Spicy Stonehenge: Proposing a SOA Case Study

Tiago Espinha, Cuiting Chen, Andy Zaidman, Hans-Gerhard Gross
Delft University of Technology, The Netherlands
{t.a.espinha, cuiting.chen, a.e.zaidman, h.g.gross}@tudelft.nl

***Abstract*—Maintenance research in the context of Service Oriented Architecture (SOA) is currently lacking a suitable standard case study that can be used by scientists in order to (1) develop and assess their research ideas and (2) compare and benchmark their solution(s). It is also well established in different fields that having such a standard case study system brings many benefits, in that it helps determine which approaches work best for specific problems. For this reason, we decided to build upon an existing open-source system and make it available for other researchers to use. This system is Spicy Stonehenge.**

I. Introduction

When looking at the past decade of research in service-orientation, we can observe that although a lot of fruitful research has been carried out (e.g., see [1], [2]), many of the research efforts are isolated in nature. While this isolation is not bad per se, it does hinder progress. Symptomatic of the isolated nature of research in this area is the absence of a common case study that can be used as a benchmark. Indeed, Sim et al. report that benchmarking, when embraced by a community, has a strong positive effect on the scientific maturity of a discipline [3]; it allows to easily compare solutions and to perform replication studies. In many fields of software engineering, researchers have resorted to benchmarking in order to compare approaches and advance the field.

We propose a system that is at the same time realistic, easy to understand and which most researchers should be able to use as a "standard case study system". The system we propose — Spicy Stonehenge [4] — is an extension to Apache Stonehenge and consists of an application composed out of several web services. The open-source nature of Spicy Stonehenge and its availability should stimulate researchers, that normally resort to small examples built specifically for the context of their research, to choose for Spicy Stonehenge, thus enabling the benchmarking process that the community needs.

II. Stonehenge

Apache Stonehenge[1] is a simulation of the stock market consisting of a web application and several web services. Stonehenge provides the possibility to buy and sell shares in a single stock market, with a single currency. Apache Stonehenge was built as a joint effort between Microsoft and the Apache Software Foundation to showcase service interoperability between different technologies. Our goal, however, is not to explore the field of interoperability but that of maintenance in SOA, and all that it entails. We chose Stonehenge as it provides a real world example of how services can interact

[1]Apache Stonehenge — https://cwiki.apache.org/STONEHENGE/

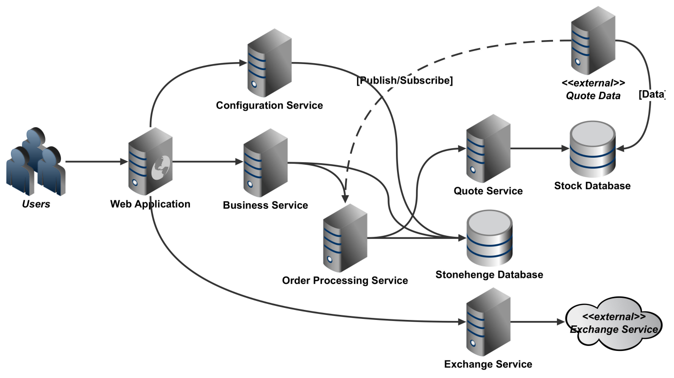

Fig. 1. Spicy Stonehenge

together to compose a software system. However, conscious of its size, we decided to extend it in order to make it more realistic and complex. We have extended it with several new features to make the system more complex on what concerns business logic and number of services. That is, we added the possibility to maintain several wallets in different currencies, to exchange money amongst the different currencies, and to use real-world data from the stock market. The result of our changes is called Spicy Stonehenge[2] which relies substantially on the business logic of the original implementation. We have also ported the original JAX-WS-based implementation to the Turmeric SOA platform[3] [4].

A. Motivation

We previously noticed that in service-oriented research there is no case study which researchers can use to compare their approaches and results [4]. Furthermore, we tried to bring forth a system that: a) is relatively easy to grasp, b) is large or at least provides many extension possibilities that all researchers can build upon and c) it must be easy to port to different frameworks. With Spicy Stonehenge we feel we have met these three criteria.

B. System Description

The current version of the system is composed out of five different services and two databases (Fig. 1). In this section we provide an overview of what each service does and further into the section, what data is stored in each table.

Also referring to Figure 1, solid arrows represent one service invoking another whereas the dashed arrow represent

[2]Spicy Stonehenge — https://github.com/SERG-Delft/spicy-stonehenge
[3]Turmeric — https://www.ebayopensource.org/index.php/Turmeric/

a publish/subscribe connection where the Order Processing service can subscribe to topics on the external service.

Services:

- The **Configuration Service** acts as a registry for all the deployed instances of the other services. All the other services need, therefore, to know in advance the endpoint of at least one instance of the *Configuration Service*.
- The **Business Service** mediates the interaction of the web application with the business logic of the system. For this reason, the Business Service contains all the operations the web application is capable of performing, i.e., buying and selling of stocks, user registration, statistical information about the market and information about stock prices.
- The **Order Processing Service** is solely responsible for processing the buying and selling of shares. It is invoked by the *Business Service* whenever a user performs a purchase or sale of shares in the web application.
- The **Exchange Service** makes use of Google's API for currency exchange. This service is invoked whenever the user explicitly requests for currency to be exchanged from a wallet in a certain currency into another wallet, with a different currency. In the future, this service will also become part of the purchase request for the cases when the user wants to buy shares in a currency A but chooses to use currency B.
- The **Quote Service** is in fact composed of two services. Referring to Figure 1, the service described as *Quote Service* is a normal *pull-based* service with a SOAP interface that the *Order Processing Service* can invoke to obtain data about specific stocks on-demand. On the other hand we also have the *Quote Data* service which performs two tasks: 1) it fills the *Stock Database* table with data and continuously updates it with data from Yahoo Finance, and 2) it provides a publish/subscribe interface (implemented using the ZeroMQ library) which other services, such as the *Order Processing Service* can bind to in order to be notified for price changes in specific stock symbols.

Databases:

- The **Stonehenge Database** contains the information necessary for the basic operation of the system. Namely it contains user information, including how much money and which stocks each user owns. It also contains information about the services' endpoints and the mapping between service instances (which instance should each service use).
- The **Stock Database** contains information about stock prices. This table is kept separately as it is meant to be filled by an external service which continuously checks whether there is new data and pushes it to the database.

C. Usage Scenarios

With these services we can then have different usage scenarios. These are summarized in Table I.

TABLE I
FEATURES AVAILABLE/PLANNED FOR SPICY STONEHENGE

Currently available features	**Planned features**
Purchase and sale of stocks	Automatic conversion of currencies
Price info about stock symbols	Multiple stock markets
Wallets in different currencies	External bank entities
Management of service endpoints	Stock options
User registration	

Our planned extensions to the existing system aim at making the interactions amongst web services more complex. For example, the existence of multiple stock markets will create the need for different instances of the Order Processing service.

III. DISCUSSION

Our personal motivation for coming up with Spicy Stonehenge was to be able to have a small, yet realistic case study with which we could work in the area of online evolution and maintenance of SOA based systems. Questions like (1) what is the actual runtime topology of the system?, (2) is a particular service out-of-use and can it be disconnected permanently?, (3) can we do online diagnosis of faulty services? and (4) what is a good time to plug-out and plug-in a newer version of a service are questions that we would like to address in our future work.

Nevertheless, we also feel that the possibilities for Spicy Stonehenge reach quite a lot farther than simply the software evolution sub-field. Not in the least because Spicy Stonehenge's source code and build material are directly available from GitHub[4]. As such, we see this paper as an open invitation to the SOA community to evaluate Spicy Stonehenge and if found useful, to use it during research.

ACKNOWLEDGMENT

The authors would like to acknowledge NWO for sponsoring this research through the Jacquard ScaleItUp project. Also many thanks to our industrial partners Adyen and Exact.

REFERENCES

[1] B. Benatallah and H. Motahari Nezhad, "Service oriented architecture: Overview and directions," in *Advances in Software Engineering*, ser. LNCS. Springer, 2008, vol. 5316, pp. 116–130.

[2] S. Benbernou, L. C. M. S. Hacid, R. Kazhamiakin, G. Kecskemeti, J.-L. Poizat, F. Silvestri, M. Uhlig, and B. Wetzstein, "State of the Art Report, Gap Analysis of Knowledge on Principles, Techniques and Methodologies for Monitoring and Adaptation of SBAs," 2008, deliverable # PO-JRA-1.2.1 of the S-Cube project.

[3] S. E. Sim, S. M. Easterbrook, and R. C. Holt, "Using benchmarking to advance research: A challenge to software engineering," in *Proc. Int'l Conf. on Software Engineering (ICSE)*. IEEE CS, 2003, pp. 74–83.

[4] T. Espinha, C. Chen, A. Zaidman, and H.-G. Gross, "Maintenance research in SOA – towards a standard case study," in *Proc. of the Conf. on Software Maintenance and Reengineering (CSMR)*. IEEE CS, 2012, pp. xxx–xxx.

[4]https://github.com/SERG-Delft/spicy-stonehenge