Software Component Identification and Composition

A Colloquium Report

Submitted in Partial Fulfillment of Requirements for the Degree of

Masters of Technology

By
Anshuman Sekhar Dash
M.Tech 3^{rd} Sem
Information Security
Reg. No: 2020IS04

Department of Computer Science and Engineering,
Motilal Nehru National Institute of Technology
Allahabad, Prayagraj.

UNDERTAKING

This is to certify, that this colloquium report submitted by me is an outcome of my independent and original work. I have duly acknowledged all the sources from which the ideas and extracts have been taken. The report is free from any plagiarism and has not been submitted elsewhere for publication.

25 October, 2021
Allahabad

(Anshuman Sekhar Dash)
(Reg. no.-2020IS04)

Abstract

Reusing of software is found to have escalate the software engineers work output and also the craftsmanship of the final software product. Component based software engineering (CBSE) is a unique area in software engineering wherein software components i.e a chunk of software that contains a set of related functions and data are searched for and pieced together to form a software system.This method promotes re usability and efficiency in development of software systems. CBSE's purpose is to compose software applications using plug and play software components on the framework.There are several strategies for software component identification in a system among which most widely used is object oriented based clustering.we will look into various approaches used in identifying a software component.We propose a approach which is based on clustering and formal methods to solve the problem of software component identification from a software system.The approach is based on describing a software component using formal methods and then using clustering algorithms that will groups the components in an hierarchy.when components are arranged in an hierarchical manners it provides a means to store ,browse and retrieve reusable software components.we will use the software components identified to compose a different system for comparison and demonstration of correctness of our approach used.

Contents

List of Figures

List of Tables

1 Motivation

Throughout last 10 years wide spread use of software systems in many aspects of life and economy has placed new demands on the software industry for faster development of and software and also to revamp the quality of software and optimizing time take in developing of software.One method of achieving this is to promote software reusability software reusability means reusing of software previously developed which fulfills or nearly fullfills the software requirement As is the case with many software system , no two software systems are developed the same way.Every software system differs in one or more requirements and hence software reusability will be limited unless a software system can be divided into software components each of which can be independently used in software development This is the reason for looking more into component based software engineering ,various techniques used to identify software components in a system and then composition of software systems using components.

2 Introduction

2.1 What is Component Based Software Engineering

From the last 10 years use of computer software has entered into every part of economy which has in turn generated new demand from the software industry to develop reliable and cost effective software quickly and efficiently. Modern software systems are becoming more and more large scale which further increases the development cost , decreases the productivity and increases the cost of maintenance. In the 90's a understanding was developed among software engineers that a software system need not be developed all the way from scratch and instead the software system can be developed by piecing together small pieces of software from previously developed software system and assembling them to make a final product.This idea was further developed into a field in software engineering named component based software engineering(CBSE).

In the context of CBSE a component will be defined as A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.

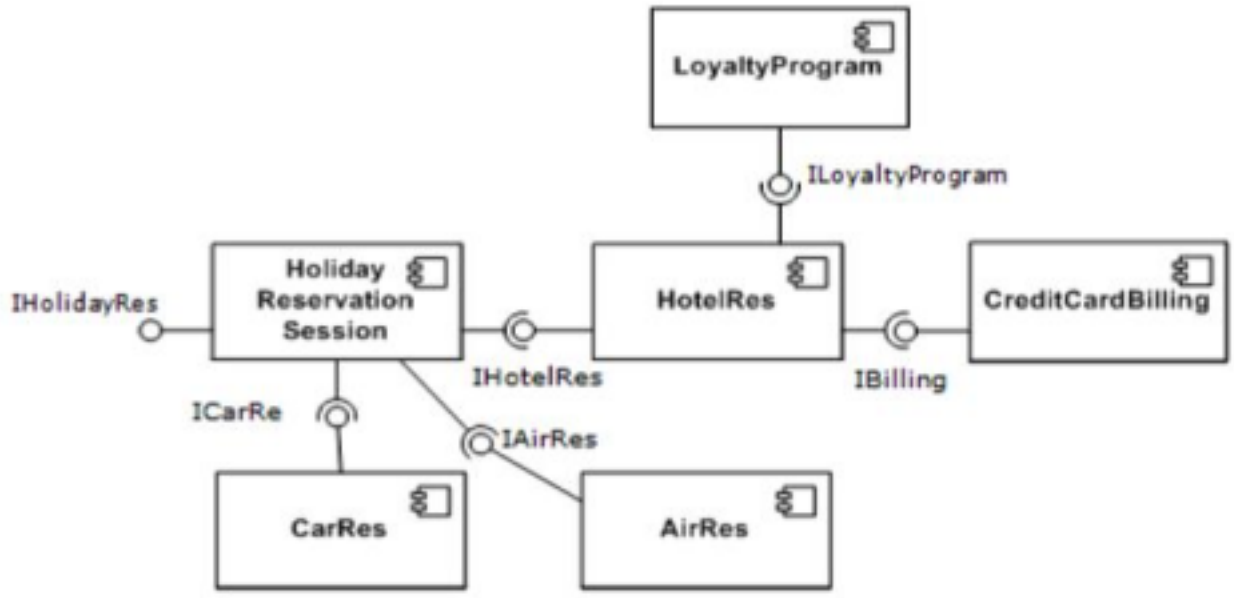


Figure 1: Software Components Example

In the work of [4] a component is defined as abstract, self-contained packages of functionality performing a specific business function within a technology framework

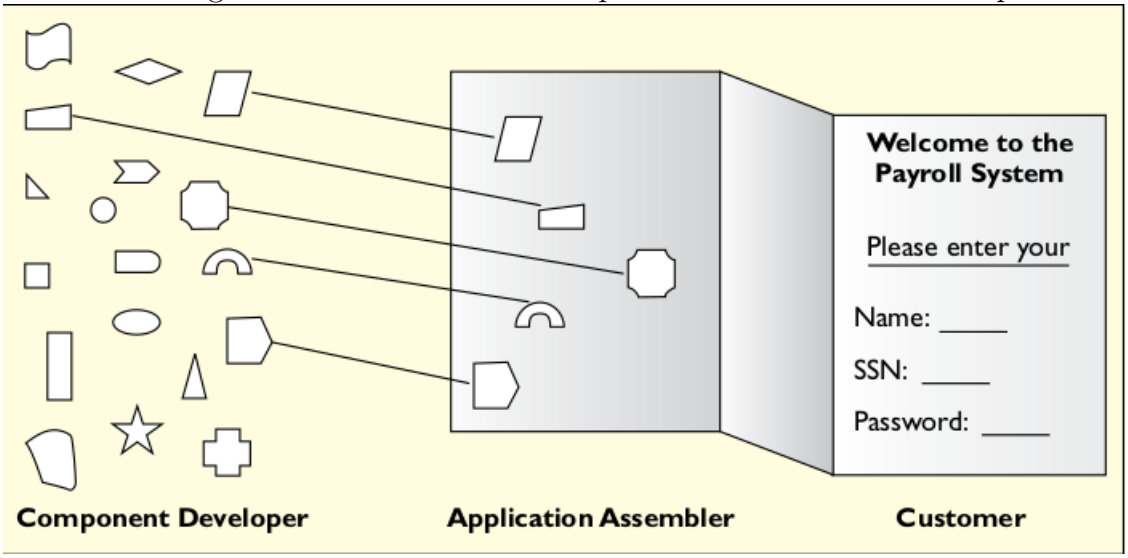
This definition ascertains the fact that a software component is a well defined and independent software that gives it services using well defined interfaces.

Component-based software development is very well known and widely used as a methodology, which helps greatly in the re usability of software and reduces the cost of development significantly.

The primary role of CBSE is to direct making of software system as stitching of parts or components , development of the individual components as reusable entities and maintenance of the system developed with timely replacement of components.

Two main and important problems in Component based software development is identifying software components that can be reused and selecting of components for the development of a software system.Software component identification is one of the primary research problems in CBSE.

Figure 2: Stakeholders in Component based software development



2.2 Stakeholders in Component based software Development

- **Component Developers**:-These are engineers that can be small freelance developers or an entire organisation whose job is to develop and catalogue the software components such that any further system can be developed using the software components they develop

- **Application Assemblers**:-Application Assemblers locate appropriate software components and according to software requirement assemble them into an integrated system.

- **System Developers**:-Developers are responsible for creating the software systems.They take decisions on which components to use based on the business domain and system requirements and availability of the components.They have to make a conscious decision whether to develop the requirement form scratch or use an existing software component which may be having more generalised functions.

- **Customers**:-They use the systems that are developed using component based software development. Their interest lies on faster delivery of system, cheap and reliable to use.

Next there are many different methods that are used to identify components.

Table 1: Artefacts used

Artefact	References
Static/dynamic Call graphs	[2]
Use Cases	[1][13][9]
Source Code	[2]
Sequence Diagram	[9][5]
Relationship between classes	[13]
Business Model Documents	[2][11]
Class Diagrams	[13][9]
Requirements Definition	[19]
Collaboration Diagram	[11]

2.3 Component Identification Problem

Component identification during the software design phase is a process in which the functionalities of a given system are partitioned into non intersecting logical components to provide the starting point for designing software architecture.[5]

Software component identification can be performed in two ways top-down and bottom up. In top-down software component identification is done by analysing business domain models. In bottom-up software component identification is done by forming reusable software components from existing software system. One advantage of bottom up manner is that it can used with legacy systems too to identify and catalogue the software components such that it can be reused.

Identifying components from a software system is a complex problem which depends on different parameters with variable values and limiting conditions.Searching for and managing a component is difficult because a component is a run type entity. This component identification problem belongs to Non-polynomial time problem which can only be solved by exhaustive search of all the different possibilities.

There are many techniques developed in recent years to identify components.These techniques varies according to how the components are grouped ,on what basis are the similarity of two components done[21].

Table 2: Component Identification Techniques

Methods	References
Clustering	
High cohesion and Low coupling	[19]
Use Case	[17]
Formal Concept Analysis	
Fuzzy based analysis	[6]
Evolutionary Based	
Artificial Neural Network	[1]
Genetic Algorithm	[13]
Particle Swarm Optimization	[12]
Other Methods	
CRUD Based	[9]
Graph Partitioning	[2]

In most of the methods the number of components needs to be known in advance but in evolutionary algorithms it is not the case.

In clustering based methods the clustering criteria includes sum of squared errors,variance ratio criterion.Some criteria also include manual weighting of features which are highly influenced by expert opinion.

The various Component identification algorithms that are studied are mentioned in table 2

2.4 Component Composition Problem

Correct and efficient reuse of software components not only depends on proper identification of software components from a system but also on the ways to merge the components. The other problem in CBSE is that of Component composition.Presently object oriented programming languages provide ad- hoc mechanisms for composing software using components. By providing a proper semantic foundation definition we will be able to cleanly integrate all the requirements of the software system into one system[22].

Run time component composition for critical software systems also pose a challenge.Run time component composition is also known as dynamic composition[**?**].

3 Methods used for solving Component Identification Problem

3.1 Clustering

Clustering Algorithms which are used to identify software components contains mainly 3 steps.First, Determining the features and criteria and collection of data.Here data may refer to class object,use case diagrams,class diagrams,methods etc..Second, computing the similarity scores of the data objects collected.Third, using the clustering algorithms to group data objects and form a component.Clustering algorithms are unsupervised and doesn't need extra information for executing[14].

3.1.1 High Cohesion and Low Coupling

[19] used high cohesion and low coupling metric to cluster classes into logical components.Key classes were selected by an expert.Other classes were assigned any one among these classes based on level of dependency.It was very much dependent on the expert since key classes were required to be chosen

3.1.2 Use Case

Use case,object models and collaboration diagrams were used to identify components.Clustering was performed on functional dependencies of use cases.In [18][8] all the different types of clustering algorithms present were used and compared.It required weighting by an expert.

3.2 Artificial Neural Network

[1] used designing phase use case diagrams and components was generated using neural network that was self organized.The criterion for neural network was cohesion and coupling.

The data from use case diagram was extracted in the form of a matrix where each row represented sample data and each column the property. each cell denoted relationship which was between dependency relationship,inclusion relationship and extension relationship.This matrix was normalized and initial number of cluster was calculates using mathematical equation.For the artificial neural network x is a vector of n size which is fed to individual neurons, and highest amount of output giving neuron is selected by the transfer function.After that competitive phase,cooperative phase and adaptation phase proceeds which gives out the software components.

The number of neurons required initially is static and any change in use case the process needs to be repeated from starting.

3.3 Genetic Algorithm

Since evolutionary algorithms are better suited for optimisation problems and identifying software components is an optimisation problem evolutionary algorithms were used[15][13].

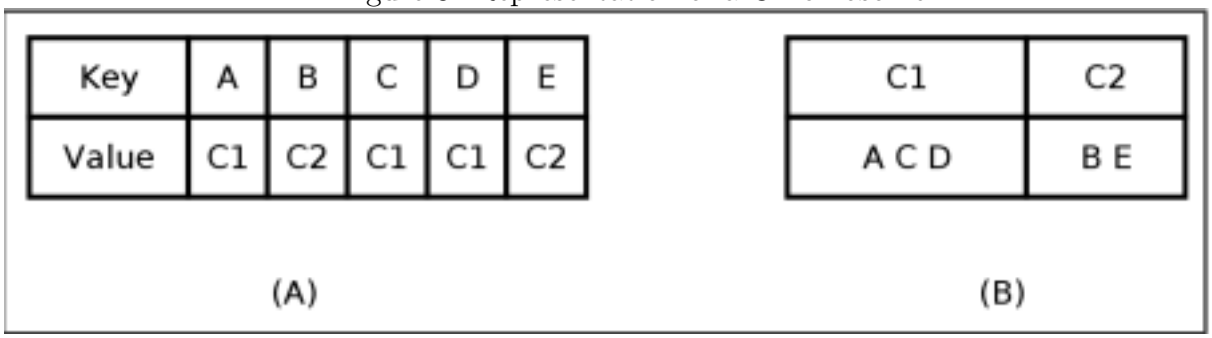
For genetic algorithms 4 elements are required to be defined.These are

- genetic representation of the component
- Fitness function
- how to perform genetic operations i.e evolution of components
- the initial population

In genetic algorithm both the cohesion and coupling between two components is combined into a single objective function.This metric is then used to reject components whose fitness value is low.

The representation is known as chromosome. The chromosome consists of set of partitions

Figure 3: Representation of a Chromosome



Key	A	B	C	D	E
Value	C1	C2	C1	C1	C2

(A)

C1	C2
A C D	B E

(B)

of all classes. The partition is a set of non overlapping and non empty subset of all classes. Evolutionary Algorithms are time consuming and it correctness of components depends a lot on the initial population. Genetic algorithm generated good results in small scale systems but failed to reciprocate good results in large scale system because in large scale systems probability of getting a good initial population is very low.

3.4 Particle Swarm Optimization

PSO is a computational and artificial intelligence technique[20].The important factor of PSO which distinguishes it from other is its fast convergence in compression with other algorithms of global optimization.Feature vector is used to represent a class. The steps performed in [12] for legacy component identification is

1. Weights and Similarity measure calculation
2. Algorithm parameters like swarm size,number of iterations , velocity equation parameters are initialized
3. initial population generated randomly
4. Fitness value calculation
5. setting best particle
6. find the pbest and gbest
7. interaction with software architect

8. process repeated until desired fitness value achieved

Shortcomings of this approach is that it can only be used with object oriented systems.

3.5 Graph Partitioning

In paper [2] object models was related with vertices and edges of graph. In graph partitioning technique the business model or the domain model is mapped into a weighted graph by an expert.This weighted graph is then used to apply graph segmentation technique .weights are manually assigned by experts.The graph is divided using heuristic from graph theories.

The one disadvantage of this method is that it depends a lot on the weights given by the expert.The paper didn't justify the conclusion.

3.6 CRUD Based

The 4 relationship used in this method is Create, Read , Write ,Update and Delete with their priority set as $C > D > U > R$ and these relationship are used to give a semantic relationship.This relationship is then transformed into a matrix .[9]Used business events as inputs.

3.7 Formal concept analysis Based

FCA is a framework which uses mathematics that is used to explain and analyse data and their relationships[3].

[11] uses a framework based on FCA that divides class diagram into logical components which is like that of clustering techniques.Adding on this paper [11] used a new method that was based on fuzzy FCA.Dispersion and distance concept was used to calculate cohesion and coupling metrics.Here a component may belong to more than one cluster.A degree of membership is given to every component with every cluster.a degree of 0 will mean that the respective component does not belong to this cluster.a degree of 1 will mean that the respective component only belongs to the cluster given. Fuzzy clustering is useful when the boundaries between clusters are not well separated.

Disadvantage of this method was that the threshold for dispersion and distance was set manually and it impacted heavily on the result.It also depends on the number of clusters and is susceptible to outliers in the components data.

4 Constructing Software Library using formal Methods

After studying all the previous works we found many limitation that need to be taken care of.Few of the solutions used use case diagrams , class diagrams which are domain level and will be difficult to implement on legacy systems.It is only top down solution and requires prior knowledge of the overall system with the use of use case diagrams sequence diagram etc.These domain models are dependent on how the domain models were formed and is not standardised. Few solution only focused on object oriented systems which cannot be used with procedural and functions based systems.Keeping all the above limitations our solution will be based on bottom up approach and will work on already developed software systems. We Propose a solution where individual software components will be specified using predicate logic[7]. We selected [16] and [10] as the base paper for our proposed solution.[16] used formal language predicate logic to specify software components which can be used for both object oriented and function based systems.since it only requires the source code to specify software components no domain level elements are required from the software system.[10] mentions the metrics which will be used to calculate the corrected of our solution. The specification of a software component will contain an abstract data type and the set of methods that operates on the abstract data type. each of this method will contain interface, type declarations , a precondition and a post condition. The steps are as follows:-

4.1 Formal Specification of Software components

Interface represents syntactic specification of the method. type declarations represents the the type of input output and local variables used. precondition describes the condition of the variables before the method is called post condition describes the condition of the variables after the method is called. The specification follows all the predicate logic conventions such as conjunction dis-junction,rules of inference etc.

type Array:

method ‖E‖ assign_element: Array × E → Array **is**

in(s: Array, e: E)

local()

out(s': Array)

{ **pre**: true }

{ **post**: $len(s') = len(s) + 1 \wedge s'(len(s')) = e$

% len: index of the last element in array with a value

}

method sort: Array → Array **is**

in(s: Array)

local(i, j, min, max: Int)

out(s': Array)

{ **pre**: true }

{ **post**: s' = permutation(s) ∧

($\forall$ i : min ≤ i ≤ max :: ($\forall$ j : min ≤ j < i :: $s'(j) \leq s'(i)$))

}

method ‖E‖ last_element: Array → E **is**

in(s: Array)

local()

out(e: E)

{ **pre**: $len(s) \neq 0$ }

{ **post**: last_element(s) $\stackrel{\text{def}}{=}$ $s(len(s))$

}

Figure 4: Example of predicate logic specification of component [16]

4.2 Lower level hierarchy

We will first generate the lower level hierarchy using the formal specification of software components. The lower level hierarchy will generate fine-grained components on which logical reasoning can be done on the basis of specifications.

The subsumption relationship between 2 components will be used to classify and group together components which can be reused. A component is said to subsume another component when the first component is more general than the second. This abstract data type can be implemented on classes as class can also be termed as an abstract data type.

The lower level hierarchy will be generated by comparing components formed using pair wise comparison algorithm.

Algorithm 1: Pair Wise Algorithm

Input: The set of components from subsumption test algorithm
$S = \{Cmp_1, Cmp_2, ..., Cmp_n\}$

Output: Hierarchy of components by applying generality principle

while $S \neq \{\}$ **do**
  choose some $Cmp_i \mathcal{E} S$;
  $S \leftarrow S \div Cmp_i$;
  $set \leftarrow S$;
  while $set \neq \{\}$ **do**
    choose some $Cmp_j \mathcal{E} S$;
    $S \leftarrow S \div Cmp_j$;
    if $C_i \sqsupseteq C_j$ **then**
      C_i is set as parent of C_j
    else if $C_j \sqsupseteq C_i$ **then**
      C_j is set as parent of C_i

After generating the lower hierarchy of software components in order to form a connect hierarchy clustering algorithm will be done.

The objective to clustering algorithm is to create a set of clusters with low inter cluster similarity and high intra cluster similarity.

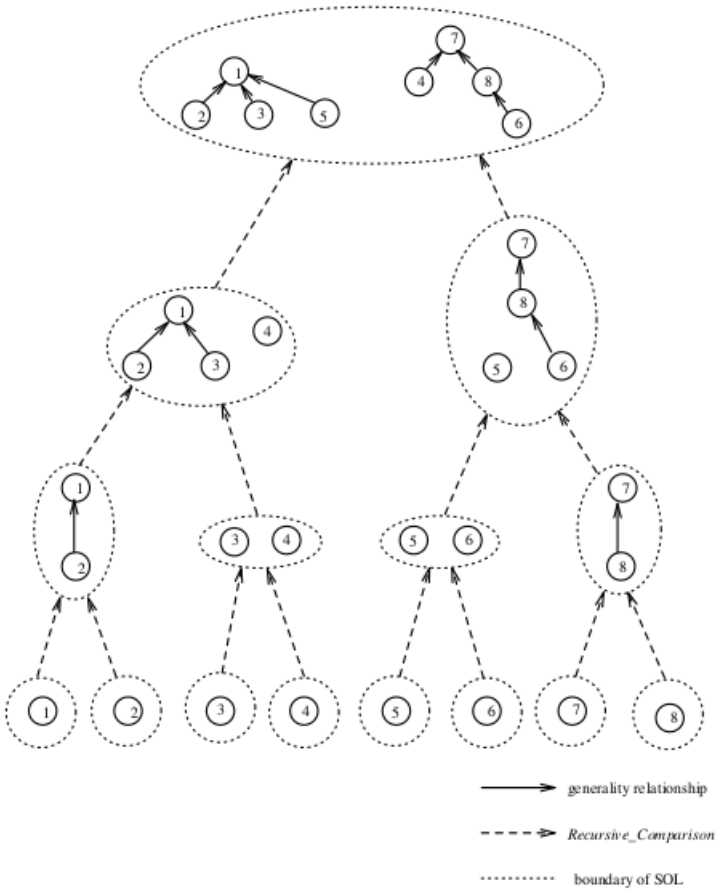


Figure 5: Lower Level Hierarchy Building using recursive comparison[16]

4.3 Measuring Similarity

Similarity of 2 components is represented as $S(x, y)$. In our solution a component is represented in disjunctive normal form consisting of conjuncts. Each conjunct is to be associated with an equivalence class.Example $X = (C_1 \wedge C_2) \vee (C_2 \wedge C_3)$ and $Y = (C_3) \vee (C_2 \wedge C_3)$.For our solution we will use equivalence class as greater(a,b). using all the above information for every component a matrix will be constructed whose dimensions are $m \times (T + 1)$ where m is the number of conjunction and T is the number of equivalence classes.

the matrix will be filled according to the formula below. here x(i,j) represents entry in row i and column j. row i denotes the i^{th} disjunction of the component x .

$$X(i, 0) = u_i \; 0 \leqq i \leqq m \leqq 1 \; and X(i, j) = l \; x_i \; has \; l \; terms \; in \; eq \; class \; j$$

Similarly Y matrix was filled.

from the above 2 matrices that were derived the similarity matrix will be constructed

$$\begin{gathered} for \; all \; i, j \; if \; X(i, 0) = Y(j, 0)) \\ then \; s^{'}(i, j) = \frac{\sum_{t=1}^{T} 2 * min(X(i,t), Y(j,t))}{(X(i,t)+Y(j,t)} \\ else \; s^{'}(i, j) = 0 \end{gathered}$$

Using all the above formulas the similarity between two components is given as

$$s(X, Y) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} s^{'}(i,j)}{N}$$

4.4 Higher Level Hierarchy using Clustering Algorithm

Cluster analysis or clustering is the project of grouping a set of items in the sort of manner that objects in the same group (called a cluster) are more comparable (in some experience) to every aside from to the ones in different group (clusters). It is a main task of exploratory information analysis, and a common technique for statistical data analysis, used in many fields, along with sample recognition, photo analysis, statistics retrieval, bioinformatics, information compression, computer photographs and machine learning. Cluster analysis on its own isn't always one unique algorithm, but the preferred task to be solved. It may be completed by way

of diverse algorithms that range extensively in their know-how of what constitutes a cluster and how to efficaciously find them. Popular notions of clusters consist of businesses with small distances between cluster individuals, dense areas of the facts space, periods or precise statistical distributions. Clustering can therefore be defined as a optimisation problem with multiple objectives. The appropriate clustering set of rules and parameter settings (such as parameters which includes the distance characteristic to apply, a density threshold or the variety of anticipated clusters) depend upon the person information set and supposed use of the results. Cluster analysis as such isn't always an automatic mission, but an iterative system of information discovery or interactive multi-objective optimisation that entails trial and failure. It is frequently essential to alter information preprocessing and model parameters till the end result achieves the preferred houses. There isn't any objectively accurate clustering set of rules, however as it became referred to, clustering choice and parameters lies with that of the programmer. The maximum suitable clustering algorithm for a particular problem often desires to be chosen experimentally, except there's a mathematical purpose to prefer one cluster version over another. An algorithm that is designed for one form of version will generally fail on a information set that carries a greatly exclusive type of model. For example, k-method can't discover non-convex clusters. Clustering algorithms can be separated into 2 categories.One is divisive algorithms and the other is agglomerative algorithms. For divisive algorithms the set X is further divided into a group of clusters using measure of dissimilarity which creates a finer partition. While in agglomerative algorithms a group of finer partitions are merged together using similarity measures to form a courser partition.Each intermediate steps helps in generating a hierarchy of clusters.It gives an understanding of the architecture of the system.The similarity measure is an important factor in the agglomerative hierarchical clustering algorithms. The agglomerative hierarchical clustering algorithms (AHCA) begin from a set of individual entities that are first allocated into small clusters which are then in turn allocated into larger clusters until reaching a final all inclusive clustering. We will be using Agglomerative Clustering since we are identifying components from a software system that is bottom up approach.

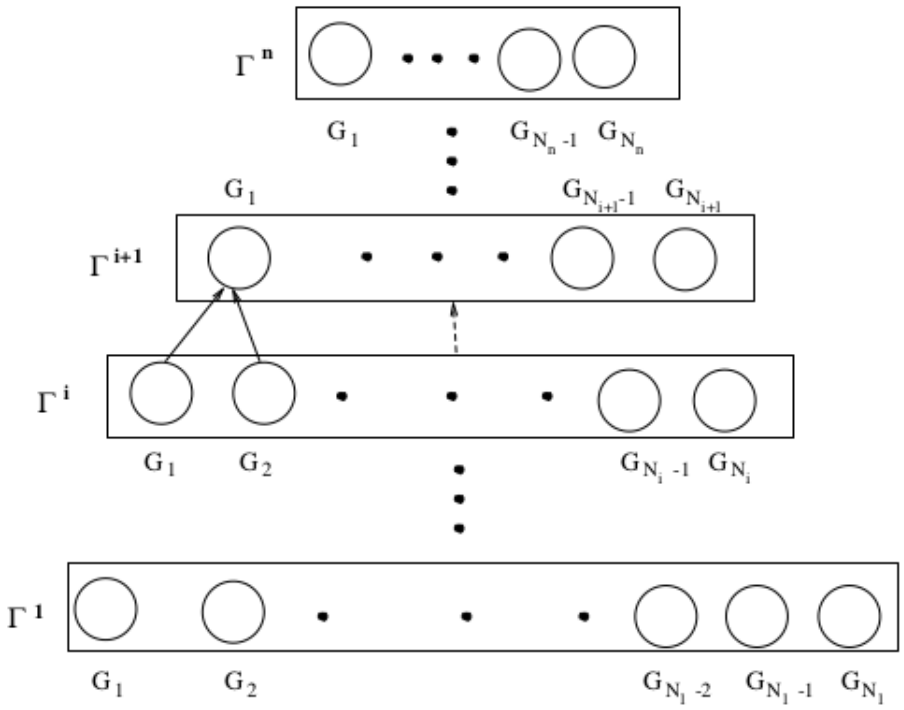


Figure 6: Visual representation of Agglomerative Clustering[16]

Algorithm 2: Agglomerative Clustering

Input: A Set of disjoint Lattices

Output: Set of Clusters Unified into one

1. each top element of the lattice is set a cluster with only single element
2. search for a pair of clusters which have maximum similarity
3. merge the the top 2 most similar clusters into a new cluster
4. calculate new similarity scores for the clusters left
5. proceed until only one cluster is left

5 Conclusion & Recommendation for Future Researches

5.1 Conclusion

We looked into what is component based software engineering,its main problem.We further discussed on various types of approaches taken for component identification problem in component based software development.We determined that Clustering based technique is most extensively used but the approaches here were limited to domain based models and not the actual source code.A different method was needed to use describe a component and which is why use of predicate logic was proposed.Specifying software components using predicate logic will help in also describing legacy systems.We divided our technique to lower level hierarchy and higher level hierarchy.The lower level hierarchy generates the fine grained components which will be used in software reusing.In higher level hierarchy for easy searching and generating a hierarchy of software components clustering will be used.For clustering method we proposed a similarity measuring formula based on absolute weighting scheme.After getting the similarity measure and the lattice for the input to clustering algorithm we used agglomerative clustering since we desired for a bottom up approach.In higher level hierarchy clustering to generate a hierarchy of software components for smooth and fast searching clustering will be used.For clustering method we proposed a similarity measuring formula based on absolute weighting scheme.After getting the similarity measure and the lattice for the input to clustering algorithm we used agglomerative clustering since we desired for a bottom up approach.

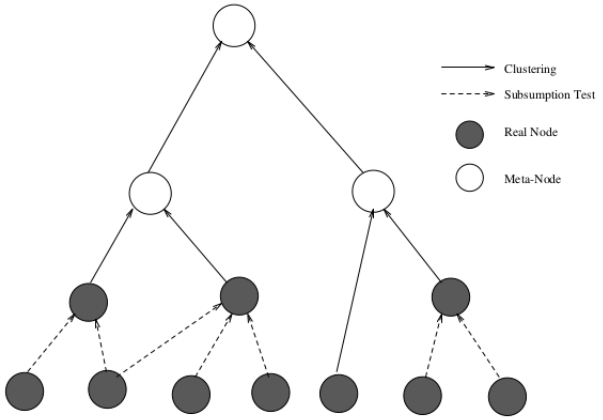


Figure 7: Final resultant hierarchy of software components[16]

5.2 Recommendation for future research

In future research we plan to use the proposed solution on various software systems which will generate reusable components.A variety of software systems will be taken such as Parking Lot system,Chain store system,Library Management system,Online marks analysis system,A B2B auction system for which components will be identified using our proposed solution.These components will be catalogued.Further we will look into composition problem of CBSE and try to find a solution to compose a Airport System using the components catalogue and composition solution. We will use these reusable components to develop our own software system and calculate various metrics like lines of code, number of reusable components degree of coupling and cohesion etc.
Tools we will be using for building the solution:-

1. AUTOSPEC tool to extract formal specifications from primary programming structures of the source code

2. Jupyter notebook for writing and running of python code for clustering

We will also look into using evolutionary methods with formal methods since evolutionary methods gives a good result in optimization problems.In a quick look, the particle swarm op-

timization and artificial neural network methods looked promising and using formal methods along with them guarantees to give out better results.

References

[1] Mohammad Ahmadzadeh, Gholamreza Shahmohammadi, and Mohammad Shayesteh. Identification of software systems components using a self-organizing map competitive artificial neural network based on cohesion and coupling. *ANDRIAS Journal*, 40:721–6513, 02 2016.

[2] Antonia Albani, Sven Overhage, and Dominik Birkmeier. Towards a systematic method for identifying business components. In *Component-Based Software Engineering*, pages 262–277, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] Gabriela Arévalo, Nicolas Desnos, Marianne Huchard, Christelle Urtado, and Sylvain Vauttier. Formal concept analysis-based service classification to dynamically build efficient software component directories. *International Journal of General Systems - INT J GEN SYSTEM*, 38:427–453, 05 2009.

[4] Greg Baster, Prabhudev Konana, and Judy E. Scott. Business components: A case study of bankers trust australia limited. *Commun. ACM*, 44(5):92–98, May 2001.

[5] Dominik Birkmeier and Sven Overhage. On component identification approaches - classification, state of the art, and comparison. In *Component-Based Software Engineering, 12th International Symposium, CBSE 2009, East Stroudsburg, PA, USA, June 24-26, 2009, Proceedings*, volume 5582 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.

[6] Zhengong Cai, Xiaohu Yang, Xinyu Wang, and Aleksander Kavs. Afuzzy formal concept analysis based approach for business component identification. *Journal of Zhejiang University - Science C*, 12:707–720, 09 2011.

[7] B.H.C. Cheng and G.C. Gannod. Abstraction of formal specifications from program code. In *[Proceedings] Third International Conference on Tools for Artificial Intelligence - TAI 91*, pages 125–128, 1991.

[8] Jian Cui and Heung Chae. Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems. *Information & Software Technology*, 53:601–614, 06 2011.

[9] R. Ganesan and S. Sengupta. O2bc: a technique for the design of component-based applications. In *Proceedings 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems. TOOLS 39*, pages 46–55, 2001.

[10] Shabnam Gholamshahi and Seyed Mohammad Hossein Hasheminejad. Software component identification and selection: A research review. *Software: Practice and Experience*, 49, 10 2018.

[11] Haitham S. Hamza. A framework for identifying reusable software components using formal concept analysis. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 813–818, 2009.

[12] Seyed Mohammad Hossein Hasheminejad and Shabnam Gholamshahi. Pci-pso: Preference-based component identification using particle swarm optimization. *Journal of Intelligent Systems*, 28(5):733–748, 2019.

[13] Seyed Mohammad Hossein Hasheminejad and Saeed Jalili. Sci-ga: Software component identification using genetic algorithm. *The Journal of Object Technology*, 12:3:1, 01 2013.

[14] Seyed Mohammad Hossein Hasheminejad and Saeed Jalili. Ccic: Clustering analysis classes to identify software components. *Information and Software Technology*, 57, 06 2014.

[15] Seyed Mohammad Hossein Hasheminejad and Saeed Jalili. An evolutionary approach to identify logical components. *Journal of Systems and Software*, 96, 10 2014.

[16] Jun Jang Jeng and Betty H. C. Cheng. Using formal methods to construct a software component library. In Ian Sommerville and Manfred Paul, editors, *Software Engineering — ESEC '93*, pages 397–417, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[17] Soo Dong Kim and Soo Ho Chang. A systematic method to identify software components. In *11th Asia-Pacific Software Engineering Conference*, pages 538–545, 2004.

[18] Sunil Kumar, Rajesh Bhatia, and Rajesh Kumar. K-means clustering of use-cases using mdl. *Communications in Computer and Information Science*, 270:57–67, 01 2012.

[19] Jong Kook Lee, Seung Jae Jung, Soo Dong Kim, Woo Hyun Jang, and Dong Han Ham. Component identification method with coupling and cohesion. In *Proceedings Eighth Asia-Pacific Software Engineering Conference*, pages 79–86, 2001.

[20] Hamid Masoud, Saeed Jalili, and Seyed Mohammad Hasheminejad. Dynamic clustering using combinatorial particle swarm optimization. *Applied Intelligence*, 38(3):289–314, April 2013.

[21] Fatihah Mohd, Suryani Ismail, Masita Masila Abdul Jalil, Fatin Izatul Nadia Mohd Zulkarnain, and Norshuhani Zamin. A guidelines for controlled experimental design to evaluate the metrics of software component reusability. In *2021 Fifth International Conference on Information Retrieval and Knowledge Management (CAMP)*, pages 85–90, 2021.

[22] Jiafu Tang, Li feng Mu, C.K. Kwong, and X.G. Luo. An optimization model for software component selection under multiple applications development. *European Journal of Operational Research*, 212:301–311, 07 2011.