Lichen: Deploy, Demo and DevOps

Statement of Work

prepared by

Michael Richardson

mcr@xelerance.com

for

Xalgorithms Foundation

2016-03-19

version 1.0

Executive Summary

This document describes an effort to produce a demo of the Xalgorithms lichen prototype xa-elegans, interfaced to an open source accounting system by the end of May 2016. Work on the xa-elegans system to make it able to process UBL is not included in this estimate, as this work is not deeply understood as yet.

Introduction

The Xalgorithms Lichen project consists of 5 functional modules:

1. xa-calcicola: Android application, version/client of xa-elegans for smartphones.
2. xa-conspersa: prototype rails application containing an Experimental Registry and Repository, where Rules processing is performed in/by the Repository. (dead end project)
3. xa-elegans: A rails application, processes UBL to apply rules to it.
4. xa-registry: A prototype internet-wide rule registry.
5. xa-repository.simple: A client API for the xa-registry code base

The work on the various components has been progressing in an organic fashion: adding components to each piece in turn as it required to support additional functionality in other components. The focus has been on processing of rules, ignoring the problem of where the UBL is obtained from or how it is consumed.

Problem to be solved

1. An accessible web-based demo is needed

Something that can be shown to technical and business experts is needed. The demo needs to be deep enough to be meaningful, but breadth of experience is more important than having all the small details correct.

2. A representative data flow from a seller to a buyer is needed.

The Rule User comes in multiple types: buyers, sellers, as well as other roles which may be combinations of them. The richness of the rule system will become apparent once it is possible for rule owners to write new rules, and for buyers and sellers to seamlessly apply them. System level verification of each step in the processing will validate the design, and provide a reference use case.

Work proposed

The following work is proposed to be done by Michael Richardson.

DevOps for prototype

A critical part of any agile effort is having a customer: the customer decides what is important to do next, and actually uses the software system. Michael will act as the target customer, initially for the xa-elagens and xa-registry components. Michael will take the code and put it into prototyped production suitable for use in the effort described below.

This will have three effects:

1. the set of software dependancies and system requirements will be documented in the form of debian packages, docker images and READMEs. This will make it clear what is required to install the code from github.
2. operational requirements to make the components will work in the field: certificates, rule owners, signature authorities, domain names, and anchor URLs will be determined. This will make it much easier for another entity to operate the code installed from github.
3. An initial database of sample rules will get populated.

The effort will be on-going, with an initial burst of 2-4 days of effort during the initial 2 weeks, followed by an ongoing half-day effort every other week. The work will be delegated by Michael to other Xelerance personnel over time.

Use of xa-elegans inside ledgersmb

LedgerSMB[1] is a web-based, open source accounting and eCommerce solution written in Perl. It was forked from smb-ledger[2]. LedgerSMB has been in production for approximately 10 years, and sql-ledger for another 7 or 8 years before that.

LedgerSMB includes an ad-hoc, partially table driven sales tax calculation module in the files lib/LedgerSMB/Taxes/Simple.pm. The code as written can deal with moderately complicated things.

For instance, it can deal with the incremental tax calculation that used to be required in Quebec, where the TVQ should be applied after the GST is applied (and include the GST in the base cost). It can also deal, via annotations to SKUs as to whether to apply a tax at all. It can not deal with more complicated situations, such as charging a tax only to a local resident, changing the tax rate based upon who the buyer is.

As a buyer, dealing with incoming invoices, dealing with shipping charges is poor in ledgersmb: it can be very difficult to determine if taxes were added before or after the shipping charges. While shippers do charge tax, but some sellers have flat rates for shipping. Further, some shippers will then collect tax as part of the customs charge, and this can be very difficult to relate back to the original purchase.

Dealing with airline tickets, which can have 6 or 7 tax-like things applied to them can be very difficult:

1 http://www.ledgersmb.org/
2 http://ww.smb-ledger.org/

the HST may be applied to the base cost of the ticket, the “Fuel Surcharge” may already include the HST, and an airport improvement tax may be GST-exempt...[3]

The proposal for this step is to show up xa-elegans can be used as a module with LedgerSMB: outsourcing the tax calculation. To do this requires:

1. LedgerSMB has to format the trial invoice (or purchase order) as UBL.
2. It has to send the resulting UBL to a co-located xa-elegans instance.
3. The xa-elegans instance has to process the UBL, search (possibly, interactively with the user) for appropriate rules to apply.
4. The xa-elegans then has to insert appropriate annotations into the UBL for the invoice, indicating what calculations it has made, as well as indicating which rules were applied.
5. The resulting annotated UBL then has to be returned to the LedgerSMB application, which must parse the UBL back into it's internal database form, noting and recording the calculated values.

There are alternate flows: such as when receiving an invoice in UBL, the book keeper must either match it up to an existing purchase order, or create a fresh purchase order. The tax amounts listed in the provided UBL must be matched up against what is calculated by the system: the system needs to know which taxes can be claimed as input tax credits, and which can not. If there is a discrepancy it is important to note it, but ultimately the “amount owing” will usually need to be respected: it is often already paid as part of an expense.

3 http://www.lop.parl.gc.ca/content/lop/researchpublications/prb0572-e.htm and https://www.westjet.com/guest/en/travel/basics/fares/taxes-fees.shtml and http://restaurantcentral.ca/Salestaxonfoodandbeverages.aspx

Schedule

The DevOps work amounts to 4 days initial effort prior to April 15. Additional maintenance work would occur as required.

There are five major activities listed the LedgerSMB/UBL/xa-elegans stage. Some of the activities are related and need to be done together, other activities There are three independant efforts:

1. export/import of UBL from LedgerSMB
2. process UBL to create a list of rules
3. apply rules to UBL and output result

https://code.credil.org/projects/lichen/issues/gantt lists a possible project schedule. This schedule does not present actual dates as it does not take into account work loads, and that while some tasks could be done concurrently, resource levels will not support such a thing. It does show relationships and approximate durations for each activity which are believed to be within 20% accuracy.

Resources

There are three resources available: Michael Richardson at approximately 1 day/week, Don Kelly at approximately 1 day/week, and Patrick Naubert, unavailability unknown at the time of this writing.

Summary

A demo could be completed by the middle of May 2016.