

# **Spritely Networked Communities Institute**

Plurality Research Network Conference

January 13th-15th 2023

**This is a lightning talk, so it is **given** that:**

**1) Centralized Social Media is Busted**

**2) We are all here looking forward to  
individual-empowered safe online  
communities.**

**We formed a non-profit to help with  
just that!**



# **Spritely Institute Founders**

- **Christine Lemmer Webber - Open & Social**
  - ActivityPub, The Spritely Project, MediaGoblin



# Spritely Institute Founders



- **Christine Lemmer Webber - Open & Social**
  - ActivityPub, The Spritely Project, MediaGoblin
- **Randy Farmer - 40+yrs Social Platforms**
  - Electric Communities, Avatars, E, JSON

The **ActivityPub** standard  
is *great* for  
federated **message sharing**.

The ActivityPub standard  
enables  
**“follow”** networks, like Mastodon.

# **ActivityPub Alone Isn't Enough**

- **Content survival**
- **Identity migration**
- **Better privacy & security**
- **Stronger anti-abuse & anti-harassment**
- **Context confusion**
- **Richer interactions**

# Disclaimer

What this talk is **NOT** about:

- **Changing the ActivityPub standard**
- **“Fixing” Mastodon, or other social apps**

**Our Focus:**  
**Re-Decentralize**  
**Networked Communities**



**A new foundation:**

**Secure and distributed is the  
*default***

**Your applications  
*represent you***

# Our social-stack of objects:

- **People**
  - Identity, Relationships, Connections
- **Communities**
  - Contexts, Memberships, Moderation
- **Applications**
  - Security, Trust, Consent -> Commerce

**No Gatekeepers Required**

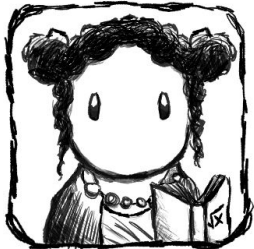
# **Mock Demonstration**



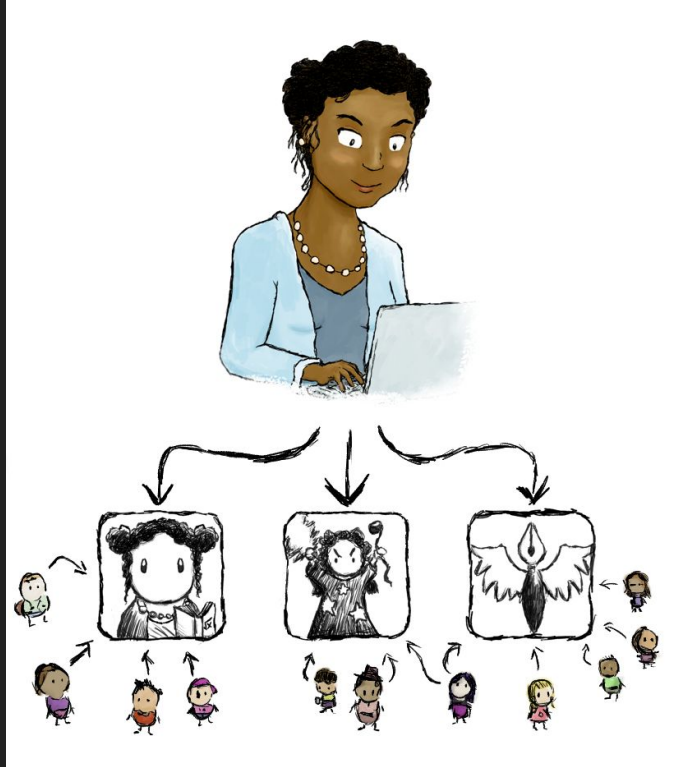
Ms A. Nym

A. Starlight

A. Penn



*[Faint, illegible text]*





Chat

Ben's Gaming Hangout



Ben

Hey AStarlight, I'm heading over to my friend ?Carol's place for board games and pizza, want to come?



?Goblin

I wish I had pizza...



AStarlight

Sure! Hello ?Carol, I'd love to join if that's ok!



?Carol

I'd love for you to join us, AStarlight!

A friend of Ben's is a friend of mine!

Looking forward to it!

Send!



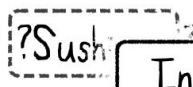
Chat

Ben



Ben

Would you like to play a game of ?Sush



Install



Click!

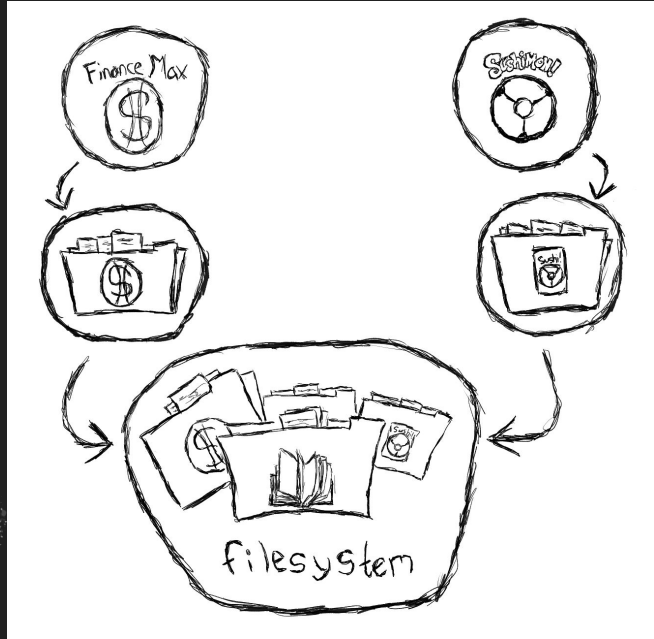


AStarlight

I've never heard of that before.

Let me install it!

Send!





Chat

Ben



AStarlight

I've never heard of that before.  
Let me install it!



AStarlight

Okay, installed!

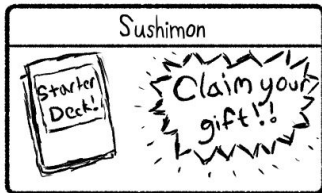


Ben

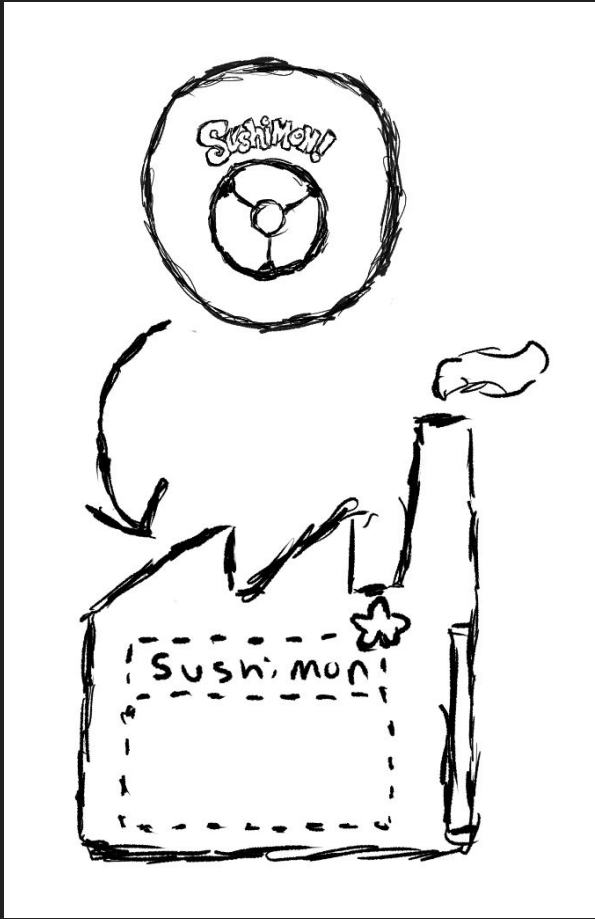
Great, now you need a starter deck!



Ben



Send!





Chat

Ben



AStarlight

I've never heard of that before.  
Let me install it!



AStarlight

Okay, installed!

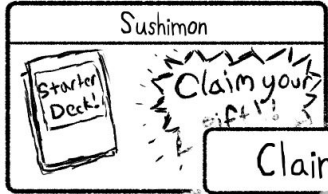


Ben

Great, now you need a starter deck!



Ben



Claim



Click!

Send!





## **What does the agency do for Alisha?**

- **Contexts and personas**
- **Mechanisms of consent**
- **Decentralized naming**
- **Robust commerce and trade**
- **Safe, cooperative apps**

**(progress report)**

# Papers, Platforms, Developers (Oh My)



The screenshot shows a web browser window with the URL <https://spritely.institute/static/papers/spritely-core.html>. The page title is "The Heart of Spritely: Distributed Objects and Capability Security". Below the title is a "Table of Contents" section with the following items:

- [1. Introduction](#)
- [2. Capability security as ordinary programming](#)
- [3. Spritely Goblins: Distributed, transactional object programming](#)
  - [2.1. On language and context choice](#)

## A Scheme Primer

### Table of Contents

- [1. Introduction](#)
- [2. Setting up](#)
- [3. Hello Scheme!](#)
- [4. Basic types, a few small functions](#)
- [5. Variables and procedures](#)
- [6. Conditionals and predicates](#)
- [7. Lists and "cons"](#)
- [8. Closures](#)
- [9. Iteration and recursion](#)
- [10. Mutation, assignment, and other kinds of side effects](#)
- [11. On the extensibility of Scheme \(and Lisps in general\)](#)
- [12. Scheme in Scheme](#)

The following is a primer for the [Scheme](#) family of programming languages. It was originally written to aid newcomers to technology developed at [The Spritely Institute](#) but is designed to be general enough to be readable by anyone who is interested in Scheme.

This document is dual-licensed under [Apache v2](#) and [Creative Commons Attribution 4.0 International](#) and its source is [publicly available](#).

## 1. Introduction

In all the world of computer programming, there are few languages as simple, clean, comprehensive, powerful, and extensible as introduction to the [R5RS](#) edition of Scheme's standardization<sup>4</sup> explains its philosophy well:

Programming languages should be designed not by piling feature on top of feature, but by removing the weaknesses and restrictions that make additional features appear necessary.

This minimalism means that the foundations of Scheme are easy to learn. The R5RS introduction continues with:

Scheme demonstrates that a very small number of rules for forming expressions, with no restrictions on how they are composed, suffice to form a practical and efficient programming language that is flexible enough to support most of the major programming paradigms in use today.

## Petnames: A humane approach to secure, decentralized naming

### Table of Contents

- [1. The what and why of petname systems](#)
- [2. Implementing petnames](#)
  - [2.1. Smartphone contact list integration](#)
  - [2.2. Web browser integration](#)
- [3. Conclusion](#)

"If we ever show a DID to a user we have failed."

Names must be human-readable in order to be widely used. Unfortunately, while DIDs and Tor .onion addresses are decentralized and globally unique, they are not human readable. How can we build user interfaces that real users might actually use? In this paper we provide an overview of petname systems, a way of mapping human readable names to cryptographically secure names, and describe changes to two user interface designs that we believe that are compatible with intuitive user expectations. We first discuss the smartphone contact list as already approximating petnames to some degree and discuss how to augment it with secure introduction. We then walk through several changes to browsers (which may be provided natively or as an extension) which add the functionality of a petname system. By utilizing petname systems we are able to collectively support individual naming definitions, community curated directories of names, as well as existing naming authorities such as certificate authorities and the domain name system, government agencies such as trademark offices, and decentralized systems such as Namecoin.

## 1. The what and why of petname systems

# Papers, Platforms, Developers (Oh My)

## NLnet grant bootstraps OCapN protocol standardization effort!

Spritely Institute -- Wed 19 October 2022



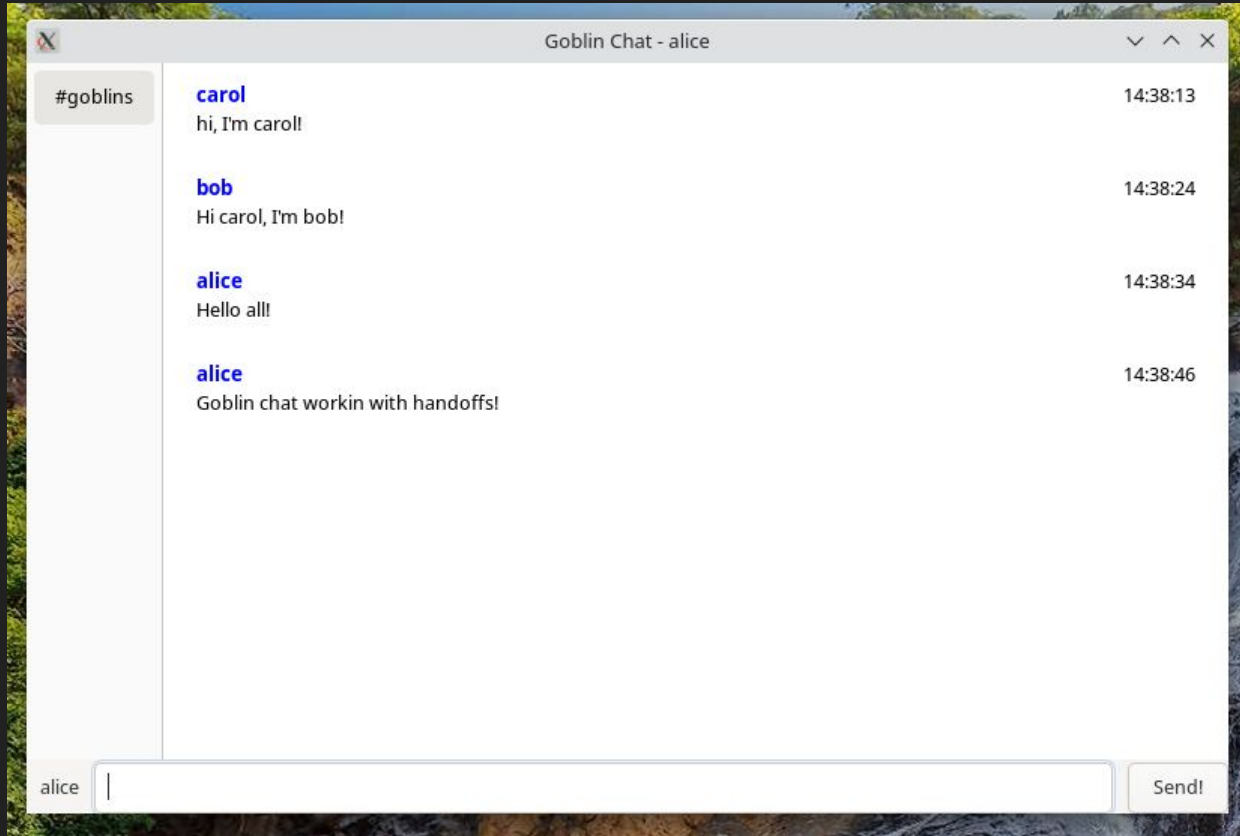
Jessica Tallon received a grant from NLnet to bootstrap the standardization process of OCapN (Object Capability Network). Jessica has worked with us on previous Spritely projects, including previous NLnet grants related to implementing petnames. Spritely Institute's role will be providing direction and support to Jessica, who will leading the effort.

We want to thank NLnet for funding this important work as standardization is critically important for the wider adoption and implementation of OcapN.

Steps:

- Initial Draft Specs
- Form a Community Group (preceding the standards Working Group)
- Compliance & Test Suite
- Implementers Guide
- Submit/Transform to [TBD] standards body

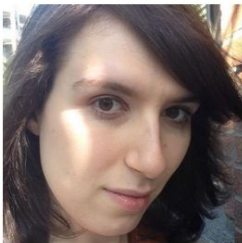
# Papers, Platforms, Developers (Oh My)



# Papers, Platforms, *Developers* (Oh My)

## WORDS

Essays, etc.



My name is Diana. I make things but generally not very well. I put thoughts here.

ME, ELSEWHERE:

Mastodon

GitHub

[🏠](#) > [A Conceptual Introduction to Spritely Goblins](#)

## A Conceptual Introduction to Spritely Goblins

I have recently been fascinated by [Goblins](#), of [Spritely](#) fame. It is, at present, a library in [Guile](#) and [Rackets](#) which provides a model of programming for peer-to-peer applications that makes permissions a kind of first-class object. In this essay I try to explain what that means, but I'll admit I've had some trouble with it so far. It's just... alien. It seems like a different order of sorcery than the likes to which I have grown stubbornly accustom. What it makes easy should be a decade of work. The principle of least authority -- an asymptote! -- made as practical as a parameter.

Goblins articulates a security paradigm of *object capabilities* which I find to be an apt name. In this paradigm, you construct objects that have capabilities, which are functions. If someone in the network can access a function in an object on your machine, it is only because you gave them permission. *If a function runs, it is because it is authorized to do so.*

That is the model Goblins hands down, not one of peers or users or identities, but of capabilities. You write object capabilities as stateless functions, and can call upon those capabilities that you can access. Applications can gather such capability-functions to create complex communal systems built upon consent. *If a function runs, it is because it is authorized to do so.*

This is not some cryptocurrency ledger. There is no append-only constraint or proof-of-work friction to contend with. The magic at work here is subtle rather than costly; its promises are thus paradigmatic

# Papers, Platforms, *Developers* (Oh My)

```
Geiser Guile REPL: community-garden* - GNU Emacs at Ikaruga
Host: localhost Port: 37146
under certain conditions; type ',show c' for details.

Enter ',help' for help.
scheme@(guile-user)> (vat-eval alice-vat ($ alice 'plant 5 4 cabbage/approve
d))
;;; <unknown-location>; warning: possibly unbound variable 'vat-eval'
ice-9/boot-9.scm:1685:16: In procedure raise-exception:

Community Garden
Spritely Institute Community Garden

      plant 5 4 cabbage/approved))
      plant 4 4 cabbage/approved))
      plant 3 4 cabbage/approved))
      dig-up 3 4))
      plant 3 4 cabbage/approved))
      plant 0 0 winter-squash))
      ation, Inc.
      r details type ',show w'.
      welcome to redistribute it
      r details.
      plant 0 0 winter-squash))
      plant 0 0 sunflower/approved))
      dig-up 0 0))
      $ = #<local-promise>
      scheme@(guile-user)> (alice-run ($ alice 'plant 0 0 sunflower/approved))
      $9 = #<local-promise>
      scheme@(guile-user)> (alice-run ($ alice 'dig-up 0 0))
      $10 = #<local-promise>
      scheme@(guile-user)>

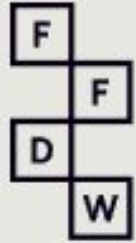
139 (methods
140 ((get-name) name)
141 ((get-bed) garden-bed)
142 ((plant x y sealed-plant))
143 (ensure-empty x y)
144 (let* ((plant ($ garden-gate 'check-plant sealed-plant))
145        (new-bed (garden-bed-set garden-bed x y plant)))
146   (bcom (^garden bcom name new-bed garden-gate)))
147 ((dig-up x y)
148  (let ((new-bed (garden-bed-
149          (bcom (^garden bcom name
150
151 (define (^visitor bcom name gar
152 (methods
153 ((get-name) name)
154 ((get-garden-name)
155 (<- garden 'get-name))
156 ((inspect-garden)
157 (<- garden 'get-bed)))
158
159 (define (^gardener bcom name ga
160 (methods
161 ((get-name) name)
162 ((get-garden-name)
163 (<- garden 'get-name))
164 ((inspect-garden)
165 (<- garden 'get-bed))
166 ((plant x y plant)
167 (<- garden 'plant x y plant
168 (dig-up x y)
169 (<- garden 'dig-up x y)))
170
171 (define the-botanist (garden-ru
172 (define the-garden-gate (garden
173 (define sunflower/approved
174 (garden-run ($ the-botanist '
175 (define cabbage/approved
176 (garden-run ($ the-botanist '
177 (define our-garden
178 (garden-run
179 (spawn ^garden
180 "Spritely Institute Community Garden"
181 (make-garden-bed 8 8)
182 the-garden-gate)))
183
184 (define alice (alice-run (spawn ^gardener "Alice" our-garden)))
185 (alice-run ($ alice 'plant 1 1 sunflower/approved))
```

**We're an Institute!**



**Let's share research!**





Filecoin  
Foundation for the  
Decentralized  
Web



**Our Supporters**



**Let's Re-Decentralize  
Community Together!**

**randy@spritely.institute  
christine@spritely.institute**

# Papers, Platforms, *Developers* (Oh My)



# Distributed Network Architecture

## Apps and Services

Discovery

Storage

Finance

## Communities

Membership  
Models

Threads &  
Posts

Governance &  
Moderation

## People

Identity &  
Authentication

Profiles &  
Attributes

Filters &  
Subscriptions

Consent &  
Capabilities

## Cooperating Mutually Suspicious Distributed Objects

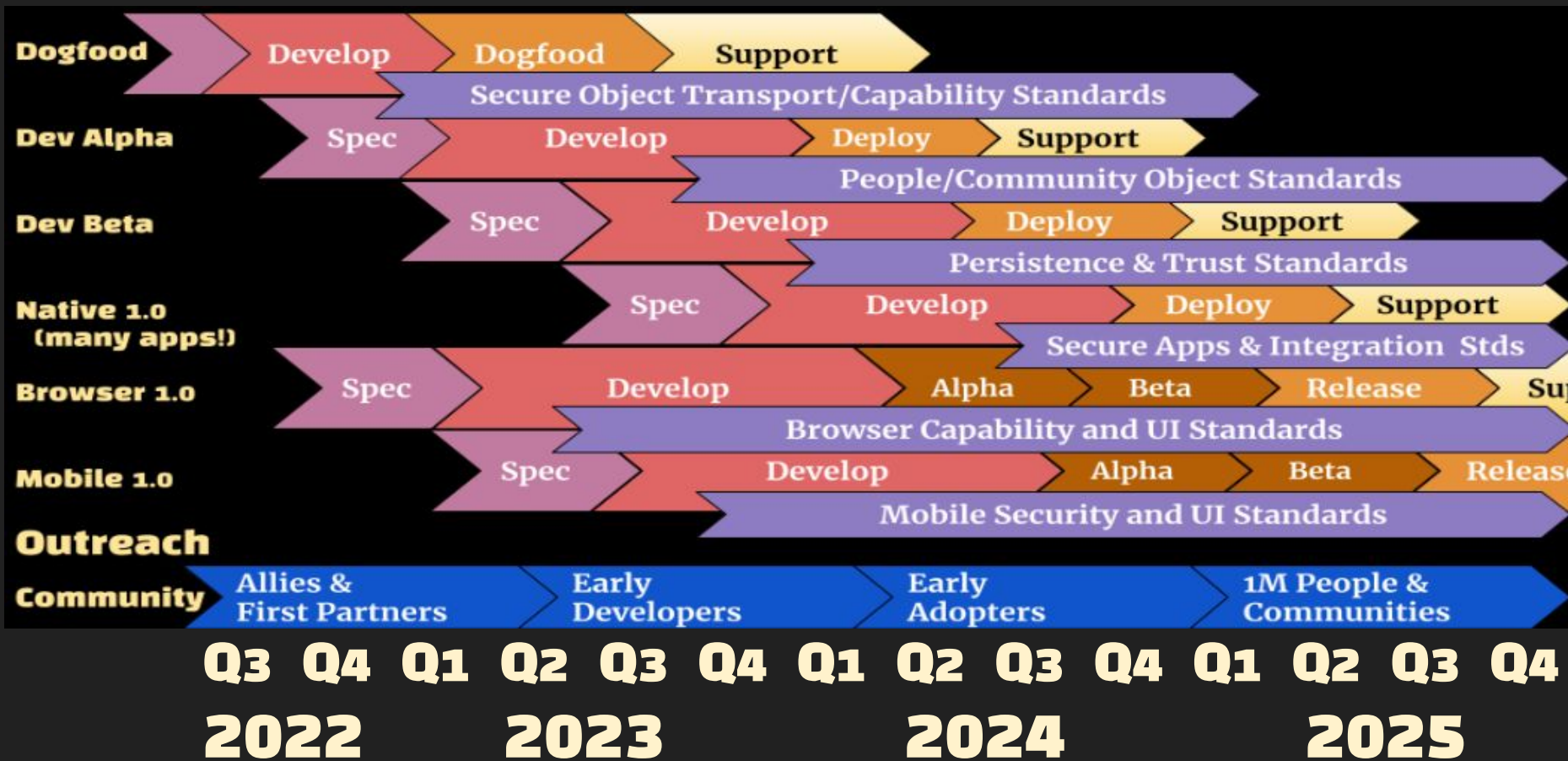
Capabilities

Trust Frameworks

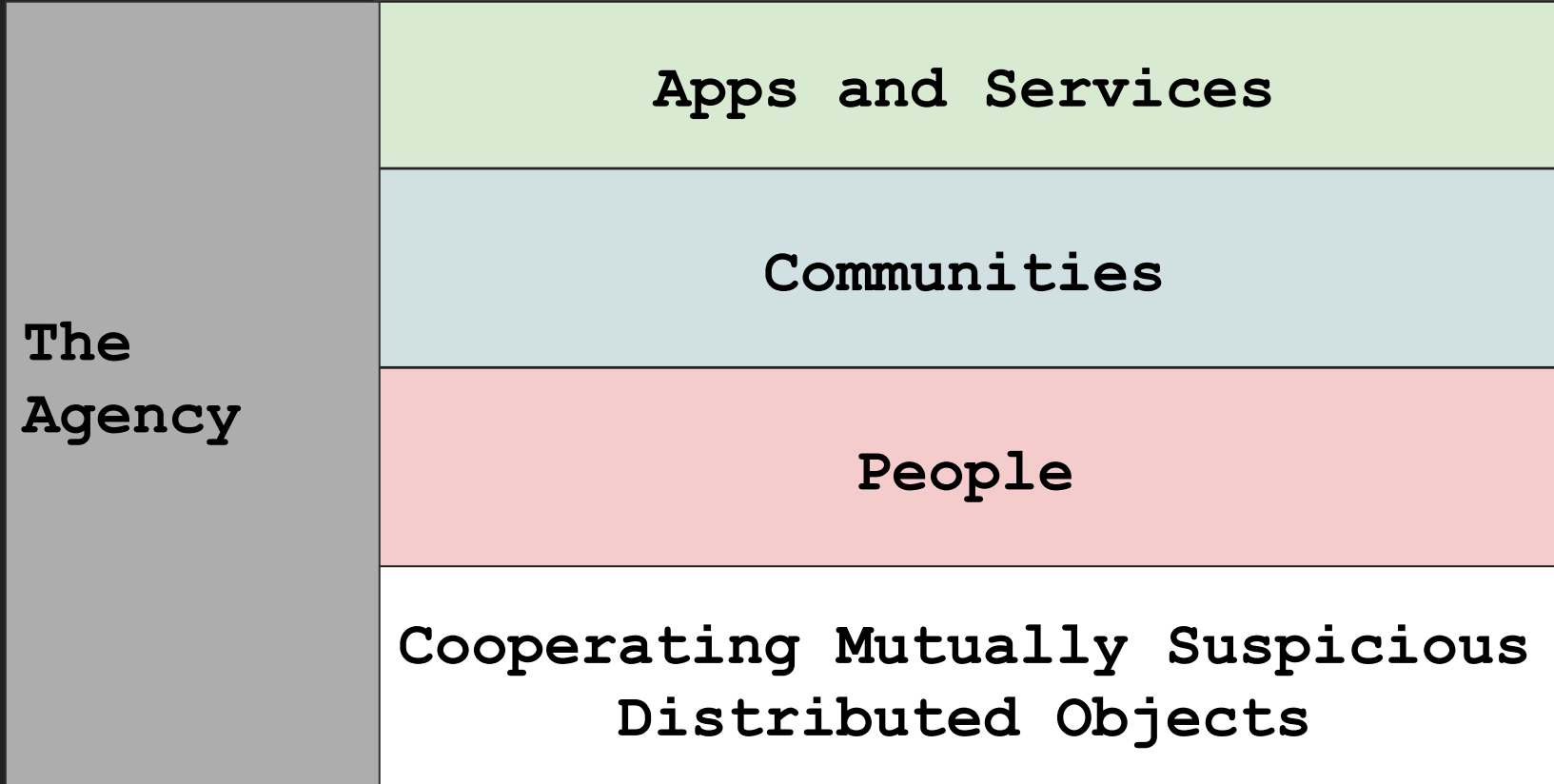
Persistence

Connections

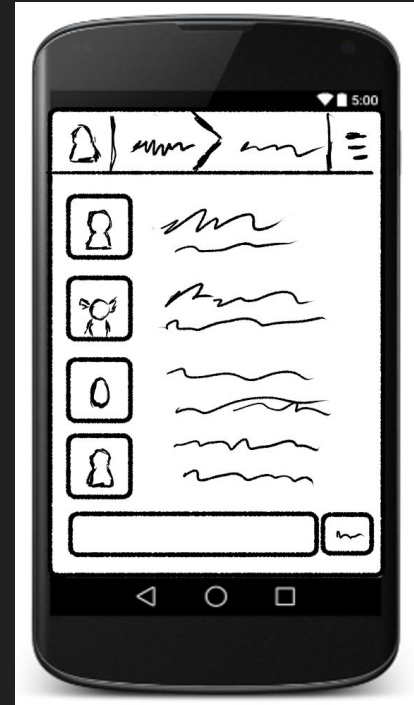
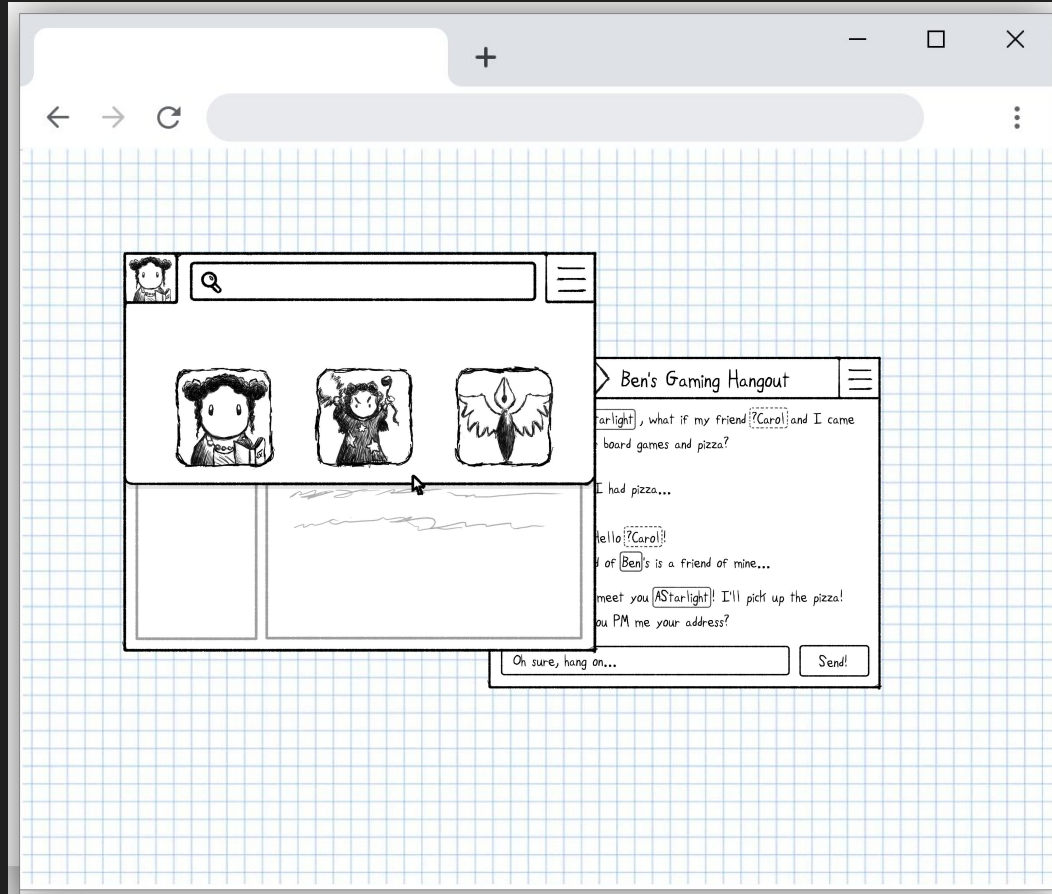
# SNCI Milestones (Fully Funded) 2022-2025



# The Agency: Client & Server



# The Agency



**But what we need to enable fully-user controlled secure communities requires so much more than federated messaging.**



# *Papers, Platforms, Developers (Oh My)*



Download

## Tutorials

### A Scheme Primer

This tutorial by Christine Lemmer-Webber and the Spritely Institute is a great introduction to everything you need to know about the Scheme programming language, with lots of examples directly applicable to Guile.