Identities Are People, Too: IAM Tooling that Works

IAM Online - December 13, 2017

Mary McKee
Duke Office of Information Technology - Identity Management
The Problem with Oversimplifying

“Everything should be made as simple as possible, but not simpler.”

Albert Einstein
A Post-KISS World

This talk is a rebuttal of that horrible aphorism: Keep It Simple, Stupid.

You hear this too much in UI design, and it isn’t helpful or advisable.

Our tools cannot be simple, because:

• Our work deals with infrastructure
• That infrastructure is necessarily complicated (and very cool)
• Infrastructure exists for humans
• Humans need a working understanding of infrastructure in order to navigate it (and appreciate how cool it is)
• Undercutting user understanding about what is really happening with our infrastructure undermines our goals for developing it in the first place.

There is a better approach.
A KICC in the Pants

Try this on for size: **Keep It Clear, Compadre!***

Simple is no longer available to us.  
We can choose to be clear...  
...or we can choose chaos.

*This is not a thing that people say, but it should be.
Why Clear is Hard

The usual suspects..

**Limited:**

- Funding
- Flexibility on vendor solutions
- Design/UI research resources
- Development resources

**Conflicting:**

- Stakeholder feedback/direction
- Legacy systems and innovative projects
- Competing priorities
- Competing policies

But more than anything…

We often struggle to make things clear to others because we haven’t been able to really make them clear to ourselves.
The Two Keys to Clear

“One should use common words to say uncommon things. “
Arthur Schopenhauer
Key #1: “I Know Kung Fu” / The Case for Mental Models

Mental models are what your service is like and is not like. Like Matrix-style brain downloads, they allow audiences to quickly infer complex relationships. For example:

<table>
<thead>
<tr>
<th>Federated Authentication IS LIKE: Passports</th>
<th>Federated Authentication IS NOT LIKE: The Social Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Federated credentials facilitate access to resources</td>
<td>• “Oh, this is like when I log in to an app with my Facebook account”</td>
</tr>
<tr>
<td>• There is a standard for how credentials look and function</td>
<td>• “I’ll just create a local account; it’ll work the same way anyway”</td>
</tr>
<tr>
<td>• They are issued by different authorities</td>
<td>• “My username – that’s probably my email address”</td>
</tr>
<tr>
<td>• The authority (not individual) vouches for the person</td>
<td>^ Please never use email for IdP discovery. Please.</td>
</tr>
<tr>
<td>• People can have credentials issued by &gt;1 authorities</td>
<td></td>
</tr>
<tr>
<td>• The access conferred by the credential depends on the reputation of the issuing authority and relationship to resource</td>
<td></td>
</tr>
</tbody>
</table>

If you don’t manage your users’ mental models, they will pick their own, and it won’t be pretty.
Key #2: “Get Off My Lawn”/The Case for Service Values

Service values are what your service is and is not. For example:

<table>
<thead>
<tr>
<th><strong>Duke OneLink IS</strong></th>
<th><strong>Duke OneLink IS NOT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>An alternate electronic credential for people who don’t have NetIDs.</td>
<td>Preferable to federation. OneLink is a credential of last resort.</td>
</tr>
<tr>
<td>A great way to accommodate external users into your Shibboleth-protected application</td>
<td>Quick and dirty - OneLink uses real (not throwaway) identities, and the vetting process is not instant.</td>
</tr>
<tr>
<td>A link between an external credential (Facebook/Google/LinkedIn/Yahoo) or standalone username/password and a Duke identity</td>
<td>An alternate identity. Signing up for OneLink will not separate your activity from your NetID.</td>
</tr>
<tr>
<td>Self-sponsored: anyone can have a OneLink credential, even if we wouldn’t give them a NetID.</td>
<td>Service-specific. Services cannot request deactivation of accounts, which are entitled to access other services.</td>
</tr>
<tr>
<td>Robust. OneLink accounts are attached to Duke UniqueIDs that can also support NetIDs, making them great for applicants, alumni and other users in transitioning roles.</td>
<td></td>
</tr>
</tbody>
</table>

**Be Flexible:** sometimes values become outdated and need to be abandoned or adapted.
“Research is what I’m doing when I don’t know what I’m doing.”

Wernher von Braun
IdM + CrUX: A Love Story

Too often, we’re too close to the problem to see it.

If you have access to a UX team, please take advantage of them!

OIT’s CrUX (Creative + UX) team is a neutral 3rd party that conducts all sorts of research for us. We work together on tight iteration cycles that consist of:

1. Producing crude wireframe of intended functionality
2. User testing (n=6) with scripted tasks
3. Collecting and analyzing feedback
4. Updating wireframe with adjustments
5. Continuing user testing and adjusting language and positions of elements on wireframes until tasks are successful
6. Designing/styling final product

CrUX advocates for the user and keeps us honest.
Design Phase: Usability Testing

If you’ve never participated in a usability test, you might be surprised by how scientific they are. Scripts control exercises that measure perception and ability to complete desired tasks.

Usability testing produces insights that back-end metrics don’t, particularly user understanding and experience.

**Participant 2 (grad student, English)**

**Task 1**
- Reads page
- User: “I’m not clear about the relationship between the columns and the buttons at the bottom.”
- Reads page again
- User: “Okay. Sakai the site is trying to get Duke records.”
- Clicks: continue without editing
- Clicks: show other information that will be released
- Reads: The following information will also...
- User: “That’s weird that I had to click on that to see more things because that’s more things than it’s already showed me.”
- Sarah: “What do you think is happening on this page?”
- User: “Duke entity and Sakai as a site is asking for info from Duke’s records to customize site, making it clear that Sakai is a different entity from Duke itself”

**Task 2**
- Click: edit choices
- Click: permit
- Click: save my choices; don’t show me this screen again
- Click: accept and continue
- User: “The actionable elements in the dark box draw your eye there first, but you need to read the header (Sakai is requesting...) in order to understand what’s going on”
Implementation Phase: Design for Metrics

Metrics are forever

Remind your developers that web UIs come and go, but a well-designed data model will endure. If you design for breadth and flexibility of reporting capability, you are almost guaranteed to fall into some fascinating trends you weren’t even looking for. Design for maximum analytical potential now and you’ll have all sorts of stuff to analyze for your next iteration.

User-first interfaces

For unauthenticated contexts where users matter (think multi-page web forms), consider getting the user to identify him/herself first. If things fail later in the process, you’ll have more options for following up.

Log Everything

Within reason. Consider how to correlate related or sequential actions across systems; we call this a “stream id”.

"Correlation"/source: https://xkcd.com/552/
Meet Kirk.

Kirk is an IdM team lead. He has a formal role on our team in support architecture. He has worked every tier of support at OIT, and can spot a support nightmare a mile away.

Kirk helps us.

Internally, Kirk sits in on all architecture/development planning and keeps us accountable for producing adequate Knowledge Base articles and identifying clear triage paths.

Kirk helps others (which helps us).

Externally, we also offer Kirk’s time to customers rolling out large projects integrating with our infrastructure.

A little up-front investment can go a long way toward ensuring that support issues, when they arise, can be properly routed to the appropriate group.
Post-Production: Tracking Problems

Ticket Analysis
Some tickets reveal bugs - this is good.
Some tickets reveal points of confusion - this is even better!

Exception Reporting
Developers need to get exception notifications - no filtering allowed.
Known bugs are not acceptable.

Talk to your Service Desk
Just because tickets aren't getting escalated to you doesn't mean there aren't support issues.
Research to Action: Case Studies (in Failure)

“Failure is simply the opportunity to begin again, this time more intelligently.”

Henry Ford
Case Study: Duke OneLink

<table>
<thead>
<tr>
<th>Version 1 Goals</th>
<th>Version 1 Realizations</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>One step, seamless registration</td>
<td>Multi-step, completely opaque points of failure</td>
<td>Our Service Desk hates us. Service owners hate us. Oh no, everyone hates us.</td>
</tr>
<tr>
<td>Let services register users via API; manage communication and branding</td>
<td>Users now think OneLink is a site-specific authentication service - doesn’t scale</td>
<td>Language in tickets, confirmed by usability testing</td>
</tr>
<tr>
<td>Guaranteed instantaneous account</td>
<td>Now users each have 100 accounts</td>
<td>Manually consolidating way too many accounts</td>
</tr>
<tr>
<td>Accommodate service-specific on-boarding (custom attribute assignment, etc.)</td>
<td>Binding dependencies to registration means existing users need to re-register</td>
<td>Irate users, jokes about “ManyLink”</td>
</tr>
<tr>
<td>Custom IdP login pages (OneLink-first, NetID-first)</td>
<td>Services not good predictors of preferred credential; users missed de-emphasized credential.</td>
<td>Log analysis, usability testing</td>
</tr>
<tr>
<td>Two classes of user: internal and external</td>
<td>It’s never this simple</td>
<td>Disillusioned architects</td>
</tr>
</tbody>
</table>

Major takeaway: we were so worried about ease of signup that we sacrificed ease of continued use.
Case Study: Duke OneLink

Imagining OneLink V2: Tough Choices

- We have to break some promises - make registration *harder*, stop accommodating service-specific needs, revoke ability for service-managed branding
- Real Duke UniqueIDs for everyone, apply same search/match for OneLink as NetID accounts

Getting Buy-in

- Launched a reporting console to highlight problems, provide benchmarks, introduced referral codes for service-specific metrics
- Extensive usability testing, development of service branding guide
- Roadshow: “OneLink v2: This Time it’s Personal”

Results

- Worries V2 would be a tough sell to V1 customers, but everyone moved voluntarily almost as soon as it was released.
- Invitation success jumped from 67% to 94% (78% instant)
- A service that has continued to double in size every six months for years, has eclipsed our NetID population, well past 100k users
# Case Study: Duke OneLink

<table>
<thead>
<tr>
<th>Version 2 Goals</th>
<th>Version 2 Realizations</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-step, centralized registration</td>
<td><strong>Success.</strong> We get some complaints that we can’t make it easier for everyone, but triage is much more straightforward and resolutions much faster.</td>
<td>Feedback from long-term users (positive), new users (mixed), and Service Desk (positive)</td>
</tr>
<tr>
<td>Real DUIDs for everyone</td>
<td><strong>Partial success.</strong> We’ve traded ‘fake’ identities for real identities without EPPNs. This makes identifier conversations with developers tricky.</td>
<td>Developer feedback - the EPPN gap is a bigger problem than we anticipated.</td>
</tr>
<tr>
<td>Only one login page option</td>
<td><strong>Success.</strong></td>
<td>No complaints about this layout. Launch better than anticipated.</td>
</tr>
<tr>
<td>Transparent metrics</td>
<td><strong>Success.</strong> Things stopped breaking and people lost interest.</td>
<td>Logs, empty seats at steering meetings.</td>
</tr>
<tr>
<td>Self-service help page (vanilla login)</td>
<td><strong>Success.</strong> Very helpful for identifying whether a login problem is a OneLink issue or another service issue.</td>
<td>Ticket analysis</td>
</tr>
</tbody>
</table>

Major takeaways: we’ve solved the registration problem, but there is still work to do:

- OneLink users need access to centralized self-service and Multi-factor Authentication
- We need a better solution to the EPPN/identifier gap
- Further automation of reconciliation process
Case Study: Duke OneLink

Left: two login pages, two confused audiences

Right: one login page, much more successful
## Case Study: Sponsored Accounts (formerly Guest Tool)

<table>
<thead>
<tr>
<th>Guest Tool Goals</th>
<th>Realizations</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make guest account request and maintenance self-service</td>
<td>This brings real value to the community and sponsors take their responsibilities seriously.</td>
<td>Log analysis, user interviews.</td>
</tr>
<tr>
<td>Accommodate different account types</td>
<td>Audit wants and relies this information, but we don’t verify or trust it.</td>
<td>Sponsorship analysis, sponsor interviews.</td>
</tr>
<tr>
<td>Accommodate different services (NetID, email, DukeCard, etc)</td>
<td>Tying these to account types incentivizes people to pick the wrong option, and they do.</td>
<td>Sponsorship analysis, sponsor interviews.</td>
</tr>
<tr>
<td>Allow Duke Health AD sponsorship for staff and faculty</td>
<td>Users confuse this with a guest account.</td>
<td>Support team feedback, ticket analysis.</td>
</tr>
<tr>
<td>Hide any functionality that a user doesn’t have access to</td>
<td>Users choose the wrong option because they can’t see the right option.</td>
<td>Support team feedback, ticket analysis.</td>
</tr>
</tbody>
</table>

Major takeaway: we see problems with the model, but users don’t mind once they get it. Help them get it first.
Case Study: Sponsored Accounts (formerly Guest Tool)

Imagining a transition to Sponsored Accounts:

- We have reason to trust our users with access requests: adopt a value of accommodate first, ask questions when reports look strange
- A sponsorship model will allow us to accommodate more types of exceptions and make on-boarding to new types of sponsorship easier

Getting Buy-in

- Interviews with stakeholders, random users, not-random users
- Behavioral logs informing reasonable defaults

Results

- Consistent growth in environment and usage
- A natural answer to facilitate Duke partnerships
- More streamlined architecture - no need to build exception cases into business logic

Sponsor all the things!/Hyperbole and a Half

Sponsor all the things!/Hyperbole and a Half
## Case Study: Sponsored Accounts (formerly Guest Tool)

<table>
<thead>
<tr>
<th>Sponsored Accounts Goals</th>
<th>Realizations</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super-sponsor enhancements: provide an example Excel template and do client-side JavaScript validation on input</td>
<td><strong>Success.</strong> We had big plans for multiple iterations of this, but the complaints stopped after the first round, so we pivoted.</td>
<td>User feedback</td>
</tr>
<tr>
<td>Introduce non-person (test and service) account management</td>
<td><strong>Success.</strong> Streamlined model, consolidated applications.</td>
<td>Team brainstorming (way to go, Teddy!)</td>
</tr>
<tr>
<td>Show what users don’t have access to</td>
<td><strong>Success.</strong> Significant ease on support.</td>
<td>Ticket analysis, UX consultation</td>
</tr>
<tr>
<td>Introduce a new layout to help user self-triage what type of account they’re sponsoring.</td>
<td><strong>Failure.</strong> Users still get confused about this. Back to the drawing board.</td>
<td>Usability testing, UX consultation, design services</td>
</tr>
<tr>
<td>Introduce “extended access”, a special type of guest account.</td>
<td><strong>Partial success.</strong> Still unable to roll out to Health System, some users fail to find the option.</td>
<td>Ticket analysis</td>
</tr>
</tbody>
</table>

Major takeaways: remove incentives to misrepresent data, remove distractions, increase guidance.
Case Study: Sponsored Accounts (formerly Guest Tool)

Current interface/issues:

- Sponsors still not finding the options they want.

- Do sponsors know what a “non-Duke user” is? This may be part of the problem.

- Is repetitive language impeding skimming for desired content?

Next steps:

- Produce alternate layouts

- Conduct usability testing

- Research, establish, and document controlled vocabulary
Case Study: Authentication Manager

Duke has more than 3,000 SPs registered against our IdP.

SP registration is self-service at Duke, but many are uncomfortable using our existing tools. It is critical to our mission that services take an active role in configuring and maintaining their SPs, but our efforts to offer analyst resources to support with navigation of the process has resulted in more demand and less customer engagement than is sustainable.

To address this gap, we are replacing our SP Registration tool with a new application called Authentication Manager, which includes a guided walkthrough, save-and-return option for SP registrations.

We earmarked time to help people feel supported in Shibboleth integrations, and by all measures, it has had the opposite effect. We need a more effective approach.

By shifting our support process from one of project-based, hands-on analyst support to one of step-based, app-assisted support, we aim to:

- Empower developers and service owners to manage and advance these integrations more independently
- Promote well-formed questions that can be addressed more efficiently by IdM
Case Study: Authentication Manager

A rough cut of V1 – piloting Jan 18

Goals for this version:

- Clarify discrete components to be reviewed, debugged, updated, and overall progress
- Facilitate project management and accurate updates to leadership with transparency about what is done and outstanding
- Provide automated support (e.g., certificate validation) where possible
- Harvest metrics about where we lose people (and iterate so we stop losing them)
- Bring Shibboleth to the people!

Register Your Site to use Shibboleth

Progress
- Process Overview
- Registration Info
- Contacts/Users
- Entity ID
- Certificate
- ACS
- Attributes
- Grouper Groups

Register the Assertion Consumer Service (ACS) URL
After a user authenticates at the Duke Identity Provider, the ACS handles the redirect back to your Service Provider.

Typically, you will want the user to be redirected to the original resource. For example, if your user initially went to https://test.com/testing, you would want them to be returned to this same URL after being redirected to the Duke login page.

Following the above example, most ACS values simply need to be:
https://your-entity-id/Shibboleth.sso/SAML2/POST

Add ACS URL. Why would I need more than one?
“Smells”: Not Just for Code

“A code smell is a surface indication that usually corresponds to a deeper problem in the system.“

# Design Process

<table>
<thead>
<tr>
<th>Smell</th>
<th>Refactored!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leadership/stakeholders dictate particulars of design.</td>
<td>Leadership/stakeholders review data-driven recommendations, provide feedback on methods used to produce recommendation.</td>
</tr>
<tr>
<td>Declarative statements about what users will do/understand/want with a particular UI function.</td>
<td>Suggestive statements about how to test hypotheses about user preference/cognition.</td>
</tr>
<tr>
<td>Not even usability experts know this without research.</td>
<td>Brainstorming about how to get more context on the problem.</td>
</tr>
<tr>
<td>Increasing visibility of key elements through repetition.</td>
<td>Increasing visibility of key elements through discipline and prioritization.</td>
</tr>
</tbody>
</table>
## UI Development/Implementation

<table>
<thead>
<tr>
<th>Smell</th>
<th>Refactored!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sloppy language</td>
<td>Consistent vocabulary</td>
</tr>
<tr>
<td>• Multiple terms for one thing</td>
<td>• Pick one: “log in” (verb) + “login” (noun), or “sign in” (verb) + “sign-in” (noun)</td>
</tr>
<tr>
<td>• Grammar</td>
<td>• Name/brand things when you need to.</td>
</tr>
<tr>
<td>Careless HTML</td>
<td>A clean DOM</td>
</tr>
<tr>
<td>• Using elements for</td>
<td>• Links are for locations (nouns), like this great place.</td>
</tr>
<tr>
<td>formatting/spacing rather</td>
<td>• Buttons are for actions (verbs), not “login” (every time you do this, an angel loses its wings)</td>
</tr>
<tr>
<td>than meaning</td>
<td>• Use list elements for lists, even if you don’t need bullets/numbers</td>
</tr>
<tr>
<td>• Code formatting: indentation</td>
<td></td>
</tr>
<tr>
<td>makes relationships clear</td>
<td></td>
</tr>
</tbody>
</table>
## Data-driven decision-making

<table>
<thead>
<tr>
<th>Smell</th>
<th>Refactored!</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brittle back-end metrics:</td>
<td>Flexible back-end metrics:</td>
</tr>
<tr>
<td>• Too limited</td>
<td>• Capture raw numbers rather than percentages</td>
</tr>
<tr>
<td>• Too focused</td>
<td>• Timestamps rather than true/false Boolean values</td>
</tr>
<tr>
<td>• Too biased</td>
<td>• “Why not” freebies</td>
</tr>
<tr>
<td>Designing metrics/reporting with a specific</td>
<td>Start tracking early and revisit often. Solicit feedback about what you’re</td>
</tr>
<tr>
<td>outcome in mind</td>
<td>’re tracking and what else might be relevant.</td>
</tr>
<tr>
<td>Making assumptions about intent behind action</td>
<td>Back-end metrics are not a replacement for talking to people.</td>
</tr>
<tr>
<td>Keeping metrics private</td>
<td>Being transparent about service weaknesses helps keep your community</td>
</tr>
<tr>
<td></td>
<td>focused on the right problems.</td>
</tr>
<tr>
<td></td>
<td>Today’s failure can be tomorrow’s victory!</td>
</tr>
</tbody>
</table>
Some Closing Thoughts (That Didn’t Fit Anywhere Else)

**Iteration is key** - Even if you could design a perfect UI in version 1, your users wouldn’t be ready for it. Iteration on development/architecture is a conversation with your community - we’re on this journey together.

**IAM is not kickball** - We must help foster IAM policy that accounts for users being in multiple constituencies. Policies have to be additive, not either-or.

**Onboarding is everything** - There is a new user enthusiasm that we are remiss not to capitalize on. When you discover something confusing about your service, think first about better ways that on-boarding can avoid that confusion.

**Nobody agrees on what A/B testing is, but we can all agree it’s great** - People use the term “A/B” to refer to all sorts of testing - automated or manual, structured or ad hoc, the important concept is to try things out and see what’s more successful.

Lastly, you don’t need a team of specialists to improve your services:

- Don’t have a designer? Follow your institution’s style guide, make sure everything lines up on a grid.
- Don’t have a UX team? Do your own user testing. Keep trying new things. Remember n=6.
- Don’t have a Kirk? Make cookies for your Service Desk. Ask for the scoop.
- Don’t have developers? Don’t underestimate what people will tolerate if they feel considered and informed. Find new ways to consider and inform.
Thank You

Email any time!
mary.mckee@duke.edu
IAM Online Evaluation
https://www.surveymonkey.com/r/IAMOnline-Dec2017

Propose topic for future IAM Online
Email woodbeck@internet2.edu
InCommon January Webinar Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 9</td>
<td>NSF Campus Cyberinfrastructure (CC*) solicitation and InCommon Requirements</td>
</tr>
<tr>
<td>January 10</td>
<td>Certificate Service Work Plan, SSO, and MFA</td>
</tr>
<tr>
<td>January 17</td>
<td>IAM Online - CI Logon 2.0</td>
</tr>
<tr>
<td>January 24</td>
<td>Baseline Expectations for Trust in Federation</td>
</tr>
<tr>
<td></td>
<td>Introduction to IdP and SP expectations</td>
</tr>
</tbody>
</table>