same DW system. In this regard, the thesis starts by establishing the ‘Decision Environment’ and derives a typology of decisions. There are two broad levels of decisions: Managerial and Tactical. Managerial decisions consist of Policy formulation decisions and Policy Enforcement level decisions, which enforce the formulated policies. Interest in this thesis is in the latter. Tactical decisions address operational level decisions. Thus, there are two kinds of decision support needed, one for policy enforcement rule formulation decisions and the other for operational decisions.

In order to identify the needed information for supporting this decision making, the thesis proposes a generic platform with 'Decision Requirement' as the central concept. Recall, the RE process is rooted in Decision Requirement which is a tuple <decision, information>. Fig 8.1 shows the generic platform having generic models of Decision Requirement, decision and information described in Chapter 2.

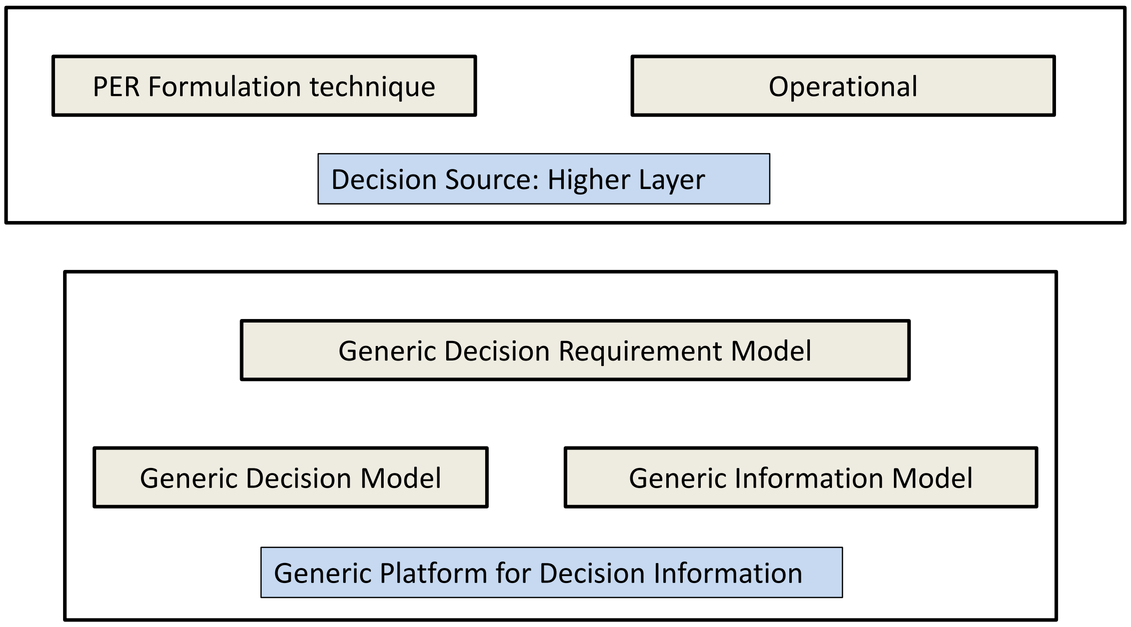

PER Formulation technique
Operational
Decision Source: Higher Layer
Generic Decision Requirement Model
Generic Decision Model
Generic Information Model
Generic Platform for Decision Information

Fig 8.1: The Generic Platform

Decision Requirement implies that RE process has two steps, first to determine the choice set of decisions and then to elicit information to choose one from the choice set. The thesis proposes a set of generic techniques for eliciting information that shall be stored in the DW. The three information elicitation techniques namely, CSFI, ENDSI and MEANSI therefore also are part of this generic platform. This information is unstructured and 'early'. It has to be converted to a structured multi-dimensional form before DW can be created.

Now, there are two sources of decisions, one from the PER layer and the other from the operational layer. This forms a higher layer that sits on the generic platform (see Fig 8.1) and exploits the defined models and information elicitation techniques. PER life cycle formulates PERs and creates Policy enforcement rule DW with PER base and its own PER early information base. Operational life cycle creates a DW for operational decision making with operational action base and information in its own operational early information base.

Information in DWper and DWop are not disjoint. Therefore the issue arose as to whether to integrate these Data Warehouses or to keep them separate. The thesis found that there are two problems with keeping separate Data Warehouses:

- Difference in refresh cycles between DWper and DWop can occur because DWop is for operational decision making and refreshed more frequently than DWper. Thus, rule formulators and operational decision makers end up taking decisions on different data in this temporal window. The larger this window, the longer this inconsistency exists.
- Loss of business control occurs when data of an operational DW calls for decision makers of the policy enforcement DW to take decisions, but the decisions are not taken because data in the latter do not suggest this need.

Thus, integration is required to maintain compatibility between PER and operational level.

The thesis shows that there are in fact two forms of integration that can exist, horizontal and vertical. While the former integrates data marts at the same level of decision making, the latter integrates data marts across PER and operational levels. For vertically integrating DWper and DWop, the thesis proposes a ‘build by integrating’ approach which can be seen as pair-wise. When a new data mart is to be built, its requirements specification is integrated with an existing one. The integrated requirements specification then goes through the development cycle. Thus, the point of integration is moved upstream into the requirements stage. The advantage of integrating upstream and in a pair-wise fashion is that downstream development effort is minimized. Secondly, it never allows completely operational data marts to be built before these are integrated. The problem of inconsistency detection and correction is pre-empted by not allowing an un-integrated data mart to be developed. Further, at any time a complete logical Data Warehouse is available for decision making.

This upstream Requirements level integration can be either done by integrating ER schemas or by integrating early information. The latter point is chosen for integration because while integrating ER schemas, development has to proceed until ER schema development. This latter is a waste of effort from the point of view of the integrated schema. Integration is proposed to be done as a four-step process consisting of the Metadata reader, Correspondence Drafter, Information Mapper and Conflict Resolver. The integrated early information obtained from this process is then converted into an ER schema and finally into the star schema.

Through vertical integration, an integrated enterprise wide DW is obtained that can be used by decision makers at both the operational and PER levels. Thus, the thesis has offered a solution for providing both strategic and operational decision-making in an integrated manner.

Contribution of the Thesis

A summary of the contributions made in the thesis is as follows:

1. Addressing full decisional making continuum: Data Warehouse support has been extended from providing just operational support to providing strategic (policy

enforcement) and operational support. Policy enforcement level provides the context for operational level decisions. Decisions at every level of the continuum interact with an integrated Data Warehouse. Policy enforcement level decisions look at decisions involving corrective actions that must be taken in the event of policy violation. At the operational level the alternate actions are input as decisions. Information associated at both levels is identified individually and later integrated.

2. Elicited Information can be traced back to members of the choice set thereby facilitating decision making. In the case of policy enforcement rules the choice set is {select A, modify A, delete A} where A is an action. For each alternative in the choice set information is elicited. This relates information to a particular member of the choice set. In the case of operations the choice set is {select A1, select A2, select A3} where again information is elicited for each alternative in the choice set relating information to a member of the choice set.

3. Discovery of early information: Information is obtained in a two-step process, early information elicitation step and late information elicitation step. Early information elicited from policies and policy enforcement rules is highly unstructured and high level. Early information has to be converted into a more structured form called late information. This is done by first converting early information into ER schema. ER to star schema conversion is done by applying existing techniques.

4. Development of computer aided tools: This thesis developed three tools to semi-automate PER formulation decisions called ELISPE, operational decisions called ELISO, and CADEI to integrate early information bases of PER and operational DWs.

Future Work and Open Problems

This thesis throws up a number of directions of future work as follows:

1. The decisional environment suggests that managerial decision making consists of Policy formulation and PER levels of decision-making. Out of these two, the present thesis has addressed the PER level only. Thus, there is a need to develop DWRE technique for Policy level of decision making. As far as the generic platform is concerned this means adding an additional Policy RE technique to the higher decision source.

   There is also a need to integrate the Policy DW system with PER and operational Data Warehouses so as to provide a comprehensive decision making environment. Again the problem here is one of vertical integration with the Policy layer at a 'higher' level than PER layer and operational layer.

2. One-time DW development carries with it long lead times to deliver. Approaches to agile development of DW development have been proposed. The incorporation of agility in development of PER-Operational Data Warehouses is an important issue that may cut down development lead times.

3. This problem pertains to the sources of operational decisions. Literature suggests two broad kinds of analysis namely, goal and business indicator as a source of operational decisions. The thesis proposed a third, policy compliance for operational decisions. There is need to investigate if all three need to be carried out to determine the set of operational decisions or is one of them is superset of the other.

4. When applying elicitation techniques, experience showed that while it was easier to perform MEANSI and ENDSI analysis, CSFI was more difficult to perform. The reasons behind the partial applicability of the CSFI technique need investigation and is an open problem falling out of this thesis.

References

1. Alshboul, R. (2012). Data Warehouse Explorative Study. *Applied Mathematical Sciences*, *6*(61), 3015-3024.
2. Antón, A. I. (1996, April). Goal-based requirements analysis. In *Requirements Engineering, Proceedings of the Second International Conference on* (pp. 136-144). IEEE.
3. Antón, A. I., and Potts, C. (1998, April). The use of goals to surface requirements for evolving systems. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on* (pp. 157-166). IEEE.
4. Auechaikul T, Vatanawood W (2007). A Development of Business Rules with Decision Tables for Business Processes. TENCON 2007- 2007 IEEE Region 10 Conference, (pp 1-4). IEEE
5. AYUSH, (2000), Government of India, Department of AYUSH, Ministry of Health and Family Welfare, No.Z.20018/4/2000-ISM (Tech)/HD Cell, National Competitive Bidding
6. Batini C , Lenzerini M. (1984), A Methodology for Data Schema Integration in the Entity Relationship Model, *IEEE Transactions on Software Engineering*, vol. 10, no. 6, 650-663, 1984
7. Batini, C., Lenzerini, M., & Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. ACM computing surveys (CSUR), 18(4), 323-364.
8. Bernstein, P. (1976) Synthesizing Third Normal Form Relations from Functional Dependencies*, ACM Transactions on Database Systems,* vol. 1, no. 4, 277-298, 1976.
9. Biskup J., Bernhard C. (1986), A Formal View Integration Method, *Int'I ACM SIGMOD Conf,* 398-407, 1986

10. Boehm, Barry W. and Philip N. Papaccio. (1988) Understanding and Controlling Software Costs, *IEEE Transactions on Software Engineering,* v. 14, no. 10, pp. 1462-1477

11. Boehnlein, M., and Ulbrich vom Ende, A. (1999, November). Deriving initial Data Warehouse Structures from the Conceptual Data Models of the Underlying Operational Information Systems. In Proc. Of Workshop on Data Warehousing and OLAP (pp. 15-21). ACM

12. Boehnlein, M., and Ulbrich vom Ende, A. (2000). Business Process Oriented Development of Data Warehouse Structures. In Proceedings of Data Warehousing 2000 (pp. 3- 21). PhysicaVerlag HD

13. Bonifati A.. Cattaneo F., CeriS., A. Fuggetta, and S. Paraboschi (2001). Designing Data Marts for Data Warehouses. ACM Trans. Software. Eng. Methodology, 10(4). (pp. 452–483).

14. Boulos, G. S., Burgess J. P., Jeffrey R.C. (2002), Computability and Logic, 4th edition, Cambridge University Press, 2002

15. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., & Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, *8*(3), 203-236.

16. BRG, Business Rules Group (2010), The Business Motivation Model: Business governance in a volatile world, Release 1.4, July 2010

17. Bruckner, R., List, B., &Scheifer, J. (2001). Developing requirements for data warehouse systems with use cases. *AMCIS 2001 Proceedings*, 66.

18. Bubenko, J., Rolland, C., Loucopoulos, P., and DeAntonellis, V. (1994, April). Facilitating fuzzy to formal requirements modelling. In *Requirements Engineering, 1994., Proceedings of the First International Conference on* (pp. 154-157). IEEE.

19. Bullen C.V., Rockart J.F. (1981), A Primer of Critical Success Factors, CISR No. 69 Sloan WP No. 1220-81 Center for Information Systems Research, Sloan School of Management Massachusetts Institute of Technology, 1981
20. Cabibbo, L., & Torlone, R. (2004, June). On the integration of autonomous data marts. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on* (pp. 223-231). IEEE.
21. Cabibbo, L., Panella, I., Torlone, R., & Tre, U. R. (2006, April). DaWaII: a Tool for the Integration of Autonomous Data Marts. In *ICDE* (p. 158).
22. Casanova M., Vidal V. (1983), A Sound Approach to View Integration, *Proceedings of the ACM Conference on Principles of Database Systems*, 35-47, 1983.
23. Castro, J., Kolp, M., and Mylopoulos, J. (2002). Towards requirements-driven information systems engineering: the Tropos project. *Information systems,27*(6), 365-389.
24. Charette, R.N. (2005) Why software FAILS*, IEEE Spectrum*, vol. 42, no. 9, pp. 42-49, 2005
25. CMP., *Data Mart Consolidation* and *Business Intelligence Standardization, available at* www.businessobjects.com/pdf/investors/data_mart_consolidation.pdf
26. Cockburn, A. Structuring Use Cases with Goals, 1997. *Journal of Object-Oriented Programming*.
27. Coman, A., and Ronen, B. (2010). Icarus predicament: Managing the pathologies of overspecification and overdesign. *International Journal of Project Management*, *28*(3), 237-244.
28. Cooke, Philip, and Kevin Morgan.(1993) "The network paradigm: new departures in corporate and regional development." *Environment and planning D: Society and space* 11,.5 543-564, 1993

29. cravero Leal, A., Mazón, J. N., & Trujillo, J. (2013). A business-oriented approach to data warehouse development. *Ingeniería e Investigación*, *33*(1), 59-65.
30. CREWS Team (1998), The CREWS glossary, CREWS report 98-1, http ://SUNSITE.informatik.rwth-aachen.de/CREWS/reports.htm
31. Dardenne, A., Van Lamsweerde, A., and Fickas, S. (1993). Goal-directed requirements acquisition. *Science of computer programming*, *20*(1), 3-50.
32. Dawyndt P., Vancanneyt M., Meyer H. D., and Swings J.. (2005). Knowledge Accumulation and Resolution of Data Inconsistencies during the Integration of Microbial Information Sources, IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 8, 2005, 1111-1126
33. Dayal, U., & Hwang, H. Y. (1984). View definition and generalization for database integration in a multidatabase system. *Software Engineering, IEEE Transactions on,* (6), 628-645.
34. Dayal, U., Castellanos, M., Simitsis, A., & Wilkinson, K. (2009, March). Data integration flows for business intelligence. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 1-11). ACM.
35. DeMarco, T., Plauger, PJ. (1979). *Structured analysis and system specification*. Prentice Hall
36. Elmasri R. , Navathe S.B. (2004), Fundamentals of Database Systems, 4th edition, Pearson Educaiton Inc. 2004
37. Eric, S. K. (1997). Why Agent-Oriented Requirements Engineering. *Proceedings of 3rd Workshop on Requirements Engineering For Software Quality*.
38. Flyvbjerg, B. and Budzier, A., (2011). Why your IT project may be riskier than you think, *Harvard businessreview*, vol. 89, no. 9, pp. 23-25, 2011

39. Fu G., Shao J., Embury S.M., Gray W.A., Liu X.(2001). A Framework for Business Rule Presentation. In Database and Expert Systems Applications, 2001. Proceedings. 12th International Workshop on (pp. 922-926). IEEE
40. R. Fuller and C. Carlsson (1996), Fuzzy multiple criteria decision making: Recent developments, Fuzzy Sets and Systems, 78, (1996), pp. 139-153
41. Gam, I., and Salinesi, C. (2006). A requirement-driven approach for designing data warehouses. *Requirements Engineering: Foundation for Software Quality (REFSQ).*
42. Gamut L.T.F. (1991), Logic, Language, and Meaning, Volume 2: Intensional Logic and Logical Grammar, University of Chicago Press, 1991
43. Gbande C.A., Akuhwa P.T. (2015), Decision Theory and Analysis; An Optima Value Creation Precursor for Organizations, Open Journal of Applied Sciences, 355-367, 2015
44. Giorgini, P., Rizzi, S., Garzetti, M. (2005). Goal-oriented requirement analysis for data warehouse design. In Proceedings of the 8th ACM international workshop on Data warehousing and OLAP (pp. 47-56). ACM.
45. Glinz, M. (1995). An integrated formal model of scenarios based on statecharts. In *Software Engineering—ESEC95* (pp. 254-271). Springer Berlin Heidelberg.
46. Goguen, J. A., & Linde, C. (1993). Techniques for requirements elicitation. *RE*, *93*, 152-164.
47. Golfarelli M., Maio D., Rizzi S. (1998, January). Conceptual Design of Data Warehouses from E/R schemes. In System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on (Vol. 7, pp. 334-343). IEEE.
48. Golfarelli, M. (2010). From User Requirements to Conceptual Design in Data Warehouse Design. *Data Warehousing Design and Advanced Engineering Applications: Methods for Complex Construction*

49. Golfarelli, M., Rizzi, S. (1999). Designing the Data Warehouse: Key Steps and Crucial Issues. In Journal of Computer Science and Information Management (JCST), Vol. 2. No.3, 88-100. Springer

50. Golfarelli, M., Rizzi, S., & Turricchia, E. (2011). Modern software engineering methodologies meet data warehouse design: 4WD. In *Data Warehousing and Knowledge Discovery* (pp. 66-79). Springer Berlin Heidelberg.

51. Harel, D. (1987). 'Statecharts: A visual formalism for complex systems'. *Science of computer programming*, *8*(3), 231-274.

52. Haumer, P., Pohl, K., and Weidenhaupt, K. (1998). Requirements elicitation and validation with real world scenes. *Software Engineering, IEEE Transactions on*,*24*(12), 1036-1054.

53. Hayen R., Rutashobya C., Vetter D., (2007), An Investigation Of The Factors Affecting Data Warehousing Success, Issues In Information Systems, Volume VIII, No. 2, 547-553, 2007

54. Hickey, A., & Davis, A. (2003). Barriers to Transferring Requirements Elicitation Techniques to Practice. In *2003 Business Information Systems Conf*.

55. Hillman A.j., Hitt M.A (1999), Corporate Political Strategy Formulation: A Model of Approach, Participation, and Strategy Decisions, Academy of Management Review, 24, 4, 825=842, 1999

56. Holbrook III, H. (1990). A scenario-based methodology for conducting requirements elicitation. *ACM SIGSOFT Software Engineering Notes*, *15*(1), 95-104.

57. Hooks, I. F., and Farry, K. A. (2001). Customer-centered products: creating successful products through smart requirements management. AMACOM Div American Mgmt Assn

58. Horkoff J., and Yu E (2012). Comparison and Evaluation of Goal-oriented Satisfaction Analysis Techniques. Requirement Engineering Journal, 1-24 Springer

59. Horkoff, J., and Yu, E. (2010). Interactive analysis of agent-goal models in enterprise modeling. *International Journal of Information System Modeling and Design (IJISMD)*, *1*(4), 1-23.

60. Hüsemann, B., Lechtenb¨orger, J., Vossen, G. (2000). Conceptual Data Warehouse Design. In Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW2000) Stockholm, Sweden

61. IBM Business Analytics for Manufacturing, White Paper,2013

62. IEEE Standard, IEEE-Std 610, 1990

63. Inmon, B. (2005) *Building the data warehouse, fourth edition*. New York: John Wiley & Sons.

64. Inmon, W. H. (1996). Building the Data Warehouse, New York, Chichester, Brisbane, Toronto, Singapur.

65. Inmon, W. H., Strauss, D., & Neushloss, G. (2010). *DW 2.0: The architecture for the next generation of data warehousing: The architecture for the next generation of data warehousing*. Morgan Kaufmann.

66. Imhoff, C., & White, C. (2008). Full Circle: Decision Intelligence (DSS 2.0). *B-Eye-Network, Published: August, 27.*

67. Jeffrey R.C. (1991), *Formal Logic: Its Scope and Limits*, 3rd ed. NY, McGraw-Hill, 1991

68. Jones, R.G. and George, J.M (2008) Contemporary Management. 5th Edition, McGraw-Hill International Edition, McGraw-Hill/Irwin, New York. 2008

69. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A., & Mayorova, D. (2014). A requirement-driven approach to the design and evolution of data warehouses.*Information Systems*, *44*, 94-119.
70. Kavakli, V., and Loucopoulos, P. (1998, January). Goal-driven business process analysis application in electricity deregulation. In *Advanced Information Systems Engineering* (pp. 305-324). Springer Berlin Heidelberg.
71. R.L. Keeney and H. Raiffa (1993), Decisions with Multiple Objectives: Preferences and Value Trade-Offs, (Cambridge University Press, 1993)
72. R.L. Keeney (1999), Foundations for Making Smart Decisions, IIE Solutions, 31, No. 5, (1999), pp. 24-30
73. Kimball R. and Ross M. (2002), The data warehouse Toolkit: The Complete Guide to Dimensional Modeling, Wiley, 2002
74. Kimball, R. (1996): The Data Warehouse Toolkit, New York: J. Wiley & Sons.
75. Kimball, R. (1997): A Dimensional Manifesto, DBMS Online, August, 1997.
76. Kotonya, G. & Sommerville, I., (1998). *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc
77. Lapouchnian, A. (2005). Goal-oriented requirements engineering: An overview of the current research. *University of Toronto*.
78. Lee, M. L., & Ling, T. W. (1995). Resolving structural conflicts in the integration of entity-relationship schemas. In *OOER95: Object-Oriented and Entity-Relationship Modeling* (pp. 424-433). Springer Berlin Heidelberg.
79. Lee, M. L., & Ling, T. W. (2003). A methodology for structural conflict resolution in the integration of entity-relationship schemas. *Knowledge and information systems*, *5*(2), 225-247.

80. Leffingwell, D., and D. Widrig. (2000). *Managing Software Requirements*, Addison-Wesley.
81. Leite J.C.S.P., Leonardi M.C. (1998, April). Business Rules as Organizational Policies. In Software Specification and Design, (pp. 68-76) IEEE
82. Lindbloom C.E. (1993), Woodhouse E.J., 3rd edition, Prentice Hall, 1993
83. List, B., Bruckner, R. M., Machaczek, K., & Schiefer, J. (2002, January). A comparison of data warehouse development methodologies case study of the process warehouse. In *Database and Expert Systems Applications* (pp. 203-215). Springer Berlin Heidelberg.
84. Liu, L., & Yu, E. (2004). Designing information systems in social context: a goal and scenario modelling approach. *Information systems*, *29*(2), 187-203.
85. Makarov, I., M., Vinogradskaya, T., M., Rubchinsky, A., A., Soko-. lov, V.B. (1987), “The Theory of Choice and Decision Making”, Mir Publishers, Moscow, 1987
86. Matulevičius, R., and Heymans, P. (2007). Comparing goal modelling languages: An experiment. In *Requirements Engineering: Foundation for Software Quality* (pp. 18-32). Springer Berlin Heidelberg.
87. Mazón, J. N., Pardillo, J., & Trujillo, J. (2007). A model-driven goal-oriented requirement engineering approach for data warehouses. In *Advances in Conceptual Modeling–Foundations and Applications* (pp. 255-264). Springer Berlin Heidelberg.
88. Mcleod R Jr., Schell G. (2001), Management Information Systems, 8th ediction, Prenticie Hall, 2001

89. Moody L.D., and Kortink M.A.R. (2000), From Enterprise Models to Dimensional Models: A Methodology for Data Warehouses and Data Mart Design, Proc. of the Intl

Workshop on Design and Management of Data Warehouses, Stockholm, Sweden, (pp. 5.1-5.12)

90. Muehlen M Z, Kamp G (2007). Business Process and Business Rule Modeling: A Representational Analysis. In EDOC Conference Workshop, 2007. EDOC07. Eleventh International IEEE (pp. 189-196). IEEE.

91. Mullins, L.J. (2010) Management and Organizational Behavior. 9th Edition, England Pearson Education Limited, London. 2010

92. Mylopoulos, J., Chung, L., & Yu, E. (1999). From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, *42*(1), 31-37.

93. Navathe S.B.,, Elmasri R., James L. (1986), Integrating User Views in Database Design, *IEEE Computer*, 50-62, 1986

94. Neto, F. M. M., and Morais, M. J. D. O. (2013). An agent-based approach for supporting the knowledge transfer in the software requirements engineering. *International Journal of Business Information Systems*, *12*(1), 23-43.

95. Kelly-Newton (1980), Lauren. *Accounting policy formulation: the role of corporate management*. Addison Wesley Publishing Company, 1980.

96. Nuseibeh, B., & Easterbrook, S. (2000, May). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*(pp. 35-46). ACM.

97. OMG. (2008). Semantics of Business Vocabulary and Business Rules (SBVR), v1.0, January

98. Paim, F. R. S., and de Castro, J. F. B. (2003, September). DWARF: An approach for requirements definition and management of data warehouse systems. In*Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International* (pp. 75-84). IEEE.

99. Park Y. T. (2000), National systems of Advanced Manufacturing Technology (AMT): hierarchical classification scheme and policy formulation process, Technovation, 20,3, 151-159, 2000

100. Plihon, V., Ralyte, J., Benjamen, A., Maiden, N. A., Sutcliffe, A., Dubois, E., & Heymans, P. (1998). A reuse-Oriented Approach for the Construction of Scenario Bases Methods. In *Proceedings of International Conference on Software Process* (pp. 1-16).

101. Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated.

102. Pohl, K., & Haumer, P. (1997, June). Modelling contextual information about scenarios. In *Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ* (Vol. 97, pp. 187-204).

103. Potts, C., Takahashi, K., &Antón, A. I. (1994). Inquiry-based requirements analysis. *IEEE software*, *11*(2), 21-32.

104. Prakash N., and Bhardwaj H. (2012). Early Information Requirements Engineering for Target Driven Data Warehouse Development. In The Practise of Enterprise Modeling (pp.188-202). Springer Berlin Heidelberg

105. Prakash N., and Gosain A. (2003). Requirements Driven Data Warehouse Development, In CAiSE Short Paper Proceedings (pp. 13-17)

106. Prakash N., and Gosain A. (2008). An Approach to Engineering the Requirements of Data Warehouses. Requirements Engineering Journal, Springer, 13 (1), 49-72

107. Prakash, N., and Bhardwaj, H. (2014). Functionality for Business Indicators in Data Warehouse Requirements Engineering. In *Advances in Conceptual Modeling* (pp. 39-48). Springer International Publishing.

108. Prakash, N., Prakash, D., & Sharma, Y. K. (2009). Towards Better Fitting Data Warehouse Systems. In *The Practice of Enterprise Modeling* (pp. 130-144). Springer Berlin Heidelberg.

109. Raver N., Hubbard G. U. (1977), Automated Logical Database Design methodology and Techniques", *IBM Systems Journal*, vol. 16, no. 3, 1977.

110. Reed, S. E., Na, D. Y., Mayo, T. C., Shapiro, L. W., Duty, J. B., Conklin, J. H., & Brown, D. E. (2010, April). Implementing and analyzing a data mart for the Arlington County initiative to manage Domestic Violence offenders. In *Systems and Information Engineering Design Symposium (SIEDS), 2010 IEEE* (pp. 82-87). IEEE.

111. Riazati, D., Thom, J. A., & Zhang, X. (2010, January). Inferring aggregation hierarchies for integration of data marts. In *Database and Expert Systems Applications* (pp. 96-110). Springer Berlin Heidelberg.

112. Ritchie J.R.B. (1988), Consensus policy formulation in tourism: Measuring resident views via survey research, Tourism Management, 9,3, 199-212, 1988

113. Robertson, S., and Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*. Addison-wesley

114. Rosca D, Greenspan S, Feblowitz M, Wild C (1997). A Decision Making Methodology in Support of the Business Rules Lifecycle. Requirements Engineering, 1997, In Proceedings of the Third IEEE International Symposium on (pp. 236-246). IEEE

115. Roy B. (1996), Multicriteria Methodology for Decision Aiding, Dordrecht, Kluwer Academic Publishers (1996)

116. Roy, B. (2005) Paradigms and challenges, Book chapter, In Multiple Criteria Decision Analysis - State of the Art Survey, Springer. editor(s) J. Figueira, S. Greco, M. Ehrgott, (2005) 3-24

117. Rubin, K. S., and Goldberg, A. (19s92). Object behavior analysis. *Communications of the ACM*, *35*(9), 48-62.

118. Saaty T.L. (1980), The Analytic Hierarchy Process, NY, McGraw Hill 1980

119. Schaefer, B. C., Tanrıkulu, E., &Breiter, A. (2011). Eliciting user requirements when there is no organization: a mixed method for an educational data warehouse project. *Procedia-Social and Behavioral Sciences*, *28*, 743-748.

120. Shoenfield, J. R. (1967), *Mathematical Logic*, Reading, Massachusetts: Addison-Wesley, 1967

121. Simon, H., A. (1977), "The new Science of Management Decision" Englewood Cliffs, NJ: Prentice Hall, 1977

122. Sommerville, I. (1995). *Software engineering*. 5th edition. Addison Wesley.

123. Sosunovas S, Vasilecas O (2006). Precise Notation for Business Rules Templates. In Proceedings of 7th International IEEE Baltic Conference on Databases and Information Systems (pp. 55-60) IEEE.

124. Standish Group (2003): Chaos Chronicles Version 3.0.: West Yarmouth, MA

125. Steen Bas, Pires, L.F., Iacob, M. E. (2010, October). Automatic Generation of Optimal Business Processes from Business Rules. In Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2010 14th IEEE International (pp. 117-126). IEEE.

126. Suchman, L., and Jordan, B. (1990). Interactional troubles in face-to-face survey interviews. *Journal of the American Statistical Association*, *85*(409), 232-241.

127. Sutcliffe, A. (2002). *User-centred requirements engineering*. Springer.

128. Sutcliffe, A. G., Maiden, N. A., Minocha, S., & Manuel, D. (1998). Supporting scenario-based requirements engineering. *Software Engineering, IEEE Transactions on*, *24*(12), 1072-1088.

129. Turban E., Aronson Jay E. (1998), “Decision Support Systems and Intelligent Systems” (5th edition) January 1998, Publisher: Prentice Hall

130. van Lamsweerde, A. (2000, June). Requirements engineering in the year 00: A research perspective. In *Proceedings of the 22nd international conference on Software engineering* (pp. 5-19). ACM.

131. van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on* (pp. 249-262). IEEE.

132. van Lamsweerde, A. (2004, September). Goal-oriented requirements enginering: a roundtrip from research to practice engineering. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International* (pp. 4-7). IEEE.

133. van Lamsweerde, A., and Willemet, L. (1998). Inferring declarative requirements specifications from operational scenarios. *Software Engineering, IEEE Transactions on*, *24*(12), 1089-1114.

134. Wetherbe J.C. (1991). Executive Information Requirements: Getting it Right, MIS Quarterly, (pp. 51-65).

135. Winter, R., and Strauch, B. (2003, January). A method for demand-driven information requirements analysis in data warehousing projects. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on* (pp. 9-pp). IEEE.

136. Winter, R., and Strauch, B. (2004, March). Information requirements engineering for data warehouse systems. In *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 1359-1365). ACM.

137. Wright, P. (1992). Whats in a scenario?. *ACM SIGCHI Bulletin*, *24*(4), 11-12.

138. Yu, E. S. (1997, January). Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on* (pp. 226-235). IEEE.

139. Yu, E. S., and Mylopoulos, J. (1994, May). Understanding why in software process modelling, analysis, and design. In *Proceedings of the 16th international conference on Software engineering* (pp. 159-168). IEEE Computer Society Press.

140. Zave, P. (1997). Classification of research efforts in requirements engineering.*ACM Computing Surveys (CSUR)*, *29*(4), 315-321.

141. Zhang, Z. (2007). Effective Requirements Development-A Comparison of Requirements Elicitation techniques. *Tampere, Finland, INSPIRE*, 225-240.

Biography of Author

Deepika Prakash is a research scholar in the Department of Computer Engineering at Delhi technological University (DTU) formerly, Delhi College of Engineering, New Delhi, India. She did her M.Tech from IIIT Bangalore in 2009. Her research interests lie in the area of Requirements Engineering.

She has both teaching (5 years) and industry (more than 2 years) experience. As a research scholar she has taught many courses to undergraduate students. These include Data Warehousing and Data Mining, Programming Fundamentals: C and C++. She has also taken a short course for post graduate students on Software Testing. She has taught Object oriented software engineering to under graduate students at NSIT, New Delhi.

She is currently working as a Software Engineer in Motherson Sumi Infotech and Design Ltd where she is in the Enterprise Data Warehouse project. Her strengths lie in upstream activities in SDLC in Requirements Engineering and Conceptual Modelling and also in development activities using SQL server. In the EDW project, she is involved in ETL operations as well as in the development of the conceptual design models.

Appendix A
List of AYUSH policies

AYUSH(x): x is a AYUSH hospital

p is a set of patients

n is a set of nurses

b is a set of beds

bedocc is a set of occupied beds

ratio(x,y) : ratio of x to y

perc(x, y): percentage of x/y

belongs(x,y) : y belongs to x

- Total Patient to bed ratio should not be higher than (1:6)

  For every AYUSH hospital, the ratio of the number of patients to beds should be less than or equal to 1:6

  ∀x∃p∃b [AYUSH(x)→LE (ratio(count(p), count(b)), ratio(1,6)) AND belongs(x,p) AND belongs(x,b)]

-

	Category of bed	Bed/Nurse Ratio (acceptable standard)
a)	General	8:1
b)	Semi-Private	4:1
c)	Private	4:1

  For every general ward, semi-private ward and private ward the ration of bed to nurses must be equal to 8:1, 4:1, 4:1 respectively.

G(x): x is a general ward

S(x): x is a semi-private ward

pvtW(x): x is a private ward

gwb is a set of beds in a general ward

spwb is a set of beds in a semi-private ward

pwb is a set of beds in a private ward

gwn is a set of nurses in a general ward

spwn is a set of nurses in a semi-private ward

pwn is a set of nurses in a private ward

$$\forall x \exists gwb\ \exists gwn \begin{bmatrix} G(x) \rightarrow \text{EQ}(\text{ratio}(\text{count}(gwb), \text{count}(gwn)), \text{ratio}(8,1)) \\ \text{AND belongs}(x, gwb) \text{ AND belongs}(x, gwn) \end{bmatrix}$$

$$\forall x \exists spwb\ \exists spwn \begin{bmatrix} S(x) \rightarrow \text{EQ}(\text{ratio}(\text{count}(spwb), \text{count}(spwn)), \text{ratio}(4,1)) \\ \text{AND belongs}(x, spwb) \text{ AND belongs}(x, spwn) \end{bmatrix}$$

$$\forall x \exists pwb\ \exists pwn \begin{bmatrix} pvtR(x) \rightarrow \text{EQ}(\text{ratio}(\text{count}(pwb), \text{count}(pwn)), \text{ratio}(4,1)) \\ \text{AND belongs}(x, pwb) \text{ AND belongs}(x, pwn) \end{bmatrix}$$

- Bed occupancy rate (Norm 50%)

  For every AYUSH hospital, the percentage of occupied beds out of the number of beds must be 50%

$$\forall x \exists bedocc \exists b \begin{bmatrix} \text{AYUSH}(x) \rightarrow \text{EQ}\ (\text{perc}(\text{count}(bedocc), \text{count}(b)),50) \\ \text{AND belongs}(x, bedocc) \text{ AND belongs}(x, b) \end{bmatrix}$$

- All the beds in the hospital should be equipped with adequate facilities such as bedside lockers, bedside stools, mattresses, pillows etc.

bed(x) : x is a bed

lock(x,y) : y is a lock of x

stool (x,y) : y is a stool of x

matt(x,y) : y is a mattress of x

pillow(x,y) : y is a pillow of x

$$\forall x \exists a \exists b \exists c \exists d \begin{bmatrix} \text{bed}(x) \rightarrow \text{lock}(x,a) \text{ AND stool}(x,b) \\ \text{AND matt}(x,c) \text{ AND pillow}(x,d) \end{bmatrix}$$

- A semi-private ward should not have more than three beds. A minimum area of 200 Sq.ft. will be required for two beds.

  For every semi-private ward, the number of beds must be less than or equal to 3. An area of 200 Sq.ft is needed for two beds.

  $\forall x \exists spwb$[S(x)$\rightarrow$LEQ(count(spwb),3)) AND belongs(x,spwb)]

  $\forall x \exists spwb$[S(x)$\rightarrow$GEQ(area(x),200) AND EQ(count(spwb),2) AND belongs(x,spwb)]

- A private ward will house only one bed and an attendance bed. It should have attached toilet cum bathroom.

  attbed is a set of attendance beds

  toiletbath(x): x is an attached *toilet cum bathroom*

$$\forall x \exists \text{attbed} \exists \text{tb} \exists \text{pwb} \begin{bmatrix} \text{pvtW}(x) \rightarrow \text{EQ(count(pwb),1) AND EQ(count(attbed),1)} \\ \text{AND toiletbath(tb) AND belongs}(x,\text{tb}) \\ \text{AND belongs}(x,\text{attbed}) \text{ AND belongs}(x,\text{pwb}) \end{bmatrix}$$

- The hospital should have a separate compounding section with minimum two qualified compounders.

  For every AYUSH hospital there exists a compounding section and a minimum of two qualified compounders.

  compset: is a set of qualified compounders

  compSec(x): x is a compounding section

  $$\forall x \exists y \exists compset\ [AYUSH(x) \rightarrow compSec(y)\ AND\ GEQ(count(compset),2)\ AND\ belongs(y,compset)]$$

- The Hospital should have diagnostic facilities of routine nature, such as blood, urine, stool, sputum examination.

  blooddiag(x,y) : y is a diagnostic facility for blood examination in x

  urinediag(x,y) : y is a diagnostic facility for urine examination in x

  stooldiag(x,y) : y is a diagnostic facility for stool examination in x

  sputumdiag(x,y) : y is a diagnostic facility for sputum examination in x

  $$\forall x \exists y \exists z \exists m \exists n \begin{bmatrix} AYUSH(x) \rightarrow blooddiag(x,y)\ AND\ urinediag(x,z) \\ AND\ stooldiag(x,m)\ AND\ sputumdiag(x,n) \end{bmatrix}$$

SPACE:

▪ Out Patient Department - (Consulting rooms, Compounding room, Pathology Laboratory, treatment Rooms, waiting lounges, toilet)

OPD(x): x is an outpatient department

consul(x): x is a consulting room

comp(x): x is a compounding room

pathlab(x): x is a pathology laboratory

trtR(x): x is a treatment room

lounge(x): x is a waiting lounge

toilet(x): x is a toilet

$$\forall x \exists a \exists b \exists c \exists d \exists e \exists f \begin{bmatrix} \text{OPD}(x) \rightarrow \text{consul}(a) \text{ and comp}(b) \text{ and} \\ \text{pathlab}(c) \text{and trtR}(d) \text{ and lounge}(e) \text{ and toilet}(f) \\ \text{AND belongs}(x, a) \text{AND belongs}(x, b) \text{AND belongs}(x, c) \\ \text{AND belongs}(x, d) \text{AND belongs}(x, e) \text{AND belongs}(x, f) \end{bmatrix}$$

- In Patient Department-

Semi-private Ward	200 Sq. ft. for 2 beds
	300 Sq. ft. for 3 beds

The area of a semi-private ward should be 200 Sq. ft. for 2 beds and 300 Sq. ft. for 2 beds.

$$\forall x \exists spwb \begin{bmatrix} S(x) \rightarrow (\text{EQ}(\text{area}(x),200) \text{ AND } \text{EQ}(\text{count}(spwb),2)) \\ \text{OR } (\text{EQ}(\text{area}(x),300) \text{ AND EQ}(\text{count}(spwb),3)) \text{ AND belongs}(x, spwb) \end{bmatrix}$$

Private Room 200 Sq. ft.

pvtR(x) : x is a private room

$$\forall x[\text{pvtR}(x) \rightarrow \text{EQ}(\text{area}(x),200]$$

- **Nurses duty Room** 100 Sq. ft. per ward.

ward(x): x is a ward

nurroom(x): x is a nurse duty room

For every ward, the nurses duty room within the ward must be 100 Sq. ft.

$$\forall x \exists y[\mathrm{ward}(x) \rightarrow \mathrm{nurroom}(y) \mathrm{AND}\ \mathrm{EQ}(\mathrm{area}(y), 100) AND\ belongs(x, y)]$$

- **Compounding Room** 500 Sq. ft.

comp(x): x is a compounding room

$$\forall x[\mathrm{comp}(x) \rightarrow \mathrm{EQ}(\mathrm{area}(x),500)]$$

Medical staff requirements for general treatment hospital

Minimum two full time doctors with recognised Post graduate qualifications preferably in Kayachikitsa, Shalya, Shalakya, Panchakarma, StriRoga & Prasuti Tantra for Ayurveda, and Maulijat for Unani

Ay(x): x is a Ayurveda centre

UN(x): x is a Unani centre

degree(x, mauli) : x has degree in maulijat

degree(x, prasutiTantra) : x has degree in prasuti Tantra

degree(x, striyoga) : x has degree in striyoga

degree(x, panchakarma) : x has degree in panchakarma

degree(x, kayachikitsa) : x has degree in kayachikitsa

degree(x, shalya) : x has degree in shalya

degree(x, shalakya) : x has degree in shalakya

FTdocA(x): x is a full time doctor of Ayurveda

FTdocU(x): x is a full time doctor of Unani

ftdA is a set of full time doctors of Ayurveda

ftdU is a set of full time doctors of Unani rd is a set of resident doctors

The full time doctor for Unani must have a degree in maulijat. Every Unani hospital must have at least two full time doctors.

$$\forall \mathrm{x}[\mathrm{FTdocU}(x) \rightarrow \mathrm{degree}(\mathrm{y}, \mathrm{mauli})]$$

$$\forall \mathrm{x} \exists \mathrm{ftdU}\ [Un(\mathrm{x}) \rightarrow \mathrm{GEQ}(\mathrm{count}(\mathrm{ftdU}\ ), 2)\ \mathrm{AND\ belongs}(\mathrm{x}, \mathrm{ftdU}\ )]$$

The full time doctors of Ayurveda must have a degree in Kayachikitsa, Shalya, Shalakya, Panchakarma, StriRoga & Prasuti Tantra. Every Ayurvedic hospital must have at least two full time doctors.

$$\forall y \begin{bmatrix} \mathrm{FTdocA(y)} \rightarrow \mathrm{degree(y, prasutiTantra)\ OR} \\ \mathrm{degree(y, striyoga) OR\ degree(y, panchakarma) OR\ \ degree(y, kayachikitsa) OR} \\ \mathrm{degree(y, shalya)\ OR\ degree(y, shalakya)} \end{bmatrix}$$

$$\forall \mathrm{x} \exists \mathrm{ftdA}\ [\mathrm{Ay(x)} \rightarrow \mathrm{GEQ}(\mathrm{count}(\mathrm{ftdA}\ ), 2\ \mathrm{ANDbelongs}(\mathrm{x}, \mathrm{ftdA}\ )]$$

Para-Medical staff

nurse(x): x is a nurse

pharmaA(x): x is a pharmacist in Ayurveda

pharmaY(x): x is a pharmacist in Yoga

pharmaU(x): x is a pharmacist in Unani

pharmaN(x): x is a pharmacist in Naturopathy

degree(x, nursing): x has recognised nursing degree

a. Nurses with recognised nursing qualification.

$$\forall \mathrm{x}[\mathrm{nurse(x)} \rightarrow \mathrm{degree(x, nursing)}]$$

b. Pharmacist:- Recognised qualification in Pharmacy education of concerned system

A pharmacist must have specialized in Ayurveda or Yoga or Unani or Naturopathy.

pharmacist(x):x is a pharmacist

$$\forall x\left[pharmacist(x)\rightarrow \begin{matrix} pharmaA(x)\ OR\ pharmaY(x) \\ OR\ pharmaU(x) OR\ pharmaN(x) \end{matrix}\right]$$

SPECIALITY TREATMENT IN AYURVEDA

Panchakarma Therapy center

P(x) : x is Panchakarma Therapy center

S1(x) : x is snehan room

S2(x) : x is swedan room

S3(x) : x is snodhan room

S4(x) : x is karma room

S5(x) : x is duty room

S6(x) : x is toilet and bathroom room

Space: (Minimum space)

- Snehan Room: - 100 Sq. ft.

  $\forall x[S1(x) \rightarrow EQ(area(x),100)]$

- Swedan Room: - 100 Sq. ft.

  $\forall x[S2(x) \rightarrow EQ(area(x),100)]$

- Shodhan Room: - 100 Sq. ft. with attached toilet cum bath room.

  $\forall x \exists y[S3(x) \rightarrow EQ(area(x),100)\ AND\ S6(y)\ AND\ belongs(x,y)]$

- Room for other Karmas - 100 Sq. ft.

$\forall x[S4(x) \rightarrow EQ(area(x),100)]$

- Duty/Staff Room:- 100 Sq. ft.

$\forall x[S5(x) \rightarrow EQ(area(x),100)]$

- Toilet & Bath Room- 100 Sq. ft.

$\forall x[S6(x) \rightarrow EQ(area(x),100)]$

Indoor department

G(x) : x is a general ward

pvtR(x) : x is a private room

S(x) : x is a semi-private ward

O(x) : x is O.P.D.

M(x) : x is Pharmacy

D(x) : x is a Dispensing room

K(x): x is a Kitchen

pvtS: s is a set of private rooms

gwb is a set of general ward beds

gwp is a set of general ward patients

spwb is a set semi-private ward beds

spwp is a set of semi-private ward patients

pwb is a set private ward beds

pwp is a set of private ward patients

General Wards – 600 Sq. ft.(Minimum 10 beds)	At least 30 patients
Private Rooms - 200 Sq. ft.(At least 4)	At least 4 patients
Semi Pvt. Ward -200 Sq. ft (2 bedded)	At least 10 patients
-300 Sq. ft (3 bedded)	At least 6 patients

O.P.D.	– 300 Sq. ft.
Pharmacy/Store	- 300 Sq. ft
Dispensing Room	- 200 Sq. ft.
Kitchen	- 100 Sq. ft.

1. For every general ward, the area must be 600 Sq. ft. with a minimum of 10 beds and 30 patients.

$$\forall x \exists gwb \exists gwp \begin{bmatrix} G(x) \rightarrow EQ(area(x),600) \text{ AND } GEQ(count(gwb),10) \\ \text{AND } GEQ(count(gwp),30) \text{ AND } belongs(x, gwb) \\ \text{AND } belongs(x, gwp) \end{bmatrix}$$

2. For every private room, the area must be 200 Sq. ft. with a minimum of 4 beds and 4 patients.

$$\forall x \exists pwb \exists pwp \begin{bmatrix} pvtR(x) \rightarrow EQ(area(x),200) \\ \text{AND } GEQ(count(pwb),4) \text{ AND } GEQ(count(pwp),4) \\ \text{AND } belongs(x, pwb) \text{ AND } belongs(x, pwp) \end{bmatrix}$$

3. For every semi-private ward, the area must be either 200 Sq. ft. with 2 beds and a minimum of 10 patients or 300 Sq. ft. with 3 beds and a minimum of 6 patients.

$$\forall x \exists spwb \exists spwp \begin{bmatrix} S(x) \rightarrow (EQ(count(spwb),2) \text{ AND } EQ(area(x),200) \\ \text{AND } GEQ(count(spwp),10)) \\ \text{OR } (EQ(count(spwb),3) \text{ and } EQ(area(x), 300) \text{AND} \\ GEQ(count(spwp),6)) \text{ AND } belongs(x, spwb) \text{ AND } belongs(x, spwp) \end{bmatrix}$$

4. $\forall x[O(x) \rightarrow EQ(area(x),300)]$

5. $\forall x[M(x) \rightarrow EQ(area(x),300)]$

6. $\forall x[D(x) \rightarrow EQ(area(x),200)]$

7. $\forall x[K(x) \rightarrow EQ(area(x),100)]$

Staff

sanst(x,y): y is a sanitation staff of x

reckep(x,y): y is a record keeper cum clerk of x

kitS(x,y) : y is kitchen staff of x

spM(x) : x is a Male Panchakarma specialist

spF(x): x is a Female Panchakarma specialist

setN24 is a set of round the clock nurses

Psp is a set of specialists

Pmo is a set of rmo

Patt is a set of attendants

Pnurse is a set of nurses

ph is a set of pharmacists

- Panchakarma Specialists (Male & Female) - 2 with (M.D.(Ayu) qualification in Kayachikitsa or Panchakarma)

One male Panchakarma Specialist with required qualification

$$\forall y \left[\begin{array}{c} spM(y) \rightarrow \\ \left(\mathrm{degree(y, panchakarma)}\, or\, \mathrm{degree(y, kayachikitsa)}\right) \end{array} \right]$$

One female Panchakarma Specialist with required qualification

$$\forall y\left[\begin{array}{c} spF(y) \rightarrow \\ (\text{degree}(\text{y}, \text{panchakarma}) or\ \text{degree}(\text{y}, \text{kayachikitsa})) \\ \end{array}\right]$$

- Resident Medical Officer - 1

  ∀x∃Pmo [P(x) →EQ(count(Pmo),1) and belongs(x, Pmo)]

- Masseurs/PanchakarmaAttendents - 4

  ∀x∃$Patt$ [P(x) →EQ(count(Patt),4) and belongs(x, Patt)]

- Staff Nurses - 4 (Round the clock)

  ∀x∃setN24[P(x) → EQ(count(setN24),1) and belongs(x, setN24)]

- Pharmacist - 2

  ∀x∃ph[P(x) → EQ(count(ph),2) and belongs(x, ph)]

- Sanitation Staff

  $\forall x \exists y[\text{P(x)} \rightarrow \text{sanst(x, y)}]$

- Record Keeper cum clerk

  $\forall x \exists y[\text{P(x)} \rightarrow \text{reckep(x, y)}]$

- Kitchen Staff

  $\forall x \exists y[\text{P(x)} \rightarrow kitS(\text{y})\ AND\ belongs(x, y)]$

List of minimum equipment

fom(x,y) : y is hot fomentation instrument in x

sy(x,y) : y is sirodhara yantra and appliance in x

vy(x,y) : y is vastiyantra in x

batt(x,y) : y is droni bath tub in x

gey(x,y) : y is geyser in x

tub(x,y) : y is wooden swedana table for Massage in x

phyl(x,y) : y is physiotherapy instruments in x

hotplate(x,y) : y is hot plate in x

utt(x,y) : y is essential utensil for panchakarma procedures in x

weigh(x,y) : y is weighing machine in x

vess(x,y) : y is a vessel for avaghanasweda in x

inh(x,y) : y is steam inhaler in x

nyapp(x,y) : x is nyasa applicator in x

prscope(x,y) : y is protoscope in x

diag(x,y) : y is diagnostic sets in x

bll(x,y) : y is tool for blood letting in x

wptset is a set of wooden Panchakarma Table for Massage

fptset is a set of fibre Panchakarma Table for Massage

wstset is a set of wooden swedana Table for Massage

fstset is a set of fibre swedana Table for Massage

sry is a set of sirodhara yantra and appliances

- Wooden/Fibre Panchakarma Table for Massage/Pizhichil - 2 Nos.

  For every Panchakarma therapy center, there must be a total of 2 Wooden/Fibre tables for Massage/Pizhichil.

$$\forall x \exists fptset \exists wptset \begin{bmatrix} P(x) \rightarrow \\ EQ(sum(count(wptset), count(fptset)),2) \\ \text{AND belongs(x, wptset)AND belongs(x, fptset)} \end{bmatrix}$$

- Wooden/Fibre Swedana Table - 2 Nos.

For every Panchakarma therapy center, there must be a total of 2 Wooden/Fibre Swedana tables.

$$\forall x \exists wstset \exists fstset \begin{bmatrix} P(x) \rightarrow \\ EQ(sum(count(wstset), count(fstset)), 2) \\ \text{AND belongs(x, wstset)AND belongs(x, fstset)} \end{bmatrix}$$

- Hot fomentation instruments (Whole body SwedanaYantra) eg. Souna bath

$$\forall x \exists y [P(x) \rightarrow fom(x, y)\ ]$$

- SirodharaYantra & appliances - 2 Nos.

$$\forall x \exists sry [P(x) \rightarrow EQ(count(sry), 2) AND\ belongs(x, sry)]$$

- VastiYantra, Droni/Bath tub, Geyser, Tub for bath, Physiotherapy Instruments, Hot plate, Essential utensils for Panchakarma procedures, Weighing Machine, vessels for AvagahanaSweda, Steam inhaler, Nasya applicator, Protoscope, Diagnostic sets, Tools for blood letting.

$$\forall x \exists a \exists b \exists c \exists d \exists e \exists f \begin{bmatrix} P(x) \rightarrow vy(x, a)\ AND\ batt(x, b)\ AND\ gey(x, c) AND \\ tub(x, d) AND\ hyI(x, e)\ AND\ nyapp(x, f) \end{bmatrix}$$

$$\forall x \exists a \exists b \exists c \exists d \exists e \exists f \exists g \exists h \left[\begin{pmatrix} P(x) \rightarrow hotplate(x, a)\ AND\ utt(x, b)\ AND\ weigh(x, c)\ AND\ vess(x, d) \\ AND\ inh(x, e)\ AND\ prscope(x, f)\ AND\ diag(x, g)\ AND\ bll(x, h) \end{pmatrix} \right]$$

YOGA AND NATUROPATHY

Space

yog(x) : x is yoga centre

nat(x): x is naturopathy

yh(x) : x is yoga hall

hydro(x): x is a hydropathy section

bath(x): x is a bathroom

stbath(x): x is a steam bath room

ene(x): x is a enema room

wc(x): x is a toilet

mud(x): x is a mud therapy section

room(x): x is a room

terrace(x): x is a terrace

mass(x): x is a massage section

table(x): x is a table

chromo(x): x is a chromotherapy section

elec(x): x is a electrotherapy section

refl(x): x is a reflexology section

Nk(x): x is a naturopathy kitchen

lib(x): x is a library cum reading room

gam(x): x is a indoor gaming facility

br is a set of bathrooms

sbr is a set of Steam bath rooms

eneR is a set of enema rooms

toilet is a set of toilets

rmMT is a set of rooms in a mud therapy section

rmMS is a set of rooms in a Massage section

trrMT is a set of terraces in a mud therapy section *tabMS is a set of tables in a massage section*

tab is a set of tables

- Total Required - 3600 Sq. ft.

$$\forall x \exists y \exists z[AYUSH(x) \rightarrow yog(y)\ AND\ nat(z)\ AND\ EQ(sum(area(y), area(z)), 3600)]$$

- Hydrotherapy Section with 4 Bathrooms, 1 Steam Bath Room, 2 enema Rooms and 4 Toilets

$$\forall x \exists y \exists br \exists sbr \exists eneR \exists toilet \begin{bmatrix} nat(x) \rightarrow hydro(y) \\ and\ (belongs(y, br)\ AND\ EQ(count(br), 4\ )) \\ and\ (belongs(y, sbr)\ AND\ EQ(count(sbr), 1)) \\ and\big(belongs(y, eneR)\ AND\ EQ(count(eneR), 2)\big) \\ and\ \big(belongs(y, toilet)\ AND\ EQ(count(toilet), 4)\big) \end{bmatrix}$$

- Mud Therapy Section 1 Room + Terrace.

$$\forall x \exists y \exists rmMT\ \exists trrMT \begin{bmatrix} nat(x) \rightarrow mud(y) and\ (belongs(y, rmMT\ ) \\ AND\ EQ(count(rmMT\ ), 1) \\ and\ (belongs(y, trrMT\ )\ AND\ EQ(count(trrMT\ ), 1) \end{bmatrix}$$

- Massage Section 2 Rooms + 4 Tables

$$\forall x \exists y \exists rmMS\ \exists tabMS \begin{bmatrix} nat(x) \rightarrow mass(y) and\ (belongs(y, rmMS\ ) \\ AND\ EQ(count(rmMS\ ), 2)) \\ and\ (belongs(y, tabMS\ ) and EQ(count(tabMS\ ), 4)) \end{bmatrix}$$

- Chromotherapy Section – Terrace

$$\forall x \exists y \exists z[nat(x) \rightarrow chromo(y) AND$$
$$(terrace(z) AND\ (belongs(y, z\ ))]$$

- Electrotherapy & Reflexology Section

$$\forall x \exists y \exists z[nat(x) \rightarrow elec(y) AND\ refl(z)]$$

- Naturopathy Kitchen

$$\forall x \exists y[nat(x) \rightarrow Nk(y)]$$

- Library cum reading room and indoor games facilities

$$\forall x \exists z \exists m[(yog(x)\ or\ nat(x)) \rightarrow lib(z)\ and\ gam(m)]$$

Staff

NPhy(x) : x is a naturopathy physician

YTt(x) : x is a yoga therapist

teac(x) : x is a teacher

trtA(x) : x is a treatment assistant

cook(x) : x is a cook

kh(x) : x is a kitchen helpers

naturePath is a set of naturopathy physicians

yogTher is a set of yoga therapist

treatAss1 is a set of Male treatment assistants

treatAss2 is a set of Female treatment assistants

cookSet is a set of cooks

kithelp is a set of kitchen helpers

- Naturopathy Physicians 2

$$\forall x \exists naturePath[nat(x) \rightarrow EQ(count(naturePath), 2)\ AND\ belongs(x, naturePath)]$$

- Yoga Therapist/Teacher 2

$$\forall x \exists yogTher[yog(x) \rightarrow EQ(count(yogTher), 2)\ AND\ belongs(x, naturePat)]$$

- Treatment Assistants 6 Male & 3 Female

$$\forall x \exists treatAss1 \exists treatAss2[yog(x)\ \ or\ nat(x)$$
$$\rightarrow EQ(count(treatAss1),6)\ AND\ EQ(count(treatAss2),3)$$
$$AND\ belongs(x, treatAss1\ )AND\ belongs(x, treatAss2)]$$

- Cook 1

$$\forall x \exists cookSet[(yog(x)\ or\ nat(x)) \rightarrow EQ(count(cookSet), 1)]$$

- Kitchen helpers/Attendants 2

$$\forall x \exists kithelp[(yog(x)\ or\ nat(x))$$
$$\rightarrow EQ(count(kithelp), 2)\ AND\ belongs(x, kithelp)]$$

Equipment for Yoga and Naturopathy section

Naturopathy section:-

Chromotherapy:-

chlensSet is a set of chromo lens

solthermSet is a set of solar thermolium

colebulb60Set is a set of coloured electric bulbs 60W

colebulb100Set is a set of coloured electric bulbs 100W

colbottlesSet is a set of coloured bottles 1 litre capacity

colglassesSet is a set of coloured glasses

tablampSet is a set of table lamps

chromboxSet is a set of chromo boxes

1. Chromo lens - 1 set

$$\forall x \exists chlensSet[nat(x) \rightarrow belongs(x, chlensSet)\ and\ EQ(count(chlensSet), 1)\ ]$$

2. Solar Thermolium – Sitting/Reclining 1

$$\forall x \exists solthermSet[nat(x) \rightarrow belongs(x, solthermSet)\ and\ EQ(count(solthermSet), 1)]$$

3. Coloured Electric Bulbs(60 watts & 100 watts) 1 set

$$\forall x \exists colebulb60Set \exists colebulb100Set \left[\begin{array}{c} nat(x) \rightarrow\ belongs(x, colebulb60Set) \\ and\ belongs(x, colebulb100Set) \\ andEQ(count(colebulb60Set), 1) \\ and\ EQ(count(colebulb100Set), 1) \end{array} \right]$$

4. Coloured Bottles (1 litre capacity) 1 set

$$\forall x \exists colbottlesSet[nat(x) \rightarrow belongs(x, colbottlesSet)\ and\ EQ(count(colbottlesSet), 1)]$$

5. Table Lamp and Chromo Box 1

$$\forall x \exists tablampSet[nat(x) \rightarrow belongs(x, tablampSet) \text{ and } EQ(count(tablampSet), 1)]$$

$$\forall x \exists chromboxSet[nat(x) \rightarrow belongs(x, chromboxSet) \text{ and } EQ(count(chromboxSet), 1)]$$

YOGA SECTION:-

darrySet is a set of darry

stjugSet is a set of steel jugs

dhpotSet is a set of dhouti pots

lotaSet is a set of lotas

stglassesSet is a set of steel glasses

buckSet is a set of buckets

stcontSet is a set of hot water steel containers

vastshSet is a set of vastradhouti

dandhSet is a set of sand dhouti

sutneti is a set of sutra neti

towelSet is a set of towels

ghesaltSet is a set of cow's ghee and salt

1. Darry - 25

$$\forall x \exists darrySet[yog(x) \rightarrow belongs(x, darrySet) \text{ and } EQ(count(darrySet), 25)]$$

2. Steel Jugs - 25

$$\forall x \exists stjugSet[yog(x) \rightarrow belongs(x, stjugSet) \text{ and } EQ(count(stjugSet), 25)]$$

3. Dhouti pots - 20

$$\forall x \exists dhpotSet[yog(x) \rightarrow belongs(x, dhpotSet)\ and\ EQ(count(dhpotSet), 20)]$$

4. Lota for JalNeti- 20

$$\forall x \exists lotaSet[yog(x) \rightarrow belongs(x, lotaSet)\ and\ EQ(count(lotaSet), 20)]$$

5. Steel glasses - 20

$$\forall x \exists stglassesSet \begin{bmatrix} yog(x) \rightarrow belongs(x, stglassesSet) \\ AND\ EQ(count(stglassesSet), 20) \end{bmatrix}$$

6. Bucket - 20

$$\forall x \exists buckSet[yog(x) \rightarrow belongs(x, buckSet)\ and\ EQ(count(buckSet), 20)]$$

7. Hot water steel container – 2

$$\forall x \exists stcontSet[yog(x) \rightarrow belongs(x, stcontSet)\ and\ EQ(count(stcontSet), 2)]$$

8. VastraDhouti -25

$$\forall x \exists vastshSet[yog(x) \rightarrow belongs(x, \mathrm{vastshSet})\ and\ EQ(count(\mathrm{vastshSet}), 25)]$$

9. DandDhouti - 25

$$\forall x \exists dandhSet[yog(x) \rightarrow belongs(x, dandhSet)\ and\ EQ(count(dandhSet), 25)]$$

10. Sutra Neti - 10

$$\forall x \exists sutnetiSet[yog(x) \\ \rightarrow bbelongs(x, sutnetiSet)\ and\ EQ(count(sutnetiSet), 10)]$$

11. Towels - 200

$$\forall x \exists towelSet[yog(x) \rightarrow belongs(x, towelSet)\ and\ EQ(count(towelSet), 200)]$$

12. Cow's ghee & Salt - 25

$$\forall x \exists ghesaltSet[yog(x) \rightarrow belongs(x, ghesaltSet)\ and\ EQ(count(ghesaltSet), 25)]$$

Unani medicine

Regimental therapy center

Space

Un(x): x is unani regimental therapy center

oalsh(x): x is a room for OaewaIshal

dalk(x) : x is a Room for Dala K

fia(x) : x is a Room for Fasad&IrsaleAlq and AmaleKae

nhm(x): x is a Room for Natolwazarad and Hejamat for male section

nhf(x): x is a Room for Natolwazarad and Hejamat for female section

dis(x): x is a Dispensary Room

- Room for OaewaIshal -180 Sq. ft.

$$\forall \mathbf{x} \exists \mathbf{y}[\mathrm{Un(x)} \rightarrow \mathrm{oaIsh(y)\ AND\ EQ(area(y), 180)}\ ]$$

- Room for Dala K -180 Sq. ft.

$$\forall \mathbf{x} \exists \mathbf{y}[\mathrm{Un(x)} \rightarrow \mathrm{dalk(y)\ AND}\ EQ(\mathrm{area(y)}, 180)]$$

- Room for Fasad&IrsaleAlq and AmaleKae – 100 Sq. ft.

$$\forall \mathbf{x} \exists \mathbf{y} [\mathrm{Un}(\mathrm{x}) \rightarrow \mathrm{fia}(\mathrm{y}) \text{ AND } \mathrm{EQ}(\mathrm{area}(\mathrm{y}), 100)]$$

- Room for Natolwazarad and Hejamat - 100 Sq. ft.

  (One each for male and female section)

$$\forall \boldsymbol{x} \exists \boldsymbol{y} \exists \boldsymbol{z} \begin{bmatrix} Un(x) \rightarrow \big(nhm(y) \text{ AND } (area(y),100)\big)\text{AND} \\ \big(nhf(z) \text{ AND } (area(z),100)\big) \end{bmatrix}$$

- O.P.D. – 300 Sq. ft.

$$\forall \boldsymbol{x} \exists \boldsymbol{y} [Un(x) \rightarrow o(y) \text{ AND } EQ(area(y), 300)]$$

- Dispensary Room - 150 Sq. ft.

$$\forall \boldsymbol{x} \exists \boldsymbol{y} [Un(x) \rightarrow dis(y)\text{AND } EQ(area(y), 150)]$$

Staff

RTS(x): x is a Regimental Therapy Specialist

UnRMO(x): x is a RMO for Unani

thrass(x): x is Therapy Assistants

Unphar(x): x is anUnani pharmacist

Unsanst(x): x is anUnani sanitation staff

Unreckep(x): x is anUnani record kepper

degree(x, kulliyat): x has an M.D.(Unani) in kulliyat

degree(x,moalijit): x has an M.D.(Unani) in moalijit

Mgender(x, male): x is a male

Fgender(x, female): x is a female

Unrts is a set of Regimental Therapy Specialists

Unmo is a set of RMOs

thr is a set of therapy assistants

Unph is a set of pharmacist

- Regimental Therapy Specialists - 2 (With M.D.(Unani) in Kulliyat/MoalijitHefzaneSehal) (One male & One female)

$$\forall \boldsymbol{x} \exists \boldsymbol{y} [Un(x) \rightarrow RTS(y)\ AND\ \left(degree(y, kulliyat)\ OR\ degree(y, moalijit)\right)$$
$$AND\ Mgender(y, male) \text{ AND belongs(x, y) }]$$

$$\forall \boldsymbol{x} \exists \boldsymbol{y} [Un(x) \rightarrow RTS(y) \text{ AND } (degree(y, kulliyat)\ OR\ degree(y, moalijit)\ )$$
$$\text{AND } Fgender(y, female) \text{ AND belongs(x, y) }]$$

- R.M.O - 2

$$\forall x \exists Unmo [Un(x) \rightarrow EQ(count(Unmo), 2) \text{ AND belongs(x, y)}]$$

- Therapy Assistants/Masseur - 4

$$\forall x \exists thr [Un(x) \rightarrow EQ(count(thr), 4) \text{ AND belongs(x, y)}]$$

- Pharmacist - 1

$$\forall x \exists y [Un(x) \rightarrow Unphar(y)\ ]$$

- Sanitation Staff

$$\forall x \exists y [Un(x) \rightarrow Unsanst(y)\ ]$$

- Record Keeper

$$\forall x \exists y [Un(x) \rightarrow Unreckep(y)]$$

Equipments

wtabset is a set of wooden table

wchset is a set of wooden special therapy chair

flset is a set of focal light

sbchse is a set of sitz bath chair

geyset is a set of geysers

inhset is a set of steam inhalers

akhset is a set of basic equipment/instruments for Fasad, AmlaeKae, Hejamat

bpset is a set of B.P. instrument

bll(x): x is a leech for blood letting

- Wooden Table - 4

$$\forall x \exists wtabset[Un(x) \rightarrow (EQ(count(wtabset), 4)) \text{ AND belongs(x,} wtabset)]$$

- Wooden Special Therapy Chairs - 4

$$\forall x \exists wchset[Un(x) \rightarrow (EQ(count(wchset), 4)) \text{ AND belongs(x,} wchset)]$$

- Focal light - 4

$$\forall x \exists flset[Un(x) \rightarrow (\ EQ(count(flset), 4)) \text{ AND belongs(x,} flset)]$$

- Sitz bath chairs - 4

$$\forall x \exists sbchset[Un(x) \rightarrow (EQ(count(sbchset), 4)) \text{ AND belongs(x,} sbchset)]$$

- Basic OT equipment

$$\forall x \exists y \left[Un(x) \rightarrow \left(oteuip(y) \right) \right]$$

- Geysers - 2

$$\forall x \exists geyset [Un(x) \rightarrow (EQ(count(geyset), 2)) \text{ AND belongs(x}, geyset)]$$

- Steam inhaler   - 1

$$\forall x \exists y [Un(x) \rightarrow (inh(y)))]$$

- Basic equipment/instruments for Fasad, AmlaeKae, Hejamat   - 2 sets

$$\forall x \exists akhset [Un(x) \rightarrow (EQ(count(akhset), 2)) \text{ AND belongs(x}, akhset)]$$

- Leeches for blood letting

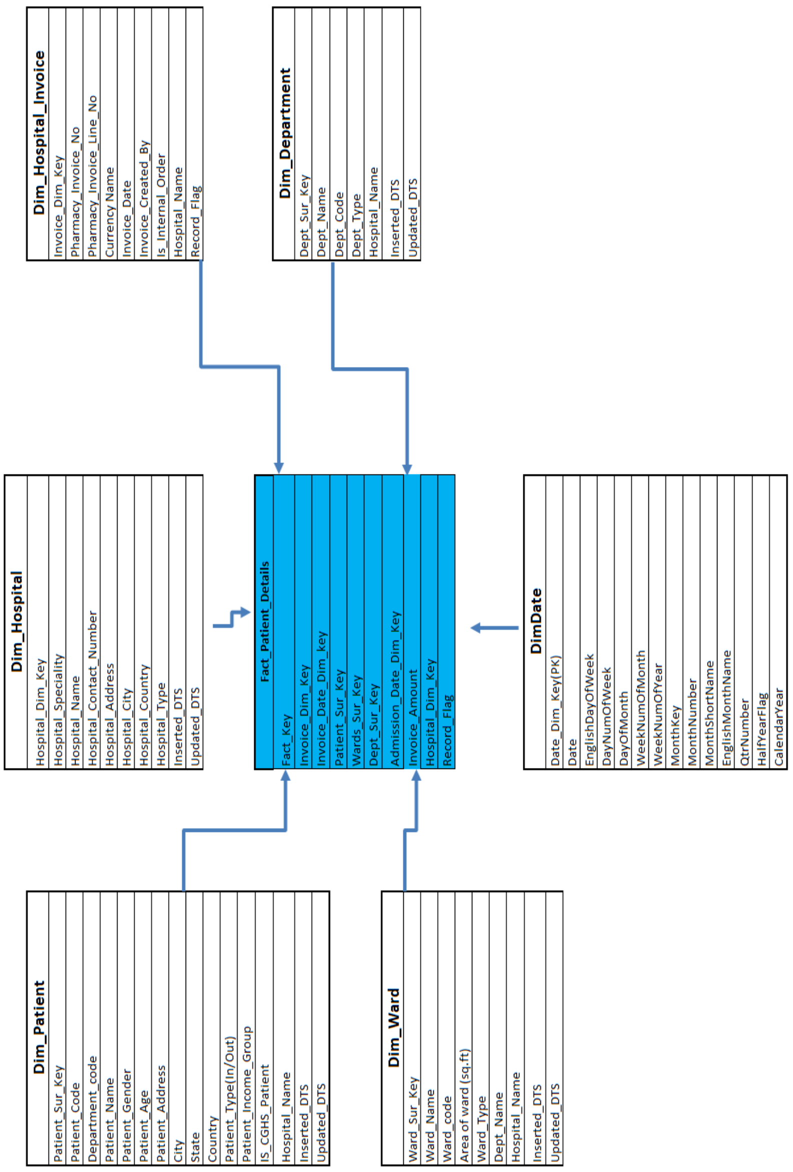
$$\forall x \exists y \left[Un(x) \rightarrow \left(bll(y) \right) \right]$$

- B.P.Instruments - 2

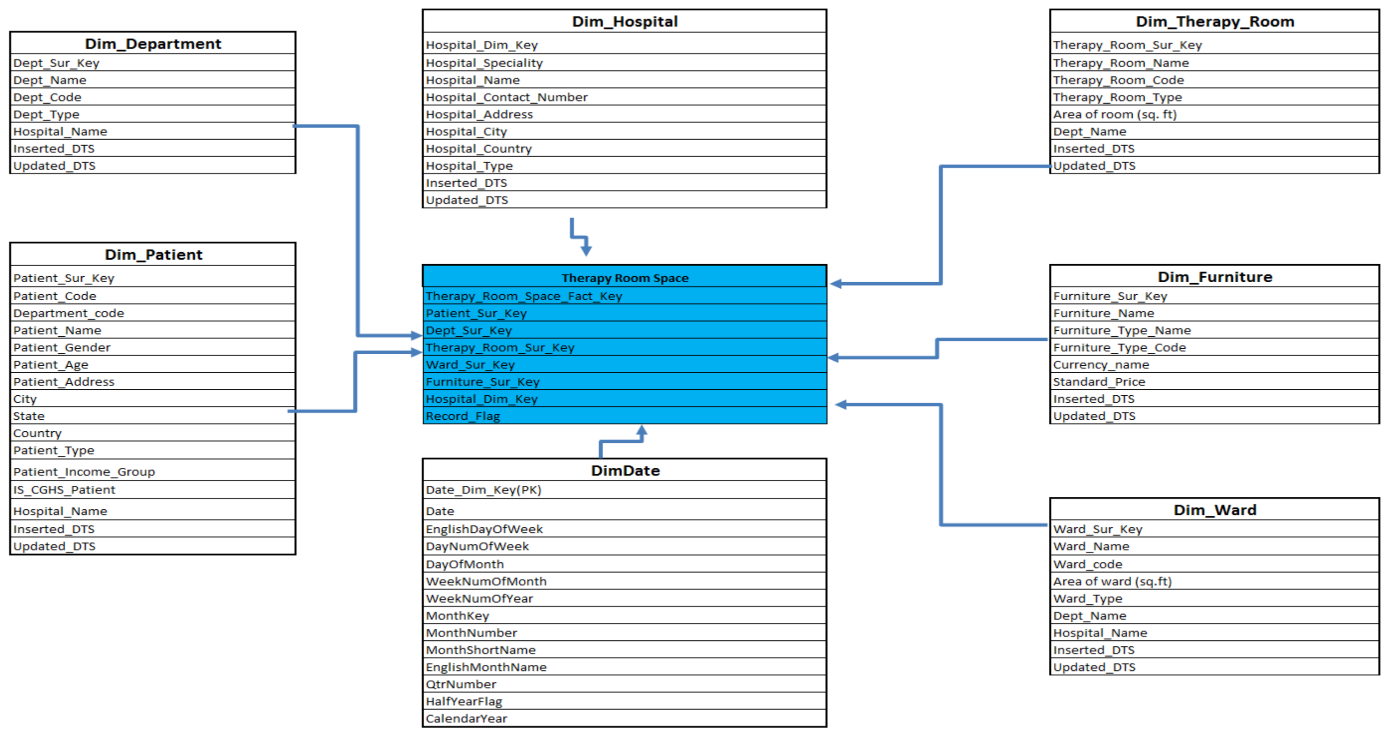
$$\forall x \exists bpset [Un(x) \rightarrow (EQ(count(bpset), 2\ )) \text{ AND belongs(x}, bpset)]$$

Appendix B

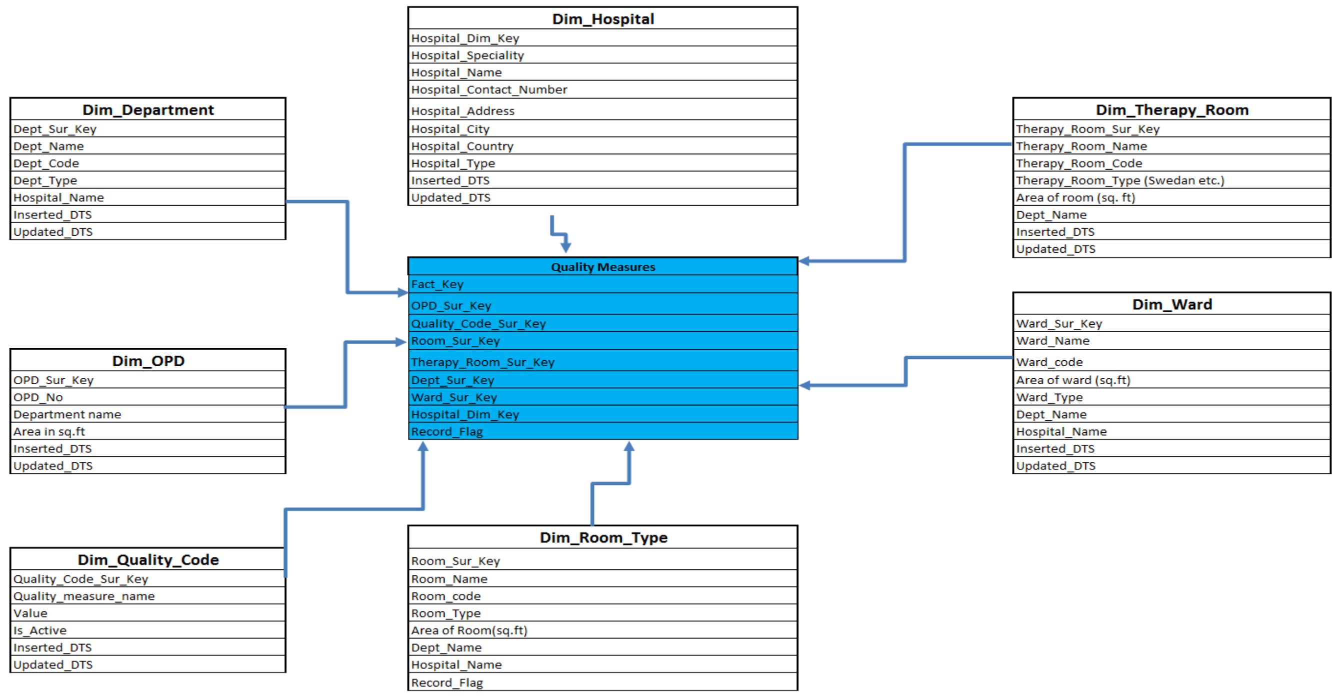
Integrated Data Warehouse Schema



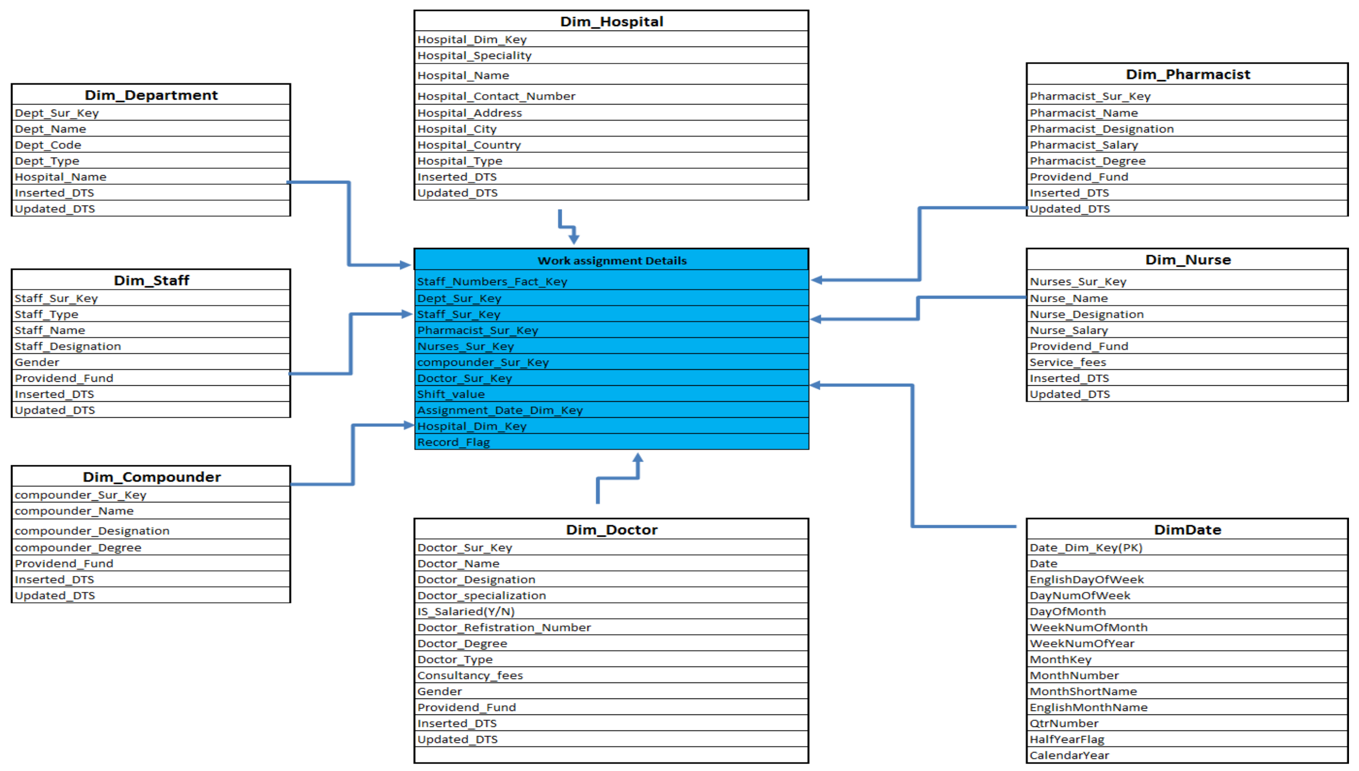
Note: Ward_Type can take values from private/semiprivate/general



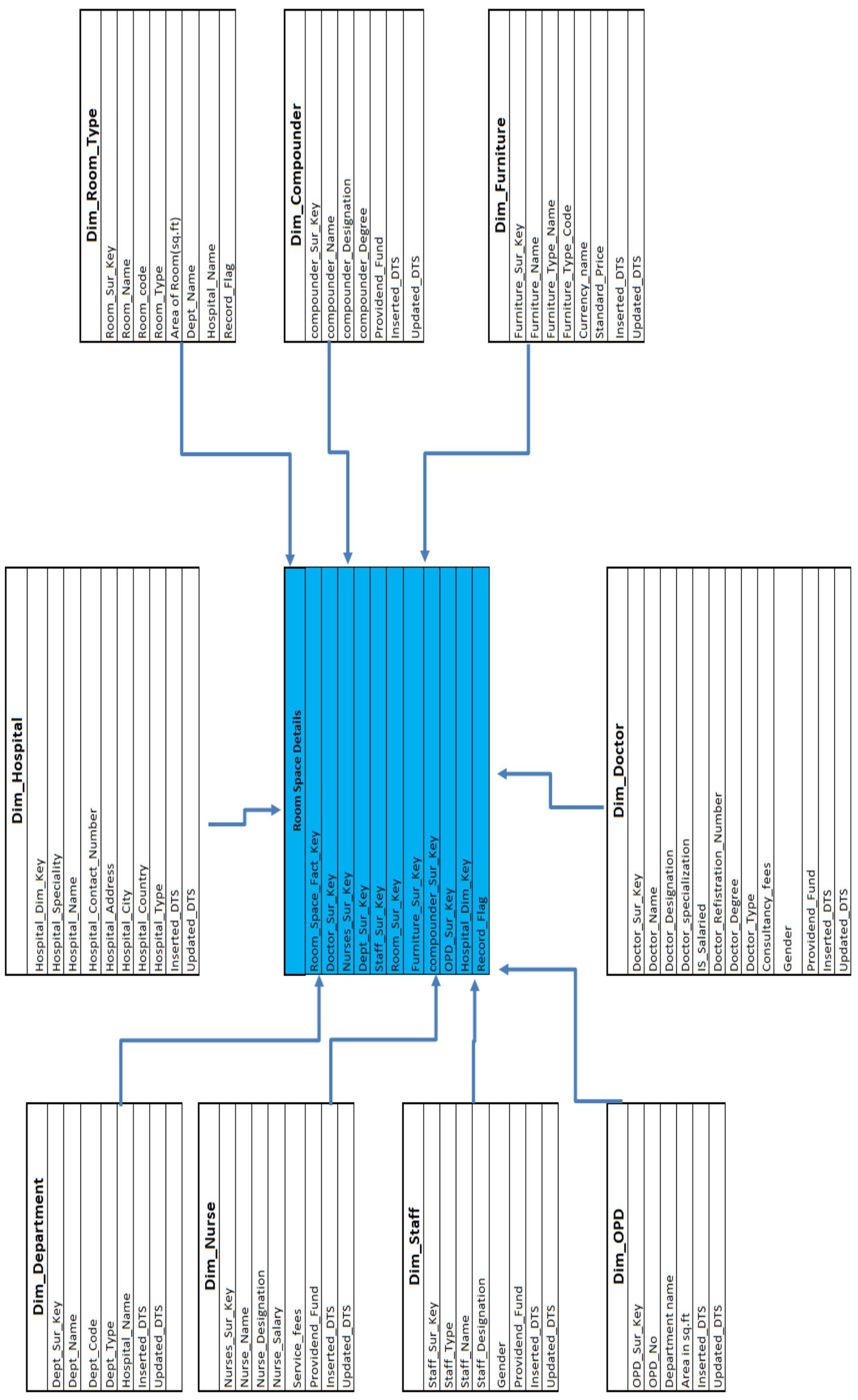
Note: Patient_Type can take values from Inpatient/Outpatient
Note: Therapy_Room_Type can take values fromSwedan Room, Snedan room, Snodhan room etc.

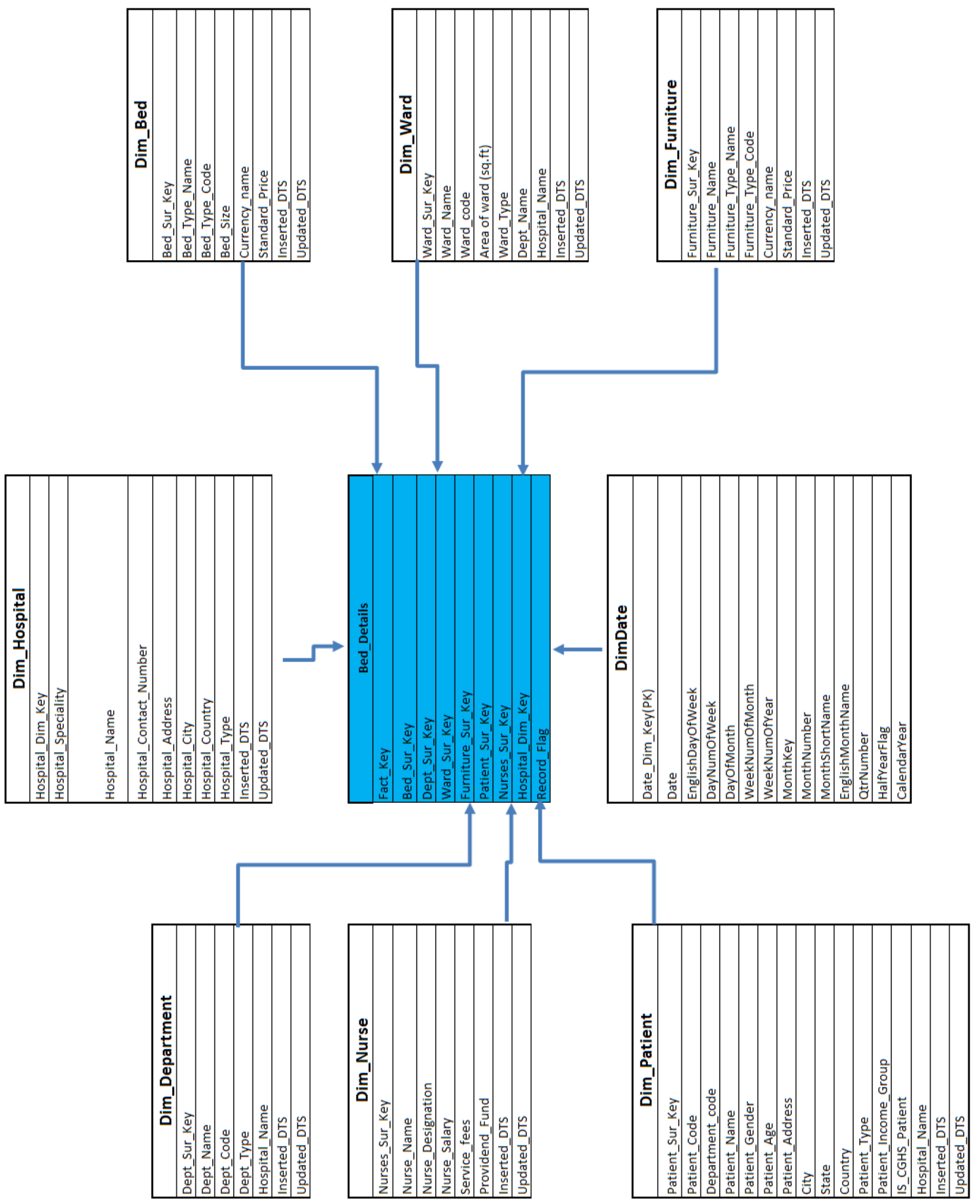


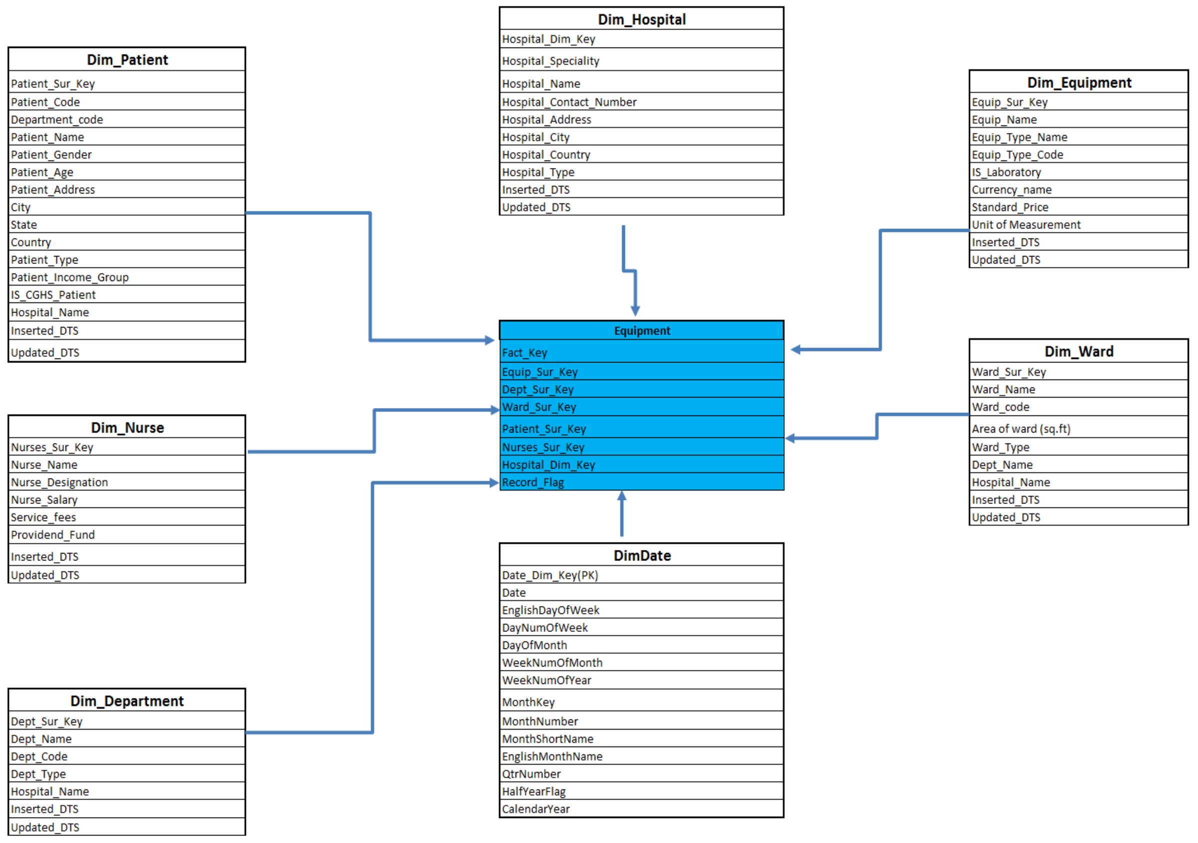
Note: Room_Type can take values from nurses duty room/doctor duty room/compunding room/Kitchen/Dispensing Room
Note: Quality_measure_Name can take values from Labour Cost, Build time, Rental Cost etc.

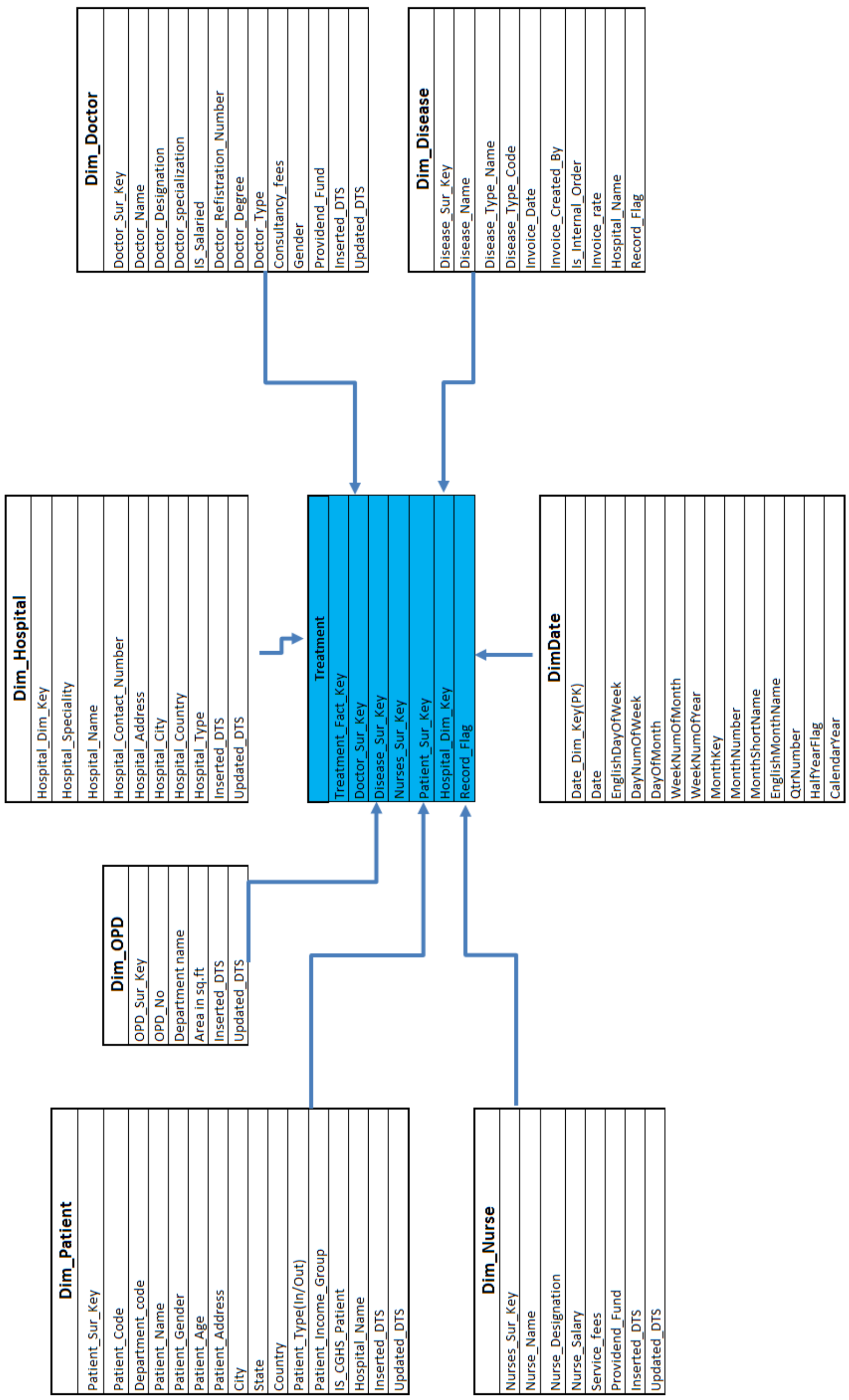


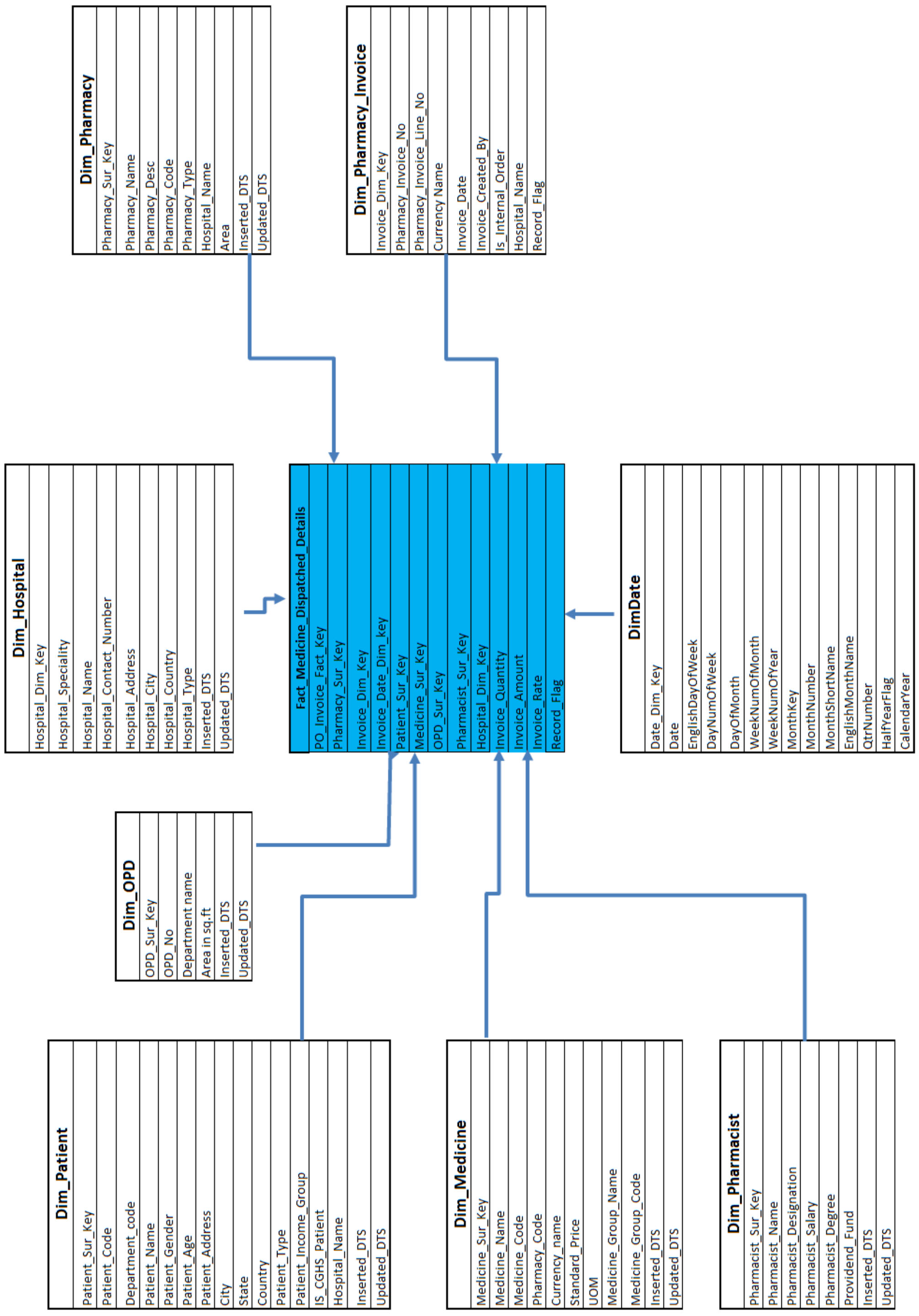
Note: Doctor_Type can take values from full time/residentRoom
Note: Staff_Type can take values from RMO,Santitation, Attendant etc

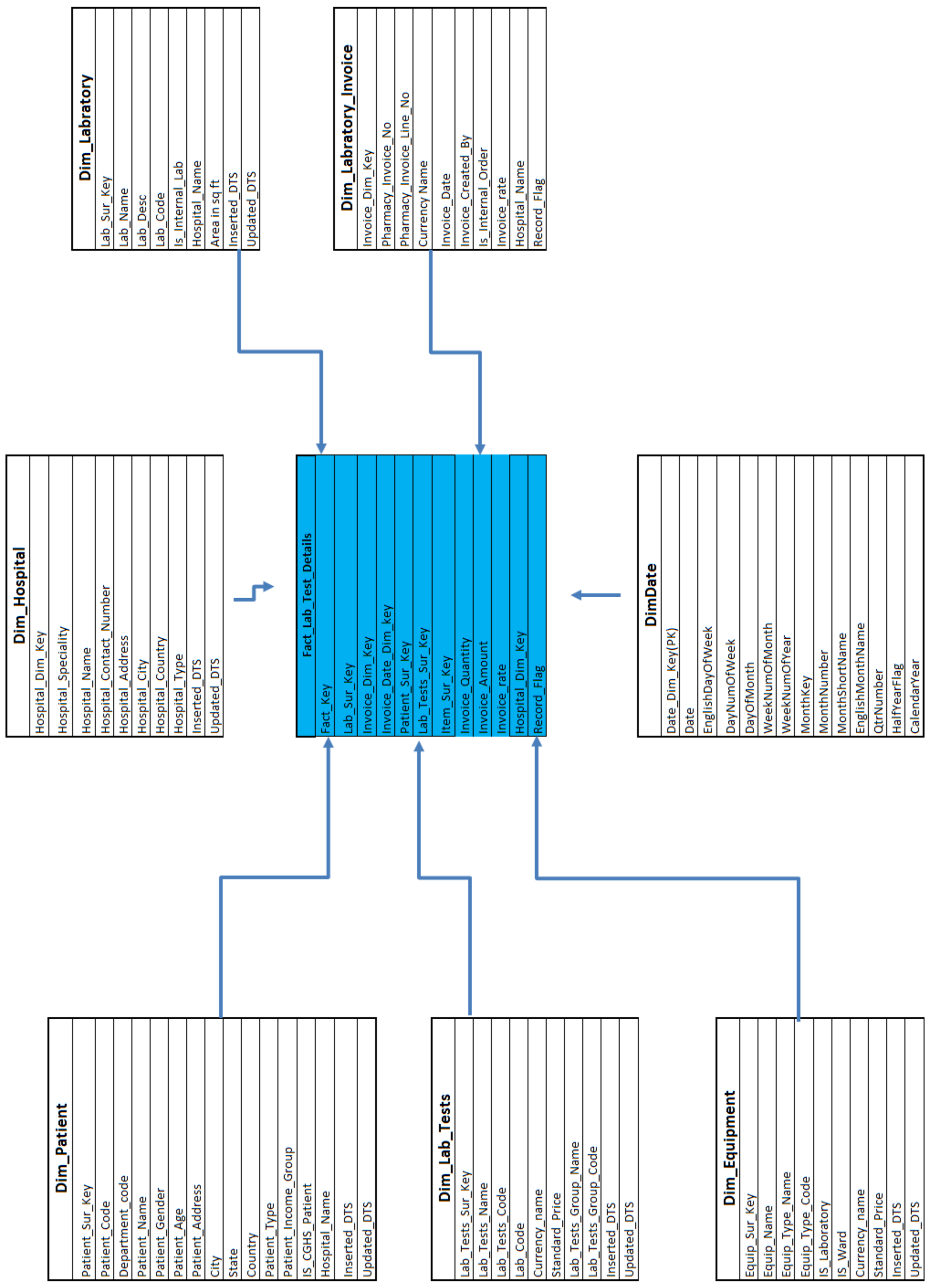

Dim_Room_Type
Room_Sur_Key
Room_Name
Room_code
Room_Type
Area of Room(sq.ft)
Dept_Name
Hospital_Name
Record_Flag
Dim_Compounder
compounder_Sur_Key
compounder_Name
compounder_Designation
compounder_Degree
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_Furniture
Furniture_Sur_Key
Furniture_Name
Furniture_Type_Name
Furniture_Type_Code
Currency_name
Standard_Price
Inserted_DTS
Updated_DTS
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Room Space Details
Room_Space_Fact_Key
Doctor_Sur_Key
Nurses_Sur_Key
Dept_Sur_Key
Staff_Sur_Key
Room_Sur_Key
Furniture_Sur_Key
compounder_Sur_Key
OPD_Sur_Key
Hospital_Dim_Key
Record_Flag
Dim_Doctor
Doctor_Sur_Key
Doctor_Name
Doctor_Designation
Doctor_specialization
IS_Salaried
Doctor_Refistration_Number
Doctor_Degree
Doctor_Type
Consultancy_fees
Gender
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_Department
Dept_Sur_Key
Dept_Name
Dept_Code
Dept_Type
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Nurse
Nurses_Sur_Key
Nurse_Name
Nurse_Designation
Nurse_Salary
Service_fees
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_Staff
Staff_Sur_Key
Staff_Type
Staff_Name
Staff_Designation
Gender
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_OPD
OPD_Sur_Key
OPD_No
Department name
Area in sq.ft
Inserted_DTS
Updated_DTS

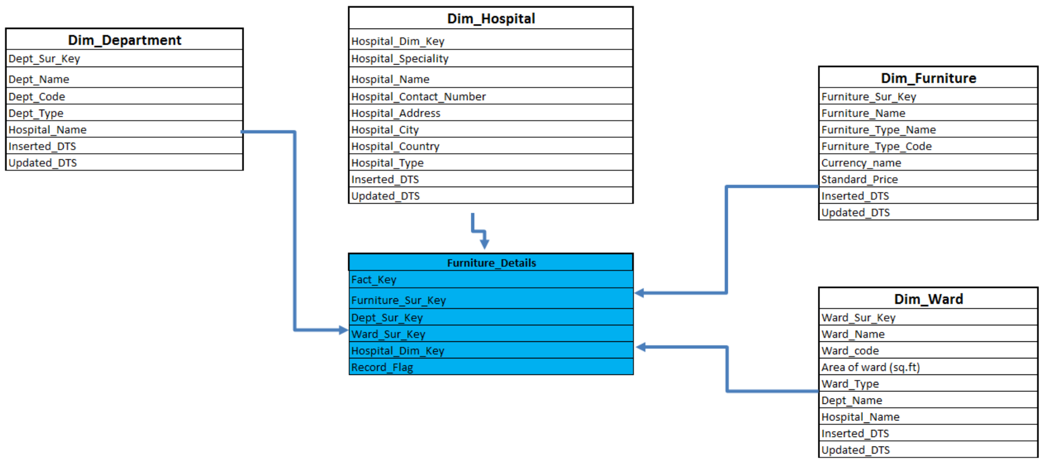

Dim_Bed
Bed_Sur_Key
Bed_Type_Name
Bed_Type_Code
Bed_Size
Currency_name
Standard_Price
Inserted_DTS
Updated_DTS
Dim_Ward
Ward_Sur_Key
Ward_Name
Ward_code
Area of ward (sq.ft)
Ward_Type
Dept_Name
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Furniture
Furniture_Sur_Key
Furniture_Name
Furniture_Type_Name
Furniture_Type_Code
Currency_name
Standard_Price
Inserted_DTS
Updated_DTS
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Bed_Details
Fact_Key
Bed_Sur_Key
Dept_Sur_Key
Ward_Sur_Key
Furniture_Sur_Key
Patient_Sur_Key
Nurses_Sur_Key
Hospital_Dim_Key
Record_Flag
DimDate
Date_Dim_Key(PK)
Date
EnglishDayOfWeek
DayNumOfWeek
DayOfMonth
WeekNumOfMonth
WeekNumOfYear
MonthKey
MonthNumber
MonthShortName
EnglishMonthName
QtrNumber
HalfYearFlag
CalendarYear
Dim_Department
Dept_Sur_Key
Dept_Name
Dept_Code
Dept_Type
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Nurse
Nurses_Sur_Key
Nurse_Name
Nurse_Designation
Nurse_Salary
Service_fees
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_Patient
Patient_Sur_Key
Patient_Code
Department_code
Patient_Name
Patient_Gender
Patient_Age
Patient_Address
City
State
Country
Patient_Type
Patient_Income_Group
IS_CGHS_Patient
Hospital_Name
Inserted_DTS
Updated_DTS


Dim_Patient
Patient_Sur_Key
Patient_Code
Department_code
Patient_Name
Patient_Gender
Patient_Age
Patient_Address
City
State
Country
Patient_Type
Patient_Income_Group
IS_CGHS_Patient
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Dim_Equipment
Equip_Sur_Key
Equip_Name
Equip_Type_Name
Equip_Type_Code
IS_Laboratory
Currency_name
Standard_Price
Unit of Measurement
Inserted_DTS
Updated_DTS
Equipment
Fact_Key
Equip_Sur_Key
Dept_Sur_Key
Ward_Sur_Key
Patient_Sur_Key
Nurses_Sur_Key
Hospital_Dim_Key
Record_Flag
Dim_Ward
Ward_Sur_Key
Ward_Name
Ward_code
Area of ward (sq.ft)
Ward_Type
Dept_Name
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Nurse
Nurses_Sur_Key
Nurse_Name
Nurse_Designation
Nurse_Salary
Service_fees
Providend_Fund
Inserted_DTS
Updated_DTS
DimDate
Date_Dim_Key(PK)
Date
EnglishDayOfWeek
DayNumOfWeek
DayOfMonth
WeekNumOfMonth
WeekNumOfYear
MonthKey
MonthNumber
MonthShortName
EnglishMonthName
QtrNumber
HalfYearFlag
CalendarYear
Dim_Department
Dept_Sur_Key
Dept_Name
Dept_Code
Dept_Type
Hospital_Name
Inserted_DTS
Updated_DTS


Dim_Doctor
Doctor_Sur_Key
Doctor_Name
Doctor_Designation
Doctor_specialization
IS_Salaried
Doctor_Refistration_Number
Doctor_Degree
Doctor_Type
Consultancy_fees
Gender
Providend_Fund
Inserted_DTS
Updated_DTS
Dim_Disease
Disease_Sur_Key
Disease_Name
Disease_Type_Name
Disease_Type_Code
Invoice_Date
Invoice_Created_By
Is_Internal_Order
Invoice_rate
Hospital_Name
Record_Flag
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Treatment
Treatment_Fact_Key
Doctor_Sur_Key
Disease_Sur_Key
Nurses_Sur_Key
Patient_Sur_Key
Hospital_Dim_Key
Record_Flag
DimDate
Date_Dim_Key(PK)
Date
EnglishDayOfWeek
DayNumOfWeek
DayOfMonth
WeekNumOfMonth
WeekNumOfYear
MonthKey
MonthNumber
MonthShortName
EnglishMonthName
QtrNumber
HalfYearFlag
CalendarYear
Dim_OPD
OPD_Sur_Key
OPD_No
Department name
Area in sq.ft
Inserted_DTS
Updated_DTS
Dim_Patient
Patient_Sur_Key
Patient_Code
Department_code
Patient_Name
Patient_Gender
Patient_Age
Patient_Address
City
State
Country
Patient_Type(In/Out)
Patient_Income_Group
IS_CGHS_Patient
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Nurse
Nurses_Sur_Key
Nurse_Name
Nurse_Designation
Nurse_Salary
Service_fees
Providend_Fund
Inserted_DTS
Updated_DTS


Dim_Pharmacy
Pharmacy_Sur_Key
Pharmacy_Name
Pharmacy_Desc
Pharmacy_Code
Pharmacy_Type
Hospital_Name
Area
Inserted_DTS
Updated_DTS
Dim_Pharmacy_Invoice
Invoice_Dim_Key
Pharmacy_Invoice_No
Pharmacy_Invoice_Line_No
Currency Name
Invoice_Date
Invoice_Created_By
Is_Internal_Order
Hospital_Name
Record_Flag
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Fact_Medicine_Dispatched_Details
PO_Invoice_Fact_Key
Pharmacy_Sur_Key
Invoice_Dim_Key
Invoice_Date_Dim_key
Patient_Sur_Key
Medicine_Sur_Key
OPD_Sur_Key
Pharmacist_Sur_Key
Hospital_Dim_Key
Invoice_Quantity
Invoice_Amount
Invoice_Rate
Record_Flag
DimDate
Date_Dim_Key
Date
EnglishDayOfWeek
DayNumOfWeek
DayOfMonth
WeekNumOfMonth
WeekNumOfYear
MonthKey
MonthNumber
MonthShortName
EnglishMonthName
QtrNumber
HalfYearFlag
CalendarYear
Dim_OPD
OPD_Sur_Key
OPD_No
Department name
Area in sq.ft
Inserted_DTS
Updated_DTS
Dim_Patient
Patient_Sur_Key
Patient_Code
Department_code
Patient_Name
Patient_Gender
Patient_Age
Patient_Address
City
State
Country
Patient_Type
Patient_Income_Group
IS_CGHS_Patient
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Medicine
Medicine_Sur_Key
Medicine_Name
Medicine_Code
Pharmacy_Code
Currency_name
Standard_Price
UOM
Medicine_Group_Name
Medicine_Group_Code
Inserted_DTS
Updated_DTS
Dim_Pharmacist
Pharmacist_Sur_Key
Pharmacist_Name
Pharmacist_Designation
Pharmacist_Salary
Pharmacist_Degree
Providend_Fund
Inserted_DTS
Updated_DTS


Dim_Labratory
Lab_Sur_Key
Lab_Name
Lab_Desc
Lab_Code
Is_Internal_Lab
Hospital_Name
Area in sq ft
Inserted_DTS
Updated_DTS
Dim_Labratory_Invoice
Invoice_Dim_Key
Pharmacy_Invoice_No
Pharmacy_Invoice_Line_No
Currency Name
Invoice_Date
Invoice_Created_By
Is_Internal_Order
Invoice_rate
Hospital_Name
Record_Flag
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Fact_Lab_Test_Details
Fact_Key
Lab_Sur_Key
Invoice_Dim_Key
Invoice_Date_Dim_key
Patient_Sur_Key
Lab_Tests_Sur_Key
Item_Sur_Key
Invoice_Quantity
Invoice_Amount
Invoice_rate
Hospital_Dim_Key
Record_Flag
DimDate
Date_Dim_Key(PK)
Date
EnglishDayOfWeek
DayNumOfWeek
DayOfMonth
WeekNumOfMonth
WeekNumOfYear
MonthKey
MonthNumber
MonthShortName
EnglishMonthName
QtrNumber
HalfYearFlag
CalendarYear
Dim_Patient
Patient_Sur_Key
Patient_Code
Department_code
Patient_Name
Patient_Gender
Patient_Age
Patient_Address
City
State
Country
Patient_Type
Patient_Income_Group
IS_CGHS_Patient
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Lab_Tests
Lab_Tests_Sur_Key
Lab_Tests_Name
Lab_Tests_Code
Lab_Code
Currency_name
Standard_Price
Lab_Tests_Group_Name
Lab_Tests_Group_Code
Inserted_DTS
Updated_DTS
Dim_Equipment
Equip_Sur_Key
Equip_Name
Equip_Type_Name
Equip_Type_Code
IS_Laboratory
IS_Ward
Currency_name
Standard_Price
Inserted_DTS
Updated_DTS


Dim_Department
Dept_Sur_Key
Dept_Name
Dept_Code
Dept_Type
Hospital_Name
Inserted_DTS
Updated_DTS
Dim_Hospital
Hospital_Dim_Key
Hospital_Speciality
Hospital_Name
Hospital_Contact_Number
Hospital_Address
Hospital_City
Hospital_Country
Hospital_Type
Inserted_DTS
Updated_DTS
Dim_Furniture
Furniture_Sur_Key
Furniture_Name
Furniture_Type_Name
Furniture_Type_Code
Currency_name
Standard_Price
Inserted_DTS
Updated_DTS
Furniture_Details
Fact_Key
Furniture_Sur_Key
Dept_Sur_Key
Ward_Sur_Key
Hospital_Dim_Key
Record_Flag
Dim_Ward
Ward_Sur_Key
Ward_Name
Ward_code
Area of ward (sq.ft)
Ward_Type
Dept_Name
Hospital_Name
Inserted_DTS
Updated_DTS