**IAES International Journal of Artificial Intelligence (IJ-AI)**
Vol. 1, No. 4, Dec 2012, pp. 182~192
ISSN: 2252-8938



# Performance Analysis of Granular Computing Model on the Basis of Software Engineering and Data Mining

**Rajashree Sasamal*, Rabindra Kumar Shial****
* Computer Science Department, National Institute Of Science & Technology
Palur Hills Berhampur761008 ODISHA India

**Article Info**

***Article history:***

Received July 12th, 2012
Revised Sept 20th, 2012
Accepted Nov 26th, 2012

***Keyword:***

Granular Computing
Software Engineering
Data Mining
AI

**ABSTRACT**

Granular Computing is not only a computing model for computer centered problem solving, but also a thinking model for human centered problem solving. Some authors have presented the structure of such kind models and investigated various perspectives of granular computing from different application point of views. In this paper we discuss the architecture of Granular computing models, strategies, and applications. Especially, the perspectives of granular computing in various aspects as data mining and phases of software engineering are presented, including requirement specification, system analysis and design, algorithm design, structured programming, software testing. AI is used for measuring the three perspective of Granular Computing model. Here we have discovered the patterns in sequence of events has been an area of active research in AI. However, the focus in this body of work is on discovering the rule underlying the generation of a given sequence in order to be able to predict a plausible sequence continuation ( the rule to predict what number will come next, given a sequence of numbers).



***Corresponding Author:***

Rajashree Sasamal
Computer Science Department,
National Institute Of Science & Technology
Palur Hills Berhampur761008 ODISHA India
Email:rajashree.sasmal@gmail.com

## 1. INTRODUCTION

Granular Computing (GrC), as a general computing paradigm of Problem solving, has been received much attentions recently, although its basic ideas and principles have been studied extensively in various research communities and application domain for a long time in explicit or implicit fashion.Pawlak proposed the rough set theory to deal with the inexact information by using rough set theory [10]. Hobbes[5] presented a theory of granularity as the base of knowledge representation, abstraction ,heuristic search, and reasoning in 1985.In this theory the problem world is represented as various gains and only interesting ones are abstracted to learn concepts. The conceptualization of the world can be performed at different granularities and switched between granularities.In 1992, Giunchigalia suggested the term “granular computing” to label this growing research field in 1997 [7].

## 2. PROPOSED METHODOLOGY

- Philosophy
- Methodology
- Computation

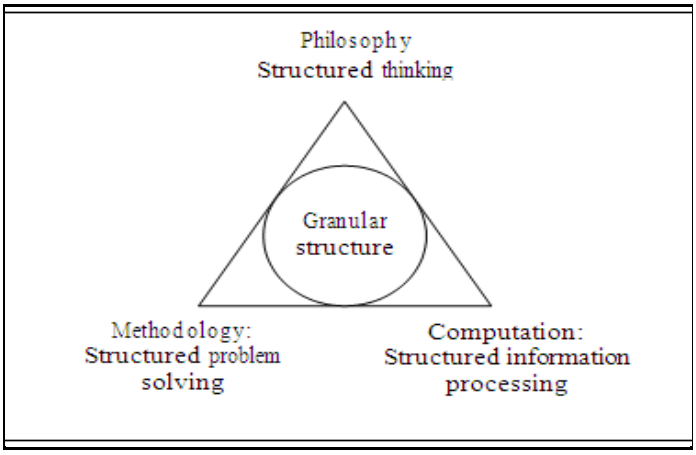



Figure 1. Perspective of Granular Computing

### 2.1 Philosophy(Structured Thinking)

As a way of structured thinking, the first task is to draw results from two complementary philosophical views about the complexity of real-world problems, "i.e," the traditional reductionist thinking and the more recent systems thinking. Here we thought to solve large data base using Granular Computing model in efficient manner.

### 2.2 Structured Problem Solving

Next task is to promote systematic approaches, effective principles, and practical heuristics and strategies that have been used effectively by humans for solving real-world problems. A central issue is the exploration of granular structures. Here we have solved a structured problem based on huge amount of data easily which we had thought.

### 2.3. Structured Information Processing

Last task is to focus on implementing knowledge-intensive systems based on granular structures. Two related basic issues are representations and processes. The objectives of these computation models are computer-centered and mainly concerned with the efficiency effectiveness, and robustness of using granules such as classes, clusters, subsets, groups and intervals in problem solving.

In this paper, we review granular computing strategies from the perspective of different phases of software engineering, especially. By investigating these strategies of granular computing in various phrases of software engineering using the above structured problem, we expect that software developers can consciously apply them in the process of software development and gain benefit of high-quality software products. In Section 2, the basic concepts and components of granular computing are introduced. In Section 3, the software development process is investigated from the perspective of granular computing, including requirement analysis, use cases, system design, program structure, algorithm design, testing, and deployment. Section 4, is We are given a large database of customer transaction of the same structured problem of a mobile company, where each transaction consists of customer_id, transaction_ time and items_brought in the transaction in a mobile company and it consist of lot management,entry of customer detail and searching of customer detail. We introduced the problem of mining sequential patterns over such databases. We present three algorithms to solve this problem and empirically evaluate their performance using synthetic data. Two of the proposed algorithms AprioriSome and AprioriAll have comparable performance albeit. AprioriSome performs a little better when minimum number of customers that must support a sequential

pattern is low scale up experiments show that both AprioriSome and AprioriAll scale up linearly with the number of customers per transaction. They also have excellent scale up properties with respect to the number of transactions per customer and the number of items in a transaction. Section 5, is the conclusion.

## 3. EMPHASIS OF SOFTWARE W ENGINEERING IN GRANULAR COMPUTING

Granular computing provides the infrastructure for data, information and knowledge engineering as well as uncertainty management. In software engineering, the strategies of granular computing are also broadly used in all phases. Granulate-and-conquer is a softer version of classical divide-and-conquer strategy. A very common technique used in the classical "non-partitioning" recursive call is dynamic programming. Functional decomposition is to partition user requirement into granules (functions). In object-oriented programming, granules are classes, intrarelationships are the interactions between components of classes such as instance fields, methods and constructors in the Java language; and the intrarelationships are class inheritance, aggregation, association, delegation, dependency, etc. Modern software engineering process based on object-oriented technologies, including requirement analysis, system analysis, design, and implementation.

### 3.1. User Requirements

In Object oriented technology user requirements analysis involves two main aspects: identifying business actors and use cases.

### 3.2. Granules

Granules are basically used to represent the elementary units of a complex system that is considered, and collectively provide a representation of the unit with respect to a particular level of granularity. Each granule may reflect a specific aspect of the problem or form a portion of the system domain. In the real-world problem solving, we may need to consider the levels of detail of a system. On the different levels, granules may represent different units. Using object-oriented software as an example, one can see that the software system is considered as a whole, its ingredients are a collection of classes.

### 3.3. Granulation criteria

It determines whether a granule should be granulated into smaller granules for top-down construction, or whether different elements/granules should be put together to form a larger granule for bottom-up construction. Partition vs. covering should be distinguished.

### 3.4. Granulation

A Granulation is to granulate complex problem into granules, including the construction, representation, and interpretation of granules. It concerns the procedures for constructing granules.

### 3.5. Granular Relationship

Relationships between granules can also be represented as binary relations and interpreted as dependency, inheritance, etc.

### 3.6. Granular computation

Identifying the process of use cases, including input data, output data and business logic, as well as the communication with actors. Consider the following Example of user requirement in Telecommunication System.

To check misuse of mobile phones by terrorists and other anti-national elements, the Centre has decided to rein in mobile operators not taking the verification process of SIM (subscriber identity module) cards seriously and indulging in unhealthy competition to boost sales. Pointing to the unhealthy competition among operators to grab the maximum share of subscribers in the world's fastest growing mobile market, the official said the sale of SIM cards needed to be streamlined. These chips were now readily available even at tea stalls, paan shops and grocery stores.

This project is aimed at developing an online CAF Digitization System that is of importance to any Telecommunication System. The CAF Digitization system is an Internet based application that can be accessed throughout the organization or a specified group. This system can be used to automate the workflow of Customer applications and their approval. It represents module 1.Login Form having UserId and Password allows the authenticated users to execute the furthere Processing. Module 2 having lot management, contact information, search CAF. The lot management refers to Module 3, which contains lot details having lot no,

lot Status, opening date, location, size and description. Module 4, represent the contact details having circle owner, client contact, SSA owner, SSA contact and area of different lot. Module 5 refers to customer details consisting of name, mobile number, date, pocafid, agent, connection type. They are mentioned in figure 2.

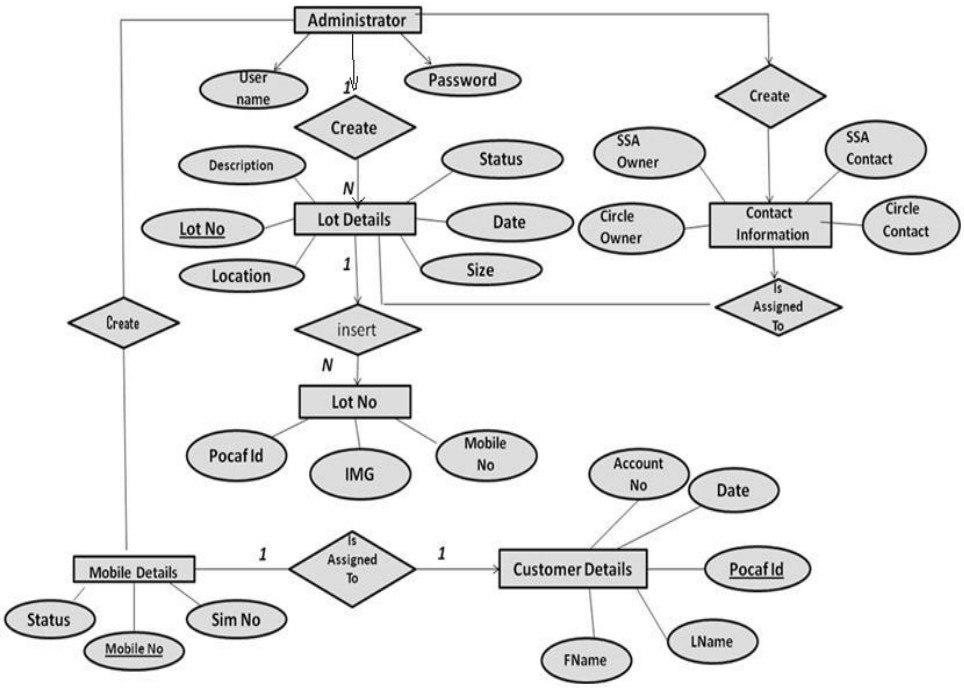



Figure2.Flow chart diagram

## 2. SYSTEM ANALYSIS

System analysis is defined as the process of gathering and interpreting facts, diagnosing problems and using the information to recommend the improvement to the system. The main goal of this phase is to find out candidate classes that describe the objects that might be relevant to the system, relationships between the classes, as well as attributes for the classes..

### 4.1. Granulation method

In system analysis it is used to identify classes. Candidate classes are often indicated by nouns in the use cases except those that represent the system, actors, boundaries, and trivial types. Two basic rules should be followed to identify classes: high cohesion inside classes and low coupling between classes. Class identification might be top-down and/or bottom-up.

### 4.2. Granular Relationship

System analysis correspond to class relationships, which can include the following types: inheritance (is-a relationship).

From the above experiment we have designed the following classes as granular methods, they are as follows:

*AddFormBean, AddMobileNo, CAFUploadAction, CAFUploadForm, DBListener, FileUpload, FileUploadAction,GroupBy,LoginAction,LoginForm,LogoutAction,LotManage,LotManage,.NewLotAction, SeachCAFAction, SearchCAF.*
*Granulation method in system analysis is to identify*
*Set (Interface)*

*HashSet (Class)*
*SortedSet (Interface)*
*TreeSet (Class)*
*…..……..*

## 3. SYSTEM DESIGN

System design involves multiple steps, including technologies can be chosen to granulate the software, thus, object-oriented system design partitions the system components into layers as below:

*User Interface / Swing*
*Control*
*Network / RMI*
*Server*
*Business*
*Persistence*
*Database / JDBC*

## 6. SYSTEM IMPLEMENTATION

Granular computing strategies have also been broadly used in software system implementation. The typical object-oriented program structure is actually a granular structure. Consider a Java-based software system, which can be characterized as follows:

*Software system*
*Packages*
*Classes*
*Instance fields*
*Constructors*
*Methods*
*Interfaces*
*Method prototypes*

## 7. SEQUENTIAL MINING PATTERN OF S/W ENGINEERING

We are given a database D of customer transactions in the above example. Each transaction consists of the following fields: customer-id, transaction-time, and the items purchased in the transaction. No customer has more than one transaction with the same transaction-time. We do not consider quantities of items bought in a transaction: each item is a binary variable representing whether an item was bought or not. An itemset is a non-empty set of items. A sequence is an ordered list of itemsets. Without loss of generality, we assume that the set of items is mapped to a set of contiguous integers. We denote an itemset i by i (i1 i2 ...im), where ij is an item. We denote a sequence s by h (s1s2...Sn), where sj is an itemset.A customer supports a sequence s if s is contained in the customer-sequence for this customer. The support for a sequence is defined as the fraction of total customers who support this sequence.

## 8. ALGORITHM

We split the problem of mining sequential patterns into the following phases:

### 8.1 Sort Phase

The database (D) is sorted, with customer-id as the major key and transaction-time as the minor key. This step implicitly converts the original transaction database into a database of customer sequences. Fig. 2 shows the sorted database for the example database.

### 8.2 Litemset Phase.

In this phase wend the set of all Litemset L. We are also simultaneously finding the set of all large 1-sequences, since the algorithm is {<l>| € L}.

### 8.3 Transformation Phase

As we will see in Section 3, we need to repeatedly determine which of a given set of large sequences are contained in a customer sequence. To make this test fast, we transform each customer sequence into an alternative representation.

### 8.4 Sequence Phase

Use the set of Litemset to find the desired sequences.

Table 1. Original database

| Database | | |
|---|---|---|
| Customer Id | Transaction Time | Items Bought |
| 2 | June 10 '93 | 10, 20 |
| 5 | June 12 '93 | 90 |
| 2 | June 15 '93 | 30 |
| 2 | June 20 '93 | 40,60,70 |
| 4 | June 25 '93 | 30 |
| 3 | June 25 '93 | 30,50,70 |
| 1 | June 25 '93 | 30 |
| 1 | June 30 '93 | 90 |
| 4 | June 30 '93 | 40,70 |
| 4 | June 25 '93 | 90 |

Table 2. Database sorted by customer id

| Database | | |
|---|---|---|
| Customer Id | Transaction Time | Items Bought |
| 1 | June 25 '93 | 30 |
| 1 | June 30 '93 | 90 |
| 2 | June 10 '93 | 10,20 |
| 2 | June 15 '93 | 30 |
| 2 | June 20 '93 | 40,60,70 |
| 3 | June 25 '93 | 30,50,70 |
| 4 | June 25 '93 | 30 |
| 4 | June 30 '93 | 40,70 |
| 4 | June 25 '93 | 90 |
| 5 | June 12 '93 | 90 |

Table 3. Transformed Database Phase

| Customer Id | Original Customer Sequence | Transformed Customer Sequence | After Mapping |
|---|---|---|---|
| 1 | ((30) (90)) | ({(30)}{(90)}) | ({1}{5}) |
| 2 | ((10 20)(30)(40 60 70)) | ({30}{(40),(70),(40,70)}) | ({1}{2,3,4} |
| 3 | ((30 50 70)) | ({(30),(70)}) | ({1,3}) |
| 4 | ((30)(40 70)(90)) | ({(30)}{(40),(70),(40 70)}{(90)}) | ({1}{2,3,4}) |
| 5 | ((90)) | ({(90)}) | ({5}) |

Given a database D of customer transactions, the problem of mining sequential patterns is to find the maximal sequences among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a sequential pattern.

We call a sequence satisfying the minimum support constraint a large sequence.

*Example*: Consider the database shown in Fig. 1. Fig. 2 shows this database sorted on customer-id and transaction-time. Fig. 3 shows this database expressed as a set of customer sequences.

With minimum support set to 25%, i.e., a minimum support of 2 customers, two sequences: ((30) (90)) and ((30) (40 70)) are maximal among those satisfying the support constraint, and are the desired sequential patterns. The sequential pattern ((30) (90)) is supported by customers 1 and 4.Customer 4 buys items (40 70) in between items 30 and 90, but supports the pattern ((30) (90)), since we are looking for patterns that are not necessarily contiguous. The sequential pattern (30 (40 70)) is supported by customers 2

and 4. Customer 2 buys 60 along with 40 and 70, but supports this pattern since (40 70) is a subset of (40 60 70). An example of a sequence that does not have minimum support is the sequence ((10 20) (30)), which is only supported by customer 2. The sequences ((30)),((40)), ((70)), ((90)), ((30) (40)), (30) (70) )and ( (40 70)) though having minimum support, are not in the answer because they are not maximal. (For some applications, the user may want to look at all sequences, or look at subsequences whose support is much higher than the support of the corresponding maximal sequence.

Table4. Customer-Sequence Version of the Database

| Database | |
|---|---|
| Customer Id | Items Bought |
| 1 | ((30) (90)) |
| 2 | ((10 20)) (30) (40 60 70)) |
| 3 | ((30 50 70)) |
| 4 | ((30)(40 70) (90)) |
| 5 | ((90)) |

Table5.The answer set

| Sequential Patterns |
|---|
| with support > 25% |
| ( (30) (90) ) |
| ((30) (40 70) ) |

### 8.5 Algorithm AprioriSome

By this algorithm we only count sequences of certain length “e.g.” we might count the sequence of length 1,2,4 and 6 in forward pass and count the sequence of 3 and 5 in the backward pass. The function next takes as parameter the length of sequences counted in the last pass and returns the length of sequence to be counted in the next pass. One extreme is next(k)=k+1(k is the length for which candidates were counted last. When all non maximal sequences are counted ,but no extensions of small candidate sequences are counted. In this case, AprioriSome degenerates into AprioriAll.

(1 2 3)
(1 2 4)
(1 3 4)
(1 3 5)
(2 3 4)
(3 4 5)

Figure 7: Candidate 3-sequences

(1 3 5)
(3 4 5)

Figure 8: Candidate 3-sequences after deleting subsequences of large 4-sequences

### 8.6 AprioriAll Algorithm

This algorithm is given in Figure 7.In each pass We use the large database from the previous pass to generate the candidate sequence and then measure their support by making a pass over the database. At the end of the pass, then mine the large sequence. In the first pass, the output of Litemset phase is used to initialize the set of large I-sequence.

| Sequence | Support |
|---|---|
| (1 2 3) | 2 |
| (1 2 4) | 2 |
| (1 3 4) | 3 |
| (1 3 5) | 2 |

(2 3 4) 2

Figure 9: Large 3-Sequences

(1 2 3 4)
(1 2 4 3)
(1 3 4 5)
(1 3 5 4)

Figure 10: Candidate 4-Sequences (after join)

(1 2 3 4)

Figure 11: Candidate 4-Sequences (after pruning)

### 8.7 Algorithm DyanamicSome

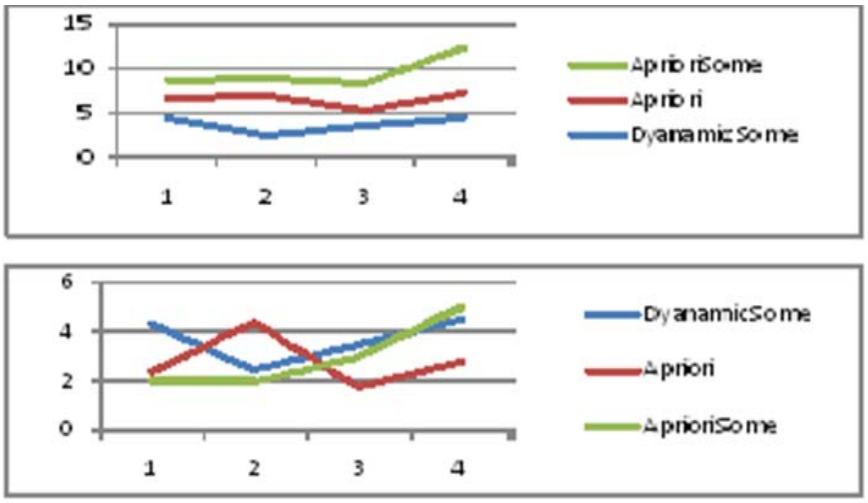
The DyanamicSome algorithm is like AprioriSome, we skip counting candidate sequence of certain lengths in the forward phase. The candidate sequence that are counted is determined by the variable step. In the Initialization phase ,all candidate sequences of length unto and including steps are counted. Then in forward phase, all sequences whose length are multiples of step are counted. Then in the forward phase ,all sequences whose length are multiple of step are counted. Thus with the step set to 3,we will count sequence of lengths 1,2, and 3 in the initialization phase , and 6,9,12 ,…. In forward phase. We really wanted to count only sequences of lengths 3,6,9,12,… We can generate the sequence of length 6 by joining the sequence of length 3. We can generate the sequence of length 9 by joining the sequence of length 6 with the sequence of length 3.etc.So we are using the forward phase.

## 9. PERFORMANCE

To assess the relative performance of the algorithms and study their scale-up properties, we performed several experiments on s/w products.

### 9.1. Scale Up

We investigated the scale-up as we increased the total number of items in a customer sequence.This increase was achieved in two different ways: i) by increasing the average number of transactions per customer, keeping the average number of items per transaction the same; and ii) by increasing the average number of items per transaction, keeping the average number transactions per customer the same. The aim of this experiment was to see how our data structures scaled with the customer-sequence size, independent of other factors like the database size and the number of large sequences.




.

Figure 12.Performance Scale

We kept the size of the database roughly constant by keeping the product of the average customer-sequence size and the numberof customers constant. Number of Customer Customer in Relative Time

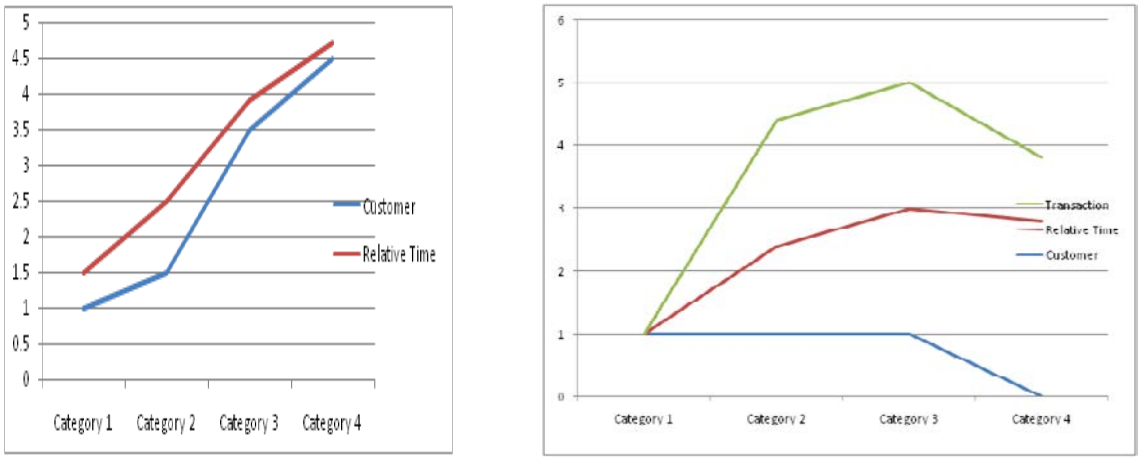



Figure 13.Scale Up Compare

**9.2.Top Down Progressive Algorithm**

The top-down progressive computing model is used in the process of writing a computer program [5]. The program construction consists of a sequence of refinement steps. In each step, a given task is broken up into a number of subtasks.

*1.begin*
*2. k←0;*
*3.set $G_0$ and $S_0$;*
*4.repeat*
*5. k←k+1;*
*6.$G_k$=granulation refinement($G_{k-1}$);*
*7.$S_k$=solution refinement($S_{k-1}$,$G_k$);*
*8.$S_k$=fitness($P_k$);*
*9.until $S_k$ satisfies a certain condition;*
*10.end;*

G0 and S0 are some initial values for granulation (G) and solution(S).In the very first step, G1 = granulation refinement(G0) produces the coarsest granulation G1 in the granulation sequence G1,G2,G3,....and S1 = solution refinement(S0,G1) is the very first solution in the solution sequenceS1,S2,..... .
This algorithm is used for handling the sequence pattern of the above problem which is mentioned with the help of Granular Computing model.

**9.3. Generation Of Synthetic Data**

To evaluate the performance of the algorithms over a large range of data characteristics, we generated synthetic customer transactions. These transactions mimic the transactions in the retailing environment. In our model of "real" world, people buy sequences of sets of items. Each such sequence of itemsets is potentially a maximal large sequence. A customer-sequence may contain more than one such sequence. For example, a customer might place an order for a dress and jacket when ordering sheets and pillow cases, where the dress and jacket together form part of another sequence. Customer-sequence sizes are typically clustered around a mean and a few customers may have many transactions . This model is a generalization of the synthetic data generation model.

| | |
|---|---|
| 1 | Number of customers(=size of Database) |
| 2 | Average number of transactions per customer |
| 3 | Average number of items per Transaction |
| 4 | Average length of maximal potentially large Sequence |
| 5 | Average size of itemsets in maximal potentially large sequences |
| 6 | Number of maximal potentially large Sequences |
| 7 | Number of maximal potentially large itemsets |
| 8 | Number of items |

Figure 12:Parameters

## 10. CONCLUSION

It evidences that granular computing as a human thinking model plays an essential role in software engineering, although software analyst, designer and developer have been unconsciously applying this strategy in their daily work.In this article, we focus on a high-level examination of granular computing as a new field of study in its own right to illustrate them, we provide an extensive list of references. One can easily find detailed discussions about the specific applications of these ideas and principles from the references. Software engineering is both a thinking process and a problem solving process. We examined different phases of object-oriented software development, including requirement analysis, system analysis, system design, and system implementation from the perspectives of granular computing. We introduced a new problem of mining sequential patterns from a database of customer sales transactions and presented three algorithms for solving this problem. Two of the algorithms, Apriori-Some and AprioriAll, have comparable performance, although AprioriSome performs a little better for the lower values of the minimum number of customers that must support a sequential pattern. Scale-up experiments show that both AprioriSome and AprioriAll scale linearly with the number of customer transactions. We conclude that consciously using granular computing methodology in software engineering and data mining will improve the quality of software products. Granular computing may have a great impact on the design and implementation of intelligent information systems, and on real world problem solving.

## REFERENCES

[1] A. Bargiela, W. Pedrycz, Granular mappings, IEEE Transactions on Systems, Man, and Cybernetics, Part A 35 (2005) 292-297.
[2] Y.H. Chen, Y.Y. Yao, Multiview intelligent data analysis based on granular computing, Proc. 2006 IEEE Int. Conf. on Granular Computing, 2006, pp. 281-286.
[3] ] C.M. Conway, M.H. Christiansen, Sequential learning in non-human primates, Trends in Cognitive Sciences 12 (2001) 539-546.
[4] T.Y. Lin, Y.Y. Yao, L.A. Zadeh (Eds.) Data Mining, Rough Sets and Granular Computing, Physica-Verlag, Heidelberg, 2002.
[5] Inuiguchi, M., Hirano, S. and Tsumoto, S. (Eds.) Rough Set Theory and Granular Computing, Springer, Berlin, 2003. .
[6] Cendrowska, J. PRISM: an algorithm for inducing modular rules, International Journal of Man-Machine Studies, 27, 349-370, 1987.
[7] C.M. Conway, M.H. Christiansen, Sequential learning in non-human primates, Trends in Cognitive Sciences 12 (2001) 539-546.

**BIBLIOGRAPHY OF AUTHORS**

Miss. Rajashree Sasamal is a Mtech Student of National Institute Of Science & Technology,Berhampur Orissa.Area of Interest is Artificial Intelligence,Software Engineering,DataMining.She has published conference papers,and attended seminars and workshops.

Mr. Rabindra Kumar Shial is Assistant Professor of National Institute Of Science & Technology,Berhampur Orissa.Area of Interest is Networking,Software Engineering,Distributed Database.He has published many journal papers and conference papers.