Software Component Identification and Composition

Presented By

Anshuman Sekhar Dash

M.Tech. 3rd Sem

Information Security

Reg No: 2020IS04

Supervisor: Prof. D. K. Yadav

Computer Science & Engineering Department

Motilal Nehru National Institute of Technology Allahabad

Prayagraj, India

Outline

Introduction

- In Component Based Software Engineering a component is defined as a unit of composition with contractually specified interfaces that performs a required task.
- Identifying and selecting a component according to given requirements from a vast set of components available is a tedious task

Main Objective

- Before starting to build anything, it is always better to know why we need it.
- Reducing the search space in component selection helps in faster development.
- **So a method is required that reduces the search space and gives out components that are according to the requirements specified**

Approach

- Agglomerative Hierarchical Clustering is performed on all the components to generate a hierarchy of components available
- User feeds the Application Requirements using a web browser tool
- The component is identified by following the hierarchy according to the given requirements

Component Features

- Each software Component has various features that includes name, language , type of component, domain, visibility and functionality
- functionality will be used for fine grained searching
- name and id is for distinguishing each component
- rest all will be used for measuring similarity for clustering the components.

Components Used

A	B	C	D	E	F	G
id	Component Name	Language	Type	Domain	Visibility	Functionality
1	Login	Java	JSP	Educational	UI	Login using username and password
2	Login	Java	JSP	Reservation	UI	Login using username and password with
3	Date Picker	Java	JSP	Booking	UI	calendar to pick dates
4	Calendar	JavaScript	React	Booking	UI	calendar to pick dates
5	City Dropdown	Java	JSP	Hotel Booking	UI	dropdown displaying all cities
6	City Dropdown	JavaScript	React	Hotel Booking	UI	dropdown displaying all cities
7	Logout Button	Java	JSP	Educational	UI	logout button
8	Logout Button	JavaScript	React	Educational	UI	logout button
9	Tax Calculator	Java	Java	Finance	Function	tax calculation
10	Tax Calculator	JavaScript	JavaScript	Finance	Function	tax calculation
11	Fare Calculator	Java	Java	Train Booking	Function	fare calculation according to distance , trai
12	Fare Calculator	Java	Java	Hotel Booking	Function	fare calculation according to type of room
13	Fare Calculator	Java	Java	Flight Booking	Function	fare calculation according to distance ,clas
14	Payment	Java	JSP	Payment	UI	payment page displaying options like credit
15	Payment	JavaScript	React	Payment	UI	payment page displaying options like credit
16	Payment Handler	Java	Java	Payment	Function	handle payment from user

Agglomerative Hierarchical Clustering

- Agglomerative hierarchical clustering is a unsupervised learning algorithm
- It groups similar feature vectors into groups
- initially each feature vector is considered a individual cluster
- until the number of clusters does not reduce to one:-
  1. compute similarity of each feature vector
  2. merge the 2 most similar feature vectors
  3. update the similarity matrix
- a hierarchy of feature vectors is generated

ACH Algorithm

Algorithm 1: Component Clustering

Result: one or more clusters

Input: The set $X = \{x_1, x_2, .., x_n\}$ *and the similarities* $s(x_i, x_j), 1 \leq i, j \leq n$

Limit=1 ,N=n

for $1 \leq i, j \leq N, i \neq j$ **do**

  $sim(G_i, G_j) = s(x_i, x_j)$

end

while $N \neq Limit$ **do**

  N=N-1

  Find pair of clusters G_p *and* G_q *such that*

  $sim(G_p, G_q) = max\{x \epsilon G_i, y \epsilon G_j \; s(x, y)\}$

  update $sim(G_i, G_j)$

end

Clustering of Components

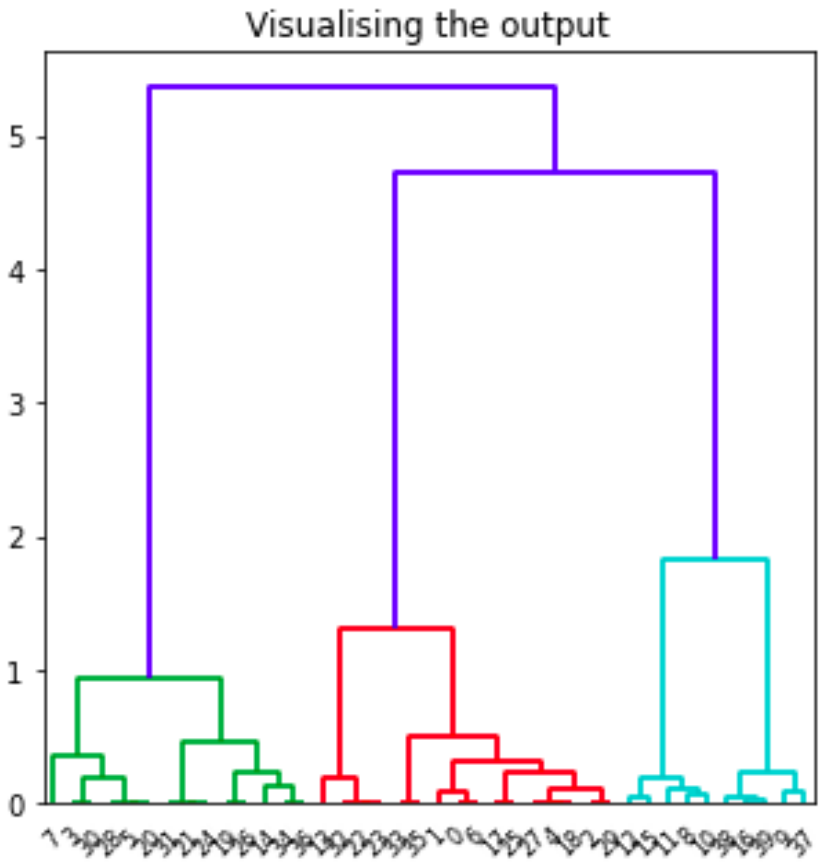

Figure 1: Hierarchical Clustering of Components

Fine grained searching

- after getting a subset of components using the clusters generated and requirements Fine grained searching is performed to get the exact set of components along with the similarity scores
- algorithm to get the similarity score of requirements keywords and functionality attribute of each component in the subset of components
- the similarity score is based on the $numberofkeywordsmatched/numberofwords$ in functionality

Example Structure

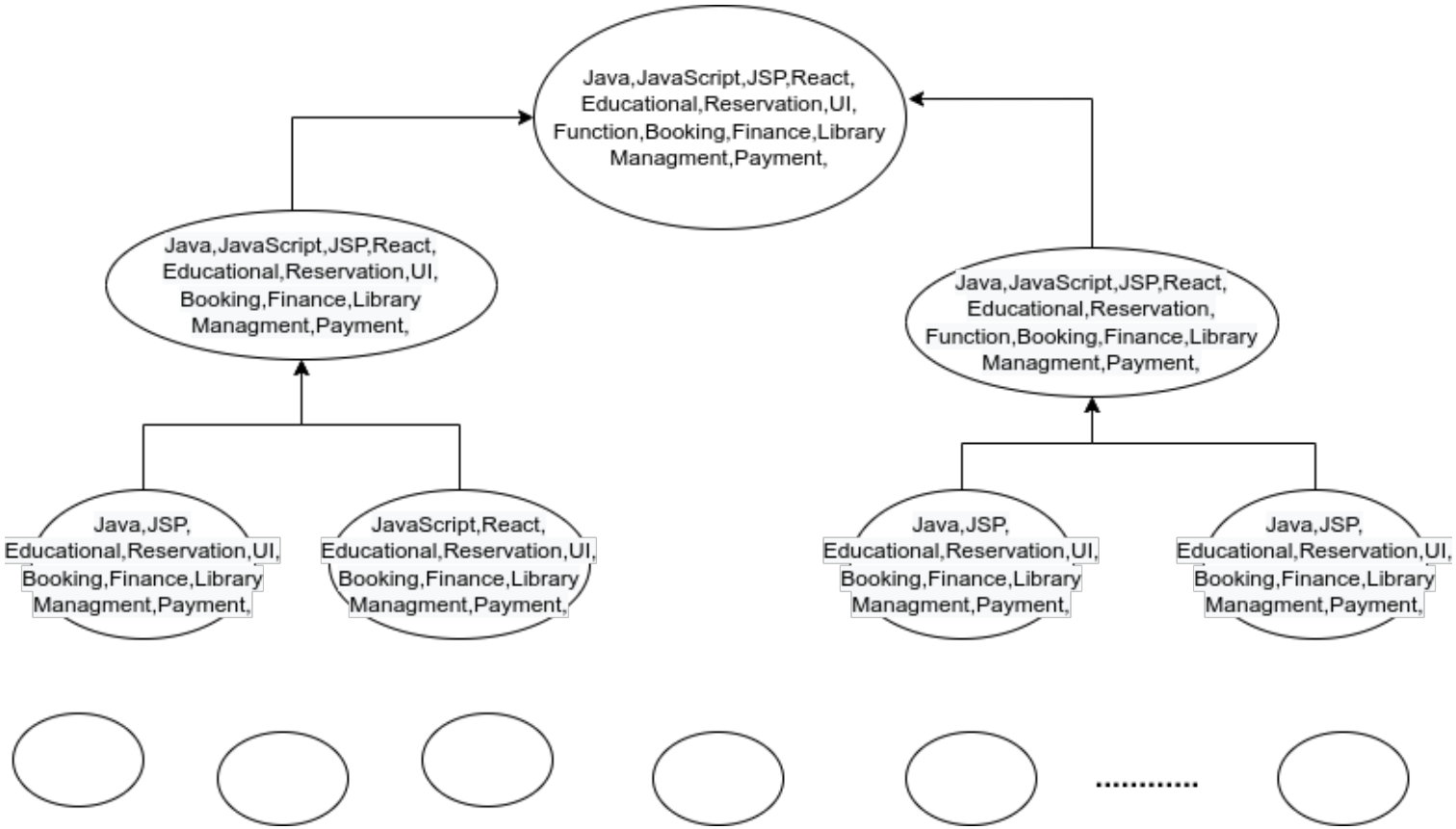

Figure 2: Example Hierarchy of Components Structure

Fine grained searching algorithm

Algorithm 2: Fine grained Searching

Result: Similarity Score for every component in x with requirement

Input: The set $X = \{ x_1, x_2, .., x_n \}$

```
i=1
while i ≠ n do
    i=i+1
    matched=0
    for keyword ←keywords do
        if keywordinx_i then
            matched← matched + 1
        end
    end


    similarityScore← matched ÷ len(x_i)
end
```

Steps to perform

1. AHC algorithm is performed on all the components
2. currently requirements are fed directly in code
3. keywords are generated from the requirements by removing punctuation and stop words
4. the cluster names are matched with keywords ,on matching the set of components for fine grained searching is reduced
5. fine grained search is performed on the subset of components and all the components are displayed in that cluster along with similarity score

Results

The query text given is "A java component for railway reservation containing login using username and password with captcha functionality , date picker , dropdown for cities , train class and quota , calculation of fare with distance , train class and quota , dedicated payment page with options like debit card , credit card , upi , page to display list of trains "

Comparison with expert

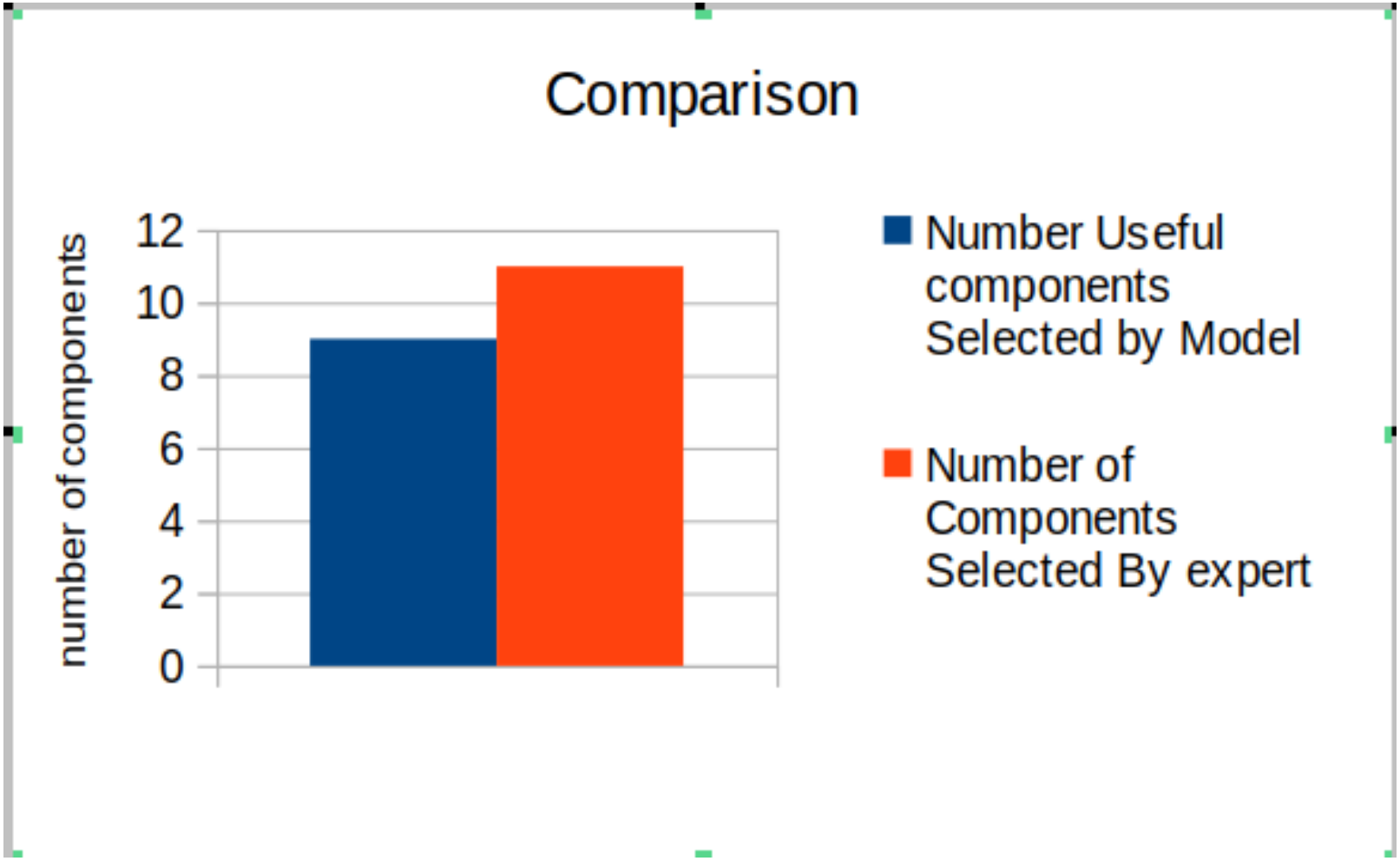

Figure 3: Comparison with an expert

End Result

	Component Name	Language	Functionality	similarityScore
10	Fare Calculator	Java	fare calculation according to distance , train...	0.309859
25	Train Selector	Java	dropdown displaying train class and quota	0.243902
0	Login	Java	Login using username and password	0.212121
1	Login	Java	Login using username and password with captcha...	0.192982
17	Train Display	Java	display list of trains according to origin and...	0.142857
29	Calculate Button	Java	Calculate Button	0.125000
13	Payment	Java	payment page displaying options like credit ca...	0.095238
8	Tax Calculator	Java	tax calculation	0.066667
2	Date Picker	Java	calendar to pick dates	0.045455
15	Payment Handler	Java	handle payment from user	0.040000

Figure 4: Components identified with similarity score

Discussion

- The login component with captcha is found to have less similarity then simple username and password component
- City selector component was not picked up in the subset of components for fine grained search

Conclusion

- Searching for reusable components according to given requirements gets challenging with the increase in number of components developed
- Reducing search space using component attributes while searching for components will be helpful in faster and correct retrieval.
- AHC is helpful in grouping together of components according to similarity based on domain , programming language ,visibility and type.
- The hierarchy generated reduces the need to perform fine grained searching on all the components.

Future Work

- Representation of Components using formal methods
- Better fine grained searching using natural language processing techniques
- using weights to some key requirements like language and domain
- getting ratings about the search results for performing supervised learning algorithms
- Compose an application using the search performed

References I

[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.

References II

[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.

References III

[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] Iria Estevez-Ayres, Luis Almeida, Marisol Garcia-Valls, and Pablo Basanta-Val. An architecture to support dynamic service composition in distributed real-time systems. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 249–256, 2007.

References IV

[10] 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.

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

[12] Maen Hammad and Rua'a Hasan Banat. Automatic class decomposition using clustering. *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*, pages 78–81, 2021.

[13] 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.

References V

[14] Mehdi Hariati. Formal verification issues for component-based development. *Informatica*, 44, 12 2020.

[15] 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.

[16] 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.

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

References VI

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

[19] 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.

[20] 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.

[21] 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.

References VII

[22] 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.

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

[24] 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.

References VIII

[25] 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.

Thank You!
Any Questions?