Accelerating Circuit Realization via a Collaborative Gateway of Innovations

Ian J. Taylor (1,2)
Adam Brinckman, Jarek Nabrzyski (1)
Ewa Deelman, Rafael Ferreira da Silva and and Karan Vahi (3)
Sandeep Gupta, Soowang Park (4)

1. Center for Research Computing, University of Notre Dame, Notre Dame, IN, USA
2. School of Computer Science & Informatics, Cardiff University, Cardiff, UK
3. Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA
4. Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA

Emails: {ian.j.taylor, abrinckm, naber}@nd.edu
{deelman, rafsilva, vahi}@isi.edu
{sandeep, soowangp}@usc.edu

Contents

- Background
  - Domain of the user community
  - General Repository Requirements
- Design Choices
  - System Design options
- Architecture and Implementation
  - Open Science Framework Integration
  - CRAFT Specific tools
- Demo

DOMAIN OF THE USER COMMUNITY

Circuit Realization at Faster Timescales (CRAFT)

CRAFT Background

- It costs up to $100 million and takes >> 2 years for a large team of engineers to design integrated circuits for specific tasks (called ASICs = application specific ICs)
  - E.g. real-time conversion of raw radar data into tactically useful 3-D imagery.
- Defense Department engineers often turn to inexpensive and readily available general-purpose circuits, and then rely on software to make those circuits perform the specialized operations they need.
  - Speeds up the process of system design
  - But the resulting system provides lower performance (speed) and consumes far more power than ASIC

CRAFT Program

- Circuit Realization at Faster Timescales (CRAFT) is a DARPA funded program with the following main goals:
  - Seeks to shorten the design cycle for custom integrated circuits from 2 years to months
  - Plans to devise design frameworks that can be readily recast when next-generation IC fabrication becomes available
  - To create a repository of innovations so that methods, documentation, and intellectual property (IP) can be repurposed, rather than reinvented, with each design and fabrication cycle.
  - To make it practical for small design teams to take on complex custom circuit development challenges that are out of their reach today.

CRAFT REQUIREMENTS

Our Role, general requirements and Craft specific requirements

Our Role in CRAFT

Our Role in CRAFT:

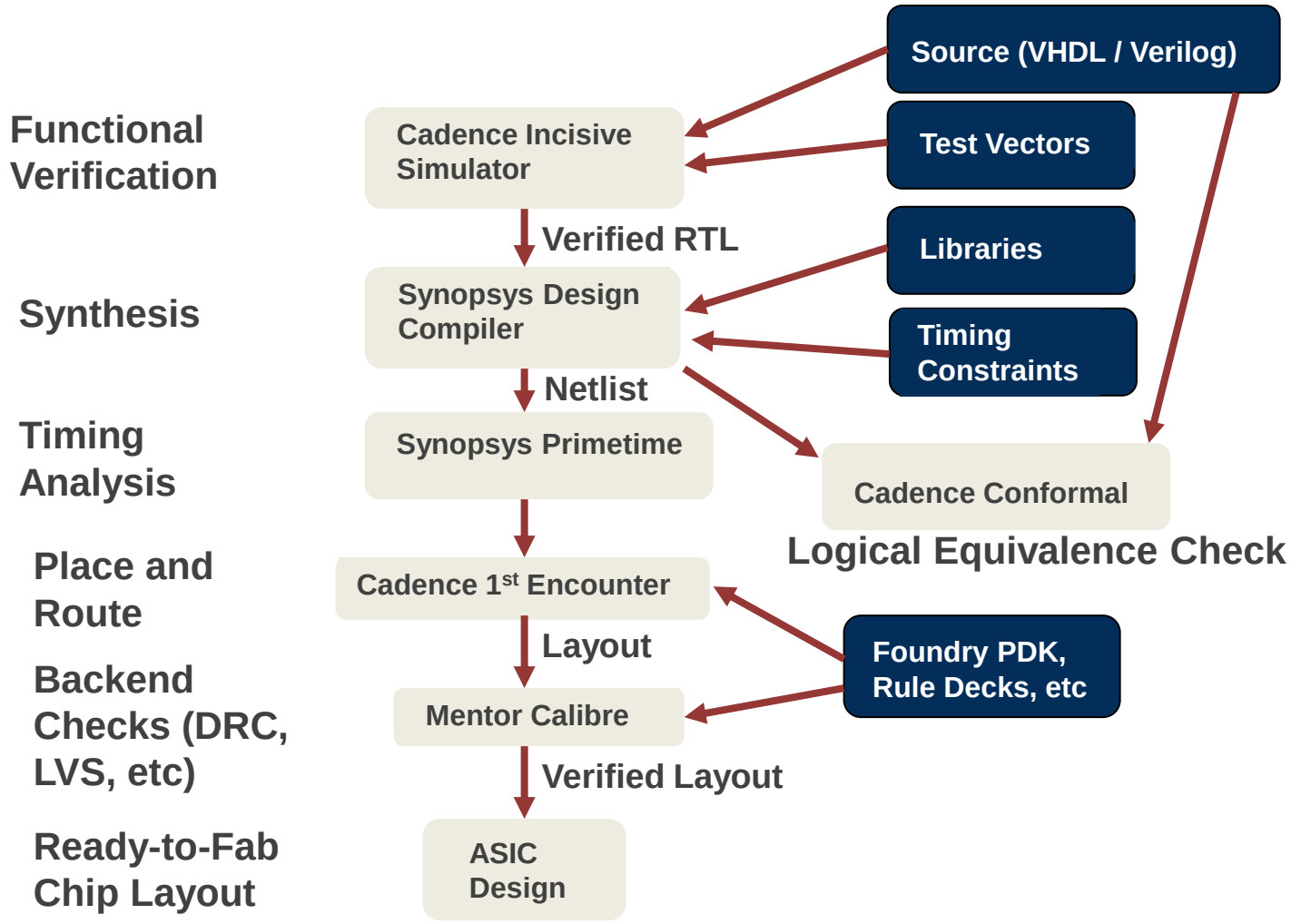
- There are 6 performer teams we are responsible for creating a repository of innovations for exposing, searching, documenting, and collaborating on design flows.
  - Need a collaborative space for development of chip ideas
  - Need to document and share *design flows*
  - And associated metadata about the flows - *intellectual property (IP)*
- There are a variety of teams, tools and IP, with potentially different needs, respecting privacy concerns

CRAFT Requirements

- High level collaborative requirements:
  - A *project* should form the basis of a CRAFT collaborative space.
  - Should contain details, participant info, shared files, a discussion forum, searchable tags, a wiki and an audit trail.
  - Sub-projects should be possible to allow hierarchical organization of CRAFT programs of research and performers
  - Each project should have a project leader (project admin e.g. the PI could be a project admin)
    - To manage that particular project and its sub-projects.
    - And invite collaborators to their project and assign them certain privileges.
  - Projects can be open or closed

CRAFT Program

- CRAFT needs support for design flows like this:



CHIP IP

- Each chip design is modular and typically uses a combination of newly designed and previously designed modules
- Module designs available for use across chip designs are called Intellectual Property cores (IP)
- We need to support complex IP Schemas that categorize IC types, features and attributes
- And provide mechanisms for a user to visualize the IP using an intuitive format

DESIGN CHOICES

Design choices, technology choices and architecture of the repository.

Design Choices

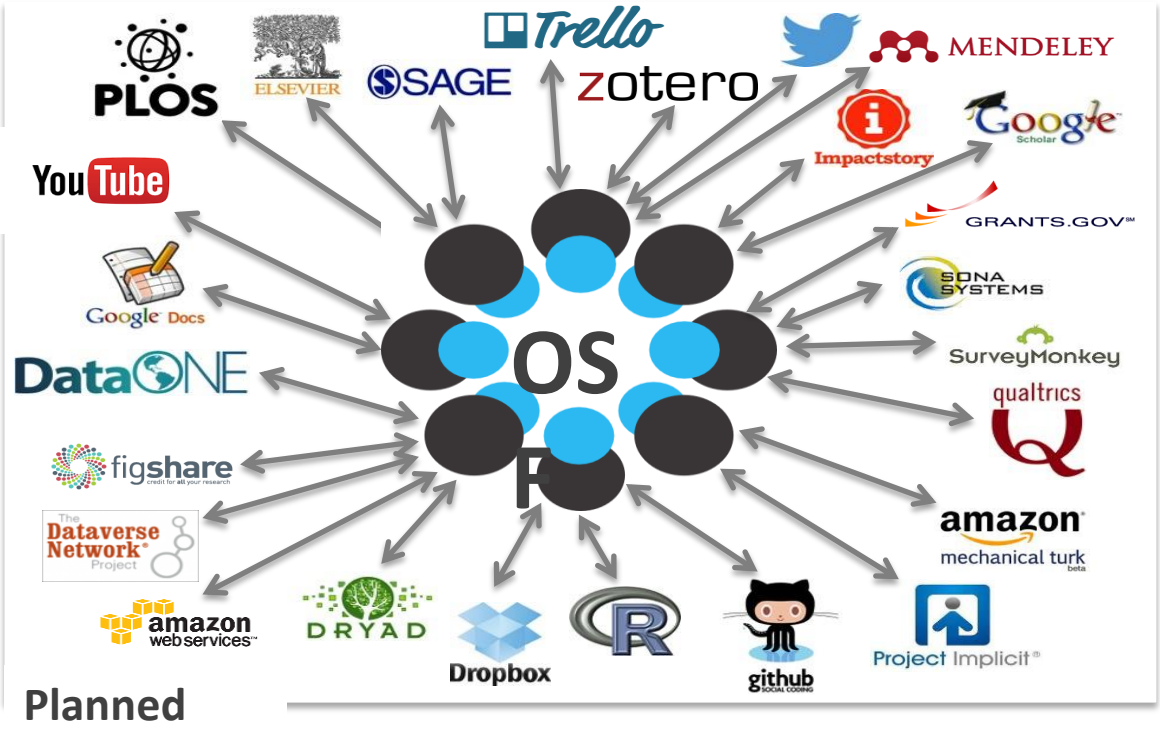
- We considered 4 different approaches
  1. architecting and implementing the system from scratch;
  2. customizing an existing system to meet the needs;
  3. creating a new interface to an existing system using a REST API;
  4. or creating an interface for CRAFT-specific features to an existing system using a REST API, and leveraging existing tools using a hybrid architecture.

Design Choices

- 1 is simple (no dependencies) but timeframe (7 months) present a high risk to create a production system meeting requirements.
  - Also, many of these features already exist on other websites; e.g., Github and Bitbucket already have many of the project-oriented features - reinventing the wheel is pointless.
- We therefore chose to research other systems – we identified two candidate systems:
  - The Open Science Framework
  - HubZero

OSF and HubZero

- Our study found that OSF had several advantages over HUBzero for CRAFT:
  - Lot of tools integrated
  - Support for wiki, tags, file sharing, comments & audit trails



- Has a full REST API – using Django REST Framework
- Also has an EmberJS binding to the models for REST API
- We therefore decided to capitalize on the recent advances of OSF to base the development of the CRAFT repository.

ARCHITECTURE & IMPLEMENTATION

Architecture technologies for implementation

Repository Organization

PROJECT

Project Information
Description
Tags

Contributors
Administrator
Participants

Design Flow
JSON File
Create/Edit/Visualize

Discussion
Topics
E-mail Notification

Data Files
OSF Storage
GoogleDrive, etc.

IP Cores
JSON File
Create/Edit/Visualize

Other Tools
Wiki
Details, etc.

SUB-PROJECT

Independent set of users, files, flows and permissions

Should be created by a user with '*write*' privileges in the parent project

Only public information will be seen by the parent project

Indefinite number of sub-projects

Craft Repository Architecture

Front End

http://www.craftproject.org

ember App

http://osf.craftproject.org

Open Science Framework

OAuth

Ember OSF API

OSF REST API

Mako Templates

Python Flask

CAS

Users

Projects | Authorization | Collaboration | Tags

Comments | Versioning | Sharing | Search

Docs

GDrive | Dropbox | Dataverse | …

Front End Craft app usies EmberJS and Semantic UI in Client's browser

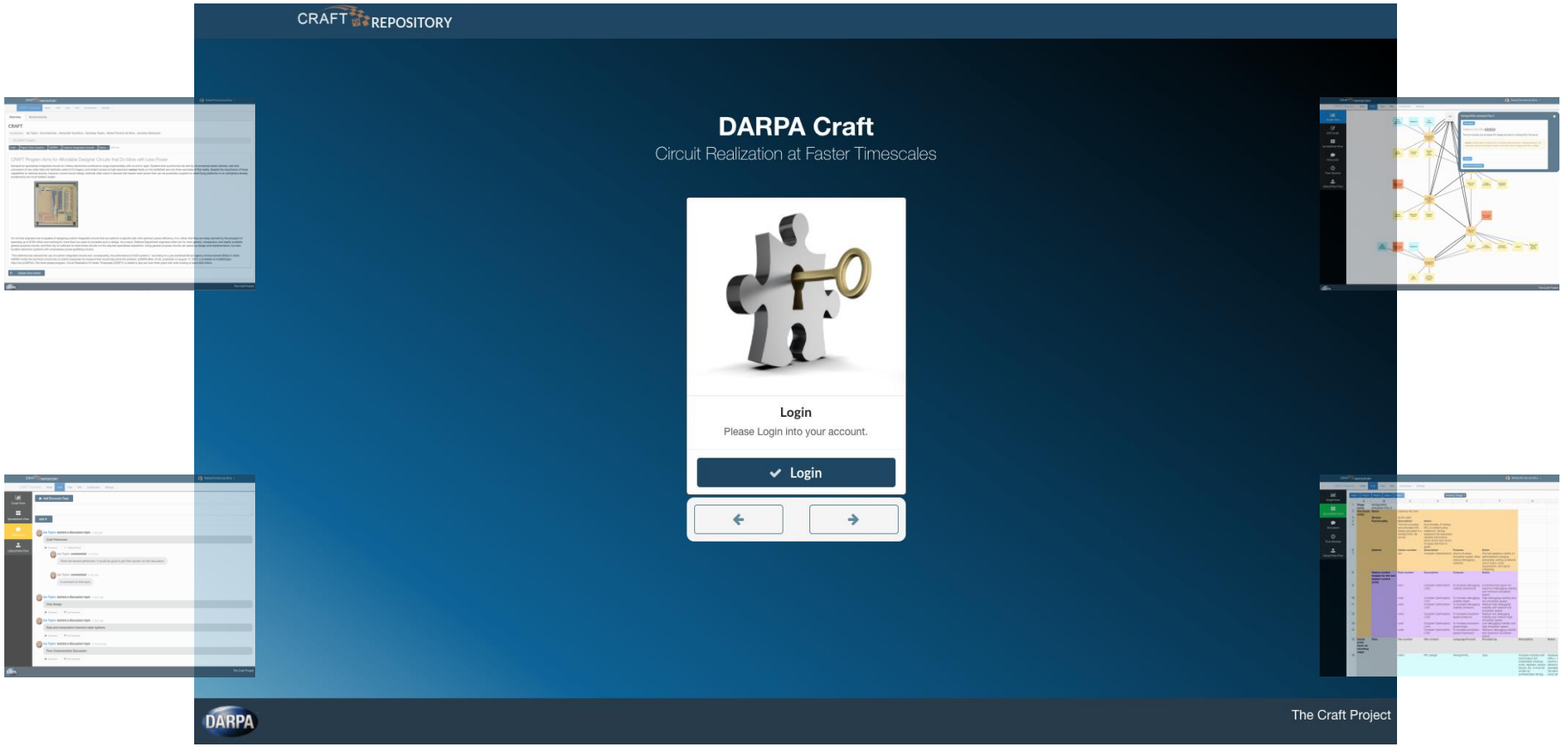
EmberJS models provide a Javascript interface to the REST API

OSF backend provides project space and API for authentication, authorization, files, comments, audit trails, search and versioning

WHY EMBER.JS?

- We wanted a SPA (Single Page Application) framework
  - A Web app that acts like an application, not a set of web pages
  - Fully portable to mobile devices – no need for native mobile development
- We looked at Backbone.js, AngularJS and Ember.js.
  - Backbone lacked too many features
- AngularJS v EmberJS
  - Ember's handlebars are more flexible than AngularJS directives, which extend HTML elements
  - Ember's data models provide full direct REST integration
    - Angularjs lacks good data integration
  - Ember Components are simpler and more modular than AngularJS directives.
    - PODS have CSS, Javascript and templates in a directory

Demo



https://craftproject.org/

- This work was funded by DARPA under contract #HR0011-16-C-0043 “Repository and Workflows for Accelerating Circuit Realization (RACE)”.

- And thanks to the Center of Open Science:
  - Jeff Spies for the vision and making this possible
  - Sam Chrisinger – for the help with the Ember OSF implementation
  - Matt Vander Werf, Caleb Reinking & Antelmo Aguilar for installing OSF at ND.

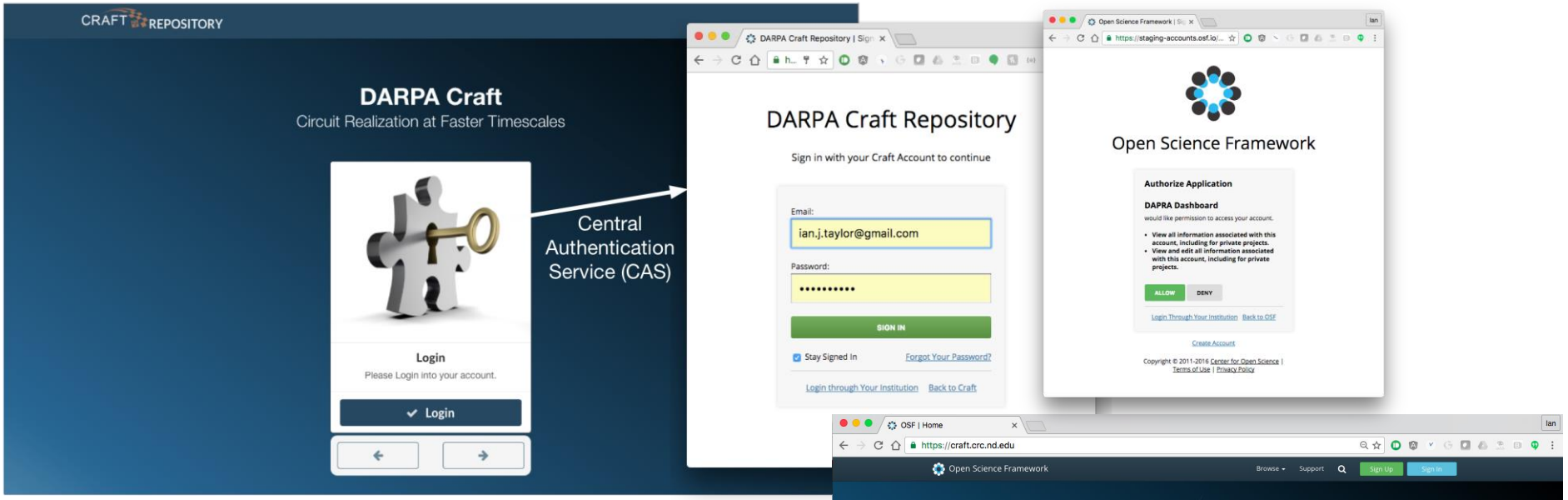
Questions?

SCREENSHOTS

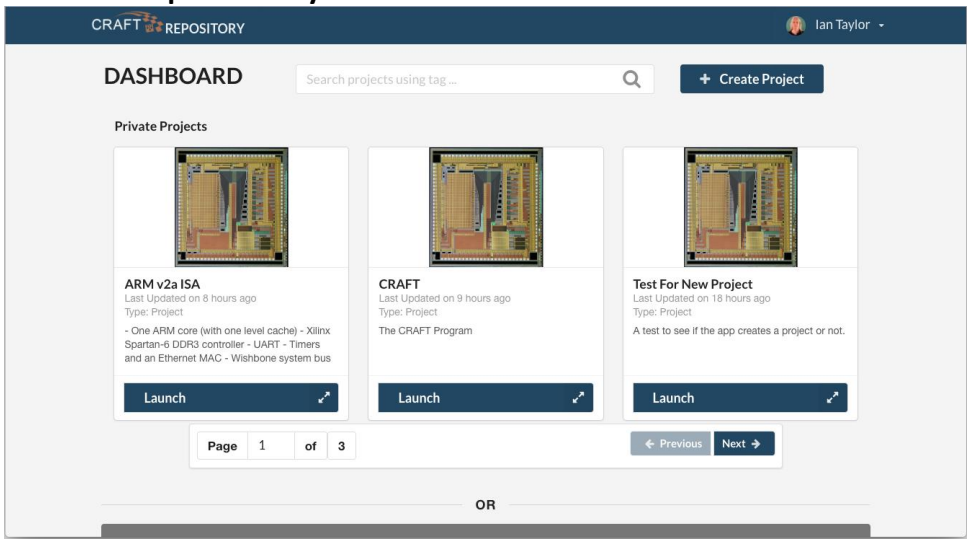
Screenshot walkthrough of GUI (in replacement of Demo, if not available)

Craft Repository

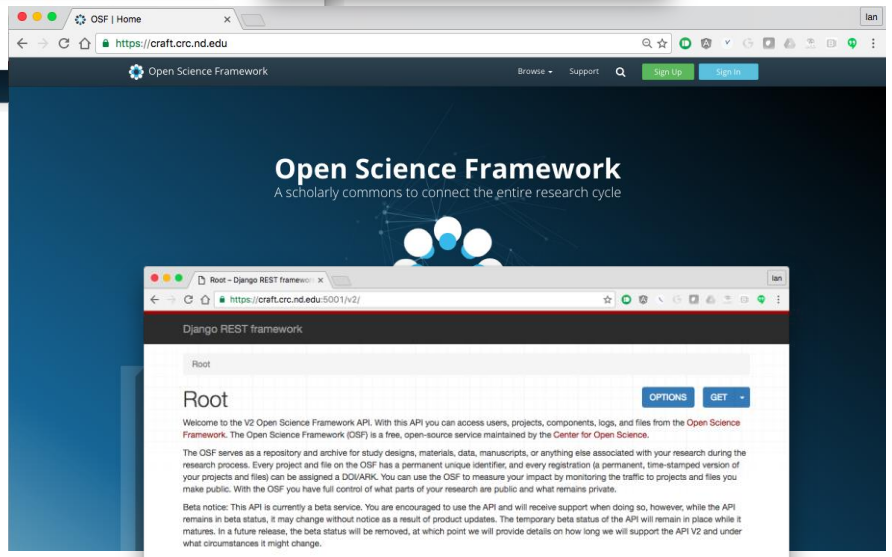
Craft Central Authentication (CAS) Service Flow



Craft Repository Dashboard



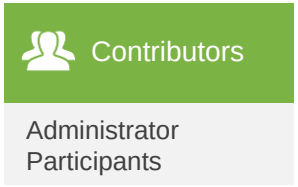
OSF Instance

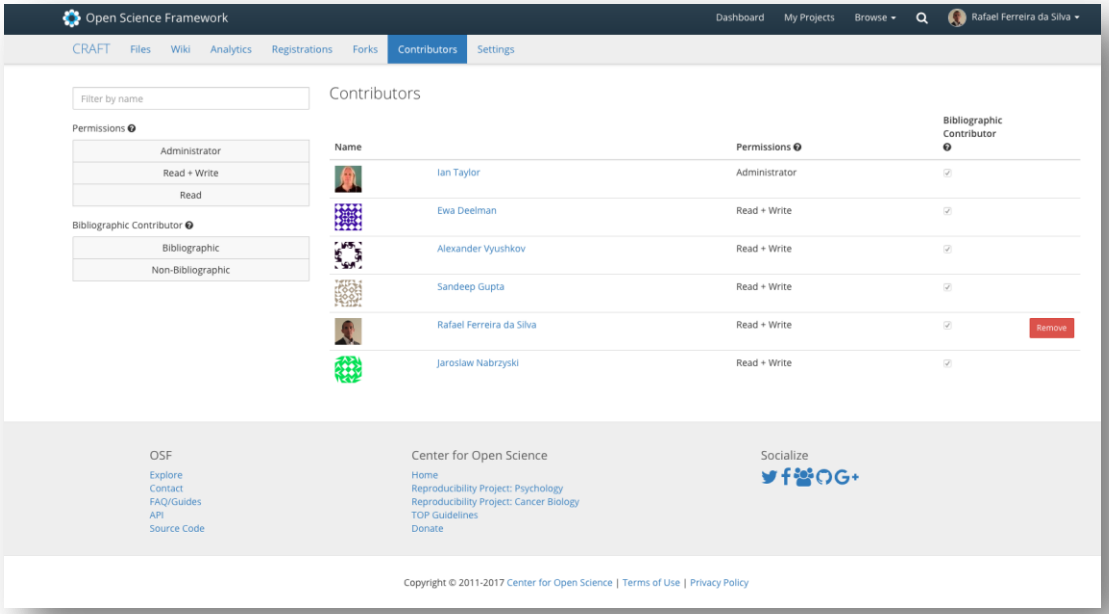


REST API

Contributors

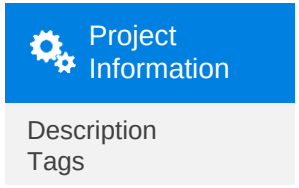
- Authentication is via a central authentication service (CAS)
- For authorization we have:
  - Add/Remove contributors to the project
  - Can invite external contributors to the project
    - Contributors of a sub-project do not necessarily need to belong to parent project
  - Define permissions
    - Administrator / Read / Write

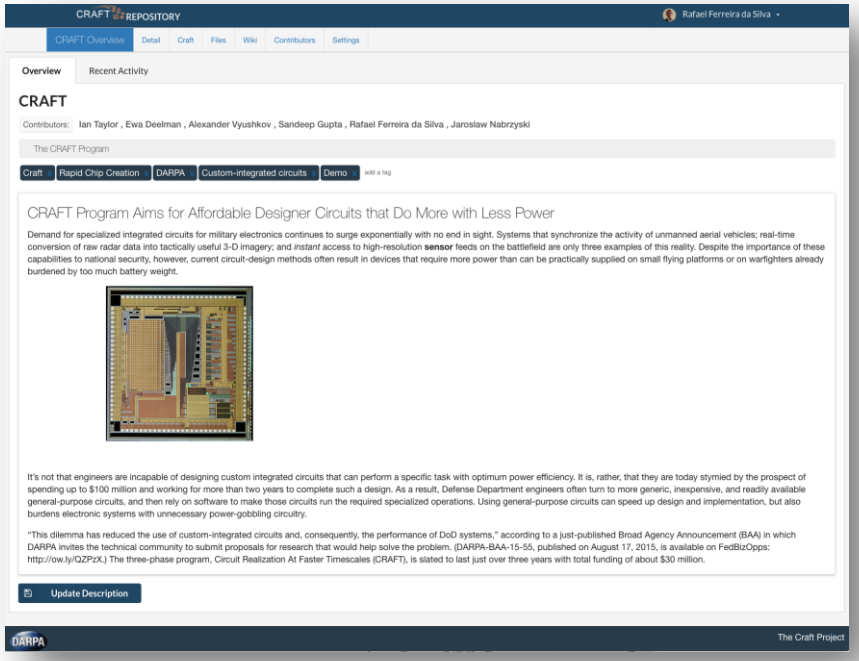




Repository: Project Information

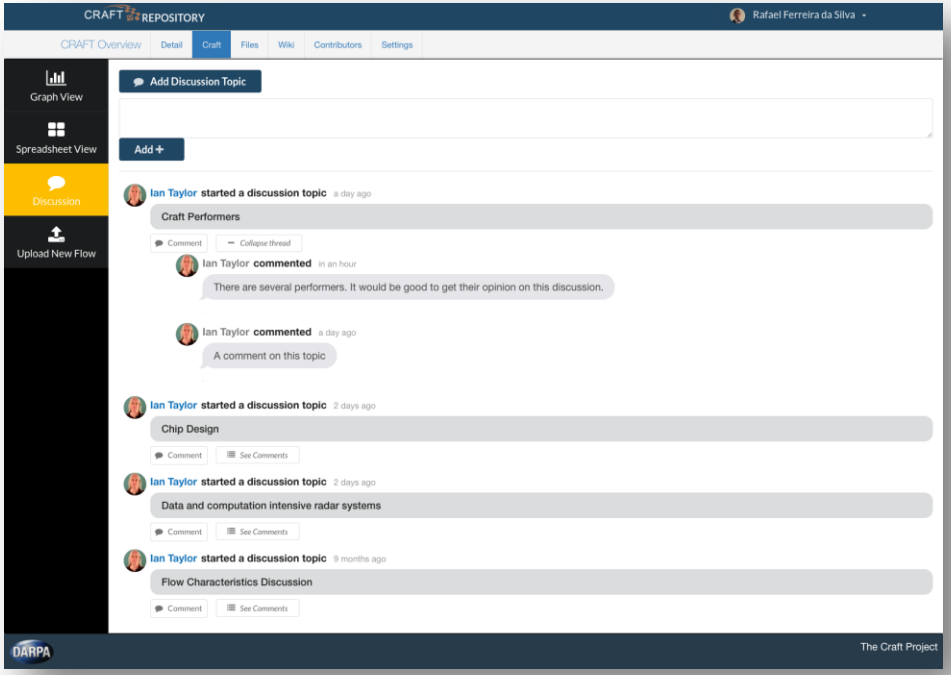
- Highly customizable Project Description
  - Uses HTML5 to enable simple edition of project description
    - What you see if what you get (WYSIWYG)
    - Images can be easily added by simple drag and drop an image file in the project description text area
  - Ability to define representative **Tags** that describe the project
  - Shows a list of contributors (users) to the project





Discussions

- Topic-based discussion forum
  - Any contributor from a project can **start/comment** a discussion topic
  - Contributors that have interacted with a topic, will be **notified** (via email) once any other contributor answers/replies a comment
  - Discussions are private to a project, if the project is private
  - **General discussions** (e.g., DARPA announcements), can be made through a separate project created only for this purpose

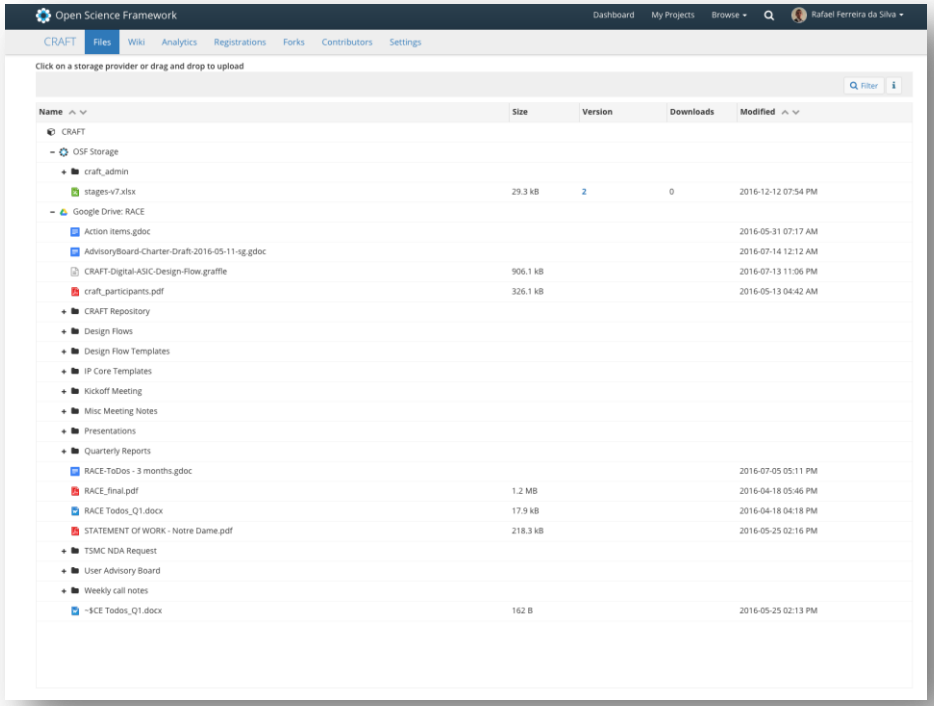


Data Files (Storage)

- The CRAFT repository allows the storage of medium-sized files (~15GB) via the browser interface
  - Files are stored in the OSF storage (ND server)

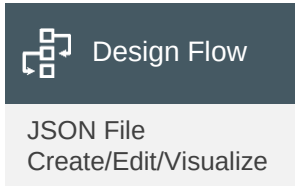
- For large files, external tools can easily be added to the repository
  - Google Drive allows users upload files up to 5TB in size

Note: a Google Drive folder connected to a project will be visible by **all project members**. To share files to a subset of contributors (**privately**), it is recommended to create a **sub-project** and share the data with the desired set of contributors

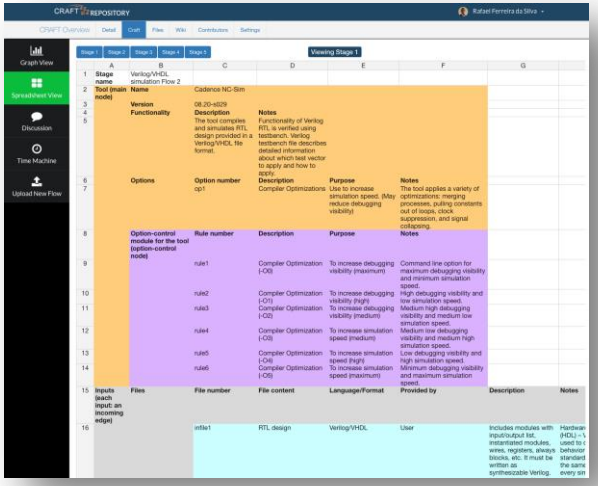


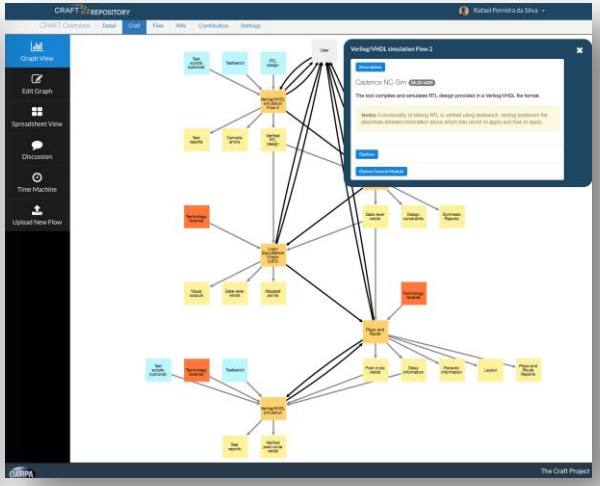
Design Flows

- Stores design flows described as JSON files
- Graphical visualization of the flow
  - High-level visualization (easier to see the control flow)
  - Table visualization (easier to see details)
- **Each project manages a *single flow***
  - Versions (history of changes) of the flow are automatically recorded in the repository
  - Sub-projects can be used to represent multiple flows from a single performer (see NVIDIA example)
  - Reason: <u>Facilitates collaborative efforts</u>



Creation/Edition of flows via the repository is currently being implemented

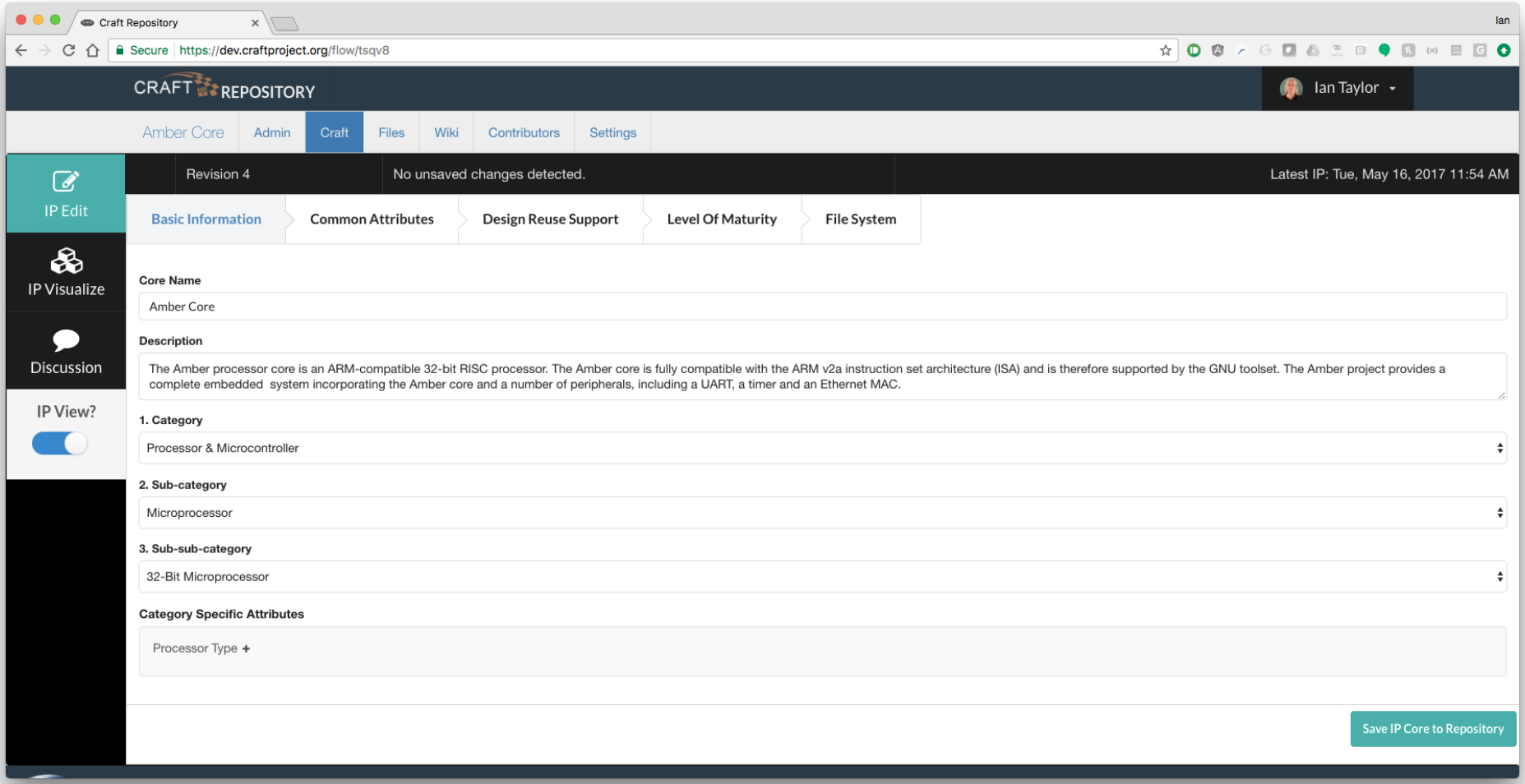




Interface for IP – Editing an IP

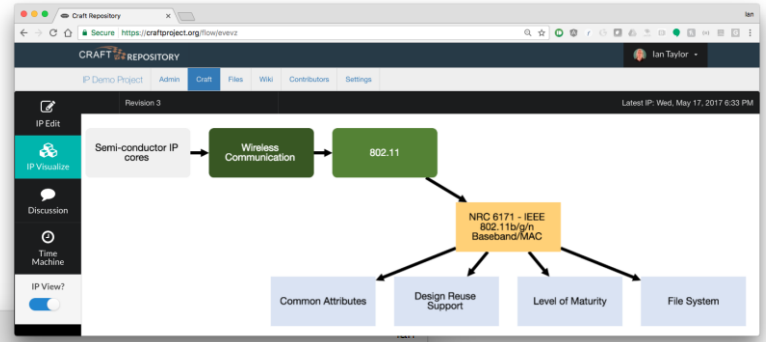
- Allows you to pick what attributes for Common Categories apply to IP

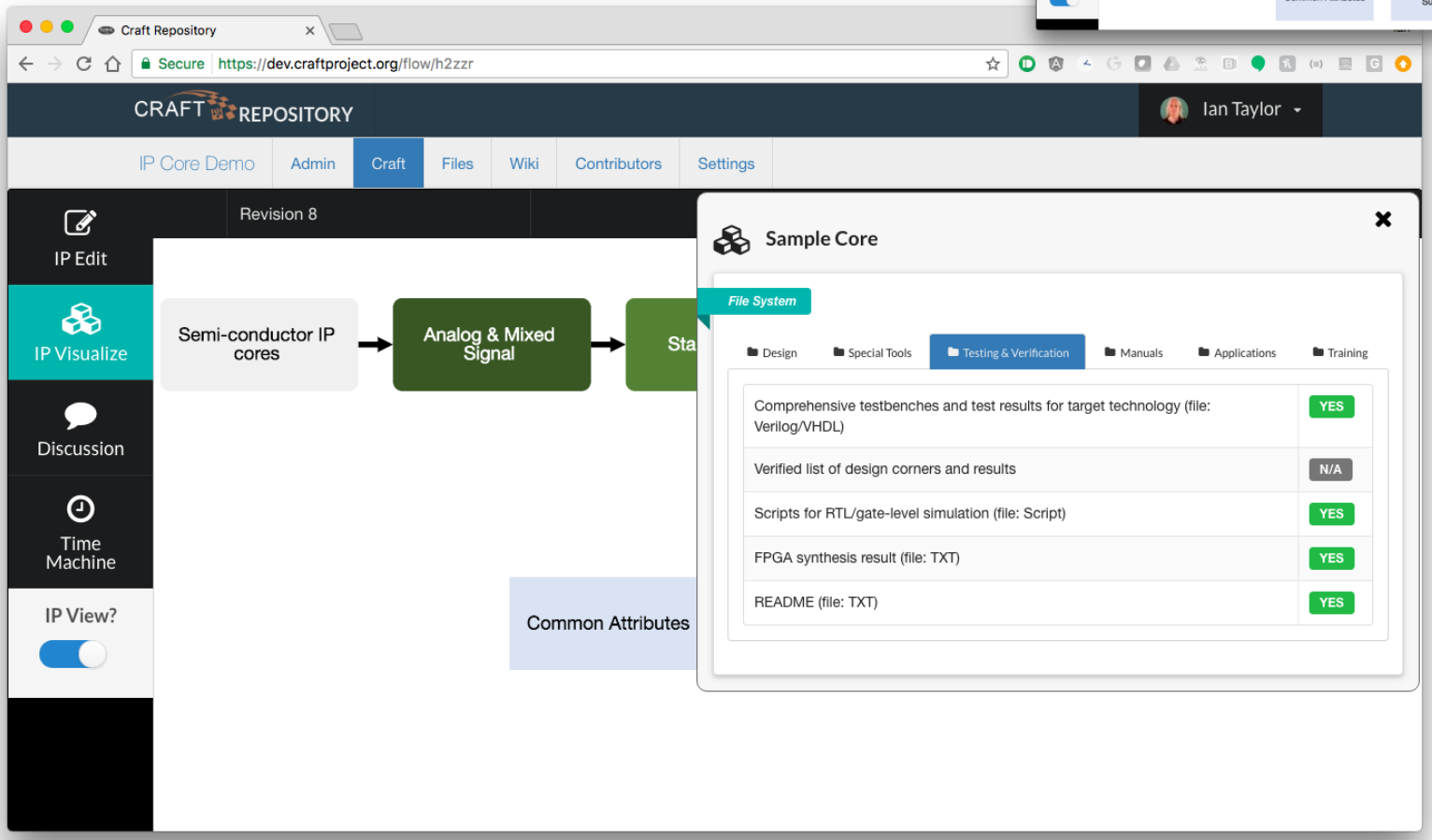
IP Cores



IP Visualization

- Shows category, sub-category, etc. to which IP belong
- At bottom shows common attributes
- Click on a (sub-)category to open the list of attributes





THE END

Of Replacement Demo