Rule-based Cloud Service Localisation

Luke Collins[1] and Frank Fowley[1] and Claus Pahl[2]
[1]*School of Computing, Dublin City University, Dublin 9, Ireland*
[2]*CNGL, School of Computing, Dublin City University, Dublin 9, Ireland*
luke.collins4@mail.dcu.ie, frank.fowley@dcu.ie, claus.pahl@dcu.ie

Keywords: Cloud Service Localisation, Cloud Computing, Cloud Architecture.

Abstract: The fundamental purpose of cloud computing is the ability to quickly provide software and hardware resources to global users. The main aim of cloud service localisation is to provide a method for facilitating the internationalisation and localisation of cloud services by allowing them to be adapted to different locales. We address lingual localisation by providing a service translation using the latest web-services technology to adapt services to different languages and currency conversion by using realtime data provided by the European Central Bank. Units and Regulatory Localisations are performed by a conversion mapping, which we have generated for a subset of locales. The aim is to provide a standardised view on the localisation of services by using runtime and middleware services to deploy a localisation implementation.

1 INTRODUCTION

Distributed web services can provide business and private consumers with computing abilities which may not be feasible for them to develop in-house (Armbrust et al., 2010; Voorsluys et al., 2011). However, cloud computing introduces new issues, for example in Europe, where there is a wide range of languages spoken, services are very often only developed for single language. Often, companies do not have the resources to develop multilingual products. Localisation encapsulates a large number of issues which need to be addressed. These include Language Translation, Regulatory Compliance, Currency Conversion or Units Conversion. Localisation is typically performed on textual content (i.e. strings) and refers to either languages only or physical location. However, the purpose of this work is to develop a method of localising services by implementing a 'mediator' type service which interacts between the APIs of the cloud service provider and the requester. We are going to focus on a number of locale dimensions such as language, taxation, currency and units. We aim to provide service-level language translation techniques to localise services (including API interfaces) to different languages. Regulatory translation which includes currency, units and taxation among other legal governance and compliance rules will be provided by standards-based mappings. Regulatory translation is important for applications to comply with varying regional laws and regulations.

A specific need is to make localisation techniques available at runtime for dynamic localisation, which is required for instance for currencies and other variable aspects. Thus, Cloud Service Localisation (CSL) provides a mechanism for converting and adapting services to the locale of the requester. A Localisation Provider act as an intermediary between the service provider and the requester. In our proposed platform, this is supported by a mediation service. By generating a common platform solution for these localisation issues, we allow the ability to dynamically localise cloud services to be made with little effort. The key contributions of our work are:

- Software Localisation at Service Level - the main concern is a standardised mapping within a potentially heterogeneous environment.
- Adaptation and Integration - the main concern is the maintenance of service quality after it has been localised through adaptation.

2 SERVICE LOCALISATION

Our focus is a platform for cloud service localisation, which makes a shift from a "one size fits all" scenario towards an end-to-end personalised service scenario. Currently, cloud computing services suffer from localisation and adaptability issues for multiple

users in different regions. These issues could be overcome if a multi-lingual and multi-regional solution was developed (Pahl, 2012; Wang et al., 2010).

A scenario which illustrates the benefits of localisation could be a cloud service provider that manages company accounts for its customers, with offices in different global locations.

- Regulatory: Conversion of data between standards and their variants, e.g. based on different units of measurement $Metric \rightarrow Imperial$.
- Currency: Conversion of between currencies, e.g. $Euro \rightarrow Dollar$.
- Lingual: Translation of service related data between languages. This could include free text, but also specific vocabularies based on business product and process standards such as GS1 or EANCOM for documents.
- Taxation: Different customers have different taxation requirements, e.g. different VAT rates. Localisation of the accounts software should take this into account for each locale.

3 PLATFORM ARCHITECTURE

Localisation of cloud services requires an architecture framework (Pahl et al., 2007) to be implemented to facilitate various localisation methods. These various methods, implemented as services in our proposed localisation platform, are used to facilitate the localisation of localisable elements or artefacts. This paper focuses on the dynamic localisation of service level descriptions.

With every cloud service there are various elements which may be localised: Service specifications/descriptions (APIs), Models (structural/behavioural), Documentation (for human consumption), and Messages exchanged between services. Services are normally written to be independent of locales, however localisation is often needed. A localisation platform should be based on attributes which vary from locale to locale, like time or date format.

A service localisation platform requires a number of elements. These elements can be pre-translated fragments in static form or can be dynamic translation systems. Figure 1 aims to demonstrate the concept of a policy and mappings based system, which can be scaled when additional processes are attached to the mediation process. In the platform architecture, user-specific locale policies are applied to service endpoints. For example, in a WSDL file we may localise messages and operation names. Rules for each type of translation would be stored in a rules

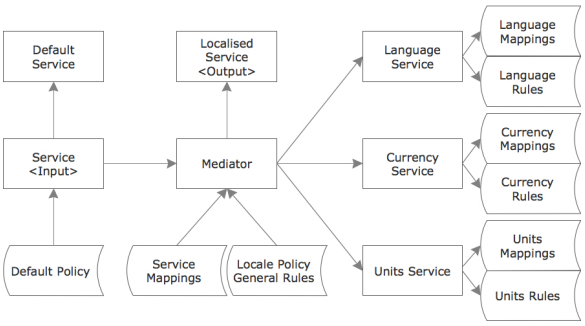


Figure 1: Architecture of the Localisation Platform.

database (General Rules Repository). Similarly, mappings between common translations would be stored in a mappings database (Translation Memory).

A mediator operates between users (with different locales) and several service providers (with different locales) by providing core localisation services, such as currency conversion and language translation. The architecture supports the following:

- Static Mappings: these could be the mapping of one language to another or one unit to another, pre-translated in translation memories.
- Dynamic Localisation: when translation mappings are not stored, then a dynamic localisation is required in order to obtain the correct translation and store the mapping.
- Policy Configuration: in order to configure the various locale policies, we must generate particular translation rules, supported by a logical reasoning component.
- Negotiation: this is the exchange of locale policies through the form of XML and SOAP from a web services point of view.
- Localisation of Services: the mappings between the remote service and the localised service description must be stored in a mappings database (Translation Memory) so the localised service has a direct relationship with the remote service.

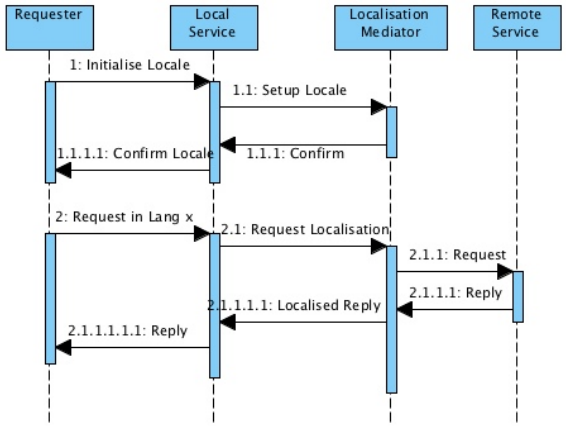
A sample locale setup is

```
<CSLContext>
  <Locales>
    <RequesterLocale>
      <LanguageCode>efr</LanguageCode>
      <CountryCode>FR</CountryCode>
      <CurrencyCode>EUR</CurrencyCode>
      <UnitCode>M</UnitCode>
    </RequesterLocale>
    <ProviderLocale>
      <LanguageCode>en</LanguageCode>
      <CountryCode>IE</CountryCode>
      <CurrencyCode>EUR</CurrencyCode>
      <UnitCode>M</UnitCode>
```

```
    </ProviderLocale>
  </Locales>
</CSLContext>
```

A mismatch between provided and required locales needs to be bridged by a mediator service.

In order to provide the localisation platform services, we need to implement a number of classes to enable a modular service localisation platform. Their interaction is summarised below.



4 LOCALISATION RULES

At the core of our service localisation platform is a language to specify the rules in relation to localisations. In most cases, languages like WSDL and other XML languages provide information regarding the services that are provided via an API. However, in order to encapsulate localisation information, there is a necessity to provide a language which will contain details in relation to the locales of the requester and the provider. For the purpose of our localisation platform, we use a policy language based on the Semantic Web Rule Language SWRL, which is based on the propositional calculus.

A localisation layer is used to encapsulate the various forms of translations. It is used to describe the relationships between localisable elements. For example, it contains the details of items which can be translated. For our localisation model these are: Documentation & Descriptions and API Messages & Operations. The rule language is used to define policies of two types: firstly, locale definitions and, secondly, conversion (translation) rules. We motivate the rule set through selected examples. There are a number of locale definition rules provided, like *Loc* or *hasCur*, by which locales for specific regions are described. A locale can also be described by other rules such as *hasTax*, *hasLang* and *hasUnit*:

$$IELoc(?l) \leftarrow Loc(?l) \wedge hasLang(?l,?z) \wedge hasCur(?l,?c) \wedge hasUnit(?l,?u) \wedge ?z = en \wedge ?c = EUR \wedge ?u = metric$$

The benefit of a formal framework for the rules is that other rules can be inferred by from partial information. For example, if we knew that a locale had USD as its currency we may be able to infer its country from it: $?c = USD \rightarrow ?l = USLocale$. These inferred rules do not apply in general - this may not work if we know a currency is Euro in which case it could be one of many locales in Europe. The purpose of these rules could be to determine inconsistencies, however. Preconditions can clarify the remit of these rules. A conversion between locales, e.g. $Locale\,A \rightarrow Locale\,B$, is given by the following general conversion rule:

$$IELoc2USLoc(?l1,?l2) \leftarrow hasLang(?l1,?z1) \wedge hasLang(?l2,?z2) \wedge hasCur(?l1,?c1) \wedge hasCur(?l2,?c2) \wedge hasUnit(?l1,?u1) \wedge hasUnit(?l2,?u2) \wedge ?z2 = convertLang(en,en,?z1) \wedge ?c2 = convertCur(EUR,USD,?c1) \wedge ?u2 = convertCur(metric,imperial,?u1)$$

5 EVALUATION

Scalability becomes more important from a cloud computing perspective when a service may have large multiples of users. Scalability has not been empirically addressed for this phase of research and will be evaluated in later prototypes. Many aspects of the platform would require modification to effectively allow the infrastructure to vertically scale-up or scale-out efficiently. However, horizontal scalability - i.e. the addition of more localisation concerns - is conceptually easily supported by the modular mediator architecture, which we will address further below in the extensibility context from an implementation view.

Performance is one of the CSFs of this platform. Poor performance often tends to affect software exponentially as multiples of users consume a cloud service at the same time. As the application scales up to multiple concurrent users, the latency would increase due to extra loads placed on the platforms services. This makes latency one of the KPIs of the project. Latency, is also an area to be assessed as adding the localisation platform to the workflow of an existing process has the potential to add lag-time. This lag-time exists due to time required to compute and also the time to initialise the various variables. As a strategy, we have aimed to improve performance by using pre-translated aspects (through stored mappings).

6 RELATED WORK

Our platform addresses the need for dynamic localisation of various artefacts by use of a translation memory and a set of logical rules.

- Software Localisation: this usually refers to human consumption of data which are produced by the software - namely messages and dialogues. Our focus is on the localisation of the service level. Service internationalisation is supported by the W3C Service Internationalisation activity (W3C, 2005; Phillips, 2005).
- Adaptation and Integration: it is not uncommon for software adaptation at service level, but the adaptation of services based on locales and using a translation memory with rules and mappings is new (Truong and Dustdar, 2009). The problem of multi-tenancy is a widespread issue in the area of cloud computing (Wang et al., 2009; Wang et al., 2010). This is an area where a lot of research is being invested in order to provide a platform for different users with different business needs to be kept separate and their data to be kept private.
- Semantics: involves the matching of services with various locales using mappings and rule-based system (Bandara et al., 2009; Anastasiou, 2011; Fujii and Suda, 2009).

IBM (IBM, 2010) presents a localisation solution for its WebSphere platform. It uses static localisation as it requires the WSDL files to be generated prior to deployment. This differs from our proposed localisation platform as our solution aims to perform transformations between locales dynamically.

7 CONCLUSIONS

A Cloud Service Localisation (CSL) implementation should allow for a seamless and transparent solution by automatically adjusting and adapting services to the requesters' own locales. We have presented a modular CSL implementation which can enable cloud services to be introduced into emerging markets which have localisation issues. Localisation hence provides a mechanism to widen a service provider's target market by enabling multi-locale solutions. The easiest solution is for a cloud service provider to provide a 'mediator' service which could act as middleware between a requester and the service provider. By allowing services to be localised, we are enabling the provision of multi-locale cloud services to create interoperable clouds. The localisation of services also allows the geographic scope for service providers to broaden and users to have the ability to combine services from different providers from different regions. These clouds can be heterogeneous in nature. Thus, the platform described in this paper must support that ability. Due to the nature of third-party services, it is more intuitive for service localisation to be performed dynamically through the use of a mediator service.

REFERENCES

Anastasiou, D. (2011). The impact of localisation on semantic web standards. *European Journal of ePractice*, 12:42–52.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Bandara, K., Wang, M., and Pahl, C. (2009). Dynamic integration of context model constraints in web service processes. *International Software Engineering Conference SE'2009*. IASTED.

Fingar, P. (2009). Cloud computing and the promise of on-demand business inovation. *InformationWeek*, July 13,2009.

Fujii, K. and Suda, T. (2009). Semantics-based context-aware dynamic service composition. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(2):12.

IBM (2010). IBM WebSphere Developer Technical Journal: Developing internationalized Web services with WebSphere Business Integration Server Foundation V5.1.

Pahl, C., Giesecke, S. and Hasselbring, W. (2007). An Ontology-based Approach for Modelling Architectural Styles. *European Conference on Software Architecture ECSA2007*. Springer.

Pahl, C. (2012). Cloud Service Localisation. *European Conference on Service-Oriented and Cloud Computing ESOCC 2012*. Springer.

Phillips, A. (2005). *Web Services and Internationalization*. Whitepaper.

Truong, H. and Dustdar, S. (2009). A survey on context-aware web service systems. *International Journal of Web Information Systems*, 5(1):5–31.

Voorsluys, W., Broberg, J., and Buyya, R. (2011). *Cloud Computing: Principles and Paradigms*. John Wiley and Sons.

W3C (2005). *Web Services Internationalization Usage Scenarios*.

Wang, M.X., Bandara, K.Y. and Pahl, C. (2009). Integrated constraint violation handling for dynamic service composition. *IEEE Intl Conf on Services Computing*. pp. 168-175.

Wang, M.X., Bandara, K.Y. and Pahl, C. (2010). Process as a service distributed multi-tenant policy-based process runtime governance. *International Conference on Services Computing (SCC)*, pages 578–585. IEEE.