Hypothes.is is a web based annotation system with its origins in the Annotateit / Open Annotator project (OA). The code base for OA is less dense. It relies on fewer supporting technologies to build its output.

**Technologies in Use**

By contrast to Hypothes.is, the list of required technologies needed is much shorter in an OA implementation. Other variants of the annotator store have other optional technologies in use.

* Python
* Elasticsearch
* Javascript
* JQuery
* CoffeeScript

**Extensibility**

The Annotator documentation ( <http://docs.annotatorjs.org/en/latest/hacking/plugin-development.html> ) lays out how to extend the OA client to have more form options, how to present data, and how to expose that data for editing by the end user.

This system will allow us to extend the client:

* links to community profiles
* tagging and voting on personas.

The disambiguation of users and their personas online has to be negotiated from the server side. That will require a system that integrates with another sign-on system (eg. the OJS authentication; or a common authentication scheme that can allow authentication in one medium (OJS vs. OA) that will convey to carry authentication in multiple domains.

A number of desirable extension are available:

Marginal comments: <https://github.com/habeanf/annotator/blob/master/src/plugin/marginviewer.coffee>

Rich text editing:

<https://github.com/danielcebrian/richText-annotator>

Tagging:

<https://github.com/lduarte1991/tags-annotator>

**Data Storage**

Like Hypothes.is, OA uses Elasticsearch at its core for data storage. Elasticsearch is a NoSQL storage system. Because the system is schema-less, admitted data can be expanded to hold additional fields and stored in the database. When output, the data from the annotation record will retain all of the data that was stored originally.

OA can use other data stores and processing engines. There are OA projects ( <https://github.com/openannotation> ) for storage in SQLite, WordPress, Drupal, etc. as well as baseline “annotator store” application.

**The Benefits of Moving To Annotator**

This shorter list of required technologies, the established list of features and extensions, and the easier ability to include new fields from the plugin level introduced to the client side makes Open Annotator a more linear option to use for the sake of ease of development.

**Approaches to Carrying Out Application Changes**

**Maginal Comments**

The Marginal comments functionality can be satisfied by implementing the Margin Viewer plugin project: <https://github.com/habeanf/annotator/blob/master/src/plugin/marginviewer.coffee>

Motivations

This part of the Open Annotation spec ( <http://www.openannotation.org/spec/core/core.html#Motivations> ) covers off the “motivation” -- the type of annotation left by a user. This would help to distinguish annotated comments from other activity (comments, replies, tags, etc.). There is a plugin available for that: <https://github.com/uq-eresearch/annotator/blob/master/src/plugin/motivations.coffee>

**User Disambiguation**

There's nothing at the moment that'll do OAuth or single sign-on outside the limited sphere of subscription management, which doesn't interact with OJS's user records. When Alec Smecher was contacted, he felt this was a “tough nut to crack.”

An approach to consider:

* Include the OAuth library from PHP (<https://code.google.com/p/oauth-php/>) to give the OJS install the capacity to authenticate off of OAuth. Use this to extend the Auth plugin.
* Include the Python OAuth library code ( <https://github.com/simplegeo/python-oauth2> ) to give this capacity to OpenAnnotator.
* These bodies of code are starting points to shortcut how much code has to be generated to create user disambiguation
* More reading on the topic of Single Sign-On and OAuth: <http://www.thread-safe.com/2012/02/designing-single-sign-on-system-using.html>

**Tagging**

Tagging is built into OA. This element was also present in Hypothes.is.

**Data Exposure Through API**

Open Annotator has some API functionality to respond to requests with JSON output from the Annotator Store API. It doesn’t have the capacity to report about users who are in the system, but that could be added if we developed a fork that could allow for a “hook” system that made the store.py code into something more liberal. Instead of responding to the methods inside of that script, it could look for associated extensions (aka modules) that could provide additional methods to listen for and then make database calls to produce data in response. This would allow for information about users to be reported: the user’s existence in the system (keyed to their OAuth email address), and the public facing posts and comments made by the user.

**Reputation Management**

Reputation management should happen off site on an independent platform. This independent site should use OAuth or similar to disambiguate the users (the Klout / Twitter / Facebook interaction is a good example). At present, users can report what annotation systems they belong to, but their role cannot be automatically discovered. There is no established means for annotation system discovery. The users should be able to declare the annotation systems they participate in. When it comes to discovery, there needs to be a W3C spec for discovery (none could be found as of this writing). Hypothes.is and OA both publish their output on port 5000. If that is the defacto port of annotations, then URLs could be spidered on port 5000 with a simple request that would yield an answer to the question of whether the URL yields an annotation system. If there is a response, the basics of the annotation system (its name, it’s bookmarklet url, its registration url and other basics).

If participation in one or more annotation systems were known, then the statistics about a particular user could be known by pinging those services and request statistics (raw data or refined data relevant to a user).