DESIGN AND IMPLEMENTATION OF TWO SOFTWARE FRAMEWORKS FOR OGC WEB PROCESSING SERVICE DEVELOPMENT

Tomasz Kubik
Institute of Computer Engineering, Control and Robotics, Wroclaw University of Technology, Janiszewskiego 11/17, Wroclaw, Poland
e-mail: tomasz.kubik@pwr.wroc.pl

Abstract

Web Processing Service (WPS) specification is one of the web services specification developed by the Open Geospatial Consortium (OGC). The specification standardizes the service interface by providing the definition of the service operations that facilitate the publishing of geospatial processes (which in fact can by associated with any arbitrary computations), and the discovery of and binding to those processes by clients.
The WPS can facilitate the exchange of data between several clients, which themselves might be network services, creating a whole cascade of processing steps. Thus, the WPS can play a role of an intermediate layer in the systems build along with the service-oriented architecture (SOA) paradigm, transmitting information between the client and particular server functionality. The WPS can serve also as a backbone for the implementation of Transformation Service recognized by the INSPIRE Directive as one of the five network service types essential for an interoperable Spatial Data Infrastructure (SDI).
The paper presents two different technological approaches to the WPS implementation: plWPS project based on .NET technology and jWPS project based on JAVA technology. Each project provides software designed and implemented in order to simplify custom OGC compliant WPS implementation. The paper also highlights the advantages and weaknesses of the specification recognized during implementation of these software frameworks. The software frameworks presented include samples of simple web processing services implementation. Both solutions are available as open-source software, and thus they can significantly contribute to the development of SDI, and in particular to the creation of various spatial data processing services.

1. Introduction

Web Processing Service can play a role of intermediate layer (the so-called middleware) within SOA based systems. The service does not need to process spatial data strictly, but it can serve as Remote Procedure Call (RPC) implementation in a very broad context. Thus, the WPS implementation offers the service interface allowing data transmitting from and to the customer, processing them on the service server side. This feature can by utilized in various ways and can also provide the base for the web

services orchestration (Cepicky, 2007; Diaz, 2008; Friis-Christensen, 2007; Foerster, 2007; Ladra S., 2008; Lanig S., 2008; Michaelis 2009, Stollberg, 2007).
WPS standard history is relatively short. The first document with WPS version 0.1.0 specification was announced on 5 May 2004. Another document released on 16 September 2005 and was a proposition of WPS version 0.4.0 specification. The first official WPS standard appeared on 8 June 2007 with 1.0.0 number. All documents are available at OGC web page (http://www.opengeospatial.org/standards/wps).
At the beginning WPS service implementations were small solutions, designed to test the service rather than to implement the productive solution on a larger scale. However, the more mature was the standard was, the more sophisticated became the solutions. Two most known open-source software frameworks containing WPS implementation are deegree (http://download.deegree.org/) and 52º North (https://52north.org/).
The paper presents two approaches to the WPS service implementation, which differs by the type of underlying technology: one based on .NET, and one based on JAVA. In both cases, the software framework designed and implemented simplifies custom OGC compliant WPS implementation. The names of frameworks follow the corresponding projects names: plWPS (for .NET based solution) and jWPS (for JAVA based solution). The frameworks are software structures developed for a specific application domain, which can be reused in the implementation of various systems in this domain.

2. Two software frameworks

2.1. WPS in .Net

Implementation of WPS service in .NET technology started as C# project within Visual Studio 2005 environment, conforming WPS 0.4.0 specification (Kubik, 2007). The purpose of this project was to verify the specification in practice, with no intention of implementing any service in a real production environment. Therefore, the service efficiency parameters by assumption were low and the project focused mainly on the implementation of the standard as a dynamic-link library. The project outcome was a custom implementation of WPS service accompanied with a simple interface to the service's administration and a generic client sample. The WPS service implemented offered all the three operations described in the standard: GetCapabilities, DescribeProcess and Execute. The framework design incorporate code optimization by the use of some specific mechanisms, such as: generic types, exception managing, and processing XML (eXtensible Markup Language) documents with DOM (Document Object Model) methods, function and templates. Processing XML documents with DOM methods is reasonable when these documents are relatively small. It is not suitable for processing larger documents. Nevertheless, because of low efficiency requirements accepted in the project, the DOM limitations were not significant.
A particularly difficult task manifested itself during implementation of the information model defined in the series of ISO standards related to the geographic information and maintained by the technical committee TC 211 Geographic information/Geomatics (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=

54904&published=on). Since the WPS standard defines basic and complex data types, and provides as an option the opportunity to declare the data using Geography Markup Language (GML, defined in ISO19139), the implementation of WPS handled only some pieces of this language necessary to achieve functionality expected. Similar problems were encountered while implementing GetCapabilities operation which had to excerpts some parts from the ISO19115 standard.

A side effect of the first WPS framework implementation was the recognition of the problems arising in the course of practical standard implementation. It turned out that the WPS specification has gaps and weaknesses in relation to the management of data processed, including data provided by the customer to the service and produced processing results. It was also noted that there is a lack of definition of supervisory mechanisms which would optimize the data transfer (which can be short vectors or large-capacity raster as well). Some of these deficiencies were completed in the WPS 1.0.0 specification.

The experience gained through the practical implementation helped in setting ambitious objectives for the next WPS framework. Thus the main objective of a new plWPS project was to develop advanced software framework and services samples, following WPS standard version 1.0.0, which would be used in a production environment. The expectation was to take into account all standard fixes and enhancements, particularly the possibility of using Simple Object Access Protocol (SOAP) to transfer requests and responses and generation of service description in Web Services Description Language (WSDL). The adoption of these objectives enforced the changes in the technology used. A new architectural solution proposed needed some optimization aimed at fulfilling requirements of intense exploitation. Therefore the technology migrated from the basic .NET libraries into ASP.NET. In contrast to the .NET WebServices (the technology used in the previous project), where the access to the operations offered by the service required a number of methods with the attribute [WebMethod]) to be implemented, ASP.NET (the technology adopted in plWPS) has enabled the creation of a single access point to the service. This point received a form of the method recognizing the type of customer information and routing it to the appropriate internal mechanism.

Another improvement build in plWPS framework was a mechanism of exceptions throwing. This mechanism allowed handling unsupported events, which do not have matching exception of the standard type. For these cases the mechanism generates an exception of NoApplicableCode type with text attachment providing detailed error description. The definitions of the exception types created followed the rules described in the WPS version 1.0.0 specification.

A key issue in plWPS framework implementation was its re-architecting. In order to increase flexibility of the solution a mechanism for service configuration trough the external configuration files was introduced. The service has been placed in a container – Internet Information Services (IIS) application server, and the configuration files were loaded dynamically by the service during its execution.

The general configuration allows defining: WPS protocol version, supported languages, path to the configuration files of the implemented operations, the location of directories for temporary files, and many other things. Configuration files of the implemented

operations consist of the XML documents and are used to build a dynamic response to the GetCapabilities request (which provides description of the service) and DescribeProcess demand (which provides a description of the process).
With this configuration, the modification of the response or the formation of a new one does not need any code compilation. Just editing existing or creating new configuration files solves the problem. This method was also used for providing replies from the service in the languages other then the default choice. Number of languages supported by the service depends on the language files provided. plWPS framework supports two languages: Polish (which is the default language) and English but it can be extended to any number of languages.
It turned out that one of the major problems with WPS implementation is the optimization and management of the temporal data on the server side. Since the standard does not define any related procedures, it was assumed, that temporal data will be stored in the catalogs controlled by the local process. The process will be responsible for removing the data older than the limits set by the service administrator. In the Windows environment this process was implemented as a system service. In the Linux operating system this task was implemented as a daemon running especially designed scripts.
Additionally a safeguard policy in a form of parameterization of disk quota limits for the process was incorporated in the solution. In the context of the WPS this meant launching a small system service that checks whether the volume of the client's input and output data exceeds the thresholds defined by the administrator.
WPS standard does not provide exact rules on how to control the process' execution. Therefore the system service or a local process should also verify whether any greedy algorithm was lunched. Such algorithm should be terminated safely, and the information with an error description should be sent to the service's client.
In the plWPS project, a new model of classes was designed and implemented. Thanks to this model, building any arbitrary process should go quickly and efficiently, and attaching resulting library to the existing service's resources should not cause any problems. The model designed offers some references to the events and delegates through which the service can acquire more control over the processes, and methods initiating and terminating the processes.
plWPS implementation supports HTTP GET and POST requests with KVP (key-value pair) and SOAP binding. The encoding schemas for operation requests provided are as follows: GetCapabilities – KVP (GET), XML (POST, SOAP); DescribeProcess – KVP (GET), XML (POST, SOAP); Execute – XML (POST, SOAP).
The way of the use of plWPS framework was illustrated with an example of the service allowing reaching selected GRASS functions remotely (GRASS is an open source software package used for geospatial data management and analysis, image processing, graphics/maps production, spatial modeling, and visualization available at http://grass.osgeo.org/). In this example, each GRASS function call was a script running within the process inside WPS implementation.

2.2. WPS in JAVA

An alterative implementation to the plWPS was designed in JAVA technology. The corresponding jWPS project was aimed at providing a software framework for WPS version 1.0.0 service implementation accompanied with a running example of such service. Following SOA philosophy, the services exposed in the network should be able to communicate using the same protocol independently from the implementation technology chosen.

The reasons that caused the project to be implemented in JAVA were the following project assumptions: efficient implementation, high scalability and flexibility of the solution, platform independency, easy deployment through the mechanism of containers, and others. The choice of this technology, in addition to the guarantee of fulfilling objectives just mentioned, gives an opportunity for designing an intuitive mechanism for extensions creation.

The final solution was optimized for the Apache Tomcat servlet container. With small changes (if at all they are necessary) this solution can be deployed on other containers, such as JBoss. Conforming to the assumption of solution simplicity the solution implemented is configurable, with a number of parameters kept at minimum level.

The main advantage of implemented jWPS software framework is the easiness of its use. The jWPS based web service runs in a servlet container which expose implemented processes to the clients. In general, the resulting solution has low hardware requirements. These requirements are the sum of container and the Java virtual machine requirements plus small jWPS coating related to the requests processing and results production. However, in some special cases, these requirements can grow significantly. This can happen for the exhaustive service process implementation. Therefore the target hardware requirements should be determined in advance based on not only obvious hardware parameters, but also considering the needs of the service processes implementation. An additional advantage of the jWPS software is its independence from the third party software. jWPS uses two external libraries only: Xerces (for XML documents processing) and velocity (for building XML documents based on templates).

During jWPS implementation encountered similar problems as in plWPS case, caused by inaccuracy of the WPS standard. Standard, apart from definitions of process input and output (and the definitions of many other relevant parts) gives only suggestions on how to implement the service. The relevant model expressed in Unified Modeling Language (UML) is an abstract, whose implementation can go in various ways – depending on interpretation. Therefore in the jWPS project some additional assumptions were adopted. The most important, related to the solution architecture, covered: an input parsing method, an output manufacturing method, a methodology of extending services to include new processes.

The operation requests conveying input data are transferred to the WPS service by the use of HTTP GET and HTTP POST methods with KVP or XML encoding and SOAP encoding as an option. Therefore within the framework it was necessary to provide adequate methods for requests parsing.

KVP is a simple encoding used in the requests of the GET type and processing such requests is trivial. It all boils down to the division of a string to substrings which are separated by the tokens "&", "@" and "=". With the type of POST requests input data format is XML or SOAP. Implementation jWPS supports both forms of encoding, although implementation of this feature has not been easy. The problem is that the parsing of XML documents is a difficult task when taking into account the required flexibility and the data model implemented in the service. Most of XML documents is parsed in two ways: using a rigid parsing mechanism based on XML schemas defined by the standard, or using XML translators that converts XML documents into application dependent, universal data format (such a solution is used in deegree framework). In the current implementation of jWPS rigid parsing mechanism is used.
Another problem was to provide flexibility in producing output in XML or SOAP format programmatically. In this case there are two possible solutions: generating XML documents using JAVA mechanisms (i.e. placing the algorithms inside the JAVA source code), or the use of the mechanisms based on templates (i.e. placing the relevant algorithms in the form of files with templates outside the JAVA source code). During the first attempt to the jWPS framework implementation the first method was considered, but finally the mechanism based on templates was incorporated into solution. The use of template based mechanism, in addition to simplifying the whole problem, provides a great flexibility and low sensitivity to changes in the standard. Any changes in the response format do not require any changes in the source code, but the only modification that has to be made is templates correction. In addition, the same mechanism can be used when implementing other OGC services within the framework. To generate a response based on templates the Apache Velocity library was used.
The most serious problem met in the project was to define the methodology for introducing new processes offered by the service (allowing developers using the framework to easily extend any existing service by introducing new processes) and to determine how to manage processes and processed data (what would help in keeping service resources tidy). The basic assumption adopted in the project was to provide simplicity of implementation. This assumption was valid when implementing methodology for introducing new processes. Thus a special design pattern was introduced. To create a new process in this pattern the user must extend the class WpsAbstractProcess and override its execute method. This method is the heart of the process as it defines what process really does. Class WpsAbstractProcess also provides a number of other methods that allow to read the input data (getInputDataById), change the status (setProcessFailed, setProcessSucceded, ...) and provide the output data (setOutputData). The pattern includes a mechanism ensuring catching and processing of all possible errors.
In order to solve the problem of managing of data sent to the server and stored by the server, additional storage service STORAGE was designed and partially implemented. This custom network service goes beyond the set of services described by the OGC specifications. The service offers results of processing (i.e. responses to requests) and the status of the current request processing (i.e. processing stage). It plays the role similar to the role of FTP server or HTTP server referred in the OGC specifications.

WPS service implementation based on jWPS framework together with STORAGE service constitutes a complete solutions conformant to the WPS specification. The shortcomings of the current version jWPS that were spotted include: lack of unit tests verifying the system operation, lack of integration with tools such as GRASS GIS (for the time being jWPS framework is illustrated with a simple service implementation offering a simple arithmetic operation, porting it to the GRASS is on the way), limited possibility of managing data in a service storage (which is rather due to deficiencies in the definition of a standard than to the inadequacy of implementation), the lack of requests validation (which is also missing in a standard), a strong correlation between proposed architecture and standard specific requirements. These deficiencies served as benchmarks determining the directions of framework improvement.

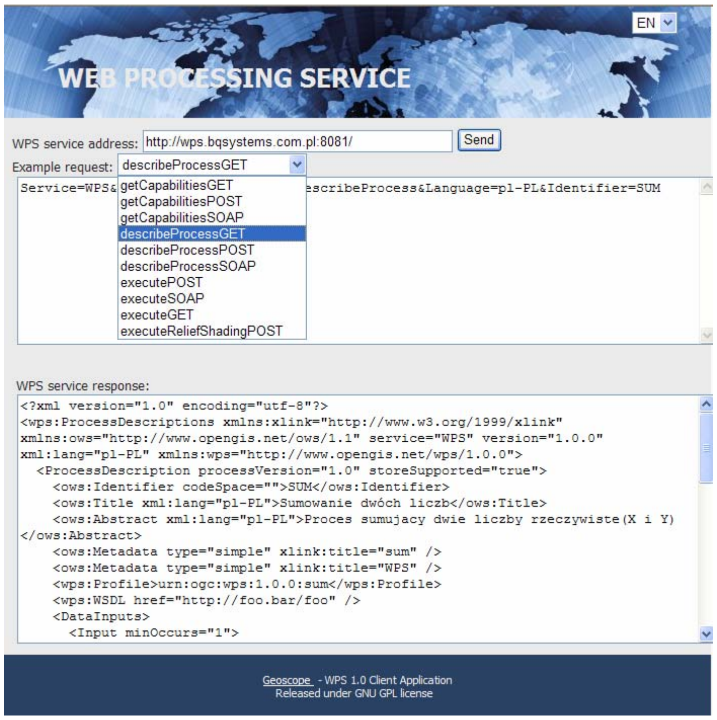
3. Experiments results

3.1. WPS in ASP

On the base of plWPS framework a service porting GRASS functionality on the internet was implemented. This service can be seen as a node of SDI enriching it with some new capabilities. For now, the service offers only a few algorithms, but the work on its extension is in progress. The software status can be regarded as Release Candidate what means that the framework is almost finished.
In order to test the WPS services a generic WPS client has been implemented and released in the form of ASP.NET pages. The client allows choosing the kind of the request encoding and editing the contents of the request. After completing, the request is sent to the WPS service of the user choice, and the service response is given back to the user. The generic WPS client interface is shown in Figure 1. The documentation, including but not limited to a detailed description of the so-called acceptance tests, belongs to the plWPS framework resources provided.

3.2. WPS in JAVA

jWPS framework is still under development. The state of the progress of the work can be described as an early alpha. Actually, some re-factorizations aimed at improving the quality of the code and its structure were initialized. The results should increase the efficiency of the code use and give a better starting point to the implementation of other services based on OGC standards. The critical parts in this context are those connected with operations of the input and output processing.
The current design of the mechanism for output generation is so flexible that it allows other web services implementation. Since the WPS standard describes the interface of the service processing data sets, the creation of the service cooperating with GRASS application is planned. Further plans have in a focus enriching the framework with implementation of other OGC web services. In order to demonstrate the simplicity of the use of jWPS framework a sample of the service that calculates a sum of two numbers is provided.



WPS service address: http://wps.bqsystems.com.pl:8081/

Example request: describeProcessGET

```
Service=WPS&                          escribeProcess&Language=pl-PL&Identifier=SUM
```

WPS service response:

```
<?xml version="1.0" encoding="utf-8"?>
<wps:ProcessDescriptions xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows/1.1" service="WPS" version="1.0.0"
xml:lang="pl-PL" xmlns:wps="http://www.opengis.net/wps/1.0.0">
  <ProcessDescription processVersion="1.0" storeSupported="true">
    <ows:Identifier codeSpace="">SUM</ows:Identifier>
    <ows:Title xml:lang="pl-PL">Sumowanie dwóch liczb</ows:Title>
    <ows:Abstract xml:lang="pl-PL">Proces sumujacy dwie liczby rzeczywiste(X i Y)
</ows:Abstract>
    <ows:Metadata type="simple" xlink:title="sum" />
    <ows:Metadata type="simple" xlink:title="WPS" />
    <wps:Profile>urn:ogc:wps:1.0.0:sum</wps:Profile>
    <wps:WSDL href="http://foo.bar/foo" />
    <DataInputs>
      <Input minOccurs="1">
```

Geoscope - WPS 1.0 Client Application
Released under GNU GPL license

Figure 1: Generic WPS client served on ASP.Net page

4. Conclusion

WPS version 1.0.0 specification does not give all answers to questions that might rise during service implementation. For example there is a lack of guidelines on process management. According to specification active processes of the WPS service can by investigated by checking their status of their execution. Although the status of the process might inform about its current state, but this information is read only. Therefore service client that lunched his own process can not influence the course of its execution (i.e. by issuing a kind of stop, suspend or resume command). It is not known how to handle the situations of process killing by the operating system (due to the long duration of its execution for example) when a fatal error appears. Of course, there might be a question whether geospatial services available can be considered as manageable parts of a distributed system (with all the baggage associated with implementation of the remote garbage collection) or not. Due to the potential risk of overloading server resources by the processes running under the service control, the possibility of process management through the service interface appears to be a legitimate expectation.

Problematic might be also the implementation of the management of the data sent to the server and stored by the server. The WPS standard does not give any specific advice here, except that the service does not provide the results generated (when status and storage functions are used by the process). The standard does not describe how long the data should be stored on the server, nor gives a direction to solve this problem. And this is a big problem, because geospatial data can occupy a significant part of the disk, so a

few service requests may completely exhaust the resources available on the server. Another issue is the inability to re-use of data already loaded on the server (for example, to call the same process, but with another set of parameters).
The way of using a status should also be clarified. Not all requests need to define their status (e.g. when a response is returned immediately). According to specification all request with the status option are process by the system which generates a response using storage mechanism.
Interesting is that it is possible to issue a request with the status option active and inactive storage – in such a case the resulting output file will be generated regardless inactive storage. Processing of large files downloaded from the web is also problematic. When the large geotiff files are set for processing, their downloading will increase network load and might take a lot of time. Therefore, processing by WPS this type of data is rather poor idea.
It seams to be a good idea to develop a standard describing programming interface for WPS processes implementation. With such a standard, it would be possible to transfer components implementing algorithms (e.g. DLLs, packages of classes, war files) between different implementations of WPS service. Currently, all components created are implementation specific and not movable.
The WPS specification defines a relatively small number of exceptions with NoApplicableCode exception used in most uncovered cases. A good complement to the standard would be a scheme of introducing user exceptions.
Finally, the use of SOAP in the WPS service, similar to other OGC services, it is very poor. It is limited only to filling body element with standard requests or responses leaving out the potential of SOAP in handling requests chains. No better looks the use of WSDL documents describing services. The current specification does not specify how to generate and share WSDL documents, and the poor examples of the tests are focused only on verifying the validity of the service protocol. Some remarks on SOAP and WSDL were provided by Sancho-Jiménez et al. (2008).
At least WPS as other OGC services does not tackle the problem of security. Anyone can call WPS processes with no restriction or data protection. The solution is to use the proxy server or another security layer (as GeoRM, which is not standardized yet).
Despite many flaws, WPS also has many advantages, which include among others: a simple interface that allows distributing calculations, the nature of an intermediate layer, the possibility of request chaining, simple interface for process definition, and the possibility of returning the result in the RAW format. Although the recognition of WPS among other OGC services (like WMS, WFS or WCS) is not so strong, the potential inherent in it can change this situation. The presented open-source framework implementations, plWPS and jWPS, can certainly contribute to increase the popularity of the WPS.

Acknowledgements

This research was supported in part by the Ministry of Science and Higher Education through grant Nr R09 011 03. The work has not been possible without collaborative effort of the programmers Bartosz Kopańczyk and Balcer Ziemowit (developers of plWPS) and Konrad Rymczak (developer of jWPS).

References

Cepicky. J. Becchi. L.. 2007. Geospatial Processing via Internet on Remote Servers - PyWPS. *OSGeo Journal*, Vol. 1. Available at http://www.osgeo.org/journal/volumel

Diaz, L. Granell, C. Gould, M. Olaya V. 2008, An open service network for geospatial data processing, In: *Proceedings of the academic track of the 2008 Free and Open Source Software for Geospatial (FOSS4G) Conference, incorporating the GISSA 2008 Conference*, Cape Town, South Africa, 29 September - 3 October 2008, pp. 410-419.

Friis-Christensen A., Lutz. M., Ostlinder. N., Bernard. L., 2007. Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions. *Transactions in GIS*, 11(6), pp.799-818.

Foerster. T. S Shffer. B., 2007. A Client for Distributed Geo-processing on the Web. In: *Proceedings of W2GIS*. Cardiff (UK). Springer: Berlin.

Kubik T., Paluszyński W., Kopańczyk B., Iwaniak A, 2007, Implementing a WPS service for automatic feature recognition in orthophoto maps. In: *Proceedings of XXIII International Cartographic Conference*, Moscow, Russia, 4-10 August 2007.

Ladra S., et all. 2008, A Toponym Resolution Service Following the OGC WPS Standard in Web and Wireless Geographical Information Systems. *Lecture Notes in Computer Science*, Vol. 5373/2008, pp.75-85.

Lanig S., Schilling A., Stollberg B., Zipf A.. 2008. Towards Standards-Based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS). In: *Computational Science and Its Applications – ICCSA 2008*. *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, Vol. 5073/2008, pp.191-203.

Michaelis, C.D.; Ames, D.P., 2009. Evaluation and Implementation of the OGC Web Processing Service for Use in Client-Side GIS. *GeoInformatica*, 13(1), pp.109-120.

Sancho-Jiménez, G., Béjar, R., Latre, M.A., and Muro-Medrano, P. R., 2008. A Method to Derivate SOAP Interfaces and WSDL Metadata from the OGC Web Processing Service Mandatory Interfaces. In *Proceedings of the ER 2008 Workshops (Cmlsa, Ecdm, Fp-Uml, M2as, Rigim, Secogis, Wism) on Advances in Conceptual Modeling: Challenges and Opportunities*. *Lecture Notes in Computer Science*, Volume 5232, Springer-Verlag, Berlin, Heidelberg.

Stollberg B., Zipf A., 2007. OGC Web processing service interface for web service orchestration-aggregating geo-processing services in a bomb finding scenario. In: *W2GIS07: Web & Wireless GIS Conference 2007*, Springer, Hong Kong.