Fermilab Controls & ACORN GUI Strategies

John Diamond & Beau Harrison
Controls GUI Strategies & the use of Centralised Application Platforms
29th September 2022
Introductions

• John Diamond – Software Engineering Manager for user applications group in Accelerator Division Controls

• Beau Harrison - Software Engineering Manager and the level 2 manager for Control System Applications on the ACORN control system modernization project
Existing GUI Architecture Overview

- **ACNET = Accelerator Controls Network**
  - 30+ year history at Fermilab
    - 2007: VAX/VMS to PC/Linux migration
    - 2022: Sybase to Postgres migration
- **Consoles are a centralized resource**
  - Users connect to a Linux VM via SSH and applications are served to the desktop via X11
  - Applications are launched via an Index page organized around accelerators and sub-systems
  - Supports consoles located in the main control room, service buildings, offices and off-site (read-only)
- **Java Controls**
  - Java APIs developed in the early 2000s
  - Rich infrastructure developed but constrained resources meant most applications were never migrated
Existing Tech Stack

“Legacy” Application Framework

- Most applications are written in C/C++
  - But some FORTRAN still exists (!)
- Linux execution environment with relics of a VMS past
- Monolithic libraries for data acquisition, graphics (X11), plotting, database access, services, etc.
- Recently migrated away from Sybase to Postgres for RDMS
- Home-grown build system to manage deployment
  - Still using CVS for SCM! (Hoping to upgrade to Git over the next 12 months)
- Linux Command-line tools for development

Java Application Framework

- GUI applications built on Swing API
- Java Web Start for launching on user machines or in the X11 console environment
- Separate build-system, but also based on CVS
- Eclipse integration with a custom plugin
- Lack of buy-in from developers
Synoptic Displays

- Built with Java APIs
- “No-code” Display Builder
- Scalable Vector Graphics (SVG) for rendering displays
- Displays can be launched in a dedicated viewer or a browser
- Still supported but development has stalled
Existing Organizational Model

• Accelerator Division (AD) is responsible for building, maintaining, and operating accelerator complex

• Inside of AD, the Controls department does most “core” application work
  • ~25 full-time SEs
  • One group with 5 SEs dedicated to supporting console application development

• Anyone can develop GUIs!

• Machine departments do most “physics” application work

• Operations department also a big contributor

• Constrained resources have impeded the planning of a new UI framework
A key component of our application modernization strategy is the consolidation of applications based on functionality.

We hope that this will reduce the number of applications we are maintaining.

Our plan is to centralize shared logic in our services architecture and routing will be handled by a web server.

We have use cases for system experts that may dictate local logic.

Web Assembly (WASM) may be a good option for client-side performance.
ACORN Proposed Tech Stack

I looked to the independent annual surveys [The State of JS](https://stateofjs.github.io/2021/) and [The State of CSS](https://stateofcss.com/) for guidance on community interest and popularity.

<table>
<thead>
<tr>
<th>Languages</th>
<th>Frameworks</th>
<th>Layout tools</th>
<th>Chart tools</th>
<th>Authentication</th>
<th>Testing frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typescript</td>
<td>React</td>
<td>Tailwind CSS</td>
<td>Nivo</td>
<td>OAuth</td>
<td>Jest</td>
</tr>
<tr>
<td>Dart, Flutter</td>
<td>Svelte</td>
<td>CSS</td>
<td>D3</td>
<td>Kerberos SPNEGO</td>
<td>Cypress</td>
</tr>
<tr>
<td>Rust</td>
<td>Flutter</td>
<td></td>
<td>Plotly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solid</td>
<td>Electron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>React Native</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This question asked respondents why languages that compile JavaScript they’ve used.

**Typescript**’s ubiquity is clear across this community!

We feel that where JavaScript is a must for browser applications, Typescript will give our application developers some assurances through strict typing and the tooling that comes with that.
ACORN Proposed Tech Stack – Languages, Dart

**Dart** is the language of choice for the Flutter framework. Both tools are Google projects.

Dart is an open-source client-optimized language for fast apps on any platform.

Dart comes as a package deal with Flutter. There are pros and cons to adopting Google projects. Dart gives us a typed objected oriented language that was created with building user interfaces as it’s goal.
Rust has grown in popularity across many industries. I count Fermilab Controls as Rust curious with one operational deployment using Rust.

Rust’s performance, reliability, and productivity are advertised as its key attractors.

Rust is appealing for many reasons. Some of those reasons, such as guaranteed memory safety, contribute to slowing the quick iterative process advocated for in the agile manifesto.
React, Svelte, and Solid are all front-end frameworks that set out to achieve the same task using different strategies.

React is the most mature of these frameworks and was built and is maintained by Facebook.

Svelte was built to challenge the technical choices the React team made and rethinks many of the aspects of building responsive user interfaces.

Solid is the newest of these frameworks with the aim of being like React while being more performant. While it appears to be delivering on performance and developer experience, it lacks the community and ecosystem of its competitors.
Electron, React Native, and Flutter are all multi-endpoint frameworks that set out to achieve the same task using different strategies.

Electron is the most mature of these frameworks and was built by GitHub and is maintained by OpenJS Foundation. There are many notable applications using Electron such as 1Password, Discord, Teams, Slack, Twitch, VSCode, and WhatsApp.

React Native was built by Facebook to provide a more reliable experience to their users. The big benefit of React Native is using the same React knowledge for browser and native applications.

Flutter (not included in the survey) is not just a framework, but a software development kit for writing applications once and deploying it to many endpoints.
Cascading Style Sheets (CSS) has had its troubles over the years, but the World Wide Web Consortium (W3C) is pressing ahead with new ideas that are revolutionizing the way browser applications are laid out. W3C includes all the companies and organizations behind the major browsers as well as other companies that have a stake in the success of web standards.

CSS frameworks can be a useful shortcut by providing pre-made components and styles as well as a standard for creating a custom set of components. Tailwind CSS is particularly popular for its set of sane defaults and developer experience. Because CSS frameworks often have a compile step, they can implement standards before they’ve been officially accepted.

2021.stateofcss.com
ACORN Proposed Tech Stack – Chart tools

**D3.js** is the defacto JavaScript library for manipulating documents based on data. Many, if not all, other JS chart libraries depend on D3.js.

The learning curve is high, but a tool like this is appealing because it gives the developers full control.

If we choose React, we have some experience with beautiful **Nivo** library. This library has the components we know we need, and they are performant for our live data viewing.

**Plotly** has an open-source version that has a lot of quality-of-life features that other tools do not. This is a popular tool and is used by experiments at Fermilab.
ACORN Proposed Tech Stack – Chart tools, examples

<- D3

Nivo ->

<- Plotly
ACORN Proposed Tech Stack – Authentication

For a long time, Kerberos has been considered our “strong authentication” standard and thus the required method for accessing the control system.

SPNEGO is a web standard that’s been around since 1996, but browser support is inconsistent, and it requires configuring the browser to trust the authenticating domain. We struggled to make this work but built a proof-of-concept.

After discussing best practices with our security and authentication team, we are now investigating OAuth as a solution which is a popular standard backing many of the common logins that we use across the web today.
Jest and Cypress are examples of testing frameworks that are trying to solve different problems.

Jest is focused on testing code correctness, while Cypress is aiming to test "end-to-end".

We don’t have a testing standard now and so this area will be completely new for us. We need to understand tradeoffs of these types of testing and maybe we adopt both.

2021.stateofcss.com
ACORN Proposed Tech Stack – User developers

My previous slides focused on the tools for our application developers, but we have a vibrant community of user developers that we plan to support with robust APIs and easy to use tools such as a command line app.
ACORN Proposed Organizational Model

The ACORN project is broken into three categories under the Control System Applications objective.

- Framework team
- Application team
- UI/UX team

We don’t know yet if these categories will translate to operations.

I would like to explore agile-like practices which will suggest a team structure that’s compatible with team-based sprints.

While it’s appealing to off-load application development to other groups, we’ve identified this as a risk due to the lack of long-term support for these applications from other groups.
We plan to centralize shared application logic into services that are available across the control system.

Logic that is unique to a single application could be in the client application.

Our control system already has many shared services that don’t require applications to do ALL the work.

We will need a tool for routing, many frameworks include one. Third-party libraries are often available.

ACORN Proposed Centralized Application Logic
Existing Challenges

• Java Web Start
  • We are stuck on Java 8 without a path forward
  • Installs are tricky
• Fortran
  • There’s little to no expertise for Fortran
ACORN Challenges

• Buy-in
  • We need our users to want what we build.

• The unknown 😨
  • We don’t know what we don’t know.

• Is there value in an open-source design system?
  • We are working with Idaho National Lab (INL) on developing a style guide for accelerator applications.
  • I believe this could be broadly applicable.
Back up slides