NICTA

My Idea of an SFA 2.0

Max Ott

NICTA

Australian Government

Department of Communications, Information Technology and the Arts

Australian Research Council

NICTA Members

ANU THE AUSTRALIAN NATIONAL UNIVERSITY

UNSW THE UNIVERSITY OF NEW SOUTH WALES

NSW First for Business Department of State and Regional Development

Victoria The Place To Be

THE UNIVERSITY OF MELBOURNE

The University of Sydney

Queensland Government

Griffith UNIVERSITY

QUT Queensland University of Technology

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

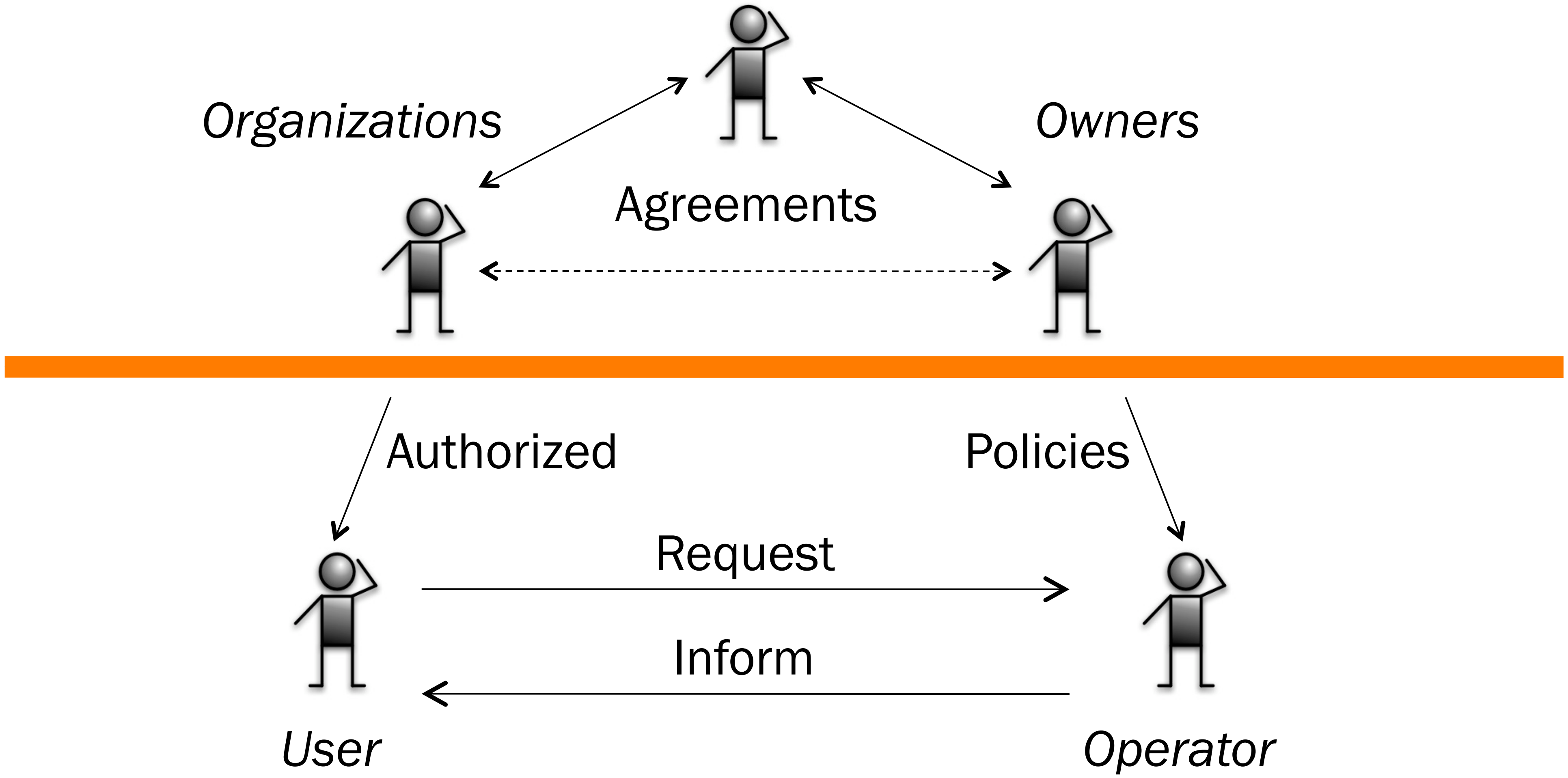
NICTA Partners

from imagination to impact

Terminology (SFA 2.0)

- Owners of parts of the network substrate
  - responsible for the externally visible behavior of their equipment
  - establish high-level policies for utilization of their resources
- Operators of parts of the network substrate,
  - often working for owners to keep the platform running, provide a service to researchers, and prevent malicious use of the platform.
- Researchers (and developers)
  - employing the substrate for running experiments, deploying experimental services, measuring, and so on.
- Identity anchors
  - drive authorization by asserting attributes (or roles) of other entities.
  - also sometimes called Identity Providers or IdPs

Actors


Organizations
Owners
Agreements
Authorized
Policies
Request
Inform
User
Operator

Resources, Components, Authorization

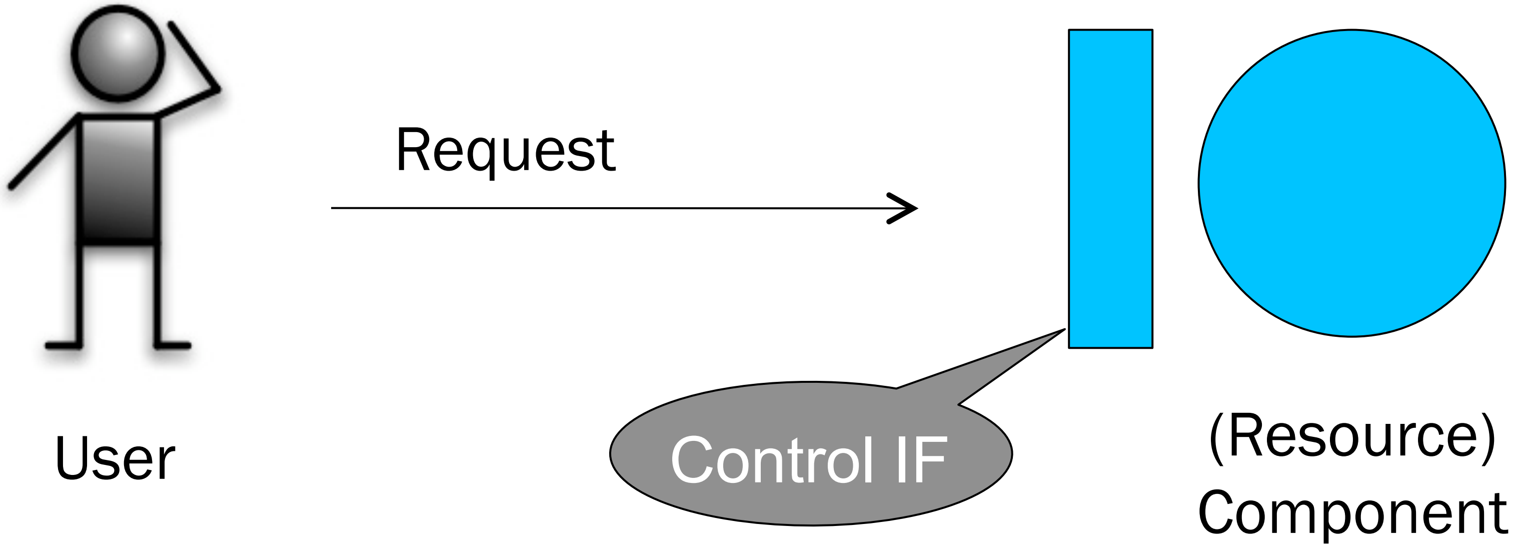
Users care about resources (and only about resources)

User

Resource

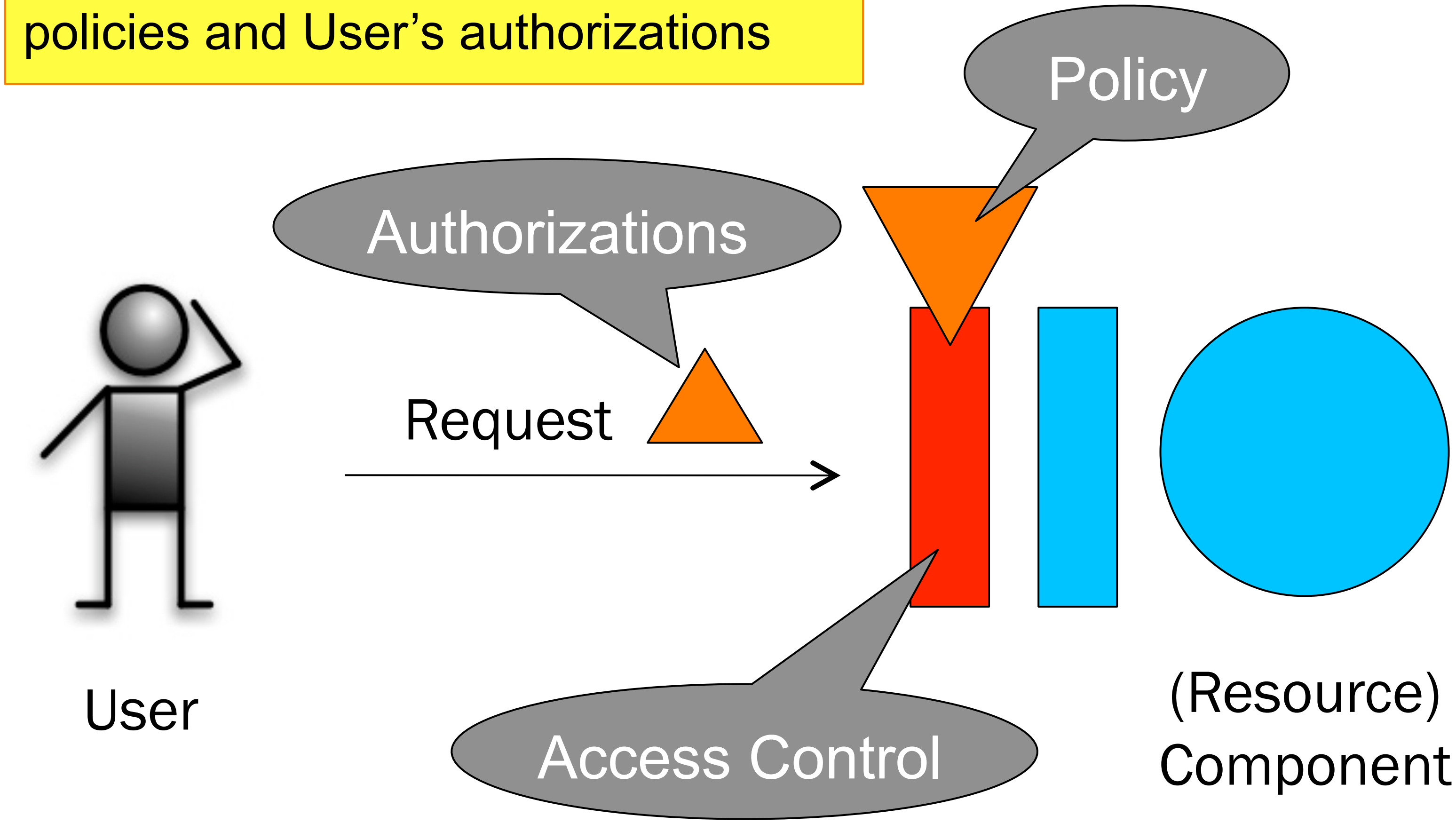
Resources, Components, Authorization

A component is a resource with a control interface



Resources, Components, Authorization

User access is governed by Owner's policies and User's authorizations

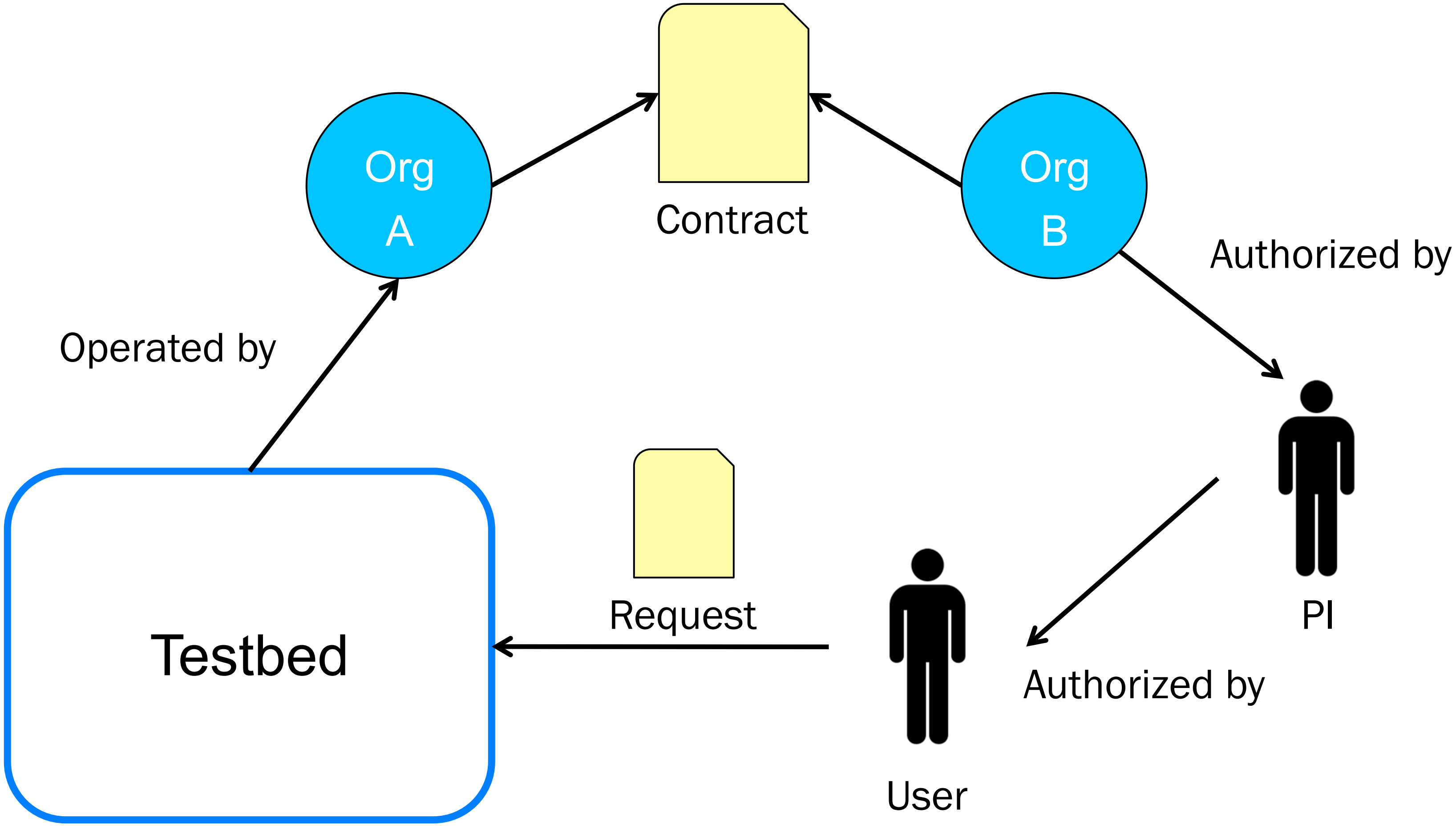


Federation

is primarily a

POLICY issue

In Federation, everything is Relative



What do we need?

- Policy Description
  - What ‘attributes’ (assertions) does a user need to access resource R during ΔT.
  - Need a mechanism to describe policies, not policies themselves
- Resource Description
  - Not necessarily Rspec, but it’s a pragmatic compromise
- Trust chains
  - Provides signed assertions about entities and their attributes
  - Identity Providers for users
    - Attributes: public key, memberships, roles, privileges, ‘budget’
  - Resource Brokers for resource assignments
    - Attributes: time duration, required user attributes

Separate Authorization from Authentication

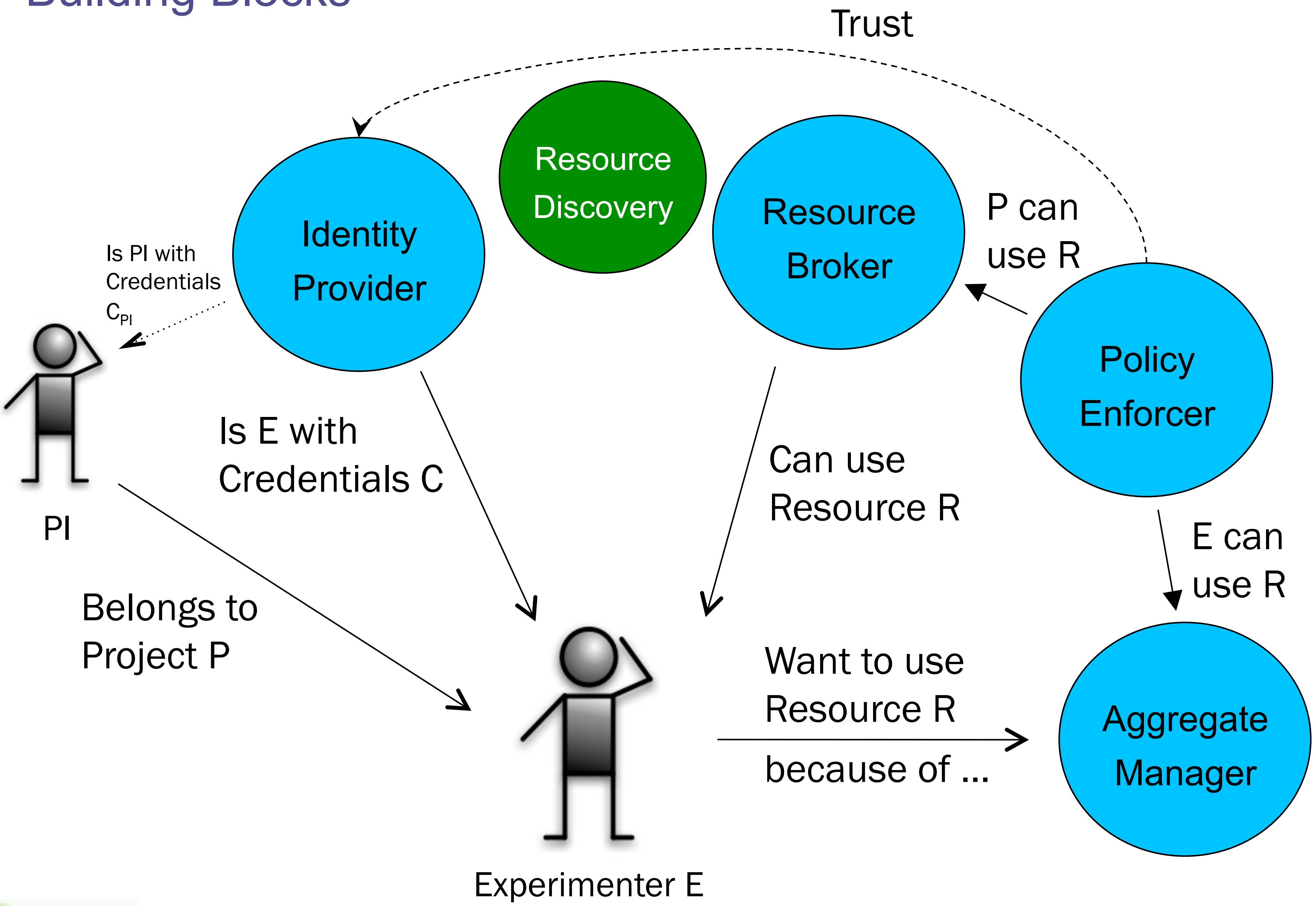
- A authorizes B to do C
  - Is A actually ALLOWED to authorize B => Authorization
  - Has A really said that? => Authentication

Assertions – A formal foundation

- Entity E asserts that Object O has Attributes A
  - Secure assertions are signed by Asserter
  - Assertions can be time and scope limited
  - Examples
    - PI A asserts that User B can perform action C on testbed D
    - Org E asserts that PI A can authorize others to perform C
    - Owner F asserts that Testbed D can allow C for users of Org E
- Policies determine necessary assertions to accept requests
  - Policies are local to ‘execution’ point
  - Examples
    - Experimenter needs to be belong to Org O
    - Reservations can/cannot be split

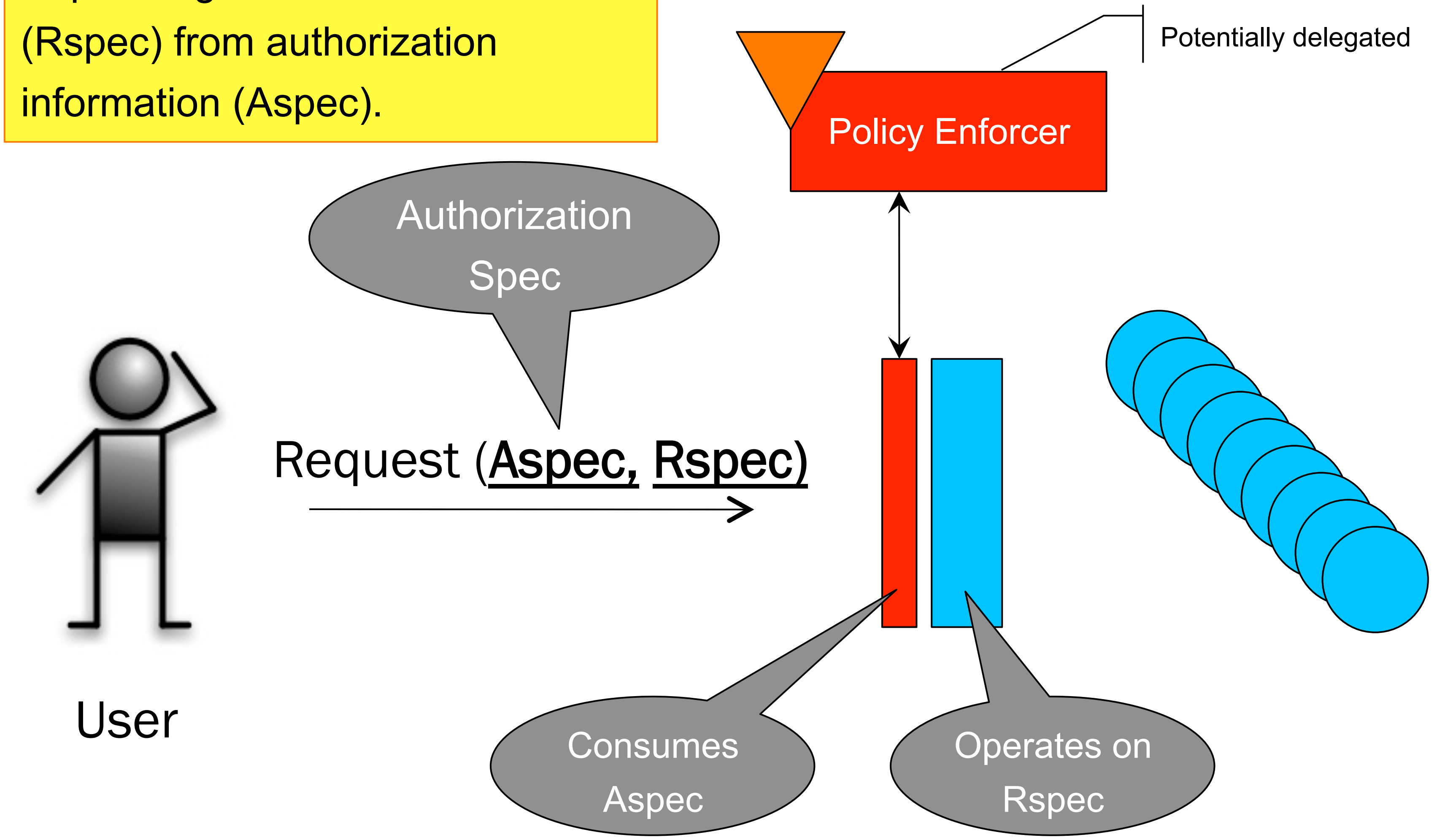
Putting it all together

Building Blocks



Aggregate Manager: Aspec + Rspec

Separating control information (Rspec) from authorization information (Aspec).



Slices, Aggregates

- Basic Principle: Many resources shared by many users
- Aggregates contain many resources operated by one
- User interacts with individual aggregates independently
  - Driven by policies in place between users and aggregates
- Limited interaction among aggregates
  - Only for stitching operations
- What is then a Slice?
  - Conceptually it is what one user/group gets from the entire cake
  - It's a concept and a grouping mechanism – that's all!

Minimal AM API

- Slice Lifecycle
  - `CreateSlice(SliceURN, ASpec) : success:fail`
    - Only creates slice context, no other resources bound
    - SliceURN is selected by user and should be globally unique label
    - SliceURN could be valid URL for AM callback (asynchronous op)
  - `DeleteSlice(SliceURN, ASpec) : success:fail`
  - `StopSlice(SliceURN, ASpec): success:fail`
    - Emergency shutdown/release of all resources in slice if authorized
- Resources Lifecycle
  - `ConfigureSlice(SliceURN, ASpec, RSpec): RSpec`
    - Provisions & configures resources listed in Rspec.
    - Release all resources no longer listed in Rspec.
    - Returns current state of resources as Rspec.
  - `InfoSlice(SliceURN, ASpec, RSpec): RSpec`
    - Returns current state of resources listed in Rspec as an Rspec

Summary

- Time to agree on basic principles so we can move on to the interesting parts.
  - Basic building blocks:
    - mechanisms to name & interact with resources
    - mechanisms to describe policies and authorizations
  - Policies, authorization, resource brokering,
- Time to shed legacy. We moved from a ‘benevolent dictatorship’ to a ‘messy federation’.
- Shift focus from control frameworks to what’s really need to be done in a federated world.
- If this is supposed to turn into an international effort, we need to make this process more inclusive

NICTA

My Idea of an SFA 2.0

Max Ott

NICTA

Australian Government

Department of Communications, Information Technology and the Arts

Australian Research Council

NICTA Members

ANU THE AUSTRALIAN NATIONAL UNIVERSITY

UNSW THE UNIVERSITY OF NEW SOUTH WALES

NSW First for Business Department of State and Regional Development

Victoria The Place To Be

THE UNIVERSITY OF MELBOURNE

The University of Sydney

Queensland Government

Griffith UNIVERSITY

QUT Queensland University of Technology

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

NICTA Partners