PolicyKit

Building Governance in Online Communities

Amy X. Zhang$^{1,2}$  axz@cs.uw.edu
Grant Hugh$^2$  ghugh@stanford.edu
Michael Bernstein$^2$  msb@cs.stanford.edu

$^1$University of Washington
$^2$Stanford University
Consider all the tools you use to participate in online communities today.
When it comes to the governance model that these tools provide to communities, they all share a strikingly similar pattern:

- **Admins**
- **Mods**
- **Regular Users**
These tools describe governance using a permissions model. So when a user wants to do something, the tool just checks what permissions they have before they can do it.

User A would like to do Action X, check their permissions.

Action X is approved.

<table>
<thead>
<tr>
<th></th>
<th>Action X</th>
<th>Action Y</th>
<th>Action Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>User A</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User B</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
And in order to modify governance, only certain roles (like an admin) can change a user’s permissions.
And in order to **modify governance**, only certain roles (like an admin) can change a user's permissions.

Meanwhile regular users have no say in the governance of their community.
This **permissions model** is also prevalent in most collaborative software that you use everyday, even down to the access control for files in UNIX-like operating systems.
A permissions model of governance can only describe a limited range of governance models—mostly top-down, autocratic, and punitive ones.

If a community wants to have a different style of governance, like a democratic system for making decisions, they have to either:

- carry it out manually in an ad hoc way
- use or build one-off software tools  
  cumbersome  error-prone  hard to maintain
But what if there was a way to express other forms of governance in the software, such as more democratic models?

What if communities had a way to build for themselves the governance that suits their needs and values?
PolicyKit

a software infrastructure that empowers online community members to **concisely author** a wide range of governance procedures and **automatically carry out** those procedures on their home platforms
PolicyKit’s main insight is to shift governance from articulating permissions to articulating procedures, where procedures can express a wide range of governance models concisely, including participatory and democratic models.
So instead of looking up a user’s permissions, we instead look up and run policies in PolicyKit, or short procedural scripts, that govern an action.

**User A would like to do Action X**

Action X is approved

**check the policies governing X**

1) (Pseudocode) Before X can happen, first it must be approved by a majority of community members

2)...
Users can also write policies to govern actions such as adding or updating a policy, allowing users to participate in **modifying governance**.

**User A would like to change the policy governing X**

The policy is changed

**check the policies governing changing existing policies**

1) (Pseudocode) To change any policy, a two-thirds majority vote must first approve the change.

2) ...
We call policies that govern everyday actions on a community platform **Platform Policies** and policies that govern actions to modify the governance itself **Constitution Policies**, taking inspiration from political science theory developed by Elinor Ostrom.
Examples

**Actions**: one-off events that users can propose

Sara would like to post the message “Hi” to the #announcements channel on Slack.

**Policies**: continually running scripts that govern actions that come up

Any posts to the #announcements channel must first be approved by a moderator.

Platform
**Actions**: one-off events that users can propose

*Sara would like to post the message “Hi” to the #announcements channel on Slack.*

**Policies**: continually running code declarations that govern actions that come up

*Any posts to the #announcements channel must first be approved by a moderator.*

**Examples**

**Platform**

*Sara would like to post the message “Hi” to the #announcements channel on Slack.*

**Constitution**

*To add any new policy, it must first be voted in by majority vote.*

*Jane would like to introduce a new policy for renaming channels.*
PolicyKit Components

- platform integrations
- policy engine
- PolicyKit website

How they all comes together:
Following is an example of a policy that creates a random jury of members to vote on renaming Slack channels.

Writing a policy involves implementing 6 functions.
The policy filters to only consider actions involving renaming Slack channels.

```python
def filter(action, policy):
    if action.action_type == 'SlackRenameChannel':
        return True
```

Then we initialize the policy by creating a random jury of 3 community members to vote on this rename.

```python
def initialize(action, policy):
    usernames = [u.username for u in users]
    jury = random.sample(usernames, k=3)
    action.data.add('jury', jury)
```

We notify those 3 jury members about what they're voting on and other instructions.

```python
def notify(action, policy):
    jury = action.data.get('jury')
    jury_users = users.filter(username__in=jury)
    action.community.notify_users(action, policy, users=jury_users,
                                   text='Please deliberate amongst yourselves before voting')
```
def check(action, policy):
    jury = action.data.get('jury')
    jury_users = users.filter(username__in=jury)
    yes_votes = action.proposal.get_votes(users=jury_users, value=True)
    if len(yes_votes) >= 2:
        return PASSED
    elif action.proposal.time_elapsed() > datetime.timedelta(days=2):
        return FAILED

def pass_action(action, policy):
    action.execute()

def fail_action(action, policy):
    return
PolicyKit:
a software infrastructure that empowers online community members to **concisely author** a wide range of governance procedures and **automatically carry out** those procedures on their home platforms
1. user renames channel
2. listener catches action
3. run policies on actions

Platform integration

4. revert rename action
5. notify jury

PolicyKit Server policy engine

Jury policy for channel renames:
- Filter
- Initialize
- Check
- Notify
1. User renames channel

2. Listener catches action

3. Run policies on actions

PolicyKit Server

4. Revert rename action

5. Notify jury

6. Listener catches votes

7. Policy loops

Platform

8. Channel is renamed

Platform integration

Policy engine

Jury policy for channel renames:
- Filter
- Initialize
- Check
- Notify

Pass
PolicyKit API

JSON blob data with each action and policy to be able to have memory

```python
def initialize(action, policy):
    usernames = [u.username for u in users]
    jury = random.sample(usernames, k=3)
    action.data.add('jury', jury)

def notify(action, policy):
    jury = action.data.get('jury')
    jury_users = users.filter(username__in=jury)
    action.community.notify_users(action, policy, users=jury_users,
                                   text='Please deliberate amongst yourselves before voting')
```
Extensions to the Basic Data Model

roles and permissions that override policies

text documents for natural language policies like community guidelines

action bundles for elections (choice between multiple actions)

policy bundles for multi-stage policies (like a two-round caucus)
PolicyKit Website

Passed Process Policies

Starter Policy: all policies pass

Passed Community Policies

Majority rule to pass any community policies

Filter Code:
```
return True
```

Init Code:
```
pass
```

Notify Code:
```
post_policy(policy, action)
```

Conditional Code:
```
if aye_user_votes => math ceil(num_in_community / 2) then 
return Proposal.PASSED
```

Action Code:
```
execute_action(action)
```

Failure Code:
```
pass
```

Roles and Permissions

Role: Base User

Permissions:
- Can add boolean vote
- Can change boolean vote
- Cannot delete boolean vote
- Cannot view boolean vote
- Cannot add community action bundle
- Can add community action bundle

Users:
- Amy Zhang
- None
- Grant Hugh

Recent Actions

Policyengine | policy
Policyengine | policy
Policyengine | policy

Quick Links

Policy filter code:
```
if action.action_type == 'SlackNameConversation':
    return True
```

Policy init code:
```
usernames = [u.username for u in users]
jury = random.choices(usernames, k=1)
action.data.add('jury', jury)
```

Policy notify code:
```
all() and 
any()
```

Policy conditional code:
```
jury = action.data.get('jury')
jury_users = users.filter(username__in=jury)
if len(jury_users) == 0:
    return PASSED
elif action.proposal.time_elapsed() > datetime.timedelta(days=2):
    return FAILED
```

Policy action code:
```
action.execute()
```

Policy failure code:
```
pass
```
We have built platform integrations with Slack, Reddit, and Discord and plan to add additional platforms.

You can learn more about PolicyKit and add it to your community at policykit.org

Our code is open-sourced at https://github.com/amyxzhang/policykit

PolicyKit: Building Governance in Online Communities
Authors: Amy X. Zhang, Grant Hugh, Michael S. Bernstein