18F

Administrative Office of the U.S. Courts

CM/ECF

Experiment & Iterate

Team

Glenn Grieves
Matt Henry
Norah Maki
Mark Meyer
Miatta Myers
Ben Peterson

ao-courts-ei-team@gsa.gov

Executive Summary

18F, a team within the General Services Administration (GSA), conducted a 12-week Experiment & Iterate (E&I) engagement involving the federal judiciary's Case Management and Electronic Case Files (CM/ECF) system, with the following goals:

- To begin employing some of our previous recommendations, including, among other things,
    - assisting the Administrative Office of the U.S. Courts (AO) in embracing human-centered design and a DevSecOps culture;
    - testing the feasibility of an API-driven CM/ECF system and the beginnings of a new data model; and
    - deploying to the cloud.
- To continue investigating ways the Judiciary can improve how it delivers digital services.

Our key takeaways from our E&I work are:

- CM/ECF software releases are far too infrequent and each new release contains far too few features meant to improve the user experience.
- The development of new CM/ECF features is disproportionately influenced by people who do not use the product on a regular basis, if at all.
- The siloing between AO departments and offices, first uncovered during the PA, contributes significantly to both the delays in releasing new software as well as to the paucity of user-facing features in each release.
- The AO's efforts to support—in a nationally deployed instance of CM/ECF— all the variations in courts' local rules, procedures, and business practices requires an unsustainable degree of software customization.

Based on these findings, 18F recommends that the Judiciary:

- Learn from regular, "hands on the keyboard" users of CM/ECF when determining what functionality to include in future versions of the product.
- Develop a strong product vision and roadmap based on input from these users.
- Reevaluate the extent to which customization of CM/ECF processes is needed and valued.
- Create a transparent process for how changes are made in CM/ECF and the criteria with which new features will be developed.

- Empower the AO's CM/ECF product owners by giving them the space to develop into a new role in which they can have a greater impact on the direction CM/ECF development takes.
- Create small autonomous CM/ECF development teams and empower them to make and deploy changes without the need for permission or technical support from those outside the team.
- Continue building capacity around DevSecOps and automation.
- Reduce the size of releases and deploy them more frequently.

Background

To get the most out of the substance of the report that follows, it may be helpful to situate the work described here against the backdrop of 18F's previous work with the Judiciary. Our engagement with the Judiciary envisions at least two phases: an initial Path Analysis (PA) Phase and the current Experiment & Iterate (E&I) Phase. During the PA Phase, we conducted over 75 interviews with more than 100 people throughout the Judiciary. These interviews covered a broad range of subjects including: CM/ECF user needs, business agility, organization and processes, and the AO's culture and legal mandates. Based on this research, the PA team produced a report that identified several issues with the current state of CM/ECF and included preliminary recommendations for a path forward. In our PA report, we specifically recommended that the Judiciary, among other things:

- Build a new, open source case management and electronic filing system with modern technology and architecture. Some components of the new system could be built in-house, but for others it may be more cost-effective and efficient to use contractors who have relevant domain expertise.
- Consider cloud computing as an option.
- Simplify AO operations and encourage courts to innovate through Application Program Interfaces (APIs).
- Adopt a DevSecOps culture to support the integration of security and operations concerns at every phase of the software development lifecycle; from the architectural design phase through, testing, deployment, delivery, and production.
- Stand up a new cross-departmental and cross-functional team to develop and maintain the new national system for courts. This team will need some domain-knowledge support but should be empowered to build the new system independently.
- Meet with CM/ECF stakeholders to leverage their domain expertise during planning and building phases.

Current Phase of Work

This report describes the work done during the E&I Phase. This work differed from our work in the PA Phase, both in scope and methodology. During the PA Phase, we looked at the state of digital service delivery throughout the Judiciary by conducting extensive interviews and then issued broad recommendations. The ambit of the E&I Phase was considerably more circumscribed: to put our hands to the keyboard and begin building a product in line with our PA recommendations, in order to further evaluate how well those recommendations can be applied and adapted to the unique context of the federal Judiciary.

Prototyping the Record on Appeal

The main focus of the E&I Phase was prototyping a new tool for compiling and transmitting a Record on Appeal. 18F developed this use case in consultation with our partners at the AO. This use case made sense for a number of reasons, including: its applicability to bankruptcy, district, and circuit courts; its discreteness and tractability for a short period of work; and the opportunity it would provide to see how best to integrate Federal Rules and differing practices into the software development process.

As noted above, our PA report included recommendations that the Judiciary adopt certain cultures and practices, including DevSecOps and human-centered design. In addition to modeling these practices while developing the Record on Appeal prototype, 18F worked with a small team from the AO to begin the process of adopting and socializing DevSecOps and human-centered design practices throughout the various AO departments and offices that support the courts with nationally deployed software.

Our PA report also emphasized the importance of employing Agile methodologies when developing software, and that one of the most critical components of Agile is working in short cycles to develop and maintain tight feedback loops. To that end, the work of the E&I Phase can be viewed as an initial test of the specific PA hypotheses listed above, and evaluate whether and how they may help the AO deliver better digital services in the future.

This report summarizes 18F's findings after working on the Record on Appeal prototype for roughly 12 weeks.

One final note before embarking on the substance of this E&I report: ultimately, the Judiciary may work with a vendor or vendors to create the next version of its case management and electronic filing software, and that work may or may not build on the code 18F delivered during the E&I Phase. Whether something like the Record on Appeal prototype becomes part of an eventual new system or is scrapped, the process of developing it helped us—and the AO—understand more about CM/ECF users, their goals and experiences, and how we can better meet their needs.

Research

Research overview during the E&I Phase

During the E&I Phase, we used the human-centered design process, which involves qualitative research activities to learn about the experiences of those who will be directly impacted by, or will ultimately use, the resulting designs. We also demonstrated this process for the AO's team of product owners, who are currently responsible for representing court interests and business needs in the process of developing and maintaining CM/ECF. Moreover, we developed initial understandings of CM/ECF's users, their goals and processes, the importance of the tasks they perform with the software and the ways CM/ECF sometimes prevents them from accomplishing those goals efficiently. Core to our research process was quickly and cheaply prototyping a Record on Appeal user interface and investigating user experiences so that the right features are prioritized.

The value of human-centered design research on product development

The four areas that drive product development decisions should be user needs, stakeholder input, technology considerations, and business needs. Currently, CM/ECF product decisions are heavily influenced by AO stakeholders and expert panels made up of judges and court staff, with little input from the community of users of a particular tool or process under consideration. While it was not the focus of our work to catalog examples of what features are not being used, the 18F team heard from multiple people about their frustration with released features that they did not find value in potentially adopting, such as Workspace and Judge Review Packets. This creates substantial inefficiencies in the development process, as valuable time and other resources are spent developing functionality that is deployed nationally but never used, used only infrequently, or used by only a very small subset of CM/ECF users. More research with the people who are expected to directly benefit from new functionality is needed to better understand how useful it will be to the national community. This will give Product Owners the insights they need to make sure new features are worth the associated development and maintenance costs, and the talking points to communicate the value of new features.

Research Activities

Our research activities during this phase focused primarily on semi-structured interviews (Contextual Inquiry) about the current Record on Appeal process and observing users as they attempted to navigate a design (Usability Testing), which in this case, was a prototype of a new Record on Appeal interface and workflow.

We spoke with and observed 19 people, intended to represent the Clerk's office to indicate CM/ECF court users, including non-managerial clerk's office staff who complete the ROA process regularly from different circuits and across appellate, bankruptcy, and district courts. Ultimately, we included 6-7 of these CM/ECF users in each of the three phases of this research. We shared our initial prototype with 6 users. During our refinement rounds of usability testing, we reached back out to the people we learned from during the contextual inquiry to gain additional insights from their interactions with the prototype. We also engaged the other users in subsequent rounds to learn where there was friction between user experiences, to better understand where the prototype might create new friction and confusion, and to better understand what an interface needs to provide in order to meet the needs of CM/ECF users who are involved in the Record on Appeal process.

We also facilitated a virtual Attorney Workshop with 9 attorneys from across the country, representing a range of practice areas including civil litigation, bankruptcy law, intellectual property law, and criminal law. The participating attorneys have experience using CM/ECF to file documents in appellate, bankruptcy, and district courts. We used a digital whiteboard during the Attorney Workshop to understand how non-judiciary CM/ECF users are interacting, and want to interact, with the system. The AO's team of product owners helped prepare for the Workshop by defining non-Judiciary CM/ECF user groups, observed the Workshop and then joined the 18F team in a post-Workshop session to discuss research strategies, and review qualitative data from the Workshop for patterns and insights.

Research Goals and Findings

In this section, we provide a summary of our goals for, and the outcome of, each round the research we conducted during the E&I Phase.

Contextual Inquiry (Research Round 1)

Goals

- Document similarities and differences in the needs of federal appellate, bankruptcy, and district courts by researching and mapping the Record on Appeal task-flow.
- Develop insights into the user experience from the perspective of people in different roles involved in appealing a case from one court to another.

Outcomes

- We documented the current Record on Appeal process by creating Journey Maps, which are visualizations of the major interactions shaping a user's experience of a product for each research participant. We also synthesized the similarities and differences between different users' task-flows and created a streamlined map of the common journey. Then, we based the rough Record on Appeal prototype (described below) on this customer journey map
- link to journey maps

Usability Testing with the Rough Record on Appeal Prototype (Research Round 2)

Goals

- Validate our understanding of the Record on Appeal process and how people manage it and use software tools to facilitate it.
- Test a more streamlined way to transfer data between courts.
- Test the concept of alert notifications to indicate that an action is ready to be taken, intended to replace the current, email-dependent practice.

Outcomes

We learned about the process and points of friction getting in the users way to enable us to draft design principles, or the essential elements that a design should contain and user stories (also called user scenarios), which are stories detailing what the users experience with a feature should entail. Both of these aided in the second round of prototyping.

Below are some of the Record on Appeal (ROA) user stories that guided our prototype development:

- As a Court User, I need to create/review/transfer the ROA/Remand so another court can easily access case records
    - ROA creation
        - As a Court User, I want the option to manually or automatically create a ROA so I have flexibility and speed in my process.
    - Docketing
        - As a Court User, I need to sort the docket so I can easily access the records I need.
        - As a Court User, I need to select and seal records on the docket so I can transmit the right files with the right level of access.
        - As a Court User, I need to select who I'm sending the ROA to so I know they go to the right person/court.
    - ROA transmission to judges
        - As a Court User, I need to unseal and review documents so I send the right information to the judge.
        - As a Court User, I need to select who I'm sending the ROA to so I know they go to the right judge(s).
        - As a Court User, I want the option to manually or automatically open a new case from a ROA so I have flexibility and speed in my process.
        - As a Court User, I want the option to assign the case to a case manager so that the right Court User is reviewing the ROA.

We also developed the following initial user experience principles for a future CM/ECF:

- Streamline Navigation
    - Reduce the number of screens and clicks needed to accomplish tasks
- Interaction Feedback
    - Guide a CM/ECF user when they encounter friction or are blocked
- Intuitive interactions
    - Simplify the process so a novice user can complete activities without training
- Default to consistency
    - Allow for customization only when necessary to allow people interacting with CM/ECF in different courts and circuits to be able to navigate the same interactions in the same way

Usability Testing with the Refined Prototype (Research Round 3)

Goals

- Validate our understanding of the Record on Appeal process and how people manage it and use software tools to facilitate it.
- Test a more streamlined way to transfer data between courts.
- Test the concept of alert notifications to indicate that an action is ready to be taken, intended to replace the current, email-dependent practice.

Outcomes

We included the AO's product owners to help us draw lessons from this round of Usability testing. They helped to label categories for patterns of comments and behaviors that surfaced during the tests. They related these categories back to the features they have been considering and speculated on what might help users in their attempts to use the prototype. Here are a few examples of what we uncovered as a joint 18F/AO research team.

Documents to reference when pulling together the Record on Appeal

Whoever is doing the activity needs more than the docket materials open to make a judgment on what to include in the record on appeal. They need to see the documents and be able to make cross references. More information is needed than simply "it could be easier." An example quote from one of the research participants about this:

- "I'd want to look at the notice of appeal, then I'd open the case by adding the parties. I'd have to find the judge on the case."

What users need, expect to see in a sample case

Right now, users don't get all the information they need just by looking at the docket. They need to go to multiple areas to see what's going on in the case. This might be an opportunity to surface more information that people are looking to find in one place. An example quote from a research participant about this:

- “A lot of people do their work by judge and terminal case number[1] and so much of what we do is referenced by the case... I always like to see the top part of the docket where it lists all the parties and attorneys for the parties.”

People expressed difficulty in the current process for collecting and transmitting documents as part of the Record on Appeal process. They were looking for signals from the prototype interface to let them know the documents they were creating (1) would be accepted by an inflexible upload process and (2) were the correct files to transmit. One participant mentioned their current process of compiling a few documents at a time to split a single ROA in multiple, smaller records to make sure all the documents would be accepted by the system, “file size is important because of file upload limits. Would split the [ROA] into multiple uploads to accommodate file size.” Another example quote from a research participant about this:

- “After hitting ‘transmit” I’d like to see the whole PDF just one last time to make sure it had everything I put there or if the things I sealed would be sealed”

Sealed/unsealed docket entries

The lock icon’s inclusion and placement in the prototype was not intuitive to users who were not sure how to interact with it. Example quotes from research participant about this:

- “Was not sure what the lock icon would do...not sure if it’s locked how it would be visible to the court parties.”
- “I guess if I'm sending this to another court they should have access to them, but if not primarily sealed on our side it wouldn't be sealed for another court.”

New concepts for interactions created confusion

The prototype introduced new language to describe streamlined interactions or actions research participants could take, but they lacked the necessary context to make them clear. Example quotes from research participant about this:

- “The ‘transmit record’ button is sending it but also docketing it on my docket?”

[1] Many clerks and judges allocate work by assigning their employees responsibility for a certain set of the judge’s (or judges’) cases based on the terminal case number, e.g., by assigning one employee responsibility for managing all cases that end in an even number and a second employee for all the cases ending in an odd number.

- "'Create docket from scratch.' Not sure what that is, maybe an electronic version instead of the record, it would create a docket?

User Interface suggested improvements for usability

There were instances where one or a few research participants made comments that did not rise to the level of a pattern but were interesting data points that could prove helpful in the future:

- "Why not just have everything (all case information) on the first screen?"
- "Prototype is lacking a lot of flexibility on how to categorize cases"

Customer Segments workshop with the AO's Product Owners

Goals

- Explore how different groups of users are looking for CM/ECF to assist them with different things
- Started creating hypotheses about Jobs, Pains, and Gains of user groups that can be validated through additional research and usability testing, and paired with product ideas using the Value Proposition Canvas.[2]

Outcomes

This activity was initiated by sharing and discussing user groups, also called Customer Segments, who are users of a product with similar goals, and problems that need to be addressed through feature development. The Customer Segment of a Clerk developed from insights gathered during our earlier rounds of research. This served as a basis on which the AO's product owners then drafted an initial Customer Segment for non-Judiciary CM/ECF users. In a Customer Segment, "Gains" represent positive outcomes, benefits, or aspirations; "Pains" represent negative impacts, risks or friction towards a goal; and "Customer Jobs" represent tasks, goals, or things they need to get, which can range from trivial or very important.

[2] The Value Proposition Canvas is a framework to visualize the fit between the users (or customers) of a product and the value a product delivers to them.

Attorney Workshop

Goals

- Learn from members of the public about their experience gathering case information or filing on their clients' behalf.
- Understand in more specific terms about how attorneys' experience with CM/ECF affects their ability to efficiently manage their work.
- Demonstrate to the AO product owners a different way to gather information about the tasks, pains, and goals of their users.

Outcomes

The workshop started with a discussion prompt about the attorneys' current experience using CM/ECF, and they responded that the easiest action is filing straightforward motions or briefs without many documents attached. The attorneys also brainstormed and agreed that the most difficult thing they do with CM/ECF is filing something non-typical that you want to give a different title/type of document than what CM/ECF has in its system.[3] This generated a discussion among the attorneys about various negative experiences they have had trying to file documents and the impact the difficulty of using the current system has on them.

Selecting the right event is difficult because of the large number of events available to choose from and the lack of clarity on what some events are or the difference between certain events. One attorney said, "The ramifications of mislabeling an event is that is an emergency: the docket might not get to the right place, that the clerk's office will reject this….Often not an event when it's an emergency or happening quickly and we don't have time to ask but emergencies are exactly when you need to be able to use the right event." Another suggested "it would be helpful if the template could make a suggestion but could be overridden," while acknowledging "that it's a problem when people put in (the wrong or generic event)." A source of this issue that the attorneys identified was that "courts have a lot of control over the events and you can tell they

[3] CM/ECF for all court types is built around "events" associated with different types of documents a user may wish to submit in a case, or actions the court might take to manage or move the case forward. The list of these events is known within the Judiciary as the CM/ECF "dictionary" and is intended to be exhaustive of anything CM/ECF users may wish to add to a case. District Courts and Bankruptcy Courts' CM/ECF systems include many different starter or standard events, while the Appellate Courts build their own dictionaries; all courts have the tools to create events and to modify or delete the standard events in order to address local needs. This adds significantly to the volume of events in use throughout the Judiciary — which number in the thousands — and the overall complexity of the system.

had specific things in mind for those events but it's not clear what that was and there's no documentation [to guide the attorneys to choose the right event type]." Their understanding of what might have led to this state of affairs was "someone at the court made the events without looking at what other courts are doing or (considering cases) that have to move between courts."

The attorneys also recognized that these challenges have ramifications for judges and court staff. "Appealed records from the lower court filing clutters up the docket with unclear stuff. Some judges don't understand what has been filed." Out of an abundance of caution, some attorneys might file a document under the most generic event category available - though other events might be more accurate - or rely on a phone call with court staff to assist them.

Thinking of what could make their work better in the future, the attorneys expressed a shared desire to be able to check documents after uploading them before they are submitted to the system. They also spoke of a system in the Texas state court system that uses Artificial Intelligence to check that documents are filed correctly, mentioned a similar feature in New York state courts, and suggested that the AO investigate that sort of functionality. The attorneys shared significant concerns with the current system's limitations around restricted access filings: "It's stressful not knowing if I uploaded the sealed one or the public one," and multiple attorneys signaled their agreement with the statement, "It's a nightmare. It can border on malpractice [if I were to accidentally file the wrong version of the document]."

The attorneys also described experiencing CM/ECF error messages indicating the system wouldn't accept a PDF but failed to provide any messages to help them troubleshoot what to do. "There's no guidance from the court on this. I don't know what the solution is because I don't know what the problem is." This seemingly small inefficiency leads to large ramifications for the attorneys and their clients:

- "I've had people in tears in my office because they've been trying for days [to file a document in CM/ECF without success]. Doesn't look good if you have to spend hours on it and when it finally works it's labeled on the docket as untimely. Clients see that!"
- "[The impact of untimely filings on the docket is] wasted time and energy because of a quirk in the system, [and it] doesn't look very good for your practice or the court."
- "[The difficulty in adding PDFs] adds lots of billable hours that shouldn't be there. It's the last thing in the process so to have done all the work on the document and have that last step not work adds a lot of stress."

The workshop attendees appreciated the workshop, viewed it as a good process for sharing their experiences with the AO, and hoped it would be the start of a collaboration between the public and the Judiciary. Comments included general statements about the workshop being a "nice way to get info and give input." The participants also responded positively to the collective nature of using an online whiteboard to facilitate the conversation. One participant stated, "I like knowing other people have the same problems." The workshop closed by asking, "Who else does the CM/ECF team need to hear from?" This question elicited the following responses about additional CM/ECF users that the AO should consider conducting more user research with:

- Federal Defenders, AUSAs, CJA counsel
- Pro se individuals or someone representing them
- Attorneys' staff (who may use CM/ECF more in their work)
- Legal Aid groups
- The press
- General Counsel from corporations
- Diverse practice areas and firm size
- Trustees in Bankruptcy

What do we recommend?

Regularly talk to the people who use CM/ECF

Talking to the people who have their hands on keyboards using CM/ECF to approach the particular business use case in question, provides the development team with insight into the realities of what actually happens when users navigate the technology, what decisions users are making on when and how to accomplish tasks, and what else is going on that might influence users' work process. Talking to people who are seen as knowledgeable experts in the process, who manage those with their hands on the keyboards, or who used to use the software will give the team a view of only what is supposed to happen, how the software was designed to be used (rather than how it is actually used), or what used to happen with the software. Relying on idealized or outdated information when making decisions on what to build, what to prioritize, and how to design specific features devalues the complex and difficult realities faced by the people CM/ECF is supposed to support. It can lead, and has already led, to the AO building things that don't meet CM/ECF user needs at all. We recommend making decisions based on matching features to the product roadmap and vision, and ultimately to the goals and needs of CM/ECF users. Conducting human-centered

design activities regularly with actual CM/ECF users helps to make sure the roadmap is prioritized correctly as new software is built out and allows the development team to have confidence that their work will continue delivering the most important value for the people who use the system even as their needs may evolve over time.

During the E&I Phase, the AO worked to identify court staff who use CM/ECF as part of the record on appeal process and who could participate in the human-centered design activities 18F would lead. At first, it was difficult getting research participants for this to work, as the people we ended up meeting were not always the actual users of CM/ECF's record on appeal functionality but were instead supervisors of the actual users or were other stakeholders who could not participate productively in human-centered design. The AO team ran up against the need to get permission from clerks of court and other supervisors before contacting individual users within a court. To make sure regular research is possible, the Judiciary should encourage and allow direct outreach to CM/ECF users within the courts, without administrative barriers. So that feedback can happen from across the diverse range of roles and responsibilities and perspectives within the court user community, there needs to be an active and ever-evolving source of potential user research participants that includes at least one CM/ECF user per court.

The Prototyping Process

What did we build?

The primary tool 18F used for the usability testing described above during the E&I Phase was a code-based prototype of a new Record on Appeal (ROA) tool. The front end of the prototype is a React application deployed to Federalist, a static website hosting platform built on Cloud.gov.

How did we build it?

The main purpose for the prototype was to put an interface in front of users as a mechanism for soliciting reactions and feedback, as well as, to demonstrate and share a different way to inform product development. The prototype was always intended to be a research tool, and nothing more. Nonetheless, it is worth a brief overview of the technologies we employed in rolling it out, which enabled the team to create an interface for users to react to and that could be adjusted as we learned from CM/ECF users how better to meet their needs.

U.S. Web Design System

The front-end prototype made heavy use of the United States Web Design System (USWDS). USWDS consists of a library of components, standardized recommendations for aesthetic options like colors and typography, and a set of user experience principles validated by usability experts to guide teams on how best to build effective digital products using USWDS tools.

Sites that leverage USWDS have a significant head start on accessibility, cross-browser compatibility, and compliance with 21st Century Integrated Digital Experiences Act (21st Century IDEA) guidelines.[4] Of course, the most salient benefit to using USWDS in the prototyping process is that using its pre-existing open source components enables rapid development.

[4] Irrespective of 21st Century IDEA's applicability to the Judiciary, these guidelines represent established user experience and web accessibility best practices.

React

The prototype is built using React, an open-source web application library developed by Facebook. React is often used to build complex applications using only (or mostly) front-end code, which is to say: applications where business logic is contained primarily in the browser rather than on a server (applications such as Facebook). These kinds of complex, mostly browser-based applications are often referred to as Single-Page Applications (SPAs). The ROA prototype is not an SPA. However, leveraging React even outside of this context provides several benefits:

- It makes it simple to create user interfaces that feel "snappy" to users, meaning they respond quickly to user interactions;
- There is a robust ecosystem of open-source tools built to work well with React, several of which we used in developing the prototype (e.g., Redux, Gatsby);
- It is an industry standard for front-end code, which makes it easier to onboard new developers to the codebase and expands the pool of developers who have experience with the technology.

None of this is to say that React was the best or only choice for building the prototype, but it allowed us to get something up and running and in front of CM/ECF users quickly.

Federalist

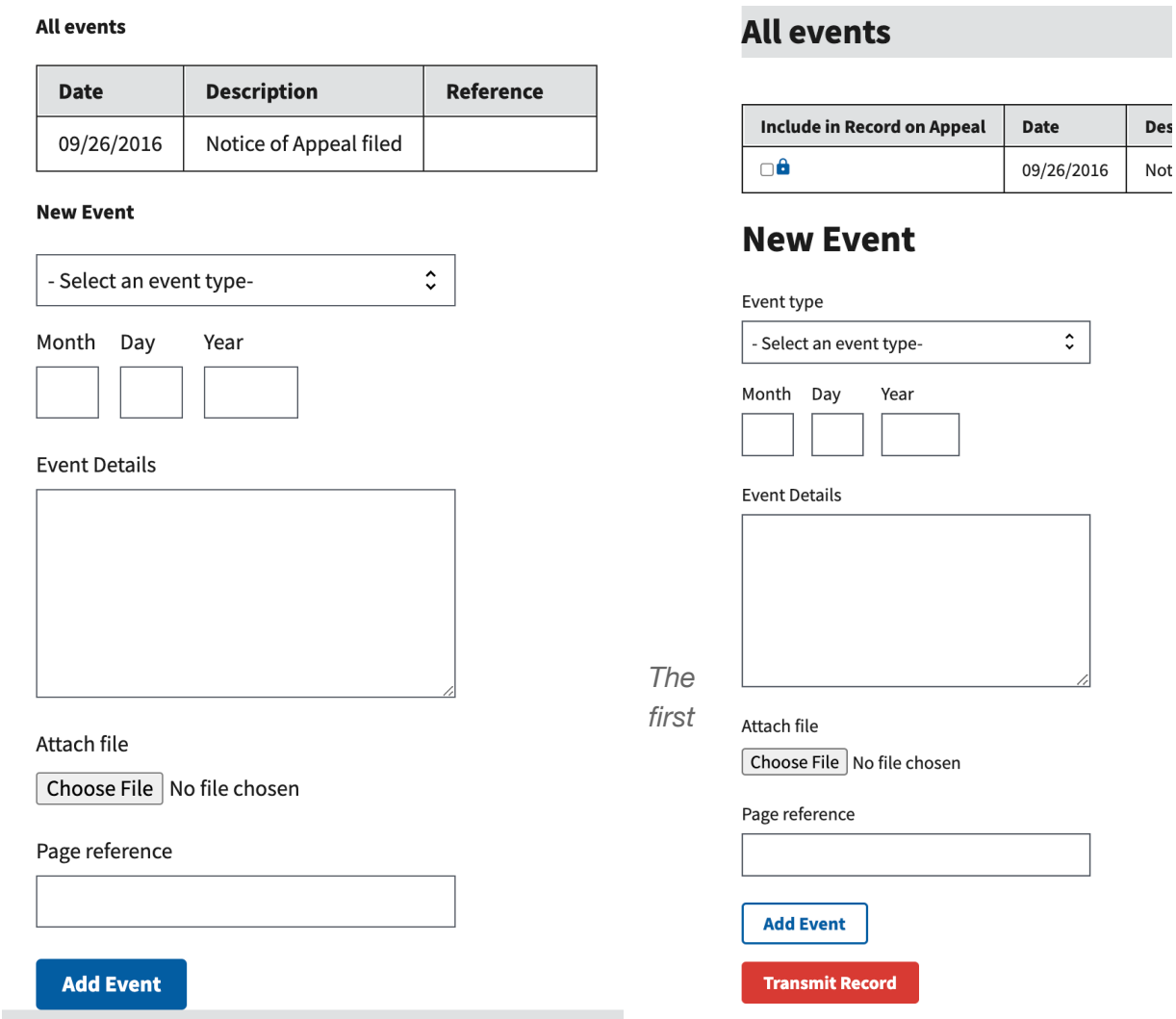
The prototype is deployed to Federalist, a static website hosting provider built on Cloud.gov. Federalist makes it easy to continuously deploy static websites from Github to cloud.gov.

What did we learn from this process?

The first round of research was primarily focused on learning about the record on appeal process and how it is currently implemented in legacy and NextGen CM/ECF. What follows is a description of how user research was used to develop the prototype, which then facilitated subsequent rounds of research, which in turn moved the prototype further forward. This demonstrates the benefits of early, frequent, and hands-on interaction between users and in-progress-but-functional tools.

Docketing and Compiling a Record on Appeal are Two Distinct Processes

Several of the participants in this first round of interviews reported being unable to receive docket information within CM/ECF from lower courts. As such, these individuals needed to recreate the case docket from scratch in their own court's version of CM/ECF.[5] The first version of the prototype addressed this apparent need by giving users the ability to add docket entries at multiple points in the process:



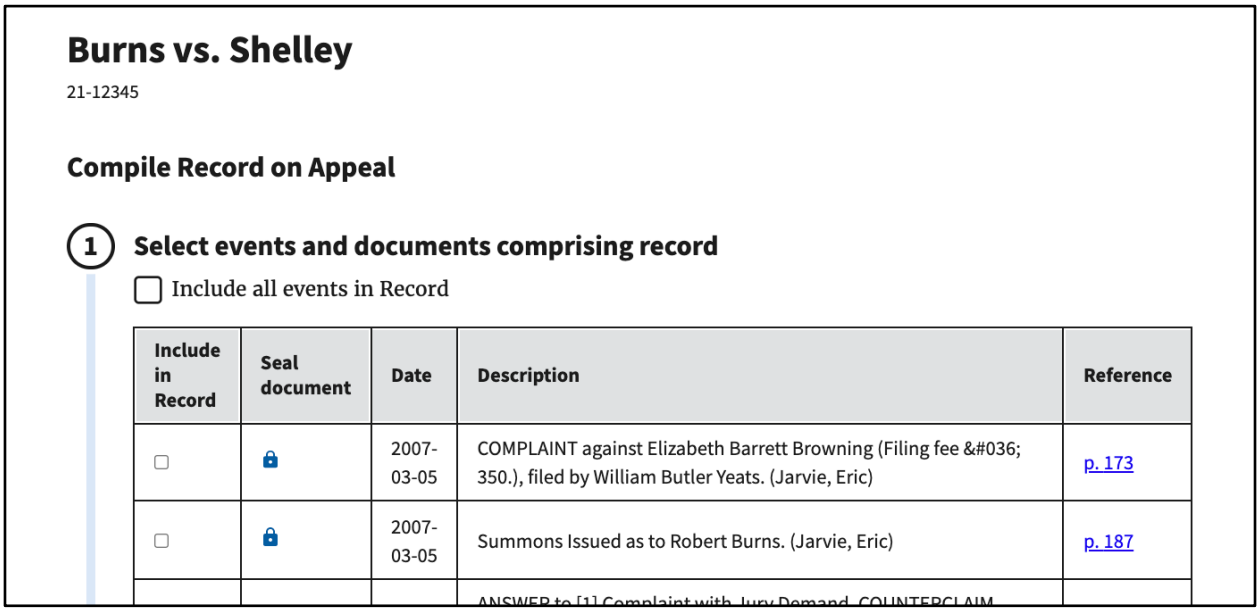
The first iteration of the prototype allowed users to add new

[5] As noted in more detail in our PA report, each court has its own "version" or "instance" of CM/ECF that it can customize and modify locally, meaning that the current system is actually comprised of 200+ versions of CM/ECF.

docket entries on multiple screens (R: "docket entry" screen; L: "Compile record" screen).

The second round of interviews was the first time users actually saw and interacted with the prototype as part of usability testing. Most second round participants were confused by having the ability to add docket entries in multiple places: the "Compile Record" screen was too similar to the "Docket Entry" screen.

To address this, we removed all of the forms that were present on the "Docket entry" page from the "Compile Record" page, and restructured it using a USWDS Process List component:



Burns vs. Shelley

21-12345

Compile Record on Appeal

1 **Select events and documents comprising record**

☐ Include all events in Record

Include in Record	Seal document	Date	Description	Reference
☐	🔒	2007-03-05	COMPLAINT against Elizabeth Barrett Browning (Filing fee $ 350.), filed by William Butler Yeats. (Jarvie, Eric)	p. 173
☐	🔒	2007-03-05	Summons Issued as to Robert Burns. (Jarvie, Eric)	p. 187
			ANSWER to [1] Complaint with Jury Demand. COUNTERCLAIM	

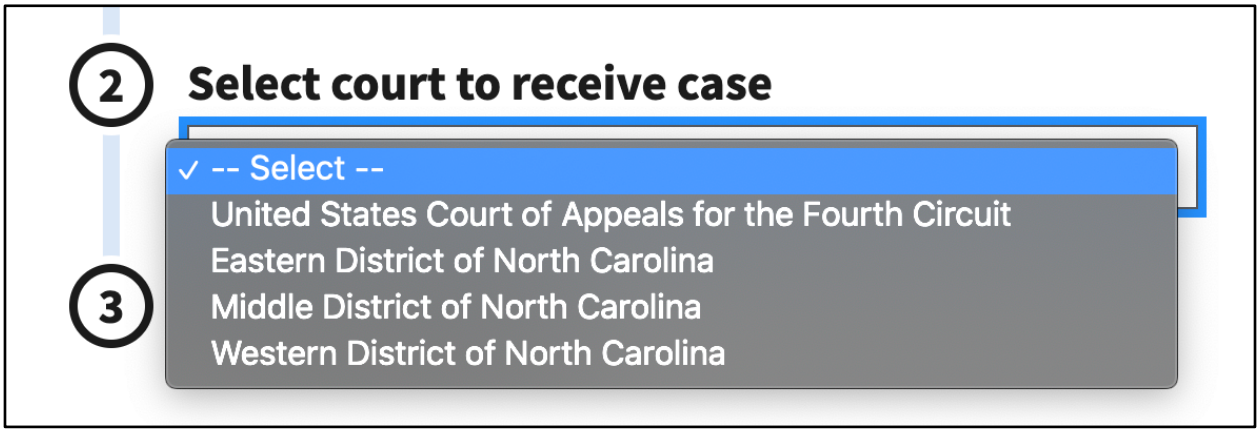
In addition to making the record more visually distinct from the docket, the process list gives the user more guidance on the sequence of steps to compile and transmit a record.

Users need to know where a record is going before hitting "Transmit"

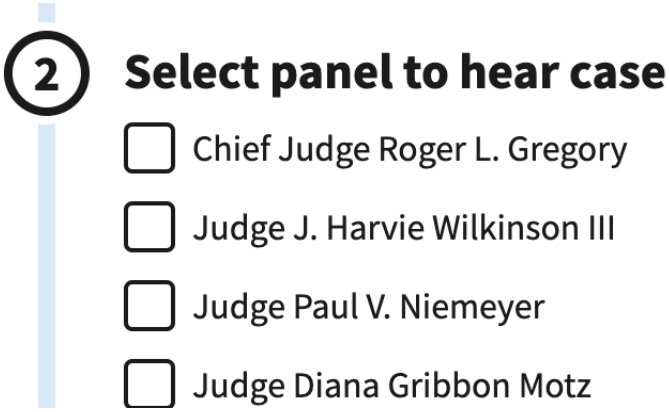
While the big red button giving users the ability to "Transmit [a] Record" was clear enough about the action the button would initiate, second round participants expressed confusion over where the record would go upon transmission. To a judge? A panel? Another court?

We tested two possible solutions to this confusion in our third round interviews. One scenario involved a bankruptcy or district court clerk sending the case on to another

court for appellate review. In this instance users had the option to select a court before transmitting the appeal:



We presented another scenario in which we asked users to imagine they were a clerk in the circuit court receiving the appeal, to see whether selecting a judge or panel of judges would be viable:



In this instance, design choices ran up against court processes. The most common user response to the "panel picker" component was that panels are not typically assigned until further along in the appellate process (such as after briefing is complete), so sending the case on to a judge or panel directly after receiving and reviewing the record would be quite unlikely. Users in that scenario would prefer just to review that the ROA was complete and would take further action later.

The legibility of the process depends on the format of the data

One significant change to the prototype between the second and third rounds of interviews was integration with the new backend API we had been developing. The user-facing impact of this addition was that the prototype could incorporate AO-

provided dummy data through the API. Filling in the docket view with genuine(-seeming) data made the prototype more realistic. Second round participants indicated that the lack of pre-populated docket events caused some confusion, so incorporating the data from the API in the third round helped to make the docket table more legible to users as a docket.

Other opportunities that could be explored

Tags to distinguish different filing types

One user in the third round of research reacted to the fact that the docket text for events involving filings styled the type of filing in all capital letters, something the user was not accustomed to seeing. As it happens, this was just a quirk of the test data 18F got from the AO. Presumably in the court in which this CM/ECF user worked, such capitalization was not standard practice, but this user responded to it positively. This suggests that a possible future improvement to displaying docket information might be to make events even more visually distinguishable. For instance, putting the filing or event type inside a USWDS Tag component:

☐	🔒	2007-03-05	ANSWER ANSWER to [1] Complaint with Jury Demand, COUNTERCLAIM against William Butler Yeats by Robert Burns. (Attachments: # (1) Affidavit # (2) Affidavit # (3) Affidavit) (Keats, John)	p. 23
☐	🔒	2007-03-05	MOTION MOTION to Appoint Expert, MOTION to Continue, MOTION for Disclosure, MOTION for Discovery by Emily Dickinson. Motions referred to Travis F Ten. (Shakespeare, William)	p. 11

Going further and color coding the tags may provide even greater benefit:

ANSWER ANSWER to [1] Complaint with Jury Demand, COUNTERCLAIM against William Butler Yeats by Robert Burns. (Attachments: # (1) Affidavit # (2) Affidavit # (3) Affidavit) (Keats, John)
MOTION MOTION to Appoint Expert, MOTION to Continue, MOTION for Disclosure, MOTION for Discovery by Emily Dickinson. Motions referred to Travis F Ten. (Shakespeare, William)

We did not show this tag concept to users, but it is an avenue that could prove fruitful beyond the record on appeal context and to docketing or displaying documents more broadly (subject to validation by further user research, of course).

Alert banners may be a helpful way to notify users of new electronic filings

One concept that we employed in every round of research was using an alert banner to 1) indicate to users that a new Notice of Appeal had been filed; and 2) to provide easy access to the relevant case:

Responses to this were generally positive. Some users, however, have their own routines and workflows that let them view incoming cases when and how they need to, and would find alerts an unwelcome interruption: “I wouldn't want alerts because then I'd feel like I’d have to do something right away. It would be disruptive.”

Ultimately, the purpose of the prototype was to prompt conversation. Putting the prototype in front of daily users of CM/ECF helped us learn more not just about how the current system works and how it may be improved, but also more generally about how courts operate as well as technical and organizational challenges users face while compiling records on appeal. Much of this is reflected in the user stories described in the preceding section. Further, some of the specific user interface concepts we tested here may work in many other contexts beyond the particular use case.

API Development

In our PA report, we recommended that the Judiciary leverage APIs when building a new CM/ECF. During the E&I Phase, we set out to test the feasibility of an API-driven CM/ECF. The results of our hypothesis-testing are summarized in this section.

What did we build?

We built a prototype back-end API service providing access to a small sample dataset stored in a relational database. This gave consumers of the API access to dummy data about cases, records on appeal, courts, and users. It also allowed users of the API to perform actions against the data such as creating records of appeal from cases. The data and functionality provided by the API followed the user research and design work of the front-end prototype, but in practice other types of systems, such as reporting applications, could use the same API.

How did we build it?

The API provided both a REST API and a GraphQL API. It was a Python application built on top of industry-standard libraries like fastAPI, SQLAlchemy, and Ariadne. Data was stored in a PostgreSQL relational database. The application automatically ran tests and deployed to cloud.gov, which, as discussed above, is a platform-as-a-service (PaaS) designed for government projects.

GraphQL for the API layer

We explored using an API layer as a centralized source of truth for our sample data, both as a means to access the data as well as to manipulate it. The diverse ways different courts access and use data in their day-to-day work is a major challenge for the AO. Different courts have different conventions and rules for processing cases making it difficult to design a single software solution for all courts to manage their cases. 18F's Path Analysis recommended using an API layer as a central tool to access data while still allowing courts autonomy over how this data is presented and used.

REST APIs define resources as HTTP endpoints and associate them with HTTP verbs such as GET, POST, and PUT to define actions. The convention of the REST API is battle-tested and well-understood. However, REST APIs present some difficulties:

- They are difficult to change once clients start using them, leading to simultaneously maintaining multiple versions of the API.
- It is usually not feasible to dynamically define what data is returned from the API, leaving clients with either more data than they need because that's what the endpoint returns, or requiring them to make multiple requests for related data.

GraphQL is a different API paradigm designed specifically to address issues of efficiency and flexibility. It defines a Schema that clients can use to craft queries specific to their use case. On the back-end, supporting the flexibility of GraphQL requires more work, but it provides a much richer interface for users of the API.

Using Python on the server

To support this phase of work, we wanted to build a small API that was easy to maintain, easy to deploy, and flexible. Naturally, one of the first decisions we made was the choice of programming language. For this phase of work we needed a language that was:

- supported by cloud services without a lot of custom work;
- well-understood by people on the team with easily-available documentation;
- mature enough to have all the tools and libraries we needed; and
- efficient with an adequate concurrency model for designing an API.

Python[6] was a natural choice for us. It is well understood by members of the team and is a common choice for projects of all sizes in both government and the private sector. It offers a multi-paradigm programming model so developers can freely mix object-oriented and functional styles, where appropriate, allowing developers a great deal of flexibility to build clean, easily readable code even when projects become large and complex.

Python is actively developed and continues to evolve, but the core of the language is stable and well-documented. It is also developed in an open-source, well-supported, and inclusive ecosystem. It also has mature library support for the domain in which we are working. This includes database libraries like SQLAlchemy, a variety of high-

[6] In our PA report, we pointed to CM/ECF's use of Perl as problematic because it is a dynamically-typed scripting language. Python is also dynamically typed, but includes support for optional typing can reduce some of these problems. It is also more appropriate for and frequently used in large projects where perl is problematic.

performance API and testing libraries, and good support for GraphQL. Strong library support allows developers to focus on the issues specific to the Judiciary's domain and deliver value more quickly.

Python is also fully supported and used frequently on cloud providers including Cloud.gov, AWS, and Google Cloud Services.

Deploying to Cloud.gov

To model a realistic DevOps value stream, we needed to deploy our work to a production-like environment. We agreed that the code should be continuously deployable in a simple, dependable way. There are many ways to do this including using scripts or infrastructure-as-code (IaC) tools such as Terraform with a cloud provider's APIs. For this phase of work, our primary infrastructure requirements were a relational database service and a virtual server capable of executing Python code. Additionally, we required a means to deploy to this service in a repeatable, automated way.

Cloud.gov provides all of the services this API currently needs. Additionally, Cloud.gov includes a simple and reliable method to deploy new instances, which works well with continuous integration/delivery tools like CircleCI or Github Actions. Moreover, Cloud.gov already has a Provisional Authority to Operate (P-ATO).

While Cloud.gov runs on top of AWS services and provides many services, including Elasticsearch, Redis, and S3, it does not provide everything available on AWS out-of-the-box. It is conceivable that future phases of this project may benefit from services such as Lambda functions, SQS queues, or SNS notifications, which are not natively available on Cloud.gov. We should note that hybrid models are possible where core services run on cloud.gov and coordinate with additional services from a provider like AWS.

DevSecOps

What DevSecOps is

DevSecOps is a portmanteau, a made-up word coined from a combination of the words **Dev**elopment, **Sec**urity, and **Op**erations. Unlike the Agile movement, there is no

manifesto for DevSecOps, which makes it difficult to point to a single, authoritative definition. It also means that reasonable people disagree about its exact definition.

DevSecOps is a culture, team structure, and set of tools and processes to support the culture and team. It begins with the understanding that there are structural problems with the way government entities and other large organizations have traditionally delivered software:

- Developers, Operations, and Security groups are in different parts of the organizational structure and are subject to conflicting priorities.
- Communication among these groups is slow and difficult.
- None of these groups has contextual awareness of the work done by the others.
- There is a mismatch in values: developers are driven by pressure to deliver quickly, while operations staff are driven by pressure to maintain stability.
- Developers, who are closest to the work, don't feel responsible for security or operations because that's someone else's job.
- Slow communications means backlogs of work accumulate leading to even slower communications and more complex deployments.

DevOps, a precursor to DevSecOps, emerged from the lean manufacturing movement in the 1980s which focused on two main tenets:

- Lead time in manufacturing (i.e., the time from converting raw materials into finished products) is one of the best predictors of customer satisfaction and quality.
- One of the best predictors of short lead times is small batch size.

The DevSecOps movement proposes addressing the problems with delivering software by forming small, cross-functional teams who work closely together to write, secure, and deploy code in small batches as a team. Because the team has development, operations, and security skills, they are empowered to work autonomously and can deliver quickly. This reduces the lead time between concept and production, which in turn creates a positive feedback cycle where the team can learn and adjust quickly to information from users which continuously improves both the product and the team. DevSecOps thrives in a high-trust, cooperative culture with open communication. A strong DevSecOps culture is one where risks are shared, silos are broken down, and mistakes are met with inquiry. This requires strong support from leadership, who must understand that DevSecOps is an institutional cultural shift, not a product or service that can be purchased from a vendor.

What DevSecOps is not

Because DevSecOps has become a buzzword, and is often used in job announcements and advertising, it is often over simplified.
DevSecOps is not:

- Automation: automation often plays a role, but automation alone is not DevSecOps.
- A tool: many tools are used to support DevSecOps, but if someone tells you DevSecOps is a tool, they are probably a vendor trying to sell you something.
- A job description: positions like "DevOps Engineer" are common, but DevSecOps is more appropriately described as a team of people within a supportive organization.

How this relates to our E&I Phase work

During this phase of work, we demonstrated many of the tools and processes associated with DevSecOps.

Continuous Integration / Continuous Delivery (CI/CD)

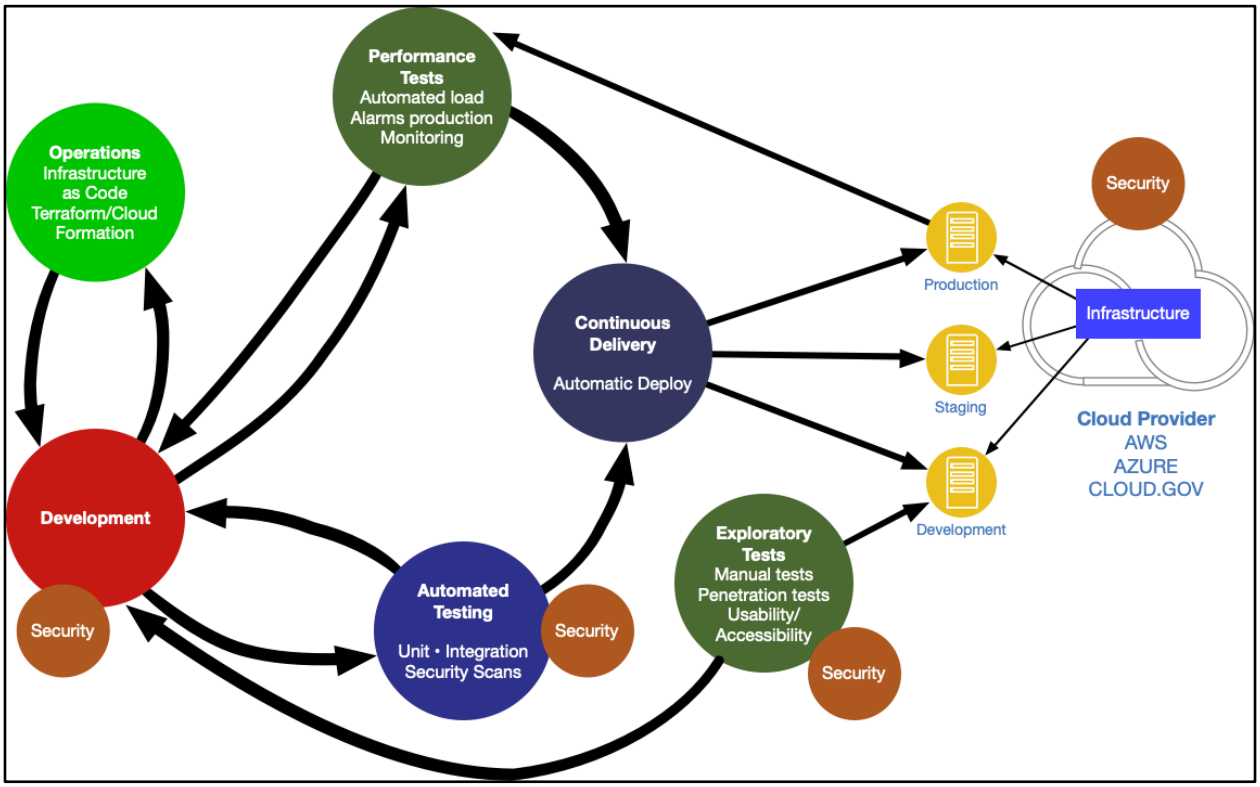
As we explained in our PA report, continuous integration is a process that allows developers to quickly merge code they have written into the main trunk of the project. This is supported by automated tests, peer-reviews, and the ability to quickly stand-up production-like environments for tests. By encouraging developers to contribute small changes frequently, rather than large changes infrequently, we avoid large merge conflicts, reduce the risk that new code introduces bugs, ease the burden on reviewers, and enable quick roll backs if changes cause problems.

Continuous delivery is the technique of frequently pushing new code into a production-like environment. Developers who write code are responsible for ensuring that the code passes tests and can successfully run in production. For the developers, the "definition-of-done" includes successful deployment of their work. This is a significant change from a process where developers write code and then hand it off to a different team to deploy.

This project modelled both continuous integration and delivery. Changes made to the main branch were subject to automated security scans and tests. Once these tests passed, scripts automatically deployed the code to Cloud.gov. The lead-time from committing code to seeing changes in production was approximately *three minutes*. This time included provisioning a new server with each deployment (a common

practice in CI/CD pipelines). In a larger project this lead time would be longer due to larger test suites, but it is still common for huge projects to push changes to production *many times per day*.

This quick CI/CD cycle has enormous benefits. Bugs and security issues can be addressed immediately without waiting for the next release. Users benefit from new features as they are developed. It creates fast feedback loops and small change sets. If any problems occur during this cycle it is a quick matter to rollback changes or move forward with a fix. The developer who wrote the code and is most likely to understand the issue is responsible for its deployability and will likely be the first person to address any problems their changes introduced. This is a stark contrast to problems that occur when an operations engineer has difficulty deploying code written by someone in a different department weeks or even months ago.



A cross-functional DevSecOps team is responsible for the entire stream of work from concept to production. In the graphic above the arrows exiting the development team represent actions they take such as provisioning infrastructure, deploying functionality, and testing work. Most of these actions are automated. Incoming arrows represent feedback from these processes including metrics and errors. Groups outside the core

team may perform their own exploratory tests such as penetration, accessibility, and usability tests. This provides an additional source of feedback for the development team. Security is considered at every step and is integrated into the work habits of everyone on the team.

Infrastructure as Code

Continuous delivery often leverages infrastructure-as-code tools. Instead of depending on operations engineers to stand up servers with the correct operating system and all the proper dependencies, which often takes weeks, developers define all of the necessary parameters for the infrastructure in code and configuration files that are saved in the code repository alongside the application code. This allows:

- Dependable reproduction of production environments;
- Easier horizontal scaling when needed;
- Testing against the same environment that is used in production;
- Developers to make necessary changes to environment and dependencies; and
- Tracking and auditing of all changes to environments.

Infrastructure as code is one of the pillars of DevSecOps. It reduces lead time by removing handoffs between developers and operations teams and increases reliability by clearly defining everything needed for the environment in a centralized source of truth.

For this phase of the project, defining the infrastructure as code was simplified by using Cloud.gov, which provides much of the infrastructure out-of-the-box. This enabled us to simply define software dependencies and basic language requirements needed for the project. This is a major advantage to using platforms-as-a-service like Cloud.gov.

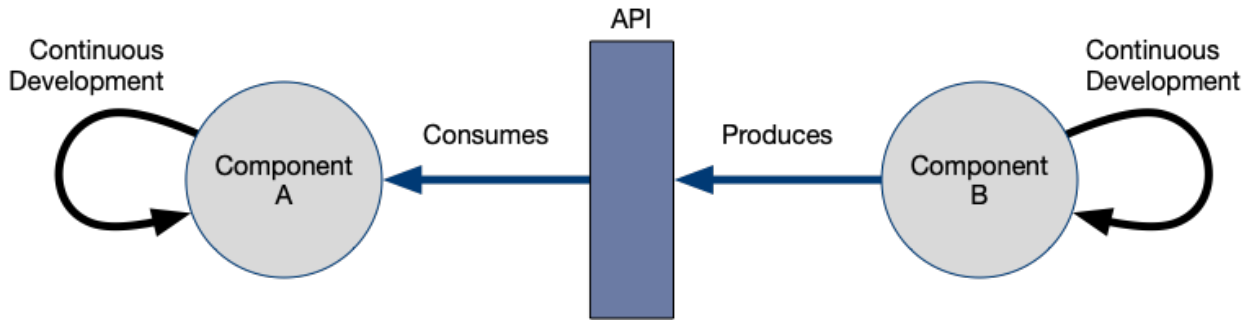
Building Loosely-Coupled Components

A common challenge in modernization projects is that legacy components are tightly-coupled and often don't have adequate tests. This makes change risky and stressful. A small change in one part of the application can have unexpected consequences in another. This is a form of technical debt often caused by pushing forward quickly with feature development without dedicating time to refactoring or by working around known problems to meet deadlines. When teams work on different, tightly-coupled components they need to coordinate carefully to avoid breaking things. This can introduce unexpected bugs, dramatically slow down development, and even lead to conflict among teams.

Loosely-coupled architecture organizes projects around modules that are implemented independently. They maintain minimal dependencies with, and well-defined boundaries between, each other. Developers typically agree on a set of interfaces through which components can interact without the need for developers or their code to understand the implementation details in other modules. One way of doing this is an application programming interface or API.

All people understand and use interfaces in daily life. To drive a car you only need to know about the brake pedal, steering wheel, and accelerator. You don't need to know anything about fuel injectors or spark plugs. Replacing components under the hood will have little effect on the way you drive the car because the interface — the steering wheel, brake pedal, and accelerator — don't change and that's the only part you, the driver, need to know about.

In software, decoupling modules allows teams to work in parallel without breaking each other's work. In the E&I Phase, we demonstrated this by designing two loosely coupled components: a front-end user interface (UI) and a back-end data layer. These modules interacted via a GraphQL API but were otherwise independent — each team could freely make changes to the implementation of their module without breaking the other's, so long as the interface remained consistent. This increased the maintainability of the system and allowed multiple streams of work.



Components A & B are loosely coupled via an API. They have no direct connection to each other, allowing developers to work on either without concerns about impacting the other. This helps future developers too because changes don't unpredictably impact other modules they may not be familiar with.

Loosely-coupled architecture, along with dependable tests, allows small agile teams to integrate quickly. In this phase of work, two developers worked on different parts of the project without the need to coordinate the details of their work beyond agreeing on an

interface: the API. Each team was free to choose appropriate tools for their particular task.

Improving delivery of digital services

The PA report included several significant recommendations regarding cultural and organizational changes that will allow the Judiciary to deliver better digital services faster. The preceding sections of this report discussed how 18F worked with a team from the AO to prototype a DevSecOps workflow to support the development of the Record on Appeal API during the E&I Phase. As the PA report suggested, and as discussed again above, DevSecOps is a culture—not just a team, a tool, or a product that can be bought—and there is a great deal of work yet to be done to create such a culture in the Judiciary. Nevertheless, our partners in the AO are taking some important initial steps toward that goal.

What did we do during the E&I Phase?

Begin piloting a small DevSecOps team within the judiciary

Two specific recommendations from the PA report were for the Judiciary to:

- Test the DevSecOps approach with a small team to learn about the specific challenges of the AO context; and

- Encourage open and frequent communication among stakeholders.

During the E&I Phase, the AO assembled a small team of its engineers, architects, and security specialists to work with 18F toward implementing the first of these recommendations. This team included representation from the Departments of Program Services (DPS) and Technology Services (DTS), and was meant to pilot new lines of communication.

To address the second of the above recommendations, the AO's DevSecOps pilot team, along with an AO product owner, participated in all agile ceremonies with the 18F team, such as standup meetings three times per week and biweekly sprint reviews and sprint planning meetings.

While 18F and the pilot DevSecOps team were only able to take preliminary steps together in this brief engagement, we will continue to build on the results of this initial collaboration.

Build product owner capacity

Of course there is more to delivering digital services than writing and deploying software. A strong vision for the direction of the product that has support from leadership and a product roadmap informed by direct, frequent communication with users are essential. Product owners need to be bought into the product vision, have the skills to turn user feedback into a product roadmap and user stories, and be empowered to ensure those stories make their way into the product.

To help the AO's product owners effectively advocate on behalf of CM/ECF users, we recommended in our PA report:

- Building a strong product thinking capability to build the most valuable software for CM/ECF users; and
- Creating opportunities to add product ownership best practices to CM/ECF.

In service of these recommendations, the 18F team worked to develop product ownership capacity by:

- Having existing product owners sit in on user-research sessions so 18F could model the practices of contextual inquiry, prototype usability testing, and user feedback workshops;
- Involving the product owners in debrief sessions after each interview; and
- Conducting numerous workshops with the AO's team of product owners to discuss defining user groups and delivering value to users of CM/ECF.

What did we learn?

Releases are too infrequent and the number of story points available in each release allotted for new user-facing features is insufficient

There is an annual release of new CM/ECF software, with semi-annual updates or "hot fixes" in between. The hot fixes start from urgent bug fixes and then other priorities are added. At the AO, the teams with developer and security resources push things into

the releases by prioritizing them, initially, and only then assign development resources to functional user needs. In this process, the product owners—who as discussed above are currently responsible for representing CM/ECF user needs—are given only limited opportunities for feature development in each release. They work with Expert Panels to determine how best to use these limited opportunities, prioritizing work to return value to the courts as best they can. We learned that, depending on the particular development team or development cycle, the portion of CM/ECF development work devoted to meeting court user needs is often less than 20% and is sometimes as little as 10%, resulting in little overall product enhancement.

Often, changes to the bankruptcy and district CM/ECF systems happen through Modification Requests (MRs) that are submitted primarily by court staff, commented on by others in the court community, and then potentially prioritized for inclusion in an upcoming release. These MRs are used to request corrections for software defects or enhancements to the system. Currently, there are over 2000 entries in the MR database, indicating a substantial backlog of work. Through the MR submission process, court staff can also create and share snippets of code, locally developed modifications, or other work arounds intended to correct bugs or gain additional functionality.

The current process is not agile. It is too dependent on governance boards, processes, and relationships; and is not centered on an empowered product owner with the understanding of actual user needs. The lengthy release cycles, and the need to spend the majority of time on bugs / fixes, is not allowing for continuous iteration and release of code in service of those user needs. There is currently too much technical debt and complexity associated with CM/ECF to create an environment where truly agile software development is feasible.

What gets included in releases is based on influence, not based on product vision

Decisions about what to build in CM/ECF are in many cases made on an ad hoc basis and are based on the demands of stakeholders with access to resources and authority. The voices who are the loudest and carry the most weight (AO leadership, judges, and CM/ECF developers) have an outsized impact on what should be included in each release and their thoughts come before the product determination of the product owner. There is a lack of consensus of what the product vision of CM/ECF should include and there is no one shepherding the system with a clear understanding of the people who use the system, the value it gives to their work, and the importance of one

feature over another to create value for CM/ECF users. In one instance, we learned of a feature that was created to support all courts of a specific type but the feature has not been adopted by any of those courts. Nevertheless, development resources are still being used to support and maintain the feature several years later.

Input regarding new features and their value does not always come from daily users of the product

The primary ways that product owners understand the needs of users of CM/ECF are by consulting with Expert Panel members, reviewing Modification Requests, and informal conversations with subject matter experts and a selection of court staff. Expert Panels typically consist of business analysts, court clerks, and judges—not daily "hands on keyboard" users of the particular CM/ECF tools and use cases that are often at issue in the development process. Product owners describe Expert Panel conversations that include judges speaking to their perception of the needs of the clerk's office staff and clerks speaking to their perception of the needs of the public, which are second hand impressions of what might be happening. This requirements gathering process puts the technology, stakeholder, and business needs before the needs of the users, leading to a backlog of ideas of what to include in the system that will never be complete and are hard to prioritize.

Expert Panels are not seen as a satisfactory process even to those who sit on them. As one member explained, "[the AO] send[s] an email asking for volunteers who will then sit on the panel for 2 years, and [the members selected] don't even use the module; where's the value in that?" Others have described being on the panels as, "if we were in the same room, there might have been blood on the floor," and "sometimes we just spin."

The AO's product owners do not have opportunities to get insights from daily users of particular CM/ECF tools and use cases to form hypotheses on what functionality the application should support. The AO's product owners tend to support clerks' requests for additional functionality because clerks are the CM/ECF users who are easiest to get feedback from whose needs are most relatable to the perspectives of the product owners team, and who, thus, are the people that product owners have top of mind as the beneficiaries of the product.. Because there are no existing channels to get feedback from the public who use CM/ECF, there is a lack of understanding of their problems, which leads some to view public users as inputs into the system, instead of

CM/ECF users. During the AO/18F debrief after the Attorney Workshop, a discussion of CM/ECF product scope included questions like, “what is the responsibility for CM/ECF to support attorneys filing correctly?” The current, general process for determining what features of CM/ECF will be included in each release is for product owners to work in collaboration with business analysts to bring a list of “must have and want to have” features. Ultimately, the Expert Panels vote to confirm feature priorities. As one product owner said of the process, “whatever we do, it’s a balancing act.”

Product owners expressed some frustration with their current tools for understanding what users of CM/ECF need. There are sandbox accounts for courts to test out new features but product owners acknowledge they are not getting much use. According to the product owners, courts have indicated they don’t have the time to use them or they don’t find much utility in using them without the local court data or custom events that would make it more real for them. Engagement with Expert Panel members during meetings are also not the robust discussions the product owners would like. Sitting in on one of these sessions, we observed that most of the talking was done by the product owner who was reviewing modification requests and various options for what they could change. One of the issues with this approach is that the panel members were given the option to vote on prescribed solutions, instead of prioritizing problems and empowering the product owner and development team to decide the best solution to fix the problem.

As long as product owners are primarily able to get input from Expert Panels and informal channels rather than being positioned to learn from regular users of CM/ECF with particular expertise—be they court staff or public users— CM/ECF functionality will not be aligned with the needs of the people who use it.

Feature ideas are modeled on ideas of what users might use instead of starting from curiosity about users’ needs and perspectives

The range of issues CM/ECF users face and the impact on them when the product fails are not well understood. Product owners frame user stories around understanding if the users would use a specific solution or seeing whether they can create something that improves upon existing features. This approach might help a user accomplish a task, but it also might not support the way that person currently works. The mindset of “what can we build” instead of “what do they need, or what would best support how they work” can end up with pre-existing ideas of how to approach solving users' problems. This runs the risk of using resources to create something that does not meet

the needs of a CM/ECF user well or might miss insights that could lead to a simpler or more intuitive way to meet people where they are with their process, work, or goals.

The existing path a feature takes to production passes through several silos within the AO.

One of our most significant findings during the PA Phase was that the AO departments and offices responsible for developing, securing, and operating CM/ECF are incredibly siloed. This diagram of the AO engineering teams that build, harden, and ship CM/ECF shows the extent of the siloing that currently exists among these teams:

Infrastructure (DTS)	Operations (DTS/CMSO)	Development (CMSO)	Testing (CMSO/TSD)	Security (CMSO/ITSO)	Performance Tests (TSD)	Release Management (RMB)
• Build data centers • Build servers for development teams • Operate data center & servers	• Works with infrastructure team to deliver servers and necessary services (OS, Java, etc.) • Create and operates CM and code pipeline capability	• Uses provided systems to build, test, and deliver software	• Validates completed software for accuracy and performance	• Runs security scans and penetration tests to ensure software is free of possible exploits	• Validates performance SLAs	• Creates installable software packages from system provided by development team

The colors for each silo correspond to the nodes on the diagram in the preceding section illustrating the division of responsibilities on a cross-functional DevSecOps team

This diagram demonstrates a standard functional team design where specialized teams are organized by function. For example, infrastructure is separate from, and in a different institutional hierarchy, from security. This has business advantages, such as cost savings through division of labor and sharing of expertise, but it is one of the major causes of long lead times in DevSecOps cycles. Additionally, people in one silo often have little visibility or context into the work happening in other silos. This increases risk, slows work, and reduces empathy between teams.

In fact, in a workshop discussion with the DevSecOps team, we learned that these silos could be compartmentalized even further. The Security silo reflects the work of two groups, CMSO and ITSO. CMSO performs the security checks while ITSO ensures that CMSO is performing the work it has laid out.

This is an antiquated way of building software, and much efficiency, both in reduced rework / bug fixes and a quicker software release cadence, could be realized by bringing these disparate groups together into cross-functional, focused development teams and continuing to build a Judiciary-wide DevSecOps culture.

The bottlenecks these silos create cause extreme delays in shipping new releases

During the PA Phase, we learned that the time between approval and release for a new feature can be over a year and a half. This is due in large part to the siloing just described. During the E&I Phase, we learned from discussions with the AO DevSecOps team that releases are further delayed by additional friction in handing off work between silos. As one engineer noted: “the handoffs from one [group] to the next end up taking a lot of time/logistics.”

Delays in shipping lead to larger change sets in each deployment which further slows the releases and increases the risk. As deployments get more difficult, there is pressure to do them less frequently, which in turn makes them more difficult still. This is a downward spiral.

Some of the work currently happening inside each of these silos can be automated

Even within silos, there are avoidable delays in shipping releases, owing to processes that could be partially or entirely automated. One such delay is in provisioning servers for releases. Provisioning servers is the process of installing the operating system, and whatever other software dependencies are required in order to run the application, and configuring it to integrate into the network software being deployed. As an example, provisioning a server for a small Python web application would involve installing a Linux operating system, the Python language, and dependencies such as a database library, providing a network address, and configuring the network to allow access to and from the server. When automated as part of a deployment pipeline, this can be completed in minutes. Of course, provisioning servers for an annual release of CM/ECF involves considerably more moving parts, such as firewall rules, security scans, and creating user accounts on the server. These steps can also be automated. Recently, however, this provisioning process for the next release of CM/ECF took about 2.5 months. By automating operations, such as provisioning servers, and adding

automated testing of new features, we can empower development teams to take responsibility for quality and deployability of their work. The developers writing the code can take ownership of the deployment scripts and the configuration of services. This gives those closest to the work and with the most contextual knowledge quick feedback when problems happen. Automation also improves the repeatability of operations and testing, which leads to more predictable and safer outcomes while also reducing lead time.

Tension between standard processes and individual court variation make it difficult to determine the right thing to build

When designing the standard Record on Appeal functionality in CM/ECF, the Judiciary intended to implement the Record on Appeal process that is set out in the Federal Rules of Appellate Procedure and the Federal Rules of Bankruptcy Procedure. The actual Record on Appeal processes that we observed courts following during our E&I Phase user research varied significantly and did not always map directly to the Rules. While the outcome of the Record on Appeal process (a selection of trial court documents were compiled into a single document a sent to the court that would be deciding the appeal) that we used as a point of discovery was the same for each court, the way the CM/ECF system was involved in that process varied:

- In some courts, the process was completed all by one person.
- In another court, three paper copies were printed and mailed to the appeals court.
- In at least one court, the process was completed by the lower court and sent via File-Transfer Protocol (FTP), adding another out-of-band communication step to the Record on Appeal process.
- More than one court had the process completed by the appellate court who logged into the lower court's version of CM/ECF, which required the appellate clerk to know the login credentials of ten district courts to be able to complete the task.

Product owners tend to see each court as an island unto itself and express that what works for one court isn't necessarily going to work for another court. "Can't guarantee that so many courts would use any new feature we build."

What do we recommend?

Develop a product vision for CM/ECF and a product roadmap for a portion of functionality

The Judiciary should define what CM/ECF will support for the courts and the public, determine who are the users for whom the system works, define a rubric for prioritizing competing interests among court types and stakeholders, and develop a product roadmap for specific core functionality and value creation for user types. During the PA Phase, 18F and the AO drafted an initial CM/ECF product vision: “A forum, designed with and for users, that supports the administration of justice by facilitating efficient and effective case management and filing.” Whether this or another vision statement is adopted by the AO, it is imperative that AO leadership and other judiciary stakeholders rally behind it.

Frame features from the users’ perspective and test software iteratively with users

More exposure and training on human-centered design would deepen the AO’s appreciation for this discipline in product development.

As the PA report recommended, CM/ECF features should be developed based on user research and understanding of user needs, and the value the feature creates for users of CM/ECF. Acceptance criteria should be included in user stories to help developers understand user success criteria. Frequent usability testing with people who actually use the particular functionality in the system is critical. The notion of “requirements gathering,” and Q/A review to show when a feature is done, in a human-centered design process goes away.

Reevaluate the need for and value of customization of CM/ECF processes

There are valid reasons for local, court-specific processes in CM/ECF, but the need for extensive customization by each court cuts against the efficiency and sustainability of the national system. And some customization might not even be necessary — we

learned from court staff that some unique processes are the result of inertia, keeping the status quo in a court rather than finding the best way. Some say it's not even clear why their court customized the software the way it did.

The AO cannot accommodate each court's unique processes to a virtually unlimited degree and provide an effective, shared system that is responsive to the overall court community's needs. The result of this attempted balancing act leaves everyone unhappy, a backlog of features that will never be developed, and a product that cannot continue to evolve based on user needs or sustained in a cost-effective manner. The Judiciary should seek to determine when desired functionality is substantially similar across court types, when a single offering can be created to reduce complexity and cost, and when customization really is necessary. As part of a product vision and developing a product roadmap, we recommend involving courts in a process of finding common ground with shared processes and making clear that with a certain level of standardization comes a better, faster product that meets more of their needs and can be more responsive to the needs of the overall court community.

Create a transparent process for how changes are made in CM/ECF and the criteria for which new features are developed in each release.

A transparent and consistent process for intaking, prioritizing, and communicating development priority feedback is key. The Judiciary should develop a product roadmap that is transparent to the courts. The development process should include more engagement from CM/ECF users throughout and include getting user feedback on early design mock-ups. The product roadmap should provide everyone with a clear understanding of how and why certain user stories are prioritized for development, and what the priority is for future development based on unmet user needs observed through user research practices.

Empower product owners by giving them the space to develop into a new role

The PA report highlighted product owners as foundational to modern software development in government, as well as the need to develop strong, empowered product ownership capabilities within the organization. The AO's product owners often anticipate resistance to even minor changes from the Expert Panels, the courts, and those with power to influence the product today. "If we change from words in all caps to another text style that would create an outcry." To be effective, product owners need influence over the product direction and decision-making authority on what will

be included in each release. This will require leadership to support product owners to lead development teams in line with the approved product vision.

In addition to product owner empowerment from AO leadership to guide CM/ECF, product owners will need to be given the ability to prioritize work consistently with the established product vision. One product owner reflecting on the ideas of what product ownership at AO could look like said, “we need a whole lot more time to focus on product ownership to do this kind of thing. I’ve said this before. [We have] limited resources, covering extreme breadth.” This will involve publicly empowering the AO’s product owner team with decision-making authority, deprioritizing or eliminating some of their current work activities, investing in training and agile coaching, and changing how their success is reported and measured.

Keep experimenting with models of feedback collection with CM/ECF users to get the engagement necessary to generate insights on future development needs

Product owners would like better techniques for engaging with users of CM/ECF. One product owner expressed it as, “when we think about the makeup of the right people, we want the system users, we need real feedback.”

During the E&I Phase product owners shadowed the contextual inquiry, prototyping process, and feedback workshops to gain experiential learning opportunities for how to uncover, develop hypotheses, and validate the needs of CM/ECF users. Product owners also observed the workshop with attorneys, about which one product owner observed: “Watching them bounce ideas off each other. Nice to know that things are coming up from multiple people.” Product owners also started the process of defining user groups and the tasks they need to accomplish. More user research should be conducted to validate these hypotheses and ensure the right investments are being made in product development.

Help desk data could be another user research data source to explore in the future. Currently, reports can be pulled to identify trends but nothing is being tracked over time on a consistent basis. Help desk tickets could be reviewed for frequent complaints or to signal workarounds that have cropped up in future development.

Continue building capacity around DevSecOps and automation

Measure Performance

Continuous improvement is a key principle in DevSecOps. The best way to know objectively if we are improving and making progress toward our goals is to identify metrics and measure them.

Lead Time

In the context of DevSecOps, lead time is a key performance indicator. This is defined as the length of time between when an idea is identified and when that idea has been deployed to users downstream. Many studies have identified lead time as the best predictor of organization performance and customer satisfaction. Reducing lead time is one of the primary goals of DevSecOps.

Percent complete and accurate (% C/A)

A short lead time is important, but deploying poor code quickly is obviously not a measure of success. Features need to be complete and successfully address user needs. The best way to measure this is to ask downstream users if the feature works as-is and can be used without the need for workarounds.

Automate routine operations

The lifecycle of a software feature falls into to broad segments:

- Design and development: This segment resembles product design. It is varied, difficult to predict, and requires creativity and problem solving.
- Testing and operations: This segment resembles product manufacturing. This phase lends itself to automation, which increases the predictability of releases, reduces variability, and can dramatically reduce lead time. As much as possible, tests, provisioning of infrastructure, and deployments should be automated.

Automating testing and operations promotes a lifecycle where developers spend time where they are most valuable: improving the software. Design and development of features should require time on the order of days to weeks while the time for testing and operations of these features should be on the order of minutes to hours.

Create small autonomous development teams and empower them

Amazon coined the term “two-pizza team” to describe the maximum effective team size as no larger than that which can be fed by two pizzas, which translates to 5-8 people. Teams need to be small enough that communication and administrative friction does not reduce their effectiveness. This allows everyone on the team to understand the scope of the team’s work. They should be autonomous and cross-functional and possess all the skills necessary to perform the work, this includes product owners, developers, and user researchers, and security experts.

Teams should be empowered to own the entire feature lifecycle, from prioritizing work to deploying features. Developers will be responsible for writing secure code that meets the acceptance criteria, passes tests, and is deployable. This requires a culture of trust and openness to allow developers to lead and implement changes and get help when they need it. As the 2021 DevOps report from Puppet[7] states:

[7]https://puppet.com/resources/report/2021-state-of-devops-report/

"If you're a team leader or individual contributor who has done great things within your team and made an impact on teams with whom you closely work, yet you don't have managerial support further up the chain... we have bad news for you. Your organization won't transform, and success will only go so far."

Reduce batch sizes and deploy frequently

Features and bug fixes should be deployed quickly and regularly. By deploying regularly, teams get better and find ways to automate routine tasks. Additionally, organizational trust is built. This leads to:

- Safer deployments: small changes are easier to understand and rollback when things go wrong.
- Faster bug fixes: when changes can be deployed quickly we don't need to wait until the next deployment cycle to fix things. Bug fixes can be deployed immediately.
- Happier users: new features can be rolled out to users more quickly.
- Happier developers: deploying large, complex changes are stressful. When deployments are routine and tests are dependable, developers are confident in developing and deploying code.
- Less technical debt: when change is hard, developers create workarounds that allow them to put-off addressing underlying issues. These workarounds create additional work, which leads to less development time and more technical debt. This can become a downward spiral. Technical debt should routinely be examined by the cross-functional development team and they should be given the time and space to work on "spikes" to remedy underlying issues and increase their productivity and efficiency.

Acquisition

During the E&I Phase, acquisition efforts were intentionally on hold because the agreed-upon focus was to conduct additional user research and explore the AO's technical and cultural environments by using Record on Appeal as a use case. However, acquisition activities still occurred during this phase of our work. We regularly met with the AO's contracting officer's representative (COR) to discuss agile acquisition principles and discuss a potential acquisition strategy for the first increment of the new CM/ECF. The strategy included the recommendations from the path analysis report such as:

- the continued use of the time and materials (T&M) contract type;
- issuing a statement of objectives to define the government's outcomes instead of an overly prescriptive performance work statement; and
- the vendor oversight roles of the AO's team during post award (i.e. vendor management).

We were unable to meet with contracting personnel in AO's Procurement Management Division (PMD) during this phase due largely to the unavailability of PMD personnel during the third and fourth quarters of the fiscal year which are the busiest time for contracting personnel across government. We look forward to partnering with PMD personnel early in the next phase of work to start laying the groundwork for the solicitation.

The next steps for the acquisition of the first increment of the new CM/ECF system will be based on all of the technical knowledge gained from the PA and E&I phases. While the 18F team are not contractors, we share a similar perspective with contractors as an external team working with the AO. We're able to identify areas of confusion, obstacles, roadblocks, and unknowns that a contractor may encounter. We work with our agency partners to streamline processes where we can and incorporate our learning into solicitations so that potential vendors have a clear picture of, not only the requirement, but also the technical landscape they will be entering when performing the contract. This is a key output of the E&I phases - to understand the AO's technical landscape and to convey that information to vendors in order to minimize the amount of time spent encountering blockers and maximize their time for development of the new CM/ECF from the first day of the contract.

Once the following items are completed, 18F can work with the AO's PMD and other AO stakeholders to create a solicitation to build the first increment of the new CM/ECF.

1. Identify the slice of work that a vendor will build as the next increment of CM/ECF
2. Identify the path to production for the vendor to work on the new system
    a. This includes prototyping and conducting user research for the slice of work
3. Refine the CM/ECF product vision and develop user stories based on the user research
4. Move toward a cloud strategy for CM/ECF.

The goal is to complete these steps by the end of December 2021. Things that could extend the timeframe include:

- delays in making decisions on the future slice of work,
- length of time it takes to conduct user interviews to develop user stories relating to that slice of work, and
- the complexity of the path to production at the AO.

Getting to an award

The Record on Appeal was selected as a use case during the E&I Phase to better understand how the AO can incorporate a DevSecOps culture and associated processes (among other things) and to uncover ways to improve the process before a vendor begins building an increment of the new CM/ECF.

Even though the Record on Appeal was identified as a use case, it may not be selected as the first system increment that a vendor will begin building. That decision will be made by the Judiciary with input from 18F. Once it's been decided what the vendor will build, additional prototyping will be performed to identify the vendor's path to production.

The path to production

Often we find in our engagements that government entities think they have a clear path to production. But when the vendor gets started, delivering and deploying code becomes a challenge that creates a lot of wasted time and effort and starts the project on the wrong foot. Identifying the path to production means that a clear road has been laid out for a vendor to deliver working code to the AO's production environment. Obstacles to deploying code in the AO's production environment are identified,

eliminated and/or mitigated. Obstacles can range from bureaucratic, security, policy-based, or institutional culture.

The path to production includes:

- Production data access
- Environments/cloud including licenses
    - Including for vendors, if applicable
- A DevOps pipeline
    - Including CI/CD, automated tests, and security scans
- Security practices that enable iterative releases to production
- Live defect protocol (including triage process)
- Rollout strategies (sometimes called “change management”) that can support iterative releases, including quality assurance, communications to stakeholders, and a user support/helpdesk program

Determining a vendor’s path to production identifies the journey that developed code will go through in order to be introduced into the AO’s production environment. This is not an opportunity to create new processes, policies, or stage gates. Rather, the path to production demonstrates the process a vendor will encounter to deliver working code into the hands of CM/ECF users.

One of the biggest risks to successful product ownership (second only to not having an empowered product owner) is not being able to get software in front of real users early and often so that learning is possible. This is something traditional government IT projects don’t consider until fairly late in the project. We can reduce risk by determining at the beginning of the project how the AO is going to get the software into production and support subsequent agile release cycles before the vendor even writes a single line of code. This de-risks the award by clearing obstacles for the vendor so that they can be productive immediately.

Pre-award

Once the first increment of the new CM/ECF system has been determined and the path to production has been identified, our team can work with the AO to confirm the product vision for the new CM/ECF and the initial set of user stories that will represent the system increment that the vendor will be building. The product vision and user stories will become the scope and requirements of the future contract (respectively) and the AO will be ready to proceed with the procurement process.

The procurement process that we're suggesting is similar to the process the AO is currently doing with a few differences. First, we suggest releasing the draft solicitation as a request for information (RFI) to allow industry an opportunity to provide feedback on the solicitation to increase clarity and understanding of the AO's objectives. In order to reach qualified agile vendors, who may not be traditional government contractors, we typically release RFIs on GitHub (in addition to traditional government avenues) to target a broader audience. After all relevant vendor RFI feedback is incorporated into the solicitation, it will go through procurement review or approval and will then be ready to be released to vendors. We typically recommend a period of no more than 30 days for vendors to provide a response to the solicitation.

Another difference is the use of verbal interviews. Verbal interviews are where we ask questions about the vendor's technical approach and are scheduled with vendors after they've submitted their responses to the solicitation. It's essentially a one-way flow of information from the vendor to the government and does not allow the vendor to revise their responses to the solicitation. Verbal interviews should not be confused with oral presentations discussed in FAR Part 15, which allow the vendor to revise their proposal after oral presentations.

Post-award vendor management

The post award phase of the acquisition is equally important because it requires the AO to be able to verify that the contractor is delivering working code that meets user needs. This verification is performed by a dedicated product owner and technical lead, both of whom should be AO employees. The product owner is part of the development team and helps prioritize team tasks, reports on the team's progress and tasks throughout the organization, and is empowered to say "no" when needed. The technical lead has the technical skills to review the vendor's code and ensure that the code is in accordance with the QASP and the requirements of the contract. The product owner can also serve as the technical lead if they have the technical expertise but oftentimes the two roles are performed by two individuals. It is extremely important that the product owner and technical lead roles are filled before the vendor begins performance. These roles ensure that user needs are being met and that proper, working code is being delivered consistently.

Selecting a cloud service provider

It's too early to decide on a cloud platform at this time but how and when a cloud service provider is selected is something that should be considered before issuing a contract to a vendor to build an increment of the new CM/ECF. Because development of the new system has not started, there are a couple of approaches the AO can consider when selecting cloud services.

The first approach prioritizes the selection of a cloud service provider, which could, in turn, limit the vendor pool to vendors who have experience with the selected cloud service. The second approach prioritizes the skills of the potential vendor team, which, in turn, could limit the cloud service selected based on the skills of the individuals on the vendor team. We would like to explore the pros and cons of these approaches with the AO in the coming months and the AO can select its preferred approach as we move closer to releasing a solicitation for an increment of the new CM/ECF system.

Depending on the approach it selects, the AO may need to look at several cloud service options. Common choices include Amazon Web Services (AWS), Google Cloud, and Microsoft's Azure. Each of these offer similar services (databases, virtual services, etc.), but to be used well, each requires that the software developers have specific skills. Because of this, it's a good idea to ask potential software development vendors if they have cloud recommendations based on the prototype(s), user research, and other information presented in the RFI phase. Those vendors may have developers who specialize in one cloud environment over another, so they will be able to recommend a cloud service based on the skills of their team.

Cloud considerations

Once a potential cloud provider has been identified, it is best that the AO purchase cloud services on their own instead of relying on the vendor to purchase cloud services on the AO's behalf. Cloud services should be a separate contract from the services of the software development vendor.

The two main options for purchasing cloud services are cloud.gov or acquiring cloud services directly from the developer or an authorized reseller.

Cloud.gov is a Platform as a Service (PaaS) that is based on open source Cloud Foundry and built on AWS GovCloud. It was designed as a simple way for agencies to comply with executive branch security compliance requirements. While the AO does

not have to comply with FITARA requirements nor obtain an Authority to Operate (ATO) for its IT systems, there are some benefits to using cloud.gov as a cloud service provider. First, Cloud.gov can be purchased via an inter-agency agreement (IAA) with GSA without needing to go through the procurement process. The IAA timeline for Cloud.gov takes a few short weeks (or days depending on how quickly both parties complete IAA signatures). Additional benefits include:

- Simplicity in deploying and designing infrastructure
- Built-in security best-practices
- Cloud.gov is responsible for keeping infrastructure up-to-date. The AO and developers can focus on their own code.
- Encourages best-practices from developers, such as stateless, disposable servers decoupled from the data layer

A drawback to using cloud.gov is that it only offers a subset of services offered by AWS. If more functionality is needed, the AO should consider using AWS to supplement cloud.gov or use AWS instead.

The AO could also choose to go directly to the developer or an authorized reseller to acquire cloud services. There are benefits to going directly to cloud providers including maximum flexibility and control.

The AO has many factors to consider when selecting a cloud provider. These factors will be examined and explored during the next phase of the E&I to ensure that the cloud decisions that are made will support the new CM/ECF in the most beneficial way.

Conclusion and Recommended Next Steps

The E&I gave 18F an opportunity to get hands-on experience working with the Judiciary to put the recommendations of the Path Analysis into practice. We learned a great deal about the struggles CM/ECF's users have in interacting with the system, as well as those of the people who work every day to meet those users' needs.

In the coming months, 18F will continue to partner with the Judiciary in developing and executing a plan to modernize CM/ECF. The next steps 18F recommends that the AO pursue through this partnership include:

- Identifying a use case for an increment of work that a vendor could begin building
- Refining the CM/ECF product vision and developing user stories based on the user research
- Collaborating with the AO DevSecOps team formed during the E&I to better map out existing processes and to identify concrete ways the Judiciary can adopt DevSecOps culture
- Identifying and documenting the path to production for new software
- Deciding on a cloud approach