Rough Set Techniques for 24 Hour Knowledge Factory

Niti Verma[1], **Neha Verma**[2] and **A. B. Patki**[3]

Submitted in May 2011; Accepted in November 2011

Abstract - 24 Hour Knowledge Factory [1] is the work culture that incorporates different people contributing together in collaborated manner on various modules of the same project. But as advancements occurred, it was found that the approach is as difficult to realize as it is to imagine. The smooth work flow amidst the personnel demands attention. This paper discusses a software solution to easily implement this idea by designing a workflow system between the programmers who are working in the different places in 24-Hour realm. The software presents the user interfaces to enable an employee to grasp the work done until now easily. The interface creates optimized tables generated using the rough set theory. This theory gives us a fair view of the work required by providing lower and upper approximation along with various rules that could help us to find these optimum sets. Software also facilitates the developer at the immediate next shift to be sure of the code in which he is going to work.

Index Terms - 24 Hour Knowledge Factory; Workflow Design, Rough set; Upper Approximation; Lower Approximation; 24- Hour Development; Follow the Sun

1. INTRODUCTION

24 hour knowledge factory may be considered as a process of working shifts at different places which are not only geographically distant but also temporally far from each other [1]. It involves collaboration of three or more centers in different time zones handing over work to each other in shifts. The centers are connected using internet or dedicated networks which are used to pass knowledge from one work location to other. Each center completes its work in its given time and then the work is handed over to another center which has the day time corresponding to this center's night period. This is practiced until whole 24 hour cycle is completed.

The concept of 24 hour Knowledge factory is not new. The work to improvise it is now for more than a decade old now. The past work are summarized in [1,13,14] carrying different perspective towards the problem. All have discussed the problem of bringing it to life very effectively. The commercial products based on "Follow the sun" alias 24 hour knowledge Factory were also introduced by IBM and HP in market [15]. For the effective utilization of sequential workers distributed across time-zones, tasks must be broken down so that they require no interaction with peers. In addition, effort required in transitioning from one employee to the next should be minimal. This paradigm requires new methodologies and tools that will allow an individual to understand in 16 minutes, the work done by others in the preceding 16 hours [7]. Also, this model requires introduction of time and state as search parameters for knowledge discovery in order to enable individuals to understand the sequence of changes being made in a project. These requirements of a 24 hour knowledge factory can be solved with the introduction of a Composite Personae (CP).

[1]*Indira Gandhi Institute of Technology, New Delhi, India*
[2]*Maharaja Surajmal Institute of Technology, New Delhi, India*
[3]*Ex-Senior Director and HOD, Department of Information Technology, MCIT, New Delhi*
E-Mail: [1]*nitiverma.cs@gmail.com*, [2]*cs.nehaverma@gmail.com* and [3]*patki@nic.in*

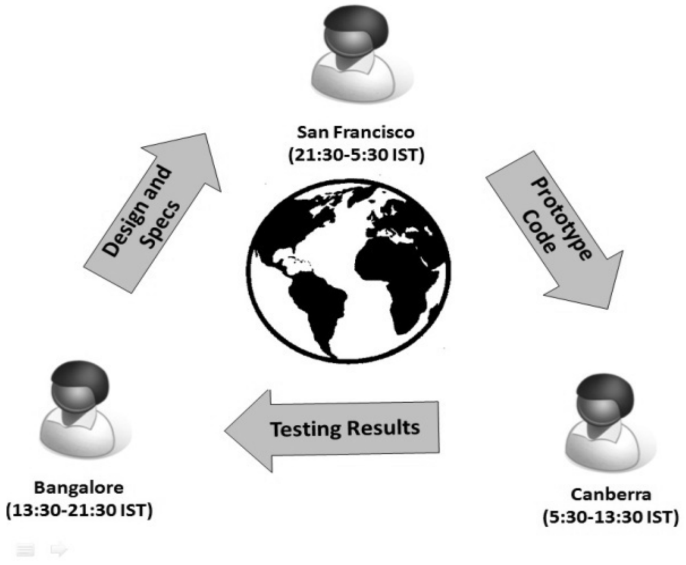


Figure 1: Cycle in 24 Hour Knowledge Factory [Adopted from Ref. 1]

A CP is a highly cohesive micro-team that posses simultaneous properties of both an individual and a collection of individuals. It is designed to act like a singular entity even though it comprises of three or more individuals at multiple sites. In a CP, only one site is active at a single point in time. Thus, development can proceed in a manner similar to traditional one, with the difference being that a component is owned by a CP and not by an individual. It is actively involved in the process of development and conflict resolution on a round-the-clock basis [1].

The present day operating systems do not have adequate support to facilitate this concept. Application of rough set theory gives us an idea about the work that has to be done by providing the lower and the upper approximation and various rules that could help to find these optimum tables. The paper consists of five sections ahead defining motivation for this software, revisiting rough set theory, implementation of rough set to 24 hour knowledge factory, future work and conclusion.

2. MOTIVATION FOR SOFTWARE

In analogous to the code generation in shifts, consider three people working on same wall building in temporal shift. The specification for construction of wall indicates that each brick is to be horizontally placed in the wall. Person A does his work efficiently by placing brick horizontally and finishes one third of the wall at the end of the shift. Now, another person B is handed over the work in next shift who inefficiently places the brick vertically and delivers the same work to person C at the end of his shift, person C is unaware of the inefficient work done by Person B and continuous to do his own work efficiently by placing the bricks horizontally. At this stage, if this work is handed over to person A in the next cycle, how can he be sure if the work done until now has been efficiently done?

Likewise, in the code building, each company has some norms to use e.g. 'if then else', 'else if', 'if then' statements in code building. Wrong practices of using these commands by the developer in their code without following the protocols make the code partially inefficient for the company's norms. This inefficiency can be checked by 24 hour knowledge factory and the developer at the next shift can be sure of the code in which he is going to invest time in. Possibly, also find out which developer delivered the inefficient work through history log.

One may argue that Concurrent Versions System (CVS) [3] and knowledge factory, both work on the basic idea of maintaining the history of database of the same projects with the temporal differences but there lies a considerable amount of difference in the processing and maintaining the relevant information in the database.

CVS just acts as a repository of information whether it's a code or it's an author of the code, whereas the knowledge factory maintains the 'knowledge' that is the relevant and useful information only. Knowledge factory incorporates the software maintaining this useful knowledge instead of acting as a repository and saving all the information available.

Knowledge factory is capable of eliminating the redundant data provided by the user and retrieving it at the time of the need where as CVS just retrieves the data stored previously without processing it. Knowledge factory can process the code written and can differentiate its basic components like classes, modules, basic elements etc. where as CVS is not capable of this at all.

3. ROUGH SET THEORY

Rough set theory has been used in many applications varying from fault diagnosis to economic predictions [2,4,6]. It basically gives the crisp idea of selecting and deselecting the component of the entity based on its lower and upper approximation. Likewise, one can decide over the lower and upper approximation of the given entity in the case of vagueness and uncertainty. The classical Rough Set Theory was introduced by Zdzisław I. Pawlak [5,10,11] in 1982.

3.1 Indiscernibility

Let A'= (U; A) be an information system, then with any $B \subseteq A$ there is associated an equivalence relation $IND_{A'}(B)$:

$$IND_{A'}(B)=\{(x,x') \in U^2 \mid \forall a \in B \; a(x) = a(x') \}$$

where $IND_A(B)$ is called the B- Indiscernibility relation. If $(x, x') \in IND_{A'}(B)$, then objects x and x' are indiscernible from each other by attributes from B. The equivalence classes of the B- Indiscernibility relation are denoted $[x]_B$

3.2 Set Approximation

If $B \subseteq A$ and $X \subseteq U$. We can approximate X from the information contained in B by constructing the B-lower and B-upper approximations of X, denoted $\underline{B}X$ and $\overline{B}X$ respectively, where

$$\underline{B}X = \{x \mid [x]_B \subseteq X\}$$
$$\overline{B}X = \{x \mid [x]_B \cap X \neq \emptyset\}$$

The accuracy of the rough-set representation of the set X is defined as:-

$$\mu_B(X) = |\underline{B}X| / |\overline{B}X|$$

The accuracy of the rough set representation of X, $\mu_B(X)$, $0 \leq \mu_B(X) \leq 1$, is the ratio of the number of objects which can be placed in X to the number of objects that can possibly be placed in X.

4. REDUCTS

From an information system, some attributes can be deleted while keeping necessary attributes. The least minimal subset of attributes which ensures the same quality of classification as the set of all attributes is called a reduct in A'. Intersection of all reducts is called the core. The core is a collection of the most significant attributes for the classification in the system.

5. RULE GENERATION

Rules represent extracted knowledge, which can be used when classifying new objects. Rules are created from the condition attribute values of the object class. They are presented in the form if "IF else" statement. A decision part comprises the resulting part of the rule. Rules that have same conditions but different decisions are called inconsistent rules.

6. ROUGH SET THEORY IN 24 HOUR KNOWLEDGE FACTORY

24 hour knowledge factory require the better infrastructure and workflow design for providing the interface to the programmer joining in second shift. This knowledge-rich workflow environments [16,17] uses rough set approach [8,9] which provides the needed information in exact and intelligent way in the compact form. It is implemented using the C# win forms application. The part of class diagram can be seen in Appendix I. We have made various assumptions regarding our project and these are as stated below:

a. All the Hardware supports of different places are assumed to be equal.
b. There should be a predefined format for the security password and Login IDs. These can be programmer's ID.
c. Current project Software requirement specifications are to be provided for the assessment of various data fields. The various upper and lower approximations of the SRS have to be matched against those obtained when the work is passed to the module developed, at the end of a particular shift.
d. SRS must specify the number of objects, classes, functions and number of modules in the project.

The User needs to login for accessing the knowledge factory. This can be integrated with the corporate accounts login in later times. User may be able to work on three different sides viz, developer, and documenter and tester side.

The win forms for one side of application is the developer side. By working on the developer side we maintain a form which enable us to view three different tables [12] which are necessary from the developer's point of view that should be provided to the other developer working at the another shift of same project. Various tables that were designed are information metric table, history log table and modular table. Modular table tell us the name of module, approach maintained, function name, date of start and completion as also the language hardware and software used for designing the particular module have been used. Later the rough set approach is used to provide just useful information to the other developer which is mandatory for him to know and proceed further for his task. If we do not use rough set approach than it will result in lot of waste of time in reading a part of work done by one developer and deciding what next is to be done.

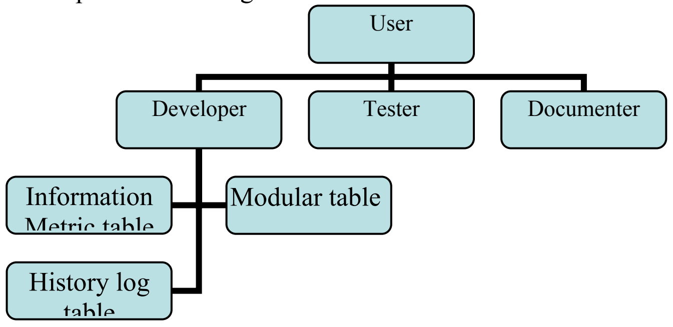


Figure 2: Design Hierarchy

Beside this, when our current approximations in rough sets are compared with the required specified approximations (provided by SRS), the developer ahead is more confident of work done by previous developer.

7. IMPLEMENTATION

We have considered various classes which will have various objects and functions which are to be considered during attribute generation. Thus, we are considering two different way of attribute generation-

a. Object of various classes may be considered to create attribute table
b. Function in the various classes.

Therefore, we have considered two different tables in our project. They will be filled in explicitly by the programmer for the time being. Now, various rough set rules can be used to find the optimum set attributes and then the upper approximation and the lower approximation will be found with help of rough set rules.

8. COMPARISON WITH THE SRS

It is essential to have software requirement specifications which show the desired set of classes, functions and objects. The Information given by the SRS will be used to find the desired lower approximation and upper approximation of the project.

Suppose the Lower Approximation and Upper Approximation of the SRS is given by Ls and Us and their current code is given by the Lc and Uc respectively. Then the calculation will be

- Ls should be equal to Lc.
- Us should be equal to Uc

It may not be the exactly equal until the code is complete. Lc and Uc will be some percent of Ls and Us. This will be given by –

$$\text{Current lower error} = Lc / Ls \times 100$$
$$\text{Current upper error} = Uc / Us \times 100$$

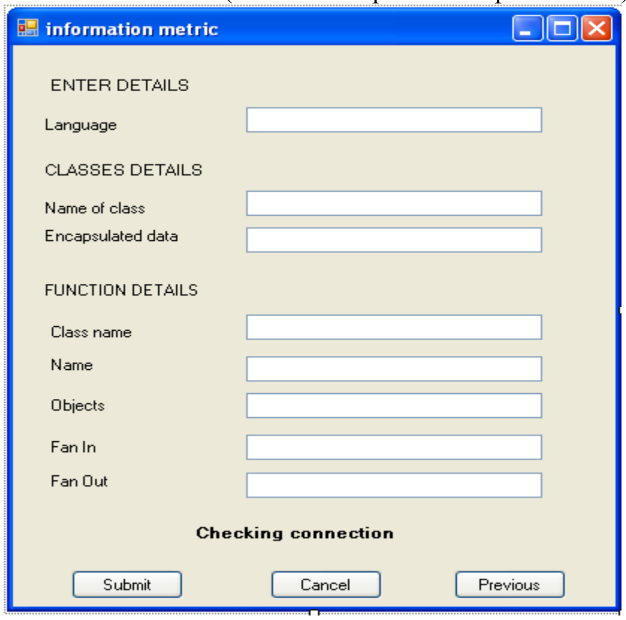
This error should lie between minimum given range. This value can be used to keep the track of code being written in the discrete time domains and it will still assure the programmer writing the current code, that he is following the correct code which was written by previous programmer.

9. PROJECT WINDOW FORMS

Three point of views in 24 hour knowledge factory are considered as there are three main strata of people which are involved in the software project generation i.e. Developer, tester and documenter. We concentrated on code developer view point in this paper.

The developer view includes three Information Metric table, Module table and History Log to store the information about the code developed. The Information Metric Table stores general information about the module developed (figure3). The Module Table stores the specific information about the module developed (figure4). The History Log stores date and time details of the module developed (figure5).

The data is collected by each developer at the end of his shift in these three tables. This data is stored in the database which is maintained at the backend. Now, depending upon the decision attributes Lower Approximation (LA) and Upper Approximation (UA) are found by using the algorithm for approximation computation [8]. The sets which are indiscernible can be reduced to single tuple and optimize the subsets to a certain degree. The approximation is specifically calculated to match the Lower and Upper approximation with those stated in the SRS (Software Requirement Specification).


information metric

ENTER DETAILS

Language

CLASSES DETAILS

Name of class

Encapsulated data

FUNCTION DETAILS

Class name

Name

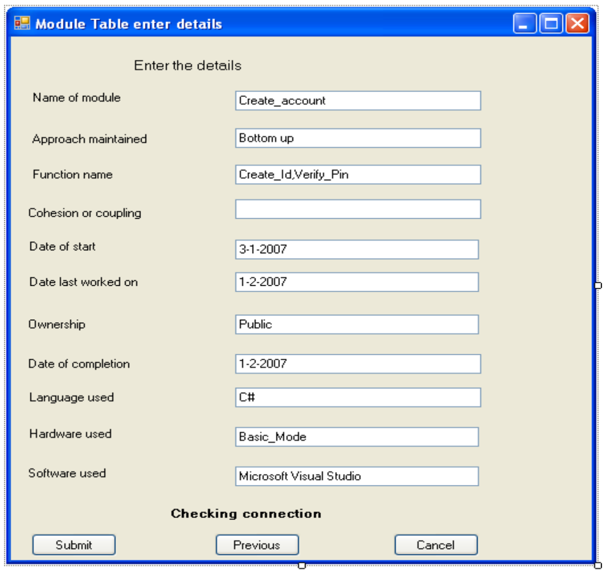
Objects

Fan In

Fan Out

Checking connection

Figure 3: Information Table


Module Table enter details

Enter the details

Name of module	Create_account
Approach maintained	Bottom up
Function name	Create_Id,Verify_Pin
Cohesion or coupling	
Date of start	3-1-2007
Date last worked on	1-2-2007
Ownership	Public
Date of completion	1-2-2007
Language used	C#
Hardware used	Basic_Mode
Software used	Microsoft Visual Studio

Checking connection

Figure 4: Module Table

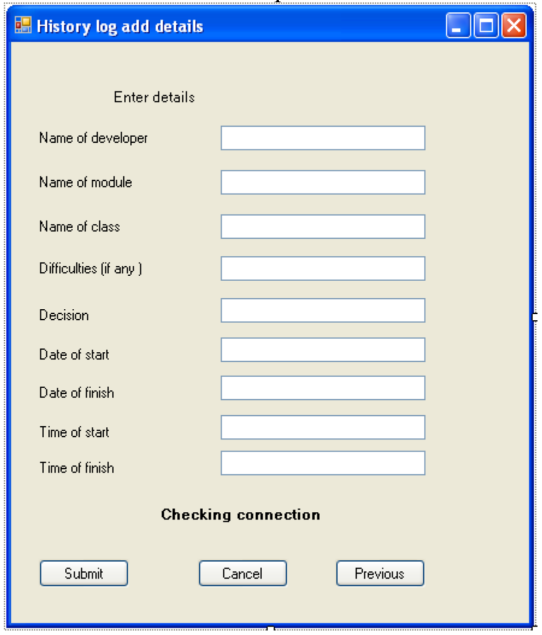
Similarly, reducts can be found by applying the reduct and core computation algorithm [8]. It is a relative reduct that contain same amount of information that is held with non reduced data set. Hence, we can call it as extracted data. This is data which will be shown to the developer working at the immediate next shift to brief him about the status of the work done.

FUTURE WORK

The event of login can be connected to a database maintained specifically for verification of user name and password. Currently a predefined user name and password are being used for verification of login.

The Software Requirement Specification is an essential input in any project. In the case where SRS tends to change, due to the market requirement changes, or any other reason, the whole of the input changes which tends to disturb the output in an unexpected way. And at worst could hamper the work progress which is the main goal of this 24 hour knowledge factory. This could be changed by making some amendments and predefined norms at the time of SRS agreement. Or a system which is ready to accept the changes made in SRS and is not that depend on it.

Currently only the developer's side of view has been considered. The other two sides i.e. the tester's and the documenter's also need to be implemented.


History log add details

Enter details

Name of developer

Name of module

Name of class

Difficulties (if any)

Decision

Date of start

Date of finish

Time of start

Time of finish

Checking connection

Figure 5: History Table

CONCLUSION

The goal of this paper is to generate interface for managing the 24 hour knowledge factory by implementing rough set theory. The software is capable of handling work different places which are not only geographically distant but also temporally far from each other by easily grasping the idea as quickly as possible by going through the optimised tables. Software also facilitates the developer at the immediate next shift to be sure of the code in which he is going to work.

ACKNOWLEDGMENTS

We thank engineering college authorities to encourage student interns in participating in challenging projects at DIT. Authors also thank the DIT student interns Tapasya Patki and Swati Verma for their initial efforts in pursuing the research and development work in rough set techniques.

REFERENCES

[1]. A.Gupta, and S. Seshasai, 24 Hour Knowledge Factory: Using Internet Technology to Leverage Spatial and Temporal Separations, ACM Transactions on Internet Technology, Vol. 7, Issue 3, August 2007.

[2]. Francis E. H. Tay and Lixiang Shen (2002), Economic and Financial Prediction using Rough Sets Model, European Journal of Operational Research 141, pp.643-661.

[3]. Dick Grune, Concurrent Versions System, a method for independent cooperation, IR 113, Vrije Universiteit, Amsterdam, pp. 9, 1986.

[4]. Israel E. Chen-Jimenez, Andrew Kornecki, Janusz Zalewski, Software Safety Analysis Using Rough Sets.

[5]. Komorowski, Lech Polkowski, Andrzej Showron- Rough Sets: A Tutorial.

[6]. Lixiang Shen, Francis E. H. Tay, Liangsheng Qu and Yudi Shen (2000), Fault Diagnosis using Rough Sets Theory , Computers in Industry, vol. 43, Issue 1, 1 August 2000, pp.61-72.

[7]. Mitra, A. and Gupta, A. 2006. Agile Systems with reusable patterns of Business Knowledge a Component Based Approach. Artech House Press, M.A.

[8]. Patki A. B. and Verma S., Implementing Data Mining Software Modules Using Rough Set Techniques, National Conference on Recent Developments in Computing and its Applications, NCRDCA.09

[9]. Patki T., Kapoor A., Khurana S., Analytical Methodologies in Soft computing: Rough Sets Techniques, Training Report No. DIT/ SD(ABP)/ MSIT/05, July 2005

[10]. Pawlak Z., Rough Classifications, International Journal of Man Machine Studies, No. 20, 1984.

[11]. Pawlak. Z. Rough Sets. Int. J.Computer and Information Science 11:341-356,1982

[12]. Robert Susmaga, Reduct and Constructs in attribute reduction, Fundamenta Informaticae Volume 61, Issue 2 (November 2003) International Conference on Soft Computing and Distributed Processing (SCDP'2002) Pages: 159 - 181. Year of Publication: 2003.

[13]. Erran Carmel, Yael Dubinsky, Alberto Espinosa. Follow The Sun Software Development: New Perspectives, Conceptual Foundation, and Exploratory Field Study. Proceedings of the 42nd Hawaii International Conference on System Sciences – 2009.

[14]. Sooraj. P. and Mohapatra, P.K.J. Modeling the 24-hour software development process, Strategic Outsourcing: An International Journal, 1, 2, (2008), 122 – 141.

[15]. J. J. Treinen, S. L. Miller-Frost. Following the sun: Case studies in global software development. IBM Systems Journal. Volume 45, Issue 4, Page 773, October 2006.

[16]. Jihie Kim, Yolanda Gil, and Marc Spraragen. Principles for Interactive Acquisition and Validation of Workflows. The Journal of Experimental and Theoretical Artificial Intelligence, 2009.

[17]. Yolanda Gil. From Data to Knowledge to Discoveries: Scientific Workflows and Artificial Intelligence. Scientific Programming, Volume 16, Number 4, 2008.

Appendix-I

Form3 class form
Fields
button1 button2 button3 label1 label2 label3 label4 label5 label6 label7 label8 label9 textbox1 textbox2 textbox3 textbox4 textbox5 textbox6

textbox7
Methods(Form3)
button1_click button2_click button3_click dispose form3 form3_load initializecomponent label5_click textbox1_textc textbox1_validate

Form 6 class form
Fields
button1 components label1 radiobutton1 radiobutton2 radiobutton3

Methods (Form 6)
button1_click dispose form6 initializecomponent radiobutton1_click

Form11 class form
Fields

a d_attr datagridveiw1 hashtable2 hashtable3 hashtable4 hashtable5 hahstable6 la no_array str ua
Methods (Form11)
Calculate Calculate_accuracy Calculate_la_ua Make_target Class(+1 overload)

Form2 class form

Fields
button1 button2 components label1 label2 label3 textbox1 textbox2
Methods (Form2)
button1_click button2_click dispose form2 initializecomponent label1_click

class form
Fields
button1 components label1 label2
Methods
button1_click dispose form1 initializecomponent label1_click
Program Static class
Methods
Main

button1_click button2_click button3_click dispose form5 initializecomponent label4_click textbox_textc

Class1 class
Fields
_items button1 button2 button3 button4 combobox1 components d_attr data_grid_veiw label1 label2 label3 label4 label5 textbox1 textbox2 textbox3
Methods
button1_click button2_click button3_click combobox1_set dispose form11(+1 overload) init1 initilizecomponent listbox1_select
Form1

Form5 class form
Fields
button1 button2 button3 components label1 label2 label3 label4 label5 label6 label7 label8 label9 label10 textbox1 textbox2 textbox3 textbox4 textbox5 textbox6 textbox7 textbox8 textbox9
Methods