An Aspect Oriented Approach for Renarrating Web Content

Gollapudi VRJ Sai Prasad
SERC
IIIT-Hyderabad
Telangana, India
saigollapudi1@gmail.com

Sridhar Chimalakonda
SERC
IIIT-Hyderabad
Telangana, India
sridhar.ch@research.iiit.ac.in

Venaktesh Choppella
SERC
IIIT-Hyderabad
Telangana, India
venkatesh.choppella@iiit.ac.in

Y. Raghu Reddy
SERC
IIIT-Hyderabad
Telangana, India
raghu.reddy@iiit.ac.in

ABSTRACT

The ability to modify the existing published web pages is what we are calling *Renarration* of the web. Such a mechanism is useful for improving accessibility and personalization of the content currently on the web. There are many techniques in place for enabling both Web Accessibility and Web Personalization. In this paper we propose a novel approach: an Aspects inspired design of renarration. Aspects have traditionally been applied to programming. Here we reinterpret concepts like Join Points, Point-cuts and Advices and apply them to web documents. To validate our approach, we designed a framework called *Rennaration Studio* that is built using microservices based architecture pattern and implemented using Python's flask platform. We demonstrate the feasibility of our proposal by renarrating different Aspects (text, language, phonetics) of two specific web documents.

CCS Concepts

•**Software and its engineering** → **Abstraction, modeling and modularity;** *Organizing principles for web applications;* •**Human-centered computing** → Accessibility theory, concepts and paradigms;

Keywords

Aspects for documents, Microservices for Annotation, Renarration, Structured Web Documents, Annotation

1. INTRODUCTION

We are interested in applying the notion of *Aspects* [15] from Software Engineering to the problem of Renarration of some web content. Renarration is the label being used for any alternate articulation of an original web source. For instance, a translated text, an annotated image or a audio replacement of some original web source can all be considered as renarrations. Renarrations could involve modifications to either the original content, flow or script. These modifications could either reinforce the original material's context, content or the intent, or, they may significantly modify it. Two main motivators for renarrating some existing, published web content are Web Accessibility and Web Personalization.

[8, 23, 7] have already established renarration and the need motivating it. In this paper, our interest lies in enabling renarration, but by using an Aspects oriented approach. Of course, there are already existing solutions that implement renarration in particular [8], and Web Accessibility solutions and Web Personalizations in general. However, these solutions either incorporate a goal-driven tightly-coupled design[8], or a client server architecture or a browser based offering. [9] discusses how accessibility can be provided at one of many control points in the workflow of a web-content presentation. For instance, Fairweather et. al, highlight that the solution can either be an external extension to the web, like a text-to-speech adapter hardware; or at the presentation level, as in with Style Sheets [16]; or at an augmented browsing experience, like providing a mixed modality solution as in VoiceXML[17]; or at a protocol level solution as in AIAP[1]. In addition to these choices, there are also several subsequent solutions [29] that have leveraged such options as a proxy based design [27], a server based design [12], browser add-ons [10] etc. However, to the best of our knowledge software engineering ideas and in particular Aspects have not been applied in the context of web content renarration, which is the core contribution of this paper. More specifically, this paper proposes the idea of using aspects as a way to facilitate transformation of web documents for renarrating web based on semantic style sheets.

Aspects have primarily emerged to address the challenges of *cross-cutting concerns* and to design code in a modular way based on separation of concerns principle. Initially, in languages like AspectJ, new programming constructs were added to represent Aspects [14]. Thereafter, several ap-

ISEC '17, February 05-07, 2017, Jaipur, India

DOI: http://dx.doi.org/10.1145/3021460.3021466

[1]Alternative Internet Access Protocol; specified by the then National Committee for Information Technology Standards (NCITS)

proaches have been proposed based on Aspects [26]. An analysis of a recent survey on Aspect oriented analysis and design [5] reveals that the primary focus of Aspects is at a programming level with several extensions to support Aspects in programming languages. Researchers have focused on early Aspects at requirements and architecture level [24] [35] as well. Existing work on Aspects has essentially focused on addressing cross-cutting concerns at code level whereas in this paper we propose the use of Aspects for renarration of web documents.

2. ASPECTS AND WEB DOCUMENTS

Aspects, Aspect Oriented Software Development (AOSD) and Aspect Oriented Programming (AOP) [15, 21] have had their hay-day in late 1990s to mid 2000s. However, of late, they have not been receiving much attention (at least in scholarly publications). Aspects initially entered the scene promoting modularity and encapsulation in the incorporation of core business logic which co-existed along with other orthogonal requirements. The idea was that *joins* would be made for *Advices* by some *weaver*, that would tangle a *cross-cutting concern* with the primary functionality of some existing code. Advices could be separated out, joins and point-cuts would be spread-out, and cross-cutting concerns would be woven-in with the other primary concerns of the application.

What is interesting to us is that the notion Aspects can be extended beyond the conventional programming paradigm and could also potentially be applied to our proposed world of renarrated web documents. Table 1 provides a brief comparison of conventional uses of Aspects with our proposed novel use of it.

Typically AOP is applied to software programs. In our case, we are applying it to HTML web documents. Typically a software developer is involved and she would code the Aspects into the source-code and run the compiler twice. In our case, an existing web document is beyond the editing phase. As it is already published, it is more of annotation than editing that has to be considered. For annotation, the original author is not involved. Instead, a third-party volunteer annotator – whom we are calling a renarrator – will integrate the Aspects after publishing. Typically AOP takes a base code and either adds, deletes, or overwrites new code on it. The byte code of the base is still available. And, the additional code is cross-compiled against the same language as the base code. For renarration, we also allow for the base document and the changes to coexist. Similarly, the annotations are also only doing add, delete, overwrite operations. And, when it comes to cross-compiling, we are not bound by the modality of the base document. Instead we can potentially overlay new mode on existing base mode. For typical AOP there exist notations, languages, related compilers and even dedicated tools. ActiveJ is a popular Java based example of this. In the case of renarration, however, there is no such existing tools or technologies. For this reason, in our proposed usage of Aspects we have limited ourselves to the ideas of Aspects and not to its field-implementation in code.

In applying the concepts of Aspects to our renarration of the web, we find that we had to embrace a more broader definition of the original AOP terminology. Table 2 highlights the various novel interpretations we are using for our work.

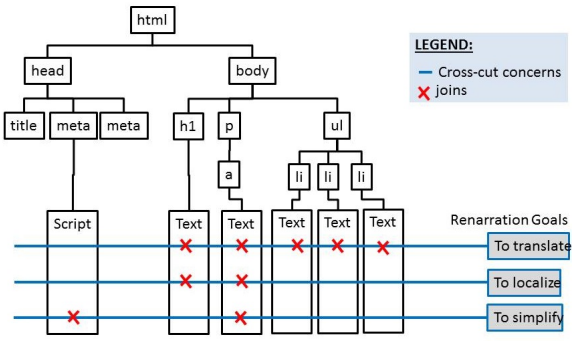

Figure 1: Cross-cutting concerns of renarration across HTML DOM objects.

2.1 Cross-Cutting Concerns

The notion of cross-cutting concerns is fundamental to Aspects. Infact, it is the orthogonality in the focus and need of the requirements that compells the advent of Aspects. In the case of renarration, a similar compelling need exists. It is infact, the renarration in itself that is cross-cutting across document. See Figure #1.

To us renarration is a modification to an existing page; it augments an HTML document; it annotates it; it transforms it. But, renarration has a purpose. It is trying to do this to either translate the original information into a different language, or to re-explain a complex concept in a better way; or to replace a less relevant reference with a more relevant example. For instance, an English text can be converted to Hindi[2]. Here the renarration of language change cross-cuts all objects of the Document Object Model (DOM) model within a HTML document.

Localization could be a goal of renarration. In such a case, wherever there are some non-localized terms, they would need to be amended or changed to reflect the local terminology. Perhaps currency values need to be changed from USD to INR, or units of measure need to be changed from Pounds to KGs. etc. In all these cases the text that needs to be localized could be spread all through the original document. So, a change for localization, would effectively mean a cross-cutting concern.

In yet another example, let us assume that the goal of renarrating some document is to detail it with more examples, explanations, pictures etc. Perhaps a scholarly document is being renarrated for consumption by a high school student. In such a case, the explanations and modifications will also cross-cut across multiple DOM elements. Many paragraphs (i.e. <p> nodes) may be impacted throughout the body of the document. Also, some new nodes with their own tags – like that for an image – may need to be included. Whatever the case, cross-cutting happens across multiple DOM elements.

In a more abstract interpretation, renarration can also be seen as a modification of the original document's context or intent. In other words, a renarrator is re-packaging an existing narrative for a newer community of users by re-contextualizing it, or by modifying the original author's intent. Even with such a abstracted interpretation, we see that renarration cross-cuts the DOM.

[2]one of the more popular Indian languages

Table 1: Comparison of Aspects

SNo.	Conventional Use Of Aspects	Our Novel Application of Aspects
1	Typically applied to software programs	We are applying it to HTML documents
2	Typically Adds, Deletes, or Overwrites base code	We use it for annotation of existing content
3	Typically applied by the software developer or the maintenance engineer who has access to the code source	In the case of the renarration of a document, the original author is not leveraging it. Instead it is utilized by some 3rd party volunteer user (who need not be the author)
4	Developed and embedded into original source code *before* compile time	Applied by 3rd party volunteer to document *after* publishing
5	Cross-compiler needs support in programming languages (eg. ActiveJ for Java; ActiveC for C++)	No technology has implemented Aspects for either documents in general or web content in particular

Therefore, because the applications of renarrations would cross-cut and scatter changes across the original content of the HTML document, and because sound software engineering practices would compel us to maintain modularity and separate out the renarration concern from the original intent of the HTML source, we feel the necessity to introduce Aspects for renarrating the web.

2.1.1 Persistence

One major requirement for sustained renarration is that the changes that a renarrator makes must persist beyond the session for subsequent use by another arbitrary user. To enable this, we expect the Join Point Model related metadata to persist outside the scope of the existing HTML document.

Therefore, for renarration of web content, Persistence of changes is yet another cross-cutting concern. In existing Aspects oriented applications, Persistence has already been addressed and popularized. This, to us, is yet another motivating reason to consider using Aspects for renarration of the web.

2.2 Join Point Model

An implementation associated with a cross-cutting concern is anchored to the original source code through the Join Point Model concept. A Joint Point Model essentially consists of three related elements: 1) the Join Point, 2) the Point-cut, and 3) the Advice.

2.2.1 Join Points

The Join Point is a location in the execution flow where the implementation for the cross-cut concern could potentially be interjected. This is typically done at a call to a method, or when accessing a member of an object, during assignment etc.

In the case of renarration, a document does not have assignments or method calls. Therefore, the Join Points can be on any part of the HTML document. From a DOM tree point of view, the Join Point can be at the node level or within the node as well. From a content point of view, a renarration related Join Point can be at a tag level (e.g. <p> or <li>) or within the tag content. Therefore, for documents, it is the tag that is considered as the Join Point.

For our pilot implementation we have only worked with Text modality. Using identifiers and class IDs, text oriented HTML tags can be used to create Join Points. For renarration of the web, we go one step further. We observe that the original HTML DOM structure of the source (with its generic document labels) may not be a sufficient semantical representation of the content. A renarrator, on the other hand, may have a better understanding of the human semantics she sees in the document. In addition, she may also have a need to semantically structure the source to enable further transformation. It is for this purpose that a renarrator may impose a new semantic structure on the source by way introducing a new Schema or a user-defined Ontology. For structuring the source HTML, a new set of tags can be deduced from such user-defined meta structures.

From an Aspects point of view, these new tags may also be used as Join Points.

Incorporation of new tags within an HTML document can be made possible by the usage of RDFa web technologies[3]. Schemas too need not be invented by the user. Some common Schemas that are already on the web[4], can also be reused.

2.2.2 Point-cuts

Point-cuts link the Join Points to Advice. The Join Points create the trigger. The Advice creates the response to address the cross-cutting concern. The Point-cuts could rely on one or more Join Points to trigger an Advice. The specification of a Point-cut is done manually by the Software Developer.

In the case of renarration, we have renarration rules that link tags (Join Points) to interpretable script code. For instance, if a portion of code is to be language translated, then the code for translation is earmarked by tags. These tag labels – which are essentially Join Points – could be linked to a Point-cut specification which connects to translators (ie. Advice) to modify that earmarked content.

In the case of localization, a currency change can be enabled by, one, tagging the currency content in the HTML page. This is done by a renarrator. The tags would use a user-defined Schema to identify the numeric value to be transformed. A renarration rule will link some tag labels for it to be triggered. That is, when the earmarked tags are encountered, the rule is triggered. The rule then invokes the appropriate Advice for doing the currency conversion. The Advice can be yet another service that is outside the current web application. And, an existing Advice can rely on other services for its functionality.

2.2.3 Advice

An Advice is implementation logic of an Aspect. It is typically presented in a modularized and reusable way. The

[3]http://rdfa.info/

[4]schema.org

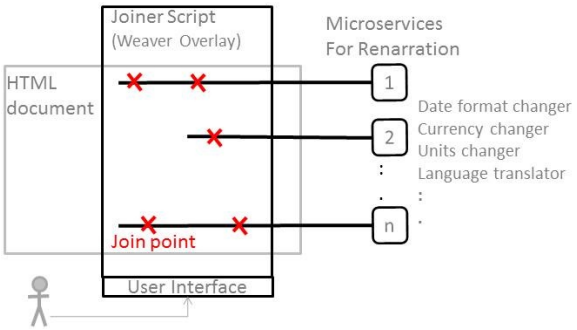

Figure 2: Abstract view of our Aspect based implementation.

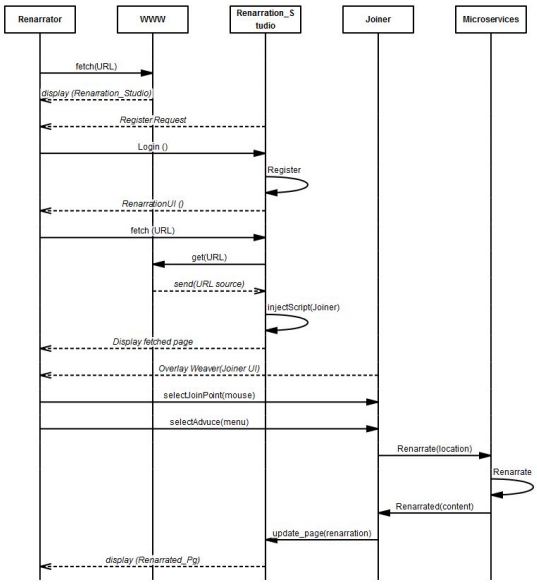

Figure 3: Renarrator's sequence flow in Renarration Studio

Join Point Model enables the linking of one Advice code to multiple Join Points.

In the case of renarration, the Advice is the place where we put the transformation logic. It is here that changes to the source are computed. For example, if we are renarrating currencies or measurement units, then it is in Advice that such calculations are made.

It is also not necessary that we restrict ourselves to just the code in the Advice. In our case for currency changes we may rely on an external service to get the latest in Forex figures. Similarly, for unit measurement transformations, we can connect with some external service to do unit transformations.

3. RELEVANT WORKS

Aspect Oriented Programming (AOP) came in on the heels of the rush around Object Oriented Programming. In early 2000s MIT predicted that AOP would be the programming paradigm for the next 10 years. A quick scan of the literature, however, indicates that the uptake was not as predicted.

"Despite the fact that aspect-oriented programming was introduced in the second half of the 1990's, very little experimental and/or quantitative work can be found in the literature [3]." Munoz et. al. concur with this and suggest that "8 years after the MIT announcement AOP is still not widely spread [18]."

As per our analysis, current literature on AOP can be segmented into three categories: Category one literature tends to explain the theory and defend the logic behind AOP [21, 15]. Category two literature tends to be about usage and implementation of AOP. Here we find literature around how AOP was used for Requirements [35, 24, 2, 34], for Security [33], for web service selection [32], for web based learning systems [13] etc. In category two we also find information on tools and techniques [14, 26, 5].

And, lastly, category three encompasses studies checking the credibility and robustness of AOP. They address questions like: Does AOP increase the speed of development [11]; Why was it not popular [18]; Does it deliver on maintainability [3] and understandability [31] etc.

To our knowledge AOP was not significantly discussed in any contexts other than programming. Neither could we find much discussion around applying the cross cutting concerns to the fields of web or documentation. From a Web Accessibility point of view, AOP need not be the only methodology to use to deliver renarration. We could have explored Object Oriented Programming, Event Driven models or Service based models. While we are cautioned by the studies and the lack of uptake of AOP, we feel that AOP as a concept is still worth exploring because of the cross-cutting requirements arising from renarration. And, it is here that we position our novel proposal of applying AOP for web content renarration as a use-case.

4. ARCHITECTURE FOR RENARRATION

Our Aspect based approach is implemented as a web application called *Renarration Studio*.

Renarration Studio is essentially a web portal which enables two things: One, it enables an arbitrary third party user – i.e. a renarrator – to renarrate an arbitrary web document. And two, it allows a typical user to logon and view any existing renarrations for a given web page. See Figure #3.

4.1 Renarrating a web page - Use Case

To renarrate a web page, a renarrator must first get to the Renarrator Studio portal. Here she must register and inform which web page she would like to renarrate. The concept behind registration is that it enables the Renarrator Studio to collect user information. This is subsequently stored (i.e. persisted) along with other annotation related parameters as useful metadata for a renarrated page.

When fetching the source page for annotation, the Renarrator Studio injects a JavaScript into source header portion of the HTML. The idea behind injecting script is that the fetched page should have the same look-and-feel as the original source, but with one exception - it should now have an overlay of Weaver like functionality with a User Interface (UI). In our case this Weaver functionality is called *Joiner*. The Joiner overlays a UI on the HTML document and places it at the bottom. We call this UI *Annolet*. See Figure # 4. Annolet UI provides the renarrator with options for doing various types of renarration.

Using Annolet UI, the renarrator tries to position the

Table 2: Application of Aspects terminology to renarration

SNo.	Principle	Original Intention	Our Application for Renarration
1	Aspect	a concern that is also part of the application but it is having an impact on the primary biz logic in such a way that this concern is impacting multiple functionalities in multiple places. This concern, cross-cuts the primary biz logic. It is scattered.	Renarration
2	Concern	a grouped functionality; could be a requiement, feature	for a document its presentation could be its primary concern; Renarration could be its cross-cutting concern
3	biz logic concern	primary functionality of a application	In the case of a document, the presentation layer is assumed to be the primary function of the document. It wants to be represented in a particular way.
4	Cross-cutting or horizontal concerns	There could be features that are orthogonal to the original intention of the program. These requirements may impact the implementation at multiple places.	Annotation component of Renarration is a cross-cutting concern that is not linked to the core concern of the document (which is its current presentation). Persistence is yet another cross-cutting concern.
5	Weaver or composition	intention is to join or tangle Advice at various points within spread code	A script overlay (which we call as Joiner) on top of a source HTML document
6	Join-point	A location where an implementation of an Advice can be connected	This is either implemented as a user-designed tag or a user-initiated mouse click on document.
7	Point-cut	this is where the Advice needs to be applied (the portion defined by an entry and exit point of the Point-cut)	For us the tags mark the Point-cut area. It is the insertion point.
8	Advice	without modifying existing code, adding new behavior	adding tags, structuring document. This is the additional code that we wish to apply. Metadata can exist here
9	cross-compiling of Aspects on base code	original code is taken as base, Aspects are cross compiled ontop in the same language	no language, or tool support exists. We are not limited by language of base implementation

Figure 4: Annolet UI overlayed on top of an existing HTML page by Joiner. This happens by JavaScript injection through Selinium in Renarration Studio. Image of ISEC Conference page reduced for diagram brevity.

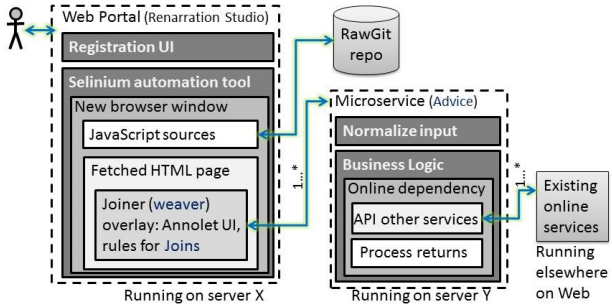

Figure 5: Architecture of Renarration Studio.

mouse on the exact portion of the web page that she would like to renarrate. Clicking on the earmarked components, the renarrator informs the system that she is ready for annotation. She has choices of services provided by the Annolet UI. Selecting one of the presented services, the renarrator makes the change (by following the UI commands). In the backend, this type of interaction with Annolet UI enables the Joiner script to grab the metadata needed by the system for subsequent use. This data (along with user registration data) is subsequently persisted for later use by other viewers.

Some of the renarration related changes (like editing, translating, phonetic-replacement to English words etc) are immediately applied to the selected text. There could be other renarration services which may be implemented at a later time.

The changes made by the renarrator are temporarily stored in the Joiner script. Later, when the renarrator presses the Submit button, the entire metadata information is pushed to a cloud based database for storage.

5. DESIGN AND IMPLEMENTATION

Renarration Studio is currently implemented as a web portal with browser client and web server model in Python using Flask Virtual Environment mechanism. See Figure 5 and 6. The Registration UI for the Renarration Studio is a simple Text Box prompting the user to enter a URL. When a renarrator enters a URL, the information is directed to a Selinium Web Automation tool. Selinium studies the browser characteristics and opens a new window. For now we are supporting Google Chrome browser only. The Selenium tool [28], which has largely been used in web application testing, is being used in our studio to fuse the Weaver based Joiner code onto the requested URL. That is, Selinium is used as a means to inject our weaver oriented JavaScript code onto the header of the fetched HTML page. The required script for injection is stored on Git[5] and fetched from the cloud for insertion. A corresponding CSS file is also linked to ensure that the UI for renarration is discreetly overlaid at the bottom of the fetched source HTML page. An alternative technology like GreaseMonkey [22] may also be used to inject a script onto an existing, published HTML page.

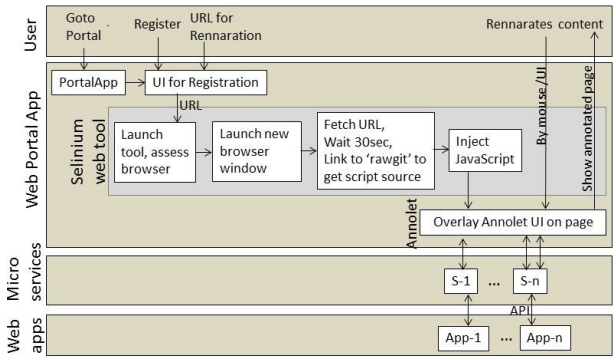

Figure 6: Task flow in Renarration Studio.

As we mentioned, our implementation of Joiner includes weaver like functionality. It essentially can be thought of as enabling a set of rules on 'how-to' and 'where-to' tangle renarration code onto the source document. The Joiner functions like a composer with a UI. It enables renarrators to Join the Cross-cutting Advices onto the HTML page. The UI in the Joiner is developed as a bookmarklet [4] and is called Annolet.

Renarration code is contained in Advices. Advices are implemented in Renarration Studio as Microservices.

5.0.1 *Motivation behind Microservices*

At a general level, Microservices are a popular architecture pattern [19, 30] in Software Engineering. We are opting to use the notion of this pattern in our design of our Renarration Studio. The reason for this choice is that Advices in AOP are modular and can be independently developed. They can always evolve and grow in quantity. They can be developed in a language other than the base code. We use Microservices because they facilitate such loosely coupled development.

In addition, the logic supported by a specific Microservice (or the logic represented by a specific Advice) does not necessarily have to match the business logic of the rest of the application. Cross-cutting concerns of logging, security, persistence are known for this orthogonality. In our case, annotation or renarration services have no strong relationship to the existing HTML document. So, by using Microservices we are essentially allowing for a good separation of cross-cutting concerns which may later be developed by different or independent software developers.

An additional reason for using Microservices is that some services (or Advices) maybe more complex and may require additional processing. The logic in one need not match another Microservice. For instance, in a case of renarrations

[5]'//rawgit.com/SSS-Studio-development/annoletjs/tagging/annolet_main.js'

involving currency or language or measurement-unit changes etc, they may need online resources to do the conversion. This dependency on online resources is usually different for each service. And, such dependencies may also lead to a delay in processing. In addition, services needing access to current environmental variables (e.g. browser type, current time, current location etc.) may also compel us to modularize and separate the concerns. This is more naturally implementable in a Microservice type architecture pattern.

In current implementation, the response time taken to renarrate a page is real-time in that the delay is not human noticeable and apparently immediate. Our implemented pilot has not yet been deployed on the web. However, we continue to develop it further and have shared the code on Github[6]. Renarration Studio can be locally cloned from our Git repository and deployed by interested users.

5.0.2 *Microservice Vs Webservices*

Despite not finding a clearcut differentiating definition between a Webservice and a Microservice, in our implementation, we opted to use the notion of Microservices [20] instead of REST based Webservices [1, 6] to represent our renarration Advices.

Here are some reasons for us choosing the notion of Microservices.

Microservices tend to be singular and independent functionalities. Whereas, a Webservice can technically be a enterprise app with an API. A Microservice can be a complete functionality. Whereas a Webservice can have dependencies with other apps like databases etc. A Microservice can be run in different sequences and over varying communication channels. Whereas webservices tend restrict themselves to SOAP[6] or REST [25]. They may also restrict themselves to XML(for SOAP) or JSON (for REST). Microservices do not have such restrictions.

In our assessment, we see that Advices tend to be more self contained, more individualistic, more independent and also more scalable. Developing Advices as Microservices thus allows us to develop, evolve and deploy individual services separately from each other. Moreover, Microservices allows us deploy our functionality in a staggered manner without impacting prior releases.

5.0.3 *Concepts not implemented by Renarration Studio*

As we have implemented the Renarrator Studio as a proof of concept and not as a production system, the current functionality is limited. Current work does not implement few of the features that were discussed in earlier portion of this paper. The unimplemented components were either seen as non-critical to our thesis, or were roadmapped to a future release effort.

In the current implementation, we have only developed the first part of the studio, which deals with a renarrate being able to annotate an existing HTML document. See Figure #3. The second part dealing with a subsequent new user being able to access a prior renarration, is not yet implemented.

Persistence maybe implemented using SWeeTs based technology. This has been introduced in [23]. In the current version of Renarration Studio, we have not integrated our work with a Sweet Store to enable persistence.

[6]https://github.com/vlead/SemanticTransformation

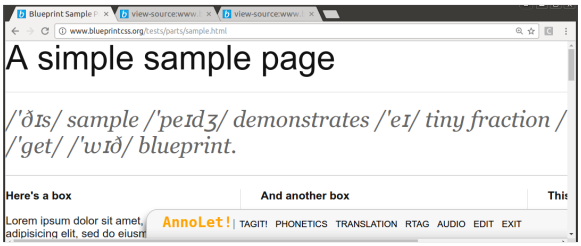

Figure 7: Sample phonetics functionality.

6. ADVICE AS MICROSERVICES

To validate our proposal that AOP can be used to renarrate, we have developed renarration services like 'Phonetics', 'Translations', 'Audio' etc as Advices. These Advices are in turn, implemented as Microservices. Our implementation of these renarration concerns are only representative and not of production quality. More robust development of these Advices is left as future work. We now present a few of our implemented renarration services.

6.1 Phonetics

The intent of this particular *phonetics* renarration service is to help non-native English speakers overcome the language barrier in accessing English content. This renarration can also aid in learning pronunciation in the English language. By using this rennarrtion, a user, who otherwise may be intimidated or overwhelmed by complex English usage, can now get pronunciation help on a renarrated source page.

6.1.1 *Functionality of Phonetics*

When the phonetics option is activated, the Microservice replaces identified English words in the source HTML document, with their phonetic equivalent symbols. A sample HTML page [7] has been used as a demo web page to showcase the validity of our Advices (as Microservices). See Figure #7. It shows how a sample text *This sample page demonstrates a tiny fraction of what you get with Blueprint*, which has been earmarked for phonetic renarration, is now replaced with its phonetic equivalent symbols.

We carried out two different trials with this webservice. In first trial we relied on 1000 most frequently used words and hard-coded their phonetic equivalents in a table, and used that to replace common or frequently used English words. This static model demonstrated our preliminary idea that a Microservice can indeed be implement as an Advice. But, the limitation of relying on a fixed set of 1000 words constrained its usage.

In our second trial, we sought to increase the range of words that can be phonetically renarrated. In this approach we worked with a live online phonetics translation service[8]. Here also we were able to replace English words with their phonetic equivalents. In addition, in this trial we were able to demonstrate the modularity and scalability of our microservice based design of an Advice. That is, we found that a microservice can be linked with other existing web services for expanding the functionality of an Advice. Functionally, this indeed increased the coverage of translatable

[7]http://www.blueprintcss.org/tests/parts/sample.html
[8]http://www.phonemicchart.com

words. But, it also came with a cost: there was a noticeable delay in service execution.

Thus, by way of these trials, we were able to demonstrate 1) the feasibility of using services for Advices, and 2) the scalability of having a service for an Advice.

6.1.2 Implementation of Phonetics

Phonetic renarration as a concern has been implemented as an independent AOP Advice. In this design, the Join Points are placed around the text which is selected by renarrator. In the current implementation, no tagging is placed but such a technique is intended for future versions. The chosen text is then Point-cut to the Advice on Phonetics. That is, the Joiner code enables the renarrator to place Join Points and Point-cut around the text selection to link it to the cross-cut concern of phonetics.

When renarrator chooses a text and presses the Phonetics button on the Annolet UI, a HTTP Request is posted to a webservice (e.g. phonetic-translive) running on a different server (e.g. //localhost:5000). If successful, the webservice outcome – i.e. phonetic equivalent – is then used to replace the original source text.

Here is the pseudo code which is triggered when renarrator chooses Phonetics option in Annolet UI of the Joiner in the source HTML document.

```
function get_phonetics(Sentence) {
   Post XMLHttpRequest ("//localhost:5000
        /phonetic-translive", Sentence)
   If readyState replace text with responseText;
}
```

Here is the functionality of Phonetics, which is implemented as a service on a server which is different from the Renarration Studio server.

```
@app.route("/phonetic-translive", methods=['POST'])
def phonetictranslive():
    get sentence, split it into words
    for each word in list
        normalize word
        send word to online translator by
             req.get('http://www.phonemic chart.com
             /transcribe/ ?w='+word)
        if successful
            answer.append(returned.string)
        else :
            answer.append(word)
return " ".join(answer)
```

6.1.3 Validation of Microservices

In the implementation of this service, we validated what we had proposed as values for using Microservices. One, we tested the idea of 'developer independence' by utilizing a new software developer who was different from the developer who implemented Joiner. Two, we tested the notion of 'language independence' by developing this Microservice in Python (while Joiner is using JavaScript). Three, we tested the notion of cascading services by using APIs from this service to connect to other existing services on the web. Four, we partially tested the idea of 'release independence' as the code repositories were separated. However, better service composition might be needed to further validate the dynamic addition and deletion of services.

Figure 8: A portion of a transcribed ISEC Conference page reduced for diagram brevity.

6.2 Translations

The intent of this particular *Translation* renarration is to help non-native English speakers get English web-content in their own vernacular. This renarration replaces English text with one of many earmarked target languages. By using this rennarration, a user, who otherwise may have been neglected by some existing published content, will now have access to its semantic contents.

Translation renarration is implemented in much the same was as Phonetics.

For implementation language translations we have relied on an external site – Yandex.com[9]. Though we relied on a specific site for translation, in implementation, our tool is agnostic to the specific approach used for translations.

This translations concern has been implemented as a AOP Advice. The Join Points are placed around the text which is selected by renarrator. In the current implementation, no tagging is placed but such a technique is intended for future versions. The chosen text is then Point-cut to the Advice on translations. That is, the Joiner code enables the renarrator to place Join Points and Point-cut around the text selection to link it to the cross-cut concern of translations.

When renarrator chooses a text and presses the Translate button on the Annolet UI, a HTTPRequest is posted to a webservice (e.g. language-translive) running on a different server (e.g. //localhost:5000). If successful, the webservice outcome – i.e. target language equivalent – is then used to replace the original source text.

Here is the functionality of translations, which is implemented as a service on the webservice server.

```
@app.route("/language-translive", methods=['POST'])
def languagetranslive():
   get sentence, from-language, to-language (default=en)
   req.get('https://translate.yandex.net
        /api/v1.5/tr/translate?key=
        '+translatekey+'&text='+sentence+
        '&lang='+fromlang+'-'+tolang
        +'&format=plain&options=0');
   soup = bs4.BeautifulSoup(res.text)
return soup.text;
```

See Figure #8. This shows a manual transcription of a single English text into multiple vernacular languages. The image showcases source text captured from the bottom portion of the ISEC 2017 conference web page. The portion edited is having text which reads *About ISEC*. In Figure #8 we wanted to showcase the language versatility in renarration. The output shown, however, is not from the translations Microservice. The output of the Microservice is similar and automatic in production.

[9]https://translate.yandex.com/m/translate

Figure 9: Sample page with highlight used for audio functionality.

6.3 Audio

Audio renarration sounds out the English text in audio modality. It is intended to help users listen to what is being shown. Users with visual challenges may benefit by this accessibility solution. See Figure #9. It shows a text being highlighted for renarration in audio format.

The implementation of Audio renarration is slightly different from the other previously documented renarration Microservices. Here we experimented with renarration code being placed in Joiner (in the Weaver) itself. That is, we did not create a new Microservice for this Advice. Our intent here was to demonstrate that Aspect implementation can be scattered anywhere in the system. By running the service from the Joiner side we are able to show that a concern can still be addressed in our system even if it is not modularized as a Microservice. That is, we show that the implementation can also be scattered.

The audio voicing of the text is provided by a back-end Text To Speech (TTS) system. For our test purposes we are using an online TTS called *Voice RSS*[10]. The documentation for the API for this system is given online as well[11].

Here is the pseudocode in Joiner that interacts with Voice RSS system.

```
function anno_audio(xpath)
{
   set Audio renarration to active
   get selected text string
   append a child element to clicked node
   child element to call
      "https://api.voicerss.org/"
      to include key
      to specify audio, src, controls, en and autoplay
           attributes.
   remove child node
}
```

The code for Audio renarration was implemented in JavaScript (to be inkeeping with the rest of the Joiner code). The function was activated upon clicking of the Audio button in Annolet UI.

This renarration validates the multimodal nature of renarration. It also demonstrates that code can be scattered and that base code language (e.g. JavaScript) can also be supported. Also, it demonstrates that other services may be triggered from within Joiner (weaver) code.

7. CONCLUSION

In general we are interested in the larger idea of improving web content accessibility by renarrating the web. Simple string replacements with pattern matching would only replace text locally. But, for improving accessibility, we are interested in renarrating the existing content at a semantic level. Adding new content, summarizing existing material, or adding links to existing content as annotation is what we consider as renarration of original source.

To enable such changes on an existing page, content needs to be replaced or modified at various points within the source page. This scatters and also cross-cuts the changes. It is here that we found a similarity between the original notion of Aspects and the applications of renarration.

In this work we were keen to extend the idea of Aspects and apply it to documentation (and not just programming).

We began by emphasizing the strong need for renarration of a an increasing number of web documents for varied contexts. We then motivated the need for Aspects oriented design by suggesting how renarration too had cross-cutting concerns that were orthogonal to the original HTML source. We then proposed to expand the usage of Aspects to now include documents. We then illustrated how the notion of Aspects can be used to facilitate renarration of web documents beyond traditional view of Aspects for programming. We then mapped different concepts like Join Points, Pointcuts and Advices for the context of web documents. We then presented *Renarration Studio* framework as an illustration of our Aspects based approach. We then detailed the use of Microservices based architecture for implementing the renarration services for text, transliteration and phonetics. We underlined how Microservices can help in adding new Advices independent of language.

We see this research work as a first direction to leverage the idea of Aspects in software engineering for the world of web documents, web personalization and semantic web.

[10]http://www.voicerss.org/

[11]http://www.voicerss.org/api/documentation.aspx

8. REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. Web services. In *Web Services*, pages 123–149. Springer, 2004.

[2] J. Araújo, A. Moreira, I. Brito, and A. Rashid. Aspect-oriented requirements with uml. In *Workshop on Aspect-oriented Modeling with UML*, volume 7. Citeseer, 2002.

[3] M. Bartsch and R. Harrison. An exploratory study of the effect of aspect-oriented programming on maintainability. *Software Quality Journal*, 16(1):23–44, 2008.

[4] S. C. Buraga and A. Panu. A web tool for extracting and viewing the semantic markups. In *International Conference on Knowledge Science, Engineering and Management*, pages 570–579. Springer, 2013.

[5] R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, M. P. Alarcon, J. Bakker, B. Tekinerdogan, S. Clarke, and A. Jackson. Survey of aspect-oriented analysis and design approaches. 2015.

[6] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86, 2002.

[7] T. Dinesh, V. Choppella, and S. Uskudarli. Re-narration as a basis for accessibility and inclusion on the world wide web.

[8] T. Dinesh, S. Uskudarli, S. Sastry, D. Aggarwal, and V. Choppella. Alipi: A framework for re-narrating web pages. In *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, page 22. ACM, 2012.

[9] P. G. Fairweather, V. L. Hanson, S. R. Detweiler, and R. S. Schwerdtfeger. From assistive technology to a web accessibility service. In *Proceedings of the fifth international ACM conference on Assistive technologies*, pages 4–8. ACM, 2002.

[10] J. L. Fuertes, R. González, E. Gutiérrez, and L. Martínez. Hera-ffx: a firefox add-on for semi-automatic web accessibility evaluation. In *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*, pages 26–35. ACM, 2009.

[11] S. Hanenberg, S. Kleinschmager, and M. Josupeit-Walter. Does aspect-oriented programming increase the development speed for crosscutting code? an empirical study. In *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 156–167. IEEE Computer Society, 2009.

[12] J. Kahan, M.-R. Koivunen, E. Prud'Hommeaux, and R. R. Swick. Annotea: an open rdf infrastructure for shared web annotations. *Computer Networks*, 39(5):589–608, 2002.

[13] M. Kersten and G. C. Murphy. Atlas: a case study in building a web-based learning environment using aspect-oriented programming. In *ACM SIGPLAN Notices*, volume 34, pages 340–352. ACM, 1999.

[14] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold. An overview of aspectj. In *European Conference on Object-Oriented Programming*, pages 327–354. Springer, 2001.

[15] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *European conference on object-oriented programming*, pages 220–242. Springer, 1997.

[16] H. W. Lie, B. Bos, C. Lilley, and I. Jacobs. Cascading style sheets. *WWW Consortium,(September 1996)*, 2005.

[17] B. Lucas. Voicexml. *Communications of the ACM*, 43(9):53, 2000.

[18] F. Munoz, B. Baudry, R. Delamare, and Y. Le Traon. Inquiring the usage of aspect-oriented programming: an empirical study. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 137–146. IEEE, 2009.

[19] D. Namiot and M. Sneps-Sneppe. On micro-services architecture. *International Journal of Open Information Technologies*, 2(9), 2014.

[20] S. Newman. *Building Microservices.* " O'Reilly Media, Inc.", 2015.

[21] N. Påhlsson. Aspect-oriented programming. *Topic Report for Software Engineering*, pages 11–03, 2002.

[22] M. Pilgrim. *Greasemonkey Hacks: Tips & Tools for Remixing the Web with Firefox.* " O'Reilly Media, Inc.", 2005.

[23] G. V. S. Prasad, T. Dinesh, and V. Choppella. Overcoming the new accessibility challenges using the sweet framework. In *Proceedings of the 11th Web for All Conference*, page 22. ACM, 2014.

[24] A. Rashid, P. Sawyer, A. Moreira, and J. Araújo. Early aspects: A model for aspect-oriented requirements engineering. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 199–202. IEEE, 2002.

[25] A. Rodriguez. Restful web services: The basics. *IBM developerWorks*, 2008.

[26] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, M. Wimmer, and G. Kappel. A survey on aspect-oriented modeling approaches. *Relatorio tecnico, Vienna University of Technology*, 2007.

[27] L. Seeman. The semantic web, web accessibility, and device independence. In *Proceedings of the 2004 international cross-disciplinary workshop on Web accessibility (W4A)*, pages 67–73. ACM, 2004.

[28] A. Sirotkin. Web application testing with selenium. *Linux Journal*, 2010(192):3, 2010.

[29] J. Thatcher, C. Waddell, and M. Burks. *Constructing accessible web sites*, volume 34. Glasshaus Birmingham, 2002.

[30] J. Thönes. Microservices. *IEEE Software*, 32(1):116–116, 2015.

[31] S. L. Tsang, S. Clarke, and E. Baniassad. An evaluation of aspect-oriented programming for java-based real-time systems development. In *Object-Oriented Real-Time Distributed Computing, 2004. Proceedings. Seventh IEEE International Symposium on*, pages 291–300. IEEE, 2004.

[32] B. Verheecke, M. A. Cibrán, and V. Jonckers. Aspect-oriented programming for dynamic web service monitoring and selection. In *Web Services*, pages 15–29. Springer, 2004.

[33] J. Viega, J. Bloch, and P. Chandra. Applying aspect-oriented programming to security. *Cutter IT Journal*, 14(2):31–39, 2001.

[34] M. A. Wehrmeister, E. P. Freitas, C. E. Pereira, and F. R. Wagner. An aspect-oriented approach for dealing with non-functional requirements in a model-driven development of distributed embedded real-time systems. In *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*, pages 428–432. IEEE, 2007.

[35] Y. Yu, J. C. Leite, and J. Mylopoulos. From goals to aspects: discovering aspects from requirements goal models. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 38–47. IEEE, 2004.