ARCHITECTURAL HELPER TOOL TO CONVERT MONOLITHIC SYSTEMS TO MICROSERVICES

Chamika Kasun Bandara

(168205D)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

ARCHITECTURAL HELPER TOOL TO CONVERT MONOLITHIC SYSTEMS TO MICROSERVICES

Balamanage Chamika Kasun Bandara

(168205D)

Thesis submitted in partial fulfillment of the requirements for the degree
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: ……………… Date: ………………..

Name: B.M Chamika Kasun Bandara

The above candidate has carried out research for the Masters Thesis under my supervision.

Name of the supervisor: Dr. Indika Perera

Signature of the supervisor: ………………………… Date: ………………..

Abstract

Today many of the developers and users expect systems to have dynamic user experience on a wide variety of clients including mobile devices. As well as expect to have high scalability and needs to roll out new updated in order to cope with the competitors, even for multiple times a day. But to gain such level of flexibility through the existing monolithic systems is quite tough and hard due to the dependability of the internal modules, components and the services. Recently researchers have discovered a new architecture called microservices architecture where it consists of independent set of services which are focused on one or small set of functionalities of the system and can be deployed as a separated service. By having isolated services, it increases the flexibility on scalability, independent deplorability, maintainability, and reusability.

Recently microservice topic has gain lots of attention from the software industry. If we search on Google or Yahoo we would fine millions of articles, blogs, discussions on social media, and conference presentations. That is because microservices has huge advantage when it comes to development and the evolving of the system. If we are doing a green field project, then there would be not much risk than managing the lots of concurrent and distributed microservices. But if it is going to be a brown field project or trying to convert the existing monolithic systems into microservices there are exist several other risks associated with it, one of major risk is deciding which services to convert into microservices.

In this research trying to find a solution to that problem and mainly focused on the OO Java based monolithic systems. In order to identify the services in the monolithic system`s legacy code we use hierarchical agglomerative clustering algorithm with customized fitness function which helps to identify the candidate microservices for the microservice based architecture, then those will be presented to the users based on the risk value. This tool is capable of providing insights about each of the services by showing risk levels of the obtained services. If the time permits those features will complete with this research in addition to microservices identification. So that people who use this architectural helper tool will get broad understanding of candidate services and they can be reordered by inputting business values to each of those services. Which reduces the risk of converting the existing monolithic system into the relevant microservices.

ACKNOWLEDGEMENT

I am grateful to Dr. Indika Perera for taking my project under his supervision and his willingness to give valuable advices, support and direction. As well as I should be thankful to Sysco Labs' Senior Director of Engineering & Architecture Hasitha Liyanage for helping me to come up with this research idea and giving me valuable points and advices. Not only that I should be thankful to Sysco Labs' Senior Director, Engineering and Architecture Rohana Kumara for his valuable ideas, and comments. I am again thankful to them for providing immense guidance for this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
API	Application Programming Interface
CORBA	Common Object Request Broker Architecture
CoC	Convention over Configuration
COTS	Commercial Off-The-Shelf
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DDD	Domain Driven Design
HTTP	Hyper Text Transfer Protocol
IR	Information Retrieval
MVC	Model – View – Controller
OAR	Options Analysis for Reengineering
SO	Service Orientation
SOA	Software Oriented Architecture
SOMA	Service Oriented Modeling and Architecture
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
YAGNI	You Ain't Going to Need It
XML	eXtensible Markup Language
DDL	Data Definition Language
TBI	To Be Implemented

Chapter 1
INTRODUCTION

1.1 Background

Today it has become a heavier burden maintaining a very large scale monolithic legacy systems due to lots of complexities. Due to that many researchers are looking for well-defined architectural solutions to cope with such systems. Among those converting existing system into microservices hold a specific place due to simplicity and related other advantages. They are rapidly heading towards the peak of inflated expectations on the Gartner Hype Cycle [1]. As well as there are some skeptics in the software community regarding the microservices convincing nothing much other than simple extension to Service Oriented Architecture (SOA). Despite both the hype and the skeptics, Microservices architecture has considerable advantages.

When it is come to the monolithic world it is very simple and less complex to build and deploy a moderate size software. Normally at the core of a monolithic system lies the core business logic of the system which is implemented by modules that define services, domain objects, and events. Surrounding the core are adapters that interface with the external world. Though is seems like a modular service architecture whole system packaged and deployed as a one system. Today most of the systems that we see in the literature are monolithic systems. But unfortunately, such a simple approach has its own disadvantages. That is when it comes to a successful application it always tends to modification and grow overtime which is basically adds many lines of codes to the existing system. Once the system quite large and complex such a system development team and the organization would be in the hell of pain. One of the major thing is becoming much more complex thus it not be possible for a one single programmer to understand the whole system code itself. As well as small change for a code causes to larger deployment times not only that small bug will cause to system crashes fully. That is where the microservices comes to rescue from such a heavier burden.

In a microservices system, idea is to build to set of smaller and interconnected layer of services to provide the required functionality thus reducing the complexities and the other pitfalls in the monolithic systems. Applications typically use the three types of scaling together. Y-axis scaling decomposes the application into microservices as shown above in the first figure in this section. At runtime, X-axis scaling runs multiple instances of each service behind a load balancer for throughput and availability. Some applications might also use Z-axis scaling to partition the services. In a microservice architecture each of the service has its own database. Since gain the ability of using diverse of the database types which can be match to the different sort of requirements. But that adds some more complexity to the system. That is controlling the distributed transaction and failovers.

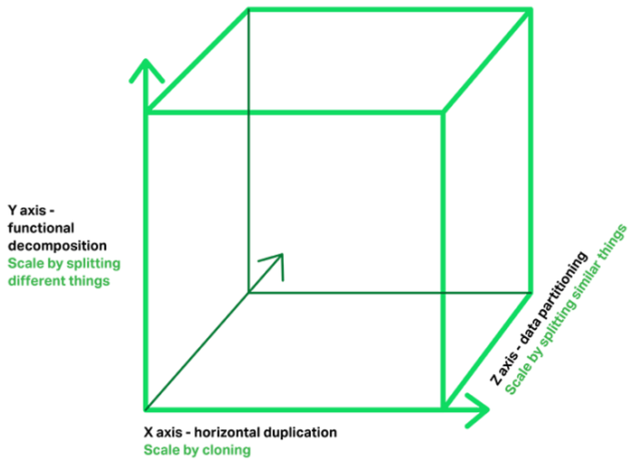

Y axis - functional decomposition
Scale by splitting different things
X axis - horizontal duplication
Scale by cloning
Z axis - data partitioning
Scale by splitting similar things

Figure 1-1: Types of Scaling.

Many people today trying to discover the benefits of converting large scale monolithic systems into smaller microservices. Those microservices have a standardized interface that enables them to tolerate individual service failures and updates without catastrophically impacting the overall business solution. But most of the people who has even the deep knowledge about the architectural concepts and related field in a dilemma of deciding whether to build a monolithic solution (Brownfield) and convert it to a microservices architecture, or to build a microservices architecture from scratch (Greenfield). But when it comes to the Greenfield solution it could be high risky and lots of time consuming. Due to that most of the architectures or the system consultants trying to move the less risk part of the system into microservices initially.

But identifying those less risky services in a very complex and large scale monolithic system would be a headache for a system architecture. There could be huge number of dependencies and complexities on each and every single service. It would not be possible to look at each line of code and decide what to move as a microservice and what not to.

1.2 Problem Statement

As pointed out earlier it is quite a challenging and complex task to identify and classify the existing service layer into microservices. Because there could be different service related metrics to consider when we are separating out from the existing services. Such as business values, dependencies, requirements, risks and deadlines etc. But it would be great and valuable to have a tool that support to identify services from the existing monolithic system which are more feasible to convert into microservices. Otherwise if we take a system which has millions or billions of lines of code, it is not feasible or possible to go through each line of the code or architectural documents to identify the feasible service to convert. Not only that only one person won't be able to understand the whole code base or the system architecture that leave us a problem to solve. What if there exist a tool which help the architectures or the system consultants that can use to decide which services need to convert and which are not to. In this research trying to address this new problem to identify the services which can convert as a microservices with less risk and effort. But it is not an easy task to provide a tool which gives such a support there could be different aspects that need to take into account when identifying those services. Those could be the architectural constraints, system limitations, business requirements, architectural complexities and dependencies. Among those using a tool we will be limited to finding a solution for problems like architectural complexities, and dependencies only. Because those are the factors that only can be measurable theoretically as well as practically using a simple tool. Other factors may require even more complex logics and research areas to accomplish (i.e. AI).

1.3 Motivation

Finding a solution to the above state problem is quite hard task, despite all of the barriers might come across when implementing the solution for the problem, solution for this problem would address several benefits which cannot be easily gained from an existing monolithic approach. As pointed out earlier monoliths are built as a single systems unit, because of that for every possible functionality such as handling HTTP requests, executing domain logic, database operations, communication with the browser/client, handling authentication and so on, that unit must bear the responsibility, which sort of lead to a problem of single point of failure, failure of that unit might cause to fail the entire system. Having such architecture, even with smaller changes in the code base it requires building and deploying the whole application.

Not only that when it comes to the scalability aspect of the system, to have a quite large scalable system, it might require have to run multiple instances of the monolith, even if you know that the bottleneck lies in one component only. For an instance if we take a simple image processing application as depicted below,

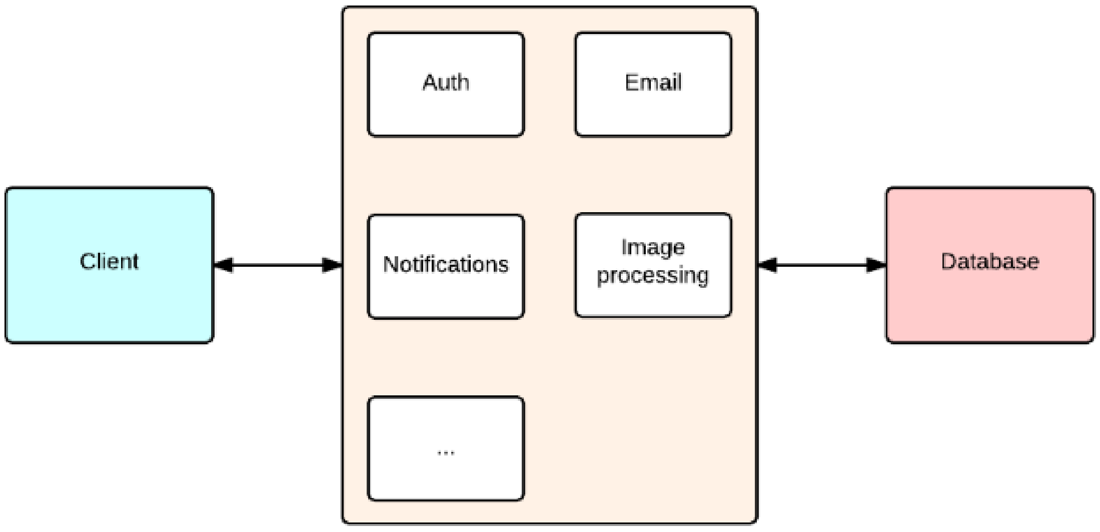


Figure 1-2: Sample Legacy System.

If many clients are accessing the services and try to upload lots of images, application servers will not be able to handle the high requests congestion, due to that whole application will suffer performance issues. In order to solve such a simple problem there, exist two possible solutions, either scale the application by running multiple instances of the monolith or move the logic into a microservice. But it is clear that there is no point of creating instance for entire system just to resolve the image processing related performance issue, instead we can easily put those services into separate microservices and we can scale those services upon the scalable requirements.

As depicted above the picture we can serve the different types of services as microservices based on the scalability requirements. Not only that, having a microservice architecture will bring much more advantages to the system. One of the biggest advantages of the microservices pattern is that it does not require you to rewrite

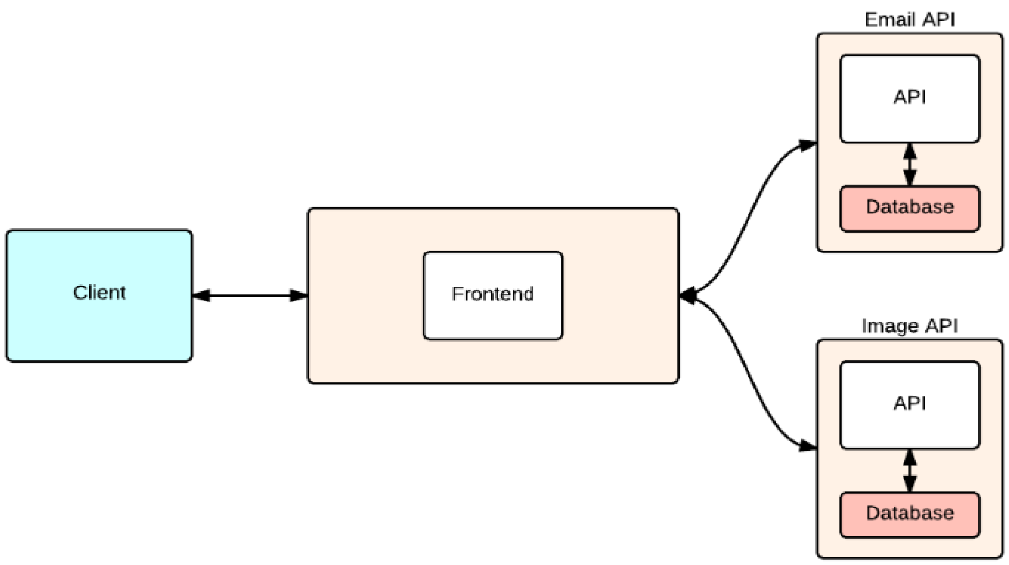


Figure 1-3: Example Microservice System.

your whole application from the ground up. Instead what you can do is to add new features as microservices and plug them into your existing application. Since each microservice deals with one concern only which ultimately wrapped into a small system for each microservices and this result in a small codebase, which means easier maintainability. And also, in order to release a newer version of some part of the system will not require entire system to rebuild and test again, since such architecture having smaller dependency circle, it only required to deploy those services. Not only that it increases the system resilience as well, as such a system is composed from multiple microservices, if some of them goes down only some features from your application will go down, not the entire application.

Because of those advantages we see from having such a system architecture it motivates to convert the existing monolithic systems into the gain the benefits as described above. This solution could be helpful for very large and complex monolithic systems to convert those existing systems into microservices based system with less risk by identifying the less risk services in beforehand of implementation.

1.4 Objectives

The main objective of this research is to create an architectural helper tool where it can be used to get insight about the microservices in the existing monolithic system. By having an insight about all the system services, it would be easy for anyone who is willing to convert existing monoliths into microservices architecture. And also, this consist with two main process activities. First step is identifying services in the existing monolithic system, and the second one is creating the architectural helper tool.

Service identification is the major role in the process of converting existing monolithic systems into microservices based system. Due to that this would be a one of major object of this research. To identify services in an existing monolithic system there, exist lots of methods in the service identification literature. Find a best and most suitable method will another sub objective of this research. And also, there will be such requirements that where we need to change the some of those identified methods in order to match with the microservice identification, since many of the service identification techniques that available in the literature are done to identify the services for Service Oriented Architecture (SOA). Both the SOA and microservices indeed have many similarities, but microservices has a special approach of breaking a monolithic application into many smaller services that talk to each other, SOA has a broader scope.

Next key objective is, develop an algorithm where we can used to filter out the most relevant microservices by using the above identified methodology. In Order to achieve the best out of those methods since many of the developed methods and algorithms used only to identify the SOA related services. Then after in the process of creating that architectural tool next objective is identify the methods and technologies to create such a tool. In this final process step will try to accommodate best technologies and methodologies which can be used to represent many information related to the each of the identified services. Using that information anyone who is interested in converting existing monolithic system into microservice architecture-based system will get much more comprehensive idea about which services to convert and which are not to base on the risk-based sorting.

1.5 Research Scope

Initial phase of this research trying to find a way to identify the microservices from the existing monolithic systems. But this research scope will be narrowed down to one architecture pattern and one language due to the time limitations. In this research it will mainly focused on MVC pattern where an architecture consists controllers, services, repositories and managers etc.

Chapter 2
LITREATURE REVIEW

2.1 Extracting Microservices

This research area is pretty much new and currently being emerging a lot. Due to that; it is quite hard to find a research which have done on the same track, but there are few that are done on the microservices related such as Designing a Business Platform using Microservices [1] by Rajendra Kharbuja, Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems [2] by Alessandra Levkovitz et al and Application Development and Microservices: A case study [3] by Pedro Felipe et al. Among the above listed research articles Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems Article is more relevant to the research which this paper describes.

In Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems [2] paper they have described a technique to identify and define Microservices on a monolithic enterprise system. As the major contribution, their evaluation demonstrate that their approach could identify good candidates to become microservices on a 750 KLOC banking system, which reduced the size of the original system and took the benefits of microservices architecture, such as services being developed and deployed independently, and technology independence. According to their proposed technique they have considered that monolithic enterprise applications have three main parts a client-side user interface, a server-side application, and a database also considered that a large system is structured on smaller subsystems and each subsystem has a well-defined set of business responsibilities not only that they have also assumed that each subsystem has a separate data store. And then they have map their selected system database tables into subsystems from that they created a dependency graph where vertices represent facades, business functions, database tables and edges represent the each of those function invokes. And then after they identified the set of services through the dependency graph to convert as microservices and finally they created API gateways to turn the migration to microservices transparent to clients.

In a blog article [4] it describes several principles regarding when converting existing monolithic systems into microservices. If a monolithic system has design in a way that matches those architectural principle those makes easier to convert them into microservices. That are Domain driven design(DDD), Separation of concerns using MVC, High cohesion, Low coupling, Do not repeat(DRY) [5], Convention over Configuration (CoC)[6], You Ain't Going to Need It (YAGNI) [5]. That logic might be helpful when deciding which systems can be easily converted into microservices and which are not.

In a separate blog article [7] which has the topic Monolithic to Microservices Refactoring for Java EE Applications, they also talk about the architectural principles that are already mentioned above. This also highlights the same point about easiness of converting those existing monolithic applications into microservices when having known architectural patterns. Because without knowing an architectural pattern that the current monolithic system follows the task of converting such a system into microservice would be pretty much difficult or even can be an impossible task to achieve.

2.2 Service Identification Techniques

2.2.1 Structured and Unstructured Analysis Approach

For this research to continue it requires to analyze the existing legacy code base and identify the existing services. For that there are couple of research papers that can get few ideas. First one is Service Identification in Legacy Code Using Structured and Unstructured Analysis by Inbal Ronen et al [7]. According to that research paper they see another aspect of reuse of some of the existing legacy code base such as Lots of resources and time have been spent on

Existing legacy systems, need to retain as much as possible of previous investments, requires identification of where a service or part of it is already implemented and can be reused and Manual identification of candidate code sections is tedious and requires domain experts. According to their paper they have discussed about 3 approaches on transforming existing legacy code into SOA based system. The first one is Top Down approach that is Define to-be model and implement it and No consideration of existing system. The second approach is Bottom Up that is Start from given legacy as-is system and expose as SOA, but this approach is harder to adapt to new business models. And the final approach is Meet in the middle that is hybrid approach of the above two method Combine Top Down and Bottom Up approaches. Start from target to be model and Map into existing system then Exploit reuse and implement if needed. Then they have discussed ways to analyze the code base and they talked about main two approaches for code analysis that is Structured Code analysis and unstructured code analysis. In Unstructured code analysis Information Retrieval (IR) techniques, e.g. tokenization, usage of thesauri and stemming of source code and comments and No consideration of code structure and semantics. In Structured code analysis, Classic static analysis, e.g. control flow, data flow and No comment analysis as well as No identification of non-exact matches. But Combination of techniques facilitates effective and precise service implementation search.

Finally, in that research paper they have discussed their approach on identification of services with SOMA – Service Oriented Modeling and Architecture. According to that An IBM end-to-end SOA method for the identification, specification, realization and implementation of services, components and flows. Under **Service Identification** combines top-down, bottom-up, and meet-in-the-middle techniques for the identification of services to be implemented in the new SOA environment. For **Service Specification** – further designs the subsystems that were found in the previous step and specifies the coordination between them. Details the components that implement the services. **Service Realization** - defines the software that realizes a given service.

According to the Service Identification method, receives a service title as input, searches for potential implementations in the code and ranks the results by relevance to the service title. This process happens under several stages, **Stage 1**: source code processing, in this stage they have Analyze code structure to Identify components of interest in code and comments then Insert code and processing results into repository. **Stage 2**: search and ranking, in this stage Search for service title matches in the artifacts that have been processed Rank match relevance, taking into account structural and semantic context. This Stage 2 can be repeated for multiple service titles over the same processed code artifacts.

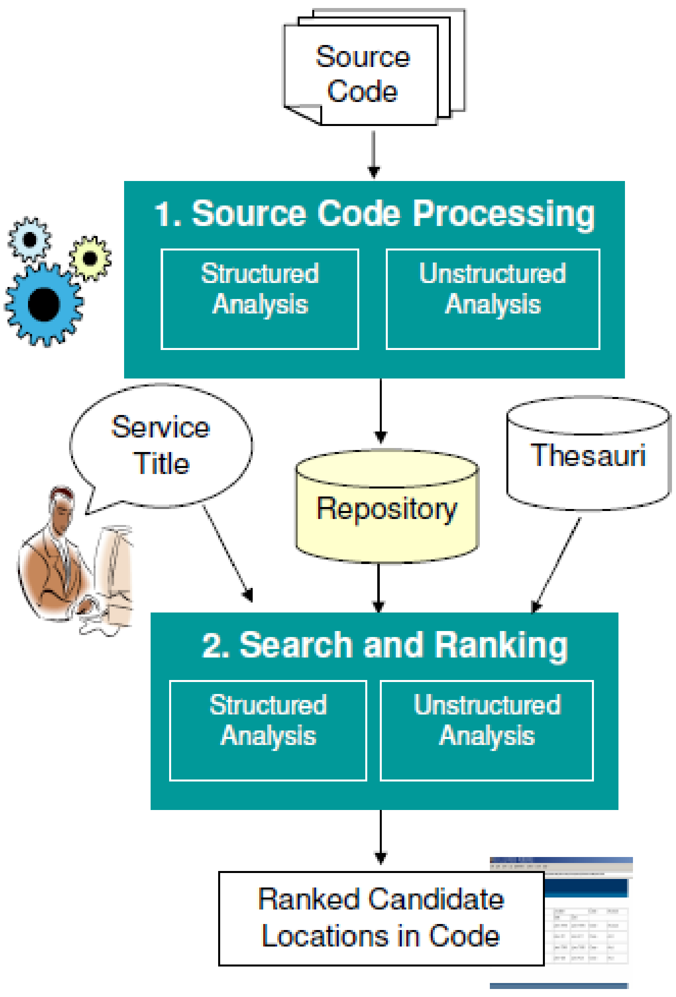


Figure 2-1: Two stage processing

\

Source Code Processing Stage

Identify programming constructs (e.g. variable declarations, procedure names, comments). In this stage basically Perform shallow analysis based mostly on a composition of regular expressions, perform deep static code analysis (control flow and data flow detection), Analyze comments, exploit conventions and Mark constructs using annotations. And also Tokenize - enable matching of substrings under this Consider special characters (spaces, commas, underscores) and code naming practices (Hungarian notation, Camel Case) then Insert tokens and annotations into repository their Ignore tokens with low semantic value (e.g., “and”, “the”) and their method uses a search engine as the repository (provides indexing and querying capabilities).

Search and Ranking Stage

In this stage Tokenize service title, that is Apply the techniques used during the source code processing stage. Construct and execute search query that is Include the tokens from the service title Exploit unstructured analysis capabilities of the search engine (e.g. stemming, thesauri and abbreviation usage) to search for inexact matches. (i.e. Provide common language and domain-specific thesauri and abbreviation dictionaries to the search engine) then Analyze query results Search engine returns the location of each query token occurrence (or its synonym), The method assembles valuable occurrence combinations such that There is exactly one match for every token (or one of its synonyms) and Token match locations are close to each other. Finally, virtual similarity: 100% for exact match of all service title tokens, Aggregate results to procedure level and Apply supplemental ranking heuristics based on semantic context.

Ranking Heuristics

They High score for match in procedure declaration. Higher score for match in procedure declaration and in the adjacent comment, Match in variable declaration or close to it for this Use data flow analysis results to find variable referencing code and Identify the reference location as a match with low score then Separate matches for service title subject (noun) and action (verb), Look for noun matches in or close to a variable declaration and Look for verb matches in or close to the variable referencing code.

2.2.2 Architecture based Approach

Next is Service Identification and Packaging in Service Oriented Reengineering research paper which was published by Zhuopeng Zhang et al [8]. In this research paper they present an architecture-based service-oriented approach to support service-oriented reengineering. It integrates and reuses software components as services over Internet by wrapping underlying computing models with XML. Service identification and service packaging processes are described in detail, which are the keys to service-oriented reengineering. Target services are defined according to service identification, which is based on a comparison of analyses in both problem domain and legacy systems. Clustering techniques are applied in service identification process to analyze recovered architectural information and to identify related modules. Service packaging is benefit from architecture-based development and reuse.

Web service technology and Service-Oriented Architectures (SOA) are rapidly developing and widely adopted. Web service technology is to build complex Web-based systems fast, rapidly adopt changes and to provide effective inter- and intra-enterprise system integration. Web services have been born as a natural evolution of different technologies, approaches and demands from communities in business and technology compared to other integration technologies such as Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM), and Distributed Computing Environment (DCE) – all involved communication within different closed technologies. Web services are based on Service-Oriented Architectures (SOA), which is the keystone of service-oriented computing. SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. The service implementation can evolve easily since it is hidden behind the service interface. The implementation is able to evolve without breaking applications that consume the service. SOA is particularly applicable when multiple applications running on varied technologies and platforms have to communicate with each other. It is an architecture evolution, and it affects the software life cycle from the software as a service point of view. Affected by this Service Orientation (SO) trend, many existing non-service-oriented software systems will become legacy systems. These legacy systems need service-oriented reengineering, which can facilitate legacy evolution in service-based computing environment. From economic aspects, a business is constantly reorganizing, changing its boundaries and reconfiguring its activities. Service-oriented reengineering enables legacy systems to adapt to continuous changes in business logic and market requirements. From technical aspects, application integration towards services and service choreography will become common in service-oriented computing.

Service identification process

Their approach is especially applicable to reengineering tasks, which have some of the special characteristics such as A legacy system needs to be migrated into a network environment, and it can be wrapped and exposed as a Web service, there are some reusable and reliable functions embedded with valuable business logic in the, Legacy system. These functions are useful to be exposed as independent services from the requirement point of view, some reuse components from the legacy system are fairly maintainable compared to maintain the whole legacy system, some components of the target system run on different platforms and vendor products and the target system will operate over the Internet where reliability and speed cannot be guaranteed.

Step 1: Legacy system evaluation.

This evaluation reveals the current status of a legacy system and specifies which phase it is in its lifecycle. A decision tree is used in the assessment in order to consider many aspects altogether. Techniques such as the Options Analysis for Reengineering (OAR) are used to guide the decision-making process. Reengineering decisions are made according to this assessment.

Step 2: Architecture recovery.

The goal of this step is to obtain design and architecture information as much as possible by reverse engineering techniques. The static and dynamic analysis techniques applied in this step will depend on the features of the legacy system. Various modularization criteria, such as coupling, cohesion, reliability, maintenance measures, are adopted to identify potential components and connectors. The results are the foundation of further reengineering work and the input of later service identification.

Step 3: Service identification.

On the one hand, it is to determine what business functionalities should be provided by the target service in the application domain. On the other hand, the business functionalities embedded in the legacy system should be identified in order to be reused in the target service.

Step 4: Service packaging.

According to the results of service identification, legacy components and newly-built functional components are composed by connectors. This is an architecture-based integration process.

Step 5: Service publication and choreography.

After a Web service is created, it will be registered under Universal Description Discovery and Integration (UDDI). UDDI creates a standard interoperable platform that enables companies and applications to quickly, easily and dynamically discover and use Web services over the Internet.

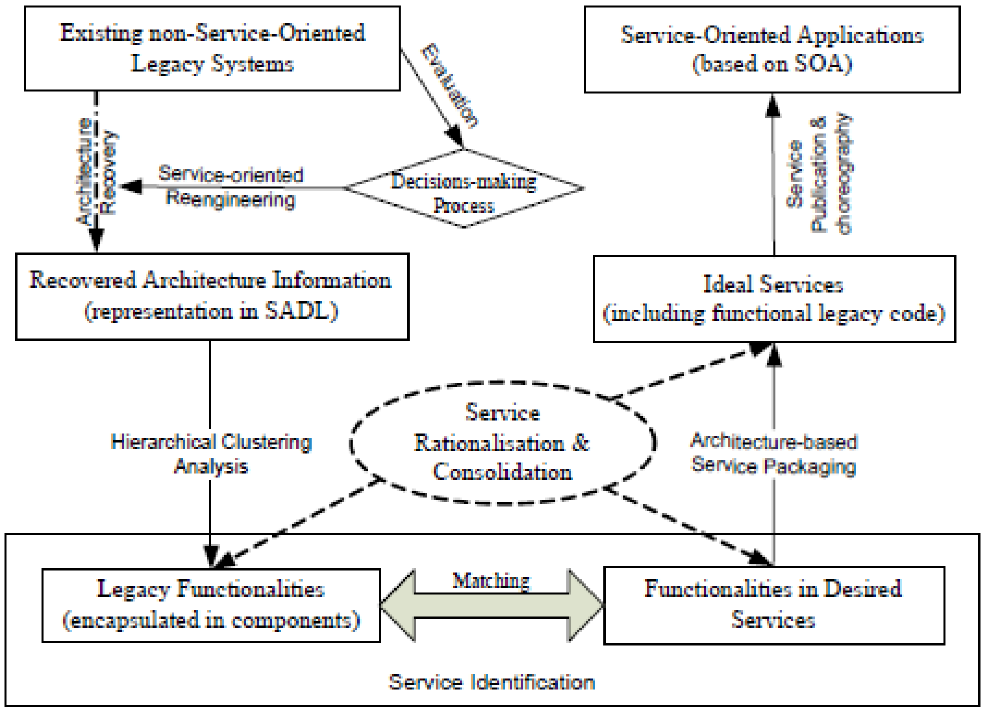

Existing non-Service-Oriented Legacy Systems
Evaluation
Service-Oriented Applications (based on SOA)
Architecture Recovery
Service-oriented Reengineering
Decisions-making Process
Service Publication & choreography
Recovered Architecture Information (representation in SADL)
Ideal Services (including functional legacy code)
Service Rationalisation & Consolidation
Hierarchical Clustering Analysis
Architecture-based Service Packaging
Legacy Functionalities (encapsulated in components)
Matching
Functionalities in Desired Services
Service Identification

Figure 2-2: Architecture based approach for Service Identification

2.2.3 Service Identification with Domain analysis

Properly identifying services and determining reusable legacy components are a critical step in architecting a service-oriented reengineering task. Service identification process consists of the following three steps

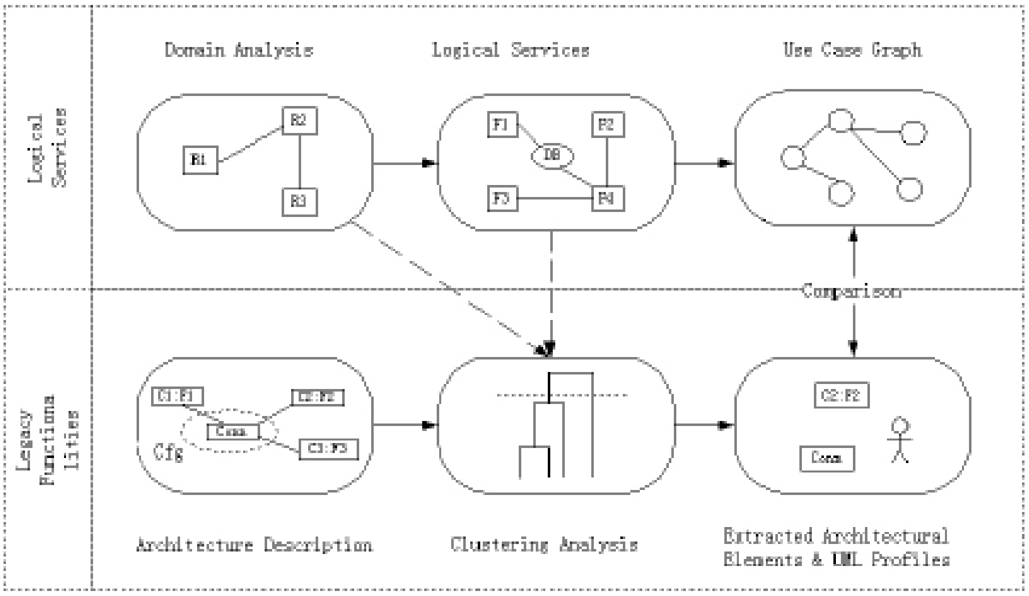


Figure 2-3: Service Identification with Domain Analysis

Step 1:

Service identification in problem domain. It starts with a domain analysis. The goal of this domain analysis is to identify and document requirements on a set of systems in the same application domain. The domain analysis process can be carried out in two steps, subdomain identification and analysis of the selected subdomain. The results of the whole domain analysis are summarized as a domain model. This domain model can be represented by UML and processed by some modelling tools, such as Rational Rose. Based on this domain model, some business functions are identified to be valuable and reusable and needed to be provided as services. In this way, some logical services are summarized, which are mainly based on requirement analysis.

Step 2:

Service identification in a legacy system. There are some abstracted components in the legacy system, which encapsulate valuable functionalities and are reusable in the target service. The recovered architecture describes these components and their relationship.

They have adopted a requirement-driven agglomerative hierarchical clustering method with a complete linkage algorithm, which can provide the most cohesive clusters according to empirical evidence. Components and connectors, which are described in SADL files, are defined as the entities to be clustered. Features are defined to measure the similarity between entities. The weight of features is calculated according to the architectural configuration and the specification of components and connectors. A dendrogram is obtained after executing this clustering algorithm on these SADL descriptions. This dendrogram is analyzed according to requirements. In order to identify a functional business service from these architectural elements, a cutting point must be determined in the dendrogram. The architectural elements in sub-clusters, which are partitioned by the cutting point, are candidates for reuse in the target services. The selection of the cutting point is an elementary factor for determining the granularity of extracted code. So far, architects are responsible for making the cutting point and selecting subtrees manually according to other recovered design information. This clustering method together with human supervision provides a powerful analysis on recovered architecture, and restructures legacy systems into coarse-grained and loose-coupling modules.

Step 3:

Matching legacy functionalities to business functions in the logical service. Because the SADL

Representation has equivalent UML profiles and the logical service model is represented in UML as well, the comparison between legacy system architecture and logical service architecture enables to determine the selection of reusable legacy functionalities and the construction of unavailable functionalities.

2.2.4 Semi-Automated Approach

In [11] research paper which is done by Ravi Khadka mainly talked about approach of developing a consolidated legacy-to-SOA migration approach that facilitates the (semi-) automated service identification and service extraction from the legacy systems and the open-source components. According to mentioned research approach it consists with several process steps as depicted in the below picture.

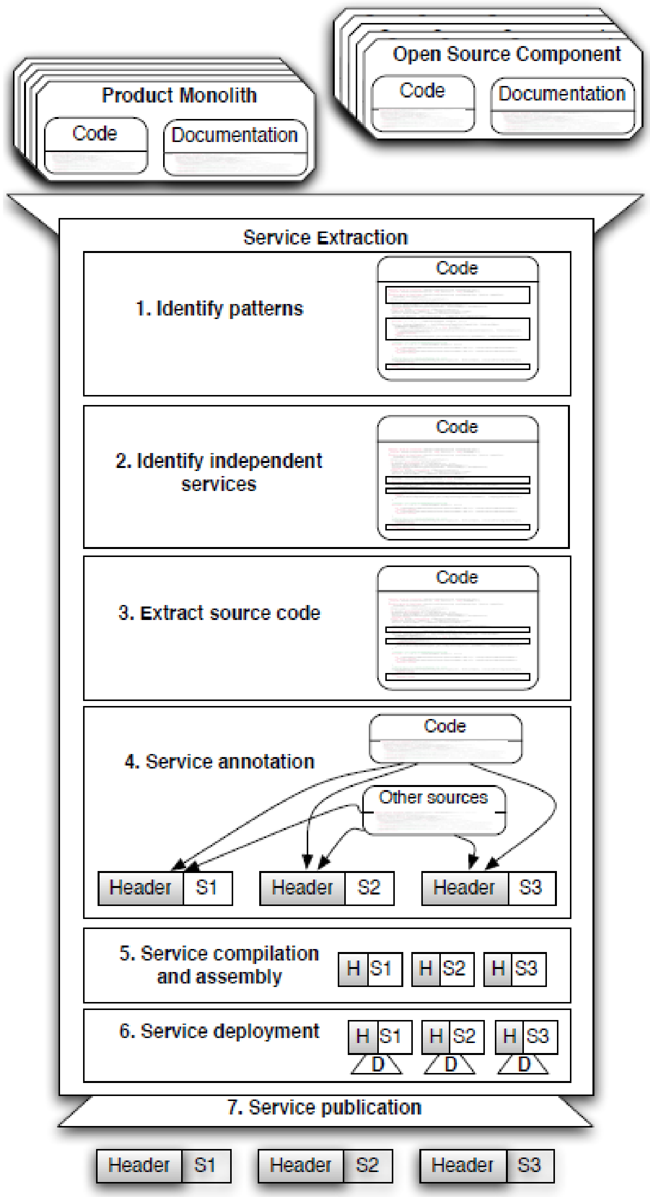


Figure 2-4: Service Identification with Semi-Automated Approach.

Consolidated legacy-to-SOA migration approach, referred as" ServiciFi method". This method has developed using [11] method engineering and [12] concept slicing which combines the migration feasibility and technology support required for the legacy-to-SOA migration. The method engineering approach is used to assemble relevant activities from existing service-oriented development methods (SODDM [13], WSIM [14] and SOMA [15]) and cost-benefit analysis as suggested by Sneed [16]. Furthermore, concept slicing is used to facilitate the extraction of the legacy code to be exposed as services.

Candidate service identification involves two steps according to that research that is patterns identification and service identification steps as depicted in the figure 2.4. Since this a most crucial and critical task it this approach is contains set of steps to follow. Initially, the research aims to exploit architectural reconstruction [17], [18] and [19] source code visualization. This step helps to explore the legacy code which gives a proper understanding about the legacy code base and to obtain structural properties of the source code. In this process next step is to identify the design patterns [20] [21]. Final step is using linguistic analysis techniques [22] identify the semantic patterns and by doing a concept analysis [23] locate the appropriate concepts that have been applied in the source code. After completing all the step in this process, it would help to identify the candidate services to extract.

Next step in the process it the service extraction, according to the research paper they have used Concept slicing [12] technique to extract the complete code representing the identified functionality. Not only that this paper has used investigates code-query technologies [24] to extract the source code in language independent fashion. This code-query technique supports so-called extract-abstract present paradigm. Finally, those code extractions map to the relations, and to the query language and to build the new relations. After that result are obtained in a manner which can be presented to the users.

2.3 Summary of Service Identification

In A Structured Legacy to SOA Migration Process and its Evaluation in Practice [25] research paper which was done by Ravi Khadka et al, talks about different approaches on candidate service identification techniques. Identifying candidate services is an important activity in the context of legacy to SOA migration as this activity enables reusability and leveraging the existing legacy features [26]. A plethora of methods are reported (Gu & Lago [27], Arsanjani et al. [28]) to identify potential services. In the literature of the Candidate Service Identification (CSI) there exist several practices. Basically, those type of practices of CSI is broadly categorized into two such as

I. Top down
II. Bottom Up

In the top-down approach, initially a business process is modeled based on the requirements and then the process is subdivided into sub-processes until these can be mapped to legacy functions. The top-down approach is used by Alahmari et al. [29], Fuhr et al. [30], Ricca & Marchetto [31], and Zillmann et al. [32]. In contrast, the bottom-up approach utilizes the legacy code to identify services using various techniques such as information retrieval [33], concept analysis [34], business rule recovery [31], and source code visualization [35].

2.4 Challenges in Service Identification

In the CSI literature many of the researchers argue that locating and identifying service rich areas in legacy applications is not only a challenging task, but also an open problem [36], [37]. The functionalities are embedded in legacy applications in such a way that it is difficult to isolate the business functionality from the complex user interfaces and data access logic. Equally challenging is the determination of the optimal granularity of the candidate services such that they contribute to the business goals. Zhang et al. [38] argue that (i) service rationalization- an analysis process to identify the least frequently accessed component as a candidate than that of more frequently accessed ones; and (ii) service consolidation- an iterative process for redefining all the similar service instances into a consolidated version that supports a superset of all the functions exposed by the individual functions, can improve the candidate service identification.

2.5 Migration Techniques used in Legacy Systems Migration

The implementation phase is related to the execution of the migration of the legacy applications to other architectures. Needless to say, that the implementation depends on factors such as proper migration strategies, assessments of available tools and techniques while executing the migration process. According to [25] they have classified migration strategies into four categories such as;

1. **Replacement** - Legacy application is replaced entirely with a commercial off-the-shelf (COTS) product
2. **Integration** - Existing legacy application is accessible via an interface
3. **Redevelopment** - Entire legacy application is Re-Developed.
4. **Migration** - Legacy application is gradually moved.

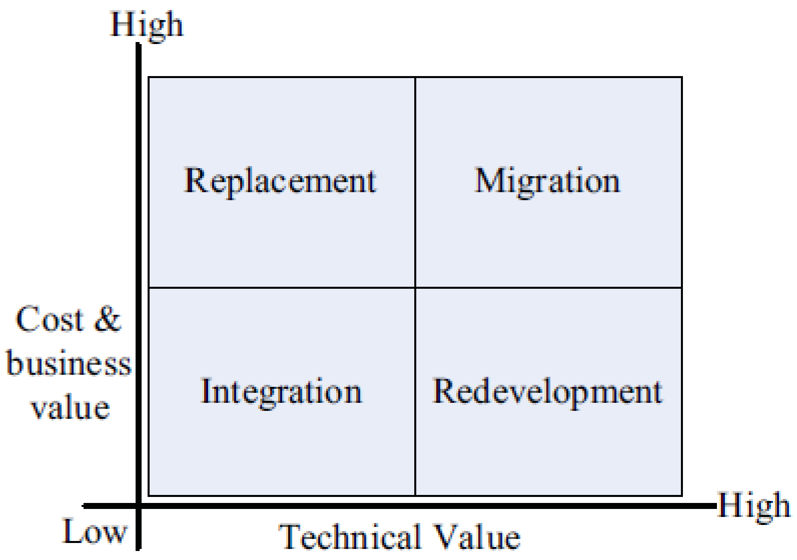


Figure 2-5: Types of Migration Techniques

Even though this research paper mainly focused on the SOA related methodologies there is much closer similarities between microservices based architecture and the SOA. When comparing a microservices architecture and a service-oriented architecture (SOA), it is nearly impossible to gain agreement on how they are related to one another. Adding application programming interfaces (APIs) into the mix makes it even more challenging to understand the differences. Some might say that these concepts are distinct, solve their own set of problems, and have a unique scope. Others might be more generous and say that they achieve similar goals and work from the same principles. They might also say that a microservices architecture is a "fine-grained SOA" or that it is "SOA done right." So that we can take those classification also relevant and valid for the microservices as well.

According to this paper implementation techniques which are heavily used in converting legacy systems to SOA based systems can be categorized into two broad groups. That is;

1. Code level techniques
2. Architectural level techniques

Those are depicted in the above Figure 8, in the literature of legacy to SOA migration their other techniques where code level group can be further divided into categories. Such as slicing [39], [38], [40], [41], wrapping [42], [43], [37], refactoring [44], code transformation [32]. Due to the several factors like fast, less risk, economical and easiness wrapping techniques has become more popular on the category of code level techniques. Not only that even in the architectural techniques there exist some other sub techniques which can be subcategorized into. Such as graph transformation techniques are used by Heckel et al. [45] and Fuhr et al. [30]. Some of the other techniques being used in legacy to SOA migration are inspired by model-driven engineering [30], [29]. Below figure shows summary of the categories that identified as major and sub techniques on used as implementation techniques for legacy systems to Software Oriented Architecture (SOA) convert.

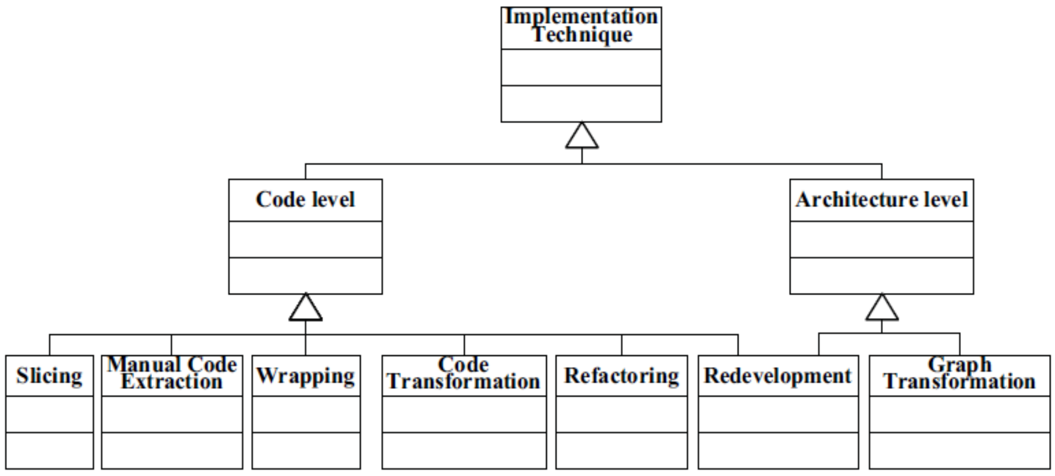
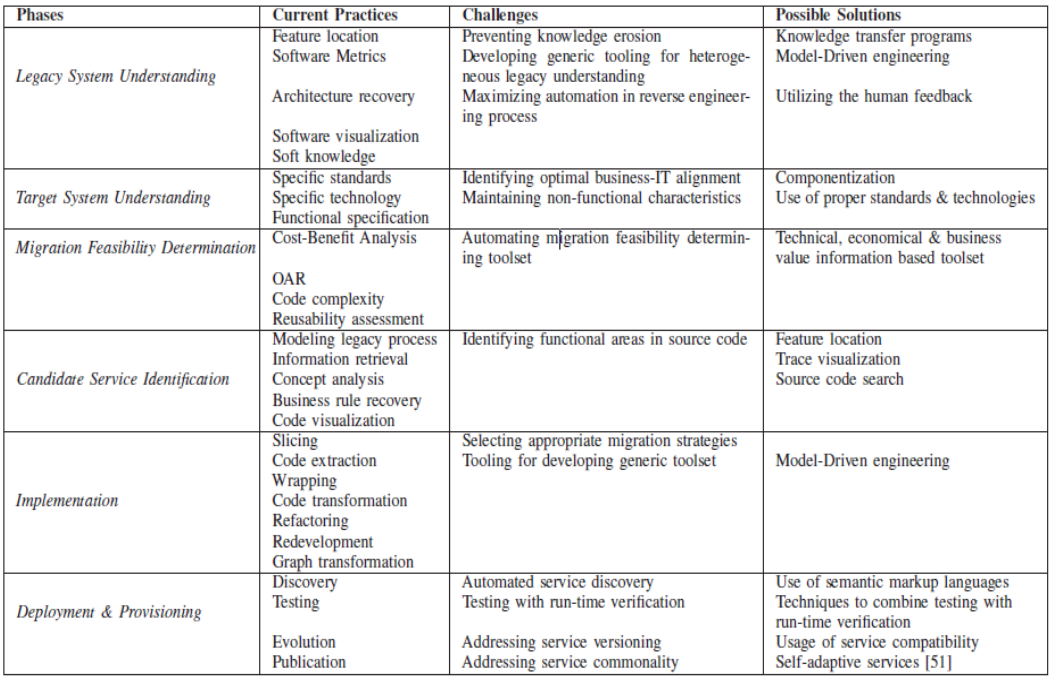


Figure 2-6: Implementation Techniques

2.6 Challenges in Choosing Implementation Techniques

One of the major challenge in this implementation phase is the identification or the implementation of the most relevant migration strategy. Because it is much more a difficult task to identify the proper guidelines for deciding which factors (e.g., business priority, technical qualities, business value, and non-functional characteristics) have to be considered while selecting a strategy. Furthermore, legacy applications are in a heterogeneous IT landscape and developing tools for each variation of the legacy languages tends to be very expensive. Ultimately this raises a question by opening up a new research topic that is whether the legacy to SOA migration be realized in a language independent manner.

2.7 Overview of the Current Practices on Legacy Systems Migration.



Phases	Current Practices	Challenges	Possible Solutions
Legacy System Understanding	Feature location	Preventing knowledge erosion	Knowledge transfer programs
	Software Metrics	Developing generic tooling for heterogeneous legacy understanding	Model-Driven engineering
	Architecture recovery	Maximizing automation in reverse engineering process	Utilizing the human feedback
	Software visualization		
	Soft knowledge		
Target System Understanding	Specific standards	Identifying optimal business-IT alignment	Componentization
	Specific technology	Maintaining non-functional characteristics	Use of proper standards & technologies
	Functional specification		
Migration Feasibility Determination	Cost-Benefit Analysis	Automating migration feasibility determining toolset	Technical, economical & business value information based toolset
	OAR		
	Code complexity		
	Reusability assessment		
Candidate Service Identification	Modeling legacy process	Identifying functional areas in source code	Feature location
	Information retrieval		Trace visualization
	Concept analysis		Source code search
	Business rule recovery		
	Code visualization		
Implementation	Slicing	Selecting appropriate migration strategies	
	Code extraction	Tooling for developing generic toolset	Model-Driven engineering
	Wrapping		
	Code transformation		
	Refactoring		
	Redevelopment		
	Graph transformation		
Deployment & Provisioning	Discovery	Automated service discovery	Use of semantic markup languages
	Testing	Testing with run-time verification	Techniques to combine testing with run-time verification
	Evolution	Addressing service versioning	Usage of service compatibility
	Publication	Addressing service commonality	Self-adaptive services [51]

Table 2-1: Current Practices on Legacy Systems Migration.

Chapter 3
METHODOLOGY

3.1 Proposed Solution

3.1.1 Overview of Proposed Solution

As a solution to above mentioned problem it is required to develop an algorithm which suggest the less risk services within the existing monolithic service layer. It will output the all possible services that can be transform as microservices along with the priority. In the too it can used to check the usages of the existing services and the dependency hierarchy including the inheritance hierarchy to get an overview of the proposed service layer. Based on the suggestions users will be able to pick the most relevant and the appropriate services to transform into as microservices.

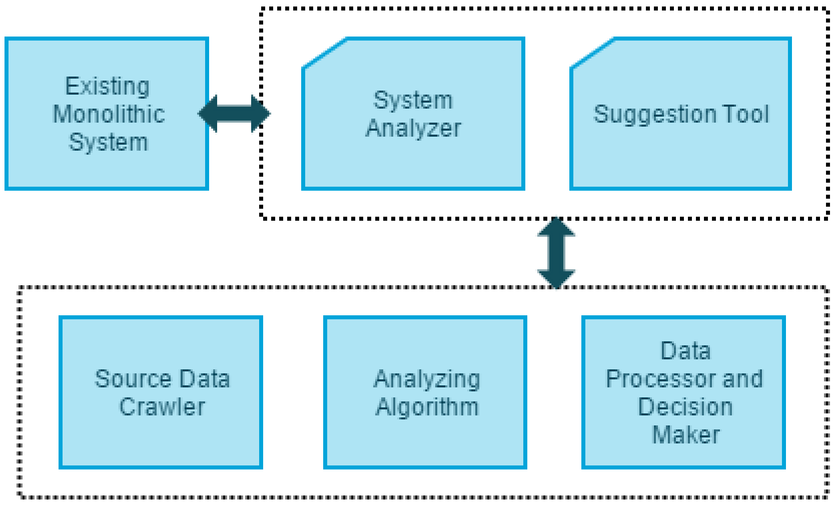


Figure 3-1:System Overview.

As depicted in the above diagram the tool consists with two main parts, namely System analyzer and the Suggestion tool. Basically, the system initially analyzes the existing code base to identify the architectural patterns to take decisions and analyzing those data using the developed algorithm make the suggestions appropriately to sort the less risk services as microservices. In addition to that tool is capable of showing the usages, dependencies as mentioned above (This is a future work). Thus, gives the architectures, system consultants to more flexibility on choosing relevant microservices from the existing monolithic service layer.

3.1.2 Scrutiny of Proposed Solution

For this research mainly it involves two steps, which is Service Identification of a Monolithic system and the creating the architectural helper tool to give suggestions which to choose as microservice from identified microservices. There are many service identification techniques available in the legacy systems service identification literature, among those many of techniques requires large amount of human interaction (e.g. system analyst, maintenance programmer, etc.) that gathers information through interviewing stakeholders in order to fill the gap between existing legacy system and target architecture. Some approaches [8] identifies services by domain analysis and business function identification, those are based on both requirements abstraction and source code levels and needs architectural and requirement information to be available. [9] Proposes an automatic approach to evaluate candidate services as found in the literature review.

Since many of those researches for identification of Services in legacy systems are done mainly rely on documentation or architecture, sometimes it could be very hard to find those resources. But the source code of a system is the always available resource, hence this research is focused on identification of candidate services through the legacy system code. In this approach candidate services are considered as groups of object-oriented classes evaluated in terms of development, maintenance and estimated replacement costs as same as [10] approach. Basically, this service identification process will be done though mainly two steps,

1. Mapping of Object to Services
2. Service Ranking

Mapping of the object classes into service is basically consider a service as a group of classes defined in object-oriented source code. From those identified group of classes, they will be provided some defined service to the system. There will be some inner classes which are usually be the internal connections to other classes of the same service. And also each of those classes interfaces defines the service operations, and each of those class's public methods will be the operations provide by the identified service.

Once the services are identified next task is to rank the services by the set of characteristics or the quality attributes, such as

- **Self-containment** - Higher the number of required interfaces is, the less the service is self-contained.
- **Composability** - Degree of direct and indirect dependence of a class on other classes in the system.
- **Functionality** - Higher the number of interfaces is, the more the service provides functionalities.
- **Usage** - Higher the number of references of the class, more the service use.

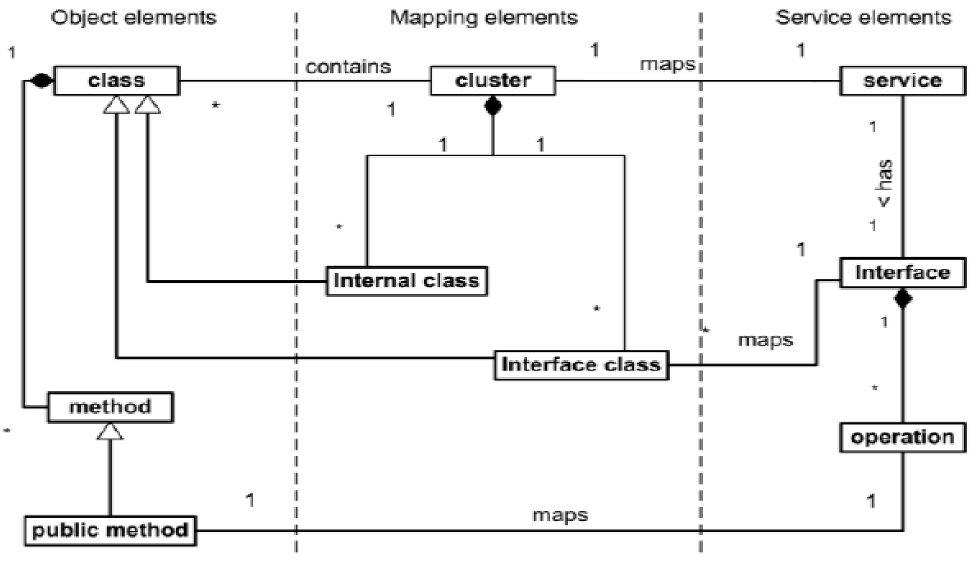


Figure 3-2: Java Objects to Service mapping.

Clustering Process

As pointed out earlier it is required to cluster the related classes in legacy code in order to figure out the services within the monolithic systems. Usually may of researches have done on this area have used hierarchical agglomerative clustering algorithm. To proceed with this approach, we need to develop and fitness function which uses the services ranking logic to cluster them accordingly, that is this algorithm groups together the classes with the minimum value of the fitness function. Initially all of the classes in the legacy code marked as a single cluster and then accordingly merge those clusters into new clusters. This repeating process carried out until the number of clusters is bigger than one. By doing so we can express the monolithic system in hierarchical dendrogram. (See Figure 3-3.)

In order to get the partition of disjoint clusters, obtained hierarchy needs to be cut at some point. Initially on the root node, compare the similarity of the current node to the similarity of it child nodes. If the current node's similarity value exceeds the average of similarity value of its children, then the current node is a cutting point, otherwise, the algorithm continues recursively through its children. Then after by the standard we can apply the aforementioned clustering algorithm to evaluate the existing monolithic system and represent all of its classes as coarse- grained and loose-coupled disjoint set of services. or else it can use a predefined depth to stop the process.

Once after completing the clustering and filtering out the services from the existing monolithic legacy code base, then it is presented to users based on the ranking of the services. And also, it gives flexibility to add the business values to the suggested services and based on the user input values for business values for each of those services raked will be re calculated and rearranged.

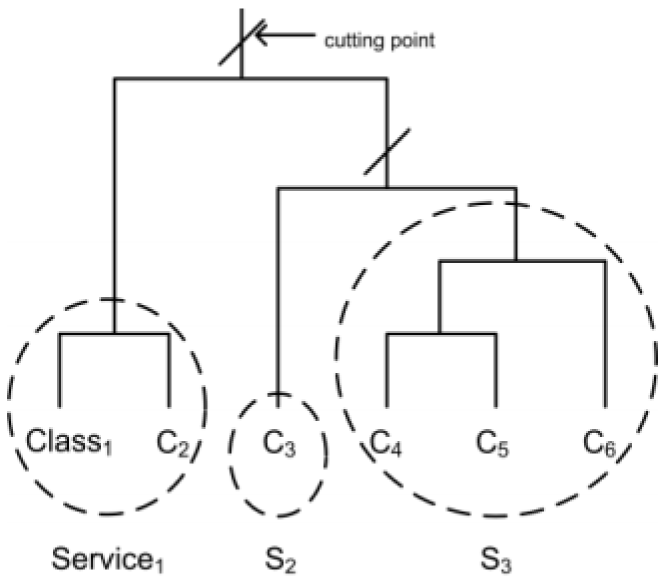


Figure 3-3: Dendrogram of Service Clusters.

Chapter 4
ARCHITECTURE AND IMPLEMENTATION

4.1 System Architecture

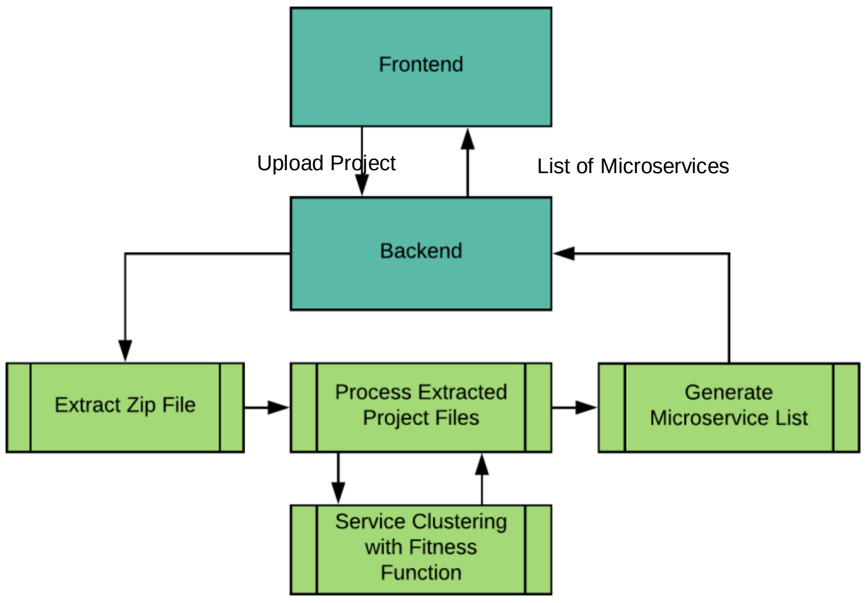

Frontend
Upload Project
List of Microservices
Backend
Extract Zip File
Process Extracted Project Files
Generate Microservice List
Service Clustering with Fitness Function

Figure 4-1: System Architecture.

According to the above system architecture whole system is divided into two main parts, one is frontend and backend. Frontend is that place where the users can upload the project files and get the processed to see the results. Once the project files are uploaded to the server users can process those files to identify the candidate microservices in the legacy code base. Once the files are uploaded and press the process from the frontend. It is starting to extract the zip files and then start to process the project files. In this step files are maps to the services and then those will be ranked based on the fitness function. Finally, those services are clustered into few clusters which are basically be the candidate microservices for the system. Then the response will be send to the frontend and will be displayed on the web with relevant colors, and risk level.

4.2 Solution

As a solution to identify the microservice in a legacy code in the proposed method it identifies services as groups of classes in the legacy software source code. Since the source code be the always available resource to analyze a code to identify service, unlike other researches that have done under the service identification this is an automated approach while many of other approaches are manual. In the basic form of this approach heavily based on the fitness function which will be defined in the sections to come. It measures semantic correctness of each group of source code elements to be considered as a service.

4.2.1 Mapping of Object to Services

As illustrated in the figure 3-2, need to define a relationship or the mapping between services and the object-oriented code in order to identify the services from the object-oriented code. Initially it considers a service as a set of classes defined in OO source code. Among these classes, some define the operations provided by the service, whereas others are inner classes. Inner classes are those which only have internal connections to other classes of the same service. Usually operations provide by the service are be the public methods of the relevant classes. Classes that define the operations provided by the service are the classes that define its interface. Inner classes do not define operations provided by the service.

4.2.2 Fitness Function

Here defines a fitness function for an identified candidate service as a linear combination between the 4 characteristics of services previously defined, such as functionality, composability, self-containment and usage. Let's take fitness function as FF(E) where E be the candidate service, and F(E), C(E), S(E) and U(E) be symbols respectively for the characteristics of the services above mentioned. Hence, we define the fitness function as follows:

$$FF(E) = \frac{\alpha F(E) + \beta C(E) + \gamma S(E) + \delta U(E)}{n}$$

where α,β,γ,δwill be the coefficient weights for each characteristic that are determined by software architect. As well as ***n*** will be calculated as follows:

$$n = \Sigma(\alpha, \beta, \gamma, \delta)$$

In order to evaluate the eligibility of the candidate services in this research we deploy the quality metrics evaluation techniques to evaluate the quality of candidate services. Using these measurements, we will be able to extract the relevant candidate services as the microservices of the monolithic system. For the different aspects or the quality attributes that we have selected above require different measurements;

- **Functionality** - Coupling and Cohesion measurements
- **Composability** - Cohesion measurement
- **Self-Containment** - External Coupling
- **Usage** - Inheritance(IS-A) and Composition(Has-A)

According to the [46] Cohesion of a service measures how strong the elements within this service are related to each other. A service is considered as highly cohesive, if it performs a set of closely related functions and cannot be split into finer elements. The metric LCC Loose Class Cohesion proposed by [47] measures the overall connectedness of the class. It is calculated as follows:

They have defined two measures of class cohesion based on the direct and indirect connections of method pairs. Let NP(C) be the total number of pairs of abstracted methods in abstract class AC(C). NP is the maximum possible number of direct or indirect connections in a class. . If there are N methods in a class C, NP(C) is N* (N-1)/2. Let NDC(C) be the number of direct connections and NIC(C) be the number of indirect connections in AC(C). Then Loose class cohesion (LCC) is the relative number of directly or indirectly connected methods give from the following equation.

$$LCC(C) = \frac{NDC(C) + NIC(C)}{NP(C)}$$

In its simplest form;

$$LCC(C) = \frac{Number\ of\ direct\ and\ indirect\ connections}{Maximum\ number\ of\ possible\ connections}$$

Coupling means the degree of direct and indirect dependence of a class on other classes in the system. Here, two measures are counted: method calls and parameter use, i.e. two classes are considered coupled to each other if the methods of one class use the methods or attributes of the other class. In our approach, Couple(E) measures the internal coupling of the candidate service E and is calculated by the ratio between number of classes inside the service that are internally called to the total number of classes within the candidate service E, and also the term ExtCouple(E) measures the coupling of the candidate service with other services. It is calculated as follows;

$$\text{ExtCouple(E)} = 1 - \text{Couple(E)}$$

According to the above they have defined few other parameters which will be used to measure the functionality, composability, self-containment and usage. That is *np(E)* refers to number of provided interfaces, *LCC(i)* refers to the average of service's interface cohesion within the interface, *LCC(I)* refers to the cohesion between interfaces, *Couple(E)* refers to the coupling inside a service, *LCC(E)* and refers to the cohesion inside a service.

Measure functionality of a candidate Service - F(E)

According to the definition we can calculate the functionality of a candidate service as follows;

$$F(E) = \frac{1}{5}\left(np(E) + \frac{1}{I}\Sigma_{t\epsilon I} LCC(i) + LCC(I) + Couple(E) + LCC(E)\right)$$

Measure composability of a candidate Service - C(E)

Similar to above we can calculate the composability if a candidate service as follows;

$$C(E) = \frac{1}{I}\Sigma_{t\epsilon I} LCC(i)$$

Here *i* refers to the interface.

Measure self-containment of a candidate Service - S(E)

According to the definition Self Containment simply be the;

$$S(E) = ExtCouple(E)$$

Measure usage of a candidate Service - U(E)

To calculate the service usage, let's take the Inheritance factor H(E) for given class and composition factor as P(E) Hence we calculate the class usage as follows;

$$U(E) = H(E) + P(E)$$

4.2.3 Service Clustering

To extract all the candidate service from the legacy monolithic OO code base, we need to group those classes based on relevant dependencies that each of those classes have on each other. In order to do that here we deploy hierarchical agglomerative clustering algorithm and this algorithm uses above defined fitness function to groups the classes with the maximized value of the fitness function. In the initial phase of every class of the code base considered as a single cluster and by recursively those classes are get paired in to set of clusters based on the fitness function value. That is this algorithm merges the pair of clusters with the highest fitness function value into a new cluster. And the it measures the fitness function between the new formed cluster and all other clusters and successively merge the pair with the highest fitness function value. For this algorithm input will be the source code classes and it outputs set of hierarchy of clusters as a dendrogram. Here is the pseudo code for the process.

```
 let each class be a cluster
      compute fitness function of pair classes
repeat
      merge two "closest" clusters based on fitness function
value;
      update list of clusters;
until only one cluster remains or reach the desired depth
return dendrogram;
```

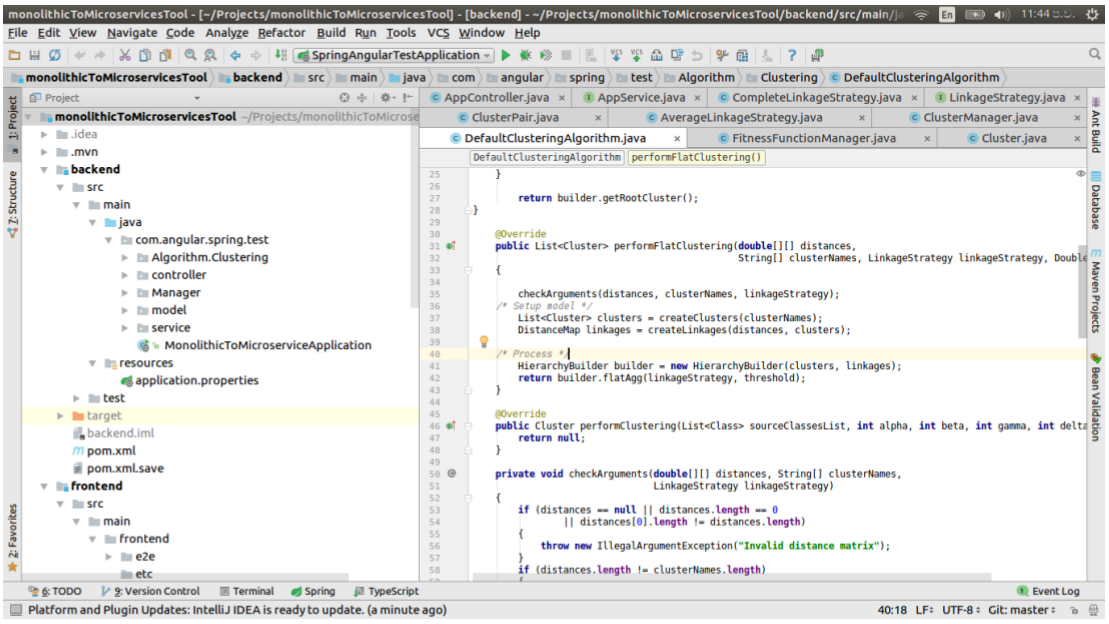
As well as in order to partition of disjoint cluster the result of the above process needs to be cut at some point, to do that we employ the standard depth first search (DFS) algorithm or the desired depth which will take as an input value. Initially on the root node, we compare the similarity of the current node to the similarity of it child nodes. If the current node's similarity value exceeds the average of similarity value of its children, then the current node is a cutting point, otherwise, the algorithm continues recursively through its children. By doing so eventually the algorithm outputs the set of classes as coarse grained and loose-coupled disjoint set of services. Ultimate result is depicted in *Figure 3-3*.

4.3 Implementation

For the implementation of this project I have used two different technologies for the frontend and backend. That is for the frontend it uses angular2 and for the backend it uses java with spring boot. This architecture works as a client server architecture where angular frontend communicates with spring java backend with REST endpoints.

GitHub: https://github.com/chamikabm/monolithicToMicroservicesTool

Project Structure:



```
monolithicToMicroservicesTool ~/Projects/monolithicToMicrose
  .idea
  .mvn
  backend
    src
      main
        java
          com.angular.spring.test
            Algorithm.Clustering
            controller
            Manager
            model
            service
            MonolithicToMicroserviceApplication
        resources
          application.properties
      test
    target
    backend.iml
    pom.xml
    pom.xml.save
  frontend
    src
      main
        frontend
          e2e
          etc
```

```java
        }

            return builder.getRootCluster();
    }

    @Override
    public List<Cluster> performFlatClustering(double[][] distances,
                                                String[] clusterNames, LinkageStrategy linkageStrategy, Double
    {

        checkArguments(distances, clusterNames, linkageStrategy);
    /* Setup model */
        List<Cluster> clusters = createClusters(clusterNames);
        DistanceMap linkages = createLinkages(distances, clusters);

    /* Process */
        HierarchyBuilder builder = new HierarchyBuilder(clusters, linkages);
        return builder.flatAgg(linkageStrategy, threshold);
    }

    @Override
    public Cluster performClustering(List<Class> sourceClassesList, int alpha, int beta, int gamma, int delta
        return null;
    }

    private void checkArguments(double[][] distances, String[] clusterNames,
                                LinkageStrategy linkageStrategy)
    {
        if (distances == null || distances.length == 0
                || distances[0].length != distances.length)
        {
            throw new IllegalArgumentException("Invalid distance matrix");
        }
        if (distances.length != clusterNames.length)
```

Platform and Plugin Updates: IntelliJ IDEA is ready to update. (a minute ago)

Figure 4-2: Project Backend.

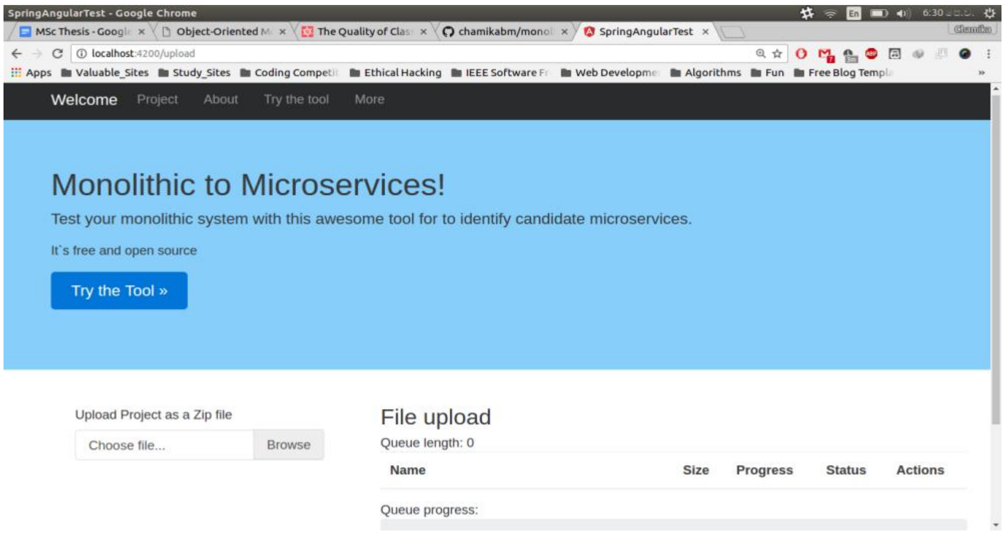
Both backend and frontend will be packaged into one project as above.

4.3.1 Frontend

According to the implementations of frontend users can upload a project file as a zip file to a server and then get processed the project. As a result, it will display set of candidate services along with their risk level, and importance.

Upload Project Files

Select project zip file from clicking the “Browse” button



Monolithic to Microservices!

Test your monolithic system with this awesome tool for to identify candidate microservices.

It`s free and open source

Upload Project as a Zip file

File upload

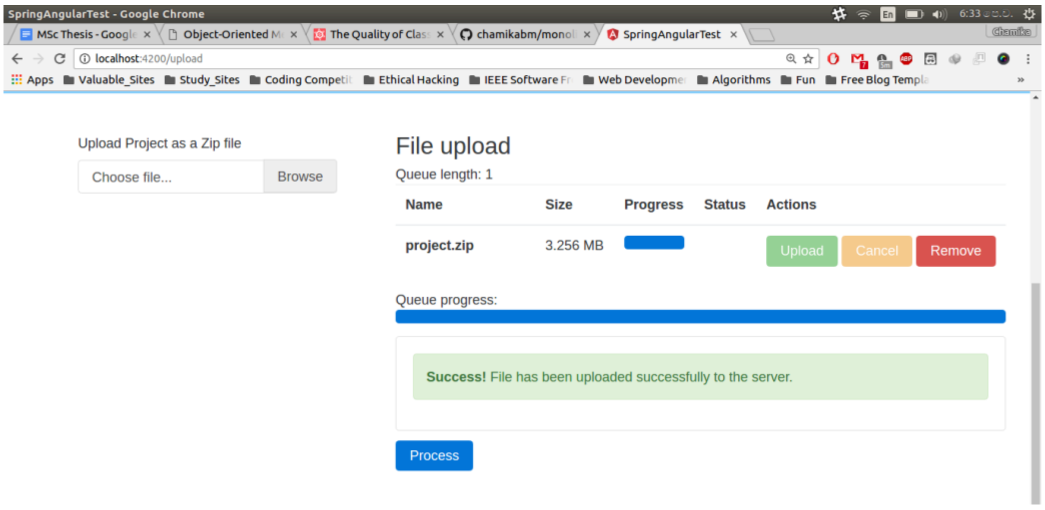
Queue length: 0

Name	Size	Progress	Status	Actions

Queue progress:

Figure 4-3: Project Frontend - File Upload.

After it will show upload button and click on upload button files will be upload to the server.



Upload Project as a Zip file

File upload

Queue length: 1

Name	Size	Progress	Status	Actions
project.zip	3.256 MB			Upload Cancel Remove

Queue progress:

Success! File has been uploaded successfully to the server.

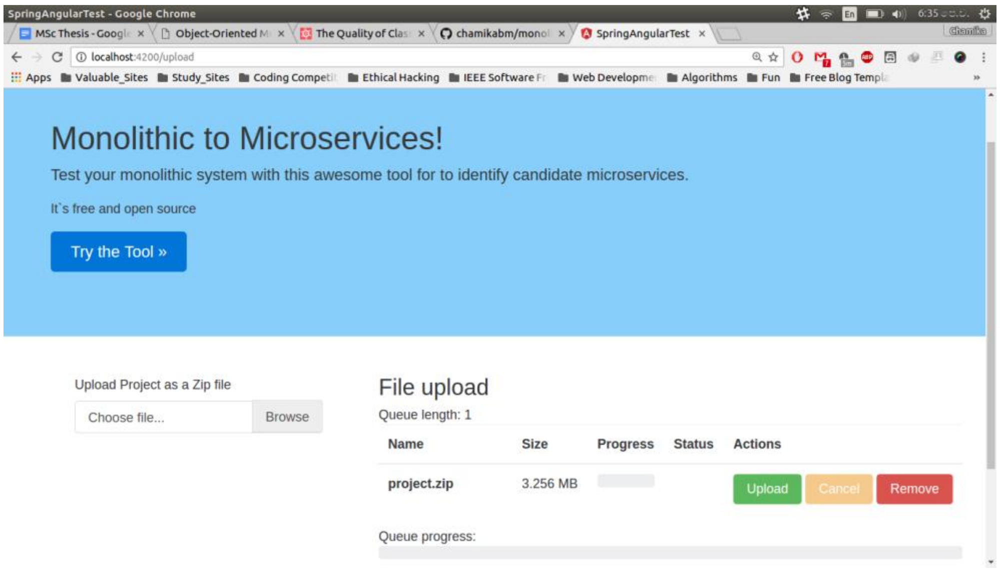
Figure 4-4: Project Frontend - Submit Files.

Process Project File

Once click on the Upload file, file will be uploaded to the service and it shows a processing button beneath the file upload success message as follows.

Results

Once after click on the process button it will start to process the project files on the server and return the set of microservices as follows along with its relevant risk level and the importance as a percentage.


Monolithic to Microservices!

Test your monolithic system with this awesome tool for to identify candidate microservices.

It`s free and open source

Upload Project as a Zip file

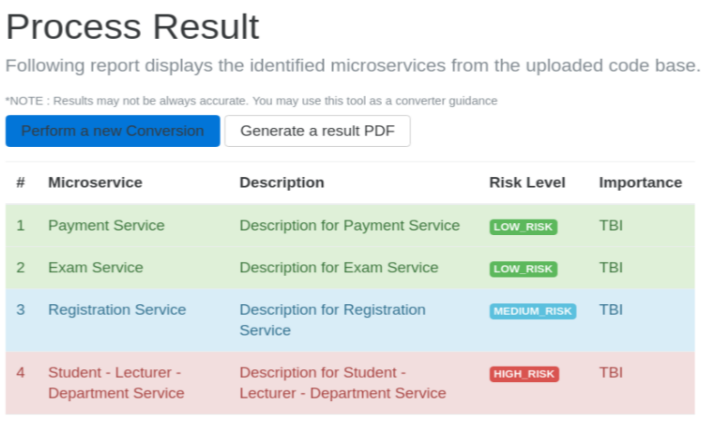
File upload

Queue length: 1

Name	Size	Progress	Status	Actions
project.zip	3.256 MB			Upload Cancel Remove

Queue progress:

Figure 4-5: Project Frontend - File Uploaded.


Process Result

Following report displays the identified microservices from the uploaded code base.

*NOTE : Results may not be always accurate. You may use this tool as a converter guidance

#	Microservice	Description	Risk Level	Importance
1	Payment Service	Description for Payment Service	LOW_RISK	TBI
2	Exam Service	Description for Exam Service	LOW_RISK	TBI
3	Registration Service	Description for Registration Service	MEDIUM_RISK	TBI
4	Student - Lecturer - Department Service	Description for Student - Lecturer - Department Service	HIGH_RISK	TBI

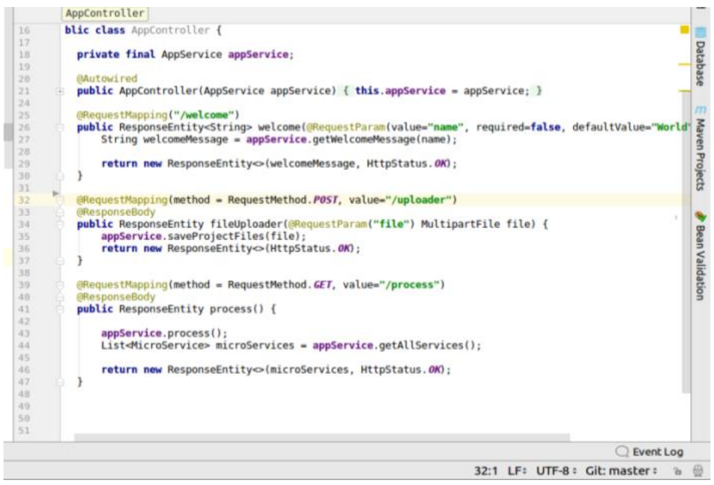
Figure 4-6: Project Frontend - Processed Results.

4.3.2 Backend

For this project backend. it serves as a REST API.

Controller

This is the REST controller which provides the endpoints for the frontend to access the services in the server. It mainly has 3 endpoints as follows.



```
AppController
blic class AppController {

  private final AppService appService;

  @Autowired
  public AppController(AppService appService) { this.appService = appService; }

  @RequestMapping("/welcome")
  public ResponseEntity<String> welcome(@RequestParam(value="name", required=false, defaultValue="World"
      String welcomeMessage = appService.getWelcomeMessage(name);

      return new ResponseEntity<>(welcomeMessage, HttpStatus.OK);
  }

  @RequestMapping(method = RequestMethod.POST, value="/uploader")
  @ResponseBody
  public ResponseEntity fileUploader(@RequestParam("file") MultipartFile file) {
      appService.saveProjectFiles(file);
      return new ResponseEntity<>(HttpStatus.OK);
  }

  @RequestMapping(method = RequestMethod.GET, value="/process")
  @ResponseBody
  public ResponseEntity process() {

      appService.process();
      List<MicroService> microServices = appService.getAllServices();

      return new ResponseEntity<>(microServices, HttpStatus.OK);
  }
```

Figure 4-7: Project Backend - Controllers.

First endpoint is to check the server health. And second endpoint can be used to upload the zip files to the server, finally the last endpoint will be used to process the uploaded files to get the candidate microservices.

Managers

Fitness Function Manager

This manager is used to calculate the fitness function values for the each of the candidate microservices. It used to calculate the functionality, composability, self-containment and usage of each of the services. Main functions of the manages are as follows:

Calculate Fitness Value:

```
public float calculateFitnessFunction (int alpha, int beta, int gamma, int
delta, MicroService microService) {
   float fitnessValue; int n = alpha + beta + gamma + delta;

   fitnessValue = (alpha*measureFunctionality(microService)+
                   beta*measureComposability(microService)+
                   gamma*measureSelfContainment(microService)+
                   delta*measureUsage(microService))/n;
   return fitnessValue;
}
```

Calculate functionality:

```
private float measureFunctionality (MicroService microService) {
   float functionalityValue;

   functionalityValue = (
           computeNumberOfProvidedInterfaces(microService)+
           measureComposability(microService)+
           computeCohesionBetweenInterfaces(microService)+
           computeInternalCoupling(microService)+
           computeLoseClassCohesion(microService))
           /5;

   return functionalityValue;
}
```

Calculate composability:

```
private float measureComposability (MicroService microService) {
   float comparabilityValue;

   int I = microService.getClasses().size();
   float                                              totalInternalCohesion=
computeCohesionWithInInterface(microService);

   comparabilityValue = totalInternalCohesion/I;

   return comparabilityValue;
}
```

Calculate Self-containment:

```
private float measureSelfContainment (MicroService microService) {
   float selfContainmentValue;

   selfContainmentValue = computeExternalCoupling(microService);

   return selfContainmentValue;
}
```

Calculate Usage:

```
private float measureUsage (MicroService microService) {
   float usageValue; float inheritanceFactor; float compositionFactor;

   inheritanceFactor = getInheritanceFactor(microService);
   compositionFactor = getCompositionFactor(microService);

   usageValue = inheritanceFactor + compositionFactor;

   return usageValue;
}
```

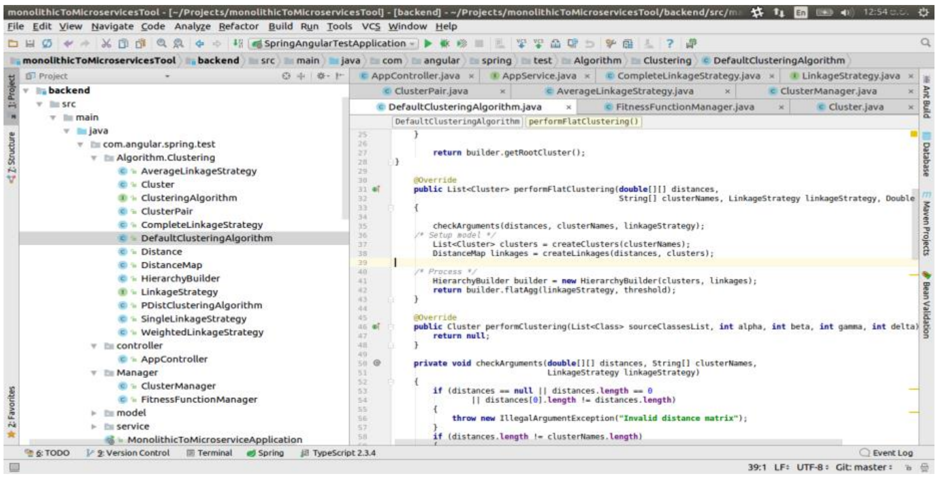
Cluster Manager

This manager is used to cluster the services into relevant clusters based on the fitness function value:

```
public class ClusterManager {
   //These values should be change according to the Architects needs.
   private final int alpha = 2;
   private final int beta = 3;
   private final int gamma = 4;
   private final int delta = 5;

   public        List<MicroService>        getMicroServiceCluster(List<Class>
sourceClassesList) {
       ClusteringAlgorithm        clusteringAlgorithm         =          new
DefaultClusteringAlgorithm ();
       Cluster cluster =
               clusteringAlgorithm.performClustering(sourceClassesList,
alpha, beta, gamma, delta);

       return cluster.getMicroServices();
   }
}
```

4.3.3 Algorithms

To identify the microservices implemented several functions in the algorithm sections where it used to generate the microservices clusters it is an open source code base extended from this project (https://github.com/lbehnke/hierarchical-clustering-java). This algorithm encompasses a basic implementation of hierarchical agglomerative clustering algorithm which can be used to cluster the services according to the provided fitness function.



```
    }

        return builder.getRootCluster();
}

    @Override
    public List<Cluster> performFlatClustering(double[][] distances,
                                               String[] clusterNames, LinkageStrategy linkageStrategy, Double
    {

        checkArguments(distances, clusterNames, linkageStrategy);
    /* Setup model */
        List<Cluster> clusters = createClusters(clusterNames);
        DistanceMap linkages = createLinkages(distances, clusters);

    /* Process */
        HierarchyBuilder builder = new HierarchyBuilder(clusters, linkages);
        return builder.flatAgg(linkageStrategy, threshold);
    }

    @Override
    public Cluster performClustering(List<Class> sourceClassesList, int alpha, int beta, int gamma, int delta)
        return null;
    }

    private void checkArguments(double[][] distances, String[] clusterNames,
                                LinkageStrategy linkageStrategy)
    {
        if (distances == null || distances.length == 0
                || distances[0].length != distances.length)
        {
            throw new IllegalArgumentException("Invalid distance matrix");
        }
        if (distances.length != clusterNames.length)
```

Figure 4-8: Project Backend - Algorithms.

Chapter 5
SYSTEM EVALUATION

5.1 Overview of Test Project

The proposed approach has been evaluated on one realistic case study which is the student data management Rest API. Source code for this case study project can be found here (https://github.com/chamikabm/StudentManagementRestApi). This API was completely developed from the scratch in java language using Spring Boot framework with Hibernate to support this project evaluation. This a rest API contains few basic Controllers such as App Controller, Student Controller, Lecturer Controller, Department Controller, Exam Controller, Payment Controller, Registration Controller. Following table will describe the main functionalities of those controllers.

#	Controller	Main Functionality
1	App Controller	Basic Controller of the API
2	Student Controller	Manages student related operations such as add / delete/ update students.
3	Lecturer Controller	Manages lecturer related operations such as add / delete/ update lecturers.
4	Department Controller	Manages department related operations such as add/remove/update departments.
5	Registration Controller	Manages student related registration functionalities such as register student/ remove registration of students.
6	Exam Controller	Manages exams related functions of students.
7	Payment Controller	Manages student related payment activities such as exam fees, course fees. etc.

Table 5-1: Test Project Functionalities.

Idea of this project to provide complete and comprehensive student management system to a private university. According to the implementations, this code base leverages the OOP concepts heavily. Related MySQL and Java file structure will be shown in coming subtopics.

5.1.1 DDL for Student Management API

```sql
DROP TABLE IF EXISTS `student`;
CREATE TABLE `student`
  (
     `id`            INT(11) NOT NULL auto_increment,
     `name`          VARCHAR(45) NOT NULL,
     `age`           INT(11) DEFAULT NULL,
     `contact_no`    VARCHAR(45) DEFAULT NULL,
     `address`       VARCHAR(45) DEFAULT NULL,
     `email`         VARCHAR(45) DEFAULT NULL,
     `department_id` INT(11) NOT NULL,
     `semester`      TINYINT(2) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_student_semester` (`semester`),
     INDEX `idx_student_department` (`department_id`)
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `lecturer`;
CREATE TABLE `lecturer`
  (
     `id`            INT(11) NOT NULL auto_increment,
     `name`          VARCHAR(45) NOT NULL,
     `age`           INT(11) DEFAULT NULL,
     `contact_no`    VARCHAR(45) DEFAULT NULL,
     `address`       VARCHAR(45) DEFAULT NULL,
     `email`         VARCHAR(45) DEFAULT NULL,
     `department_id` INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_lecturer_department` (`department_id`)
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `department`;
CREATE TABLE `department`
  (
     `id`                 INT(11) NOT NULL auto_increment,
     `name`               VARCHAR(45) NOT NULL,
     `department_head_id` INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     CONSTRAINT `fk_department_lecturer_department_head_id` FOREIGN KEY (
     `department_head_id`) REFERENCES lecturer(`id`) ON UPDATE no action ON
     DELETE no action
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `course`;
CREATE TABLE `course`
  (
     `id`                     INT(11) NOT NULL auto_increment,
     `name`                   VARCHAR(45) NOT NULL,
     `offering_department_id` INT(11) NOT NULL,
     `fee`                    DECIMAL(6, 2) NOT NULL DEFAULT 0.00,
     `lecturer_id`            INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_course_offering_department_id` (`offering_department_id`),
     INDEX `idx_course_lecturer_id` (`lecturer_id`),
     CONSTRAINT `fk_course_department_offering_department_id` FOREIGN KEY (
     `offering_department_id`) REFERENCES department(`id`) ON UPDATE no action
     ON DELETE no action
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;
```

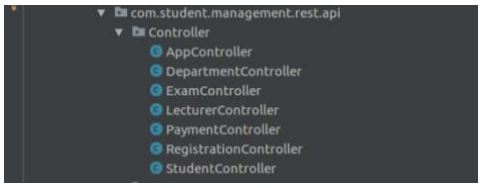
```
DROP TABLE IF EXISTS `student_course`;
CREATE TABLE `student_course`
  (
     `id`         INT(11) NOT NULL auto_increment,
     `student_id` INT(11) NOT NULL,
     `course_id`  INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_student_course_1` (`student_id`),
     INDEX `idx_student_course_2` (`course_id`, `student_id`),
     CONSTRAINT `fk_student_course_student_id` FOREIGN KEY (`student_id`)
     REFERENCES student(`id`) ON UPDATE no action ON DELETE no action,
     CONSTRAINT `fk_student_course_course_id` FOREIGN KEY (`course_id`)
     REFERENCES course(`id`) ON UPDATE no action ON DELETE no action
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `semester_course`;
CREATE TABLE `semester_course`
  (
     `id`        INT(11) NOT NULL auto_increment,
     `semester`  TINYINT(2) NOT NULL,
     `course_id` INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_semester_course_1` (`course_id`),
     INDEX `idx_semester_course_2` (`semester`, `course_id`),
     CONSTRAINT `fk_semester_course_course_id` FOREIGN KEY (`course_id`)
     REFERENCES course(`id`) ON UPDATE no action ON DELETE no action
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `exam`;
CREATE TABLE `exam`
  (
     `id`            INT(11) NOT NULL auto_increment,
     `name`          VARCHAR(45) NOT NULL,
     `department_id` INT(11) NOT NULL,
     `course_id`     INT(11) NOT NULL,
     `date`          DATE DEFAULT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_exam_department` (`department_id`)
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `student_exam`;
CREATE TABLE `student_exam`
  (
     `id`         INT(11) NOT NULL auto_increment,
     `student_id` INT(11) NOT NULL,
     `exam_id`    INT(11) NOT NULL,
     PRIMARY KEY (`id`),
     INDEX `idx_student_exam_1` (`student_id`),
     INDEX `idx_student_exam_2` (`student_id`, `exam_id`),
     CONSTRAINT `fk_student_exam_student_id` FOREIGN KEY (`student_id`)
     REFERENCES student(`id`) ON UPDATE no action ON DELETE no action,
     CONSTRAINT `fk_student_exam_exam_id` FOREIGN KEY (`exam_id`) REFERENCES
     exam(`id`) ON UPDATE no action ON DELETE no action
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;




DROP TABLE IF EXISTS `payment`;
```

```
CREATE TABLE `payment`
  (
     `id`           INT(11) NOT NULL auto_increment,
     `course_id`    INT(11) NOT NULL,
     `student_id`   INT(11) NOT NULL,
     `amount`       DECIMAL(6, 2) NOT NULL DEFAULT 0.00,
     `semester`     TINYINT(2) NOT NULL,
     `payment_date` DATE DEFAULT NULL,
     `status`       ENUM('PAID', 'PENDING', 'FREE') NOT NULL DEFAULT 'PENDING',
     PRIMARY KEY (`id`),
     INDEX `idx_payment_course_id` (`course_id`),
     INDEX `idx_payment_student_id` (`student_id`),
     INDEX `idx_payment_status` (`status`)
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;

DROP TABLE IF EXISTS `student_register`;
CREATE TABLE `student_register`
  (
     `id`              INT(11) NOT NULL auto_increment,
     `student_id`      INT(11) NOT NULL,
     `is_registered`   TINYINT(1) UNSIGNED NOT NULL DEFAULT 0,
     `registered_date` DATE DEFAULT NULL,
     PRIMARY KEY (`id`),
     CONSTRAINT `fk_student_register_student_id` FOREIGN KEY (`student_id`)
     REFERENCES student(`id`) ON UPDATE no action ON DELETE no action,
     INDEX `idx_student_register_1` (`student_id`, `is_registered`)
  )
engine=innodb
auto_increment=1
DEFAULT charset=utf8;
```

5.1.2 File Structure of Student Management API.

Controllers

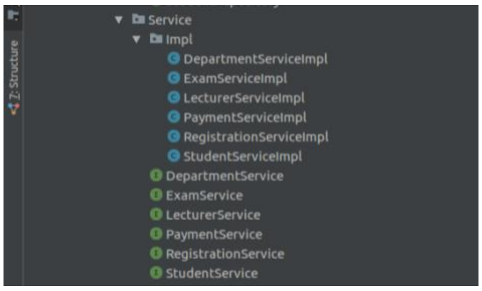
Controllers are the one that exposes the endpoints to the outside.



```
▼ com.student.management.rest.api
  ▼ Controller
      AppController
      DepartmentController
      ExamController
      LecturerController
      PaymentController
      RegistrationController
      StudentController
```

Services

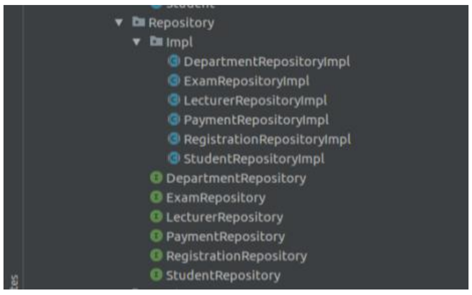
Services are the one that required services to the controller functions.



```
  ▼ Service
    ▼ Impl
        DepartmentServiceImpl
        ExamServiceImpl
        LecturerServiceImpl
        PaymentServiceImpl
        RegistrationServiceImpl
        StudentServiceImpl
      DepartmentService
      ExamService
      LecturerService
      PaymentService
      RegistrationService
      StudentService
```

Repositories

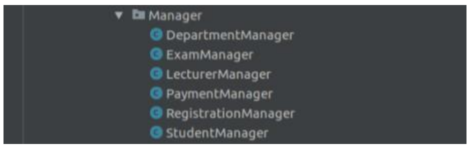
Repositories are the one that access the database layer based on the MySQL queries.



```
  ▼ Repository
    ▼ Impl
        DepartmentRepositoryImpl
        ExamRepositoryImpl
        LecturerRepositoryImpl
        PaymentRepositoryImpl
        RegistrationRepositoryImpl
        StudentRepositoryImpl
      DepartmentRepository
      ExamRepository
      LecturerRepository
      PaymentRepository
      RegistrationRepository
      StudentRepository
```

Managers

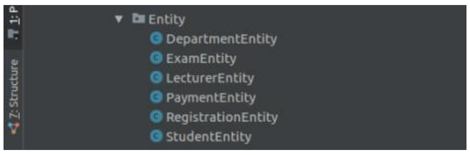
Managers will be mainly responsible for managing and manipulating data objects.



```
  ▼ Manager
      DepartmentManager
      ExamManager
      LecturerManager
      PaymentManager
      RegistrationManager
      StudentManager
```

Entities

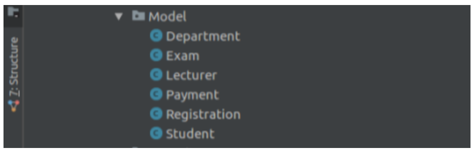
Those will be the DAO which used by repositories to access the MySQL database.



```
▼ Entity
    DepartmentEntity
    ExamEntity
    LecturerEntity
    PaymentEntity
    RegistrationEntity
    StudentEntity
```

Models

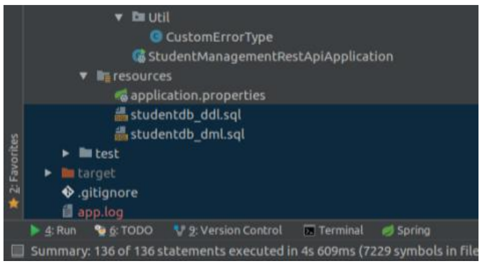
Those will be the responses to the outside from the API.



```
▼ Model
    Department
    Exam
    Lecturer
    Payment
    Registration
    Student
```

Utils

There are the utility classes which used as API utility functions and models.



```
      ▼ Util
          CustomErrorType
        StudentManagementRestApiApplication
  ▼ resources
      application.properties
      studentdb_ddl.sql
      studentdb_dml.sql
 ▸ test
▸ target
 .gitignore
 app.log
```

```
Summary: 136 of 136 statements executed in 4s 609ms (7229 symbols in file
```

As seen on the file structure there are separate folders dedicated for services, managers, repositories and utilities. Services are the type of classes that leverage the managers and repositories to serve the controller requests. Repositories mainly focused on performing CRUD operations on the MySQL database. As per the implementations there are dependencies on some of the services with managers and other repositories in order to cater some advanced functionalities for the API. For an instance in order to create a new user we need to leverage 3 services such as Student Service, Registration Service and Payment Service. That is when creating a student, it needs to make him/her a registered student and also need to create a PENDING payment for the semester. Along with that associated manager functions will validate and do the data manipulations to the data objects as required.

5.2 Microservice Identification

In order to identify the microservices in the code base which need to be verified (i.e Student Management API which explained the previous section.) uploaded to the tool as a .zip file which have created with the logics and the functionalities which describes in the Implementation section. (See 4.3 for more details.)

5.2.1 Results

This research was carried out for 3 iterations in order to find out most suitable values for the weighted coefficient values which were fed to the fitness function. There values should be defined by an experienced software architect in order to gain the maximum out of the method which used to identify microservice in the monolithic system. Here ran the tool for 3 times with different values to identify how the results can vary with the values and the suitable parameters to filter out the correct microservices from the system. Results and input values are as follows. Note that the cluster depth has set to 2 since this is a fairly small system.

Iteration 1: α =1, $\beta = 2$, $\gamma = 3$, δ=4

In this iteration it has given more weight to the usage of the services.

$[\alpha - Functionality, \beta - Composability, \gamma - Self\ Containment\ , \delta - Usage]$

Logs Related to the Process:

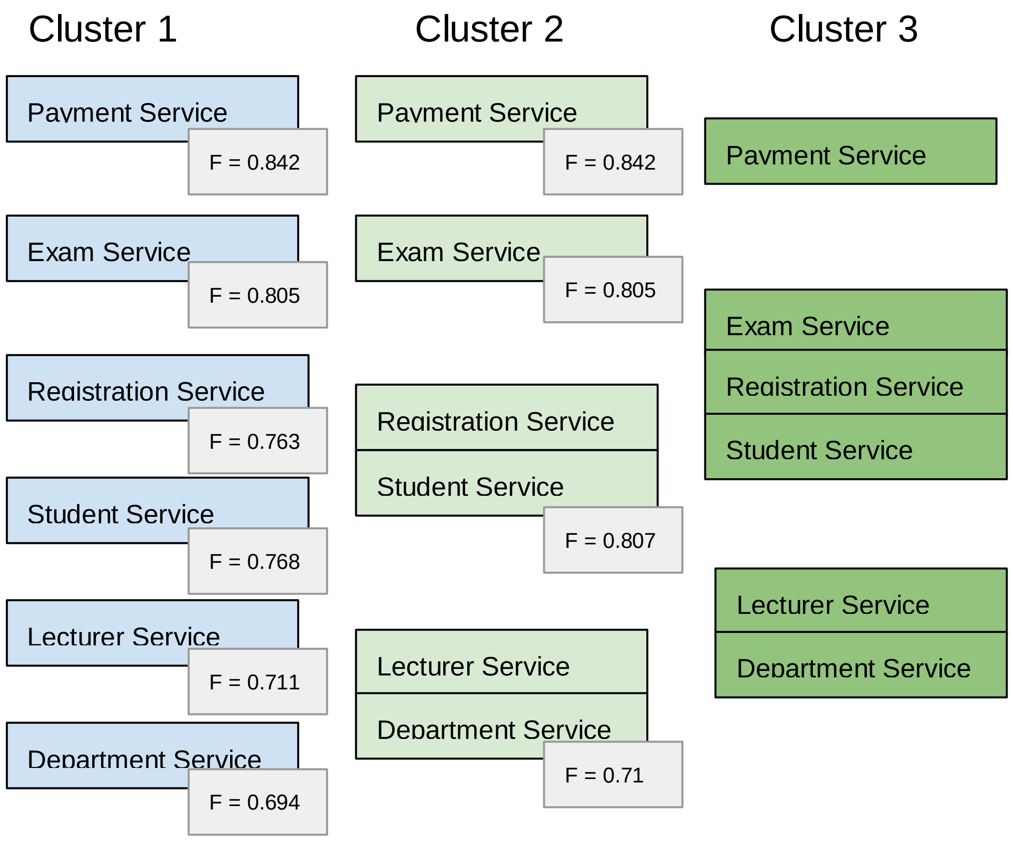
```
  2018-01-09 01:59:13.481 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
alpha=1, beta=2, gamma = 3, delta = 4.

 2018-01-09 01:59:13.491 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Clustering Process Starting.

 2018-01-09 01:59:13.492 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Total
service classes = 6.

 2018-01-09 01:59:13.492 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Clustering Process Starting.

 2018-01-09 01:59:13.493 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Calculating Values for Cluster 1.

 2018-01-09 01:59:13.556 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Calculating Functionality for Service : Payment Service.

 2018-01-09 01:59:13.557 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Functionality for Service - Payment Service: 0.75.

 2018-01-09 01:59:13.638 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager    : SMAPI -
Calculating Composability for Service - Payment Service.
```

```
  2018-01-09 01:59:15.068 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Payment Service : 0.89.

  2018-01-09 01:59:15.069 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Payment Service.

  2018-01-09 01:59:15.069 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Payment Service : 0.91.

  2018-01-09 01:59:15.819 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Payment Service.

  2018-01-09 01:59:15.838 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Payment Service : 0.79.

  2018-01-09 01:59:15.839 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Payment Service : 0.842.

  2018-01-09 01:59:15.931 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Functionality for Service : Exam Service.

  2018-01-09 01:59:15.932 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Exam Service: 0.8.

  2018-01-09 01:59:15.933 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Exam Service.

  2018-01-09 01:59:16.078 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Exam Service : 0.86.

  2018-01-09 01:59:16.083 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Exam Service.

  2018-01-09 01:59:16.084 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Exam Service : 0.83.

  2018-01-09 01:59:16.084 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Exam Service.

  2018-01-09 01:59:16.084 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Exam Service : 0.76.

  2018-01-09 01:59:16.116 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Exam Service : 0.805.

  2018-01-09 01:59:16.826 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Functionality for Service : Registration Service.

  2018-01-09 01:59:17.807 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Registration Service: 0.66.

  2018-01-09 01:59:18.708 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Registration Service.

  2018-01-09 01:59:18.818 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Registration Service : 0.71.

  2018-01-09 01:59:18.837 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Registration Service.

  2018-01-09 01:59:18.838 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Registration Service : 0.73.

  2018-01-09 01:59:18.846 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Registration Service.

  2018-01-09 01:59:18.847 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Registration Service : 0.84.

  2018-01-09 01:59:18.848 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Registration Service : 0.763.

  2018-01-09 01:59:18.851 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Functionality for Service : Student Service.

  2018-01-09 01:59:18.853 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Student Service: 0.76.

  2018-01-09 01:59:18.854 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Student Service.

  2018-01-09 01:59:18.854 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Student Service : 0.72.

  2018-01-09 01:59:18.864 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Student Service.
  2018-01-09 01:59:18.871 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Student Service : 0.80.

  2018-01-09  01:59:18.871  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Student Service.

  2018-01-09 01:59:18.873 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Student Service : 0.77.

  2018-01-09 01:59:18.873 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Student Service : 0.768.

  2018-01-09  01:59:18.881  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Lecturer Service.

  2018-01-09  01:59:18.882  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Lecturer Service : 0.72.

  2018-01-09  01:59:18.882  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Lecturer Service.

  2018-01-09 01:59:18.882 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Lecturer Service : 0.7.

  2018-01-09  01:59:18.883  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Lecturer Service.

  2018-01-09 01:59:18.883 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Lecturer Service : 0.71.

  2018-01-09 01:59:18.884 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Lecturer Service : 0.711.

  2018-01-09  01:59:18.885  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Department Service.

  2018-01-09  01:59:18.888  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Department Service: 0.74.

  2018-01-09  01:59:18.888  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Department Service.

  2018-01-09  01:59:18.889  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Department Department : 0.7.

  2018-01-09  01:59:18.890  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Department Service.

  2018-01-09 01:59:18.894 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Department Service : 0.72.

  2018-01-09  01:59:18.895  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Department Service.

  2018-01-09 01:59:18.897 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Department Service : 0.66.

  2018-01-09 01:59:18.898 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Department Service : 0.694.

  2018-01-09  01:59:18.899  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Clustering Related Services Started.

  2018-01-09 01:59:18.900 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Services
[Payment Service, Student Service+Registration Service, Lecturer Service + Department Service].

  2018-01-09  01:59:18.901  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Clustering Related Services Ended.

  2018-01-09 01:59:18.901 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - alpha=1,
beta=2, gamma = 3, delta = 4.

  2018-01-09  01:59:18.902  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Clustering Process Starting.

  2018-01-09 01:59:18.904 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Total
service classes = 4.

  2018-01-09  01:59:18.905  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Clustering Process Starting.

  2018-01-09  01:59:18.906  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Values for Cluster 2.

  2018-01-09  01:59:18.907  INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Payment Service.
```

```
  2018-01-09 01:59:18.908 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Payment Service: 0.75.

  2018-01-09 01:59:18.908 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Payment Service.

  2018-01-09 01:59:18.909 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Payment Service : 0.89.

  2018-01-09 01:59:18.911 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Payment Service.

  2018-01-09 01:59:18.913 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Payment Service : 0.91.

  2018-01-09 01:59:18.914 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Payment Service.

  2018-01-09 01:59:18.940 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Payment Service : 0.79.

  2018-01-09 01:59:18.941 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Payment Service : 0.842.

  2018-01-09 01:59:18.941 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Functionality for Service : Registration Service - Student Service.

  2018-01-09 01:59:18.942 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Registration Service - Student Service : 0.75.

  2018-01-09 01:59:18.942 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Registration Service - Student Service.

  2018-01-09 01:59:18.943 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Registration Service - Student Service : 0.71.

  2018-01-09 01:59:18.943 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Registration Service - Student Service.

  2018-01-09 01:59:18.943 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Registration Service - Student Service : 0.82.

  2018-01-09 01:59:18.944 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Registration Service - Student Service.

  2018-01-09 01:59:18.944 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Registration Service - Student Service : 0.86.

  2018-01-09 01:59:18.945 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Registration Service - Student Service : 0.807.

  2018-01-09 01:59:18.945 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Functionality for Service : Department Service - Lecturer Service.

  2018-01-09 01:59:18.945 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Functionality for Service - Department Service - Lecturer Service: 0.73.

  2018-01-09 01:59:18.946 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Composability for Service - Department Service - Lecturer Service.

  2018-01-09 01:59:18.946 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Composability for Service - Department Department - Lecturer Service: 0.71.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Self-Containment for Service - Department Service - Lecturer Service.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Department Service - Lecturer Service : 0.73.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Calculating Usage for Service - Department Service - Lecturer Service.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Department Service - Lecturer Service : 0.69.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Department Service - Lecturer Service : 0.71.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Clustering Related Services Started.

  2018-01-09 01:59:18.947 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Services [Payment Service, Exam Service + Registration Service + Student Service, Lecturer Service + Department Service].

  2018-01-09 01:59:18.948 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager     : SMAPI - Clustering Related Services Ended.
```

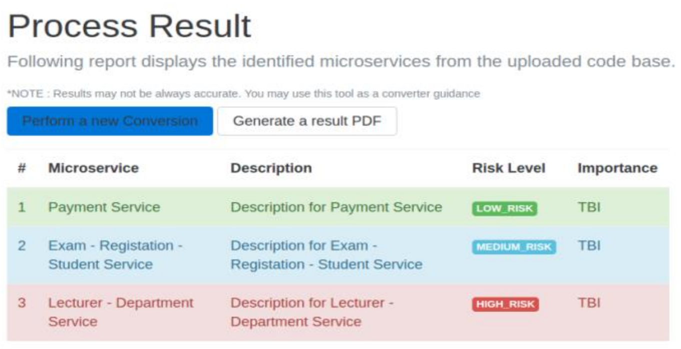
Clustering Data:

As seen in the above logs following diagram has drawn to understand how the clustering process has been executed by the tool. In the below diagram each of the Services boxes are tagged with relevant fitness function values which calculated by the tool and found from the above logs.



Tool Result:

Below picture shows how the identified services are displayed on the tool.



Process Result

Following report displays the identified microservices from the uploaded code base.

*NOTE : Results may not be always accurate. You may use this tool as a converter guidance

#	Microservice	Description	Risk Level	Importance
1	Payment Service	Description for Payment Service	LOW_RISK	TBI
2	Exam - Registation - Student Service	Description for Exam - Registation - Student Service	MEDIUM_RISK	TBI
3	Lecturer - Department Service	Description for Lecturer - Department Service	HIGH_RISK	TBI

Figure 5-1: Iteration 1 Final Result

Iteration 2: α =4, $\beta = 2$, $\gamma = 3$, δ=1

In this iteration is has given more weight to the functionality

$[\alpha - Functionality, \beta - Composability, \gamma - Self\ Containment\ , \delta - Usage]$

Logs Related to the Process:

```
2018-01-09 01:59:18.948  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - alpha=4,
beta=2, gamma = 3, delta = 1.

2018-01-09 01:59:18.948  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager  : SMAPI - Clustering
Process Starting.

2018-01-09 01:59:18.948  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Total
service classes = 6.

2018-01-09 01:59:18.948  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager  : SMAPI - Clustering
Process Starting.

2018-01-09  01:59:18.948   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Values for Cluster 1.

2018-01-09  01:59:18.948   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Payment Service.

2018-01-09  01:59:18.948   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Payment Service: 0.75.

2018-01-09  01:59:18.948   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Payment Service.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Payment Service : 0.89.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Payment Service.

2018-01-09 01:59:18.949  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Payment Service : 0.91.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Payment Service.

2018-01-09 01:59:18.949  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Payment Service : 0.79.

2018-01-09 01:59:18.949  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Payment Service : 0.832.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Exam Service.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Exam Service: 0.8.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Exam Service.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Exam Service : 0.86.

2018-01-09  01:59:18.949   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Exam Service.

2018-01-09 01:59:18.949  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Exam Service : 0.83.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Exam Service.

2018-01-09 01:59:18.950  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Exam Service : 0.76.

2018-01-09 01:59:18.950  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Exam Service : 0.697.
```

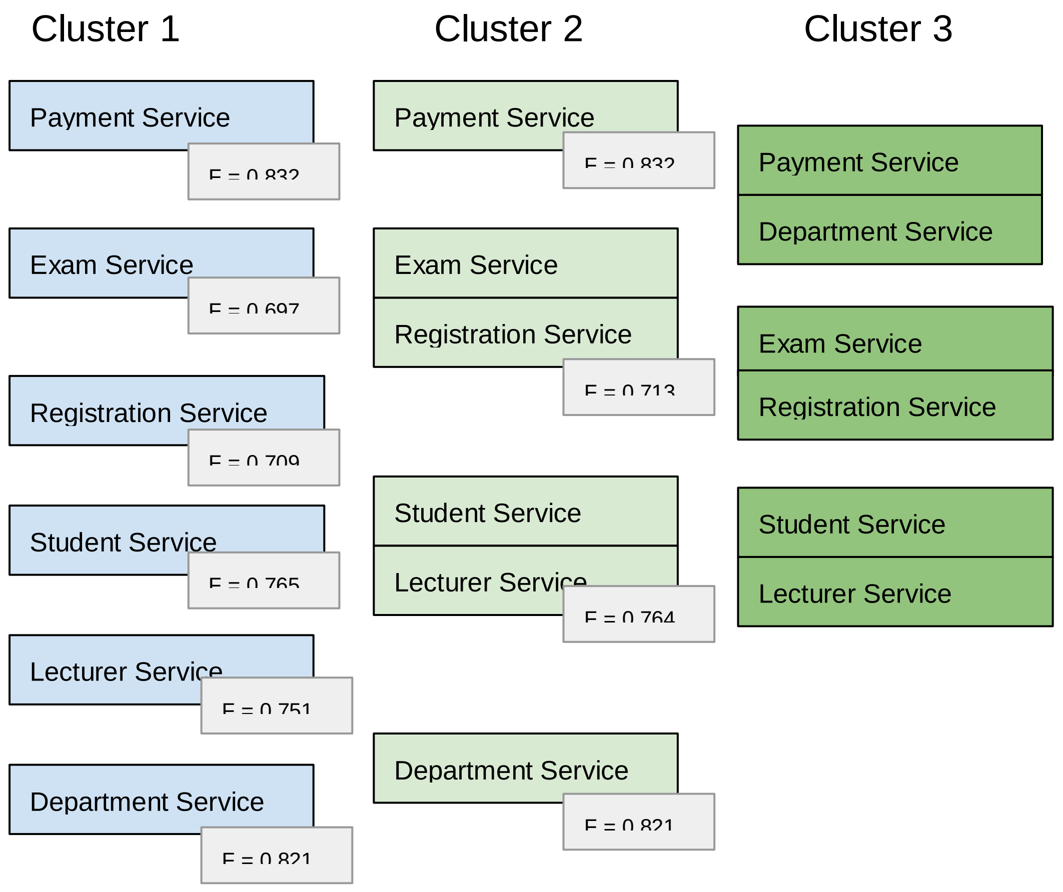
```
2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Registration Service.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Registration Service: 0.66.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Registration Service.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Registration Service : 0.71.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Registration Service.

2018-01-09 01:59:18.950  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Registration Service : 0.73.

2018-01-09  01:59:18.950   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Registration Service.

2018-01-09 01:59:18.951  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Registration Service : 0.84.

2018-01-09 01:59:18.951  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Registration Service : 0.709.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Student Service.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Student Service: 0.76.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Student Service.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Student Service : 0.72.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Student Service.

2018-01-09 01:59:18.951  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Student Service : 0.80.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Student Service.

2018-01-09 01:59:18.951  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Student Service : 0.77.

2018-01-09 01:59:18.951  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Student Service : 0.765.

2018-01-09  01:59:18.951   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Lecturer Service.

2018-01-09  01:59:18.952   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Lecturer Service: 0.73.

2018-01-09  01:59:18.952   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Lecturer Service.

2018-01-09  01:59:18.960   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Lecturer Service : 0.72.

2018-01-09  01:59:18.961   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Lecturer Service.

2018-01-09 01:59:18.962  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Lecturer Service : 0.7.

2018-01-09  01:59:18.962   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Lecturer Service.

2018-01-09 01:59:18.962  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Lecturer Service : 0.71.

2018-01-09 01:59:18.962  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Lecturer Service : 0.75.

2018-01-09  01:59:18.962   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Department Service.

2018-01-09  01:59:18.962   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Department Service: 0.74.
```

```
2018-01-09 01:59:18.962 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Department Service.

2018-01-09 01:59:18.962 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Composability for Service - Department Department : 0.7.

2018-01-09 01:59:18.962 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Self-Containment for Service - Department Service.

2018-01-09 01:59:18.962 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Self-Containment for Service - Department Service : 0.72.

2018-01-09 01:59:18.962 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Usage for Service - Department Service.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Department Service : 0.66.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Department Service : 0.821.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Related Services Started.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Services [Payment Service, Exam Service + Registration Service, Student Service + Lecturer Service, Department Service].

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Related Services Ended.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - alpha=4, beta=2, gamma = 3, delta = 1.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Process Starting.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Total service classes = 4.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Process Starting.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Values for Cluster 2.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Functionality for Service : Payment Service.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Functionality for Service - Payment Service: 0.75.

2018-01-09 01:59:18.963 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Payment Service.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Composability for Service - Payment Service : 0.89.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Self-Containment for Service - Payment Service.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Self-Containment for Service - Payment Service : 0.91.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Usage for Service - Payment Service.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Payment Service : 0.79.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Payment Service : 0.832.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Functionality for Service : Registration Service - Student Service.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Functionality for Service - Exam Service - Registration Service : 0.75.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Exam Service - Registration Service.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Composability for Service - Exam Service - Registration Service : 0.71.

2018-01-09 01:59:18.964 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Self-Containment for Service - Exam Service - Registration Service.
```

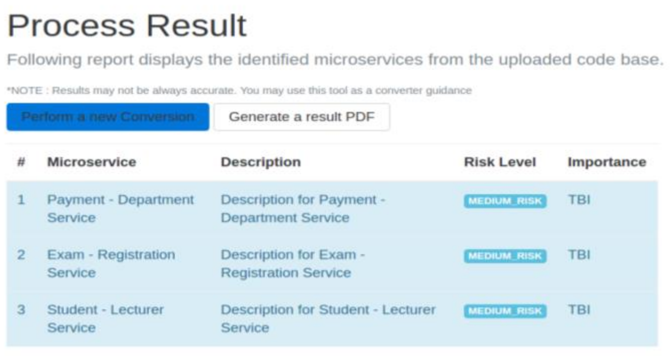
```
2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Exam Service - Registration Service : 0.82.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Usage for Service - Exam Service - Registration Service.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Exam Service - Registration Service : 0.86.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Exam Service - Registration Service : 0.713.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Functionality for Service : Student Service - Lecturer Service.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Functionality for Service - Student Service - Lecturer Service: 0.73.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Composability for Service - Student Service - Lecturer Service.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Composability for Service - Department Department - Lecturer Service: 0.71.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Self-Containment for Service - Student Service - Lecturer Service.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Student Service - Lecturer Service : 0.73.

2018-01-09 01:59:18.965  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Usage for Service - Student Service - Lecturer Service.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Student Service - Lecturer Service : 0.69.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Student Service - Lecturer Service : 0.71.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Functionality for Service : Department Service.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Functionality for Service - Department Service: 0.74.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Composability for Service - Department Service.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Composability for Service - Department Department : 0.7.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Self-Containment for Service - Department Service.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Department Service : 0.72.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Usage for Service - Department Service.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Department Service : 0.66.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Department Service : 0.821.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Related Services Started.

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Services [Payment Service + Department Service, Exam Service + Registration Service, Student Service + Lecturer Service].

2018-01-09 01:59:18.966  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Related Services Ended.

2018-01-09 01:59:18.967  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Controller- Clustering Process Ended.
```

Clustering Data:

As seen in the above logs following diagram has drawn to understand how the clustering process has been executed by the tool. In the below diagram each of the Services boxes are tagged with relevant fitness function values which calculated by the tool and found from the above logs.



Tool Result:



Process Result

Following report displays the identified microservices from the uploaded code base.

*NOTE : Results may not be always accurate. You may use this tool as a converter guidance

#	Microservice	Description	Risk Level	Importance
1	Payment - Department Service	Description for Payment - Department Service	MEDIUM_RISK	TBI
2	Exam - Registration Service	Description for Exam - Registration Service	MEDIUM_RISK	TBI
3	Student - Lecturer Service	Description for Student - Lecturer Service	MEDIUM_RISK	TBI

Figure 5-2: Iteration 2 Final Result.

Iteration 3: α =2, $\beta = 3, \gamma = 4$, δ=1

In this iteration is has given more weight to the self-containment

$[\alpha - Functionality, \beta - Composability, \gamma - Self\ Containment\ , \delta - Usage]$

Logs Related to the Process:

```
2018-01-14 22:22:09.577  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - alpha=2,
beta=3, gamma = 4, delta = 1.

2018-01-14 22:22:09.594  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager  : SMAPI - Clustering
Process Starting.

2018-01-14 22:22:09.595  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Total
service classes = 6.

2018-01-14 22:22:09.596  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager  : SMAPI - Clustering
Process Starting.

2018-01-14  22:22:09.597   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Values for Cluster 1.

2018-01-14  22:22:09.703   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Payment Service.

2018-01-14  22:22:09.703   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Payment Service: 0.75.

2018-01-14  22:22:09.831   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Payment Service.

2018-01-14  22:22:11.507   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Payment Service : 0.89.

2018-01-14  22:22:11.511   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Payment Service.

2018-01-14 22:22:11.512  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Payment Service : 0.91.

2018-01-14  22:22:12.192   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Payment Service.

2018-01-14 22:22:12.214  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Payment Service : 0.79.

2018-01-14 22:22:12.216  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Payment Service : 0.901.

2018-01-14  22:22:12.349   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Exam Service.

2018-01-14  22:22:12.351   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Exam Service: 0.8.

2018-01-14  22:22:12.353   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Exam Service.

2018-01-14  22:22:12.599   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Exam Service : 0.86.

2018-01-14  22:22:12.617   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Exam Service.

2018-01-14 22:22:12.618  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Exam Service : 0.83.

2018-01-14  22:22:12.618   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Exam Service.

2018-01-14 22:22:12.618  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Exam Service : 0.76.

2018-01-14 22:22:12.684  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Exam Service : 0.831.

2018-01-14  22:22:13.832   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Registration Service.
```

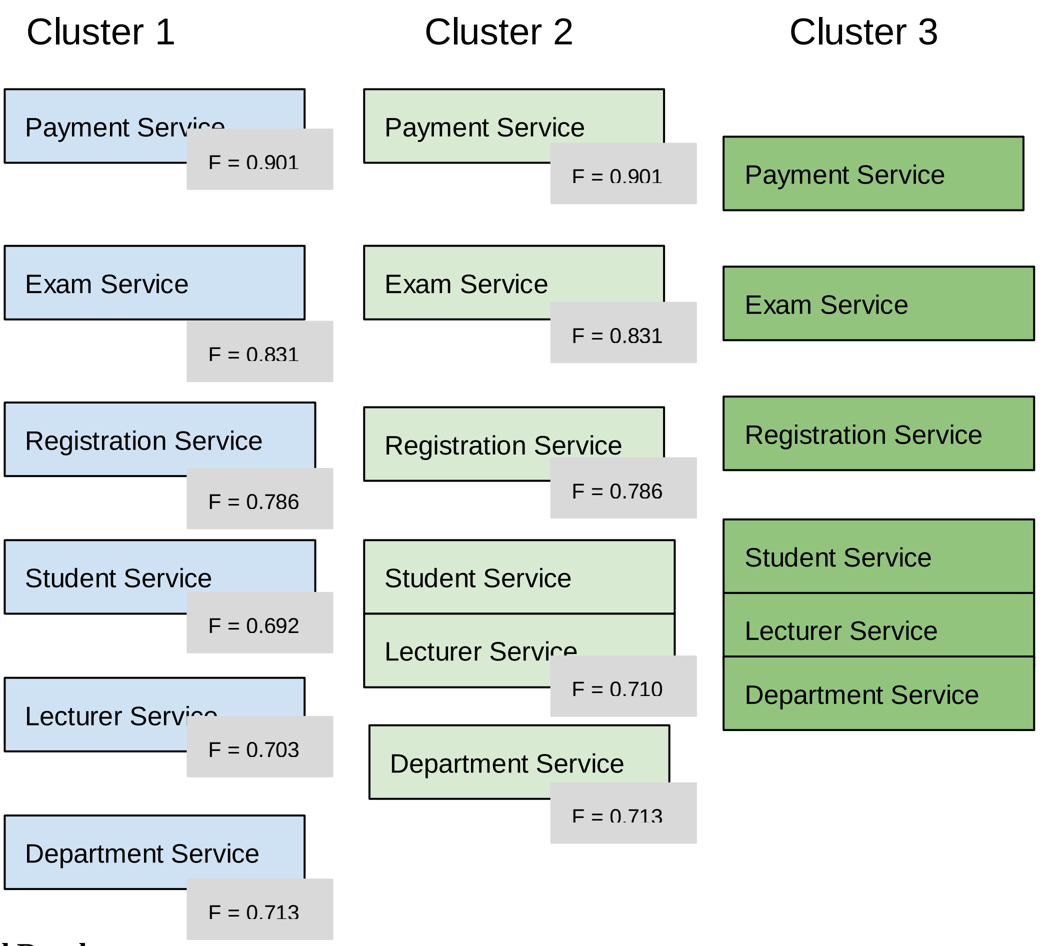
```
2018-01-14  22:22:14.927   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Registration Service: 0.66.

2018-01-14  22:22:16.020   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Registration Service.

2018-01-14  22:22:16.537   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Registration Service : 0.71.

2018-01-14  22:22:16.554   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Registration Service.

2018-01-14 22:22:16.555  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Registration Service : 0.73.

2018-01-14  22:22:16.559   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Registration Service.

2018-01-14 22:22:16.559  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Registration Service : 0.84.

2018-01-14 22:22:16.560  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Registration Service : 0.786.

2018-01-14  22:22:16.561   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Student Service.

2018-01-14  22:22:16.561   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Student Service: 0.76.

2018-01-14  22:22:16.562   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Student Service.

2018-01-14  22:22:16.562   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Student Service : 0.72.

2018-01-14  22:22:16.565   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Student Service.

2018-01-14 22:22:16.566  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Student Service : 0.80.

2018-01-14  22:22:16.566   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Student Service.

2018-01-14 22:22:16.566  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Student Service : 0.77.

2018-01-14 22:22:16.574  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Student Service : 0.692.

2018-01-14  22:22:16.575   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Lecturer Service.

2018-01-14  22:22:16.576   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Lecturer Service: 0.73.

2018-01-14  22:22:16.591   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Lecturer Service.

2018-01-14  22:22:16.592   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Composability for Service - Lecturer Service : 0.72.

2018-01-14  22:22:16.598   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Self-Containment for Service - Lecturer Service.

2018-01-14 22:22:16.598  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-
Containment for Service - Lecturer Service : 0.7.

2018-01-14  22:22:16.599   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Usage for Service - Lecturer Service.

2018-01-14 22:22:16.599  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage
for Service - Lecturer Service : 0.71.

2018-01-14 22:22:16.599  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness
function value for Service - Lecturer Service : 0.703.

2018-01-14  22:22:16.608   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Functionality for Service : Department Service.

2018-01-14  22:22:16.609   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Functionality for Service - Department Service: 0.74.

2018-01-14  22:22:16.609   INFO  7596  ---  [http-nio-8080-exec-1]  c.s.m.r.a.Manager.ClusterManager     :  SMAPI  -
Calculating Composability for Service - Department Service.
```

```
2018-01-14 22:22:16.613  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Composability for Service - Department Department : 0.7.

2018-01-14 22:22:16.614  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Self-Containment for Service - Department Service.

2018-01-14 22:22:16.617  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Department Service : 0.72.

2018-01-14 22:22:16.617  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Usage for Service - Department Service.

2018-01-14 22:22:16.626  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Department Service : 0.66.

2018-01-14 22:22:16.626  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Department Service : 0.713.

2018-01-14 22:22:16.626  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Related Services Started.

2018-01-14 22:22:16.627  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Services [Payment Service, Exam Service + Registration Service, Student Service + Lecturer Service, Department Service].

2018-01-14 22:22:16.627  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Related Services Ended.

2018-01-14 22:22:16.629  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - alpha=2, beta=3, gamma = 4, delta = 1.

2018-01-14 22:22:16.630  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Process Starting.

2018-01-14 22:22:16.637  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Total service classes = 5.

2018-01-14 22:22:16.637  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Clustering Process Starting.

2018-01-14 22:22:16.638  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Values for Cluster 2.

2018-01-14 22:22:16.638  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Functionality for Service : Payment Service.

2018-01-14 22:22:16.638  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Functionality for Service - Payment Service: 0.75.

2018-01-14 22:22:16.639  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Composability for Service - Payment Service.

2018-01-14 22:22:16.640  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Composability for Service - Payment Service : 0.89.

2018-01-14 22:22:16.650  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Self-Containment for Service - Payment Service.

2018-01-14 22:22:16.650  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Payment Service : 0.91.

2018-01-14 22:22:16.653  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Usage for Service - Payment Service.

2018-01-14 22:22:16.689  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Usage for Service - Payment Service : 0.79.

2018-01-14 22:22:16.690  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Fitness function value for Service - Payment Service : 0.901.

2018-01-14 22:22:16.690  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Functionality for Service : Exam Service.

2018-01-14 22:22:16.691  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Functionality for Service - Exam Service: 0.8.

2018-01-14 22:22:16.691  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Composability for Service - Exam Service.

2018-01-14 22:22:16.691  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Composability for Service - Exam Service : 0.86.

2018-01-14 22:22:16.703  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Calculating Self-Containment for Service - Exam Service.

2018-01-14 22:22:16.704  INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager   : SMAPI - Self-Containment for Service - Exam Service : 0.83.
```

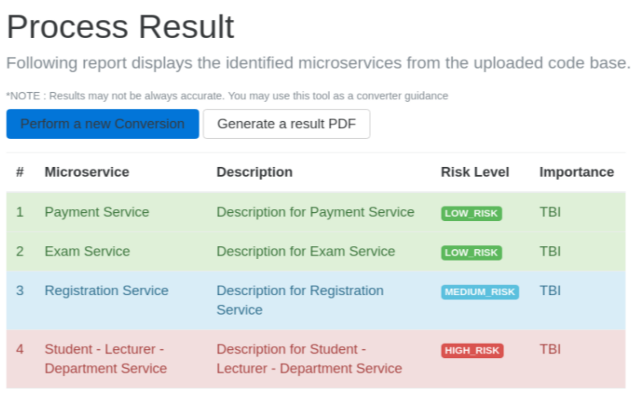
```
2018-01-14 22:22:16.704 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Usage for Service - Exam Service.

2018-01-14 22:22:16.704 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Exam Service : 0.76.

2018-01-14 22:22:16.704 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Exam Service : 0.831.

2018-01-14 22:22:16.704 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Functionality for Service : Registration Service.

2018-01-14 22:22:16.704 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Functionality for Service - Registration Service: 0.66.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Registration Service.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Composability for Service - Registration Service : 0.71.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Self-Containment for Service - Registration Service.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Self-Containment for Service - Registration Service : 0.73.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Usage for Service - Registration Service.

2018-01-14 22:22:16.705 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Registration Service : 0.84.

2018-01-14 22:22:16.709 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Registration Service : 0.786.

2018-01-14 22:22:16.716 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Functionality for Service - Student Service - Lecturer Service: 0.73.

2018-01-14 22:22:16.717 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Student Service - Lecturer Service.

for Service - Student Service - Lecturer Service.

2018-01-14 22:22:16.720 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Student Service - Lecturer Service : 0.69.

2018-01-14 22:22:16.720 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Student Service - Lecturer Service : 0.71.

2018-01-14 22:22:16.721 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Functionality for Service : Department Service.

2018-01-14 22:22:16.721 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Functionality for Service - Department Service: 0.74.

2018-01-14 22:22:16.722 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Composability for Service - Department Service.

2018-01-14 22:22:16.722 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Composability for Service - Department Department : 0.7.

2018-01-14 22:22:16.722 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Self-Containment for Service - Department Service.

2018-01-14 22:22:16.723 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Self-Containment for Service - Department Service : 0.72.

2018-01-14 22:22:16.723 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Calculating Usage for Service - Department Service.

2018-01-14 22:22:16.724 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Usage for Service - Department Service : 0.66.

2018-01-14 22:22:16.724 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Fitness function value for Service - Department Service : 0.713.

2018-01-14 22:22:16.724 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Related Services Started.

2018-01-14 22:22:16.724 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Services [Payment Service, Exam Service, Registration Service, Student Service + Lecturer Service + Department Service].

2018-01-14 22:22:16.724 INFO 7596 --- [http-nio-8080-exec-1] c.s.m.r.a.Manager.ClusterManager : SMAPI - Clustering Related Services Ended.
```

Clustering Data:

As seen in the above logs following diagram has drawn to understand how the clustering process has been executed by the tool. In the below diagram each of the Services boxes are tagged with relevant fitness function values which calculated by the tool and found from the above logs.



Tool Result:



Process Result

Following report displays the identified microservices from the uploaded code base.

*NOTE : Results may not be always accurate. You may use this tool as a converter guidance

#	Microservice	Description	Risk Level	Importance
1	Payment Service	Description for Payment Service	LOW_RISK	TBI
2	Exam Service	Description for Exam Service	LOW_RISK	TBI
3	Registration Service	Description for Registration Service	MEDIUM_RISK	TBI
4	Student - Lecturer - Department Service	Description for Student - Lecturer - Department Service	HIGH_RISK	TBI

Figure 5-3: Iteration 3 Final Result.

5.3 Discussion and Validation

5.3.1 Discussion

From the results obtain from the tool from each iteration gives very good understanding about how the each of the identified quality attributes are important to identify the micro services. In order to classify existing monolithic system into microservices we first identified 4 major quality characteristics such as functionality, composability, self-containment and usage. Related to those characteristics which has identified through a thorough research [45] [46] [47] as discussed in implementation section of this research paper ran the tool against those characteristics by changing the weighted value. Mainly ran this tool under 3 configurations set even though there can be many variations. Those will be the highly impacted configurations as discussed in many research papers and supervisors. Hence ran the tool under following configurations in order to identify the microservices for the input monolithic system as well as the matching configurations.

Iteration 1:

For the iteration 1 ran the tool with following configuration.

Configurations: α =1, $\beta = 2, \gamma = 3$, δ=4

$[\alpha - Functionality, \beta - Composability, \gamma - Self\ Containment\ , \delta - Usage]$
After running the tool and according to the collected data it has given following results as a summary.

Result Summary:

Service	Merged Services - Cluster 2	Fitness Function Value
1	Payment Service	0.842
2	Exam Service	0.805
	Register Service	0.807
	Student Service	
3	Lecturer Service	0.71
	Department Service	

Table 5-2: Iteration 1 Final Results Cluster Values.

When we increase the bias for the usage of initial services the end results from the tool according to the implementation of Student Management API it has given a result more bias to the service usages. In this case it is harder to remove those as services as they have much more dependencies.

Iteration 2:

For the iteration 2 ran the tool with following configuration.

Configurations:

α =4, $\beta\ = 2, \gamma\ = 3, \delta$=1
$[\alpha - Functionality, \beta\ - Composability, \gamma\ - Self\ Containment\ , \delta - Usage]$

After running the tool and according to the collected data it has given following results as a summary.

Result Summary:

Service	**Merged Services**	**Fitness Function Value**
1	Payment Service	0.832
	Department Service	0.821
2	Exam Service	0.713
	Register Service	
3	Student Service	0.764
	Lecturer Service	

Table 5-3: Iteration 2 Final Results Cluster Values.

When we increase the bias for the functionality of initial services the end results from the tool according to the implementation of Student Management API it has given a result more bias to the service functionality.

Iteration 3:

For the iteration 3 ran the tool with following configuration.

Configurations:

α =4, $\beta = 2$, $\gamma = 3$, δ=1
$[\alpha - Functionality, \beta - Composability, \gamma - Self\ Containment\ , \delta - Usage]$

After running the tool and according to the collected data it has given following results as a summary.

Result Summary:

Service	**Merged Services**	**Fitness Function Value**
1	Payment Service	0.901
2	Exam Service	0.831
3	Register Service	0.786
4	Student Service	0.710
	Lecturer Service	
	Department Service	0.713

Table 5-4: Iteration 3 Final Results Cluster Values.

When we increase the bias for the Self-Containment of initial services the end results from the tool according to the implementation of Student Management API it has given a result more bias to the service isolatability. Which means these services can be isolated very easily.

Risk Level

In addition to the above results from the tool it shows additional information such as ***Service Description***, ***Risk Level*** and ***Importance***. For this research scope it will only implement the ***Risk Level***. It is an indicator which show the dependency of this service other services. From the qualities point of view for a service it will be the ***Self-Containment*** value. In order to decide the level from the tool it has device that value into 4 groups as follows.

Risk Level	Value Margins
High Risk	< 0.75
Medium Risk	0.75 - 0.80
Low Risk	0.80 - 0.95
No Risk	> 0.95

Table 5-5: Risk Level Value Margins.

5.3.2 Validation

In order to validate the result obtained from the method used by this research to identify microservices in the monolithic system can be divided into two streams.

1. **Manual Approach**
   - This approach will be the most suitable and reliable approach since a person or set of persons whose having good understanding about the microservices and the system architecture of the system which going to analyzed can give more correct prediction to the outcome.
2. **Automated Approach**
   - This approach is not the most reliable since this also a tool or a method which will be used to validate another tool or a method. But using this method can validate the outcome to some level.

Manual Approach

As explained briefly above about this method in order to validate the methodology which used to identify the microservice from the existing monolithic system, will manually predict the microservices in the monolithic system by looking at the current system architecture and the implementation. Following figure will give more insights about the monolithic system architecture of Student Data Management API related to the controllers, services and its dependencies. (NOTE: The arrows in the diagram shows its being used by the pointed entity, e.g.: Department Controller uses Department Service)

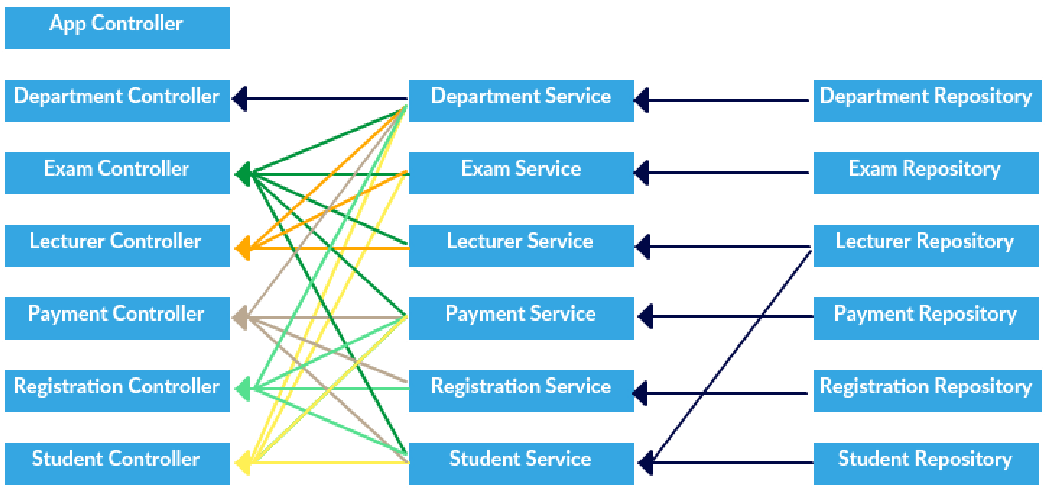


Figure 5-4: System Dependencies for Repositories.

As depicted above in the above picture many of the controllers have dependencies to many of the services in the system. And also, this architecture has many dependencies from services to managers as follows (See figure 5-5).

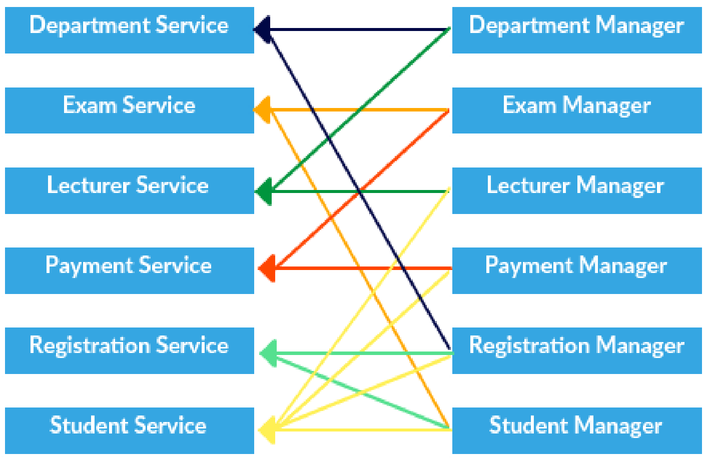


Figure 5-5: System Dependencies for Managers.

According to the above architecture and applying the knowledge on microservices it can say that this system can be developed as a microservice architecture as follows.

1. Student Service + Lecture Service (Student - Lecture Service)
2. Department Service
3. Exam Service
4. Registration Service
5. Payment Service

Because the functionality vise Student Service and Lecture Service have similarity when looking at the architecture and the code base. As well as other services can be developed as separate services as they much not depend on other service. Hence according to the manual validation process the results obtained on the ***iteration 3*** is more relevant and accurate comparing to the other iterations results. That is because in the iteration 3 it has given more weight to the ***Self-Containment*** and ***Cohesion*** which are the main aspects or the qualities when developing the microservice architecture. By looking at those results it can clearly notice that the there is a direct influence from those qualities when identifying the microservices from the monolithic system by using its quality attributes. By looking at the results show that close to 80% of extracted microservices from the tool are accurate. It can be assumed that by feeding context related knowledge to the tool it can be more accurate than this, but it is too hard to implement for this research with the given time frame.

Automated Approach

This approach is not most accurate as the manual approach since this also a using a concept to validate the results obtained from another tool. The idea of this method is to validate the results we obtained from the tool by using the logs that have added to the Student Data Management API when executing few common functions. Any after gathering those logs new tool which created to analyze the logs from the Student Data Management API will analyze how frequent those services and related manager functions have used to accomplish those tasks.

Tasks which will be taken to analyze:

1. Add a Student
2. Add a Lecturer
3. Add a Department
4. Add an Exam
5. Make payment for an exam

(Log Analyzing Tool: https://github.com/chamikabm/logAnalizerForSMAPI)

Logs for above operations:

```
2018-01-15  20:06:36.930  DEBUG  3376  ---  [http-nio-8080-exec-1]  o.s.web.servlet.DispatcherServlet              :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/student/all]

2018-01-15 20:06:36.934 DEBUG 3376 --- [http-nio-8080-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /student/all

2018-01-15 20:06:36.936 DEBUG 3376 --- [http-nio-8080-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                                                method                                                [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Student>>
com.student.management.rest.api.Controller.StudentController.listAllStudents()]

2018-01-15  20:06:36.937  DEBUG  3376  ---  [http-nio-8080-exec-1]  o.s.web.servlet.DispatcherServlet          :  Last-
Modified value for [/studentapi/student/all] is: -1

2018-01-15 20:06:36.959  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.Controller.StudentController   : SMAPI -
Student - Controller- listAllStudents request received.

2018-01-15 20:06:36.959  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.S.Impl.StudentServiceImpl      : SSMAPI -
tudent - Service- findAllStudents method invoked.

2018-01-15 20:06:37.075  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.S.Impl.StudentServiceImpl       : SMAPI -
Student - Service- findAllStudents method processed.

2018-01-15 20:06:37.196  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.Controller.StudentController   : SMAPI -
Student - Controller- listAllStudents request processed.

2018-01-15 20:06:37.318 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Student@2fe9ac3a,   com.student.management.rest.api.Model.Student@47b473f,
com.student.management.rest.api.Model.Student@7e5ce404,    com.student.management.rest.api.Model.Student@3a937fd0,
com.student.management.rest.api.Model.Student@58f99cc4,     com.student.management.rest.api.Model.Student@6e56a5b,
com.student.management.rest.api.Model.Student@1f00d247,    com.student.management.rest.api.Model.Student@3e4c5388,
com.student.management.rest.api.Model.Student@2ac7e739,     com.student.management.rest.api.Model.Student@75de788,
com.student.management.rest.api.Model.Student@1e569fce]] as "application/json" using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:06:37.318 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:06:37.320 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Successfully
completed request

2018-01-15 20:07:16.633 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/student/add]

2018-01-15 20:07:16.634 DEBUG 3376 --- [http-nio-8080-exec-2] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /student/add

2018-01-15 20:07:16.634 DEBUG 3376 --- [http-nio-8080-exec-2] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler method [public org.springframework.http.ResponseEntity<?>
com.student.management.rest.api.Controller.StudentController.createStudent(com.student.management.rest.api.Model.S
tudent,org.springframework.web.util.UriComponentsBuilder)]

2018-01-15 20:07:16.702 DEBUG 3376 --- [http-nio-8080-exec-2] m.m.a.RequestResponseBodyMethodProcessor : Read [class
com.student.management.rest.api.Model.Student] as "application/json;charset=UTF-8" with
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:07:16.722  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.Controller.StudentController   : SMAPI -
Student - Controller- createStudent request received.

2018-01-15 20:07:16.722  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.Impl.StudentServiceImpl      : SMAPI -
Student - Service- saveStudent method invoked.

2018-01-15 20:07:16.723  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.Impl.StudentServiceImpl      : SMAPI -
Student - Service- saveStudent method processed.

2018-01-15 20:07:16.723  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.rest.api.Manager.StudentManager    : SMAPI -
Student - Manager- isValidStudent method invoked.

2018-01-15 20:07:16.723  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.rest.api.Manager.StudentManager    : SMAPI -
Student - Manager- isValidStudent method processed.

2018-01-15 20:07:16.754  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.I.RegistrationServiceImpl    : SMAPI -
Registration - Service- registerNewStudent method invoked.

2018-01-15 20:07:16.755  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.I.RegistrationServiceImpl    : SMAPI -
Registration - Service- registerNewStudent method processed.

2018-01-15 20:07:16.763  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.Controller.StudentController   : SMAPI -
Student - Controller- createStudent request processed.

2018-01-15 20:07:16.847 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:07:16.849 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet : Successfully
completed request

2018-01-15 20:08:01.326 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/lecturer/all]

2018-01-15 20:08:01.327 DEBUG 3376 --- [http-nio-8080-exec-3] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /lecturer/all

2018-01-15 20:08:01.328 DEBUG 3376 --- [http-nio-8080-exec-3] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler method [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Lecturer>>
com.student.management.rest.api.Controller.LecturerController.listAllLecturers()]

2018-01-15 20:08:01.328 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet : Last-
Modified value for [/studentapi/lecturer/all] is: -1

2018-01-15 20:08:01.329  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.Controller.LecturerController  : SMAPI -
Lecturer - Controller- listAllLecturers request received.

2018-01-15 20:08:01.329  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.S.Impl.LecturerServiceImpl     : SMAPI -
Lecturer - Service- findAllLecturers method invoked.

2018-01-15 20:08:01.345  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.S.Impl.LecturerServiceImpl     : SMAPI -
Lecturer - Service- findAllLecturers method processed.

2018-01-15 20:08:01.354  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.Controller.LecturerController  : SMAPI -
Lecturer - Controller- listAllLecturers request processed.

2018-01-15 20:08:01.358 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Lecturer@394617d5,
com.student.management.rest.api.Model.Lecturer@10b5510b, com.student.management.rest.api.Model.Lecturer@4e2068d6,
com.student.management.rest.api.Model.Lecturer@2e74c941, com.student.management.rest.api.Model.Lecturer@27858be5,
com.student.management.rest.api.Model.Lecturer@1d02bb17]]           as           "application/json"           using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:08:01.359 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:08:01.360 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:08:47.965 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/lecturer/all]

2018-01-15 20:08:47.966 DEBUG 3376 --- [http-nio-8080-exec-4] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /lecturer/all

2018-01-15 20:08:47.981 DEBUG 3376 --- [http-nio-8080-exec-4] .m.m.a.ExceptionHandlerExceptionResolver : Resolving
exception from handler [null]: org.springframework.web.HttpRequestMethodNotSupportedException: Request method
'POST' not supported

2018-01-15 20:08:47.981 DEBUG 3376 --- [http-nio-8080-exec-4] .w.s.m.a.ResponseStatusExceptionResolver : Resolving
exception from handler [null]: org.springframework.web.HttpRequestMethodNotSupportedException: Request method
'POST' not supported

2018-01-15 20:08:47.986 DEBUG 3376 --- [http-nio-8080-exec-4] .w.s.m.s.DefaultHandlerExceptionResolver : Resolving
exception from handler [null]: org.springframework.web.HttpRequestMethodNotSupportedException: Request method
'POST' not supported

2018-01-15 20:08:47.986  WARN 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.PageNotFound             : Request
method 'POST' not supported

2018-01-15 20:08:47.987 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:08:47.988 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:08:47.997 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/error]

2018-01-15 20:08:47.997 DEBUG 3376 --- [http-nio-8080-exec-4] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /error

2018-01-15 20:08:47.999 DEBUG 3376 --- [http-nio-8080-exec-4] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler method [public org.springframework.http.ResponseEntity<java.util.Map<java.lang.String, java.lang.Object>>
org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)]

2018-01-15 20:08:48.018 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.w.s.m.m.a.HttpEntityMethodProcessor   : Written
[{timestamp=Mon     Jan     15     20:08:48     IST     2018,     status=405,     error=Method     Not     Allowed,
exception=org.springframework.web.HttpRequestMethodNotSupportedException,   message=Request   method   'POST'   not
supported,           path=/studentapi/lecturer/all}]           as           "application/json"           using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:08:48.019 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:08:48.019 DEBUG 3376 --- [http-nio-8080-exec-4] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:08:54.533 DEBUG 3376 --- [http-nio-8080-exec-5] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/lecturer/add]

2018-01-15 20:08:54.533 DEBUG 3376 --- [http-nio-8080-exec-5] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /lecturer/add

2018-01-15 20:08:54.533 DEBUG 3376 --- [http-nio-8080-exec-5] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler              method              [public              org.springframework.http.ResponseEntity<?>
com.student.management.rest.api.Controller.LecturerController.createLecturer(com.student.management.rest.api.Model
.Lecturer,org.springframework.web.util.UriComponentsBuilder)]

2018-01-15 20:08:54.544 DEBUG 3376 --- [http-nio-8080-exec-5] m.m.a.RequestResponseBodyMethodProcessor : Read [class
com.student.management.rest.api.Model.Lecturer]         as         "application/json;charset=UTF-8"         with
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:08:54.546  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.r.a.Controller.LecturerController  : SMAPI -
Lecturer - Controller- createLecturer request received.

2018-01-15 20:08:54.546  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.r.a.S.Impl.LecturerServiceImpl     : SMAPI -
Lecturer - Service- addNewLecturer method invoked.

2018-01-15 20:08:54.552  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.rest.api.Manager.LecturerManager   : SMAPI -
Lecturer - Manager- isValidLecturer method invoked.

2018-01-15 20:08:54.552  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.rest.api.Manager.LecturerManager   : SMAPI -
Lecturer - Manager- isValidLecturer method processed.
```

```
2018-01-15 20:08:54.702  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.r.a.S.Impl.LecturerServiceImpl      : SMAPI -
Lecturer - Service- addNewLecturer method processed.

2018-01-15 20:08:54.703  INFO 3376 --- [http-nio-8080-exec-5] c.s.m.r.a.Controller.LecturerController  : SMAPI -
Lecturer - Controller- createLecturer request processed.

2018-01-15 20:08:54.703 DEBUG 3376 --- [http-nio-8080-exec-5] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:08:54.704 DEBUG 3376 --- [http-nio-8080-exec-5] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:09:14.490 DEBUG 3376 --- [http-nio-8080-exec-6] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/payment/all]

2018-01-15 20:09:14.490 DEBUG 3376 --- [http-nio-8080-exec-6] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /payment/all

2018-01-15 20:09:14.491 DEBUG 3376 --- [http-nio-8080-exec-6] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                                               method                                               [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Payment>>
com.student.management.rest.api.Controller.PaymentController.listAllPayments()]

2018-01-15 20:09:14.492 DEBUG 3376 --- [http-nio-8080-exec-6] o.s.web.servlet.DispatcherServlet          : Last-
Modified value for [/studentapi/payment/all] is: -1

2018-01-15 20:09:14.493  INFO 3376 --- [http-nio-8080-exec-6] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- listAllPayments request received.

2018-01-15 20:09:14.493  INFO 3376 --- [http-nio-8080-exec-6] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- findAllPayments method invoked.

2018-01-15 20:09:14.517  INFO 3376 --- [http-nio-8080-exec-6] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- findAllPayments method processed.

2018-01-15 20:09:14.552  INFO 3376 --- [http-nio-8080-exec-6] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- listAllPayments request processed.

2018-01-15 20:09:14.565 DEBUG 3376 --- [http-nio-8080-exec-6] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Payment@2914240,   com.student.management.rest.api.Model.Payment@146bfeea,
com.student.management.rest.api.Model.Payment@12068116,     com.student.management.rest.api.Model.Payment@e3abaf6,
com.student.management.rest.api.Model.Payment@6316a33,     com.student.management.rest.api.Model.Payment@4d6658b1,
com.student.management.rest.api.Model.Payment@6b4e7e3a,    com.student.management.rest.api.Model.Payment@30f884af,
com.student.management.rest.api.Model.Payment@3d150071,    com.student.management.rest.api.Model.Payment@424974b2,
com.student.management.rest.api.Model.Payment@5a6e7d6f,    com.student.management.rest.api.Model.Payment@2ec75771,
com.student.management.rest.api.Model.Payment@3b44cc50,    com.student.management.rest.api.Model.Payment@3221d36e,
com.student.management.rest.api.Model.Payment@1b8d5540,     com.student.management.rest.api.Model.Payment@fe5f59f,
com.student.management.rest.api.Model.Payment@125e9e7f,    com.student.management.rest.api.Model.Payment@2ed895f9,
com.student.management.rest.api.Model.Payment@5f8ef237,    com.student.management.rest.api.Model.Payment@7fe7ad54,
com.student.management.rest.api.Model.Payment@192c79b1]]          as          "application/json"          using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:09:14.566 DEBUG 3376 --- [http-nio-8080-exec-6] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:09:14.566 DEBUG 3376 --- [http-nio-8080-exec-6] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:09:30.934 DEBUG 3376 --- [http-nio-8080-exec-7] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/department/all]

2018-01-15 20:09:30.934 DEBUG 3376 --- [http-nio-8080-exec-7] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /department/all

2018-01-15 20:09:30.935 DEBUG 3376 --- [http-nio-8080-exec-7] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                                               method                                               [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Department>>
com.student.management.rest.api.Controller.DepartmentController.listAllDepartments()]

2018-01-15 20:09:30.935 DEBUG 3376 --- [http-nio-8080-exec-7] o.s.web.servlet.DispatcherServlet          : Last-
Modified value for [/studentapi/department/all] is: -1

2018-01-15 20:09:30.935  INFO 3376 --- [http-nio-8080-exec-7] c.s.m.r.a.C.DepartmentController         : SMAPI -
Department - Controller- listAllDepartments request received.

2018-01-15 20:09:30.936  INFO 3376 --- [http-nio-8080-exec-7] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- findAllDepartments method invoked.

2018-01-15 20:09:30.944  INFO 3376 --- [http-nio-8080-exec-7] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- findAllDepartments method processed.

2018-01-15 20:09:30.948  INFO 3376 --- [http-nio-8080-exec-7] c.s.m.r.a.C.DepartmentController         : SMAPI -
Department - Controller- listAllDepartments request processed.
```

```
2018-01-15 20:09:30.951 DEBUG 3376 --- [http-nio-8080-exec-7] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Department@324b9508,
com.student.management.rest.api.Model.Department@2f437b47,
com.student.management.rest.api.Model.Department@22e4cf07,
com.student.management.rest.api.Model.Department@5a1e50d2,
com.student.management.rest.api.Model.Department@34936d4c,
com.student.management.rest.api.Model.Department@beb1da]]          as          "application/json"          using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:09:30.951 DEBUG 3376 --- [http-nio-8080-exec-7] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:09:30.952 DEBUG 3376 --- [http-nio-8080-exec-7] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:10:07.986 DEBUG 3376 --- [http-nio-8080-exec-8] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/department/all]

2018-01-15 20:10:07.987 DEBUG 3376 --- [http-nio-8080-exec-8] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /department/all

2018-01-15 20:10:07.989 DEBUG 3376 --- [http-nio-8080-exec-8] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                                     method                                      [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Department>>
com.student.management.rest.api.Controller.DepartmentController.listAllDepartments()]

2018-01-15 20:10:07.989 DEBUG 3376 --- [http-nio-8080-exec-8] o.s.web.servlet.DispatcherServlet         : Last-
Modified value for [/studentapi/department/all] is: -1

2018-01-15 20:10:07.990  INFO 3376 --- [http-nio-8080-exec-8] c.s.m.r.a.C.DepartmentController         : SMAPI -
Department - Controller- listAllDepartments request received.

2018-01-15 20:10:07.990  INFO 3376 --- [http-nio-8080-exec-8] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- findAllDepartments method invoked.

2018-01-15 20:10:07.994  INFO 3376 --- [http-nio-8080-exec-8] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- findAllDepartments method processed.

2018-01-15 20:10:07.998  INFO 3376 --- [http-nio-8080-exec-8] c.s.m.r.a.C.DepartmentController         : SMAPI -
Department - Controller- listAllDepartments request processed.

2018-01-15 20:10:08.002 DEBUG 3376 --- [http-nio-8080-exec-8] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Department@6ccd9d5f,
com.student.management.rest.api.Model.Department@71d96829,
com.student.management.rest.api.Model.Department@71beaef5,
com.student.management.rest.api.Model.Department@3469f217,
com.student.management.rest.api.Model.Department@34ceb6a0,
com.student.management.rest.api.Model.Department@eec1830]]          as          "application/json"          using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:10:08.003 DEBUG 3376 --- [http-nio-8080-exec-8] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:10:08.004 DEBUG 3376 --- [http-nio-8080-exec-8] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:10:25.952 DEBUG 3376 --- [http-nio-8080-exec-9] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/department/add]

2018-01-15 20:10:25.953 DEBUG 3376 --- [http-nio-8080-exec-9] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /department/add

2018-01-15 20:10:25.953 DEBUG 3376 --- [http-nio-8080-exec-9] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                 method                 [public                 org.springframework.http.ResponseEntity<?>
com.student.management.rest.api.Controller.DepartmentController.createDepartment(com.student.management.rest.api.M
odel.Department,org.springframework.web.util.UriComponentsBuilder)]

2018-01-15 20:10:25.956 DEBUG 3376 --- [http-nio-8080-exec-9] m.m.a.RequestResponseBodyMethodProcessor : Read [class
com.student.management.rest.api.Model.Department]        as        "application/json;charset=UTF-8"        with
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:10:25.957  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.a.C.DepartmentController         : SMAPI -
Department - Controller- createDepartment request received.

2018-01-15 20:10:25.958  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- addNewDepartment method invoked.

2018-01-15 20:10:25.958  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.api.Manager.DepartmentManager    : SMAPI -
Department - Manager- isValidDepartment method invoked.

2018-01-15 20:10:25.958  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.api.Manager.DepartmentManager    : SMAPI -
Department - Manager- isValidDepartment method processed.

2018-01-15 20:10:26.091  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.a.S.Impl.DepartmentServiceImpl   : SMAPI -
Department - Service- addNewDepartment method processed.
```

```
2018-01-15 20:10:26.092  INFO 3376 --- [http-nio-8080-exec-9] c.s.m.r.a.C.DepartmentController         : SMAPI - Department - Controller- createDepartment request processed.

2018-01-15 20:10:26.092 DEBUG 3376 --- [http-nio-8080-exec-9] o.s.web.servlet.DispatcherServlet        : Null ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request handling

2018-01-15 20:10:26.093 DEBUG 3376 --- [http-nio-8080-exec-9] o.s.web.servlet.DispatcherServlet        : Successfully completed request

2018-01-15 20:10:39.460 DEBUG 3376 --- [http-nio-8080-exec-10] o.s.web.servlet.DispatcherServlet       : DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/exam/all]

2018-01-15 20:10:39.460 DEBUG 3376 --- [http-nio-8080-exec-10] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up handler method for path /exam/all

2018-01-15 20:10:39.460 DEBUG 3376 --- [http-nio-8080-exec-10] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning handler method [public org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Exam>> com.student.management.rest.api.Controller.ExamController.listAllExams()]

2018-01-15 20:10:39.461 DEBUG 3376 --- [http-nio-8080-exec-10] o.s.web.servlet.DispatcherServlet        : Last-Modified value for [/studentapi/exam/all] is: -1

2018-01-15 20:10:39.461  INFO 3376 --- [http-nio-8080-exec-10] c.s.m.r.api.Controller.ExamController    : SMAPI - Exam - Controller- listAllExams request received.

2018-01-15 20:10:39.461  INFO 3376 --- [http-nio-8080-exec-10] c.s.m.r.a.Service.Impl.ExamServiceImpl   : SMAPI - Exam - Service- findAllExams method invoked.

2018-01-15 20:10:39.476  INFO 3376 --- [http-nio-8080-exec-10] c.s.m.r.a.Service.Impl.ExamServiceImpl   : SMAPI - Exam - Service- findAllExams method processed.

2018-01-15 20:10:39.489  INFO 3376 --- [http-nio-8080-exec-10] c.s.m.r.api.Controller.ExamController    : SMAPI - Exam - Controller- listAllExams request processed.

2018-01-15 20:10:39.492 DEBUG 3376 --- [http-nio-8080-exec-10] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written [[com.student.management.rest.api.Model.Exam@7ae49138, com.student.management.rest.api.Model.Exam@5ec035d1, com.student.management.rest.api.Model.Exam@2d1028a5, com.student.management.rest.api.Model.Exam@5e76a320, com.student.management.rest.api.Model.Exam@458487, com.student.management.rest.api.Model.Exam@5ff447e3, com.student.management.rest.api.Model.Exam@730d19b8, com.student.management.rest.api.Model.Exam@1809cb5a, com.student.management.rest.api.Model.Exam@34c40560, com.student.management.rest.api.Model.Exam@49f29744, com.student.management.rest.api.Model.Exam@52a9971b]] as "application/json" using [org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:10:39.493 DEBUG 3376 --- [http-nio-8080-exec-10] o.s.web.servlet.DispatcherServlet       : Null ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request handling

2018-01-15 20:10:39.493 DEBUG 3376 --- [http-nio-8080-exec-10] o.s.web.servlet.DispatcherServlet       : Successfully completed request

2018-01-15 20:11:05.728 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/exam/add]

2018-01-15 20:11:05.729 DEBUG 3376 --- [http-nio-8080-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up handler method for path /exam/add

2018-01-15 20:11:05.729 DEBUG 3376 --- [http-nio-8080-exec-1] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning handler method [public org.springframework.http.ResponseEntity<?> com.student.management.rest.api.Controller.ExamController.createExam(com.student.management.rest.api.Model.Exam,org.springframework.web.util.UriComponentsBuilder)]

2018-01-15 20:11:05.739 DEBUG 3376 --- [http-nio-8080-exec-1] m.m.a.RequestResponseBodyMethodProcessor : Read [class com.student.management.rest.api.Model.Exam] as "application/json;charset=UTF-8" with [org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:11:05.740  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.api.Controller.ExamController    : SMAPI - Exam - Controller- createExam request received.

2018-01-15 20:11:05.740  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.Service.Impl.ExamServiceImpl   : SMAPI - Exam - Service- addNewExam method invoked.

2018-01-15 20:11:05.741  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.rest.api.Manager.ExamManager       : SMAPI - Exam - Manager- isValidExam method invoked.

2018-01-15 20:11:05.741  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.rest.api.Manager.ExamManager       : SMAPI - Exam - Manager- isValidExam method processed.

2018-01-15 20:11:05.811  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.a.Service.Impl.ExamServiceImpl   : SMAPI - Exam - Service- addNewExam method processed.

2018-01-15 20:11:05.811  INFO 3376 --- [http-nio-8080-exec-1] c.s.m.r.api.Controller.ExamController    : SMAPI - Exam - Controller- createExam request processed.
```

```
2018-01-15 20:11:05.812 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.web.servlet.DispatcherServlet         : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:11:05.812 DEBUG 3376 --- [http-nio-8080-exec-1] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:11:10.599 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing GET request for [/studentapi/payment/all]

2018-01-15 20:11:10.599 DEBUG 3376 --- [http-nio-8080-exec-2] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /payment/all

2018-01-15 20:11:10.600 DEBUG 3376 --- [http-nio-8080-exec-2] s.w.s.m.m.a.RequestMappingHandlerMapping : Returning
handler                                        method                                         [public
org.springframework.http.ResponseEntity<java.util.List<com.student.management.rest.api.Model.Payment>>
com.student.management.rest.api.Controller.PaymentController.listAllPayments()]

2018-01-15 20:11:10.600 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet         : Last-
Modified value for [/studentapi/payment/all] is: -1

2018-01-15 20:11:10.600  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- listAllPayments request received.

2018-01-15 20:11:10.600  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- findAllPayments method invoked.

2018-01-15 20:11:10.609  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- findAllPayments method processed.

2018-01-15 20:11:10.643  INFO 3376 --- [http-nio-8080-exec-2] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- listAllPayments request processed.

2018-01-15 20:11:10.645 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.w.s.m.m.a.HttpEntityMethodProcessor  : Written
[[com.student.management.rest.api.Model.Payment@22a212ac,  com.student.management.rest.api.Model.Payment@681ff543,
com.student.management.rest.api.Model.Payment@1add4dd9,    com.student.management.rest.api.Model.Payment@40520514,
com.student.management.rest.api.Model.Payment@72d748c0,    com.student.management.rest.api.Model.Payment@65750e34,
com.student.management.rest.api.Model.Payment@153624c6,    com.student.management.rest.api.Model.Payment@459cf9d1,
com.student.management.rest.api.Model.Payment@d4538ea,     com.student.management.rest.api.Model.Payment@400c83d8,
com.student.management.rest.api.Model.Payment@64b89f0b,    com.student.management.rest.api.Model.Payment@46c8b03c,
com.student.management.rest.api.Model.Payment@62ee5267,    com.student.management.rest.api.Model.Payment@46581fa1,
com.student.management.rest.api.Model.Payment@66b890a6,    com.student.management.rest.api.Model.Payment@65f2951d,
com.student.management.rest.api.Model.Payment@751f9eae,    com.student.management.rest.api.Model.Payment@733c22bb,
com.student.management.rest.api.Model.Payment@5123a1da,    com.student.management.rest.api.Model.Payment@20089c9e,
com.student.management.rest.api.Model.Payment@296b8cdf]]          as          "application/json"          using
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:11:10.645 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet         : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:11:10.646 DEBUG 3376 --- [http-nio-8080-exec-2] o.s.web.servlet.DispatcherServlet      : Successfully
completed request

2018-01-15 20:11:36.159 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet            :
DispatcherServlet with name 'dispatcherServlet' processing POST request for [/studentapi/payment/add]

2018-01-15 20:11:36.159 DEBUG 3376 --- [http-nio-8080-exec-3] s.w.s.m.m.a.RequestMappingHandlerMapping : Looking up
handler method for path /payment/add

\2018-01-15 20:11:36.169 DEBUG 3376 --- [http-nio-8080-exec-3] m.m.a.RequestResponseBodyMethodProcessor : Read
[class     com.student.management.rest.api.Model.Payment]     as     "application/json;charset=UTF-8"     with
[org.springframework.http.converter.json.MappingJackson2HttpMessageConverter@7781e32a]

2018-01-15 20:11:36.170  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- createPayment request received. data id : null

2018-01-15 20:11:36.172  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- addNewPayment method invoked.

2018-01-15 20:11:36.172  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.rest.api.Manager.PaymentManager    : SMAPI -
Payment - Manager- isValidPayment method invoked.

2018-01-15 20:11:36.172  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.rest.api.Manager.PaymentManager    : SMAPI -
Payment - Manager- isValidPayment method processed.

2018-01-15 20:11:36.172  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.rest.api.Manager.PaymentManager    : SMAPI -
Payment - Manager- isValidAmount method invoked.

2018-01-15 20:11:36.172  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.rest.api.Manager.PaymentManager    : SMAPI -
Payment - Manager- isValidAmount method processed.

2018-01-15 20:11:36.237  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.S.Impl.PaymentServiceImpl      : SMAPI -
Payment - Service- addNewPayment method processed.

2018-01-15 20:11:36.238  INFO 3376 --- [http-nio-8080-exec-3] c.s.m.r.a.Controller.PaymentController   : SMAPI -
Payment - Controller- createPayment request processed.
```

```
2018-01-15 20:11:36.238 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet          : Null
ModelAndView returned to DispatcherServlet with name 'dispatcherServlet': assuming HandlerAdapter completed request
handling

2018-01-15 20:11:36.238 DEBUG 3376 --- [http-nio-8080-exec-3] o.s.web.servlet.DispatcherServlet      : Successfully
completed request
```

Logs Summary:

By looking at the above logs it can be categorized the services with its usage as follows:

Service	Usage (Approximately)
Payment Service	70%
Exam Service	71%
Registration Service	80%
Student Service	90%
Lecturer Service	75%
Department Service	72%

Table 5-6: Usage Summary as a Percentage.

Above table shows how each service are used when performing above defined tasks from the Student Data Management API. By looking at the percentages it gives slight idea about how important is each of the service in the Student Data Management API when performing defined tasks. Following breakdown shows as a percentage how each service is correlated when performing a task. Those results were obtained by analyzing the logs from the tool which described earlier.

Adding a Student:

Service	Usage (Approximately)
Registration Service	60%
Student Service	100%
Lecturer Service	65%
Department Service	55%

Table 5-7: Adding Student Usage Percentages.

Adding a Lecturer:

Service	Usage (Approximately)
Registration Service	60%
Lecturer Service	100%
Department Service	65%

Table 5-8: Adding Lecturer Usage Percentages.

Adding a Department:

Service	Usage (Approximately)
Lecturer Service	40%
Department Service	100%

.

Table 5-9: Adding Department Usage Percentages.

Adding an Exam:

Service	Usage (Approximately)
Payment Service	40%
Exam Service	100%
Student Service	20%
Lecturer Service	20%
Department Service	20%

Table 5-10: Adding Exam Usage Percentages.

Making a Payment:

Service	Usage (Approximately)
Payment Service	100%
Exam Service	40%
Student Service	20%

Table 5-11: Making a Payment Usage Percentages.

By looking at the above results it can make an assumption Student Service has dependencies to many of the actions performed. Whereas Payment, Exam has less interactions to other actions comparing to the Student and Lecturer Services. Moreover, the Department Service and Registration Service have some impact when performing above tasks but those are not significant as the Student or the Lecturer Services interactions as the percentage wise. Hence as a summary it can be categorized the services as follows:

1. Student Service + Lecture Service (Student - Lecture Service)
2. Department Service + Registration Service (Department - Registration Service)
3. Exam Service
4. Payment Service

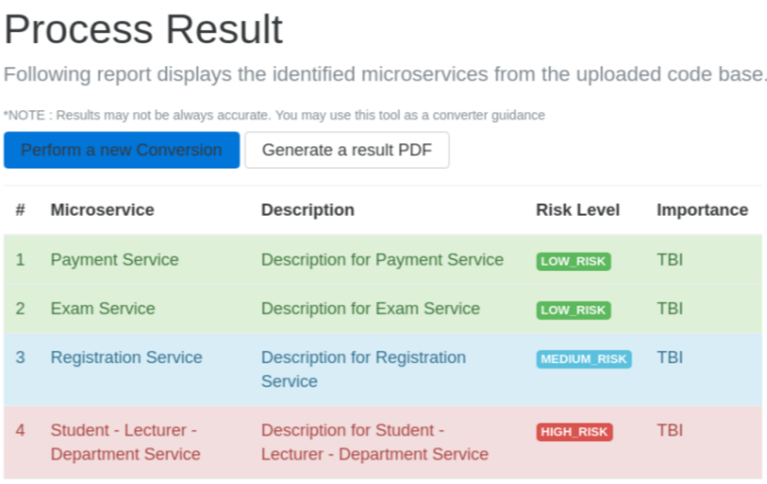
But as above the obtained results those are not accurate as much as the manual approach. Even though those are not accurate, it can used to validate the results obtained from the monolithic to microservice conversion tool.

5.3.3 Validation and Discussion Summary.

From the both of the manual and automated validation method suggest that the Student Service and the Lecture Service should be merge as one microservice. Additionally, the manual validation approach suggests that all other should be a one microservice whereas automated approach suggest having merge of Department Service and Registration Service. But looking at the code and the architecture of the Student Data Management API it is clear that Department Service and the Registration Service should keep as separate services as they don't much depend each other and easy to separate out due to having larger self-containment.

By the method implement from the tool it has suggested different combination of services from ***iteration 1, iteration 2 and iteration 3*** based on the weights feed to the tool for different quality attributes. As comparing to the methods used to validate the tool which created from this research ***iteration 3*** has the most accurate to the point.

That is because in that iteration it has fed the mode weight on the ***Cohesion*** and ***Self-Containment.*** As per the observations and many researchers the key aspect to separate out microservices from a existing service or to build such from the scratch the service which chosen should have high ***Cohesion*** and ***Self-Containment***. That is service should be able to independently develop and deployed. No coordination should be needed with other service teams if no breaking API changes have been made. Each service is effectively its own product with its own codebase and lifecycle [47]. According to the ***iteration 3*** results it suggest following results from the tool.



Process Result

Following report displays the identified microservices from the uploaded code base.

*NOTE : Results may not be always accurate. You may use this tool as a converter guidance

#	Microservice	Description	Risk Level	Importance
1	Payment Service	Description for Payment Service	LOW_RISK	TBI
2	Exam Service	Description for Exam Service	LOW_RISK	TBI
3	Registration Service	Description for Registration Service	MEDIUM_RISK	TBI
4	Student - Lecturer - Department Service	Description for Student - Lecturer - Department Service	HIGH_RISK	TBI

Figure 5-6: 3rd Iteration Final Cluster Results.

From the tool it partitions the source code of Student Management API into a set of clusters. Each cluster is composed of one or more classes. Each resulting cluster corresponds to one microservice. Following table shows the final and most accurate results in terms of number of obtained microservices for the Student Management API and the corresponding average service quality value for each of the four characteristics: functionality, composability, self-containment and usage after the Clustering process. The distribution rate of controllers to services is 7/4= 1.75 for the Student Management API. Even more, it can notice that almost all classes of same service are grouped to offer single functionality. And also, it can be noticeable that functionally relevant classes were grouped in one cluster. For an instance similar and relevant services like Student, Lecturer and Department has been clustered as one Microservice.

#	Microservice	Functionality	Composability	Self-Containment	Usage	Fitness Function Value
1	Student - Lecturer -	0.80	0.88	0.71	0.81	0.710
	Department Service	0.78	0.76	0.70	0.82	0.713
2	Registration Service	0.71	0.76	0.75	0.79	0.786
3	Exam Service	0.76	0.72	0.81	0.87	0.831
4	Payment Service	0.76	0.83	0.91	0.94	0.901

Table 5-12: 3rd Iteration Cluster Values.

And also, it is noticeable that some microservices were directly mapped to one corresponding controller, service and related repository in the architecture, such as Payment Service, Registration Service and Exam Service. In total, 4 services were successfully mapped to 7 controllers or the services in the system architecture. By looking at the further the microservices concepts it can clearly see even the 4th service which is extracted by the tool can be make it as a more granular microservices by separating them to 2 services such as Student Service and Lecturer Service as one Service and Department Service as another Service as the manual validation approach suggested. The results show that close to 80% of extracted services from the tool are accurate. It can be assumed that by feeding context related knowledge to the tool it can be more accurate than this, but it is too hard to implement for this research with the given time frame.

Chapter 6
CONCLUSION

6.1 Research Contributions

The main contribution of the work presented in this paper is the identification of microservices in legacy source code or the monolithic system source code based on service quality characteristics. In order to achieve that for this purpose, it first set a mapping model between java classes and microservice concepts. Then, unlike most ad-hoc identification approaches, then in the research introduced a fitness function that measures the quality of identified services. Initially it assumed that all of the service classes are as one single microservice then from the method in this research it calculates a fitness function value for each of the services on each iteration until it exists from the program. To calculate the fitness function value as described in the implementation section it uses 4 attributes with weighted coefficient. Then based on the fitness function value of each of the identified services those will be clustered into set to microservices at each of the iteration. To demonstrate the applicability of this paper proposed approach, created a simple REST API which has the basic functionality to manage the student data for an institute and it has proven that this method has somewhat significant effect on filtering out microservice from the existing legacy or the monolithic system code bases. By feeding the context knowledge to the tool those results obtained by the tool will be more accurate, but that will some part of the future work.

Further from this research, by trying few of the weighted values against the quality attributes ***functionality, composability, self-containment*** and ***usage*** it has proven that the most applicable values to identify microservices are be the self-containment and the composability. That is because when using higher values for the ***composability*** and the ***self-containment*** when extracting microservices from the monolithic system by using the tool which have created in this research has given the close results as to the same as manual identification process. Moreover, it also has proven that the attributes like those attributes have a direct relationship with identifying the microservices. That is when designing an microservice based architecture-based system it should design in a way that each service in the architecture should;

1. Be able to Independently develop and deployed.
2. Loosely coupled.

3. Have minimal dependence on other services,
4. Have high Cohesion.
5. Closely related functionality should stay together in the same service.
6. Have high Self-Containment.

Finally, it can say that this research has proven that the converting existing monolithic systems into microservices can be done through an automated tool with a well-defined approach. Even though this research has somewhat accuracy around 70% to 80% it has been able to break the myth of difficulty of converting existing monolithic systems to microservices in an automated approach. Basically, this research has proven that we can improve this methodology to have more accurate results with minimum effort even though it missing documentations, peoples who developed the system etc. to convert to microservices.

6.2 Study Limitations

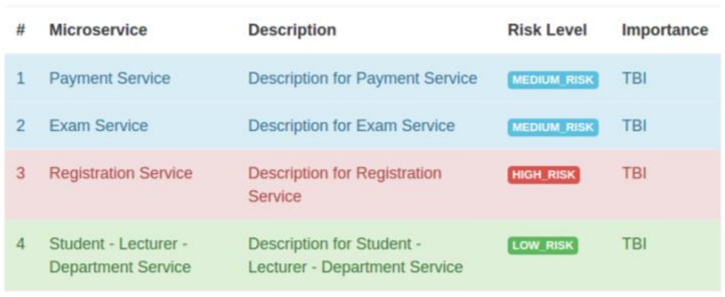
For this study it has selected one type of architectural pattern and the programming language since the given time period of this research is not feasible enough to find out a generic solution to the problem and here it tries to prove that the feasibility of identifying microservices in a monolithic system and the effective attributes. Hence this research focused on java language and the MVC architecture pattern when identifying microservices. Moreover, some of the additional features will be listed under the future work since some interested party can continue the work on any of those topics.

As well as with the given time frame it was not possible to complete the features like giving more insights about the identified services as planned. Such as service descriptions including its usages, dependency graphs etc. That also can be carried out as a separate project in a future research. Being the first few researches of this kind it was a touch task to finding resources and knowledge to carry out to produce the maximum output from the research work.

6.3 Future Work

As per the results obtained from the tool which developed to identify the microservices from the existing monolithic system, it is lack of some of the key aspects to improve its suggestions. Such as context knowledge and the previous analysis knowledge. Since this version of this tool is not having such a capability those results can be vary from the accuracy point of view. For the future work to improve this tool, it can leverage some of the aspects of context knowledge and the previous analysis knowledge.

And also, as you can see in the following picture there are few features still to be implemented such as Description and the Importance.



#	Microservice	Description	Risk Level	Importance
1	Payment Service	Description for Payment Service	MEDIUM_RISK	TBI
2	Exam Service	Description for Exam Service	MEDIUM_RISK	TBI
3	Registration Service	Description for Registration Service	HIGH_RISK	TBI
4	Student - Lecturer - Department Service	Description for Student - Lecturer - Department Service	LOW_RISK	TBI

Figure 6-1: Features to be Implemented.

Description is mean to be show the depended classes for each of the services and its current usages. As well Importance is mean to be show that how much that particular identified services are going to be important to extracted as a microservice, but in order to implement such functionality the tool should leverage some of the aspects of context knowledge about the monolithic system which has fed to the tool to extract the microservices.

Moreover, the current implementation of the tool only support for the Java programs which has the MVC architectural pattern, as a future work it can extend to accept more languages with more architectural patterns. That also will require some aspects of the machine learning techniques to identify the architectural patterns etc.

Chapter 7
REFERENCES

[1]. Levcovitz, Alessandra, Ricardo Terra, and Marco Tulio Valente. "Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems." arXiv preprint arXiv:1605.03175 (2016).

[2]. Kharbuja, Rajendra. "FAKULTÄT FÜR INFORMATIK.".

[3]. Moreira, Pedro Felipe Marques, and Delano Medeiros Beder. "Desenvolvimento de Aplicações e Micro Serviços: Um estudo de caso."Revista TIS 4.3 (2016).

[4]Micor Service Principles to Retreive, Retreive from https://www.linkedin.com/pulse/microservices-principles-converting-monolith-arati-joshi

[5].Key Software Principles , Retrieved from https://code.tutsplus.com/tutorials/3-key-software-principles-you-must-understand--net-25161

[6].Convention over Configuration , Retreived from https://en.wikipedia.org/wiki/Convention_over_configuration

[7]. Ronen, Inbal, Netta Aizenbud, and Ksenya Kveler. "Service identification in legacy code using structured and unstructured analysis." *IBM Programming Languages and Development Environments Seminar*. 2007.

[8]. Zhang, Zhuopeng, Ruimin Liu, and Hongji Yang. "Service Identification and Packaging in Service Oriented Reengineering." *SEKE*. Vol. 5. 2005.

[9]. Sneed, Harry M. "Integrating legacy software into a service oriented architecture." *Conference on Software Maintenance and Reengineering (CSMR'06)*. IEEE, 2006.

[10]. Khadka, Ravi. "Service identification strategies in legacy-to-SOA migration." *the 27th IEEE International Conference on Software Maintenance-Doctoral Consortium (ICSM-DC 2011)*. 2011.

[11]. S. Brinkkemper, “Method engineering: engineering of information systems development methods and tools,” Information and Software Technology, vol. 38, no. 4, pp. 275–280, 1996.

[12]. N. Gold, M. Harman, D. Binkley, and R. Hierons, “Unifying program slicing and concept assignment for higher-level executable source code extraction,” Software:

[13]. M. Papazoglou and W. Van Den Heuvel, “Service-oriented design and development methodology,” International Journal of Web Engineering and Technology, vol. 2, no. 4, pp. 412–442, 2006.

[14]. S. P. Lee, L. P. Chan, and E. W. Lee, “Web services implementation methodology for soa application,” in Proceeding of the 4th IEEE International Conference on Industrial Informatics, 2006, pp. 335–340.

[15]. A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, “SOMA: A method for developing service-oriented solutions,” IBM Systems Journal, vol. 47, no. 3, pp. 377–396, 2008.

[16]. H. Sneed, “Planning the reengineering of legacy systems,” Software, vol. 12, no. 1, pp. 24–34, 1995.

[17]. L. O’Brien, D. Smith, and G. Lewis, “Supporting migration to services using software architecture reconstruction,” in International Workshop on Software Technology and Engineering Practice (STEP’05). Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 81–91.

[18]. D. Harris, A. Yeh, and H. Reubenstein, “Extracting architectural features from source code,” Automated Software Engineering, vol. 3, no. 1, pp. 109–138, 1996.

[19]. J. Van Geet and S. Demeyer, “Lightweight visualisations of cobol code for supporting migration to soa,” in 3rd International ERCIM Symposium on Software Evolution, October, 2007.

[20]. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: elements of reusable object-oriented software. Addison-wesley Reading, MA, 1995, vol. 206.

[21]. F. Arcelli, C. Tosi, and M. Zanoni, “Can design pattern detection be useful for legacy system migration towards soa?” in Proceedings of the 2nd international workshop on Systems development in SOA environments. ACM, 2008, pp. 63–68.

[22]. N. och Dag, B. Regnell, V. Gervasi, and S. Brinkkemper, “A linguisticengineering approach to large-scale requirements management,” Software, IEEE, vol. 22, no. 1, pp. 32–39, 2005.

[23]. U. Priss, “Formal concept analysis in information science,” Annual review of information science and technology, vol. 40, p. 521, 2006.

[24]. S. R. Tilley, D. B. Smith, and S. Paul, “Towards a framework for program understanding,” in 4th International Workshop on Program Comprehension (WPC’96), 1996, pp. 19–28.

[25]. Khadka, Ravi, et al. "A structured legacy to SOA migration process and its evaluation in practice." *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2013 IEEE 7th International Symposium on the*. IEEE, 2013.

[26]. G. Lewis, E. Morris, L. O’Brien, D. Smith, and L. Wrage, “SMART: The service-oriented migration and reuse technique,” CMU/SEI, Tech. Rep. CMU/SEI-2005-TN-029, Sept 2005.

[27]. Q. Gu and P. Lago, “Service identification methods: a systematic literature review,” in Towards a Service-Based Internet. Springer, 2010, pp. 37–50.

[28]. A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, “SOMA: A method for developing service-oriented solutions,” IBM Sys. J., vol. 47, no. 3, pp. 377–396, 2008.

[29]. S. Alahmari, E. Zaluska, and D. De Roure, “A service identification framework for legacy system migration into SOA,” in SCC’10. IEEE, 2010, pp. 614–617.

[30]. A. Fuhr, T. Horn, V. Riediger, and A. Winter, “Model-driven software migration into service-oriented architectures,” CSRD, vol. 28, no. 1, pp. 65–84, 2011.

[31]. A. Marchetto and F. Ricca, “Transforming a java application in an equivalent web-services based application: toward a tool supported stepwise approach,” in WSE’08. IEEE, 2008, pp. 27–36.

[32]. C. Zillmann, A. Winter, A. Herget, W. Teppe, M. Theurer, A. Fuhr, T. Horn, V. Riediger, U. Erdmenger, U. Kaiser et al., “The SOAMIG Process Model in Industrial Applications,” in CMSR’11. IEEE, 2011, pp. 339–342.

[33]. L. Aversano, L. Cerulo, and C. Palumbo, “Mining candidate web services from legacy code,” in WSE’08. IEEE, 2008, pp. 37–40.

[34]. Z. Zhang, H. Yang, and W. Chu, “Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration,” in QSIC’06. IEEE, 2006, pp. 385– 392.

[35]. J. Van Geet and S. Demeyer, “Lightweight visualisations of COBOL code for supporting migration to SOA,” in Soft Evol’07, 2007.

[36]. Q. Gu and P. Lago, “Service identification methods: a systematic literature review,” in Towards a Service-Based Internet. Springer, 2010, pp. 37–50.

[37] A. Marchetto and F. Ricca, “Transforming a java application in an equivalent web-services based application: toward a tool supported stepwise approach,” in WSE’08. IEEE, 2008, pp. 27–36.

[38]. Z. Zhang, R. Liu, and H. Yang, “Service identification and packaging in service oriented reengineering,” in SEKE’05, 2005, pp. 219–26.

[39]. R. Khadka, G. Reijnders, A. Saeidi, S. Jansen, and J. Hage, “A method engineering based legacy to SOA migration method,” in ICSM’11. IEEE, 2011, pp. 163–172.

[40]. A. Jansen and J. Bosch, “Software architecture as a set of architectural design decisions,” in WICSA’05. IEEE, 2005, pp. 109–120.

[41]. F. Chen, Z. Zhang, J. Li, J. Kang, and H. Yang, “Service identification via ontology mapping,” in COMPSAC’09. IEEE, 2009, pp. 486–491.

[42]. H. Sneed, “COB2WEB a toolset for migrating to web services,” in WSE’08. IEEE, 2008, pp. 19–25.

[43]. H. Sneed, “A pilot project for migrating COBOL code to web services,” STTT, vol. 11, no. 6, pp. 441–451, 2009.

[44]. F. Cuadrado, B. Garc´ıa, J. Dueas, and H. Parada, “A case study on software evolution towards service-oriented architecture,” in AINAW’08. IEEE, 2008, pp. 1399–1404.

[45]. R. Heckel, R. Correia, C. Matos, M. El-Ramly, G. Koutsoukos, and L. Andrade, “Architectural transformations: From legacy to three-tier and services,” in Soft. Evol. Springer, 2008, pp. 139–170.

[46]. Patidar, K., R. Gupta, and Gajendra Singh Chandel. "Coupling and cohesion measures in object oriented programming." *International Journal of Advanced Research in Computer Science and Software Engineering* 3.3 (2013).

[47]. Bieman, James M., and Byung-Kyoo Kang. "Cohesion and reuse in an object-oriented system." *ACM SIGSOFT Software Engineering Notes*. Vol. 20. No. SI. ACM, 1995.

[48]. Sahni, V. (2017). Best Practices for Building a Microservice Architecture. [online] Vinay Sahni. Available at: https://www.vinaysahni.com/best-practices-for-building-a-microservice-architecture#independent [Accessed 6 Jun. 2018].