# Study on Agile Process Methodology and Emergence of Unsupervised Learning to Identify Patterns from Object Oriented System

Mulugu Narendhar[1] and K. Anuradha[2]

[1] Department of CSE, BSIT,
Hyderabad, A.P, India
mnarender53@gmail.com
[2] Department of CSE, GREIT,
Hyderabad, A.P, India
kodalianuradha@yahoo.com

**Abstract.** Data mining is knowledge extraction for secure software engineering, improves the quality and productivity, poses several challenges, requiring various algorithms to effectively mine text, graph from such database. Fact that building models in the context of the framework one of the task data miners, almost important though all other tasks associated with data mining. Data mining techniques are tackling the right business problem, must understand the data this is available and turn noisy data into data from which we can build robust models. It is important to be aware data mining is really what we might call an agile model. The concept of agility comes from the agile software engineering principles includes increment development of the business requirements and need to check whether the requirement satisfies with the client inputs our testing and rebuilding models improves the performance. For software engineering code execution, code changes list of bugs and requirement engineering our system uses mining techniques to explore valuable data patterns in order to meet better projects inputs and higher quality software systems that delivered on time. Our research uses frequent mining, pattern matching and machine learning applied to agile software architecture model in gathering and also extracting security requirements best effort business rules for novel research.

**Keywords:** Agile Model, Data Mining, Software Engineering, Architecture & Design Pattern.

## 1 Introduction

Features of data mining such as database tasks is the fact that knowledge acquired from mining, classification association rules and retrieval of relevant information produce the most accurate results. Similar in software engineering we can see two distinct ways data mining information retrieval and supervised & unsupervised methods are applied. Software engineering is artifact of software development that are document hierarchies, code repositories, bug report data bases for the purpose of learning new interesting information about the underlying patterns.

Application data mining use the term certain related activities and techniques from machine learning, natural language processing and information retrieval in different processes in software lifecycle with the automating and improving performance on the tasks involved. Mining techniques are typically applicable to intensive tasks and are capable of speeding up the performance the order of magnitude. Some many benefits of applying data mining to specific tasks a human analyst must always access correct results of the automated methods. As the goal data mining methods is improvement of the process observe that only result is seen by others is generated by the analyst. Today's mainstream software processes do not effectively address two major issues on cost and time there is emerging producing new processes known as "Agile Software Process" [1] . The new processes focus more on people interactions development of code than on documentation, this work presents how mining techniques impact on agile software process to extract text mining, graph mining.

Frequent patterns are item sets that subsequently appear in data set with frequency for example a set of items such as brad and jam that appear together in a transaction data set is a frequent item set. Frequent pattern mining was introduction by Agrawal for market basket analysis in the form of association rule mining for instance customer buying bread how likely they are going buy jam, sugar on the same trip to the supermarket. Frequent Patterns: Many commercial applications need pattern that are more complicated than a frequent item sets such patterns go beyond sets and sequences toward trees, graphs. Among the various kinds of graph patterns frequent substructure are the basic patterns, we have two types of frequent pattern mining one is [2] Apriority-based approach and other is pattern-growth.

Apriori is frequent substructure mining algorithm suppose we are searching for small size of graph and will proceed in bottom up manner here at each iteration the size redirects frequent substructures is increased by one. These are first generated by joining two similar but slightly different sub graphs that were discovered already.Apriori frequent substructure mining algorithm uses a vertex based candidate generation method that increases the substructure size by one vertex at each iteration. Two size k frequent graphs are joined only when two graphs have the same size (k-1) sub graphs.

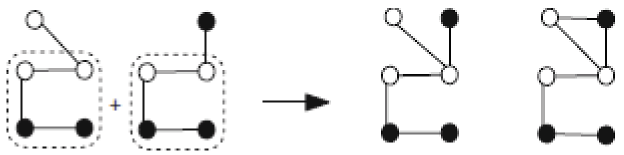


**Fig. 1.** Two sub graphs joined by two chains

The graph size means the number of vertices in a graph and new candidate includes the common size (k-1) sub graph and the additional vertices from the two size k patterns. FSG is an edge-based candidate generation strategy that increases the sub structure size by one edge in each iteration. Two size patterns are merged if and only if they share the same sub graph with k-1 edges.

## 2 Software Architecture

Architecture is a process that defines the solutions final meets the goals of end product technical and functionally while optimizing common quality attributes such as performance and security. Architecture encompasses the set of tasks and decisions about the system including the selection of structural elements and their interfaces by which the system is composed behavior as specified in collaboration among those elements composition of these structural and behavioral elements into larger subsystems and an architectural style [3]. It also involves functionality usability resilience performance reuse the tradeoffs and aesthetic concerns.

Like common problems, key scenarios, complex structure, software must be built on a solid foundation, modern tools and platform help to simple task of building applications but they do not replace the need to design application carefully based on specific scenario. Systems should be designed with consideration for the user, system and business goals. For each of these should outline key scenario and identify important quality attributes and satisfaction and dissatisfaction where possible.

Software architecture use viewpoints such as functional, information, deployment to guide the process of capturing the architecture as a set of views with the development of each view being by the use of a specific viewpoint. Architecture focuses on how the major elements and components within an application are used or interact with major elements and components within the application. The selection of data structures and algorithms implementation details of components are design concern.

Application architecture to build a bridge between requirements and technical requirements by understanding use cases and then finding way to implement those use cases in the system software. The architecture goal is to identify the requirements that affect the structure of the application which reduces the business risks associated with building technical solutions.

To develop architecture we need to follow below steps:

Expose the structure of the system but hide the implementation details.
Realize all of the use cases and scenarios.
Try to address the requirements of various stakeholders.
Handle the functional and quality requirements.

Software architecture is classifies as shown here. Artifact driven defines the relation between subsystems, groups the related artifact in subsystems these are the architecture components. Use case driven identifies fundamental classes from use cases and groups these classes in packages are the architecture components. Pattern starts with requirement specification which select appropriate patterns from a pattern base are architecture components. Pattern is a generic and reusable design solution for recurring problems in a context.

Software pattern is design rationale constraints facilitate reuse and sharing of models and design artifacts to fix a particular problem. Developers do not invent software patterns they discover patterns from building systems. Each pattern states problem to which the pattern is applicable constraints of using the pattern in practice and often some other information about the pattern. Developers need to understand software pattern does not require knowledge of programming skills only require a extra effort in order to recurring solutions to specific problem.

## 3 Problem Definition

In this research, our descriptive approach primarily used to generate structural and behavioral perspective explanation of the study. The project may include the mining techniques in agile method software architecture design. In previous work we identify the problem in software engineering that reliability of code, complexity of structure, more effort and cost we analyze the problem mining techniques are efficient to solution

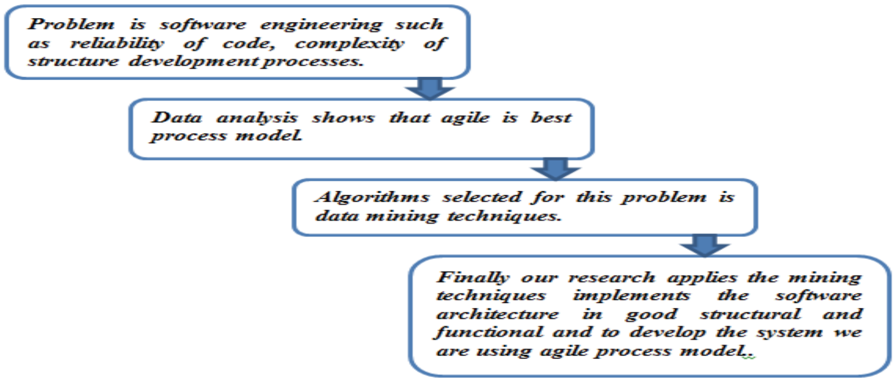

Problem is software engineering such as reliability of code, complexity of structure development processes.
Data analysis shows that agile is best process model.
Algorithms selected for this problem is data mining techniques.
Finally our research applies the mining techniques implements the software architecture in good structural and functional and to develop the system we are using agile process model.

**Fig. 2.** Flow diagrams for proposed Work

Software architecture & design pattern is common problem in design phase of software engineering. Discovery of pattern engineering in design or coding phase represents a program understanding process, identifying the patterns using unsupervised learning would be useful to find objects in architecture & design improves quality attributes of system.

## 4 Agile Process Technique

Computer programmers, computer scientists, software engineers, and management scientists have been trying to solve computer programming problems such as productivity, quality, reliability, customer satisfaction, cost effectiveness, and time-to-market for more than five decades. The concept has been introduced in the 1960's and now, the transition to agile processes is a growing trend that will have lasting effects on the industry and the people involved.

## 4.1 Managing Constraints, Scope and Quality

Agile Methodology manages changes and constraints. The way the project is managed will allow the customer to minimize cost of change while taking into account constraints.

In Agile environment we can identify two types of constraints:

*External constraints:* The project manager is to handle the external constraints that are imposed by the project environment and the customer through contractual terms. The project manager is still guarantying those constraints imposed on time, budget quality and scope.

*Internal constraints:* The project manager rely on the team to handle the internal constraints which are often more technical. "Agile projects engage people with profound knowledge of the system; team is typically diverse of generalizing specialist.

In Agile Methodology, Typically, when the project manager is defining the project scope with the customer and other stakeholders, the project charter, very often, consists of several white paper boards with color-coded markings and post-its. In traditional project management, before proceeding to the initiation phase, the scope has to be well defined and understood by the project manager, whereas agile methodology is demanding more than just understanding the scope.

## 4.2 Agile Estimation Algorithm

Agile Software Methodology estimation process followed by the cost, size and duration CSD algorithm for the same Agile software process. Agile estimation process is classified into two phases: Early Estimation (EE) and Iterative Estimation (IE) as shown in fig.3. An estimator can update the estimates whenever the uncertainty in requirements reduces. The purpose of EE is to identify the scope of the project by envisaging the upfront with just enough requirements. This provides just enough understanding to put together an initial budget and schedule without paying the price of excessive documentation. EE also provides the platform for fix budget agile project and generates a satisfaction amongst the stakeholders by providing range of estimate to reflect temporal risk.IE is an iterative activity that starts at the beginning of iteration to incorporate new requirements/ changes. Agile estimation process considers only clear specified requirements and other factors that affect the estimation to derive CSD of the project as shown in Fig. 3. Here, estimation process starts after the prioritization of requirements and EE is just to understand the CSD involved in project. After finalization of project, iteration planning starts and continues till all the requirements/ changes required by customers are exhausted. EE and IE use story points for estimating CSD. Existing AEMs use expert opinion and historical data for evaluating story points. In this section, we propose *CAEA* to evaluate story points after the inclusion of various CSD affecting factors in AEM. Computation of variables in various projects to establish the fact that inclusion of vital factors in agile estimation generates realistic and more precise CSD estimation. An algorithm *CAEA* is based on above constructive agile estimation and computes the story points for CSD estimation. It incorporates the vital factors such as project domain, performance, configuration etc. as discussed in Section 3. We have graded the intensity of these factors on the scale of low, medium and high based upon the complexity of the project. It is preferred to map the intensity levels with mathematical series such as square series (1, 4, 9) or Fibonacci series (2, 3, 5). Square series has been proved to be the most preferred series in agile estimation since it provides realistic level of accuracy for complex and ill-defined project.

Formal description of proposed AEA is described as follows:

***Algorithm: AEA***

This algorithm computes the of story points on the basis of the input as the grades of vital factors of a project

*Step 1:* Vital factors of project are identified on the grade of low, medium and high

Using square series or Fibonacci series.

*Step 2:* Compute sum of all grades of various factors for a project denoted as Unadjusted Value (UV).

*Step 3:* Decompose the project in small tasks or stories.

*Step 4:* Assign Story Point (SP) to each story based upon the size.

*Step 5:* Compute New Story Point (NSP) by using equation (1).

*Step 6:* Compute SOP by using equation (2).

*Step 7:* Compute DOP through equation (3).

$$\text{NSP} = \text{SP} + 0.1 * \text{UV} \tag{1}$$

$$\text{Size of Project(SOP)} = \sum_{i=1..n} \text{NSP}_i \tag{2}$$

$$\text{Duration of project (DOP)} = \text{SOP} / \text{Velocity} \tag{3}$$

Where SP story point of a story, UV is unadjusted value,
NSPi is NSP of ith the story and n is total number of stories of project,
SOP is size of project,
Velocity is number of NSP developed in iteration.

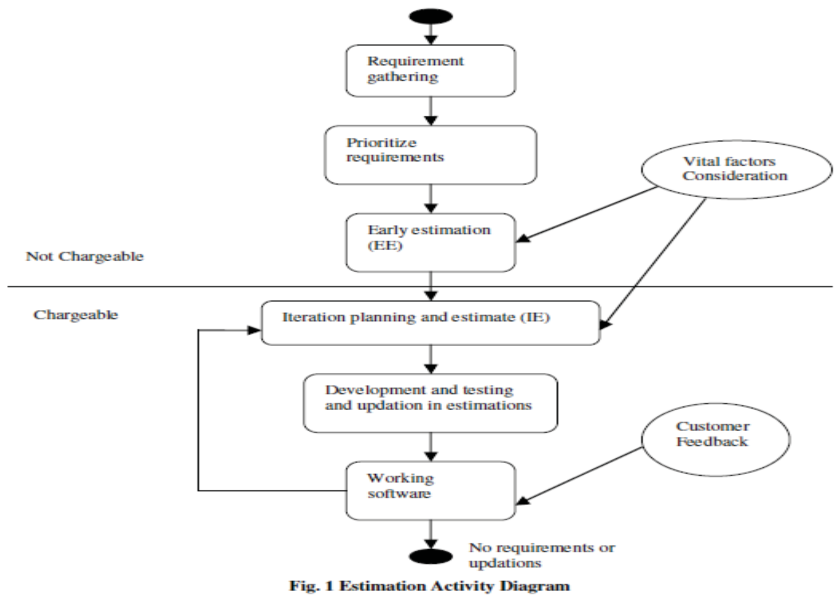

Requirement gathering
Prioritize requirements
Vital factors Consideration
Early estimation (EE)
Not Chargeable
Chargeable
Iteration planning and estimate (IE)
Development and testing and updation in estimations
Customer Feedback
Working software
No requirements or updations
Fig. 1 Estimation Activity Diagram

**Fig. 3.** Flow diagram of Agile Process

### 4.3 Agile Software Process Methodologies

XP and Scrum iterative approach process models are widely used in the Agile Philosophy

#### 4.3.1 Extreme Programming

Extreme programming(XP), concentrates on the development rather than managerial aspects of software projects. XP was designed so that organizations would be free to adopt all or part of the methodology. XP projects start with a release planning, followed by several iterations, each of which concludes with user acceptance testing. User representative part of the XP team can add detail requirements as the software is being built. This allows requirements to evolve as both users and developers define what the product will look like. In release plan, team breaks up the development tasks into iterations. Release plan defines each iteration plan, which drives the development for that iteration. At the end of iteration, conduct the acceptance tests against the user stories. If they find bugs, fixing the bugs becomes a step in the iteration. If users decide that enough user stories have been delivered, the team can choose to terminate the project before all of the originally planned user stories have been implemented.

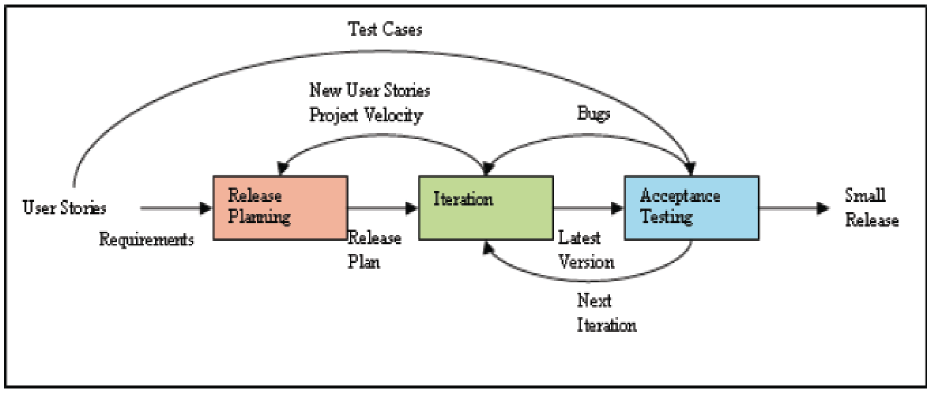


**Fig. 4.** Extreme Programming Process Model shows a simplified version of XP. Full XP includes many steps in release planning, iteration, and acceptance.

***Integrate Often:*** Development teams must integrate changes into the development baseline at least once a day.

***Project Velocity:*** Velocity is a measure of how much work is getting done on the project. This important metric drives release planning and schedule updates.

***Pair Programming:*** All code for a production release is created by two people working together at a single computer. XP proposes that two coders working together will satisfy user stories at the same rate as two coders working alone, but with much higher quality.

***User Story:*** A user story describes problems to be solved by the system being built. These stories must be written by the user and should be about three sentences long. User stories do not describe a solution, use technical language, or contain traditional requirements-speak, such as "shall" statements. Instead, a sample user story might go like this: Search for customers. The user tells the application to search for customers. The application asks the user to specify which customers.

### 4.3.2 Scrum

Scrum is the term for a huddled mass of players engaged with each other to get a job done. In software development, the job is to put out a release. Scrum for software development came out of the rapid prototyping community because prototypes wanted a methodology that would support an environment in which the requirements were not only incomplete at the start, but also could change rapidly during development.

Scrum project is a backlog of work to be done. This backlog is populated during the planning phase of a release and defines the scope of the release. After the team completes the project scope and high-level designs, it divides the development process into a series of short iterations called sprints. Each sprint aims to implement a fixed number of backlog items. Before each sprint, the team members identify the backlog items for the sprint. At the end of a sprint, the team reviews the sprint to articulate lessons learned and check progress. During a sprint, the team has a daily meeting called a scrum. Each team member describes the work to be done that day, progress from the day before, and any blocks that must be cleared. To keep the meetings short, the scrum is supposed to be conducted with everyone in the same room—standing up for the whole meeting. When enough of the backlog has been implemented so that the end users believe the release is worth putting into production, management closes development. The team then performs integration testing, training, and documentation as necessary for product release.

Scrum development process concentrates on managing sprints. Before each sprint begins, the team plans the sprint, identifying the backlog items and assigning teams to these items. Teams develop, wrap, review, and adjust each of the backlog items. During development, the team determines the changes necessary to implement a backlog item. The team then writes the code, tests it, and documents the changes. During wrap, the team creates the executable necessary to demonstrate the changes. In review, the team demonstrates the new features, adds new backlog items, and assesses risk. Finally, the team consolidates data from the review to update the changes as necessary.

## 5 Clustering

Cluster based on the information found in the data describing the instances, combining similar objects in one cluster and dissimilar objects in other cluster. In many applications clusters are not well different from another most cluster seeks a result a crisp classification of the data into non-overlapping groups.

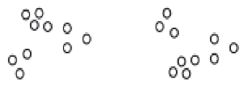


**Fig. 5.** (a) Instance

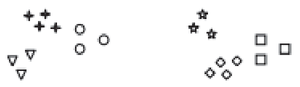


**Fig. 5b.** Many clusters

Figure a consider instances of in different ways that they can be divided into clusters. If we allow clusters to be nested then the interpretation of structure of these points is that each of which has many sub clusters. Cluster is a classification of instances from the data does not allow previously assigned class labels except perhaps for verification. Cluster is different from pattern recognition and decision analysis seeks to find rules for in a given set of pre-defined objects.

Hierarchical construct the clusters by recursively partitioning the instance in top-down or bottom-up approach classified.

Agglomerative clustering, each object initially represents a cluster of its own clusters then clusters are successively merged until the desired cluster structure is obtained. Divisive cluster is all objects initially belong to one cluster then cluster divided into sub clusters which are successively divided into their own sub-clusters.

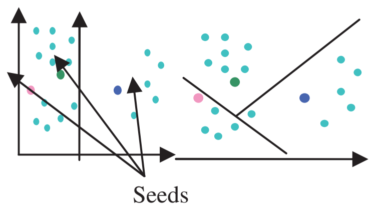
Hierarchical cluster results in dendrogram representing the nested grouping of objects and similarity levels at which groupings change.

Complete link clustering also called the diameter maximum method or the further neighbor methods that consider the distance between two clusters to be equal to the longest distance from any member of one cluster to any member of the other cluster.

Average link clustering method that consider distance between two clusters to be equal to the average distance from any number of clusters to any number of the cluster.

Partitioning methods relocates instances by moving them from one cluster to another starting from an initial.

For the partition can be of K-means [15] & K-mediod. The purpose solution is based on K-means (Unsupervised) clustering combine with Id3 Decision Tree type of Classification (Supervised) under mentioned section describes in details of K-means & Decision Tree. K-means [3], [14] is a centroid based technique each cluster is represented by the center of gravity of the cluster so that the intra cluster similarity is high and inters cluster similarity is low. This technique is scalable and efficient in processing large data sets because the computational complexity is O(nkt) where n-total number of objects, k is number of clusters, t is number of iterations and k<<n and t<<n.



(a) Un clustered Data Instances (b) Resultant Clusters

**Fig. 6.** Formation of clusters using seed points

*C. K-mean algorithm*

1. Select k centroids arbitrarily (called as seed as shown in the figure) for each cluster $C_i$,iε [1, k]
2. Assign each data point to the cluster whose centroid is closest to the data point.
3. Calculate the centroid $C_i$of cluster $C_{i,}$iε [1, k] In short
4. Repeat steps 2 and 3 until no points change between clusters. A major disadvantage of K means is that one must specify the clusters in advance and further the algorithm is very sensitive of noise, mixed pixels and outliers. The definition of means limit the application to only numerical variables.

## 6 Experiment Setup

Software design patterns offer elegant solutions to common problems in software engineering. From a programming standards and a reverse engineering perspective the discovery of patterns in a software artifact represents a step in the program understanding process.

Consider the Object-oriented system as input.

Apply data mining unsupervised learning to identify the patterns in the system.

Clustering is grouping the similar patterns in one cluster and dissimilar pattern in other cluster.

In clustering we choose any one of cluster either Hierarchal or Partitioned which is best to identify the pattern in object oriented system.

Then will show the end product of software engineering is reliable, cost effective with high positive rates.

## 7 Conclusion

The proposed pattern recognition system provides a solution to emphasizing design patterns and applied to the architecture. The model also has extensive contributions to the fields of Agile process Methodology, web Intelligence, Recommendation Systems, and Information Systems and reviews the software engineering. Our work extends to investigate the methods that generate and implement user instance from object oriented system. The present work is framework need to develop implementation of software architecture & design patterns from object oriented system with high positive rates and best quality attributes.

## References

[1] Dekhtyar, A., Hayes, J.H., Menzies, T.: Text is Software Too. In: Proceedings of the International Workshop on Mining of Software Repositories (MSR), Edinborough, Scotland, pp. 22–27 (May 2004)

[2] Hayes, J.H., Dekhtyar, A., Osbourne, J.: Improving Requirements Tracing via Information Retrieval. In: Proceedings of the International Conference on Requirements Engineering (RE), Monterey, California, pp. 151–161 (September 2003)

[3] Hayes, J.H., Dekhtyar, A., Sundaram, K.S., Howard, S.: Helping Analysts Trace Requirements: An Objective Look. In: Proceedings of the International Conference on Requirements Engineering (RE), Kyoto, Japan (September 2004)

[4] Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall (2001)

[5] Beck, K., et al.: The Agile Manifesto (2001), `http://www.agilemanifesto.org` (downloaded March 6, 2009 )

[6] Babchuk, N., Goode, W.J.: Work incentives in a self-determined group. American Sociological Review 16(5), 679–687 (1951)

[7] Badani, M.: Mapping Agile development practices to Traditional PMBOK (June 4, 2001), `http://www.pmiissig.Org/pds/DOCS/BadaniBioandAbstractMappingAgileDevelopmentPractices.doc` (accessed February 29, 2007)

[8] Sliger, M.: Survival guide to going Agile, Rally Software Development Corporation, p. 8 (2006)