B. Tech. Project
Experiment as a Service

(Faculty: Prof. Venkatesh Choppella)
(Panel: Prof. Soma Paul)
Submitted By:
Vishal Kaja (201401065)

Motivation

Virtual Labs is using Open edX as the platform for serving virtual labs. Open edX platform has the following features:

- learning management system (LMS), LMS provides features that allow an instructor to manage the learning of students by setting up a learning plan, evaluation, fixing dates for submission of homeworks, etc.
- content management server (CMS), CMS allows content to be added, edited and served.

Though Virtual Labs is happy to leverage the features of LMS - single-sign-on, persistence, uniform UI, the CMS becomes a bottleneck since content editing is done through Open Edx studio. The tying of content editing through the studio undermines the freedom of using different editors and formats that are more flexible to generate content - read as html - that a browser can interpret. Apart from untethering the content creation from the hosting platform, the platform should be able to provide other services like single sign on, persistence, analytics and uniform user interface.

Prerequisites

Monolithic applications

a monolithic application is built as a single logical unit. All the logic for handling a request runs in a single process, allowing the use of basic features of the language to divide up the application into classes, functions, and namespaces. With some care, running and testing the application on a developer's laptop, and the use of deployment pipeline can ensure that changes are properly tested and deployed into production. This can horizontally scale the monolith by running many instances behind a load-balancer.

Monolithic applications can be successful, but are becoming increasingly frustrating especially as more applications are being deployed to the cloud. The change cycles are tied together i.e. a change made to a small part of the application, requires the entire monolith to be rebuilt and deployed. Over time it's often hard to keep a good modular structure, making it harder to keep changes that ought to only affect one module within that module. Scaling requires scaling of the entire application rather than parts of it that require greater resource. This has led to microservice application style to be adopted.

Microservice applications

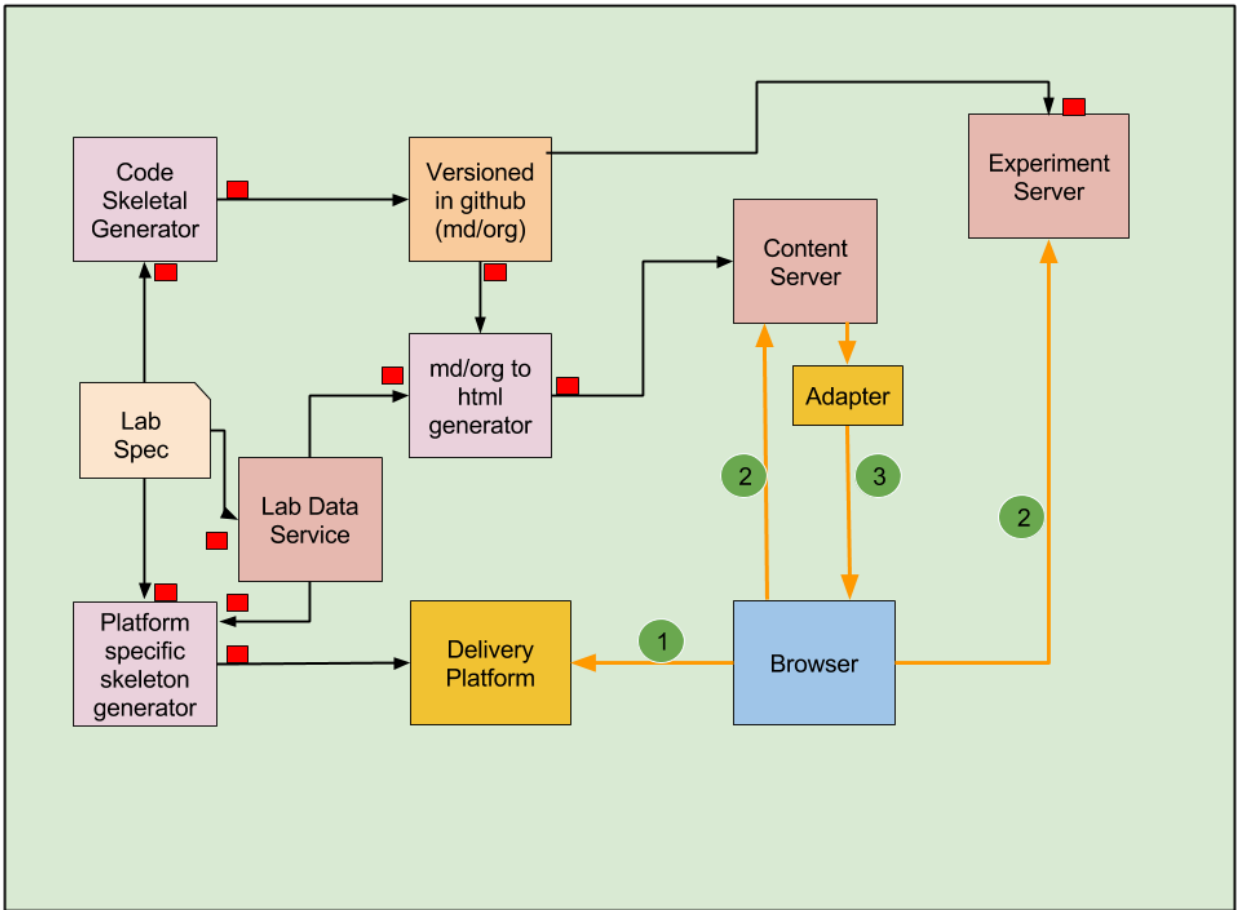
Microservice architectures will use libraries, but their primary way of componentizing their own software is by breaking down into services. One main reason for using services as components (rather than libraries) is that services are independently deployable and scalable, each service also provides a firm module boundary, even allowing for different services to be written in different programming languages.

Drawbacks of microservices

Using services like this does have downsides. Remote calls are more expensive than in-process calls, and thus remote APIs need to be coarser-grained, which is often more awkward to use. If you need to change the allocation of responsibilities between components, such

movements of behavior are harder to do when you're crossing process boundaries.

Virtual Labs design



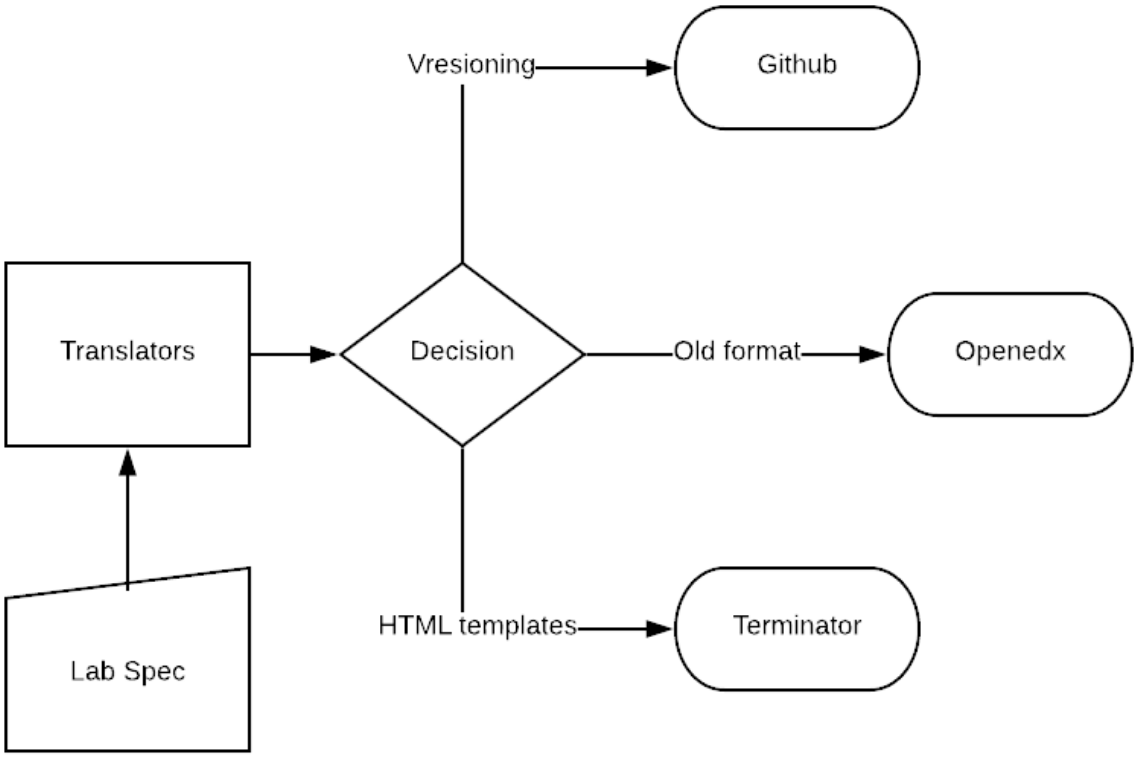
Problem Statement

- Build the Translators application: The Translators application is a microservice part of the VLabs platform design. The module works as skeleton generator for an experiment.
- Build the content adapter: The Content Adapter module is a micro-service part of the VLabs platform design. It is responsible for styling the content stored in the content server. The Content Adapter module tries to imitate a CMS tool.

Designing the Translators application

The Translators app accepts a lab specification for an experiment through its REST interface. The REST module interacts with the system module through the system interface and generates a version controlled experiment on a remote Github server.

The Translators app can also use the openedx module and generate the lab specification structure for openedx that can be deployed on the previous version of the vlabs platform. The system module executes the automated script of the openedx module and generates the lab structure specific to openedx.

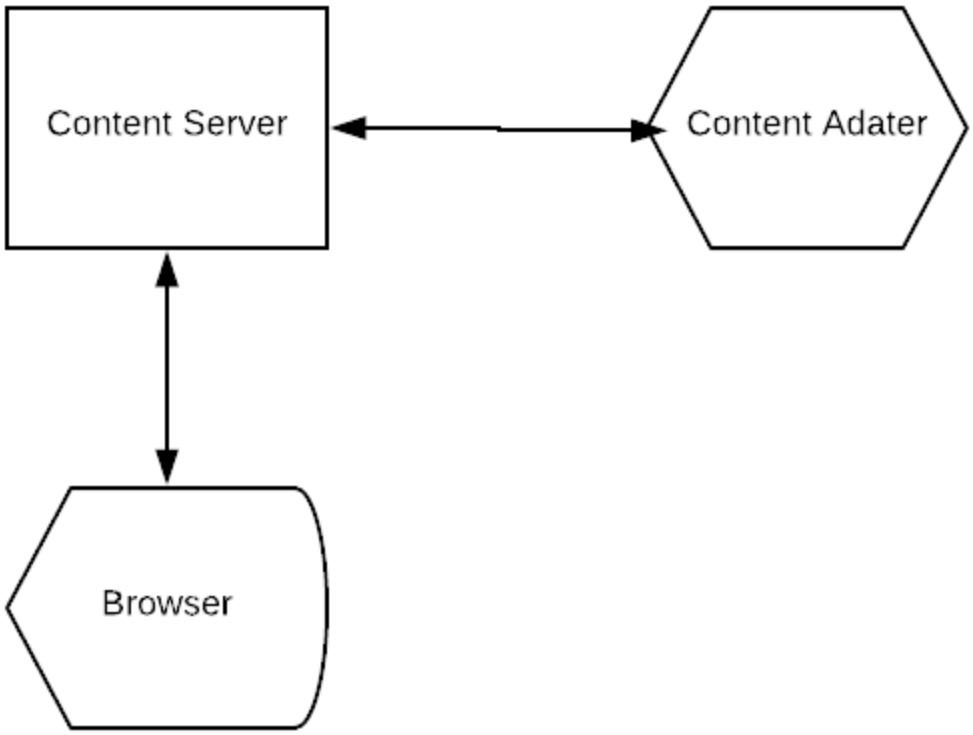


Source code: https://github.com/vlead/translators

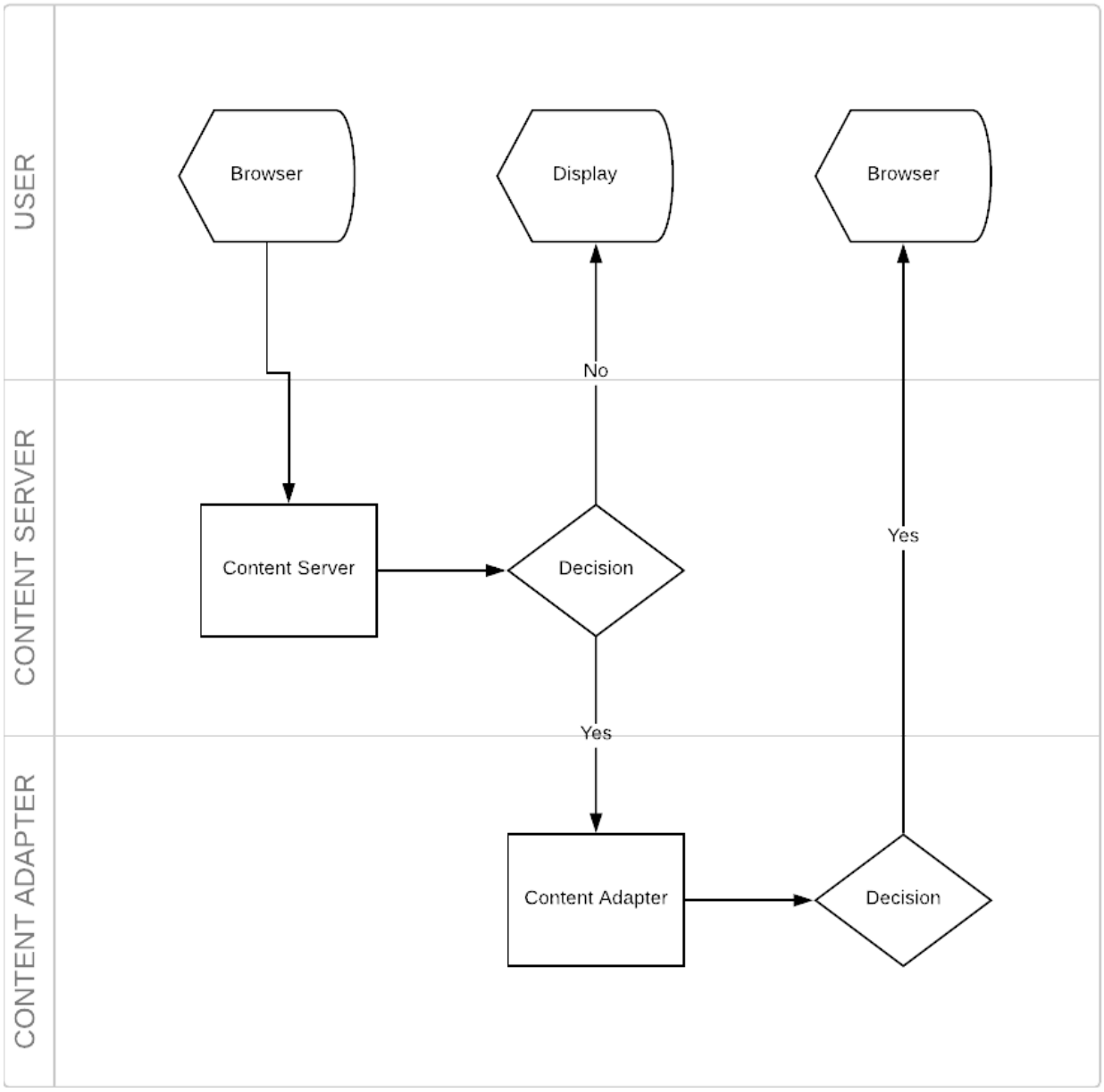
Designing the content adapter

When the web browser makes a request for a particular resource to the content server, the latter finds the resource and sends it to the Content Adapter.

The Content Adapter module receives an unstyled HTML template and modifies it based on the stored configuration of the adapter. This modified resource is sent back to the content server, which in turn forwards it the web browsers that originally made the request.



CONTENT ADAPTER FLOW



Source code: https://github.com/vlead/content-adapter

Observations and Results

- A friendly UI can be easily built to manage the content adapter.

- The content server integrates well with the content adapter. Similar modules such as authentication, analytics, and other content specific modules can be built using the same design principles.
- Currently, a broad format of content is being parsed by the content server, if this process of parsing is needed to be automated then a set strict restrictions need to imposed on the authored content.
- Further, with these restrictions, the authors can create events in their content.
- Once, there exists a predefined structure over the authored content several analytics tools can be tailored to work with virtual labs platform as microservices.