Applying J2EE patterns to develop a SOA-based architecture for ecological niche modelling

Renata L. Stange, Fabiana S. Santana, Antonio M. Saraiva

Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, nº 158, sala C2-56, Engenharia Elétrica, Cidade Universitária, São Paulo, 05508-900
rlstange@usp.br; fabiana.santana@usp.br; amsaraiv@usp.br

Abstract - The purpose of ecological niche modelling is to obtain a probabilistic species distribution based on species localization and environmental data. The problem is highly complex from the architectural viewpoint because its requirements include different distributed, interoperable and integration issues. Monolithic solutions were proposed and implemented but they were not able to achieve all requirements. Therefore, it is not unusual the researcher to need many different software packages to perform a complete ecological niche modelling process. A reference Service Oriented Architecture, SOA, was proposed to improve the previous monolithic solutions and enable many missing functionalities, such as distributed processing and data exchange among different providers. The proposed architecture was organized into layers and, to implement the application corresponding to this architectural solution, the adoption of standards was required to enable distributed operations and to allow component reuse and the evolution of the ecological niche modelling technique with minimum development and integration efforts. The J2EE patterns were chosen to accomplish part of the implementation task because of its adequacy to the SOA architectural requirements, since essential facilities to implement this category of solutions are contemplated by J2EE platform. This work describes the application of J2EE patterns in order to develop and implement a SOA-based architecture for solving the ecological niche modelling problem, offering a highly improved solution to the researchers of the biodiversity area, due to its distributed nature. Besides, as standards solutions were considered in all phases of the system project and development, this implementation does not present the same restrictions of the other available software packages for ecological niche modelling, specially related to the integration viewpoint. In addition to solve the ecological niche modelling specific problem, this work shows the patterns application and the main steps required to develop and implement solutions to problems with similar requirements, thus it also may be used as a reference to solve problems in other areas.

Keywords: J2EE, SOA, patterns, ecological niche modelling, system architecture.

I. INTRODUCTION

Ecological modelling main purpose is to present the geographic distribution of biological species. This distribution may be obtained by applying ecological niche concepts, which are related to the conditions that enable species survival without external factors, such as human interference, in a widely applied technique known as ecological niche modelling [18].

The technique combines environmental and ecological data species with occurrence points so as to generate models able to describe the probabilistic distributions of a species in a specific area. It provides a way to generate a model to predict potential geographical distribution of species in time and space dimensions, considering past, present, and future projections in addition to extrapolation of the model to areas environmentally similar to those where species in fact occurs.

All input data must be in the form of geo-referenced coordinates (latitude and longitude). Species data are related to localities of species occurrence while environmental layers are selected among biotic and abiotic factors, such as temperature, precipitation and vegetation.

Despite limitations of the technique by disregarding external factors, ecological niche models were already applied to propose scenarios for sustainable use of the environment [3] to evaluate the potential of invasive species [19] and climatic changes impacts on biodiversity, to delineate potential routes of infectious diseases [19] and to indicate potential priority areas for conservation [16], among many other examples.

The overall problem is highly complex, since many different types of integration skills are required. From the software engineering viewpoint, problem requirements are related to a wide variety of modelling techniques, many different types of data analysis providing possibly from different data sources and in different formats and transmission requirements [23]. Therefore, the adoption of information systems is essential for ecological niche modelling success.

The distributed nature of the problem lead to the conclusion that a service-oriented architecture (SOA) is an adequate alternative to solve this problem and, in fact, [23] already proposed a reference SOA-based architecture for biodiversity modelling systems. A general overview of this solution is presented in Fig. 1.

The service bus receives requests from applications and calls appropriate services; after processing is finished, answers are stored in the model repository and the client is notified (e.g.: e-mail, SMS). Repository should store models that require a long processing time, thus they may be used in asynchronous way and the user does not need to wait logged into the portal.

Clients may access applications using a portal, which offers a standard interface. Applications interact with the service bus to invoke potentially geographically distributed services. Service bus must guarantee the delivery of the requests and transform the data before calling a service, if necessary.

Other services required for the modelling process [24], such
as modelling services, geospatial services and cluster services, may be hosted in many different providers available on the Internet. Any service technology may be applied for this implementation, such as web services [25].

Note that this reference architecture may be applied for any ecological modelling system. It is only necessary to replace the ecological niche modelling methods for other, based on different modelling approaches and, if necessary, to increase the providers list in order to fulfill the new problem requirements.

Therefore, results presented in this paper are more general than the ecological niche modelling domain purposes and may be adopted for other related system package implementation. However, as this work is part of the openModeller project [http://openmodeller.sourceforge.net/], which is focused on ecological niche modelling, a solution for this specific problem, based on this reference architecture, is already implemented and available on the Internet [www.pcs.usp.br/~laa/]. Fig. 2 presents the applications’ Home Page.

II. MATERIALS AND METHODS

A. Ecological niche modelling process

The ecological niche modelling process is complex and require from the user the knowledge of all its steps in addition to the expertise to potentially have to deal with many technical issues and software packages, which may not be directly related which his/her field of study.

The process is described in details in [24]. It starts by defining the experiment purposes of some of its restrictions, such as the area to be studied and the resolution of each cell. Following steps purposes are to obtain and prepare environmental data and species occurrences points related to the niche model, which depends on each species to be studied. For instance, a set of layers for climate and vegetation may be the required as environmental input data for such a species while other species may require soil information. Next two steps consist of algorithm chosen and parameters definition. Algorithms are related to modelling methods and they usually have parameters to guide their behaviour. Once the choices are performed, the software package (e.g.: openModeller) generates and projects a model onto a map of a pre-defined georeferenced region. The software package may also evaluate the model using some statistical measures and, finally, the researcher must decide if the model is acceptable or, eventually, to return to a previous step and re-start the modelling process.

Some important requirements for ecological niche modelling are to provide necessary technical environment to solve problems with acceptable performance skills, to offer metadata integration, to allow automated data importation, to offer GIS integration and GUIs projected considering usability concepts.

Among available solutions for ecological niche modelling are openModeller [http://openmodeller.sourceforge.net/], DesktopGarp [http://nhm.ku.edu/desktopgarp/] and MaxEnt [http://www.cs.princeton.edu/~schapire/maxent/]. All of them are based on monolithic architectures and not necessarily
consider technology standards in their development.

Therefore, integration problems are relatively frequent. Sometimes, it is necessary to construct specific plug-ins to provide integration and, in other times, it is simply not possible. These represent a considerable disadvantage, since integration requirements are an intrinsic part of the modelling problem, due to its distributed nature.

Environmental data bases and species collections are available worldwide and to obtain and adequate the data to same georeferenced format and resolution may be a challenge. Other challenges are related to the use of cluster machines, which may require special algorithmic treatment.

Besides, as the biodiversity modelling science is constantly expanding its boundaries, new methods, algorithms, data sources and applications are being developed. Therefore, software architectures with more facilities to integrate new resources and with more interoperable skills are certainly more indicated.

B. Basic concepts in software architectures

Reference models, architectural styles and reference architectures are initial software design decisions [14] to capture elements of the system [21], architectural mechanisms and styles [1]. Reference models are standard decompositions of a problem into parts so as to solve problems cooperatively. Architectural pattern are descriptions of elements and relationships, in addition to software constraints, to define a family of architectures. Reference architectures are reference models mapped onto software elements.

Software architectures [7] represent fundamental organizations of systems. They should be presented as collections of interrelated components to guide the design (and evolution) of a software package. Therefore, a system is a collection of components [1].

In order to increase reusability and modifiability of systems, architecture should be organized into layers [1]. In this paper, the layers will be considered as presented in Fig. 3. They are: 1) Client – composed of devices to access the system, e.g.: web browsers, wap-phones or pagers; 2) Presentation – composed of services offered by a web portal so as to create a single entry point for the system; 3) Integration – composed of resources to orchestrate services using some service-based technology in order to implement a complete business process; 4) Business – composed of sets of services required to execute business processes; and 4) Resources – data base applications, legacy systems and other devices required for ecological niche modelling.

SOC is a new computing paradigm which applies services for software development. It has evolved from component-based software frameworks, such as J2EE, CORBA and .NET, and important companies (e.g.: IBM, Microsoft, SUN) consider Web services as an adequate approach to SOC adoption [6].

SOA is the architectural paradigm adequate to SOC. Therefore, applications that present distributed requirements, should consider SOA as architectural pattern and SOC as development paradigm [6]. Fig. 4 presents the architectural paradigms over time, from Monolithic to Services-based solutions. Service-based solutions do not require Web implementation, but it is usually a better approach to systems which have high distribution needs and data imported from service providers which also adopt Web services as a principal communication pattern.

Thus, the distributed nature of ecological niche modelling leads to a SOA-based architecture definition based on SOC computer paradigm. Many SOA definitions consider a relationship between service providers, which register their services in a central repository, and service consumers, which query the repository to search for each requested service. They suppose any service can directly interact with each other, without explaining the differences among the middlewares standards and the service-oriented technology [25].

Ecological niche modelling may be a process which requires much effort from the researcher and, in some cases, also in terms of computer resources. In order to dealing with large amout of data and improve the performance of current solutions, a cluster was adopted. In a monolithic desktop-based solution, already developed, a cluster remote connection may lead top several kind of integration problems (e.g.: synchronization and data exchange). Furthermore, the cluster also have a limited number of resources, what may require a treatment based on a centralized queue to control the access to these resources. An web-based
approach with ESB resources, which is part of the adopted solution, may help solving the cluster access problems in a more effective way.

To approach such questions, the following aspects should be defined in a SOA context [25]:

1) Interfaces and contracts to describe how services are provided and offered;
2) Definition of communication styles which are available for the interaction among providers and consumers, including contents and semantics;
3) Search and registration of services to make them known and localizable;
4) Definition of an infrastructure to state and activation of services; and
5) Processes to allow the combination of different and independent services in order to accomplish a business activity, which is related to service coordination and orchestration.

The implementation of SOA-based solutions must consider patterns able to decouple clients and service implementations, which may be achieved using interfaces. These interfaces may work as abstraction layers in order to remove clients’ and services’ dependency and to achieve loosely coupling. The overall purpose must be not to follow a monolithic approach but provide a distributed and integrated solution. According to [25], J2EE standards are a possible solution to SOA implementation.

III. THE APPLCATION OF J2EE STANDARDS

Applying SOA-based architecture concepts to build software infrastructures is essential to reduce dependencies among applications. On the other hand, it is necessary to use architectural concepts to effectively develop softwares for running on these infrastructures. Several patterns have been propose to introduced best-practice for developing web-based systems and some of these patterns can be applied to SOA.

According to [25], some purposes should be achieved in the development of SOA-based applications, such as decouple clients and services from infrastructural issues (e.g.: service discovery, loose coupling and transparency) and increase application development productivity.

Developing components and services using standards represents an excellent alternative to achieve the necessary aims for the evolution of applications over time. Patterns are suggested [25] for developers to apply in the leverage of these proposals.
Fig. 5 presents a simplified view of the application of core J2EE patterns in the context of SOA-based architectures.

The Business Delegate pattern turns the aspects of remote communication with services transparent to client applications, such as service discovery, message transfer or exception handling.

The J2EE Service Locator pattern must be defined to access a specific service, since a client must first locate it. Enterprise applications require a way to look up for the service objects that provide access to distributed components. In order to achieve this purpose, developers may hard-code location information in the client code, but this is not recommended because it increases the coupling between client and service.

The Lookup Service represents, in Fig. 5, the correspondent pattern to solve this architectural decoupling problem.

The Session Facade offers functionalities to initiate and manage the whole processes. This pattern should reduce the communication overhead in addition to decouple clients from any back-end changes, such as workflow reorganizations.

Back-end functionalities and services should not depend on low-level persistence mechanisms. Hence, Data Access Object pattern represents an intermediate layer able to shield services and components from specific definitions of database and enterprise information systems.

In spite of J2EE patterns were originally used to build only Java EE applications, Stal [25] shows that they also may be applied to develop SOA-based environments.

In this paper, some well-known Core J2EE patterns [http://java.sun.com/blueprints/corej2epatterns/Patterns/] are applied, among other technologies, so as to allow developers to build efficient SOA applications.

The study was conducted focusing on a reference architecture for ecological niche modelling proposed in [28]. Fig. 6 presents this reference architecture, organized into layers, as recommended in [1].

Client layer provides the application access devices. In the case of the presented solution, only web browsers accesses are implemented, but other similar devices could be used. This would require an adequation of the technology, in order to accomplish the requirements of the respective device.

Fig. 7 presents the components diagram which corresponds to part of the implementation of the SOA-based architecture for the ecological niche modelling problem. Other parts are already implemented following similar methodologies. This part was chosen because it is enough to demonstrate the main steps of the ecological niche modelling process and comprehend some of the more complex implementations requirements. To implement other steps, it was just necessary to follow same pattern definitions.

Presentation layer is responsible for receiving client requisitions, and to input and present data through a web-based graphic interface. In the solution proposed in this work, Java Server Pages, JSP, was adopted. In the ecological niche mod-
elling application, there are JSPs for dealing with species point and environmental data, algorithm choice and parameters definition, and model generation and projection.

In order to control the requisitions of system clients, a Servlet was implemented. Besides, Struts framework was used to implement the design pattern Model-View-Controller [26]. This layer corresponds to WEB component in Fig. 7.

Integration layer is responsible to integrate applications with services. In Fig. 6, this layer must contain a service bus. The adopted solution was the Open Enterprise Service Bus [https://open-esb.dev.java.net/], OpenESB. OpenESB is a Java based open source Enterprise Service Bus that can be used as a platform for both Enterprise Application Integration and SOA. OpenESB is built based on open standards [https://open-esb.dev.java.net/].

This layer also uses middlewares to integrate local and remote services, data bases, legacy systems and other necessary resources for implementing a complete business process.

Thus, the Data Access Object, DAO, pattern is used in this level to represent an intermediate layer for decoupling business tier and allowing extension and reuse of the business logic. In the application, DaoFactory class, for example, uses the JDBC middleware, which is responsible to connect a database with the informations about users and modelling experiments.

Business layer is represented in the form of several services, such as modelling, data cleaning, species data services, environmental data services, and also for controlling the business logic. In this case, it applies the business delegate pattern to separate client application from business layer details. For instance, the class which implements the business delegate pattern in ecological niche modelling application is responsible for encapsulating details of layer business and treats EJB specific exceptions for isolating them from the client.

Persistence layer represents the resources and devices which are necessary to complet an ecological niche modelling process. These resources include data bases and legacy systems, for example.

IV. CONCLUSION

The development of the proposed solution for ecological niche modelling, applying J2EE patterns has proved to be, simultaneously, simple and effective for attending distributed, interoperability and integration requirements. Patterns application has also shown to be very important for decoupling software layers, which was one of the main requirements of the reference architecture. Besides, the software tools which support J2EE are also easy to deal with, and the implementation process is relatively faster than applying other related technologies.

![Diagram](https://open-esb.dev.java.net/)
As the standards were followed during all implementation process, some problems presented in other software packages developed for ecological niche modelling does not occur in this solution and it also improves the overall approach to this problem, by predicting the integrated usage of a cluster. In addition, as J2EE is a widely applied platform, the adoption of this technology has several advantages from the commercial viewpoint, since there are responsible companies interested in maintain its continuity.

V. ACKNOWLEDGEMENTS

The authors are grateful to FAPESP – Fundação de Amparo a Pesquisa do Estado de São Paulo – Brazil, for the support to the openModeller project, under grant 04/11012-0.

REFERENCES


