A Novel Approach for Enhancement of Scalability and Dependency Elimination in Legacy Monolithic Systems by Innovating Metrics Based Clustering

Abdul Razzaq
Information Science & Technology
Zhejiang University
Zhoushan, China

Muhammad Waseem
Computer Science & Software Engineering
Wuhan University
Wuhan, China

Abstract
Scalability is one of the concerns highlighted in recent literature and is directly related to issues that are encountered in state-of-the-art technology. Architecture and system application's scalability is also an emerging challenge as system applications cannot be scaled up towards high-level dependencies of components. Similarly, with the current methodologies, one is unable to decompose the tightly coupled systems and scale them accordingly. This research work focuses on lightweight independent component-based system applications scalability enhancement. This study is addressing the tightly coupled concerns and issues in the context of system scalability. This effort contributes to research knowledge enhancement by proposing a new model to identify the system's components that are dependent and independent exclusively. The proposed solution aims to remove the dependency on components of the system under investigation. The research presents the validation proofs using experimental data presented results clearly show the effectiveness of the proposed over current established literature. The proposed solution guides the developer and system architect to effectively identify the dependent and independent components. This research effort also establishes a precise method for removing the dependency of dependent components and increases the scalability of the resulting application.

Keywords:
Monolithic Architecture, Systems Components, Traditional Application Development, Dependency, Scalability of Legacy Systems

1. INTRODUCTION

Featured of Method: This research is an effort to fulfilling the identifying gap in the literature, which is based on academic and industries work. It is useful for industrial work for scaling the developed applications. The previously developed system applications can scale by identifying system components in both the context of dependent and independent components for system application. It is also useful for those system applications which have no documentation and do not know what components are, how can a system be scaled further. It helps to achieve high cohesion and low coupling of system application's components.

1.1 Components

Components play a significant role in reducing the scalability issue of a system application [1]. Components are simply the smallest self-managing, autonomous, and helpful sets of a system that works in various environments. Components are regularly circulated objects consolidating propelled self-administration features [2]. The components are utilized in these essential programming applications are comprised of central squares that can be joined together, relying on the prerequisite [3]. Components increase the productivity of application developers and improve by and large programming quality because of the high level of reusability [4]. It is vital for an application developer to most likely alter parts since it is uncommon to discover the components coordinating their functional and non-functional requirements in the new applications [4]. Software engineering is favored attributable to the reasons. It can address every one of the worries beginning from the prerequisite of the undertaking. The activities to be performed by the product of a specific business for which it is planned [30]. Additionally, it can likewise guarantee the duty regarding individual groups. Software development based on components has emerged as a successful way to deal with building an adaptable system. Components based work has risen as a compelling way to deal with complex programming system [28]; its advantages incorporate decreased improvement costs through reusing off-the-shelf components and expanded flexibility through including, evacuating, or replacing Components [5], [6].

In the event, the applications are to be kept running in the cloud with effectiveness, and it requires substantially more expertise than what is essential to convey any programming in virtual machines. It is continuously prescribed to oversee cloud applications consistently to use their assets as per the approaching burden and to confront the disappointments, to duplicate and rehash every one of the segments to give flexibility if there should arise an occurrence of the inconsistent framework [7]. When a program or programming is

Manuscript received December 5, 2025
Manuscript revised December 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.12.7

planned to keep in view every one of the prerequisites, it turns out to be very extreme for the software architect to present radical changes which are later on requested by plans of action or client as often as possible since it turns out to be progressively entangled for the designer to make changes when the code begins extending as a result of the inclusion of various individuals or master who make changes in the product [8]. As increasingly more exertion is required to facilitate for updating in a tightly coupled model of monolithic design, this entire procedure, at last, makes the discharge cycle of the application moderate [9]. It likewise makes the model delicate and untrustworthy. Versatility is, likewise, an essential element that requires the task and advancement of large enterprise applications [10].

The major downside of the monolithic application is its lack of scalability when a specific errand is to be executed inside the components [11]. Long software cycle in light of the multifaceted nature of framework is likewise an obstacle in current, dependable administrations. In this strategy, figuring out likewise delivers wanted outcomes. The method utilized with the end goal of figuring out is bunching, which is considered the least essential and challenging procedure utilized in building and science [3]. The primary and most essential target of executing this system is to mention the objective facts clearer to build up a superior comprehension. This better understanding makes it simple to create complex information structures from given highlights. Grouping strategy or technique is commonly liked to distinguish all the related segments of System Software Application alongside their duties. As the info utilized in this procedure features the interconnectivity of every one of these parts, this grouping strategy is beneficial to limit the interconnection among various components to create ideal outcomes.

1.2 Clustering

In this proposed method, reverse engineering is also used to produces the wanted outcomes. This method utilized with the end goal of reverse engineering that is considered the least complicated, and the primary method utilized in engineering and science [3], [12]. The primary and most imperative target of implementing this method is to mention the objective facts clearer to build up a superior understanding. This awareness makes it simple to create a sophisticated learning structure from given highlights. Bunching strategy or technique is commonly liked to recognize all the related parts of System Software Application alongside their obligations [13]. As the information utilized in this strategic feature the interconnectivity of every one of these components, this metric based clustering method is very valuable to limit the interconnection among various parts to create ideal outcomes [29], [14]. Clustering is a procedure in which huge frameworks are decayed into pieces and littler. This sensible framework unmistakably that the substances which bear closeness with each other have a place with a similar subsystem, while the elements with a contrast among each other are ordered into various subsystems [3]. The clustering procedure is commonly utilized in distinguishing the product parts. It is tremendous that receives one measurement with the goal that the closeness of segments might be estimated. The principle focal points of this procedure are that low coupling and high union of parts are accomplished [7]. These points of interest assume essential job to take care of the issues which require software evolution [15].

A software developer or programmer with vast experiences are highlighted two kinds of issues in practice. The first issue is inserted in the way that it is challenging to decide an explicit cluster, which is utilized for profoundly coupled parts [14], [16]. The second issue in line is to decide the bunch mapping, which connects to software modules. Upon examination, the method of decomposition of application has ensured that the source code of system application is as per every one of the prerequisites accumulated [17].

1.3 System Application Challenges

In this sort of way to deal with discover the solution, software architecture, open-source advancement, authoritative structure, and obligation are vertically disintegrated [15]. During the time spent on software development, a complex system is challenging to be architected expertly. So, architecture refinement plays a critical job in the description of software architecture slowly [18]. In a stepwise refinement, a succession of steps beginning from a unique detail of the design prompts a reliable, execution-focused, and building model [19]. As far as a component, the conceptual architecture could be refined to a progressively robust design by sets. After parts, the architecture comprises of two parts and a connector. The reality of software architecture representation gives numerous points of interest amid all periods of the programming life cycle [20]. By and by, for some, systems, like inheritance ones, there is no available representation of their architectures [21, 27].

The interface of a component should be the essential concern of its designer or developer. Since the components are intended for use in an assortment of systems and need to give the administrations paying little heed to the setting, designers endeavoring to utilize a segment must almost certainly distinguish the capacity of a component and the methods for invoking this behavior [22].

Monolithic applications come up with failure when the number of clients getting to a system becomes excessively high or when too many features are integrated into a single system [9]. Component Architecture gives way to software engineers to deal with the multifaceted nature of vast scale logical recreations and to push toward a fitting and-play condition for elite figuring [23].

Some vast, and along expensive, software systems work in a constant domain under requesting execution prerequisites. A few investigations show that a large portion of the expense for an item is spent on support [24]. Numerous advantages can be picked up by partitioning a system into components. Additionally, Maintainability and scalability are accomplished [31].

1.4 Metrics

Metrics is the way to collect the set of data into the related group. In this research, the metrics play an essential role in the components of a system application [25], [26]. Using metrics, we made the different clusters for all related components of the system. We found all relations of all that system components which are depended and in-depended.

2. Research Problem

2.1 Problem Statement

After the evaluation of relevant research, it has been established that dependency removal of the component to increase the scalability of the system is an area where a serious effort is required to bridge the identified gaps in the literature. Below are the research questions this work answers when viewed in the predefined context.

2.2 Research Questions:

1. What techniques/methods are reported in the literature to identify the components of the developed system's applications and remove the dependency of applications' components?
2. Why is it necessary to find and remove the dependency of the application's components?
3. How to resolve the dependency issue of application components to increase the scalability of the system?
4. What is the impact of the novel proposed when seen in context to state of the art?

No.	Discussed
1	We did not find any method or technique to find the components of developed systems and to remove the dependency of dependent components.
2	The scalability is the primary quality attribute to achieve it by using the proposed method to remove the dependency of the application's components.
3	We developed the metrics-based clustering method to identify the dependent and independent components of the applications and added the few new lines of code to remove the dependency of dependent components.
4	We identified the conceptual architecture of system application and identified its relations among all components of the application. It's abstract level architecture.

3. Solution Model

3.1 Solution Brief

In this proposed method, we used different techniques. This method is divided into different categories. In first, we extract the abstract level solution model for understanding the method flow and its different sections, figure 1. It is an abstract level model design. After this, we made the detailed solution design model with different - sub-phases, which show the holistic flow of the solution design that how it works, Figure 2.

3.2 Project Brief

System application, it was a .net project. There were two projects, and this first one is large, and the second is small as a pilot study. This technique applied to both projects. We measured the results, what type of impact on the large and small projects through this proposed method. In this work, we used the proposed method on classes and methods/functions to find the similarity and dependency of each class and method/function, which helped to identify the components of the system application. To identify the system application's components, the researcher has made two sets of categories of components, the first one is Independent components, and the second is Dependent components. How to verify dependent components being mentioned in section 4 below. Finally, the researcher removed the dependency of components and scaled the system application by adding a new component.

This project was at a medium level. The project's development was in the MVC .NET Framework. The duration of this project was four months with two developers. In this project, the Lines of Code is 7,944, and a total of 330 classes consist of the depth of inheritance: further detail and significant parts of the project mentioned in this given Table 1.

***TABLE 1**. PROJECT CODE DETAIL*

Parts (Operations)	Class Coupling	LOC
Areas		
Admin Section	7	29
Admin Controller	82	1117
Admin Models	95	1906
Approval	4	4

Approval Controllers	57	912
Budget	4	4
Budget Controllers	2	7
Budget Models	17	71
Request	4	4
Request Controllers	29	109
Request Models	24	217
User Management	4	4
User Management Controllers	117	1638
User Management Models	70	807
Vendor	4	4
Vendor Controllers	35	170
Vendor Models	31	213

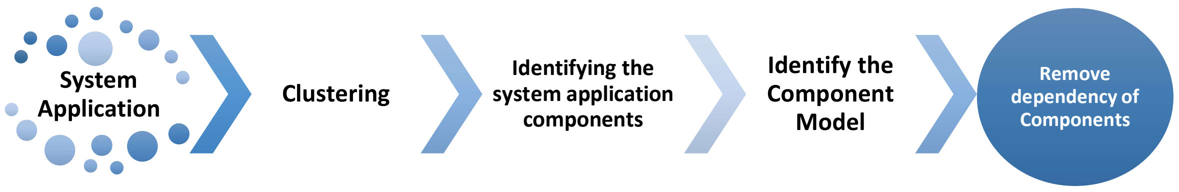
System Application
Clustering
Identifying the system application components
Identify the Component Model
Remove dependency of Components

Figure 1. Proposed Solution Abstract Model

In Figure 1, it expresses the procedures at the abstract level of our proposed method and its flow. In this proposed method has five processes. The first process is a project. Projects must have classes or methods/functions. The second process is the clustering technique, which we used with metrices to make our method. The third process has identified components by applying our proposed method. The fourth process gives us an abstract level architecture model of components relations. In the last process, we have identified both types of components in the context of dependent and independent. We add a few lines of code to remove its dependency on dependent components where it is necessary.

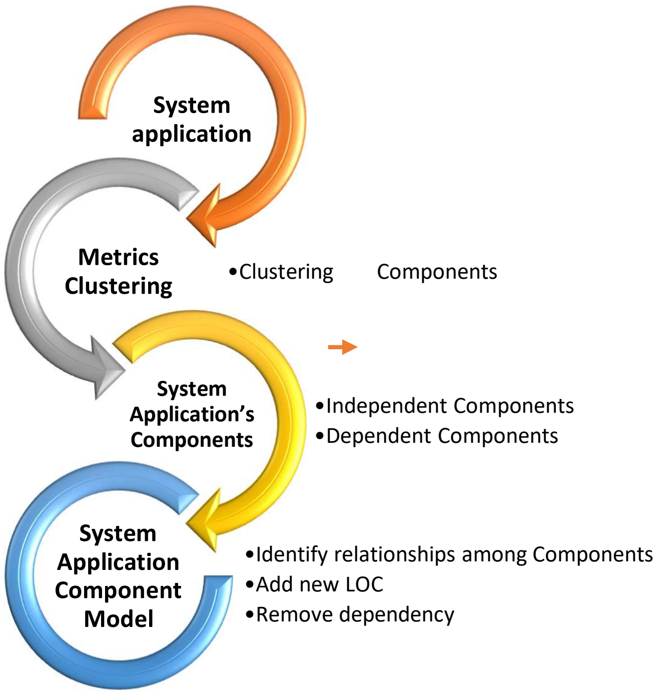

Figure 2. Holistic Proposed Solution Model

1. In the first process, it’s a system application that can be web application or desktop software and can be written in any programing language.
2. The second process is based on our proposed method, which we designed using the clustering technique. This proposed method is based on metrices. We used clustering techniques in our method just for making groups or related methods that are being used in system applications. We find the code methods/functions and insert each method/function in the metric also writes the value of each method that how many times this method/function is used in other methods. Also, we check and write the accessibility of each method for another method. We count the number of usage values by applying intersection and write down the exact values in the metric that how many times it is being used in another method.
3. The third process is a system application’s components. In this process, we find components based on our metrices, we check the relations and make groups by using clustering technique and make them separate into further groups. Each group expresses its related component. We measure the relationships among methods/functions and find the relation among components that show the dependency among components and find the independent components.
4. The last fourth process is a components architecture model of system application. To reach this process, we have identified dependent and independent components of system application and relationships among all components. We see the complexity of the methods and complexity of components’ relationships to make it independent. We make new classes or write code for methods to make it independent. We make the abstract level architecture model of components by using their relationships. It is a high-level architecture model.

3.3 LCOM

Lack of Cohesion of Methods (LCOM): The single duty rule expresses that a class ought not to have more than one motivation to change. Such a class is said to be durable. A high LCOM esteem, for the most part, pinpoints an ineffectively durable class.

• Cohesion is maximal: all methods access variables LCOM = 0
• No cohesion: every method accesses a unique variable LCOM = 1

4. Research Methodology

The research method comprises of the following components in order:

• Critical literature evaluation and analysis
• Proposed a method for proof of concept is built in order to validate results.
• Case study results and analysis of data for answering the raised research questions
• Concluding the research based on analyzed data and own interpretive deductions

4.1 Rationale

The researcher undertook the study to find the dependent and independent components of the system’s application. After finding all the components of the system’s application, the researcher removes the dependency of dependent components and classes by using the proposed model with metrics based clustering. The system’s application can scale by using this proposed method.

Type of study: The contextual investigation is similar nature that will be utilized to study. The comparative study is conducted to find out the impact of a large and small project in order to remove the dependency of components.
Study analysis: we measured and compared the results of the project in order to reusability of components.
Case Study Context: Proposing a new technique is the primary context of the case.

Expected result: by using the newly proposed method for increasing the scalability of system application is completed successfully.

5. Results & Analysis

System application, using the proposed method, the experiments are performed on the industry-based project, and then results from as is measured that found the impact. We performed the clustering techniques with metrices on components to find the similarity and dependency of each method, which are used in the system application. Based on the proposed method, we identify the system application’s components in two different categories of components, the first one is independent components, and the second is dependent components. Independent components easily scaled and reused without any risk, but dependent components cannot be easily scaled or upgrade in the system application, so it was needed to make the dependent components into independent. We also identified the abstract level architecture of system application; once we identified all the system's components, we created the architecture of system application based on components' relations and method's logic figure 7.

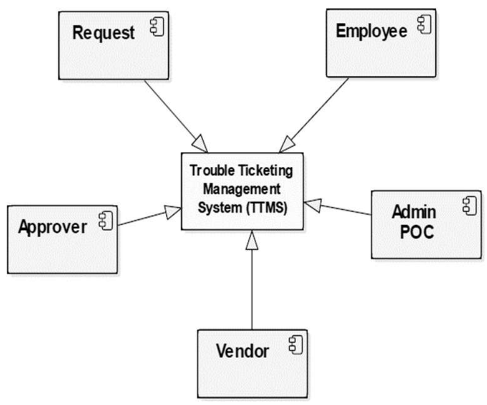

Figure 3. TTMS Components Abstract Architecture Model

Figure 3 shows all the components used in this project. These components are fully functional in the system. We designed all components models in Enterprise Architect Tool. We used the standard rules, the "provider," and the "required" interfaces. Figure 3 is discussed below.

Request: it has the functionality to execute the request by Employee, Admin POC, or Approver. Requests can be generated under different conditions like a priority, category, location wise.

Employee: it creates the request and checks the progress of its requests also email generated on the biases of each request's action. An employee can communicate directly to its related POC based on the request. An employee can only create a request based on an auto detected location and authenticated by active directory (AD).

Admin POC: it can create a request for itself also on behalf of the employee. It has the capability and roles to make the request on behalf of employees who are under the POC. Once a request is created on behalf of an employee, an email is sent to an employee about its update. POC also has limited rights based on locations and categories rights. POC can only create or proceed with the request, which is under the categories and locations' rights. POC proceeds the request that is created by the employee or itself. POC can entertain any request which comes under its roles. POC can reject the request made by the employee and send a comment about rejection also can communicate for further information about the request. After completion, the all proceeding of request and return from the vendor, the POC verify the request and generate the invoice according to verification. All detail of the invoice automatically fetched from the database, which has been updated by the vendor. POC can adjust the amount if something is missed by the vendor and let them know. This adjustment history is maintained with previous and new records. POC can generate reports in detail or summary and with different filters to see like categories, locations, and dates.

Approver: it can proceed with the new request by itself. The approver can proceed with the request, which is forwarded by POC for approval. Approver can entertain any request to submit it to the vendor, which comes under its roles and amount limit. Approver can communicate with POC related to request or also can reject the request back to POC with reasons. An approver can check the request's detail. Approver has also roles to see the reports.

Vendor: The vendor must deliver the required items in requests. If a vendor does not have something or missed so a vendor can update about the missing items in request. The vendor can communicate with POC if the vendor needs more detail about the request's items. The vendor can respond to the request and can check its progress. Once the job is done and delivered all the items of request, the vendor closed the request as a completed. If some items are delivered of request, the vendor can update the request's status as a partial and can close it.

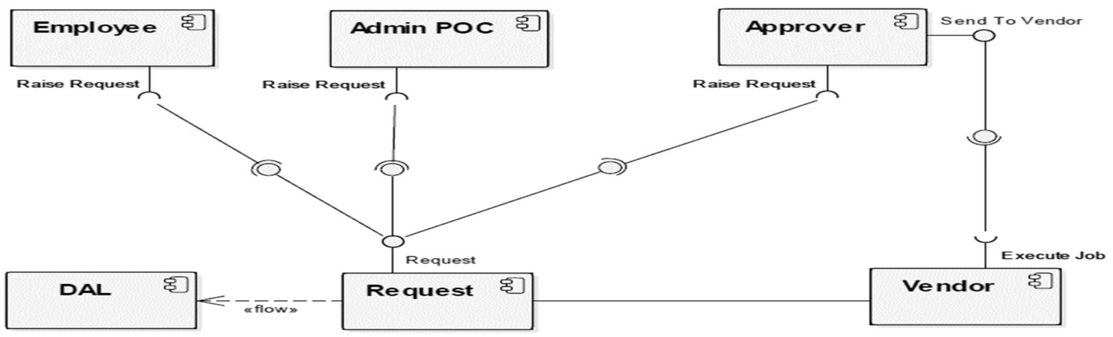

Figure 4. TTMS Components Relation Architecture Design with Required & Provider interface.

Figure 4 shows the component's interface, required, and provider to each component that is providing the interface, and which must be required an interface. This model shows how this system works and how the components are interacting with each other. Figure 4 is the dependent component model before applying the proposed method. This model shows that all components Employee, Admin POC, and Approver are dependent on one component of Request.

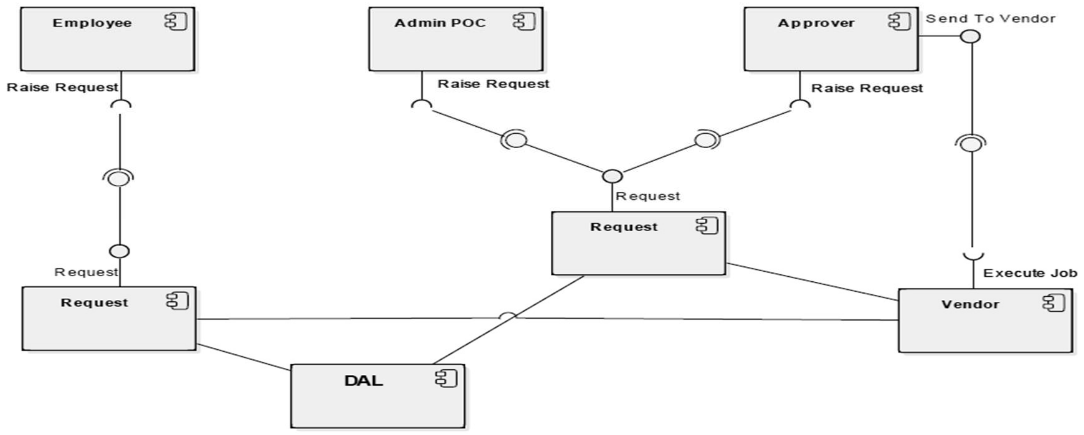

Figure 5. Independent Components Model

Figure 5 shows the independency of the requester component among all components. This components model is created after applying our proposed model, which clearly shows the elimination of components dependency. Employee, Admin POC, and Approver components were dependent on the Request component before, as we have mentioned in figure 4. Now, after removing the dependency, the employee component is separated from another Request component.

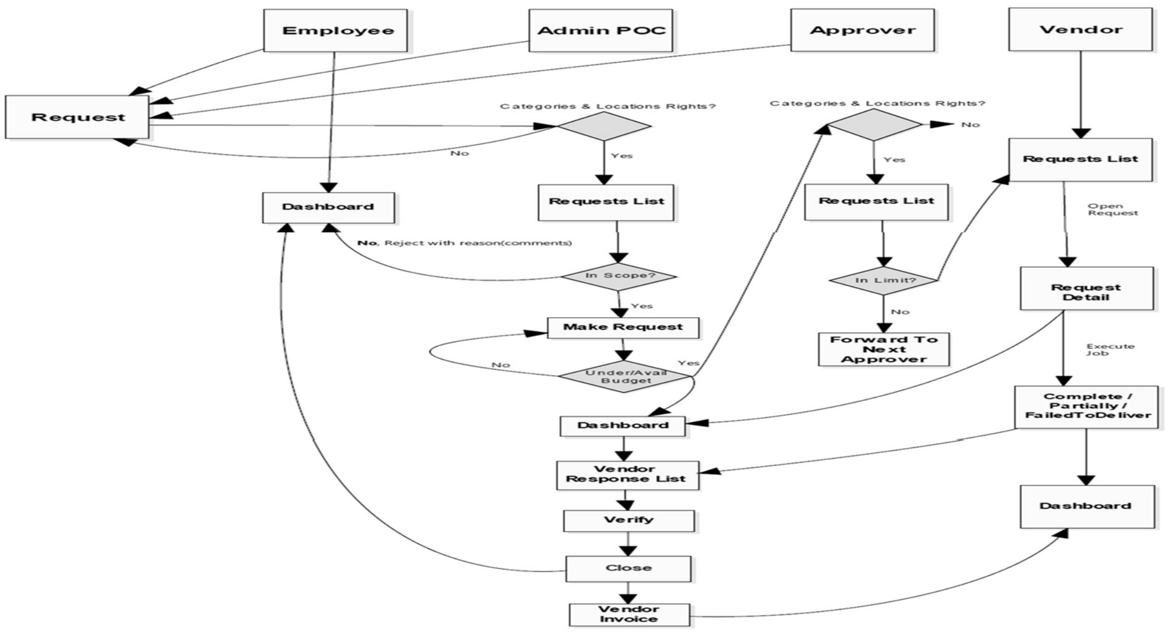

Figure. 6. TTMS Flow Diagram with dependent Request Component

This diagram figure 6 shows the flow of application for each component and its roles. This flow diagram clearly shows the dependency of the component with other system's application components. Each component Employee, Admin POC, and Approver create a ticket by calling the Request component, and then this ticket proceeds further. The rest of the flow is simple and discussed in detail above. After removing the dependency, the system's application components are independent.

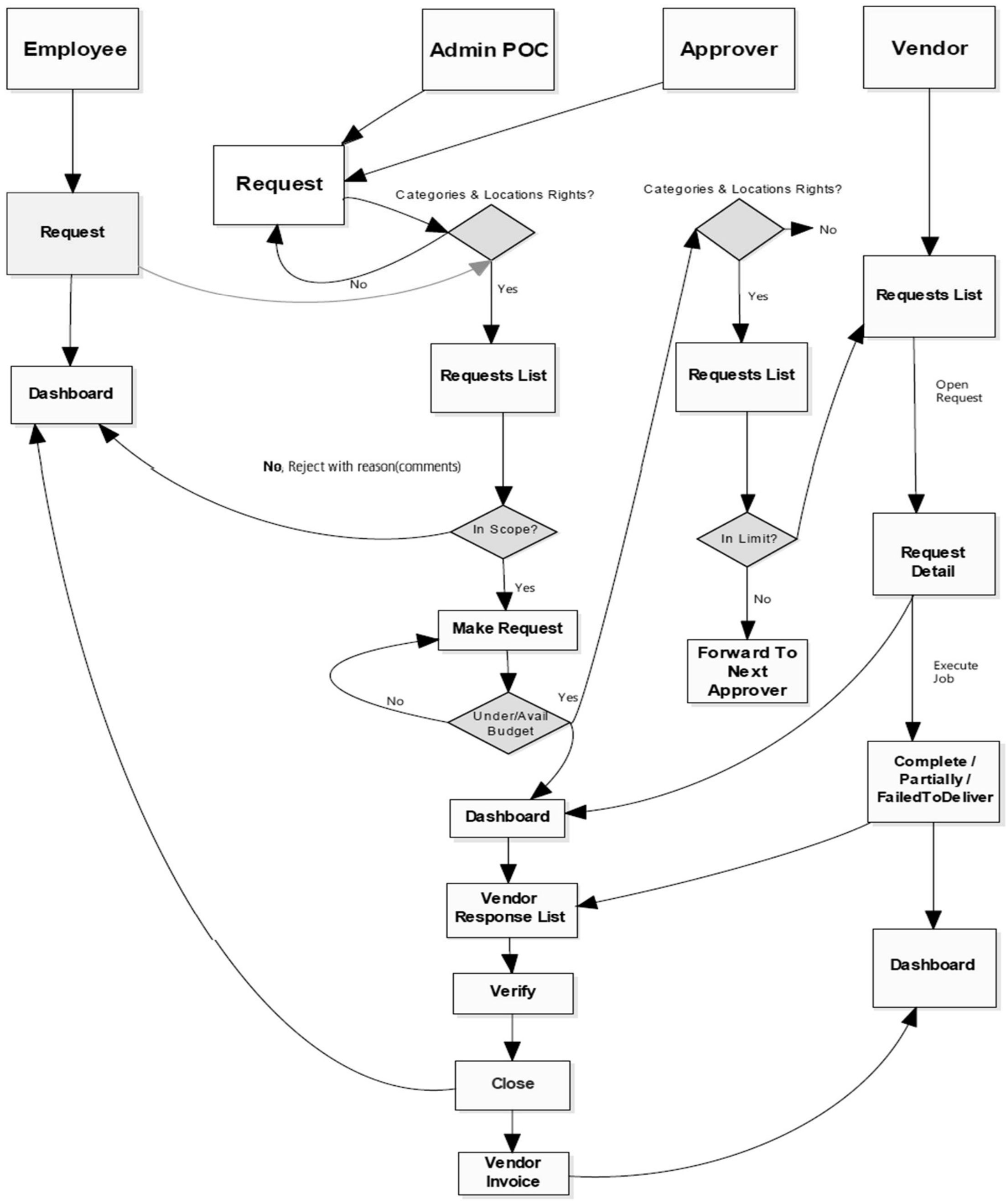

Figure. 7. TTMS Flow Diagram with independent Request Component

Figure 7 shows the flow with the independent component procedure, and the employee requester component is working independently. This flow of components Employee, Admin POC, and Approver were dependent before, as is mentioned in figure 6. Now the Employee component's ticket is being forwarded separately using separate Request component. Table 2 shows in a quantitative form that no method is dependent on others.

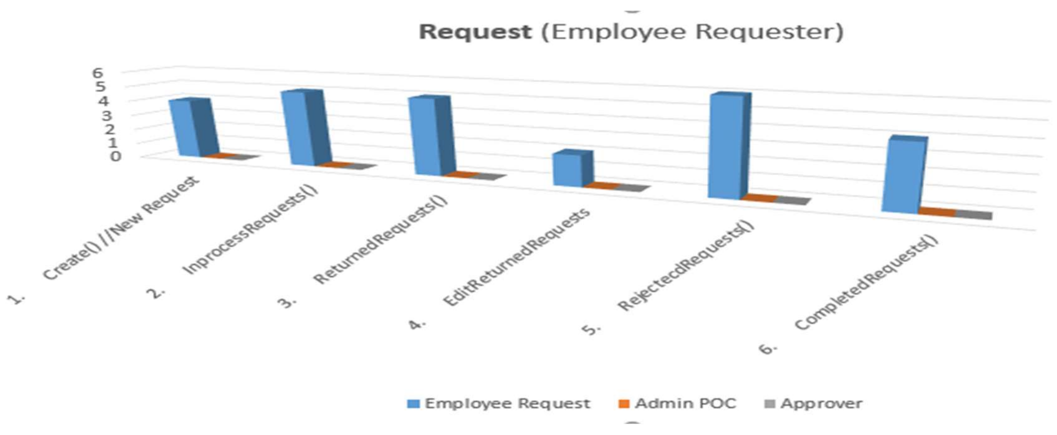

Fig. 8. Employee Request Component

Figure 8 shows the dependency, among other methods, and other components are bonded with it.

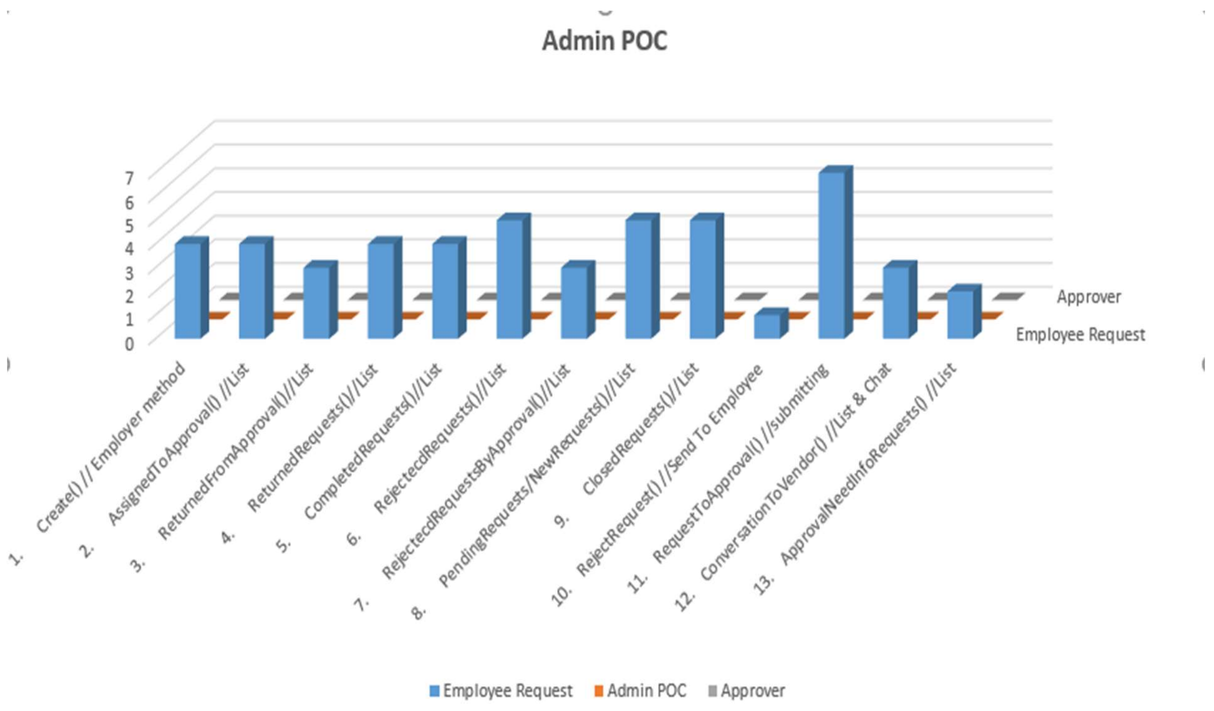

Fig. 9. Admin POC Dependent Component

Figure 9 shows the dependency of all methods with request component. This component depends on the requester component to create the request. We removed the dependency of the Admin POC component that was dependent on the Request component is shown in figure 14.

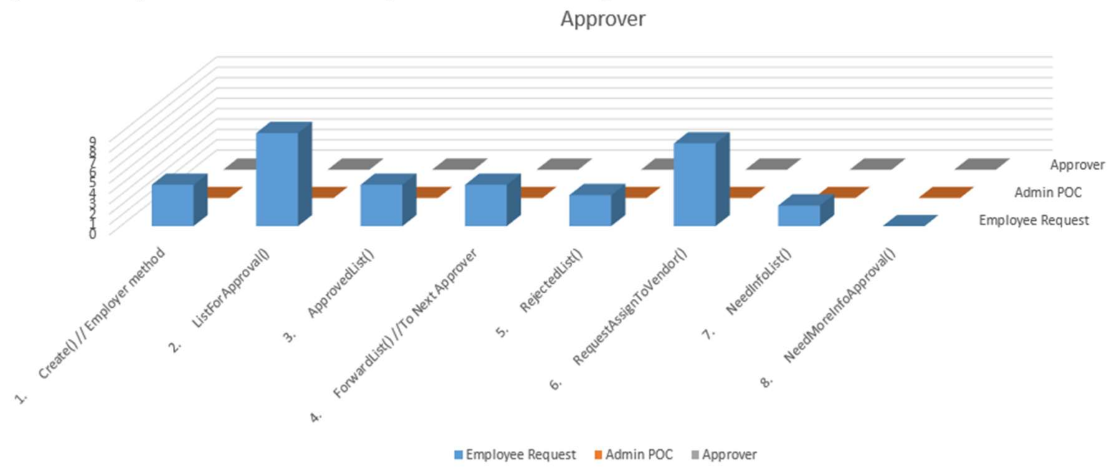

Figure 10. Approver Dependent Component

Figure 10 shows the dependency of all methods with request component and Recreate one more class of request model: but request creating component is the same being used. After removing the dependency of the Approver's component is mentioned in figure 15.
Table 2 shows the dependency among all components. Methods of all components are mentioned in table 2 and show the coupling among all methods. Table 2 shows the coupling of components' methods. After applying our proposed method, we removed dependency among methods and components, removing dependency in methods and components, the results are clearly mentioned in table 3.

TABLE 2. DEPENDENT COMPONENTS' METHODS

Components	Methods	**Employee Requester Area (RequestModel)**	**Admin POC (RequestModel)**	**Approver**
Request (Employee Requester)	1. Create() //New Request	4	0	0
	2. InprocessRequests()	5	0	0
	3. ReturnedRequests()	5	0	0
	4. EditReturnedRequests	2	0	0
	5. RejectecdRequests()	6	0	0
	6. CompletedRequests()	4	0	0
Admin POC	7. Create() // Employer method	4	0	0
	8. AssignedToApproval() //List	4	0	0
	9. ReturnedFromApproval()//List	3	0	0
	10. ReturnedRequests()//List	4	0	0
	11. CompletedRequests()//List	4	0	0
	12. RejectecdRequests()//List	5	0	0
	13. RejectecdRequestsByApproval()//List	3	0	0
	14. PendingRequests/NewRequests()//List	5	0	0
	15. ClosedRequests()//List	5	0	0
	16. RejectRequest() //Send To Employee	1	0	0
	17. RequestToApproval() //submitting	7	0	0
	18. ConversationToVendor() //List & Chat	3	0	0
	19. ApprovalNeedInfoRequests() //List	2	0	0
Approver	20. Create() // Employer method	4	0	0
	21. ListForApproval()	9	0	0
	22. ApprovedList()	4	0	0

	23. ForwardList() //To Next Approver	4	0	0
	24. RejectedList()	3	0	0
	25. RequestAssignToVendor()	8	0	0
	26. NeedInfoList()	2	0	0
	27. NeedMoreInfoApproval()	0	0	0

TABLE 3. INDEPDENDENCT COMPONENTS' METHODS

Components	Methods	**Employee Requester Area (RequestModel)**	**Admin POC (RequestModel)**	**Approver**
Request (Employee Requester)	28. Create() //New Request	4	0	0
	29. InprocessRequests()	5	0	0
	30. ReturnedRequests()	5	0	0
	31. EditReturnedRequests	2	0	0
	32. RejectecdRequests()	6	0	0
	33. CompletedRequests()	4	0	0
Admin POC	34. Create() // Employer method	0	4	0
	35. AssignedToApproval() //List	0	4	0
	36. ReturnedFromApproval()//List	0	3	0
	37. ReturnedRequests()//List	0	4	0
	38. CompletedRequests()//List	0	4	0
	39. RejectecdRequests()//List	0	5	0
	40. RejectecdRequestsByApproval()//List	0	3	0
	41. PendingRequests/NewRequests()//List	0	5	0
	42. ClosedRequests()//List	0	5	0
	43. RejectRequest() //Send To Employee	0	1	0
	44. RequestToApproval() //submitting	0	7	0
	45. ConversationToVendor() //List & Chat	0	3	0
	46. ApprovalNeedInfoRequests() //List	0	2	0
Approver	47. Create() // Employer method	0	4	0
	48. ListForApproval()	0	9	0
	49. ApprovedList()	0	4	0
	50. ForwardList() //To Next Approver	0	4	0
	51. RejectedList()	0	3	0
	52. RequestAssignToVendor()	0	8	0
	53. NeedInfoList()	0	2	0
	54. NeedMoreInfoApproval()	0	0	0

Table 4 provides information about the dependency of all components in a qualitative form. As it shows that Employee (Request) and Admin both are entirely dependent on each other, but Approver's method is dependent only on the "Create()" method. After removing dependency among the components, and table 5 shows the independent components' results. Now the system's components are independent. Table 5 shows the main result of independent components that were dependent before, as we have discussed and mentioned table 2. After applying our method, we found this result successfully.

TABLE 4. DEPENDENT COMPONENTS OF SYSTEM'S APPLICATION

Components	Methods		Employee	Admin POC	Approver
Employee	Create() //New Request		Yes	Yes	Yes
	InprocessRequests()		Yes	Yes	No
Admin POC	ReturnedRequests()		Yes	Yes	No
	EditReturnedRequests		Yes	Yes	No
Approver	RejectecdRequests()		Yes	Yes	No
	CompletedRequests()		Yes	Yes	No

TABLE 5. INDEPENDENT COMPONENTS OF THE SYSTEM'S APPLICATION

Components	Methods		Employee	Admin POC	Approver
Employee	Create() //New Request		Yes	No	No
	InprocessRequests()		Yes	No	No
Admin POC	ReturnedRequests()		Yes	No	No
	EditReturnedRequests		Yes	No	No
Approver	RejectecdRequests()		Yes	No	No
	CompletedRequests()		Yes	No	No

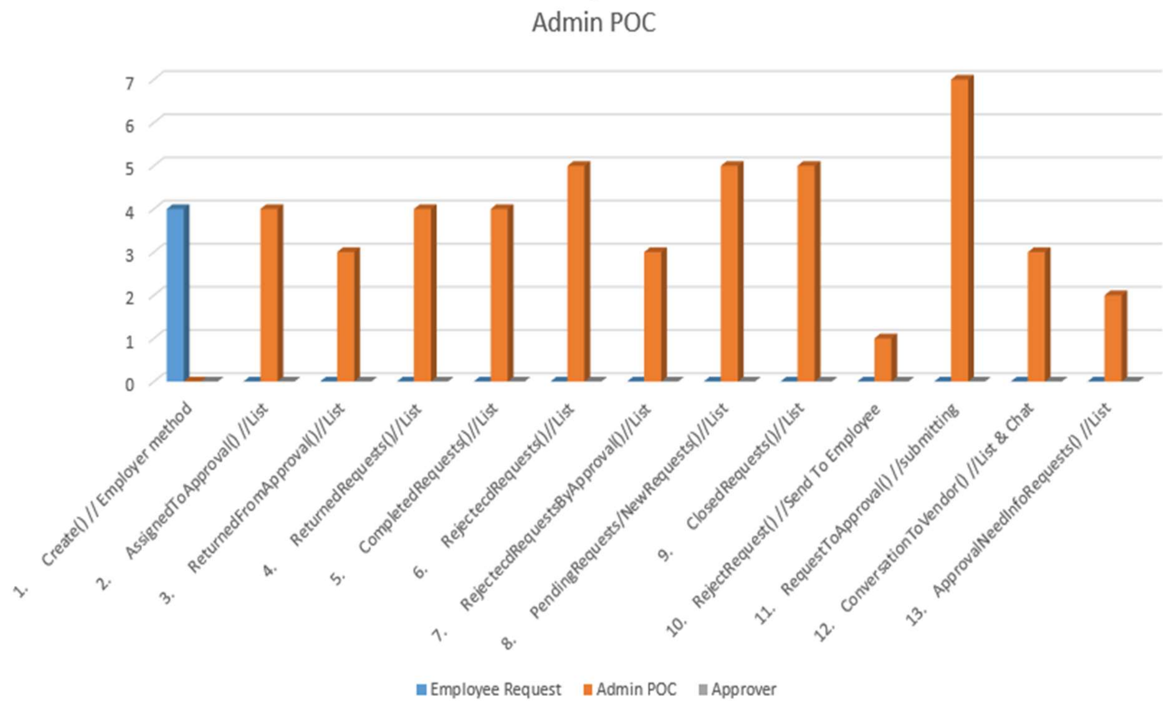

Figure 11. Admin POC Component dependent with Request creator method

Figure 11 shows the dependency with a requester component. After removing the dependency of this component, it is only dependent on a method of requester component. This graph is based on table 2. Figure 14 of this component is showing dependency free.

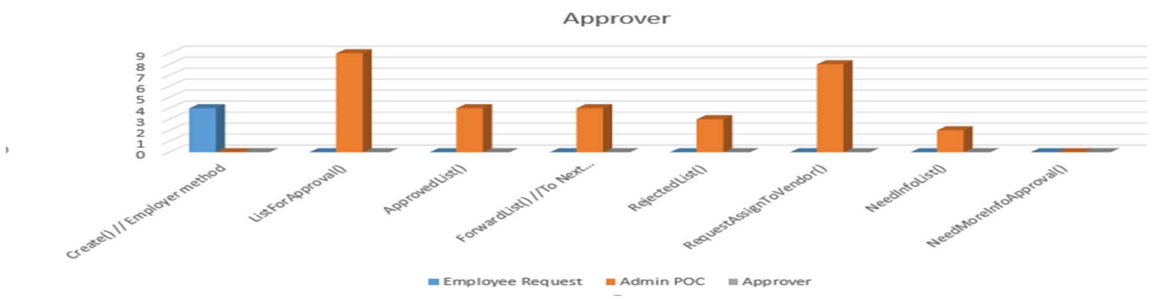

Figure 12. Approver Component dependent with Request creator method

Figure 12 shows the dependency on the request component. The create() method is dependent on both components. This component is also dependency free in figure 15.

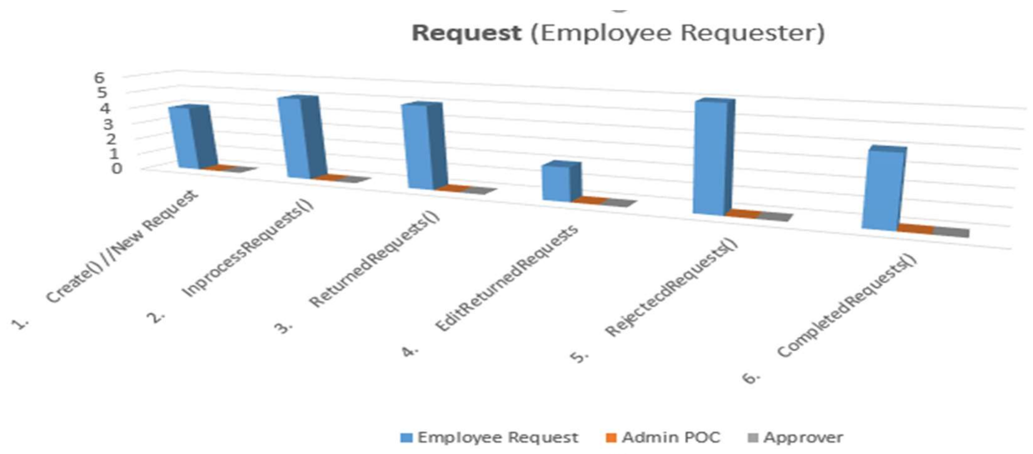

Figure 13. Employee Request Independent Component

Figure 13 shows the independent components which are independent of the Admin POC and Approver.

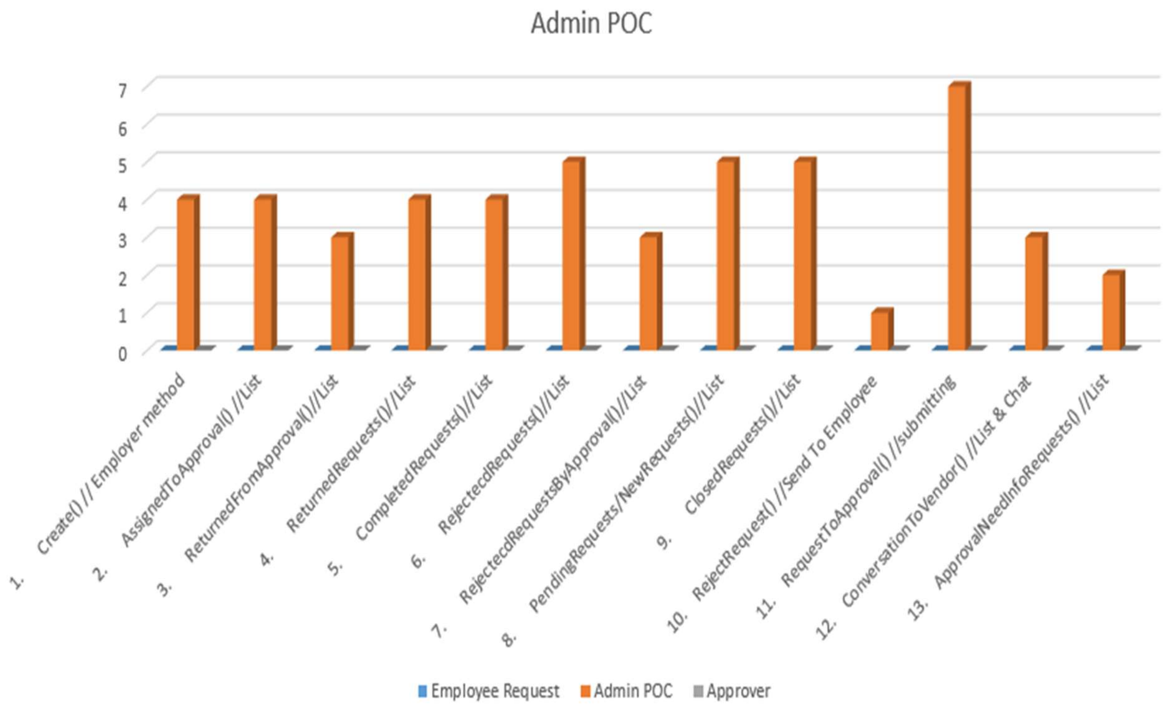

Figure 14. Admin POC Component Independent from Employee Request Component

Figure 14 shows the independent component, which is entirely independent of the Employee Requester Component.

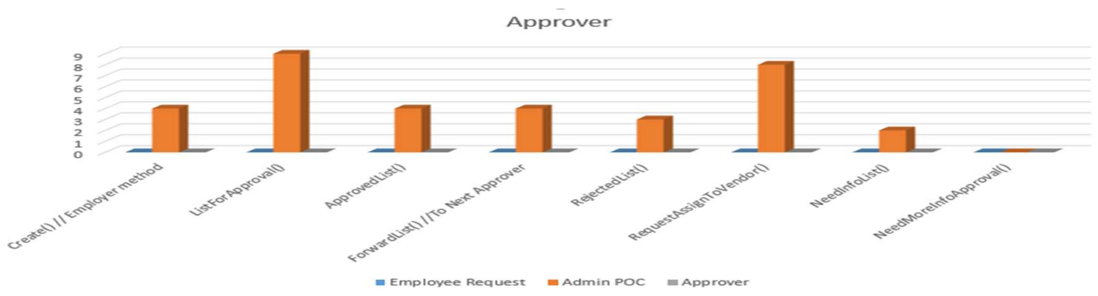

Figure 15. Approver Component dependent from Employee Request Component

Figure 15 shows the independent component, which is entirely independent of the Employee Requester Component, but it is dependent on Admin POC.

6. Previous & Present Work

To the best of our knowledge, we did not find a method/technique in the literature to identify the components of the developed system's applications in the context of dependent and independent components of the system's applications. It is necessary to scale the developed system's applications. By using our proposed method, we identified the components but also found the relations among components and made an abstract level architecture of the system's application.

7. Contribution

We created the method to identify the dependent and independent components of the system's applications. We designed this method based on metrics and clustering techniques. We applied this method in the multinational industrial project and increased the scalability in this project successfully. We identified the components in the context of dependent and independent, then remove the dependency among the dependent components by adding a few lines of code. We also achieve the abstract level architecture of this project.

8. Conclusion

This work presents a novel approach for identifying the dependent and independent components of the system's applications. This research gives awareness to understand and increase the scalability of the system's applications, which are already developed. The favorable primary position of this proposed method is that low coupling and high cohesion of components accomplished. These useful points assume an urgent job to tackle the issues which require software evolution. As literature reports, no effort that proposes any comprehensive technique or framework to find the components with relation and did not find the method to remove the dependency of dependent components. From existing literature, the researcher derived four Research Questions on the bases of these issues regarding the system. In this paper, the significant section of the proposed solution, where the operations executed that provides proof of results in terms of validation with all the aspects of the research results. These experiments performed on the industry-based project.

8. Future Work

In future work, we need to create an automated tool that helps to find the dependent and independent components from the project based on the proposed method.

References

[1] F. Barbier, "Component-based design of large-scale distributed systems," *25th Annual International Computer Software and Applications Conference. COMPSAC 2001*, Chicago, IL, USA, 2001, pp. 19-24.

[2] C. Geisterfer and S. Ghosh, "Software Component Specification: A Study in Perspective of Component Selection and Reuse," *Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS'05)*.

[3] D. Liu, C. Lung and S. Ajila, "Adaptive Clustering Techniques for Software Components and Architecture," 2015 IEEE 39th Annual Computer Software and Applications Conference, 2015.

[4] S. Yau, C. Taweponsomkiat, and D. Huang, "A Framework for Extensible Component Customization for Component-based Software Development," *2006 Sixth International Conference on Quality Software (QSIC'06)*, 2006.

[5] J. Sanchez Cuadrado, E. Guerra and J. de Lara, "A Component Model for Model Transformations," *IEEE Transactions on Software Engineering*, vol. 40, no. 11, pp. 1042-1060, 2014.

[6] N. Parlavantzas, M. Morel, V. Getov, F. Baude, and D. Caromel, "Performance and Scalability of a Component-Based Grid Application," *2007 IEEE International Parallel and Distributed Processing Symposium*, Rome, 2007, pp. 1-8.

[7] Y. Liu, I. Gorton, and A. Fekete, "Design-level performance prediction of component-based applications," in *IEEE Transactions on Software Engineering*, vol. 31, no. 11, pp. 928-941, Nov. 2005.

[8] D. Chaudhari, M. Zulkernine, and K. Weldemariam, "Towards a ranking framework for software components," *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, 2013.

[9] S. Lewandowski "Frameworks for Component-Based Client/Server Computing ACM Computing Surveys" in ACM Press vol. 30 no. 1 pp. 3-27 1998.

[10] Martínez-Ortiz, D. Lizcano, M. Ortega, L. Ruiz, and G. López, "A quality model for web components," Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services - iiWAS '16, 2016.

[11] Matthias Vianden, Horst Lichter, Andreas Steffens. Experience on a Microservice-based Reference Architecture for Measurement Systems, 2014 21st Asia- Pacific Software Engineering Conference. IEEE, 2014.

[12] J. Kaur and P. Tomar, "Validation of Software Component Selection Algorithms based on Clustering," Indian Journal of Science and Technology, vol. 9, no. 45, 2016.

[13] Al Khalid, C. Lung, D. Liu, and S. Ajila, "Software Architecture Decomposition Using Clustering Techniques," 2013 IEEE 37th Annual Computer Software and Applications Conference, 2013.

[14] Suresh Marru, Marlon Pierce. Apache Airavata as a Laboratory: Architecture and Case Study for Component-Based Gateway Middleware. ACM, 2015.

[15] Hussain, I., Khanum, A., Abbasi, A.Q., & Javed, M.Y. A novel approach for software architecture recovery using particle swarm optimization. *Int. Arab J. Inf. Technol., 12*, 32-41. 2015.

[16] Crnkovic, I., Sentilles, S., Vulgarakis, A., & Chaudron, A Classification Framework for Software Component Models. IEEE Transactions on Software Engineering, 37(5), 593–615. 2011.

[17] Giovanni Toffetti, Sandro Brunner, Martin Bl ochlinger. An architecture for self-managing microservices. ACM, 2015.

[18] F. Brosch, H. Koziolek, B. Buhnova, and R. Reussner, "Architecture-Based Reliability Prediction with the Palladio Component Model," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1319-1339, 2012.

[19] J. Zhang, X. Ban, Q. Lv, J. Chen and D. Wu, "A component-based method for software architecture refinement," *Proceedings of the 29th Chinese Control Conference*, Beijing, 2010, pp. 4251-4256.

[20] K. Sartipi, "Software architecture recovery based on pattern matching," International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.

[21] S. Chardigny, A. Serial, M. Oussalah, and D. Tamzalit, "Extraction of Component-Based Architecture from Object-Oriented Systems," *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, Vancouver, BC, 2008, pp. 285-288.

[22] W. Chengjun, "Architecture Driven Component Development for Top-Down Software Reuse," *2008 International Conference on Computer Science and Software Engineering*, 2008.

[23] D.E. Bernholdt B.A. Allan R. Armstrong F. Bertrand K. Chiu et al. "A Component Architecture for High-Performance Scientific Computing" ACTS Collection special issue Intl. J. High-Perf. Computing Applications 20, 2006.

[24] H. Algestam M. Offesson L. Lundberg "Using Components to Increase Maintainability in a Large Telecommunication System" Ninth Asia-Pacific Software Engineering Conference (APSEC02) p. 65 2002.

[25] K. Chahal and H. Singh, "A Metrics Based Approach to Evaluate Design of Software Components," *2008 IEEE International Conference on Global Software Engineering*, 2008.

[26] J. Chen, W. Yeap and S. Bruda, "A Review of Component Coupling Metrics for Component-Based Development," *2009 WRI World Congress on Software Engineering*, 2009.

[27] K. Lau and S. Di Cola, "(Reference) architecture = components + composition (+ variation points)?," *2015 1st International Workshop on Exploring Component-based Techniques for Constructing Reference Architectures (CobRA)*, Montreal, QC, 2015, pp. 1-4.

[28] W. Chengjun, "Architecture Driven Component Development for Top-Down Software Reuse," *2008 International Conference on Computer Science and Software Engineering*, Hubei, 2008, pp. 1349-1352.

[29] Gholam Reza Shahmohammadi, Saeed Jalili. Identification of System Software Components Using Clustering Approach. 2010.

[30] Shahbaz Ahmed Khan Ghayyur, Abdul Razzaq, Saeed Ullah, and Salman Ahmed, "Matrix Clustering-based Migration of System Application to Microservices Architecture" International Journal of Advanced Computer Science and Applications(IJACSA), 9(1), 2018

[31] Pooja Rana, Rajender Singh, "A Soft Computing Approach To Optimize Component-Based Software Complexity Metrics" Journal of Theoretical and Applied Information Technology, JATIT, 2019