Knowledge Management for
Individuals Loan Disbursement in Bank

A Dissertation submitted to the University of Hyderabad in partial fulfillment of the degree of

MASTER OF TECHNOLOGY

in

Information Technology

by

DEBPROSAD DAS

Department of Computer and Information Sciences

School of Mathematics, Computer and Information Sciences

University of Hyderabad
(P.O.) Central University, Gachibowli
Hyderabad – 500 046
Andhra Pradesh
India

CERTIFICATE

This is to certify that the dissertation entitled "**Knowledge Management for Individuals Loan Disbursement in Bank**" submitted by **Debprosad Das** bearing Reg. No **09MCMB36** in partial fulfillment of the requirements for the award of Master of Technology in Information Technology is a bonafide work carried out by him under my supervision and guidance.

The dissertation has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. Mahil Carr
Associate Professor, IDRBT.

Head of the Department

Dean of the School

DECLARATION

I **Debprosad Das** hereby declare that this Dissertation entitled "**Knowledge Management for Individuals Loan Disbursement in Bank**", submitted by me under the guidance and supervision of **Dr. Mahil Carr, Associate Professor, IDRBT**, is a bonafide work. I also declare that it has not been submitted previously in part or in full to this University or other University or Institution for the award of any degree or diploma.

Date:

Name : **Debprosad Das**

Signature of the Student:

Regd. No. **09MCMB36**

Acknowledgments

I would like to convey my humble appreciations to my Supervisor Dr.Mahil Carr for his relentless guidance and precious suggestions. Not only just academic suggestions, but also his moral values and commitment had a profound impact on me. This one year of project happened to be a memorable voyage of ups and downs.

My honorable thank to the chairman of Institute for Development of Research and Banking Technology for providing excellent lab and library. I like to thank to the staff of IDRBT.

I also like to thank our Head of the Department Prof. C.R.Rao for providing such a nice computing facilities and labs to work round the clock.

Above all, I would like to express my gratitude to parents who made all the above come true. Last but not least; I would like to thank all my friends who made my life during project lively, colorful and entertaining.

Debprosad Das

Abstract

Preparing this project has been an enriching experience in not only improving my understanding about the Indian banking sector, the various technique used but also in understanding how important knowledge, information and data are in today's scenario. Being knowledgeable is the need of the hour not only for everyone but also for banks. Knowledge management prevents the organizations collective knowledge from perishing and it is the employee's knowledge that will help the organization achieves optimum organizational efficiency.

In this project I have tried conveying what exactly Knowledge Management is in the banks perspective and then perform collection, elicitation, analysis, modeling and validation of knowledge from different banks and finally defined a declarative rule base for banks individual loan like car loan, home loan, personal loan and educational loan to implement loan disbursement system.

Contents

LIST OF FIGURES

Chapter 1

Introduction

Knowledge is defined as the combination of information; experience and judgment, adding value to information, creating relationships and logic that allows applying it to good use, take the right decisions and make good choice. Some researchers have defined knowledge in the context of know-why, know-what, know- who and know-when, in order to relate it with managing knowledge concepts.

1.1Knowledge

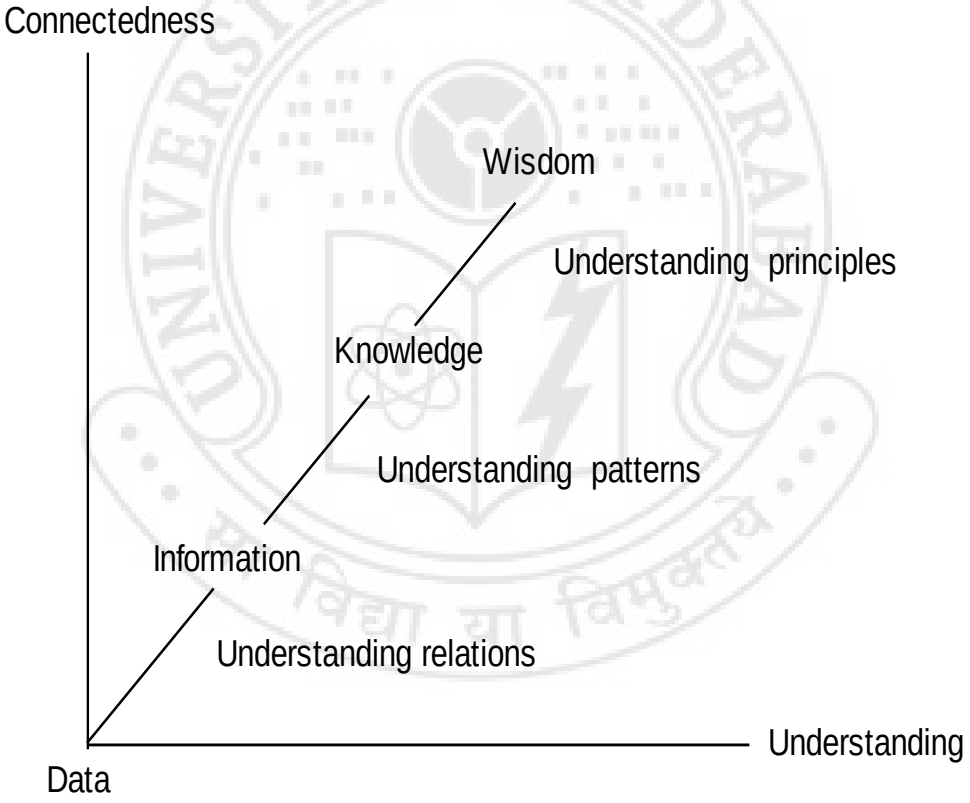
- Expertise, and skills acquired through experience or education
- The theoretical or practical understanding of a subject
- What is known in a particular field or in total
- Facts and Information
- Awareness or familiarity gained by experience of a fact or situation

Researcher Van den Bosch and Van Wijk (2001) presented a conceptual framework of managerial knowledge integration. Know-what can be defined as something people carry round in their minds and pass on to each other. In contrast, Know-how embraces the ability to put Know-what into practice. On the other hand, Japanese researchers like Nonaka, Toyomo and Konno (2002 defined Knowledge by emphasizing on the relative, dynamic and humanistic dimensions rather than the traditional western epistemology (the study of knowledge) that focused on absolute, static and non-human view of knowledge. The term ‘knowledge’ is one of the more confusing aspects of KM. The terms ‘information’ and ‘data’ are often used as synonyms for the term knowledge. In fact they are different from one another.

1.2 Data, Information and Knowledge

Data is set of discrete acts. It is unorganized and has no interpretation. It is like an event out of context, a letter out context, a word out of context. Generally, when we come across a piece of data, we try to associate it with other things to understand it. Information is the data that is organized, patterned and grouped. A collection of data for which there is no relation cannot be called as information in a true sense.

Knowledge is derived from information. When a pattern of relationship exists between the data and information, the pattern has a potential to represent knowledge. It becomes knowledge, only when a person is able to understand the patterns and its implications.



Figures 1.1: Distinction between Data, Information and Knowledge

1.3 Types of Knowledge: Explicit and Tacit

Basically, knowledge is of two types. They are explicit and tacit knowledge.

- Tacit knowledge

Tacit knowledge is personal, contextual knowledge that have people individuals in mind based on their experiences involving such intangible factors as personal belief, perspective, and values. Tacit knowledge is difficult to formulize, record, articulate and communicate to another person.

- Explicit knowledge

The term explicit knowledge can be defined as that component of knowledge which can be codified and transmitted in a methodical and prescribed language. For example, documents, databases, Webs, E-mails, charts, etc.

It is estimated that tacit knowledge constitutes about 70% of all organizations knowledge and is difficult to identify and convert into real value unless a structured approach is adopted to manage knowledge through KM techniques involving intensive dialogues, discussions and sharing in teams.

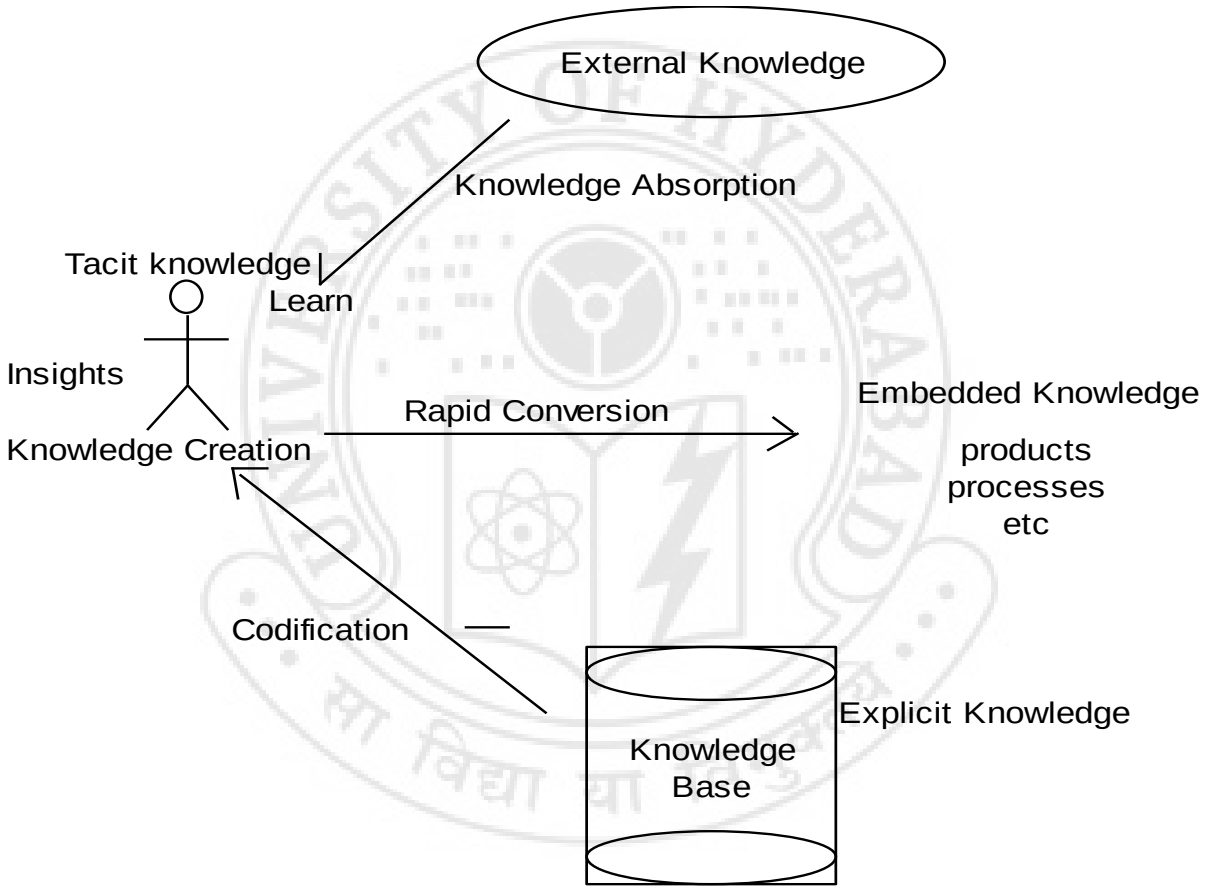
KM refers to management of knowledge. It is an important component of organizational intangible assets. Continuous changes in the market expectation and the demand for new products have replaced capital and labor intensive firms by knowledge-intensive firms. It is a concept in which enterprise gathers, organizes shares and analyzes its knowledge in terms of sources, documents and skills of the people.

It is a process of increasing organizational intelligence through involvement and participation in the organization's knowledge capital, for creating business value and generating a competitive advantage.

KM is also viewed as a conscious strategy of putting knowledge into action as a means to increase organizational efficiency.

1.4 Knowledge Flow

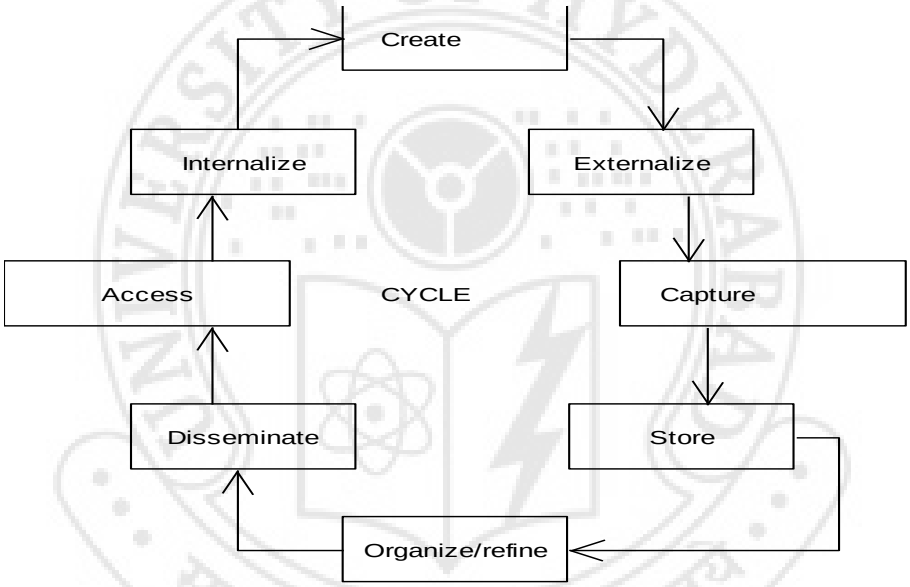
In organizations, knowledge is stored in the brains of individual employees as well as in "artifacts" such as books, manuals, files, and databases. These knowledge artifacts are often locked, are difficult to access, or are destroyed soon after they are created. The complexity of business means that knowledge often needs to flow from one person or place to another within the organization.



Figures 1.2: Organizational knowledge flow

1.5 Knowledge Sharing Cycle

- Internalizing
- Create
- Externalizing
- Capturing
- Storing
- Organize / Refine
- Disseminate
- Access



Figures 1.3: Knowledge sharing cycle

Internalize: Process of understanding information, putting it into context with existing knowledge, and therefore transforming the information into knowledge. This part of the cycle happens inside the human mind and therefore can be supported but not replaced by technology.

Create: New knowledge is created in the minds of individuals, typically by combining previous knowledge they have received from other people with their own insights or experiences. Other ways to create knowledge in a company are hiring new people, renting knowledge by hiring consultants or professionals for specific jobs, or acquisition of whole companies including their knowledge base. Again, this part of the cycle is outside the scope of technology.

Externalize: Process of explaining knowledge, thus transforming it into information. This can be as simple as speaking to somebody, writing a document, drawing a figure, preparing a presentation, or teaching. A significant problem where many KM initiatives fail is motivating

people to actually do this, and does it well. Unfortunately, there is not much that technology can do to increase motivation, as it is primarily an issue of corporate culture.

Capturing: It is the processes of transforming information into data, so that it can be stored and processed by computers (also called codifying). This process should be made as easy as possible. On one hand it should be easy to capture the information, on the other hand the information should be searchable, enriched with metadata when possible (most importantly who created it and when), and accessible by a wide range of users (e.g. over the Internet) without requiring special hardware or software.

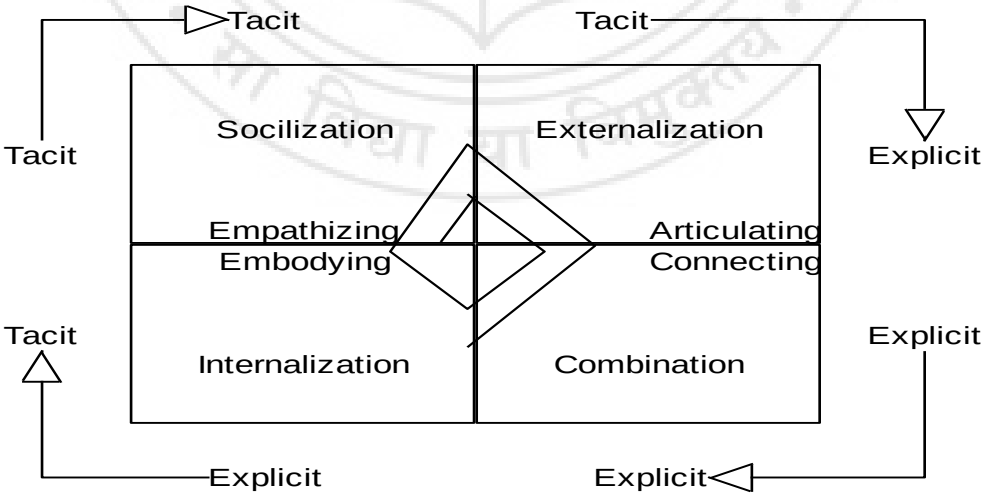
Storing: It is the processes of entering data into the corporate knowledge base. It should be as simple as possible for users to store files, data or metadata.

Organize / Refine: Once information is stored in the system, its usability and accessibility can be greatly improved by putting it into context and enriching it with additional knowledge.

Disseminate: Process that publishes access the data stored in the corporate knowledge base. Typically this uses Internet/ Intranet/ Extranet technology. Publishing and access schemes may vary, depending on the application and requirements at hand.

Access: Process where the data or knowledge artifacts are invoked by authorized users so that they can make use of it.

1.6 Knowledge Conversion: SECI Model



Figures 1.4: Knowledge Conversion: SECI Model

1.6.1 Socialization: From Tacit to Tacit

Socialization is the process where sharing tacit knowledge through face-to-face communication or shared experience. An example is an apprenticeship. In this process creates a systematic knowledge through the sharing of experiences, the development of mental models and technical skills. Language is not necessary. Socialization is connected to group processing and organization culture.

1.6.2 Externalization: From Tacit to Explicit

In this process developing concepts, which is embedded the combine tacit knowledge and which enable its communication. In the externalization creates a conceptual knowledge through knowledge articulation using language. Dialogue and collective reflection needed. For example: Mazda a company develop RX-8 concept. This concept describe as an “automatic sports car that provides an exciting and comfortable derive". The concept is deducted as car marker corporate slogan “create new values and present joyful driving pleasure.

1.6.3 Combination: From Explicit to Explicit

Combination of various elements of explicit knowledge, building a prototype is an example. In this process creates a systematic knowledge through the systemizing of ideas. May involve many media, documentation, telephone conversion, meeting, computerized communication network and can lead to new knowledge through adding, sorting, combining and categorizing. Combination is the root of information processing. For instance, business concepts or product concepts. Middle manager of an organization plays an important role.

1.6.4 Internalization: From Explicit to Tacit

Explicit knowledge becomes part of the individual’s knowledge base. In the internalization process creates an operational knowledge through learning by doing. Internalization is closely related to organization learning.

1.7 Five Phase Model of the Organization Knowledge Creation Process

- Sharing of tacit knowledge
- Creating concepts
- Justifying concepts
- Building an archetype
- Cross-leveling knowledge

1.7.1 The first Phase: Sharing Tacit Knowledge

The organization knowledge creation process starts with the sharing tacit knowledge, which corresponds roughly to socialization, since the rich and untapped knowledge that resides in individuals must first be amplified within the organization. Tacit knowledge cannot be communicative or passed to others easily, since it is acquired primarily through experience and not easily express in words. Thus sharing of tacit knowledge among multiple individuals with different background, perspectives, and motivations becomes the critical step for organization knowledge creation to take place. The individual's emotions, felling, and mental models have to be shared to build mutual trust.

1.7.2 Second Phase: Creating Concepts

Tacit knowledge shared by for example a self-organizing team is converted to explicit knowledge in the form of a new concept, process similar to externalization.

1.7.3 Third Phase: Justify Concepts

In this phase organization justify, in which the organization determines if the new concept is truly worthy of pursuit.

1.7.4 Fourth Phase: Building an Archetype

The concept of the third phase are now converted in an archetype, which can take the form of a prototype in the case of hard product development or an operating mechanism in the case of soft innovation ,such as a new corporate value, a novel managerial system, or an innovative organizational structure.

1.7.5 Five Phase: Cross Leveling Knowledge

This phase extend in to knowledge creation in, for example, a division to the other in the division, across to other divisions, or even to outside constituents in what we term cross-leveling of knowledge. These outside constituents include consumer, affiliated companies, universities, and distributors.

Chapter 2
Knowledge Management

2.1 Knowledge Management

Today, the world is on the threshold of a new revolution, i.e. knowledge revolution. Knowledge has emerged as the most essential ingredient for the success of any individual and organization. Knowledge is very important to survive and prosper in today's global market. It is an intangible asset. Organizations are required to learn to manage their intangible assets in order to compete and attain success. This practice is generally known as Knowledge Management

Knowledge management is the process to identifying and leveraging the collective knowledge in an organization to help the organization compete.

KM can also be defined as performing the activities involved in discovering, capturing, sharing, and applying knowledge so as to enhance, in a cost effective fashion, the impact of knowledge on the unit's goal achievement.

Dr. Yogesh Malhotra, one of the experts and founder contributor in the development of the concept of Km has defined the KM as under:

"Knowledge Management caters to the critical issues of organizational adaptation, survival and competence in the face of increasingly discontinuous environmental change. Essentially, it embodies organizational processes that seek synergistic combination of data and information processing capacity of information technologies, and the creative and innovative capacity of human beings."

1. Improved organization performance.

2. Increased competency, do more at less time with greater ease.

3. Providing better products and services in a shorter time with better quality, Innovation.

4. It can help decision making.

5. Avoid problems, expand our horizons, and achieve easier life.

6. Continuous improvement of an organization is possible if we should manage knowledge.

2.2 Four Pillars of Knowledge Management

1. Leadership
2. Organization
3. Technology
4. Learning

2.2.1 Leadership

The knowledge-driven organization needs leadership from the governing board, the chief executive and the management team. They will be the knowledge champions. At the same time, someone will have to assume the important new role of knowledge manager and it will be difficult to find the person who will fit the position profile. Colleges aren't turning out knowledge management professionals just yet, although a few business programs are beginning to offer the course. Meanwhile, knowledge manager will likely be found in one of those old pigeonholes we hope to abolish. One answer to this dilemma might be the formation of the first network, made up of a blend of talented people from operations, human resources, education, communication, member services, marketing and information services. No one person possesses all that knowledge, but this little network would. Then the network coordinator could act as knowledge manager. That would not only provide a solution to the problem but demonstrate the very purpose of knowledge networking.

- Leadership develops business
- Leadership develops operational strategies
- Leadership will help to survive and position for success in a dynamic Environment
- Those strategies determine vision, and must align KM with business tactics
- A successful KM requires a champion near the top of an organization who can provide the strong and dedicated leadership needed for cultural change

2.2.2 Organization

- The value of knowledge creation and collaboration should be intertwined throughout an enterprise.
- Operational processes must align with the KM framework and strategy, including all performance metrics and objectives.
- While operational needs dictate organizational alignment, a KM system must be designed to facilitate KM throughout the organization.

- Operational processes must be aligned with the new vision while redesigning the organization and identifying key levers of change, including roles and responsibilities.
- Introducing knowledge management requires organizational change and KM inevitably acts as a catalyst to transform the change, organization's culture.
- The increasing value placed on highly capable people, rising job complexity and the universal availability of information on the Internet are fundamental changes contributing to the move by organizations to leverage KM solutions.
- In order to begin changing the organization, knowledge management must be integrated into business processes.

2.2.3 Technology

- Capture and store
- Search and retrieve
- Send critical information to individuals or groups
- Structure and navigate
- Share and collaborate
- Synthesize
- Profile and personalize
- Solve or recommend
- Integrate with business applications
- Maintenance

2.2.4 Learning

- The best tools and processes alone will not achieve a KM strategy. Ultimately, people are responsible for using the tools and performing the operations
- Creating organizational behavior that supports a KM strategy will continue long after the system is established.
- Organizational learning must be addressed with approaches such as increasing internal communications, promoting cross-functional teams and creating a learning community
- Learning is an integral part of knowledge management. In this context, learning can be described as the acquisition of knowledge or a skill through study, experience or instruction
- Enterprises must recognize that people operate and communicate through learning that includes the social processes of collaborating, sharing knowledge and building on each other's ideas.
- Managers must recognize that knowledge resides in people, and knowledge

2.3 Knowledge Management Approach

Knowledge management programs typically have one or more of the following activities:

- Appointment of a knowledge leader - to promote the agenda, develop a framework
- Creation of knowledge teams - people from all disciplines to develop the methods and skills
- Development of knowledge bases - best practices, expertise directories, market intelligence etc.
- Intranet portal - a 'one-stop-shop' that gives access to explicit knowledge as well as connections to experts
- Establishing Knowledge centers as a focal points for knowledge skills and facilitating knowledge flow
- Knowledge sharing mechanisms - such as facilitated events that encourage greater sharing of knowledge than would normally take place
- Intellectual asset management - methods to identify and account for intellectual capital.

2.4 Importance of Knowledge Management

It would also be of interest to assess as to why KM has all of a sudden become so important now. It would be sacrilege to say that there is something new about knowledge or management of knowledge. The importance of acquiring knowledge or the pursuit of knowledge has always been most sought after. One of the earliest examples of knowledge management can be seen in the Stone Age days when people sat around a fire in the evening and shared their experiences of hunting and gathering food.

However there are several factors, which have given prominence to knowledge management concept and theories, like never before. High levels of uncertainty and inadequate ability to predict the future course of events characterize the new world of banking. The entire gamut of business operations is becoming highly challenging. The current scenario is characterized by an exceedingly competitive environment and a fast changing market place where changes takes place at the Speed of Thought and banking is no exception to this. It is crucial to look at strategies that would give the best chance for survival. Some of the factors are as under:

Globalization, Liberalization and deregulation have totally changed the face of financial markets. Banking today is experiencing a fact changing environment. The shelf life of products and services is getting shorter and shorter with requirements and aspirations of customers increasing by the day. Target markets are experiencing radical shifts leaving organizations with wrong products at the wrong time.

Knowledge is the only input that can help in coping the radical changes and take corrective measures before it is too late. Knowledge alone can accelerate product innovation and boost revenues. In today's information-driven economy, organizations uncover the most opportunities and ultimately derive the most value – from intellectual rather than physical assets. As such knowledge is rapidly displacing capital, monetary prowess, natural resources and labor as the quintessential economic resource.

Knowledge provides effective decision support. Effective knowledge sharing of past successes, failures, projects, initiatives enables better decisions creating more economic value for the organization.

Whenever an employee retires or leaves the organization, his experience and knowledge leaves with him too. Sometimes this knowledge goes to work for the competitor as well. It is becoming increasing critical for the organizations to somehow store this knowledge and retrieve it when required for correct decision making.

As the business environment changes, the assumptions, heuristics and processes associated in doing things also undergo a change. Organizations are often caught up in the past and continue to apply old practices, methods and processes which no longer apply. As such the need to unlearn old and non-relevant methods has become necessary more than ever before.

In order to respond to the competitive challenges, and accomplish assignments, spanning traditional boundaries and functional areas, cross functional collaboration is critical to success. Knowledge from diverse disciplines and personal skill based perspectives are necessary for creativity and innovation. Knowledge management encourages conversation and discussion and enables the process of creating, sharing and applying knowledge through varying degrees of collaboration.

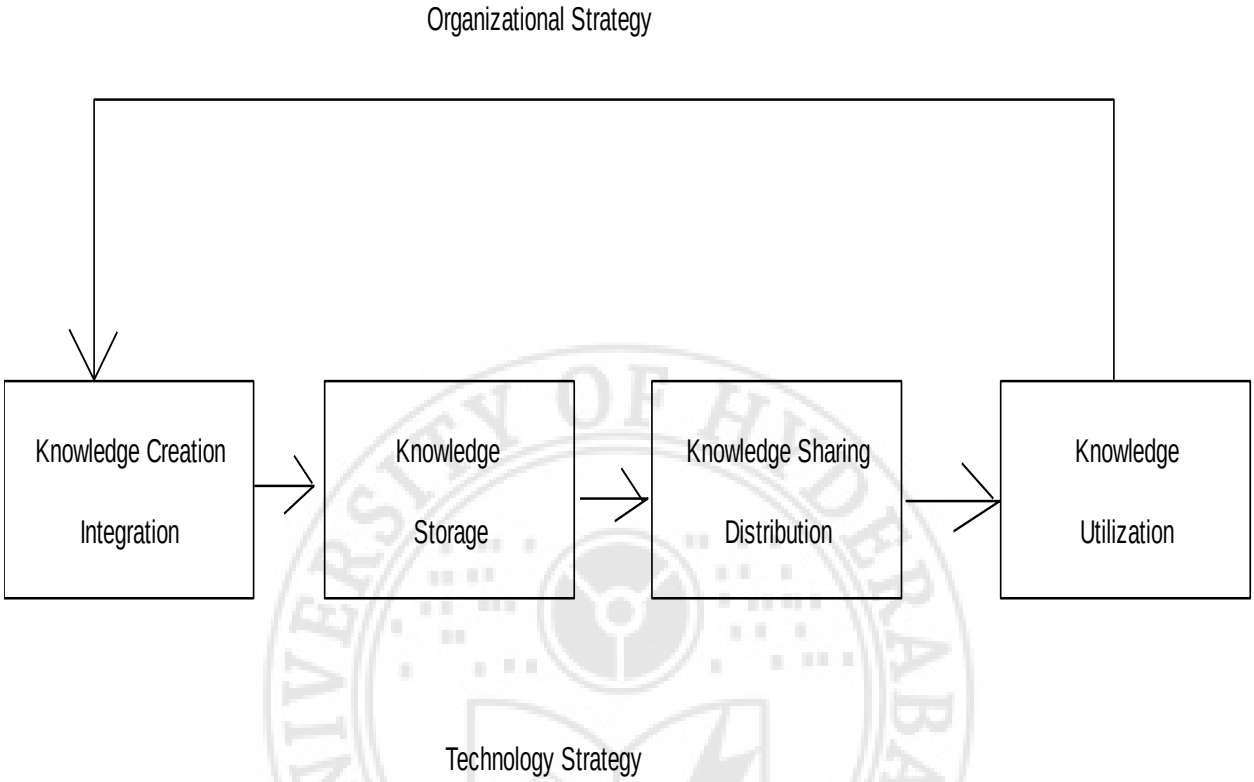
2.5 Knowledge Management Life Cycle

Knowledge Management is the methodology, tools and techniques to gather, integrate and disseminate knowledge. It involves processes involving management of knowledge creation, acquisition, storage, organization, distribution, sharing and application. These can be further classified into organization and technology components.

The organization component consists of organization-wide strategy, standard and guidelines, policies, and socio-cultural environment.

The technology component consists of tools and techniques to implement effective knowledge management practice which provides values to its business, employees, customers and partners.

The tools can furthers be classified into knowledge creation, knowledge integration, knowledge sharing and knowledge utilization.



Figures 2.1: Knowledge Management life cycle

The various steps are described here:

2.5.1 Knowledge Creation

Knowledge is created either as explicit or tacit knowledge. Explicit knowledge is put in paper or electronic format. It is recorded and made accessible to others. Tacit knowledge is created in minds of people. This knowledge resides within individuals. This knowledge needs to be transformed into explicit knowledge so that it can record and shared with others in the organization.

2.5.2 Knowledge Storage

Knowledge is stored and organized in a repository. The decision on how and where lies with the organization. But the objective of this phase to enable organization to be able to contribute, organize and share knowledge with.

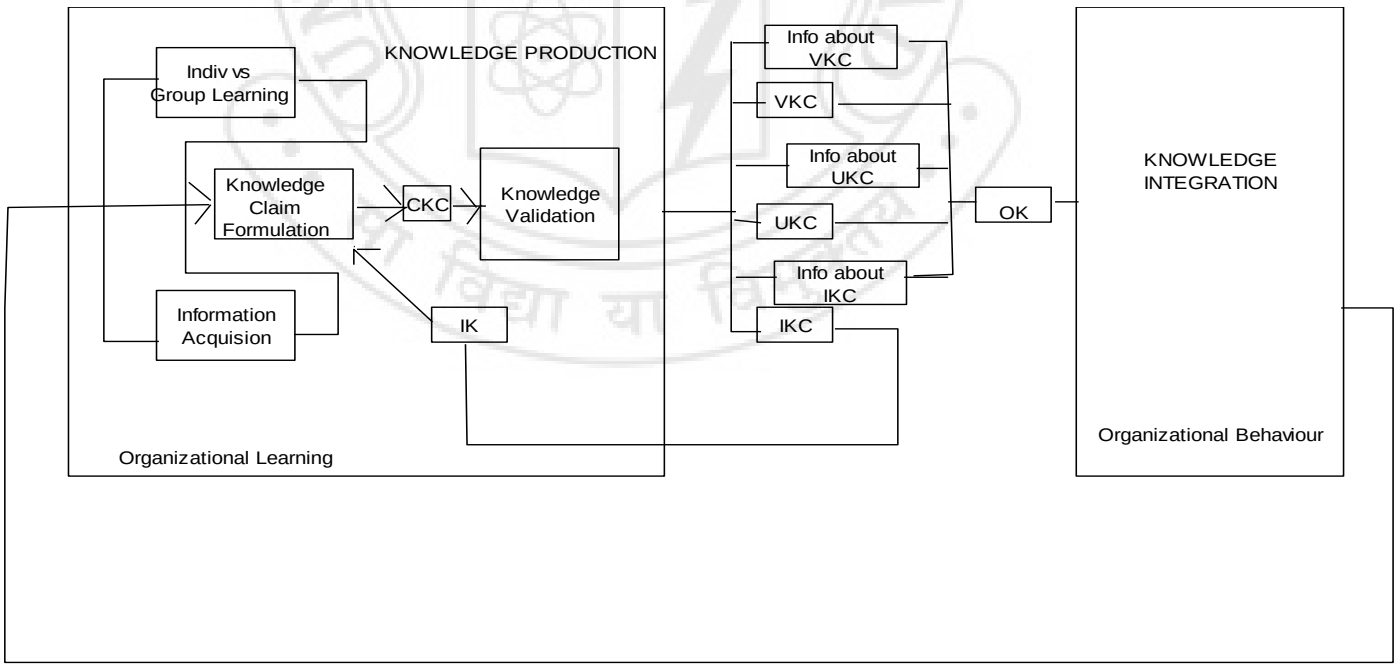
2.5.3 Knowledge Sharing

Knowledge is shared and accessed by people. They can either search or navigate to the knowledge items.

2.5.4 Knowledge Utilization

This is end goal of knowledge practice. The knowledge management does not have any value if knowledge created is not utilized to its potential. The more knowledge is created as knowledge is applied and utilized.

2.6 The Mark W. McElroy Knowledge Management Life Cycle



Figures 2.2: Mark W. McElroy Knowledge Management life cycle

2.6.1 Individual & Group Learning

Definition

• A process involving human interaction, knowledge claim formulation, and validation by which new individual and/or group knowledge is created.

• Unfolds in accordance with same life cycle process, but at an individual and group level.

• The first step in organizational learning:

"Organizations learn only through individuals who learn."

Dynamics

• Influenced by feedback from prior validation efforts.

• Produces input to Knowledge Claim Formulation process.

• Knowledge to some is only information to others until it is validated.

Potential KM Interventions

• Innovation initiatives at the individual and group level.

• Formalized feedback mechanisms between knowledge integration phase and individual and group learning activities.

• Formalized feed-*forward* mechanisms to knowledge claim formulation.

• User-driven desktop environments (Demand-side KM technologies).

2.6.2 Knowledge Claim Formulation

Definition

• A process involving human interaction by which new organizational knowledge claims are formulated.

• The codification of knowledge claims at an organizational level.

Dynamics

• Consolidates knowledge claims derived from individual and group learning, as well as information acquired from outside sources.

• Results in production of codified knowledge claims for submission to organizational validation process.

Potential KM Interventions

• Create formalized procedure for receipt and codification of individual and group innovations at an enterprise level.

• Formalize concept of "knowledge structures" and associated expressions of knowledge claims (e.g., business process vs. strategy innovations, etc.).

• Formalize with designated

2.6.3 Information Acquisition

Definition

• A process by which an organization either deliberately or serendipitously acquires knowledge claims or information produced by others external to the organization.

Dynamics

• Plays a fundamental role in the formulation of new knowledge claims at the organizational level in conjunction with knowledge claims pit forth by individuals and groups.
• Also influenced by feedback received from knowledge integration activities.

Potential KM Interventions

• Significant involvement in outside initiatives, consortia, thinks tanks, etc.
• Research initiatives.
• Industry conferences and outside training programs.
• Targeted subscription services (Gartner, etc.).
• In-house IA functions (library services, etc.).
• Highly customized desktop systems that enable self-styled research on an Individual basis.

2.6.4 Knowledge Validation

Definition

• A process by which knowledge claims are subjected to organizational criteria to determine their value and veracity.

Dynamics

• A managed function which formally evaluates competing knowledge claims when put forth by the Knowledge Claim Formulation process.
• A process which precedes the transformation of organizationally held knowledge in cases where new knowledge claims are deemed to be of a higher value to the firm than "currently held rules."

Potential KM Interventions

• Institutionalize validation process as a formalized way of processing new knowledge claims.
• KM practitioners must establish validation criteria (i.e., the rules by which new rules will be evaluated: "meta-rules").
• Requires the maintenance of codified organizational knowledge for use as a backdrop in considering newly formulated knowledge claims ("knowledge structures").

2.6.5 Knowledge Integration

Definition

• The process by which an organization introduces new knowledge claims to its operating environment and retires old ones.

• Knowledge Integration includes all knowledge transmission, teaching, knowledge sharing, and other social activity that communicates either an understanding of previously produced organizational knowledge to knowledge workers, or the knowledge that certain sets of knowledge claims have been tested and that they and information about their validity strength is available in the organizational knowledge base, or some degree of understanding between these alternatives.

Dynamics

• Knowledge integration processes may also include the transmission and integration of information.

• Driven by outcome of knowledge validation process with emphasis on implementation of new knowledge sets and associated business behaviors.

• Produces feedback for reference in future knowledge production activities.

Potential KM Interventions

• A complex array of business model redesigns activities focusing on knowledge structure transformation (business processes, strategies, etc.).

Chapter 3

Methodologies

3 Knowledge Acquisition Techniques

Knowledge acquisition includes the elicitation, collection, analysis, modeling and validation of knowledge-From different banks.

The important issues in knowledge acquisitions are:

- Knowledge is in the head of experts
- Expert have vast amounts of tacit, explicit knowledge
    - They do not know all that they know and use.
    - Tacit knowledge is hard to describe
- Experts are very busy and valuable person.
- One expert does not know everything.
- Knowledge has a “shelf life”.

The techniques for acquiring, analyzing and modeling knowledge from individuals and groups are

1. Interviewing experts
2. Learning by being told
3. Learning by observation
4. Protocol-generation techniques
5. Protocol analysis techniques
6. Hierarchy-generation techniques
7. Matrix-based techniques
8. Sorting techniques
9. Limited-information and constrained-processing tasks
10. Diagram-based techniques
11. Storytelling
12. Questionnaires or Surveys
13. Brainstorming or Ad-hoc Sessions

14. Focus Groups
15. Learning Histories
16. Documentation
17. Participation
18. Learning from others

These are applicable to the capture of tacit knowledge. In many cases, the approaches can be combined.

3.1 Interviewing experts

Structured interviews of subject matter experts is the most often used technique to render key tacit knowledge of an individual into more explicit forms. In many organizations, structured interviewing is performed through exit interviews that are held when knowledgeable staffs are near retirement age.

3.2 Learning by being told

The interviewee expresses and refines his or her knowledge and at the same time, the interviewer or knowledge engineer clarifies and validates the knowledge thus rendering the knowledge in an explicit form. This form of knowledge acquisition typically involves domain and task analysis.

3.3 Learning by observation

Observation is an important tool that can provide a wealth of information. Silent observation is best used to capture the spontaneous nature of a particular process or procedure.

3.4 Protocol-generation techniques

Includes different types of interviews such as unstructured, semi structured, structured, reporting and observational techniques.

3.5 Protocol analysis techniques

Those are used with transcripts of interviews or other text-based information to identify basic knowledge objects within a protocol. Such as goal, decisions, relationships and attributes.

3.6 Hierarchy-generation techniques

Involve creation, reviewing and modification of hierarchical knowledge that are used to build taxonomies or other hierarchical structures such as goal trees and decision network.

3.7 Matrix-based techniques

Involve the construction and filling –in a 2-D matrix (table) indicating problem and solutions between tasks and resources, etc.The elements within the matrix can contain symbols such as crosses, questions marks, color, number, text.

3.8 Sorting techniques

Used for capturing the way people compare and order concepts; it may reveal knowledge about class, properties and priorities.

3.9 Limited-information and constrained-processing tasks

Are techniques that either limits the time and/or information available to the expert when performing tasks? For instance, the twenty-question technique provides an efficient way of accessing the key information in a domain in a prioritized order.

3.9 Diagram-based techniques

Include generation and use of concept maps, state transition networks, event diagrams and process maps. These are particularly important in capturing the" what, how, when, who and why" of tasks and events.

3.10 Storytelling

Stories are another excellent vehicle for both capturing and coding tacit knowledge. An organizational story is a detailed narrative of management actions, employee interactions, and other intra organizational events that are communicated informally within the organization. Conveying information in a story provides a rich context, causing the story to remain in the conscious memory longer and creating more memory traces than is possible with information not in context. Stories can greatly increase organizational learning, communicate common values and rule sets, and serve as an excellent vehicle for capturing, coding, and transmitting variable tacit knowledge

3.11 Questionnaires or Surveys

When a large group of people should be interviewed, a questionnaire could be a first step, followed by individual interviews. The questionnaire could include close-ended and/or open-ended questions. The latter are best for gaining more information as they do not limit the respondent to a set of predefined answers.

3.12 Brainstorming or Ad-hoc Sessions

In this method took of no more than 30 minutes for sharing ideas in a stimulating and focused atmosphere. They can take place as face-to-face meetings or make use of technologies such as instant messaging, e-mail, teleconferencing, and chat rooms.

3.13 Focus Groups

Include structured sessions in which a group of bank manager is asked to share their views about a previously presented solution.

3.14 Learning Histories

Represent a retrospective history of significant events that occurred in the organization's recent past, as described in the voice of the people who took part in them. The learning history process starts with planning which establishes the scope of the learning history to be captured. After that participants are asked to share their analysis, evaluation, and the judgment they used. Other insights emerge and the capture and codification of these insights helps increase the organization's reflective capacity. Next, the information that was gathered from the interviews is synthesized into a summary format that will make it very easy for others to access, read, and understand. A learning history is thus a systematic review of successes and failures in order to capture best practices and lessons learned.

3.15 Documentation

It could include documentation from existing systems, archival information, policies and procedural manuals, reports, memos, meeting notes, standards, e-mails, public regulations and guides etc.

3.16 Participation

Learning-by-doing or on-the-job-training is invaluable both for experience and for obtaining knowledge. It is experimental, deductive learning that seeks to make sense of occurrences and to establish causal links between actions and outcomes. Apprenticeships, internships or traineeships and mentoring are forms of experienced skilled persons passing knowledge to a novice.

3.17 Learning from others

Can involve activities such as external benchmarking, which involves learning about what the leaders are doing in terms of their best practices, either through publications or site visits, and then adapting and adopting their best practices. Other learning sources include company acquisitions or mergers, attending conferences and expositions and commissioning specific studies. Inviting guest speakers to an organization presents yet another opportunity to bring a fresh perspective or point of view.

3.2 Knowledge Codification Techniques

For Bank's individual loan serves the pivotal role of allowing what is known in the Bank Individuals loan to be used collectively. By converting knowledge into a tangible, explicit form knowledge must be codified in order to be understood. However, it is difficult to codify in a document knowledge, skills, expertise, understanding and passion of a bank manager or assistance bank manager. The codification of explicit knowledge can be achieved through a variety of techniques such as

1. Cognitive mapping
2. Knowledge taxonomies
3. Task analysis
4. Decision trees

3.2.1 Cognitive Maps

A cognitive map is a representation of the "mental model" of a person's knowledge and provides a good form of codified knowledge. Thus, cognitive mapping is based on concept mapping, and allows experts to construct knowledge models. Once expertise, experience, and know-how have been rendered explicit, the resulting content can be represented as a cognitive map.

3.2.2 Knowledge Taxonomies

Concepts can be viewed as the building blocks of knowledge and expertise. Taxonomies are basic classification systems that enable us to describe concepts and their dependencies – typically in a hierarchical fashion. The higher up the concept is placed, the more general or generic the concept is. The lower the concept is placed, the more specific an instance it is of the higher-level categories. This approach allows lower or more specific concepts in the taxonomy to directly incorporate the attributes of the higher level or the parent concepts.

3.2.3 Task Analysis

An approach that looks at each key task an expert performs and characterizes the tasks in terms of prerequisite knowledge/skills required, consequences of error, frequency, difficulty, and interrelationships with other tasks and individuals, as well as how the task is perceived by the person (routine, dreaded, or eagerly anticipated). It could be done by observation (silently) or as an interview by the knowledge engineer.

3.2.4 Decision Trees

Decision tree is typically in the form of a flowchart, with alternate paths indicating the impact of different decisions being made at that juncture point. A decision tree can represent many "rules," and when execute the logic by following a certain path.

Chapter 4

Knowledge Acquisition

To collect data from different banks, more than ten banks was visited. These banks are State Bank of India (SBI), ICICI Bank, Axis Bank, HDFC Bank, Canara Bank, CITI Bank, Bank Of India, Federal Bank, State Bank of Hyderabad and Andhra Bank. Bank loan managers give documents on how the process works. Interviews are taken on loan, what are the criteria they should consider to give loan. The information what I found are given bellows.

Bank's individual loan is four types

1. Car Loan

2. Personal Loan

3. Home Loan

4. Educational Loan

4.1 Car Loan

To banks, credit profile is the most important factor that will consider for giving a car loan before funding borrower. Credit profile tells banks if one is able to and intends to pay back the loan. The eligibility components are

1. Nationality
2. Value of car
3. Annual income
4. Age
5. Amount of loan
6. Employment type[Salaried/Self-employed, Residential, Identity Proof]

4.2 Personal Loan

An unsecured loan is called a Personal loan. This implies that does not need to give any security, Personal loan can be used for anything and everything. Loan eligibility depends upon various factors which differ from banks to banks. The main factor of course, is to ability to repay the loan.

1. Nationality
2. Loan amount
3. Income per month
4. Age
5. Living area[Rural, Urban]

4.3 Home Loan

The Home Loan is a loan taken by a borrower from the bank issued against the property. f the borrower is failed to pay back the loan, the banker can retrieve the lent money by selling the property. Your Home Loan eligibility is determined by

1. Nationality
2. Age
3. Employment type[Salaried/Self-employed, Residential, Identity Proof]
4. Amount of loan
5. Annual income
6. Current experiences[Salaried/Self-employed]

4.4 Educational Loan

Different banks have different opinion for educational loan approval. Most of the bank said, "Citizen Ship should be India". Some bank said caste SC/ST are needed 50% marks and OBC/OC are need 60% marks. Age must be between 16 to 40. Study at India, most of the banks considers Management, engineering and Medical course but study at overseas bank also considers Bachelor or Master Course.

1. Nationality
2. Good Academic Result(Marks 10^{th} +12^{th})
3. Caste
4. Age
5. Study At India or Foreign country
6. Course

Chapter 5
Knowledge Codification

The knowledge gathered through interviews are codified using Decision Tree. Bank's individual loan is four types

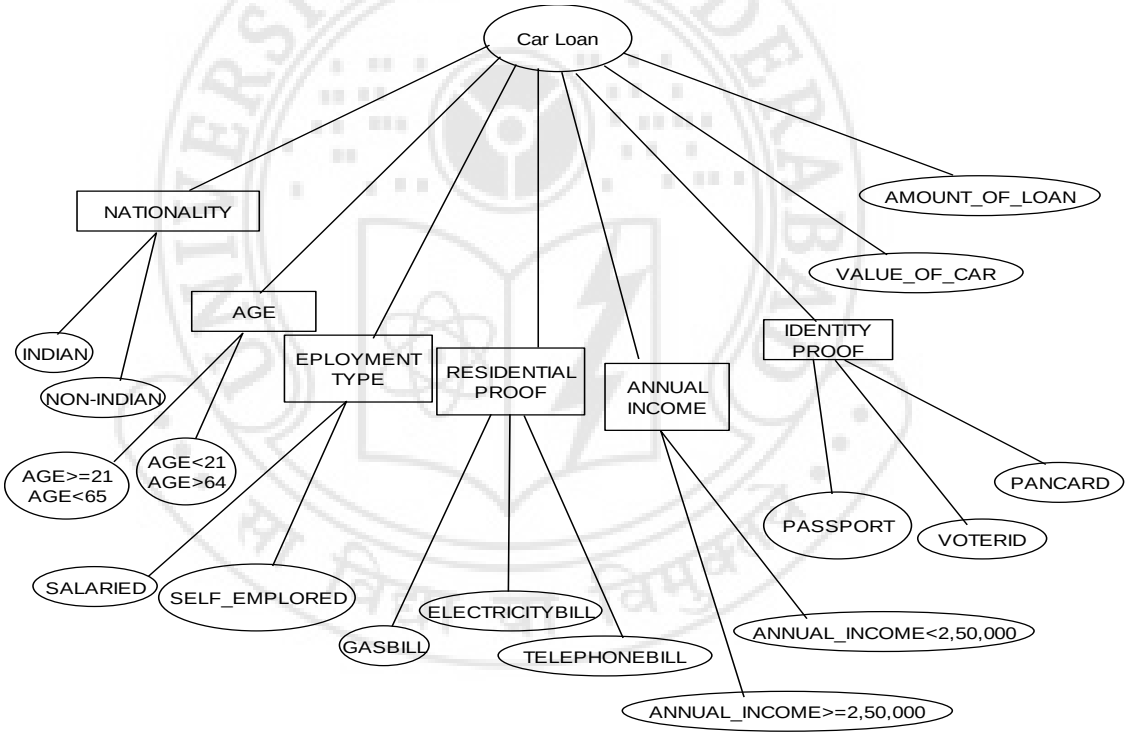
1. Car Loan

2. Personal Loan

3. Home Loan

4. Educational Loan

5.1 Car Loan Decision Tree



Figures 5.1: Car Loan Decision Tree

5.1.1 Rule Declaration for Car Loan

if nationality is indian then return true

else if

nationality is non-indian then return false

if age >=21 and age<65 then return true

else if

age <21 and age>=65 then return false

if amount_of_loan<value_of_car then return true

else if

amount_of_loan>=value_of_car then return false

if employment type is salaried and identity proof is (passport or voterid or pancard) then return true

else if employment type is selfemployed and residential proof is (gasbill or electricitybill or telephonebill) then return true

else return false

if person annual income >=2, 50,000 then return true

else

if person annual income <2, 50,000 then return false

if

nationality is true,

employment type and (identity prof or residential prof) is true,

age is true,

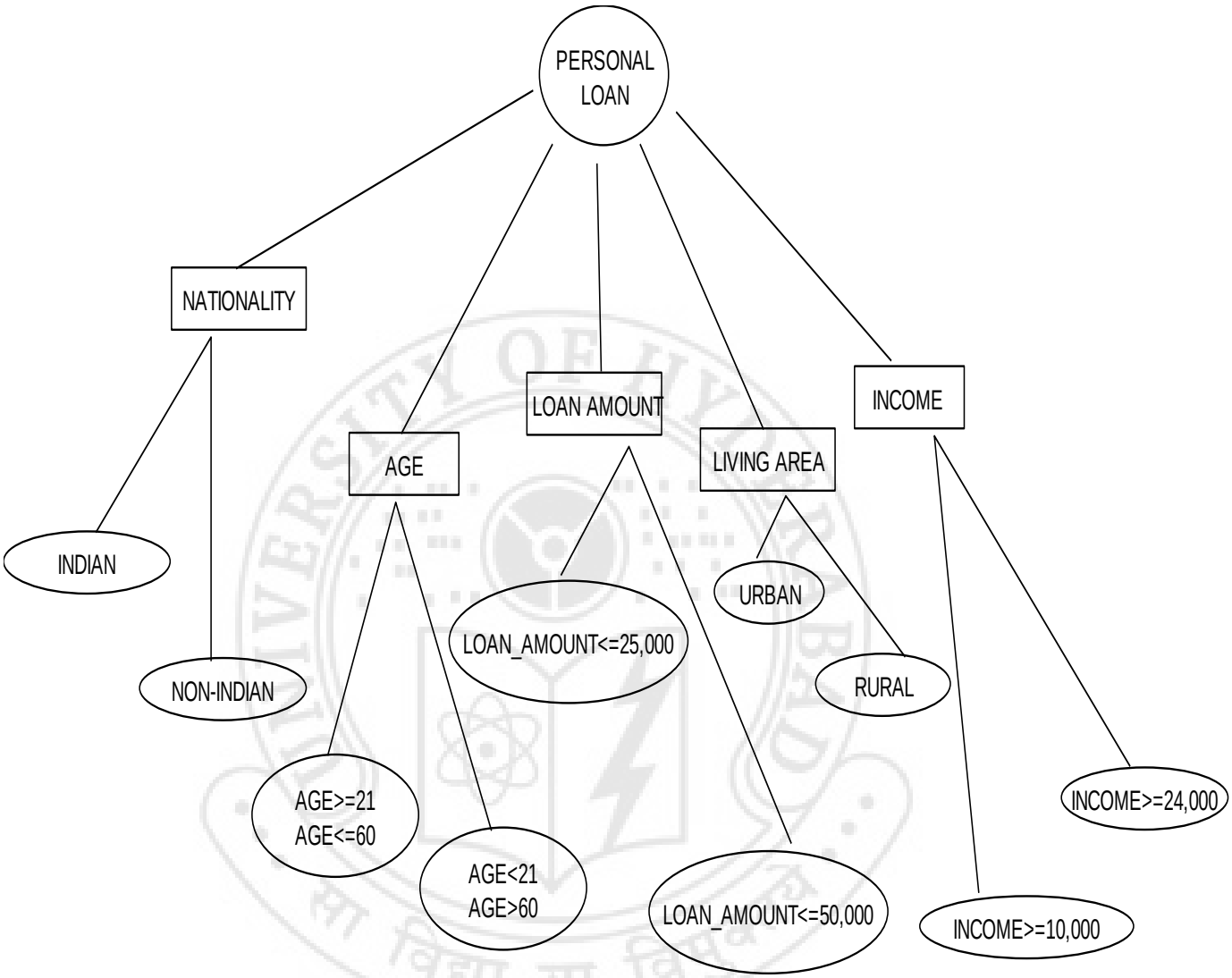
amount_of_loan is true,

personal annual is true,

then return true

else return false

5.2 Personal Loan Decision Tree



Figures 5.2: Personal Loan Decision Tree

5.2.1 Rule Declaration for Personal Loan

if nationality is indian then return true

else

if nationality is non-indian then return false

if age >=21 and age<61 then return true

else

if age <21 and age>=60 then return false

if loan_amount<=25,000 and living_area is rural then return true

else

if loan_amount<=50,000 and living_area is urban then return true

else return false

if living_area is urban and income >=24,000 then return true

else

if living_area is rural and income >=10,000 then return true

else return false

if

nationality is true,

age is true,

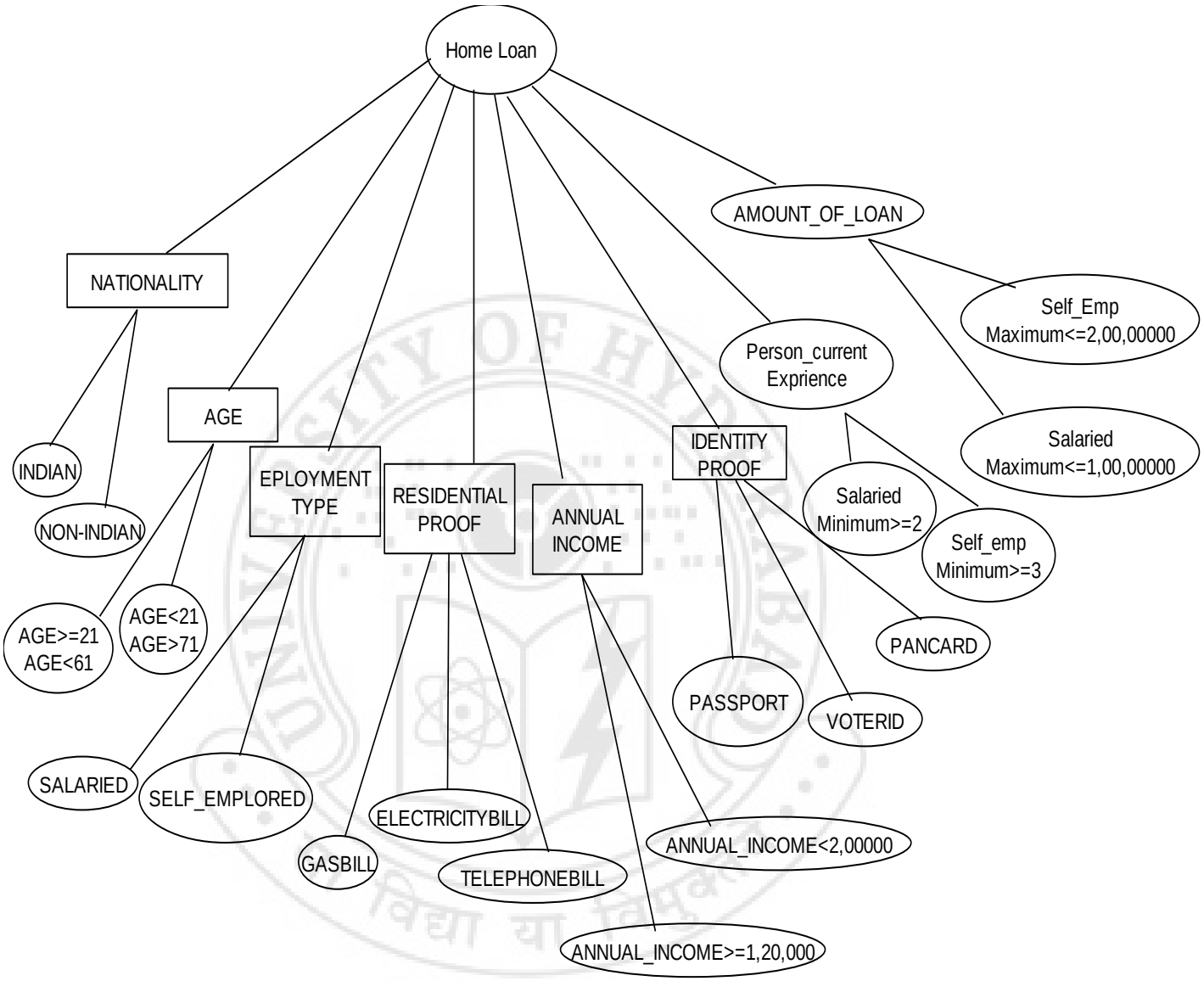
loan_amount_living area is true,

income is true,

then return true

else return false

5.3 Home Loan Decision Tree



Figures 5.3: Home Loan Decision Tree

5.3.1 Rule Declaration for Home Loan

if

nationality is indian,

employment type is salaried,

identity proof is (passport or voterid or pancard) ,

age >=21 and age<= 60,

income>=1, 20000,

job experience minimum>=2,

amount_loan <=1, 00, 00000 then return true

else if

nationality is indian,

employment type is selfemployed,

residential proof is (gasbill or electricitybill or telephonebill) ,

age >=21 and age<= 70,

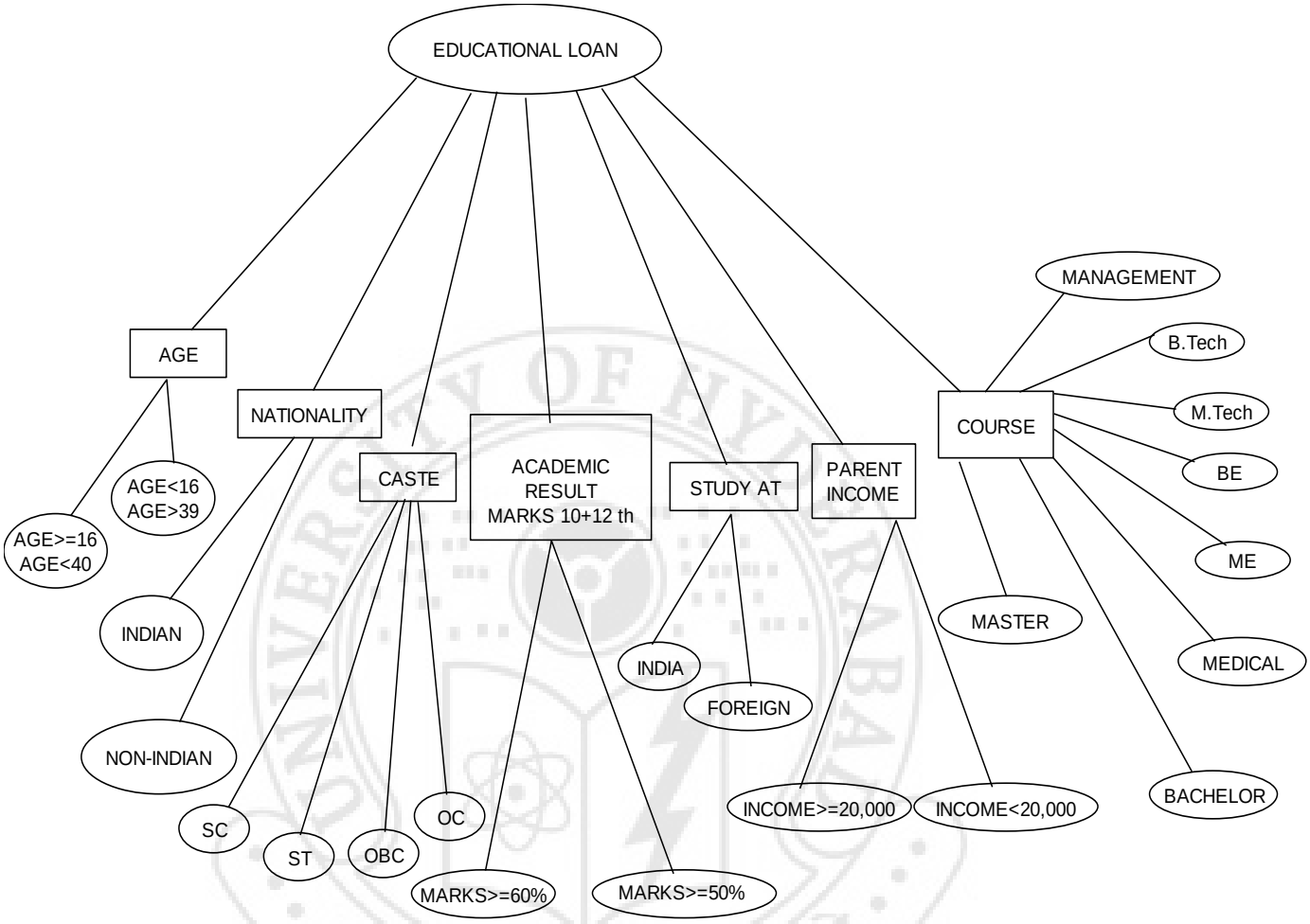
income>=2, 00000,

job experience minimum>=3,

amount_loan <=2, 00, 00000 then return true

else return false

5.4 Educational Loan Decision Tree



Figures 5.4: Educational Loan Decision Tree

5.4.1 Rule Declaration for Educational Loan

if age>=16 and age <40 then return true

else if age<16 and age >39 then return false

if nationality is indian then return true

else if nationality is non-indian then return false

if study is (sc or st) and academic result marks>=50% then return true

else

if cast is (obc or oc) and academic result marks>=60% then return false

if study at india and course is (management or b.tech or m.tech or be or me or medical) then return true

else

if study at foreign and course is (management or b.tech or m.tech or be or me or medical or bachelor or master) then return true

else

if study at india and course is (bachelor or master) then return false

if parent_income>=20,000 then return true

else

if parent_income<20,000 then return false

if

age is true,

nationality is true,

cast_academic result marks is true,

study at_course is true,

parent_income is true,

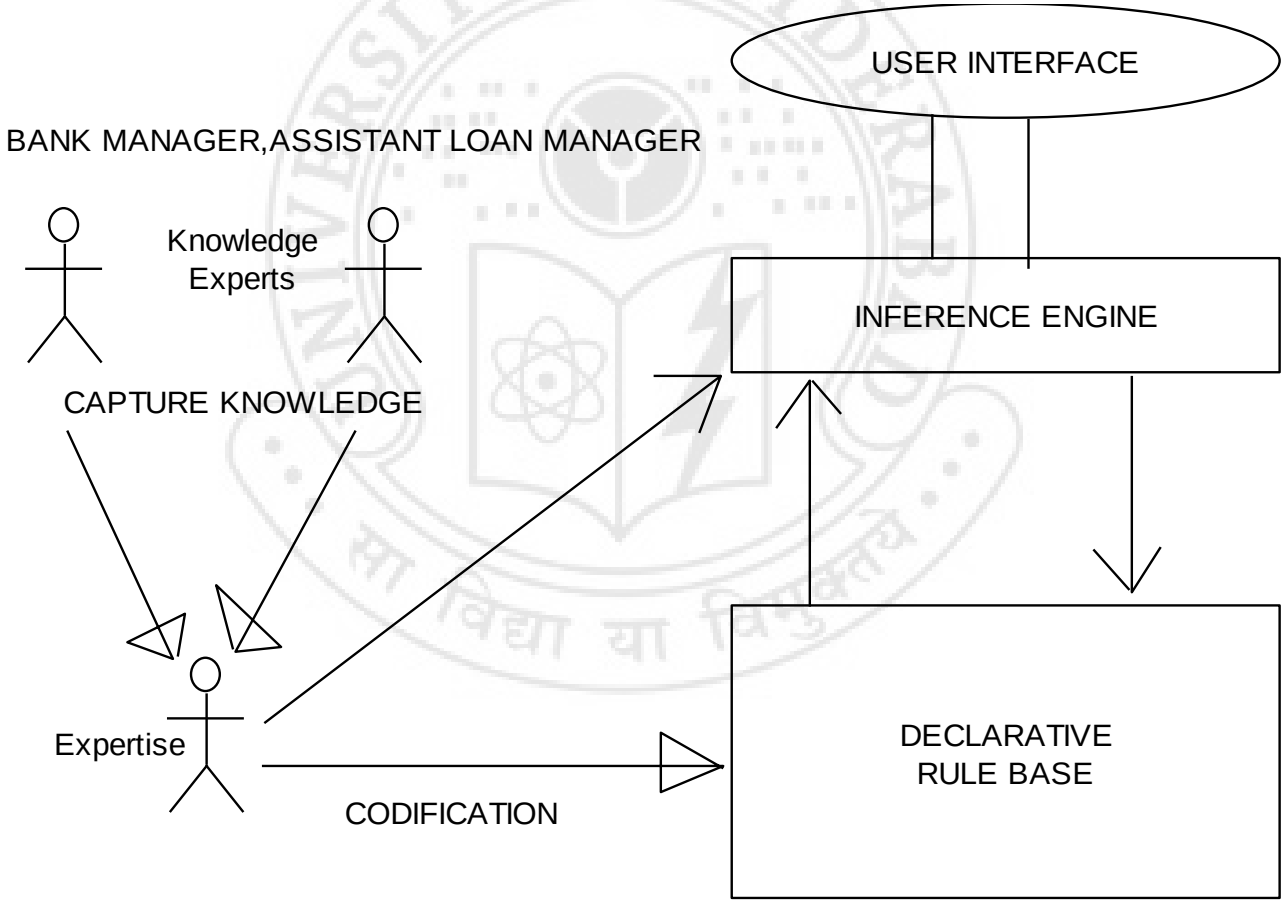
then return true

else return false

Chapter 6

Knowledge Inference Engine and Results

Define declarative rule base and inference engine I was used Visual Prolog (VIP).It is an Object-Oriented Programming Language that can be used to create programs that run under MsWindows.The programs should support a GUI, a Graphical User Interface. Basically visual prolog based on Horn clause logic and concept using backward chaining algorithm.



Figures 6.1: Project Infrastructure

6.1 Components and interfaces

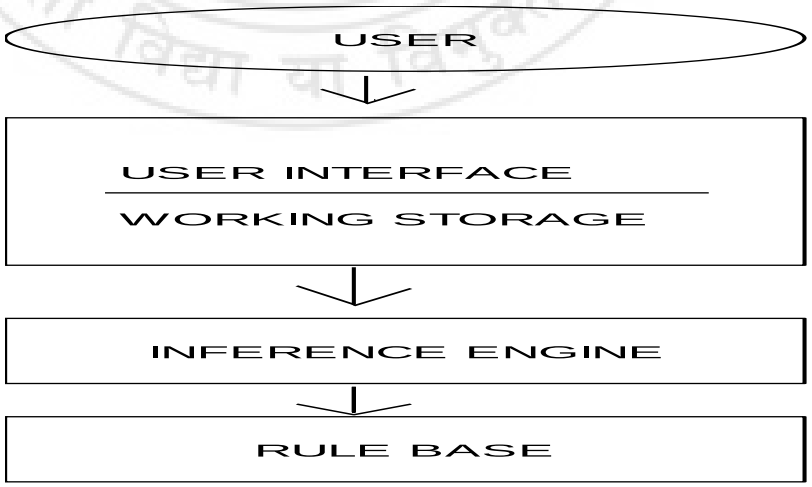
- **Knowledge base**

A declarative rule base for loan disbursement like if then rules. A system contains formal representation of the information provide by the domain expert. This information may be in the form of problem solving rules, procedures, or data intrinsic to the domain. To incorporate these information into the system, it is necessary to make use of one or more knowledge representation methods. Transferring knowledge from the human expert to a computer is often the most difficult part of building a system. The knowledge acquired from the human expert must be encoded in such a way that it remains a faithful representation of what expert knows, and it can be manipulated by a computer.

- **Working storage**

The data which is specific to a problem being solved. Working memory refers to task-specific data for a problem. The contents of the working memory, changes each problem situation.Consequently,it is the most dynamic component of a system, assuming that is kept current.

- Every problem in a domain has some unique data associated with it
- Data may consist of the set of conditions leading to the problem, its parameters and so on.
- Data specific to the problem needs to be input by the user at the time of using, means consulting the system. The Working storage is related to the user.
- Fig below shows how working memory is closely related to the user interface of the system.



Figures 6.2: How working memory is closely related to the user interface of the system

- **Inference Engine**

The code of the system which derives recommendations from the knowledge base and problem specific data in working storage. The inference engine is a generic control mechanism for navigating through and manipulating and deduces results in an organized manner. The inference engine's generic control mechanism applies the axiomatic knowledge present in the knowledge base to the task-specific data to arrive at some conclusion.

- **User interface**

The code that controls the message between the user and the system, such as pop-up Manu, windows, and mice requires Graphical interface. The acceptability of a system depends to a great extent on the quality of the user interface. The easiest to implement interfaces communicate with the user through a scrolling dialog. The user can enter commands, and respond to questions. The system responds to commands, and asks questions during the inference process.

6.2 Roles of individuals who interact with the system

- Domain Expert: The individuals who are expert in solving loan disbursement such as Bank manager, Loan manager, Assistant loan manager.
- System Engineer: Builds the user interface, declarative knowledge base, implements the inference engine.
- User: The individual who will be consulting with the system.

Human experts usually tend to think along

Condition ⟹ action or Situation ⟹ conclusion

Rules "IF-THEN" are predominant from of encoding knowledge in system. These are of the form:

IF a (1), a (2),……., a (n)

THEN b (1), b (2),……..., b (n) where

Each a (i) is a condition or situation, and

Each b (i) is an action or a conclusion.

6.3 Horn Clause Logic

- Visual Prolog is based on Horn Clause logic. Horn Clause logic is a formal system for reasoning about things and the way these things are related to each other. An example: in natural language I can express a statement like Applicant Information**:** Age is 23, Nationality is INDIAN, Caste is SC, Country studied INDIA, Marks have 60%, and Parent income was 25000.

- Here I have "things": 23, INDIAN, SC, INDIA, 60, 25000 and "relation" between them, namely that Age, Nationality, Caste, Study_at, Marks and Parent_income. In Horn Clause Logic I can formalize these statement in the following way:

Natural Language	Horn Cluses Logic
Age is 23	Age(23)
Nationality is INDIAN	Nationality("INDIAN")
Caste is SC	Caste("SC")
Country studied INDIA	Study_at("INDIA")
Marks have 60%	Marks(60)
Parent income was 25000	Parent_income(25000)

Now for another general example, John is the father of Bill.
Here I have two "things": John and Bill, and a "relation" between these, namely that one is the father of the other. In Horn Clause Logic this statement can represent as like as
father ("Bill", "John"). In the formalization I put the name of the relation in front and put the two "things" between brackets, separated by a comma. The names of the two things are in computer slang known as string literals or strings. The apostrophes are used to delimit the strings. To add some more concepts: we say that "father" is a *predicate* (or a relation) taking two arguments, "Bill" and "John". The predicate describes the relation between two *objects*, in this case Bill and John. In the interpretation that I give here the second argument "John" is the father of the first argument "Bill". Notice that *I* have chosen that the second object should be the father of the first. I might as well have chosen it the other way around: The order of the arguments is the choice of the designer of the formalization. However, once you have chosen, you must be consistent. So in my formalization the father must always be the second person. You can say that in a computer program you create a world and that in this world the second argument of the predicate "father" represents the father in the relation.

I have chosen to represent the persons by their first names and to represent the names by strings. In a more complex world this would not be sufficient because many people have the same first name. But for now we will be content with these simple formalization.With formalizations like the one above I can state any kind of family relation between any persons. Or more general, with

a predicate and arguments I can specify any relation between objects. Take a look at some examples.
lives_in(“Thomas”, “Groningen”).
is_married_to(“John”,”Claire”). Are examples of relations between two objects? Any number of arguments is possible. E.g. one argument:
is_dumb(“John”).
But also three objects are possible:
has_degree(“Jeannet”, “shorthand”, “Spanish”).
The number of arguments is only limited by your imagination.
Specifications like these are called facts. A fact consists of a predicate, a number of arguments between brackets and separated by comma’s and it is ended by a full stop, a period. A fact describes a static relation between objects.Facts are useful to know, but they get boring as you find out when you watch quizzes on TV. To make facts more interesting, you need more general relations between objects or groups of objects. General relations between (groups of) objects are called *rules*.

As an example we can extend the father example with a rule about grandfathers.
Person3 is the grandfather of Person1, if Person2 is the father of Person1 and Person 3 is the father of Person 2 where Person1, Person2 and Person3 indicate individual persons (objects). In Horn Clause Logic, formalize this rule as:

grandFather(Person1, Person3) :-
father(Person1, Person2), father(Person2, Person3).
In this rule we use, next to the predicates two so called logical operators: the words “if” and the word “and”. In this formalization the word “if” is replaced by the symbol “:-“and the word “and” is replaced by a comma “,”. These are the symbols that are used in Prolog to represent logical operators. And clearly in “grandFather(Person1, Person3)” I want to indicate that Person3 is the grandfather of Person1. A formalization like this is called a *rule*. Take a closer look at grammar of the rule. The first part “grandFather(Person1, Person3)” is called the conclusion. The conclusion is followed by the if sign “:-“. After the if-sign there are one or more fact-predicates, separated by commas and ended by a period. In a rule these fact-predicates are called the premisses of the rule. Notice also that “grandFather” starts with a lower case letter. This is Prolog syntax; a predicate must start with a lower case letter. Arguments like “Person1" start with an upper case letter. In Prolog this indicates that it is a variable. It means that in this place you can use any object that is of type person. Of course the rule will not hold for any combination of persons, but that is another matter. Notice that in this rule the premisses are facts. It is also possible to use conclusions of other rules as premisses.

If you know about logic, then you will recognize the pattern that is known as deduction. The difference is that in logic we are used to write something like:
IF p AND q THEN r

while in prolog we would write:
r :- p, q.
When the premisses are true, then the conclusion is also true. This is more or less the case in Prolog.
The general form of a rule is:
conclusion :- premiss1, premiss2, ...,premissN.

6.4 Backward Chaining Algorithms

Goal-driven reasoning, or backward chaining, is an efficient way to solve problems that can be modeled as "structured selection" problems. That is, the aim of the system is to pick the best choice from many enumerated possibilities. For example, an identification problem falls in this category.

The knowledge is structured in rules which describe how each of the possibilities might be selected. The rule breaks the problem into sub-problems.

Example:

Kb contains rule set:

Rule 1: IF A and C Then F

Rule 2: IF A and E Then G

Rule 3: IF B Then E

Rule 4: IF G Then D

Problem: prove

IF A and B TRUE Then D IS TRUE

Logic Interpretation

Rule 1: A^C =>F

Rule 2: A^E =>G

Rule 3: E =>B

Rule 4: G =>D

Problem: prove A^B=>D

Now Prolog Formulation

Rule 1: F:- A , C

Rule 2: G:- A, E

Rule 3: B:- E

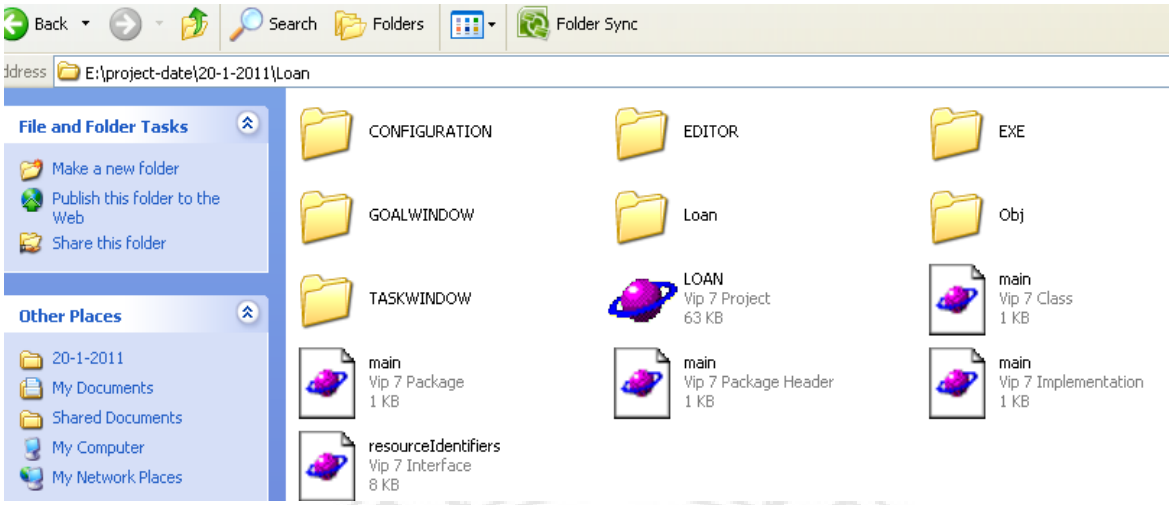
Rule 4: D:- G

Problem: prove D:- A,B

Solution:

1. Start with goal D is true then
   go backward/up till a rule “fires” is found.
   First iteration:
2. Rule 4 fires :conclusion E is true
   New sub goal to prove G is true
   Go backward
3. Rule 2 “ fires”;conclusion: A is true
   New sub goal to prove E is true
   Go backward
4. No other rule fires; end of the first iteration.
   New sub goal found at 3;
   Go for second iteration
   Second iteration:
5. Rule 3 fires:
   Conclusion B is true
   Both inputs A and B ascertained
   Proved

6.5 Different files in the loan project



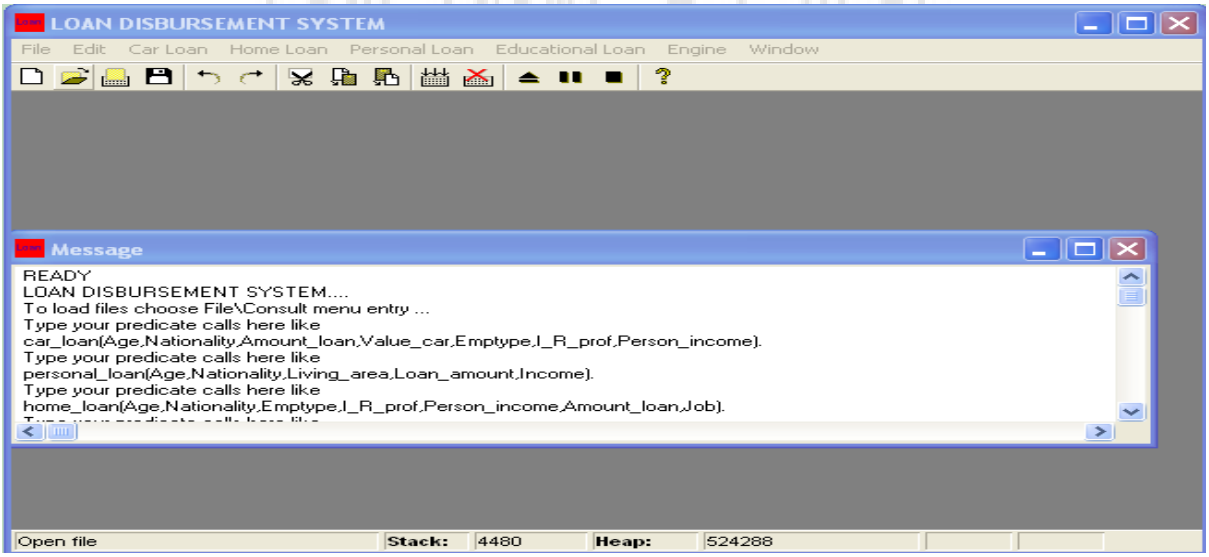
Figures 6.3: Various files in the loan projects

6.6 Input/output of the program

• The Task Menu. It is the menu that contains options like "File","Edit","Car Loan"," Personal Loan", "Home Loan", "Educational Loan", "Engine" and each menu shows different bank's individual loan information.

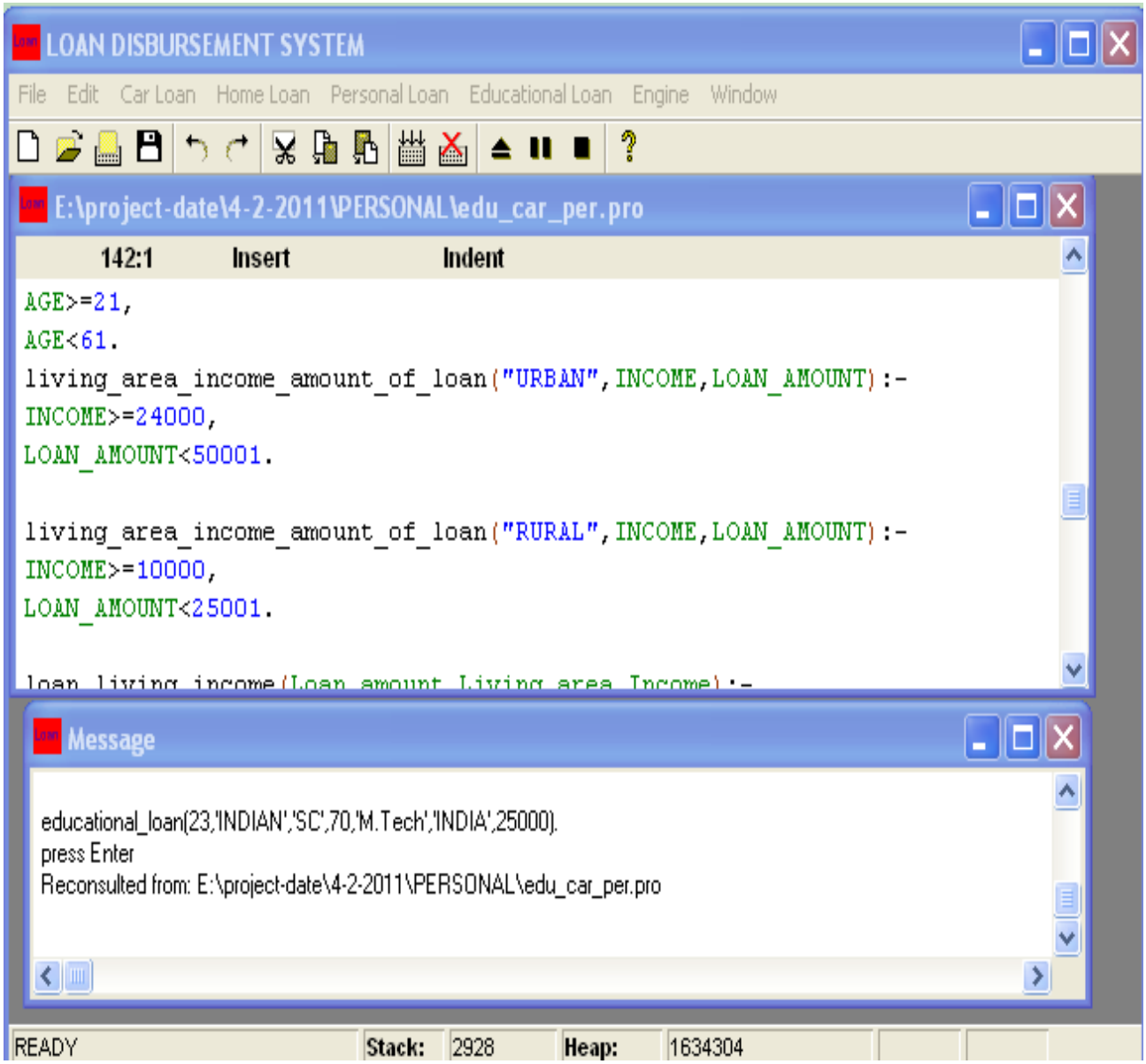
• The Task Menu Bar. It contains icons for the most used options in the Task Menu.

• The Message Window. This window is used to write messages to consult user and engine.



Figures 6.4: Running Inference Engine

After loading the rule base file via file -→open---→path….>consult Engine[Trace on]



Figures 6.5: Loading rule base file into the inference engine

Education loan……

READY
LOAN DISBURSEMENT SYSTEM....
To load files choose File\Consult menu entry ...
Type your predicate calls here like
car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income).
Type your predicate calls here like
personal_loan(Age,Nationality,Living_area,Loan_amount,Income).
Type your predicate calls here like
home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job).
Type your predicate calls here like
educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income).
Example like...
educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
press Enter
Reconsulted from: E:\project-date\4-2-2011\PERSONAL\edu_car_per.pro
Trace is On
educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
Trace: >> CALL: educational_loan(23,INDIAN,SC,70,M.Tech,INDIA,25000)
Trace: >> CALL: educational_loan_age(23)
Trace: >> CALL: 23 >= 16
Trace: >> RETURN: 23 >= 16
Trace: >> CALL: 23 < 40
Trace: >> RETURN: 23 < 40
Trace: >> RETURN: educational_loan_age(23)
Trace: >> CALL: nationality(INDIAN)
Trace: >> RETURN: nationality(INDIAN)
Trace: >> CALL: academic_result(SC,70)
Trace: >> CALL: cast(SC,70)
Trace: >> CALL: 70 >= 50
Trace: >> RETURN: 70 >= 50
Trace: >> RETURN: cast(SC,70)
Trace: >> RETURN: academic_result(SC,70)
Trace: >> CALL: study_at(M.Tech,INDIA)
Trace: >> CALL: hascourse(M.Tech,INDIA)
Trace: >> RETURN: hascourse(M.Tech,INDIA)
Trace: >> RETURN: study_at(M.Tech,INDIA)
Trace: >> CALL: educational_loan_parent(25000)
Trace: >> CALL: 25000 >= 20000
Trace: >> RETURN: 25000 >= 20000

Trace: >> RETURN: educational_loan_parent(25000)
Trace: >> RETURN: educational_loan(23,INDIAN,SC,70,M.Tech,INDIA,25000)
True
1 Solution

Home loan……..

READY
LOAN DISBURSEMENT SYSTEM....
To load files choose File\Consult menu entry ...
Type your predicate calls here like
car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income).
Type your predicate calls here like
personal_loan(Age,Nationality,Living_area,Loan_amount,Income).
Type your predicate calls here like
home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job).
Type your predicate calls here like
educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income).
Example like...
educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
press Enter
Reconsulted from: E:\project-date\4-2-2011\PERSONAL\edu_car_per.pro
Trace is On
home_loan(32,'INDIAN','SALARIED','PASSPORT',300000,3500000,2).
Trace: >> CALL: home_loan(32,INDIAN,SALARIED,PASSPORT,300000,3500000,2)
Trace: >> CALL: nationality(INDIAN)
Trace: >> RETURN: nationality(INDIAN)
Trace: >> CALL: employment_i_r_prof(SALARIED,PASSPORT,32,300000,3500000,2)
Trace: >> CALL: emp_i_r_prof(SALARIED,PASSPORT,32,300000,3500000,2)
Trace: >> CALL: identity_prof(PASSPORT)
Trace: >> RETURN: identity_prof(PASSPORT)
Trace: >> CALL: home_loan_age_s(32)
Trace: >> CALL: 32 >= 21
Trace: >> RETURN: 32 >= 21
Trace: >> CALL: 32 < 61
Trace: >> RETURN: 32 < 61
Trace: >> RETURN: home_loan_age_s(32)
Trace: >> CALL: home_loan_income_s(300000)
Trace: >> CALL: 300000 >= 120000
Trace: >> RETURN: 300000 >= 120000
Trace: >> RETURN: home_loan_income_s(300000)

Trace: >> CALL: amount_of_loan_s(3500000)
Trace: >> CALL: 3500000 < 10000001
Trace: >> RETURN: 3500000 < 10000001
Trace: >> RETURN: amount_of_loan_s(3500000)
Trace: >> CALL: job_exprience_s(2)
Trace: >> CALL: 2 >= 2
Trace: >> RETURN: 2 >= 2
Trace: >> RETURN: job_exprience_s(2)
Trace: >> RETURN: emp_i_r_prof(SALARIED,PASSPORT,32,300000,3500000,2)
Trace: >> RETURN: employment_i_r_prof(SALARIED,PASSPORT,32,300000,3500000,2)
Trace: >> RETURN: home_loan(32,INDIAN,SALARIED,PASSPORT,300000,3500000,2)
True
1 Solution

Personal loan……..

READY
LOAN DISBURSEMENT SYSTEM....
To load files choose File\Consult menu entry ...
Type your predicate calls here like
car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income).
Type your predicate calls here like
personal_loan(Age,Nationality,Living_area,Loan_amount,Income).
Type your predicate calls here like
home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job).
Type your predicate calls here like
educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income).
Example like...

educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
press Enter
Reconsulted from: E:\project-date\4-2-2011\PERSONAL\edu_car_per.pro
Trace is On

personal_loan(25,'INDIAN','URBAN',30000,24000).
Trace: >> CALL: personal_loan(25,INDIAN,URBAN,30000,24000)
Trace: >> CALL: personal_loan_age(25)
Trace: >> CALL: 25 >= 21
Trace: >> RETURN: 25 >= 21
Trace: >> CALL: 25 < 61
Trace: >> RETURN: 25 < 61

Trace: >> RETURN: personal_loan_age(25)
Trace: >> CALL: nationality(INDIAN)
Trace: >> RETURN: nationality(INDIAN)
Trace: >> CALL: loan_living_income(30000,URBAN,24000)
Trace: >> CALL: living_area_income_amount_of_loan(URBAN,24000,30000)
Trace: >> CALL: 24000 >= 24000
Trace: >> RETURN: 24000 >= 24000
Trace: >> CALL: 30000 < 50001
Trace: >> RETURN: 30000 < 50001
Trace: >> RETURN: living_area_income_amount_of_loan(URBAN,24000,30000)
Trace: >> RETURN: loan_living_income(30000,URBAN,24000)
Trace: >> RETURN: personal_loan(25,INDIAN,URBAN,30000,24000)
True
1 Solution

Car loan……

READY
LOAN DISBURSEMENT SYSTEM....
To load files choose File\Consult menu entry ...
Type your predicate calls here like
car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income).
Type your predicate calls here like
personal_loan(Age,Nationality,Living_area,Loan_amount,Income).
Type your predicate calls here like
home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job).
Type your predicate calls here like
educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income).
Example like...

educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
press Enter
Reconsulted from: E:\project-date\4-2-2011\PERSONAL\edu_car_per.pro
Trace is On
car_loan(27,'INDIAN',400000,500000,'SELFEMPLOYED','GASBILL',320000).
Trace: >> CALL: car_loan(27,INDIAN,400000,500000,SELFEMPLOYED,GASBILL,320000)
Trace: >> CALL: car_loan_age(27)
Trace: >> CALL: 27 >= 21
Trace: >> RETURN: 27 >= 21
Trace: >> CALL: 27 < 65
Trace: >> RETURN: 27 < 65

Trace: >> RETURN: car_loan_age(27)
Trace: >> CALL: nationality(INDIAN)
Trace: >> RETURN: nationality(INDIAN)
Trace: >> CALL: amount_of_loan_value_of_car(400000,500000)
Trace: >> CALL: 400000 < 500000
Trace: >> RETURN: 400000 < 500000
Trace: >> RETURN: amount_of_loan_value_of_car(400000,500000)
Trace: >> CALL: employment_i_r_prof(SELFEMPLOYED,GASBILL)
Trace: >> CALL: emp_i_r_prof(SELFEMPLOYED,GASBILL)
Trace: >> CALL: residential_prof(GASBILL)
Trace: >> RETURN: residential_prof(GASBILL)
Trace: >> RETURN: emp_i_r_prof(SELFEMPLOYED,GASBILL)
Trace: >> RETURN: employment_i_r_prof(SELFEMPLOYED,GASBILL)
Trace: >> CALL: car_loan_person_annual_income(320000)
Trace: >> CALL: 320000 >= 250000
Trace: >> RETURN: 320000 >= 250000
Trace: >> RETURN: car_loan_person_annual_income(320000)
Trace: >> RETURN: car_loan(27,INDIAN,400000,500000,SELFEMPLOYED,GASBILL,3200
00)
True
1 Solution

READY
LOAN DISBURSEMENT SYSTEM....
To load files choose File\Consult menu entry ...
Type your predicate calls here like
car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income).
Type your predicate calls here like
personal_loan(Age,Nationality,Living_area,Loan_amount,Income).
Type your predicate calls here like
home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job).
Type your predicate calls here like
educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income).
Example like...

educational_loan(23,'INDIAN','SC',70,'M.Tech','INDIA',25000).
press Enter
Reconsulted from: E:\project-date\4-2-2011\PERSONAL\edu_car_per.pro
Trace is On
car_loan(27,'INDIAN',400000,300000,'SELFEMPLOYED','GASBILL',320000).
Trace: >> CALL: car_loan(27,INDIAN,400000,300000,SELFEMPLOYED,GASBILL,320000)

Trace: >> CALL: car_loan_age(27)
Trace: >> CALL: 27 >= 21
Trace: >> RETURN: 27 >= 21
Trace: >> CALL: 27 < 65
Trace: >> RETURN: 27 < 65
Trace: >> RETURN: car_loan_age(27)
Trace: >> CALL: nationality(INDIAN)
Trace: >> RETURN: nationality(INDIAN)
Trace: >> CALL: amount_of_loan_value_of_car(400000,300000)
Trace: >> CALL: 400000 < 300000
Trace: >> FAIL: 400000 < 300000
Trace: >> FAIL: amount_of_loan_value_of_car(400000,300000)
Trace: >> REDO: nationality(INDIAN)
Trace: >> FAIL: nationality(INDIAN)
Trace: >> REDO: car_loan_age(27)
Trace: >> REDO: 27 < 65
Trace: >> FAIL: 27 < 65
Trace: >> REDO: 27 >= 21
Trace: >> FAIL: 27 >= 21
Trace: >> FAIL: car_loan_age(27)
Trace: >> FAIL: car_loan(27,INDIAN,400000,300000,SELFEMPLOYED,GASBILL,320000)
No solutions

Chapter 7

Conclusions and Future Work

7.1 Conclusions

To introduce a simple Loan Disbursement system in the banking industry first knowledge has to be identified from the Banks. Knowledge needs to be captured from experts such as bank manager, loan manager, assistance loan manager and then create a declarative rule base. Next these rule base file is loaded into the engine. Finally user consult to the system giving query and system response loan is accepted or rejected base on the information.

7.2 Future work

There are several domains within the banking industry where knowledge is applied. These include areas such as Loan or Credit, Foreign Exchange, Trade Finance, Treasury, Information Technology, Risk Management and Business Intelligence etc. In each of these domains knowledge management can be applied. My project work is limited to loan disbursement. However, KM technologies can be extended other areas in banking.

Appendix

Implementing Engine

```
predicates
   stringlist_to_term : (string* List) -> term Term.
clauses
   stringlist_to_term([]) = nill.
   stringlist_to_term([H|Tail]) = list(str(H), stringlist_to_term(Tail)).

predicates
   term_to_slist : (term Term) -> string* List determ.
clauses
   term_to_slist(nill) = [].
   term_to_slist(list(str(S), Tail)) = [S|term_to_slist(Tail)].

predicates
   string_slist : (string String) -> string* Slist.
clauses
   string_slist(String) = [H|string_slist(Rest)] :-
      string::frontToken(String, H, Rest),
      !.
   string_slist(String) = [String].

%- backtracking ------------------------------------------------------------------------
% resetting variables at backtrack.

facts
   resetPoint : (resetPoint) nondeterm.

predicates
   getCurrentChoicePoint : () -> resetPoint.
clauses
   getCurrentChoicePoint() = RP:-
      resetPoint(RP),
      !.
   getCurrentChoicePoint() = _:-
      raise_pie_exception(classInfo, predicate_name(), " Empty stack").

predicates
   backtrackpoint : () multi.
clauses
   backtrackpoint() :-
      RP = resetPoint::new(),
      (  % Stack on forward
         asserta(resetPoint(RP))
```

```
        or % reset on backtrack
            reset(RP),
            fail
        ).

predicates
    reset : (resetPoint CP).
clauses
    reset(RP):-
        TRP = getCurrentChoicePoint(),
        retractAll(resetPoint(TRP)),
        if RP = TRP then
            RP:reset()
        else
            TRP:reset(),
            reset(RP)
        end if.

predicates
    choice : (A, A) -> A multi.
clauses
    choice(A,_) = A.
    choice(_,A) = A.

predicates
    handle_assert : (clauseBase::position, parser::sterm, environment2).
clauses
    handle_assert(Poscode, Term, Env) :-
        CALL = variable::newAnonymous(),
        unify_sTerm(CALL, Term, Env),
        STerm = sTermCreator::toSTerm(CALL),
        assertclause(Poscode, STerm),
        fail.   % Remove generated identifiers from environment
    handle_assert(_, _, _).
predicates
    handle_retract : (term) nondeterm.
clauses
    handle_retract(cmp(":-", [cmp(ID, TermL), Body])) :-
        !,
        clauseBase:named_clause(ID, STermL, SBody),
        backtrackPoint(),
        Env = environment2::new(),
        unify_sTerml(TermL, STermL, Env),
        unify_sTerm(Body, SBody, Env),
        clauseBase:retractclause(parser::cmp(ID, STermL), SBody).
```

```
    handle_retract(cmp(":-", [var(HeadVar), Body])) :-
        !,
        clauseBase:clause(SHead, SBody),
        backtrackPoint(),
        Env = environment2::new(),
        unify_sTerm(var(HeadVar), SHead, Env),
        unify_sTerm(Body, SBody, Env),
        clauseBase:retractclause(SHead, SBody).

    handle_retract(cmp(ID, TermL)) :-
        clauseBase:named_clause(ID, TermL1, parser::atom("true")), % facts
        backtrackPoint(),
        Env = environment2::new(),
        unify_sTerml(TermL, TermL1, Env),
        clauseBase:retractclause(parser::cmp(ID, TermL1), parser::atom("true")).

predicates
    unify_sTerml : (terml, parser::sterml, environment2) determ.
clauses
    unify_sTerml([], [], _) :-
        !.
    unify_sTerml([Term1|TL1], [Term2|TL2], Env) :-
        unify_sTerm(Term1, Term2, Env),
        unify_sTerml(TL1, TL2, Env).

predicates
    unify_sTerm : (term, parser::sterm, environment2) determ.
clauses
    unify_sTerm(Term, STerm, Env) :-
        unify(Term, Env:mk_term(STerm)).

clauses
    boundTerm(T):-
        not(normalize(T) = var(_)).

clauses
    freeTerm(T):-
        normalize(T) = var(_).

clauses
    unify(A, B) :-
        unify_1(normalize(A), normalize(B)).

predicates
    unify_1 : (term A, term B) determ.
clauses
```

```
    unify_1(list(H1, T1), list(H2, T2)) :- !,
        unify(H1, H2),
        unify(T1, T2).
    unify_1(cmp(ID, L1), cmp(ID, L2)) :- !,
        unify_list(L1, L2).
    unify_1(T, T) :- !.
    unify_1(T, var(Var)) :- !,
        Var:setTerm(T,getCurrentChoicePoint()).
    unify_1(var(Var), T) :- !,
        Var:setTerm(T,getCurrentChoicePoint()).
    unify_1(atom(T), str(T)).
    unify_1(str(T), atom(T)).


predicates
    unify_list : (terml AL, terml BL) determ.
clauses
    unify_list([], []).
    unify_list([A|AL], [B|BL]) :-
        unify(A, B),
        unify list(AL, BL).
clauses
    normalize(var(Var)) = Var:getTerm() :- !.
    normalize(T) = T.

clauses
    normalizeList([]) = [].
    normalizeList([H|T]) = [normalize(H)|normalizeList(T)].

predicates
    unify_body : (parser::sterm, environment2, programControl::stackMark) nondeterm.
clauses
    unify_body(parser::atom("true"), _, _) :-
        !.

    unify_body(parser::cmp(",", [Term1, Term2]), Env, BTOP) :-
        !,
        unify_body(Term1, Env, BTOP),
        unify_body(Term2, Env, BTOP).

    unify_body(parser::atom("!"), _, BTOP) :-
        !,
        programControl::cutBackTrack(BTOP).

    unify_body(parser::cmp(";", [Term1,Term2]), Env, BTOP) :-
        Term = choice(Term1,Term2),
```

```
        backtrackpoint(),
        unify_body(Term, Env, BTOP).

unify_body(parser::cmp("not", [Term]), Env, _) :-
    backtrackPoint(),
    RS = getCurrentChoicePoint(),
    BTOP = programControl::getBackTrack(),
    unify_body(Term, Env, BTOP),
    reset(RS),
    !,
    fail.

unify_body(parser::cmp("not", _), _, _) :-
    !.

unify_body(parser::cmp("call", [Term]), Env, _) :-
    !,
    BTOP = programControl::getBackTrack(),
    unify_body(Term, Env, BTOP),
    backtrackpoint().

unify_body(parser::cmp("assert", [Term]), Env, _) :-
    !,
    handle_assert(clauseBase::last, Term, Env).

unify_body(parser::cmp("asserta", [Term]), Env, _) :-
    !,
    handle_assert(clauseBase::first, Term, Env).

unify_body(parser::cmp("assertz", [Term]), Env, _) :-
    !,
    handle_assert(clauseBase::last, Term, Env).

unify_body(parser::cmp(PID, TermL), Env, _) :-
    Args = Env:mk_term_list(TermL),
   trace_call(PID, Args).

unify_body(parser::var(ID), Env, _) :-
    !,
    VarTerm = Env:lookUpOrCreate(ID),
    trace_call_var(normalize(VarTerm)).

unify_body(parser::atom(PID), _, _) :-
    trace_call(PID, []).

predicates
```

```
   trace_call_var : (term) nondeterm.
clauses
   trace_call_var(atom(PID)) :-
      trace_call(PID, []).
   trace_call_var(cmp(PID, TermL)) :-
      trace_call(PID, TermL).

predicates
   trace_call : (string, terml) nondeterm.
clauses
   trace_call(PID, TermL) :-
      if retract(stop_execution) then
         retract(topmost(BTOP)),
         programControl::cutBackTrack(BTOP),
         fail
      else
         if not(check_pause_execution) then
            trace_call1(PID, normalizeList(TermL))
         end if
      end if.

predicates
   trace_call1 : (string, terml) nondeterm.
clauses
   trace_call1(PID, TermL) :-
      not(traceflag),
      !,
      CallName = string::toLowerCase(PID),
      call(CallName, TermL).
   trace_call1(PID, TermL) :-
      CallName = string::toLowerCase(PID),
      showtrace("CALL:   ", PID, TermL),
      call(CallName, TermL),
      report_redo(CallName, TermL),
      showtrace("RETURN: ", PID, TermL).
   trace_call1(PID, TermL) :-
      showtrace("FAIL:   ", PID, TermL),
      fail.

facts
   isPredicateSupportedResponder : isPredicateSupportedResponder:= defaultIsPredic
ateSupportedResponder.

predicates
   defaultIsPredicateSupportedResponder : isPredicateSupportedResponder.
clauses
```

```
    defaultIsPredicateSupportedResponder(_,_) = _:-
      fail.

predicates
    call : (string, terml) nondeterm.
clauses
    call(PredicateId, ArgList) :-
        Invoker=isPredicateSupportedResponder(This,PredicateId),
        !,
        Invoker(This,PredicateId,ArgList),
        backtrackpoint().

    call("fail", []) :-
        !,
        fail.

    call("repeat", []) :-
        !,
        repeat,
        backtrackpoint().

    call("for", [T, int(FROM), int(TO)]) :-
        !,
        I = fromTo(FROM, TO),
            backtrackpoint(),
            unify(T, int(I)).

    call("halt", []) :-
        !,
        retract(topmost(TOP)),
        program Control::cutBackTrack(TOP),
        stdio::write("Execution terminated"), stdio::nl,
        fail.

    call("write", TermL) :-
        !,
        writeterml(output::write(), TermL).

    call("nl", []) :-
        !,
        stdio::nl.
    call("display", TermL) :-
        !,
        writeterml(output::display(), TermL).
    call("eof", []) :-
        !,
```

```
    fileHandling::handle_eof.

call("retract", [Term]) :-
    !,
    handle_retract(Term).  %backtrackPoint called in handle_retract

call("tell", [str(Filename)]) :-
    !,
    if file::existExactFile(FileName) then
        fileSystem_native::fileAttributeReadOnly <> bit::bitand(fileSystem_native::getFile
Attributes(FileName), fileSystem_native::fileAttributeReadOnly)
    end if,
    fileHandling::tell(Filename).

call("telling", [T]) :-
    !,
    Filename = fileHandling::telling_name,
    unify(T, str(Filename)).

call("told", []) :-
    !,
    fileHandling::told.

call("see", [str(Filename)]) :-
    !,
    if file::existExactFile(FileName) then
      fileHandling::see(Filename)
    end if.

call("seeing", [T]) :-
    !,
    fileHandling::seeingP(Filename),
    unify(T, str(Filename)).
call("seen", []) :-
    !,
    fileHandling::seen.

call("=..", [cmp(ID, TermL), Term]) :-
    !,
    list_terml(LIST, TermL),
    unify(Term, list(atom(ID), LIST)).

call("=..", [Term, list(atom(ID), LIST)]) :-
    !,
    list_terml(LIST, TermL),
    unify(Term, cmp(ID, TermL)).
```

```
call("arg", [int(N), cmp(_, TermL), R]) :-
    !,
    N>0,
    unify(R, arg(N, TermL)).

call("functor", [Term, FID, ARITY]) :-
    !,
    functor(Term, FID, ARITY).

call("clause", [Head, Body]) :-
    !,
    clauseBase:clause(SHead, SBody),
    backtrackpoint(),
    Env = environment2::new(),
    unify_sTerm(Head, SHead, Env),
    unify_sTerm(Body, SBody, Env).

call("concat", [str(A), str(B), TC]) :-
    !,
    C = string::concat(A, B),
    unify(TC, str(C)).

call("concat", [TA, str(B), str(C)]) :-
    !,
    CH = string::searchLast(C, B, string::caseSensitive),
    string::front(C, CH, A, _),
    unify(TA, str(A)).

call("concat", [str(A), TB, str(C)]) :-
    !,
    CH0 = string::search(C, A),
    CH1 = string::length(A),
    string::front(C, CH0+CH1, _, B),
    unify(TB, str(B)).

call("str_int", [str(STR), TI]) :-
    !,
    I = math::tryToInteger(STR),
    unify(TI, int(I)).

call("str_int", [TS, int(I)]) :-
    !,
    STR = toString(I),
    unify(TS, str(STR)).
```

```
call("str_atom", [str(STR), T]) :-
    !,
    unify(T, atom(STR)).

call("str_atom", [T, atom(STR)]) :-
    !,
    unify(T, str(STR)).

call("is", [R, T2]) :-
    !,
    unify(R, int(evalInt(T2))).

call("==", [T1, T2]) :-
    !,
    eeq(T1, T2).

call("\\==", [T1, T2]) :-
    !,
    not(eeq(T1, T2)).

call("=", [T1, T2]) :-
    !,
    unify(T1, T2).

call("\\=", [X, Y]) :-
    !,
    not(X=Y).

call("<", [T1, T2]) :-
    !,
    evalInt(T1) < evalInt(T2).

call(">", [T1, T2]) :-
    !,
    evalInt(T1) > evalInt(T2).

call("=<", [T1, T2]) :-
    !,
    evalInt(T1) <= evalInt(T2).

call("<=", [T1, T2]) :-
    !,
    evalInt(T1) <= evalInt(T2).

call(">=", [T1, T2]) :-
    !,
```

```
    evalInt(T1) >= evalInt(T2).

call("><", [T1, T2]) :-
    !,
    evalInt(T1) <> evalInt(T2).

call("<>", [T1, T2]) :-
    !,
    evalInt(T1) <> evalInt(T2).

call("integer", [T]) :-
    !,
    T = int(_).

call("var", [T]) :-
    !,
    T = var(_).

call("nonvar", [Term]) :-
    !,
    not(Term = var(_)).

call("list", TermL) :-
    handle_list(TermL).

call("trace", _) :-
    traceflag,
    !.

call("trace", _) :-
    !,
    assert(traceflag).

call("notrace", _) :-
    !,
    retractall(traceflag).

call("char_int", [char(CH), T]) :-
    !,
    INT= convert(integer, string::getCharValue(CH)),
    unify(T, int(INT)).

call("char_int", [T, int(INT)]) :-
    !,
    CH = string::getCharFromValue(tryConvert(unsigned16, INT)),
    unify(T, char(CH)).
```

```
    call("consult", [Term]) :-
        !,
        getfilename(Term, Filename), cons(Filename).

    call("reconsult", [Term]) :-
        !,
        getfilename(Term, Filename), recons(Filename).
    call("save", [Term]) :-
        !,
        getfilename(Term, Filename),
        if file::existExactFile(FileName) then
            fileSystem_native::fileAttributeReadOnly <> bit::bitand(fileSystem_native::getFile
Attributes(FileName), fileSystem_native::fileAttributeReadOnly)
        end if,
        sav(Filename).

    call("op", [Prior, Assoc, Op]) :-
        !,
        operator::getop_nd(P1, A1, O1),
            backtrackpoint(),
            unify(Prior, int(P1)),
            unify(Assoc, atom(A1)),
            unify(Op, atom(O1)).
```

Implementing trace

```
predicates
   check_pause_execution : () failure.
clauses
   check_pause_execution() :-
      not(pause_execution),
      !,
      fail.

predicates
   report_redo : (string, terml) multi.
clauses
   report_redo(_, _).
   report_redo(PID, TermL) :-
      showtrace("REDO:   ", PID, TermL),
      fail.

predicates
   showtrace : (string, string, terml) procedure.
clauses
   showtrace(STR, PID, TermL) :-
      stdio::write("Trace: >> ", STR),
      output::wterm(output::write(), cmp(PID, TermL)),
      stdio::nl,
```

Rule codification

%@EDUCATIONAL LOAN AND CAR LOAN HOME LOAN PERSONAL LOAN SANCTI
ON PROCEDURE
nationality("INDIAN").

hascourse("B.Tech","INDIA").
hascourse("M.Tech","INDIA").
hascourse("BE","INDIA").
hascourse("ME","INDIA").
hascourse("MEDICAL","INDIA").
hascourse("B.Tech","FOREIGN").
hascourse("M.Tech","FOREIGN").
hascourse("BE","FOREIGN").
hascourse("ME","FOREIGN").
hascourse("MEDICAL","FOREIGN").
hascourse("MASTER","FOREIGN").
hascourse("BACHELOR","FOREIGN").
identity_prof("PASSPORT").
identity_prof("VOTERID").
identity_prof("PANCARD").
residential_prof("GASBILL").
residential_prof("ELECTRICITYBILL").
residential_prof("TELEPHONEBILL").

educational_loan_age(AGE):-
AGE>=16,
AGE<40.

educational_loan_parent(INCOME):-
INCOME>=20000.

cast("SC",MARKS):-
MARKS>=50.

cast("ST",MARKS):-
MARKS>=50.

cast("OBC",MARKS):-
MARKS>=60.

cast("OC",MARKS):-
MARKS>=60.
academic_result(CAST,MARKS):-
cast(CAST,MARKS).

study_at(COURSE,COUNTRY):-

```
hascourse(COURSE,COUNTRY).

educational_loan(Age,Nationality,Cast,Marks,Course,Country,Parent_income):-
educational_loan_age(Age),
nationality(Nationality),
academic_result(Cast,Marks),
study_at(Course,Country),
educational_loan_parent(Parent_income).

car_loan_age(AGE):-
AGE>=21,
AGE<65.

car_loan_person_annual_income(INCOME):-
INCOME>=250000.

amount_of_loan_value_of_car(AMOUNT_LOAN,VALUE_CAR):-
AMOUNT_LOAN<VALUE_CAR.

emp_i_r_prof("SALARIED",I_R_PROF):-
identity_prof(I_R_PROF).

emp_i_r_prof("SELFEMPLOYED",I_R_PROF):-
residential_prof(I_R_PROF).

employment_i_r_prof(EMPTYPE,I_R_PROF):-
emp_i_r_prof(EMPTYPE,I_R_PROF).

car_loan(Age,Nationality,Amount_loan,Value_car,Emptype,I_R_prof,Person_income):-
car_loan_age(Age),
nationality(Nationality),
amount_of_loan_value_of_car(Amount_loan,Value_car),
employment_i_r_prof(Emptype,I_R_prof),
car_loan_person_annual_income(Person_income).
personal_loan_age(AGE):-
AGE>=21,
AGE<61.
living_area_income_amount_of_loan("URBAN",INCOME,LOAN_AMOUNT):-
INCOME>=24000,
LOAN_AMOUNT<500000.

living_area_income_amount_of_loan("RURAL",INCOME,LOAN_AMOUNT):-
INCOME>=10000,
LOAN_AMOUNT<250000.

loan_living_income(Loan_amount,Living_area,Income):-
```

```
living_area_income_amount_of_loan(Living_area,Income,Loan_amount).

personal_loan(Age,Nationality,Living_area,Loan_amount,Income):-
personal_loan_age(Age),
nationality(Nationality),
loan_living_income(Loan_amount,Living_area,Income).
home_loan_age_S(AGE):-
AGE>=21,
AGE<61.

home_loan_income_S(INCOME):-
INCOME>=120000.

job_exprience_S(JOB):-
JOB>=2.

amount_of_loan_S(AMOUNT_LOAN):-
AMOUNT_LOAN<10000001.

emp_i_r_prof("SALARIED",I_R_PROF,AGE,INCOME,AMOUNT_LOAN,JOB):-
identity_prof(I_R_PROF),
home_loan_age_S(AGE),
home_loan_income_S(INCOME),
amount_of_loan_S(AMOUNT_LOAN),
job_exprience_S(JOB).

home_loan_age_Self(AGE):-AGE>=21,AGE<71.
home_loan_income_Self(INCOME):-INCOME>=200000.
amount_of_loan_Self(AMOUNT_LOAN):-AMOUNT_LOAN<20000001.
job_exprience_Self(JOB):-JOB>=3.

emp_i_r_prof("SELFEMPLOYED",I_R_PROF,AGE,INCOME,AMOUNT_LOAN,JOB):-
residential_prof(I_R_PROF),
home_loan_age_Self(AGE),
home_loan_income_Self(INCOME),
amount_of_loan_Self(AMOUNT_LOAN),
job_exprience_Self(JOB).

employment_i_r_prof(EMPTYPE,I_R_PROF,AGE,INCOME,AMOUNT_LOAN,JOB):-
emp_i_r_prof(EMPTYPE,I_R_PROF,AGE,INCOME,AMOUNT_LOAN,JOB).

home_loan(Age,Nationality,Emptype,I_R_prof,Person_income,Amount_loan,Job):-
nationality(Nationality),
employment_i_r_prof(Emptype,I_R_prof,Age,Person_income,Amount_loan,Job).
```

Bibliography

1. Takeuchi, H., & Nonaka, I. (1995).Theory of Organizational Knowledge Creation
2. Morey, D., Maybury, M., & Thuraisingham, B. (Eds.) (2001). *Knowledge Management: Classic and Contemporary Works*. Massachusetts: MIT Press.
3. Polanyi, M. (1966).*The Tacit Dimension*. London: Penguin Books.
4. Nonaka, I. (1991).The Knowledge Creating Company. *Harvard Business Review,* 69, 96-104.
5. Merritt, D.(2009).Building Expert Systems in Prolog, 358,1989-07.
6. Boer, T.(2010).A Beginners Guide to Visual Prolog:Comprehensive Book
7. Puthran, V. (2008). Knowledge management (km) in banks. University of Mumbi.
8. Joseph, M. & McElroy, M.(2005).Doing Knowledge Management ,Vol. 12, No.2.
9. Eid,N.(2008).Knowledge Management on Transforming Tele-centres into networked Knowledge Hubs:ESCWA Press.
10. Costa,E.(2010).Visual Prolog for Tyros: Comprehensive Book.
11. Retrieved from http://www.trainmor-knowmore.eu/D646C11F.en.aspx
12. Retrieved from http://www.cognitivedesignsolutions.com/KM/Cycle.html
13. Retrieved from http://www.deal4loans.com
14. Retrieved from http://www.epistemics.co.uk/Notes/63-0-0.html
15. Retrieved from http://kmlearning.blogspot.com/2007/07/knowledge-management-process.html