

# A framework for reusable domain specific software component extraction based on demand

N Md Jubair Basha<sup>1</sup>, Gopinath Ganapathy<sup>1</sup>, Moulana Mohammed<sup>2</sup>

<sup>1</sup>Department of Computer Science and Engineering, Bharathidasan University, Tiruchirappalli, India

<sup>2</sup>Department of CSE, Koneru Lakshmaiah Education Foundation, Vijayawada, India

## Article Info

### Article history:

Received Mar 21, 2024

Revised May 12, 2024

Accepted Jun 5, 2024

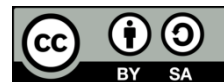
### Keywords:

Demand based extraction  
Domain specific component  
Feature points  
Reusability matrix  
Versioning

## ABSTRACT

The majority of organizations use an agile software development methodology. Standard analysis and design processes are abandoned due to the enormous demand of generating the product within time and budget. This may result in a lack of high-quality software while components are not constructively reused. The components are identified at a later stage in the majority of component approaches. To address such challenges, a methodology for extracting demand-based domain-specific software components from the repository was developed. The process for reusing current components is described in depth with various domain-specific components, and the suggested framework is for extracting demand-based reusable domain-specific software components.

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

N Md Jubair Basha

Department of Computer Science and Engineering, Bharathidasan University

Tiruchirappalli, Tamil Nadu, India

Email: jubairbasha@gmail.com

## 1. INTRODUCTION

Utilising pre-existing software assets to develop or change software systems is known as software reuse [1]. The software community has given the idea of software reuse a lot of attention because of its alleged benefits, which includes improved product quality, lower costs and schedules. The goal is to create and maintain a library of interchangeable parts that form the basis for new products in a certain functional area. The employment of reusable parts is gradually replacing the use of proprietary and monolithic technology [2]. The necessity to reduce life cycle costs, enhance software quality, and maximize resources needed for system development and testing is the driving force behind this shift. More productivity, quality, and dependability can be achieved with an efficient software reuse process, which also shortens implementation times and costs. Starting a software reuse process requires an upfront cost, but after a certain number of reuses, it becomes cost-effective. To summarize, creating a reuse process and repository results in the creation of a knowledge base that becomes better with each reuse. This lessens the risk associated with new initiatives that rely on repository knowledge by reducing the amount of development work that will eventually be needed for subsequent projects.

The use of domain-specific components has a number of noteworthy advantages. The application of component reuse reduces costs and schedules by doing away with the need to create the component from the ground up. It is possible for the component to be modified if it is seen necessary. The word "reduced" describes a situation or state in which something is lessened or when resources are devoted primarily-more than 60% to testing operations in software development. The testing effort is reduced through the use of domain-specific components.

The section that follows is the structure of this work. Section 1 contains the paper's introduction. In section 2, the pertinent literature is outlined and the inclusion of software reuse is clarified. In contrast, the domain engineering process is summarised in section 3. Section 4 discusses the various domain-specific component frameworks. Section 5 looks at the procedure for extracting domain-specific, demand-based software components. This paper is clearly concluded in section 6.

## 2. RELATED WORK

Software reuse is the act of creating new software solutions by utilising pre-existing software components or by leveraging software knowledge. The two main types of reusable assets are software knowledge and software that can be reused. The likelihood that a software asset will be used again is known as reusability [3]. Software reuse is the practice of repeatedly employing pre-existing software components, sometimes known as "designed software for reuse," during the development process [4]. Reusing software helps companies in a number of ways, such as when it comes to managing the complexity of software development, producing better products, and increasing production efficiency. The use of design reuse practices has increased significantly in the modern era, especially when it comes to object-oriented class libraries, application frameworks, design patterns, and related source code [5]. The key strategy for accomplishing software product line development is still component containment [6]. To make the process of retrieving components easier, a significant amount of data needs to be collected, stored, and examined. Maurizio has created a method for creating a software catalogue automatically that incorporates tools for information retrieval and preservation [7]. There are two primary categories of software reuse: process and product reuse. Product reuse refers to the process of reusing a software component by assembling and integrating components to create a new component. The process of reusing outdated components that are acquired from a repository is referred to as process reuse. These parts may be used again, either exactly as they are now or with minor adjustments. Versioning these components can lead to the archiving of the modified software component. Based on the particular domain needs, these components can subsequently be categorised and chosen [8].

If multiple products use the component repository, that is good. This implies that several jobs should be handled simultaneously by the component system. Purchasing parts that are necessary for the development process is a prerequisite for initiating new initiatives. The project ideas should be reviewed by a committee consisting of seasoned designers and a department representative from the component department; this will form a committee for software components. It is necessary to evaluate the necessity of developing the recommended components. On the day that the decision to proceed with component development is made, the component construction phase starts. When the component is prepared, it is changed to the version state depicted in Figure 1 and added to the repository. Analysing the software component group's worth should be done when the component is in use. Which portion is utilised the most frequently? Which products see absolutely no use at all? To what extent do the components provide benefits? This work contributes to the advancement of the component system.

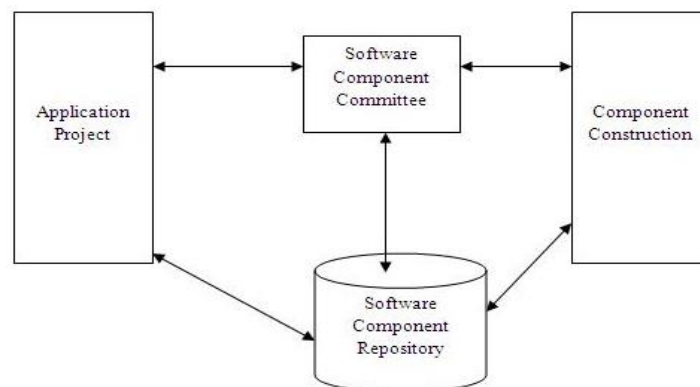


Figure 1. Component management organisation

This study's collection of interface measurements has demonstrated that measuring component interfaces can provide more accurate and relevant data for component reusability research. Metrics are significantly more effective at delivering a lot of valuable information through interfaces than non-

automatable techniques. These metrics provide a deeper insight into the resources associated with component interfaces. Completing a reusability research on the components that were assessed was the task for the metrics lecture, which required in-depth understanding of these components. The application of reusability analysis to components is the main emphasis of this investigation, as metrics practitioners still find it difficult to understand. There is nothing in the user's text that has to be redone [9]. Al Omara *et al.* [10] via examining Stack Overflow, revealed insights about how engineers discuss software reuse. Component identification has been defined as an NP-Complete job by Cai *et al.* [11]. Numerous studies have frequently examined a matrix study, mostly depending on graph-based methods [12], clustering analyses [13]-[15], evolutionary techniques [16]-[18] for component identification, and [19], [20] for software component evaluation that only takes coupling and cohesive characteristics into account.

Architecture is shown from programme execution in [21]-[23] by defining a set of components; however, the interactions between classes are not mentioned. In an effort to compile the key reusability factors for component-based systems. In order to construct a modern software system, Aggarwal and Kumar [24] were unable to identify the software that was essentially reusable. Following a careful analysis of the literature, it was discovered that certain approaches to component design did not use a preplanned form, while other techniques used a corresponding plan of component types to structure a component. This represents a study gap in the earlier studies. As stated in [25], [26] when assessing and evaluating the reusability of the components, the interaction and invocation of the components are not taken into account. Presenting a strategy to component identification that considers component interaction through component reusability analysis and assessment is urgently needed to bridge this research gap.

### 3. OVERVIEW OF DOMAIN ENGINEERING

Domain engineering (DE) is a crucial procedure that generates and efficiently manages reusable components to guarantee that the architectural design sufficiently satisfies the particular requirements of the assigned domain [27]. A group of application systems with comparable software needs is referred to as a "domain" when it comes to the functional areas they cover [28]. The DE technique is depicted in Figure 2. Domain analysis, domain design, and domain implementation are the three keystones of DE. Domain analysis is carried out using the DARE-COTS tool [29]. For a collection of systems to have the generic variable attributes, there needs to be a relevant domain in the pre-phase.

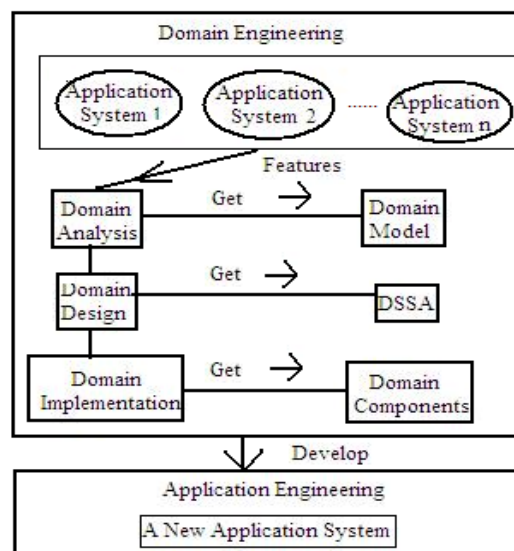


Figure 2. Domain engineering process

### 4. DOMAIN SPECIFIC COMPONENT FRAMEWORKS

The development of domain specific component frameworks (DCSF) is becoming increasingly necessary in light of the notable advancements in software system development across multiple disciplines. Agile principles are included into the development procedures of several software development strategies. Pattern recognition has shown the development of domain specific component frameworks. In order to

provide a thorough framework, Loiret *et al.* [30] conducted a pattern analysis and presented the idea of domain components in their study. By looking at the domain specific services, this framework provides a uniform method for defining the semantics of domain components.

**5. A DEMAND-BASED EXTRACTION METHODOLOGY FOR REUSABLE DOMAIN-SPECIFIC SOFTWARE COMPONENTS**

Because of the benefits of reuse-driven techniques, software systems are not created from scratch. The reuse-driven approach helps to reduce testing costs to some level while also allowing for product delivery on schedule and under budget. The organisation consequently becomes increasingly productive. The method for reusing the parts is as follows:

- Step 1: determine the functional specifications (increments).
- Step 2: search the component repository
- Step 3: the trade-off benefits may be evaluated at the conclusion of the reuse-driven cycle, which not only saves money and time but also yields a high-quality output.

Reusing components that have comparable functionality can help produce high-quality products faster and on a less budget. A component is saved in the component repository along with the updated version number, effort, and time required. A component with partial functionality can be used to modify the source component. The component repository grows as a result of contributions of updated or new components. The component extraction technique is applied if no component is able to provide the desired functionality.

Figure 3 presents how to identify functional requirements and mine components from the repository. Developing a new component or modifying the existing components requires certain effort. The following metrics are used to maintain and evaluate the optimization of effort in reusing a component. Assuming that there are n modules and m concerns (functional requirements), the reusability information can be obtained using the following matrix (Re-usability matrix) RMF.

$$C_i M_j = 1 \text{ (if the concern } C_i \text{ is implemented in Module } M_j) \\ = 0 \text{ (otherwise)}$$

Similarly, non-functional requirements (RMN) needs could also have a reusability matrix.

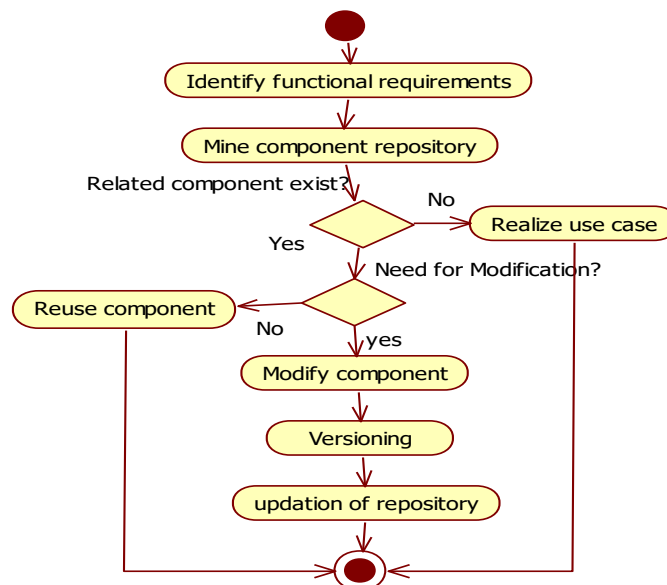


Figure 3. Activity diagram for extraction of reusable domain specific components

As new components are created or old ones are altered, the concerns will be uploaded to the repository. When a new system needs to be put into place, it can be broken down into modules, and a reusability matrix can be created once the issues that can be reused are determined. This also applies to the altered parts. Table 1 lists the reusability matrix, which incorporates the invocation of different components. Numerous issues have been detected with nearly 15 components connected to the nine modules. The

corresponding method or modules will be invoked whenever there is an issue with linked domain-specific components, as seen in Figure 4.

Table 1. Reusability matrix (RMN)

	M1	M2	M3	M4	M5	M6	M7	M8	M9
C1	1	1	0	0	0	1	1	1	1
C2	0	0	1	1	1	0	0	0	0
C3	1	1	0	0	0	1	1	1	1
C4	1	1	1	1	0	0	0	1	1
C5	0	0	1	0	1	0	1	1	1
C6	1	1	1	1	1	1	1	0	0
C7	1	1	1	1	0	0	0	0	1
C8	1	1	1	1	1	1	0	0	0
C9	0	0	0	1	1	0	1	1	1
C10	1	1	1	1	1	1	1	0	0
C11	0	0	0	1	1	0	1	1	1
C12	1	1	1	0	0	0	1	1	1
C13	1	1	1	1	1	1	1	1	1
C14	0	0	0	0	0	0	0	1	1
C15	1	1	1	1	1	1	0	0	0

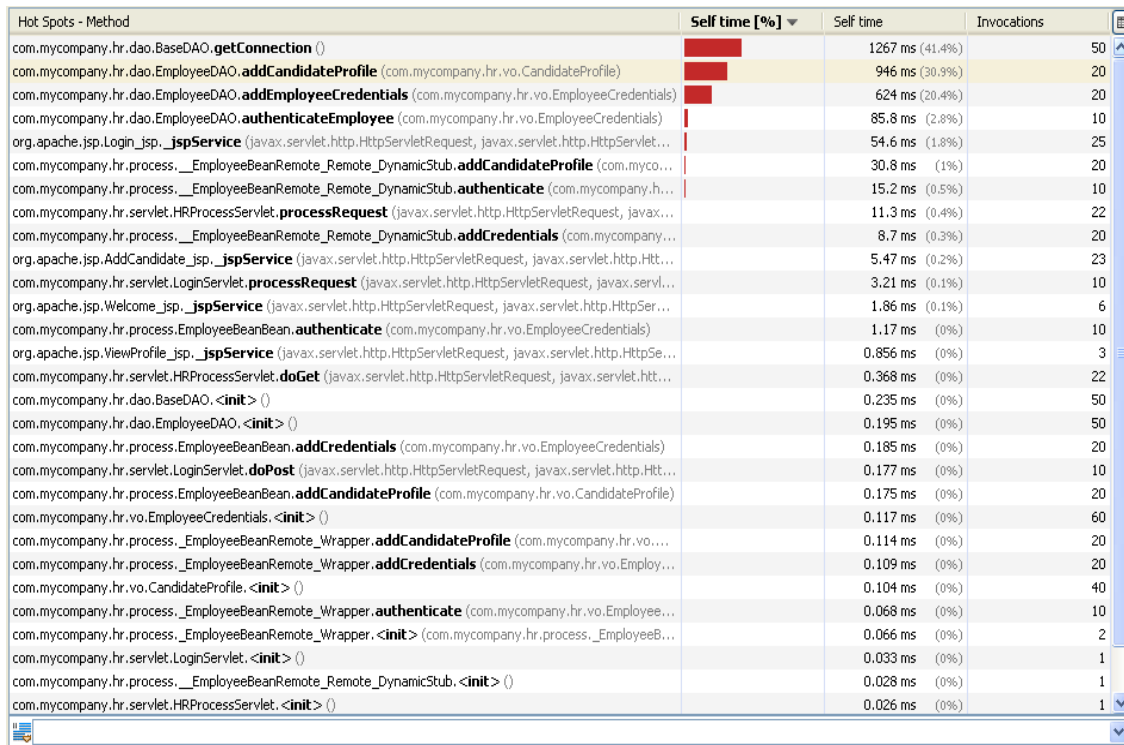


Figure 4. Components interaction and invocation extraction

According to the requirements, the developers can easily recognize the common behavior of the components using the feature points (FP). In Figure 5, FP1 and FP2 are the feature points identified in the C1 'configured reusable component in the façade which is in new versions. FP3 is the feature point identified in the C2 'configured reusable component in the façade which is in new versions. FP1, FP2 and FP3 are identified using the versioning of various components. The feature points were identified as a versioning of the features with their behavioral aspects of the components. It was found that FP1, FP2 and FP3 show at least 2 behavioral characteristics of the selected components. This is achieved through the lack of method cohesion (LCOM). Thus, feature points help identify the variations of configured reusable components in the features. Finally, Figure 6 explores how multi-levels of domain-specific software components required for various products.

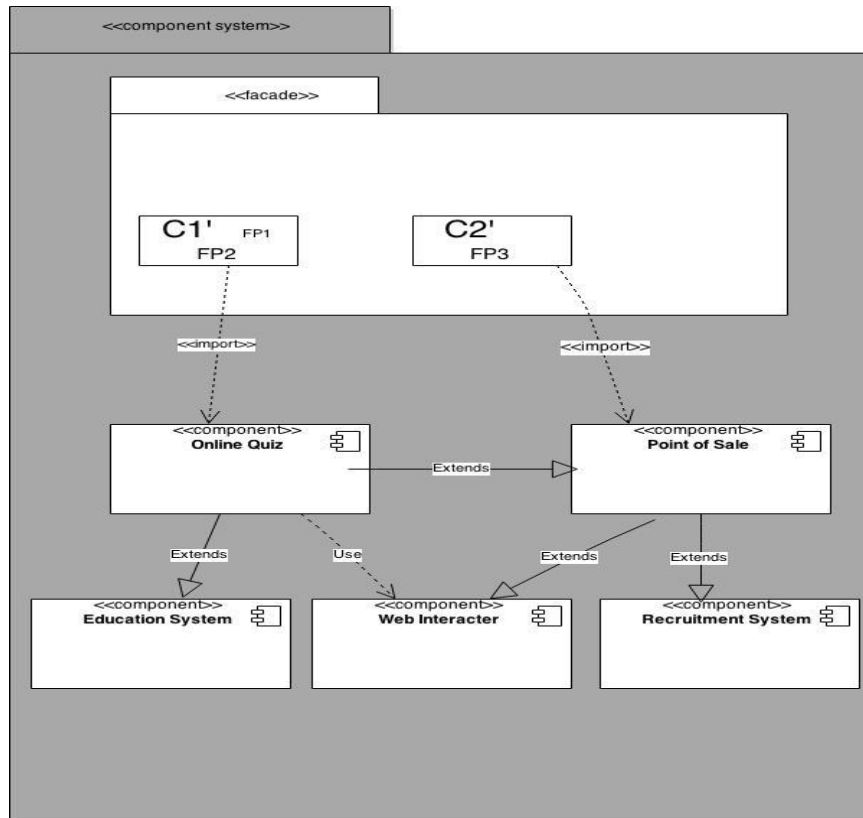


Figure 5. Domain specific reusable software components extraction

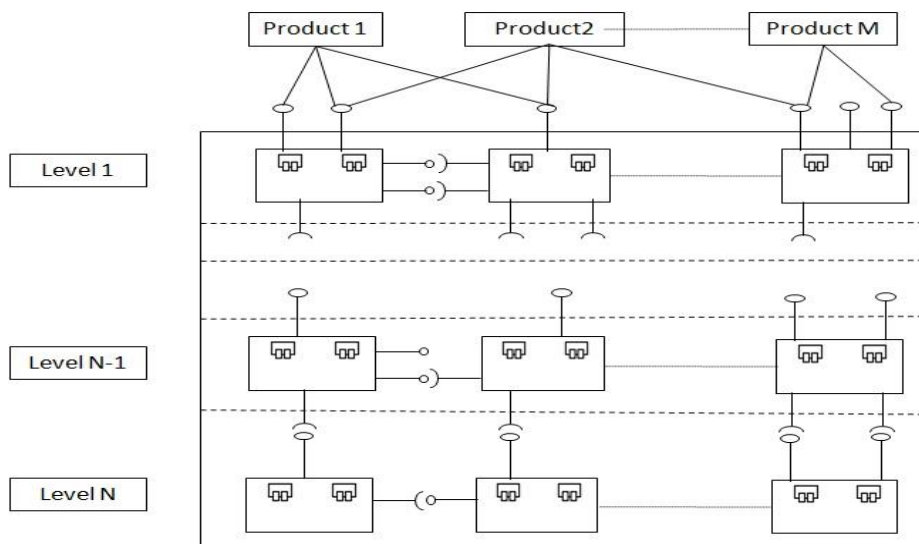


Figure 6. Multi-levels of domain specific software components required for various products

**6. CONCLUSION**

Organizations benefit greatly from component-based systems built with demand based reusable domain specific software components. The most difficult aspect of reuse is selecting the appropriate reusable from a vast array of options and adapting reusability to current requirements. Different techniques and design flaws in reusable domain-specific components have been examined in earlier work. Until date, it appears that there has been no widely acknowledged standard for the design of demand based reusable domain-specific software components. The proposed methodology is used to develop configurable reusable components.

As mentioned in the preceding sections, the suggested strategy is realized and executed utilizing several domain-specific components and implemented using various domain-specific applications. The degree of reusability of domain-specific components is evaluated and compared to several ways, with the suggested framework being found to be superior to the others and extracted the demand-based domain-specific components. The suggested framework is well-suited, may be used in the software business, and can improve outcomes when extracting demand based reusable domain-specific software components. The reuse design guidelines for component quality characteristics can be applied in the future as part of the ongoing effort.




## REFERENCES

- [1] S. Mahmood, R. Lai and Y. S. Kim, "Survey of component-based software development," *IET Software*, vol. 1, no. 2, 2007, doi: 10.1049/iet-sen:20060045.
- [2] H. K. Kim and Y. K. Chung, "Transforming a legacy system into components," *Springer-Verlag Berlin Heidelberg*, 2006, doi: 10.5815/ijitcs.2015.09.07.
- [3] W. B. Frakes and K. Kang, "Software reuse research: status and future," in *IEEE Transactions on Software Engineering*, vol. 31, no. 7, pp. 529-536, July 2005, doi: 10.1109/TSE.2005.85.
- [4] X. Wang and L. Wang, "Software reuse and distributed object technology," *2011 Fourth International Joint Conference on Computational Sciences and Optimization*, Kunming and Lijiang City, China, 2011, pp. 804-807, doi: 10.1109/CSO.2011.243.
- [5] J. Sametinger, "Software engineering with reusable components," *Springer-Verlag*, ISBN 3-540-62695-6, 1997, doi: 10.1007/978-3-662-03345-6
- [6] J. He, R. Chen and W. Gu, "A new method for component reuse," *2009 2nd IEEE International Conference on Computer Science and Information Technology*, Beijing, 2009, pp. 304-307, doi: 10.1109/ICCSIT.2009.5234941.
- [7] M. Pighin, "A new methodology for component reuse and maintenance," *Proceedings Fifth European Conference on Software Maintenance and Reengineering*, Lisbon, Portugal, 2001, pp. 196-199, doi: 10.1109/CSMR.2001.914987.
- [8] Y. Liu and A. Yang, "Research and application of software-reuse," *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, Qingdao, China, 2007, pp. 588-593, doi: 10.1109/SNPD.2007.436.
- [9] M. A. S. Boxall and S. Araban, "Interface metrics for reusability analysis of components," *2004 Australian Software Engineering Conference. Proceedings.*, Melbourne, VIC, Australia, 2004, pp. 40-51, doi: 10.1109/ASWEC.2004.1290456.
- [10] E. A. AlOmar, A. Peruma, M. W. Mkaouer, C. Newman, and A. Ouni, "How is software reuse discussed in stack overflow?," in *The Proceedings of the 2023 Conference on Systems Engineering Research. CSER 2023*, 2024, pp. 357-372, doi: 10.1007/978-3-031-49179-5\_24.
- [11] Z.-G. Cai, X.-H. Yang, X.-Y. Wang and A. J. Kavs, "A fuzzy formal concept analysis-based approach for business component identification," *Journal of Zhejiang University Science C*, vol. 12 no. 9, pp. 707-720, 2011, doi: .10.1631/jzus.C1000337.
- [12] M. A. Khana and S. Mahmood, "A graph based requirements clustering approach for component selection," *Advances in Engineering Software*, vol. 54, pp. 1-16, 2012, doi: 10.1016/j.advensoft.2012.08.002.
- [13] S. M. H. Hasheminejad and S. Jalili, "CCIC: clustering analysis classes to identify software components," *Information and Software Technology*, vol. 57, pp. 329-351, 2015, doi: 10.1016/j.infsof.2014.05.013.
- [14] J. F. Cui and H. S. Chae, "Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems," *Information and Software Technology*, vol. 53, no. 6, 2011, doi: 10.1016/j.infsof.2011.01.006.
- [15] G. Shahmohammadi, S. Jalili, and S. M. H. Hasheminejad, "Identification of system software components using clustering approach," *The Journal of Object Technology*, vol. 9, no. 6, pp. 77-98. doi: 10.5381/jot.2010.9.6.a4.
- [16] S. M. H. Hasheminejad and S. Jalili, "SCI-GA: software component identification using genetic algorithm," *The Journal of Object Technology*, vol. 12, no. 2, 2013, doi: 10.5381/jot.2013.12.2.a3.
- [17] S. M. H. Hasheminejad and S. Jalili, "An evolutionary approach to identify logical components," *Journal of Systems and Software*, vol. 96, pp. 24-50, 2014, doi: 10.1016/j.jss.2014.05.033.
- [18] N. Padhy, R. P. Singh, and S. C. Satapathy, "Software reusability metrics estimation: algorithms, models and optimization techniques," *Computers & Electrical Engineering*, vol. 69, pp. 653-668, 2018, doi: 10.1016/j.compeleceng.2017.11.022.
- [19] G. Priyalakshmi and R. Latha, "Evaluation of software reusability based on coupling and cohesion," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 10, pp. 1455-1485, 2018, doi: 10.1142/S0218194018500420.
- [20] K. Kaur and G. Kaur, "Component reusability of a software system based on cohesion and coupling," *Indian Journal of Science and Technology*, vol. 9, no. 27, pp. 1-6, 2016, doi: 10.17485/ijst/2016/v9i27/94727.
- [21] C.Liu, B. F. van Dongen, N. Assy and W. M.P. van der Aalst, "A general framework to identify software components from execution data," *ENASE*, 2019, doi: 10.5220/0007655902340241.
- [22] N. M. J. Basha and S. A. Moiz, "A methodology to reconfigure victim components using modularity," *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, Indore, India, 2012, pp. 1-6, doi: 10.1109/CONSEG.2012.6349523.
- [23] N. Md. J. Basha and S. Choudhury, "Assessment of reusability levels on domain-specific components using heuristic function," *Innovations in Computer Science and Engineering: Proceedings of the Third ICICSE*, 2015, pp. 153-161, doi: 10.1007/978-981-10-0419-3\_19.
- [24] J. Aggarwal and M. Kumar, "Software metrics for reusability of component based software system: a review," *The International Arab Journal of Information Technology*, vol 18, no. 3, 2021, doi: 10.34028/iajit/18/3/.
- [25] N. Md. J. Basha, G. Ganapathy, and M. Moulana, "A preliminary exploration on component based software engineering," *IJCSNS International Journal of Computer Science and Network Security*, vol. 22, no. 9, 2022, doi: 10.22937/IJCSNS.2022.22.9.22.
- [26] N. Md. J. Basha, G. Ganapathy and M. Moulana, "CREA-components reusability evaluation and assessment: an algorithmic perspective," *International Conference on Advanced Informatics for Computing Research*, 2022, pp. 132-142, doi: 10.1007/978-3-031-09469-9\_12.
- [27] N. Md. J. Basha, S. A. Moiz, and A. A. M. Qyser, "Performance analysis of hr portal domain components extraction," *International Journal of Computer Science & Information Technologies (IJCSIT)*, vol. 2, no. 5, pp. 2326-2331, 2011, doi: 10.48550/arXiv.1203.1328.




- [28] Y. Meng, X. Wu and Y. Ding, "Research and design on product quality tracking system based on domain engineering," *2010 Third International Symposium on Information Processing*, Qingdao, China, 2010, pp. 572-575, doi: 10.1109/ISIP.2010.24.
- [29] W. Frakes, R. Prieto-Díaz and C. Fox, "DARE-COTS. A domain analysis support tool," *Proceedings 17th International Conference of the Chilean Computer Science Society*, Valparaiso, Chile, 1997, pp. 73-77, doi: 10.1109/SCCC.1997.636929.
- [30] F. Loiret, A. Plšek, P. Merle, L. Seinturier and M. Malohlava, "Constructing domain-specific component frameworks through architecture refinement," *2009 35th Euromicro Conference on Software Engineering and Advanced Applications*, Patras, Greece, 2009, pp. 375-382, doi: 10.1109/SEAA.2009.24.

## BIOGRAPHIES OF AUTHORS






**N Md Jubair Basha**    received his B.Tech. (IT) and M. Tech (IT) from JNTUH, Hyderabad. Currently he is the part-time research scholar at Bharathidasan University. He is presently working as Associate Professor in Computer Science & Engineering Department Kallam Haranadhareddy Institute of Technology, Guntur, AP, India. He has 18 years of teaching experience. He had authored many research papers in various national/international conferences and international journals. His research interests include software reusability, component-based software development, empirical software engineering. He had also published 2 patents. He received many Faculty Excellence Awards. He is a senior member of ACM and Life member of Computer Society of India. He can be contacted at email: jubairbasha@gmail.com.



**Prof. Gopinath Ganapathy**    Ph.D. has 35 years of total experience in academia, industry, research, and consultancy services. He is currently the Registrar of Bharathidasan University, India. He has around 8 years international experience in the U.S and U.K. He served as a consultant for a few fortune500 companies that include IBM, Lucent-Bell Labs, and Merrill LynchToyota. He specialized in designing and architecting multi-tier and EAI technologies. He is a Professional Member in IEEE, ACM, and IAENG. He is a Life Member in Indian Science Congress, Indian Society for Technical Education, and Computer Society of India. He was earlier Chair, School of Computer Science and Engineering, the Director, Technology Park, Bharathidasan University. He can be contacted at email: gganapathy@gmail.com.



**Dr Moulana Mohammed**    received his Ph.D. in Computer Science from Bharathiar University in 2018 and M. Tech in CSE from JNTUK in 2009. He is a Professor in Department of Computer Science and Engineering, K L University. He has 16 years of teaching years of teaching experience. His research areas include data mining, data science, bioinformatics, IoT, and big data analytics. He can be contacted at email: moulanaphd@gmail.com.