

SYSTEM AND METHOD FOR APPLICATION ENVIRONMENT MIGRATION

FIELD OF THE INVENTION

The present disclosure generally relates to the field of computers, more specifically the present disclosure relates to a system and method for application environment migration which allows applications to be automatically migrated from an older environment to a newer and faster environment without the cumbersome hassle and time being taken by users, while maintaining structural soundness and quality standards.

BACKGROUND OF THE INVENTION

With the changing times, technology has also changed and from the invention of the wheel, we now stand at an era where commute can be ordered through a mobile device. In such, technology has taken a huge leap from times that computers were as big as ballrooms to now when they are as small as a thumbnail. Little by little, we carved technology to help the human race by way of multiple applications. In this ever-changing and ever-reforming field of technology, the application environments keep improving and there arises a necessity to keep updating the applications to better-suit their advanced requirements.

Conventionally, applications were migrated onto newer environments by completely reproducing the application in accordance with the newer environment which was time-taking and did not provide much benefit of the already existing application. With quicker environment updates, there has been inadequate time to update applications manually since by the time an application gets migrated to a newer environment, a newer one would be released. Further, migrating applications to different environments also came with its own unique challenges including but not limited to assuring the logic does not get lost during transformation, and that the migration corresponds to the older and newer environments.

Cloud adoption has become mainstream with enterprises leveraging Cloud as a key part of their IT strategy. This is evident from the year on year growth of leading cloud providers like AWS, Azure and GCP over the last 3 years. As a first phase these enterprises have successfully adopted cloud by developing new applications on cloud using cloud native design principle. Having realized success with that, enterprises are now looking for ways to migrate rest of their important applications on cloud to realize broad benefits of cloud. The challenge however is migrating

existing applications to cloud poses a different set of issues, for example what is the right cloud migration approach, balancing of risk, cost and timeline to complete the cloud migration, complexity of existing applications especially old applications without an SME, maximize total cost of ownership (TCO) savings and increase productivity for future releases, execution speed, enhanced performance, and the like.

Due to the complex parameters and effort required in migrating application environments, along with the crunch in time with regular updates, there exists a need for developing a system and method for application environment migration which migrates an application from one environment to another without or minimally requiring manual assistance while addressing the variety of issues associated with application, and its components.

SUMMARY OF THE INVENTION

This summary is provided to introduce concepts related to systems and methods for application environment migration and the concepts are further described below in the detailed description. This summary is neither intended to identify essential features of the claimed subject matter nor is it intended for use in determining or limiting the scope of the claimed subject matter.

In one implementation, a method for application environment migration is disclosed. The method comprising assessing, by a processor of an application server, a source application code of a source application environment and ascertaining, by the processor, a quantum change for migrating the source application code to a target application code and forecasting, by the processor, an assessment statistic, wherein the assessment statistic provides at least one functional readiness and a timeline to complete the migration of the source application code. The method further comprising scanning, by the processor, the source application code for identifying a business logic and generating, by the processor, a re-factored code for the source application code, wherein the re-factored code is generated by breaking the source application code into macro-services and repackaging the macro-services in accordance with the target application code while retaining the business logic. The method further comprising updating, by the processor, components of the source application environment as per the forecasted assessment statistic and the re-factored code and thereby migrating, by the processor, the source application environment to the target application environment, wherein the migration re-

platforms the updated components and the re-factored code of the source application environment to the target application environment.

In yet another implementation, the method has the quantum change for migrating the source application code to a target application code calculated on the basis of the size and complexity of the application.

In yet another implementation, the method has the re-factored code generated utilizing a continuous integration and deployment framework and an automated test framework.

In yet another implementation, the method has the assessment statistic provide at least one inhibitor to complete the migration of the source application code.

In yet another implementation, the method scans the source application code that includes scanning links between components of the source application code.

In yet another implementation, the method has repackaging the macro-services in accordance with the target application code is completed while retaining links between components of the source application code.

In yet another implementation, the method has the macro-services identified by an AI Engine.

In another implementation, the method comprises, migrating, by the processor, libraries and design patterns from the source application environment in accordance with the target application environment.

In another implementation, the method comprises, implementing, by the processor, security rules from the source application environment in accordance with the target application environment.

In another implementation, the method comprises, packaging and building, by the processor, each service into a separate container by generating an application program interface proxy utilizing a gateway and a continuous integration and continuous deployment framework.

In another implementation, the method comprises, accessing, by the processor, a plurality of storage units comprising a business logic, links, rule engines, libraries of available environments, standard tools, and coding languages.

In yet another implementation, the method has the source application environment migrate to the target application environment comprising the steps of: a source to target environment re-development, a source to target application re-factoring, a source to target application re-hosting, and a source to target application re-platforming.

In one implementation, a system for application environment migration is disclosed. The system comprises a processor and a memory coupled to the processor, wherein the processor executes a plurality of modules stored in the memory. The plurality of modules further comprises an assessment module, for assessing a source application code of a source application environment and ascertaining a quantum change for migrating the source application code to a target application code, wherein the assessment module forecasts an assessment statistic that provides at least one functional readiness and a timeline to complete the migration of the source application code. The plurality of modules further comprises a re-factor module, for scanning the source application code to understand a business logic and generating a re-factored code for the source application code, wherein the re-factored code is generated by breaking the source application code into macro-services and repackaging the macro-services in accordance with the target application code while retaining the business logic. The plurality of modules further comprises a re-platform module, for updating components of the source application environment as per the forecasted assessment statistic and the re-factored code and migrating the source application environment to the target application environment, wherein the migration re-platforms the updated components and the re-factored code of the source application environment to the target application environment.

In yet another implementation, the system has the quantum change for migrating the source application code to a target application code is calculated on the basis of the size and complexity of the application.

In yet another implementation, the system has the re-factored code generated utilizing a continuous integration and deployment framework and an automated test framework.

In yet another implementation, the system has the assessment statistic provide at least one inhibitor to complete the migration of the source application code.

In yet another implementation, the system scans the source application code that includes scanning links between components of the source application code.

In yet another implementation, the system has repackaging of the macro-services in accordance with the target application code is completed while retaining links between components of the source application code.

In yet another implementation, the system has the macro-services identified by an AI Engine.

In another implementation, the system comprises the re-factor module that migrates libraries and design patterns from the source application environment in accordance with the target application environment.

In another implementation, the system comprises the re-platform module that implements security rules from the source application environment in accordance with the target application environment.

In another implementation, the system comprises the re-factor module that packages and builds each service into a separate container by generating an application program interface proxy utilizing a gateway and a continuous integration and continuous deployment framework.

In another implementation, the system comprises the plurality of modules that executes accessing a plurality of storage units comprising a business logic, links, rule engines, libraries of available environments, standard tools, and coding languages.

In yet another implementation, the system has the source application environment migrate to the target application environment comprising the steps of: a source to target environment re-development, a source to target application re-factoring, a source to target application re-hosting, and a source to target application re-platforming.

It is primary object of the subject matter to provide a system and method for application environment migration that may be used to migrate an application from one platform to another and more specifically, it may be used to migrate an application from an older environment to a newer environment. The system and method for application environment migration may be customized based on the application to be migrated as well as the environments including the older environment the application is developed in and the newer environment to which the application has to be migrated.

It is another object of the subject matter to provide an application environment migration that caters to a number of applications while retaining their business logic. Furthermore, the system and method for application environment migration may be able to migrate the selected application to be migrated from a wide range of environments to another wide range of environments.

It is another object of the subject matter to provide a system and method for application environment migration that migrates an application's environment in an automated manner without the requirement of manual inputs.

It is another object of the subject matter to provide an application environment migration that reduces the time and effort taken by development teams to migrate an application from an older environment to a newer environment, by eliminating the repeatable patterns of work done by the development team, thereby also reducing costs.

It is another object of the subject matter to provide an application environment migration that can be utilized by development teams to migrate an existing application from an older environment to a newer environment.

It is another object of the subject matter to provide an application environment migration that behaves like a parser which studies an existing chunk of code, analyses it, processes it and thereafter repackages it.

It is another object of the subject matter to provide an application environment migration that can be utilized for migrating an application from an older environment like a web application coded in JAVA or .NET, to a newer environment like the cloud interface.

It is another object of the subject matter to provide an application environment migration that migrates applications from an older environment to a newer one by reducing implementation costs, reducing total cost of ownership thereby increasing profitability, increasing the execution speed, enhancing the application's performance and enhancing the application's productivity.

It is another object of the subject matter to provide a number of advantages depending on the particular aspect, embodiment, implementation and/or configuration.

It is another object of the subject matter to provide a platform that can provide reliable execution, scalability, and value-added services, while controlling operating effort and costs.

It is another object of the subject matter to efficiently manage numerous instances simultaneously, work in different regulatory requirements, enable resources to collaborate and work together closely, efficiently and collectively with user friendly interfaces.

These and other implementations, embodiments, processes and features of the subject matter will become more fully apparent when the following detailed description is read with the accompanying experimental details. However, both the foregoing summary of the subject matter and the following detailed description of it represent one potential implementation or embodiment and are not restrictive of the present disclosure or other alternate implementations or embodiments of the subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

A clear understanding of the key features of the subject matter summarized above may be had by reference to the appended drawings, which illustrate the method and system of the subject matter, although it will be understood that such drawings depict preferred embodiments of the subject matter and, therefore, are not to be considered as limiting its scope with regard to other embodiments which the subject matter is capable of contemplating. Accordingly:

FIGURE.1 illustrates a schematic module diagram depicting a method of application environment migration, in accordance with an embodiment of the present subject matter.

FIGURE.2 illustrates a system diagram describing the working of an exemplary application environment migration system, in accordance with an embodiment of the present subject matter.

FIGURE.3 illustrates an exemplary assessment report automatically generated by an exemplary application environment migration system, in accordance with an embodiment of the present subject matter.

FIGURE.4 depicts an exemplary view of an exemplary application environment migration method, in accordance with an embodiment of the present subject matter.

FIGURE.5 illustrates an exemplary flowchart of a method of application environment migration, in accordance with an embodiment of the present subject matter.

FIGURE.6 illustrates another exemplary flowchart of a method of application environment migration, in accordance with an embodiment of the present subject matter.

DETAILED DESCRIPTION OF THE INVENTION

The following is a detailed description of implementations of the present disclosure depicted in the accompanying drawings. The implementations are in such detail as to clearly communicate the disclosure. However, the amount of detail offered is not intended to limit the implementations but it is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present disclosure. While aspects of described systems and methods for application environment migration can be implemented in any number of different computing systems, environments, and/or configurations, the embodiments are described in the context of the following exemplary system(s).

The present disclosure provides an application cloud re-platforming from a source application environment to a target application environment. The cloud re-platform of applications involves migrating an existing on-premise application to cloud to make the source application environment work efficiently by taking advantage of scalability and elastic features of cloud. The source application environment structure will also be modified to reduce a total cost of ownership (TCO) of these applications which involve adopting more open source and cloud friendly software to remove a licensing cost for virtual machines, application server and databases. Actual TCO reduction will vary from application to application and determined after application assessment. The present disclosure lists a few important steps to be performed in applications, although they may vary from application to application, modify existing application to work effectively on Cloud Platform or PaaS, all blocker related to source and on-premise design patterns will be changed, all underlying libraries will be upgraded to make the application compatible with container or PaaS, application servers (Weblogic, WebSphere, JBoss,...) and databases may be replaced with open source cloud friendly software, monolith application may be de-coupled into smaller macro services for efficiency and velocity.

The present disclosure provides a mechanism for cloud re-platforming by simple containerization involves removing all code blockers that impede containerization and deploy a source application in a container like Docker to make it work on Cloud. Further, the source application release process will also be automated which involves, test case and CI/CD automation. Still further, the source application and/or the source database may also be re-platformed to a lightweight open source server like Tomcat for JAVA applications to reduce the

TCO of these applications by eliminating the license cost and reducing the compute cost on cloud. However, to guarantee stability and performance the monolith application will have to be broken down to smaller services, make all of these applications stateless and deploy them into separate Tomcat servers with each deployed in a separate container. Thus, the source application will be migrated to a target application platform by decoupling the monolith application and exposing it through APIs by using macro-services concepts to make it cloud ready application. Along with breaking monolithic application into macro-services, CI/CD will also be implemented to achieve the fast and automated deployment. A component wise structure and a step wise method is listed below.

FIGURE.1 illustrates a schematic module diagram 100 depicting a method of application environment migration, in accordance with an embodiment of the present subject matter.

In one implementation, an application environment migration system 120 implements a method for application environment migration on a server 102, the system 120 includes a processor(s) 122, interface(s) 124, framework(s) 150, an AI Engine 152, and a memory 126 coupled or in communication to the processor(s) 122. The processor(s) 122 may be implemented as one or more microprocessors, microcomputers, microcontrollers, digital signal processors, central processing units, state machines, logic circuitries, and/or any devices that manipulate signals based on application environment migration instructions. Among other capabilities, the processor(s) 122 is configured to fetch and execute computer-readable instructions stored in the memory 126. The framework 150 includes but not limited to a continuous integration and continuous deployment framework, an automated test framework, and the like. The AI Engine 152 including but not limited to a database (DB) re-platform engine, an app re-platform engine, a batch and scheduler re-platform engine, a message broker re-platform engine, an open source AI engine, and the like.

Although the present disclosure is explained by considering a scenario that the system is implemented as an application on a server, the systems and methods can be implemented in a variety of computing systems. The computing systems that can implement the described method(s) include, but are not restricted to, mainframe computers, workstations, personal computers, desktop computers, minicomputers, servers, multiprocessor systems, laptops, tablets, SCADA systems, smartphones, mobile computing devices and the like.

The interface(s) 124 may include a variety of software and hardware interfaces, for example, a web interface, a graphical user interface, etc., allowing the system 120 to interact with a user.

Further, the interface(s) 124 may enable the system 120 to communicate with other computing devices, such as web servers and external data servers (not shown in figure). The interface(s) 124 can facilitate multiple communications within a wide variety of networks and protocol types, including wired networks, for example LAN, cable, etc., and wireless networks such as WLAN, cellular, or satellite. The interface(s) 124 may include one or more ports for connecting a number of devices to each other or to another server.

A network used for communicating between all elements in application server and cloud environment may be a wireless network, a wired network or a combination thereof. The network can be implemented as one of the different types of networks, such as intranet, local area network LAN, wide area network WAN, the internet, and the like. The network may either be a dedicated network or a shared network. The shared network represents an association of the different types of networks that use a variety of protocols, for example, Hypertext Transfer Protocol HTTP, Transmission Control Protocol/Internet Protocol TCP/IP, Wireless Application Protocol WAP, and the like, to communicate with one another. Further the network may include a variety of network devices, including routers, bridges, servers, computing devices. The network further has access to storage devices residing at a client site computer, a host site server or computer, over the cloud, or a combination thereof and the like. The storage has one or many local and remote computer storage media, including one or many memory storage devices, databases, and the like.

The memory 126 can include any computer-readable medium known in the art including, for example, volatile memory (e.g., RAM), and/or non-volatile memory (e.g., EPROM, flash memory, etc.). In one embodiment, the memory 126 includes module(s) 128 and system data 140.

The modules 128 further includes an assessment module 130, a re-factor module 132, a re-platform module 134, and other modules 136 including but not limited to a blocker assessment module, a script generator module and the like. The script generator module further including but not limited to a test case generator a build script generator, a docker script generator, a pipeline script generator, a deployment script generator, and the like. It will be appreciated that such modules may be represented as a single module or a combination of different modules. Additionally, the memory 126 further includes system data 140 that serves, amongst other things, as a repository for storing data fetched, processed, received and generated by one or more of the modules 128. The system data 140 includes, for example, operational data,

workflow data, and other data at a storage 144. The system data 140 has the storage 144, represented by 144a, 144b, ..., 144n, as the case may be. In one embodiment, the system data 140 has access to the other databases over a web or cloud network. The storage 144 includes multiple databases including but not limited to an assessment module data, a re-factor module data, a re-platform module data, libraries, link identifiers, a database adapter, a database dictionary, a database parser, a technology dictionary, a file adapter, an application parser, a pattern recognizer, open source libraries, technology stack, and the like. It will be appreciated that such databases may be represented as a single database or a combination of different databases. In one embodiment data may be stored in the memory 126 in the form of data structures. Additionally, the aforementioned data can be organized using data models, such as relational or hierarchical data models.

The server 102 is further connected or connectable or a part of a cloud environment 110 that has a plurality of computing systems including but not limited to a target application environment server 104, and an another target application environment server 106, and databases including but not limited to a database 108. It is imperative that the computing systems and databases communicate with each other under a cloud computing rules and also with the server 102 under the web or other available communication mediums/ protocols. The computing systems of the target application server 104, 106 generally are distributed processing systems including multiple computing devices connected by and communicating over a network. Software applications may be run "in the cloud" by configuring them to execute across one or more of the computing devices in a particular cloud computing environment/ system. The computing devices of a cloud computing system may each execute separate copies of the software application, or, in some cases, the operations of the software application may be split among different computing devices and executed in parallel. A cloud computing system may include a plurality of cloud computing instances representing resources available for executing applications. Each instance may be a physical computing device having particular capabilities (storage size, processing speed, network bandwidth, etc.), or may be a virtual computing device having particular capabilities. A particular cloud computing system may offer different instance types having different sets of capabilities for executing software applications.

In one implementation, at first, a user including but not limited to a migration professional, a database administrator, an application developer, a quality analyst or a test professional may use a user access device to access the system 120 via the interface 124 using the server 102. The

working of the system 120 and related method and associated modules, sub modules, methods may be explained in detail also using FIG.2, FIG.3, FIG.4, FIG.5, and FIG.6 explained below.

In one embodiment, the system 120 receives a user instruction data through the interface 124 on the server 102. The modules 128 of the system 120 processes the instructions using the processor 122 while using the system data 140, the other modules 136, the frameworks 150 and supporting components. The assessment module 130 for application environment migration is utilized before initiating the migration of an application from one environment to another. The assessment module 130 performs an assessment of a source application environment to ascertain an in-depth understanding of a quantum change the source application environment will have to undergo while assessing the risk, costs, timeline as well as the impact the change would make to other dependent applications to migrate to a target application environment. The assessment module 130 understands the underlying application source code developed by various developers having a variety of coding styles. Furthermore, a forecasted assessment report 302 generated by the assessment module 130 using the AI engine 152 provides in-depth details regarding but not limited to a timeline required, a percentage of code that needs to be changed, a plurality of readiness parameters, a plurality of inhibitors or code blockers and warnings along with the reason, highlighted portions where code needs to be modified, number of services a monolith can be broken into, the kind of services a monolith may be broken into and the readiness of the application for continuous integration and continuous deployment. The assessment report 302 generated by the assessment module 130 provides an accurate estimate and timeline to complete the migration of the source application environment. In one embodiment, an RPA bot is configured to generate an assessment report 302 using a proactive application environment migration forecasting (PAEMF) algorithm. In one embodiment, the RPA bot is a software bot or a combination of a software and hardware bot. In an embodiment, the software bot is a computer program enabling a processor to perform robotic process automation by utilizing AI. In another embodiment, the bot has a combination of hardware and software, where the hardware includes memory, processor, controller and other associated chipsets especially dedicated to perform functions that enable robotic process automation for application environment migration.

The re-factor module 132 for application environment migration supports the automated re-platforming for the source application environment migration to the target application environment migration by converting the pre-existing code of the monolith web application and

generate a re-factored code which corresponds to the target application environment without changing business logic(s). The re-factored code is generated by breaking a source application code into macro-services and repackaging the macro-services in accordance with a target application code while retaining the business logic. The re-factor module 132 generates the re-factored code for the source application environment utilizing the continuous integration and deployment framework 150, the automated test framework 150, and middleware re-platform and environment-friendly design patterns.

The re-platform module 134 for application environment migration supports the automated re-platforming for the source application environment migration to the target application environment to transform an application from one environment to another on the basis of the size of the application as well as its complexity by performing activities including but not limited to upgrading underlying libraries, de-coupling the front-end and back-end from the application monolith, breaking the back-end into several services in accordance to the new environment, packaging and building each service into a separate container, generating the application program interface proxy utilizing a gateway, generating a continuous integration and deployment framework 150, and the like. The re-platform module 134 updates the supporting components of the source application environment as per a forecasted assessment statistic of the assessment report 302 and the re-factored code and thereby migrating the source application environment to the target application environment by re-platforming the updated supporting components and the re-factored code of the source application environment to the target application environment. The re-platform module 134 also perform activities including but not limited to integrating patterns of the source application environment with design patterns of the target application environment, fine-tuning the code, resolving deployment issues, implementing security in order to ascertain the migrated application to be fully functional, and the like. The re-platform module 134 assists in multiple forms of environment migration including but not limited to a re-hosting environment migration, an application re-platforming, an application re-factoring and a native environment development, and the like.

The blocker assessment module of the other modules 136, any application like a monolithic web application having on-premise design patterns may be inputted in the system and method for application environment migration wherein the blocker assessment module of the system and method for application environment migration may discover patterns in the on-premise design patterns of the monolithic web application. Furthermore, this information may be transferred to

the automated re-platforming module of the system and method for application environment migration as per figure 1.

The reference architecture of the system 120 further has the supporting components, such as, test generator- test generator generates the test case for unit, functional and performance test cases for generated API. Build Script Generator- build script generator generates the scripts to successfully build and run the application. Docker Script Generator- Docker script generator generates the scripts to containerize the application and its dependencies. Pipeline Script Generator- pipeline script generator generates the scripts to create the CI/CD pipeline for dev, test and prod environments. Deployment Script Generator- deployment script generator generates the scripts to deploy the applications to the target environment. Database Adapter- database adapter utility establishes the connectivity to multiple database servers such as IBM DB2, Oracle, Microsoft SQL Server, Sybase ASE and PostgreSQL. Database Dictionary- database dictionary contains vast database related keywords and built-in functions used by the multiple database servers mentioned above. Database Parser- database parser parses the database objects by using the database dictionary to find the patterns which are not compatible with the target database. AI Engine- AI engine identifies the existing and potential macro/micro services in the legacy application. DB Re-platform Engine- DB re-platform engine re-platforms the source database objects such as tables, constraints, sequences, triggers, functions procedures etc. to the target database compatible objects. File Adapter- file adapter provides access to the source files of the application. the files are scanned to prepare the report and eventually re-platformed as the target application. Technology Dictionary- technology dictionary contains vast technology related patterns used inside the monolith applications. Application Parser- application parser parses the application source code with the help of pattern recognizer to find out the whole inventory of the technologies the application is using. this inventory is used for the assessment and re-platforming. Pattern Recognizer- pattern recognizer recognizes the pattern based on the technology dictionary to find the technologies used in the application. App Re-platform Engine- App re-platform engine re-platforms the source application which was parsed by the application parser. this engine generates the application which is ready to be deployed to the cloud. Technology Stack including but not limited to JAVA 1.8; Maven; JUnit, NUnit, XUnit JMeter; Docker Jenkins; AWS CLI, Azure CLI, and the like.

FIGURE.2 illustrates a system diagram 200 describing the working of an exemplary application environment migration system, in accordance with an embodiment of the present subject matter.

The present disclosure depicts two important activities which are a source application environment assessment and a source application code transformation to the desired target platform. These two activities are explained below in the flowchart 200.

In one embodiment, a monolith web application 202, such as a JAVA application or a .NET application, is a source application of a source application environment. The monolith web application 202 is linked to various supporting components 204, including but not limited to, databases, batch, integration, App, scheduler, message broker, and the like.

In a first step, a cloud blocker assessment 206 assess the source application and generates an assessment report 212 comprising code impediments, source code insights, and on-premise design patterns 214. The on-premise design patterns 214 further including, but not limited to EJB, RMI, STRUTS/SPRING, SOAP/XML, JSP/ASP, COM/COM++, ESB, and the like. The assessment of the source application involves technology stack of the current applications and code analysis. The assessment provides automated insights into the time, effort and risk involved in re-platforming the source application. The assessment starts with a code analysis, where it performs an automated assessment of the source application static code and identifies the inhibitors or blockers (Packages/Frameworks/API/Libraries etc.) which will fail the containerized cloud migration of the application and readiness parameters and thereby provides the detailed assessment report 212 further detailed as the assessment report 302. It performs this code assessment for entire source application ecosystem including applications, databases, batch programs and schedulers, integration message brokers, build and deployment modules. The cloud blocker assessment 206 gets an in depth understanding of the quantum of changes the source environment will have to go through to assess the risk, cost, timeline and the impact to other dependent components and applications. The report primarily provides percentage of code that needs to be changed, code readiness, blockers and warnings along with the reason, exact places where the code needs to be modified, number of services into which the monolith can be broken down, DevOps and CI/CD readiness, and a timeline for such execution.

In the next step, an automated transformation/ re-platforming 208 for application and supporting components re-platforming from a source environment to a target environment is performed to generate a cloud ready application and supporting components 210, such as a JAVA application or a .NET application, an Oracle database or a DB2 database comprising of a continuous integration and deployment framework 216, an automated test framework 218, a middle ware re-platform 220, and cloud friendly design patterns 222 after re-factoring a source code of the

source application environment while retaining a business logic of the source application in the target application. The sub-steps in automated re-platforming 208 is to upgrade underlying libraries and other supporting components, de-couple front-end and back-end from the monolith, break down the back-end into several macro services, package and build each macro service into a separate container, rest of the activities like integrating web-front with macro services, fine tuning the code, resolving deployment issues and, implementing security would be completed by the consultants to make the application fully functional. In one example, the application environment transformation further includes generating API proxy using Zuul, generating test framework for APIs, generating CI/CD framework. In one example, Table A and Table B may be used as an example to illustrate an exemplary code transformation.

Table A: Exemplary code transformation features

S.No.	Source Technology	Target Technology
1	Session EJB	Spring Boot
2	Entity EJB	JPA
3	Message driven EJB	Spring JMS
4	Batch	Spring batch
5	SOAP service	SOAP service + REST end point
6	JAVA POJO service	Spring boot with REST end points
7	.NET	.NET Core
8	Telerik ORM	Entity framework
9	Weblogic/WebSphere	Tomcat
10	Oracle/SQL Server/DB2	Postgres

Table B: Exemplary code generation features

S.No.	Features
1	API test automation
2	Junit and Nunit test cases (asserts are manual)
3	API Proxy (Zuul)
4	OAuth2 authentication
5	CI/CD framework

The cloud re-platforming activities includes converting the source code from on-premise design patterns to cloud friendly design patterns EJBs to POJOs on Spring framework, upgrading underlying libraries used by the application to make it compatible with cloud containers or PaaS. JDK, Spring, Mybatis and others, de-coupling the back-end business logic into multiple services,

re-platform heavy weight Weblogic or WebSphere application servers to Tomcat servers, developing unit test cases, API test automation, CI/CD pipeline and others. Further, application cloud re-platforming involves several repetitive steps like code analysis, code refactoring, library upgrades, compiling build scripts, deployment scripts, containerization scripts etc. This results in significant improvement in Time to Market which in turn can reduce the overall cost and ensure high quality re-platforming.

FIGURE.3 illustrates an exemplary assessment report 302 automatically generated by an exemplary application environment migration system, in accordance with an embodiment of the present subject matter.

In one embodiment, an assessment module 130 generates an assessment report 302 after assessing a source application environment comprising code impediments, source code insights, and on-premise design patterns 214. The assessment report 302 provides automated insights into the time, effort and risk involved in re-platforming the source application environment.

The assessment includes a code analysis, where it performs an automated assessment of the source application static code and identifies the inhibitors or blockers (Packages/ Frameworks/ API/ Libraries etc.) which will fail the containerized cloud migration of the application and readiness parameters and thereby provides a detailed assessment report 302. The generation of the report is an automated process along with a few user configurations based on the source application and target application environment.

The assessment report 302 gets an in depth understanding of the quantum of changes that the source application will have to go through to assess the risk, cost, timeline and the impact to other dependent components and applications in form of readiness parameters 304 and a visual composition 306. The quantum of changes is summation of all changes required in the application migration. The assessment report has various sections, a few exemplary illustrations for an application snapshot includes parameters, such as a containerization readiness 304a, a re-platform readiness 304b, a DevOps Readiness 304c, a Macro/Micro API readiness 304n, a scheduler re-platform readiness 304n, a batch readiness, a message broker readiness, a DB re-platform readiness 304n, among others. Each of the parameters shows a readiness state in percentage (%) or the like to showcase how much ready and/or how much not-ready/ blocking/inhibitor states are present for the respective parameter. For example, the containerization readiness 304a shows a container ready percentage with a readiness state and a refactoring required/ inhibitor percentage, the re-platform readiness 304b shows a Tomcat ready

percentage with a readiness state and a refactoring required/ inhibitor percentage, the DevOps Readiness 304c shows a CD ready percentage with a readiness state and a CI readiness/ inhibitor percentage, the Macro/Micro API readiness 304n shows a Macro API existing pointer with a readiness state and a Micro API existing/ inhibitor state, the scheduler re-platform readiness 304n shows a re-platform readiness percentage with an AWS batch readiness state and an Azure Logic App ready/ inhibitor percentage, and the DB re-platform readiness 304n shows a re-platform readiness percentage with a PostgreSQL readiness state and a refactoring required blocker/ inhibitor percentage, and the like. One of the other section of the report is the visual composition 306, such as a database object spread [#Objects] 306a, a database object spread [#LoC] 306b, a platform composition by Number 306n, a platform composition by LoC 306n, and the like. In one example, the database object spread [#Objects] 306a depicts a spread or count or percentage of triggers, sequences, packages, procedures, views, materialised views, functions and tables, the database object spread [#LoC] 306b depicts a spread or count or percentage of triggers, sequences, packages, procedures, views, materialised views, functions and tables, the platform composition by Number 306n depicts a spread or count or percentage of JMS, XML configuration, Web services, Maven build, configuration, HTML, Servlet, Style sheet, reactJS, JSP, Junit, POJO, the platform composition by LoC 306n depicts a spread or count or percentage of JMS, XML configuration, Web services, Maven build, configuration, HTML, Servlet, Style sheet, reactJS, JSP, Junit, POJO, and the like.

Further in one example, various sections of the assessment report 302 provides following information sections application snapshot consist of framework/libraries/packages used, application containerization readiness, application server re-platforming readiness to Tomcat or Kestrel application server, database server re-platforming readiness to PostgreSQL or alike, containerization readiness (to Docker/Kubernetes), CI/CD readiness (unit /functional/performance test case, build and continuous integration, automation available), macro and micro API candidates, API readiness (existing and potential API in the application).

The present method provides end to end transformational capabilities by assessing the source application environment, followed by transforming them to cloud using cloud friendly design patterns and finally manages this entire journey by providing your stakeholders and decision makers with transparent visibility for quick decision making. All application dependencies starting from databases to batch programs, schedulers to message brokers will be holistically transformed to cloud. In application analysis section of the assessment report 302 the method analyses the entire source application to identify blockers in source code including design

patterns that inhibits cloud migration. All dependencies are also analysed to provide a holistic view for design and planning purposes. It discovers and analyse application components inventory, technology stacks and their dependencies, identify Application's business, data Services and their API readiness, identify the Containerization and Cloud Re-platforming blockers & DevOps readiness and efforts for changes such as JAVA vX.X to JAVA v1.8; EJB (Session Bean/ Entity Bean/MDB) to Spring Boot Service with Spring Framework (Pojo); .NET vX.X to .NET Core v2.2.

In database analysis section of the assessment report 302 the method analyses the database to analyse and discover database inventory, data objects details and database size, identify database re-platform readiness and blockers, identify the application blockers and dependencies for database re-platforming and efforts for changes such as Oracle/ SQL Server/ DB2/ Sybase to Postgres/ MySQL. In batch & scheduler analysis section of the assessment report 302 the method analyses the batch & scheduler, to analyse and discover batch program inventory and batch schedulers, identify batch scheduler re-platform readiness, blockers and dependencies, identify application batch services for batch re-platforming and efforts for changes such as Autosys/ ControlM to AWS Batch (AWS Cloud)/ Azure Logic App (Azure Cloud)/ Quartz (Private Cloud). In message broker analysis section of the assessment report 302 the method analyses the message broker to analyse and discover message broker inventory and messaging objects, identify message broker re-platform readiness, blockers and dependencies, identify application blockers and dependencies for message broker re-platforming and efforts for changes such as TIBCO EMS/ Rendezvous or WebLogic/ WebSphere or Message Queue or Sonic Message Queue to Active MQ.

FIGURE.4 depicts an exemplary view 400 of an exemplary application environment migration method, in accordance with an embodiment of the present subject matter.

In one embodiment, an application deployment view 400 with pre and post re-platforming is depicted using the method and system of a source application environment migration to a target application environment, in accordance with an embodiment of the present subject matter. A monolith application is generally belonging to a source application environment residing or accessible by a server 102 transforms or migrates to a modernised re-platformed application belonging to a target application environment residing or accessible by a cloud environment 110. Firstly, a code assessment 206 happens for the source application environment migration to the target application environment. Assessment of the application determines a quantum of change

required in code change to remove and change code inhibitors for containerization. The migration is performed with following steps, an application web server with application as accessible by the server 102 is converted to web front instances 422a, 422b, ..., 422n as the case may be. An application server (IIS cluster) 202 is further re-factored into API management layer 426 with various App macro-service instances 426a, 426b, 426c, ...426n. The macro-service instances are based on the size and complexity of the source application. In one implementation, a re-factor module 132 performs factoring and repackaging 208 of the macro services into the target application environment.

The source application database 204 is also re-factored PaaS/ CaaS cluster 424a of a target database 424. The web front instances 422a, 422b, ..., 422n and the macro-service instances 426a, 426b, 426c, ...426n are further linked to components, including but not limited to a service mesh, an AKS cluster, an App service, an ACI, and the like of the cloud computing environment 110. In one implementation, a source Oracle database is migrated to a PostgreSQL database along with a container/ orchestration platform of the cloud environment 110.

Virtual machine instances 408a, 408b, ..., 408n of the source application are migrated to container cluster of the cloud environment 110. Thus, an on-premise data centre 410 of the source application environment 410 having an on-premise ecosystem 412 is re-platformed 210/ migrated to a private/ public cloud 428 of the cloud environment 110. In one implementation, components (412a, 412b, ..., 412n) of the on-premise ecosystem 412, including but not limited to, an EBS, a caching, local files, rules engines, a batch and scheduler, a message broker, an authentication protocol, security platforms, and the like are re-platformed 210/ migrated to private/ public cloud 428 independent or standard services/ components (430a, 430b, 430c, ..., 430n) including but not limited to, Azure queues, Event Hub, Service Bus, AppInsights, Redis Cache, Storage, Vnet, Azure AD, and the like.

In one implementation, the method modifies existing on-premise application to work efficiently on cloud using a container or PaaS, upgrades or changes all blockers related to source code and design patterns and all underlying libraries to make them compatible with container or PaaS, application servers and database may re-placed to reduce TCO, and monolith application may be de-coupled into services to improve efficiency and velocity.

In one example, Table C may be used as an example to illustrate a re-platforming migration.

Table C: Exemplary As-Is State to To-Be State

As-Is State	To-be State
JSF, Tiles, Servlets, Struts	JSF, Tiles, Servlets, Struts
(EJB) Session Bean	REST Client, Redis HttpSession Cache
(EJB) Entity Bean, Hibernate	Tomcat Server
	API Gateway
	REST API
	Spring Boot Macro Services
	(ORM) JPA , Hibernate
WebLogic App Server 12c	Tomcat Server 9
Oracle Database 12c	PostgreSQL

In one implementation, as-is state is JSP based UI to session EJB calls with a few data services and monolithic web application with commercial application server, session persistence and stateful services as-is and migrated to-be state is decoupled web application with JSF WebFront and RESTful Spring Boot macro-services on containers (independent scaling) over API gateway

FIGURE.5 illustrates an exemplary flowchart 500 of a method of application environment migration, in accordance with an embodiment of the present subject matter.

In one implementation, a method 500 for application environment migration is shown. The method may be described in the general context of computer executable instructions. Generally, computer executable instructions can include routines, programs, objects, components, data structures, procedures, modules, functions, etc., that perform particular functions or implement particular abstract data types. The method may also be practiced in a distributed computing environment where functions are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, computer executable instructions may be located in both local and remote computer storage media, including memory storage devices. The order in which the method is described and is not intended to be construed as a limitation, and any number of the described method blocks can be combined in any order to implement the method or alternate methods. Additionally, individual blocks may be deleted from the method without departing from the spirit and scope of the

disclosure described herein. Furthermore, the method may be implemented in any suitable hardware, software, firmware, or combination thereof. However, for ease of explanation, in the embodiments described below, the method may be implemented in the above-described system.

At step/ block 502, assess a source application code of a source application environment by a processor 122 of a system 120 of an application server 102. In one implementation, the source application code may be assessed by an assessment module 130 of the system 120.

At step/ block 504, ascertain a quantum change for migrating the source application code to a target application code. In one implementation, the quantum change may be ascertained by the assessment module 130.

At step/ block 506, forecast an assessment statistic that provides at least one functional readiness and a timeline to complete the migration of the source application code. In one implementation, the assessment statistic may be forecasted by the assessment module 130.

Thus, the method 500 helps in application environment migration by providing the forecasted assessment statistic and a timeline for migration from a source application environment to a target application environment.

FIGURE.6 illustrates another exemplary flowchart of a method of application environment migration, in accordance with an embodiment of the present subject matter.

In one implementation, a method 500 for application environment migration is shown. The method may be described in the general context of computer executable instructions. Generally, computer executable instructions can include routines, programs, objects, components, data structures, procedures, modules, functions, etc., that perform particular functions or implement particular abstract data types. The method may also be practiced in a distributed computing environment where functions are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, computer executable instructions may be located in both local and remote computer storage media, including memory storage devices. The order in which the method is described and is not intended to be construed as a limitation, and any number of the described method blocks can be combined in any order to implement the method or alternate methods. Additionally, individual blocks may be deleted from the method without departing from the spirit and scope of the disclosure described herein. Furthermore, the method may be implemented in any suitable

hardware, software, firmware, or combination thereof. However, for ease of explanation, in the embodiments described below, the method may be implemented in the above-described system.

At step/ block 602, assess a source application code of a source application environment by a processor 122 of a system 120 of an application server 102. In one implementation, the source application code may be assessed by an assessment module 130 of the system 120.

At step/ block 604, ascertain a quantum change for migrating the source application code to a target application code. In one implementation, the quantum change may be ascertained by the assessment module 130.

At step/ block 606, forecast an assessment statistic that provides at least one functional readiness and a timeline to complete the migration of the source application code. In one implementation, the assessment statistic may be forecasted by the assessment module 130.

At step/ block 608, scan the source application code for identifying a business logic. In one implementation, the source application code may be scanned by a re-factor module 132.

At step/ block 610, generate a re-factored code for the source application code by breaking the source application code into macro-services and repackaging the macro-services in accordance with the target application code while retaining the business logic. In one implementation, the re-factored code may be generated by the re-factor module 132.

At step/ block 612, update components of the source application environment as per the forecasted assessment statistic and the re-factored code. In one implementation, the components of the source application environment may be updated by a re-platform module 134.

At step/ block 614, migrating the source application environment to the target application environment by re-platforming the updated components and the re-factored code of the source application environment to the target application environment. In one implementation, the source application environment may be migrated to the target application environment by the re-platform module 134.

Thus, the method 600 helps in application environment migration by providing a forecasted assessment, re-factoring the code and re-platforming the application for migration from a source application environment to a target application environment.

The present solution is used to re-platform complex applications to cloud and deploy them in production environments. A few examples are shown using the method and system of the source application environment migration to a target application environment, in accordance with an embodiment of the present subject matter. The following shows preparation, planning and execution along with outcome of the studies. In one example, a cloud re-platforming of JAVA Batch App for a leading secondary mortgage firm, a source application environment belongs to a leading secondary mortgage company providing mortgage-backed securities that is a multi-billion-dollar enterprise with operations in USA and is also a publicly traded firm. The primary business challenges are complex JAVA Batch application developed 10 years ago on Autosys & EJB on WebLogic MQ by using Message Driven Bean EJB (MDB) Pojo Services, JDBC etc. alongside using Oracle database. This application is business-critical and handles key long-running processes. Non-availability of the key SME who developed this application posing challenges in cloud re-platforming. Using the present solution approach, the Oracle database was re-platformed to Postgres using the method steps, the Autosys Scheduler was re-platformed to Quartz and back-end EJB (MDB) were re-platformed to independent Spring JMS with Spring Boot/Batch Service Apps with REST end points. Each service App were then deployed on separate containers on AWS ECS VPC providing infinite scalability, high availability, high fault tolerance and improved performance. Centralized JOB dependencies of Autosys were refactored to self-managed dependencies using App specific quartz scheduler resulting in reduced operational complexity. Among the top benefits, TCO reduction by over 50%, mainly due to the elimination of WebLogic & Autosys and also by containerization, increase in development productivity due to macro-services and CI/CD automation, total cost of implementation is reduced by over 50% using automation, implementation is to be completed in 6 weeks (the same would have taken 9 months if done manually).

In one another example, present solution is used for cloud re-platforming of JAVA Web App for a leading financial services firm, a source application environment belongs to a leading financial services company that caters to corporations, institutional investors, and individuals, multi-billion-dollar company and one of the largest banks in the United States, operations in US, Europe, Australia, Asia Pacific etc. The primary business challenges are complex JAVA web application developed 12 years ago on WebLogic (App Server and MQ) by using Session EJB, Entity Bean, Struts, JSP etc. This application is business-critical and handles key Financial business processes, non-availability of the key SME who developed this application posing challenges in cloud re-platforming. Using the present solution approach, the source application

is to be broken into Web front-end and Service Back-end. The back-end EJB (Session Beans) is re-platformed to independent Spring Boot Service Apps with REST End Points. The Web Front and Each Service App are then deployed in Tomcat on separate containers on Kubernetes Cluster on Private Cloud providing infinite scalability, high availability, high fault tolerance and improved performance. The web front and macro services are made stateless with Redis cache. The Service REST API End Points are secured with oAuth2. Automated Testing + CI/CD is to be enabled for Web-Front and All Macro Services. Among the top benefits, TCO reduction by over 50% mainly due to elimination of WebLogic & MQ, implementation cost reduction by more than 50%, manual efforts reduction by 50-75%, developer productivity improvement by up to 25% due to macro services and CI/CD automation, cloud re-platforming completed in just 6 weeks as compared to 8 months if done manually.

In one another example, present solution is used for cloud re-platforming of .NET Web App for a leading us consulting firm, a source application environment belongs to one of the largest professional services and accounting audits with content management firms with employees in over 700 offices in 150 countries around the world. The primary business challenges are Complex .NET web application developed over the years using ASP .NET, C#, Telerik ORM and SQL Server. This application is business critical handling the key B2C function for a leading Professional Service provider. This task became more complicated due to the non-availability of key SME who developed this application. Using the present solution approach, the source application is to be broken into Web front-end and Service Back-end. The back-end legacy .NET services with Telerik ORM is re-platformed to independent .NET Core Service Apps with entity framework with REST End Points. The web front and each service App is then deployed in Kestrel & NGINX on separate containers on Azure cloud providing infinite scalability, high availability, high fault tolerance and improved performance. The web front and macro services were made stateless with Redis cache. The service REST API end points were secured with oAuth2. Automated Testing + CI/CD was enabled for Web-Front and All Macro Services. Among the top benefits, re-platform resulted TCO reduction of over 45% mainly due to elimination of IIS & Telerik ORM, total cost of implementation is reduced by over 45%, increased in development productivity due to macro services and CI/CD automation, completed in six weeks compared to 7 months if done manually.

Although implementations of system and method for application environment migration have been described in language specific to structural features and/or methods, it is to be understood that the appended claims are not necessarily limited to the specific features or methods

described. Rather, the specific features and methods are disclosed as examples of implementations for application environment migration.

Claims

1. A method for application environment migration, the method comprising:
 - assessing, by a processor (122) of an application server (102), a source application code of a source application environment;
 - ascertaining, by the processor (122), a quantum change for migrating the source application code to a target application code;
 - forecasting, by the processor (122), an assessment statistic (302), wherein the assessment statistic (302) provides at least one functional readiness (304) and a timeline (306) to complete the migration of the source application code;
 - scanning, by the processor (122), the source application code for identifying a business logic;
 - generating, by the processor (122), a re-factored code for the source application code, wherein the re-factored code is generated by breaking the source application code into macro-services (426a, ..., 426n) and repackaging the macro-services (426a, ..., 426n) in accordance with the target application code while retaining the business logic;
 - updating, by the processor (122), components of the source application environment as per the forecasted assessment statistic (302) and the re-factored code;
 - migrating, by the processor (122), the source application environment to the target application environment, wherein the migration re-platforms the updated components and the re-factored code of the source application environment to the target application environment.
2. The method as claimed in claim 1, wherein the quantum change for migrating the source application code to the target application code is calculated on the basis of the size and complexity of the source application environment.
3. The method as claimed in claim 1, wherein the re-factored code is generated utilizing a continuous integration and deployment framework (150) and an automated test framework (150).
4. The method as claimed in claim 1, wherein the assessment statistic (302) provides at least one inhibitor to complete the migration of the source application code.

5. The method as claimed in claim 1, wherein scanning the source application code includes scanning links between components of the source application environment.
6. The method as claimed in claim 1, wherein repackaging the macro-services (426a, ..., 426n) in accordance with the target application code is completed while retaining links between components of the source application environment.
7. The method as claimed in claim 1, wherein the macro-services (426a, ..., 426n) are identified by an AI Engine (152).
8. The method as claimed in claim 1, further comprising: migrating, by the processor (122), libraries and design patterns from the source application environment in accordance with the target application environment.
9. The method as claimed in claim 1, further comprising: implementing, by the processor (122), security rules from the source application environment in accordance with the target application environment.
10. The method as claimed in claim 1, further comprising: packaging and building, by the processor (122), each service into a separate container by generating an application program interface proxy utilizing a gateway and a continuous integration and continuous deployment framework (150).
11. The method as claimed in claim 1, further comprising: accessing, by the processor (122), a plurality of storage units (144) comprising a business logic, links, rule engines, libraries of available environments, standard tools, and coding languages.
12. The method as claimed in claim 1, wherein the source application environment migrates to the target application environment comprising the steps of: a source to target environment re-development, a source to target application re-factoring, a source to target application re-hosting, and a source to target application re-platforming.
13. A system for application environment migration, the system comprising:
a processor (122); and

a memory (126) coupled to the processor (122), wherein the processor (122) executes a plurality of modules (128) stored in the memory (126), and wherein the plurality of modules (128) comprising:

an assessment module (130), for assessing a source application code of a source application environment and ascertaining a quantum change for migrating the source application code to a target application code, wherein the assessment module (130) forecasts an assessment statistic (302) that provides at least one functional readiness (304) and a timeline (306) to complete the migration of the source application code;

a re-factor module (132), for scanning the source application code to understand a business logic and generating a re-factored code for the source application code, wherein the re-factored code is generated by breaking the source application code into macro-services (426a, ..., 426n) and repackaging the macro-services (426a, ..., 426n) in accordance with the target application code while retaining the business logic;

a re-platform module (134), for updating components of the source application environment as per the forecasted assessment statistic (302) and the re-factored code and migrating the source application environment to the target application environment, wherein the migration re-platforms the updated components and the re-factored code of the source application environment to the target application environment.

14. The system as claimed in claim 13, wherein the quantum change for migrating the source application code to a target application code is calculated on the basis of the size and complexity of the source application environment.
15. The system as claimed in claim 13, wherein the re-factored code is generated utilizing a continuous integration and deployment framework (150) and an automated test framework (150).
16. The system as claimed in claim 13, wherein the assessment statistic (302) provides at least one inhibitor to complete the migration of the source application code.
17. The system as claimed in claim 13, wherein scanning the source application code includes scanning links between components of the source application environment.

18. The system as claimed in claim 12, wherein repackaging the macro-services (426a, ..., 426n) in accordance with the target application code is completed while retaining links between components of the source application environment.
19. The system as claimed in claim 13, wherein the macro-services (426a, ..., 426n) are identified by an AI Engine (152).
20. The system as claimed in claim 13, wherein the re-factor module (132) further comprising: migrating, by the processor (122), libraries and design patterns from the source application environment in accordance with the target application environment.
21. The system as claimed in claim 13, wherein the re-platform module (134) further comprising: implementing, by the processor (122), security rules from the source application environment in accordance with the target application environment.
22. The system as claimed in claim 13, wherein the re-factor module (132) further comprising: packaging and building, by the processor (122), each service into a separate container by generating an application program interface proxy utilizing a gateway and a continuous integration and continuous deployment framework (150).
23. The system as claimed in claim 13, wherein the plurality of modules (128) further executes accessing, by the processor (122), a plurality of storage units (144) comprising a business logic, links, rule engines, libraries of available environments, standard tools, and coding languages.
24. The system as claimed in claim 13, wherein the source application environment migrates to the target application environment comprising the steps of: a source to target environment re-development, a source to target application re-factoring, a source to target application re-hosting, and a source to target application re-platforming.