Model Driven Platform Policy

Alex Vul – Intel

Andy Mayer – AT&T

Pamela Dragosh – AT&T

December 11-13, 2017

Abstract

This 2-hour session is both presentations followed by panel discussion on how models can be used to specify platform policy.

For ONAP to be truly self-service and to support adaptive model drive management and control, any Dev Operations team must be able to easily specify and validate platform level policy providing the foundation for service and resource related policies. Every company has Dev Ops requirements that share common needs, but differ in current infrastructure requirements. Having a flexible, simple information and data model that captures platform policy is necessary to support all the possible scenarios. In addition, platform-level policies must be able to integrate well and work efficiently with the service and resource level policies that evolve over their lifecycle.

- How can we define models that can meet the dynamic and changing needs of a policy-enabled ONAP without the time and expense of modifying software code?
- How can operation teams use ONAP policies to automatically respond to events and conditions, in order to reduce the time required to react while increasing overall network and service availability thereby reducing the labor and effort required to manage the complex infrastructure and services?
- To support self-service, how does a model-driven Policy Platform support the capabilities to administer policies and monitor policy execution, including: policy creation and editing, policy translation, policy harmonization (conflict detection/resolution), policy validation, policy activation and tracking, policy monitoring, and policy reporting?
- How can the ONAP community bring together the Policy Information and Data Models, the Use-Cases, Common Reference Points and exchange mechanisms that are essential to enabling a policy-based model-driven ONAP?
- How do Policies that were defined during on-boarding of a VNF evolve when they are consumed as part of the on-boarding and instantiation of service-level policies?

Agenda

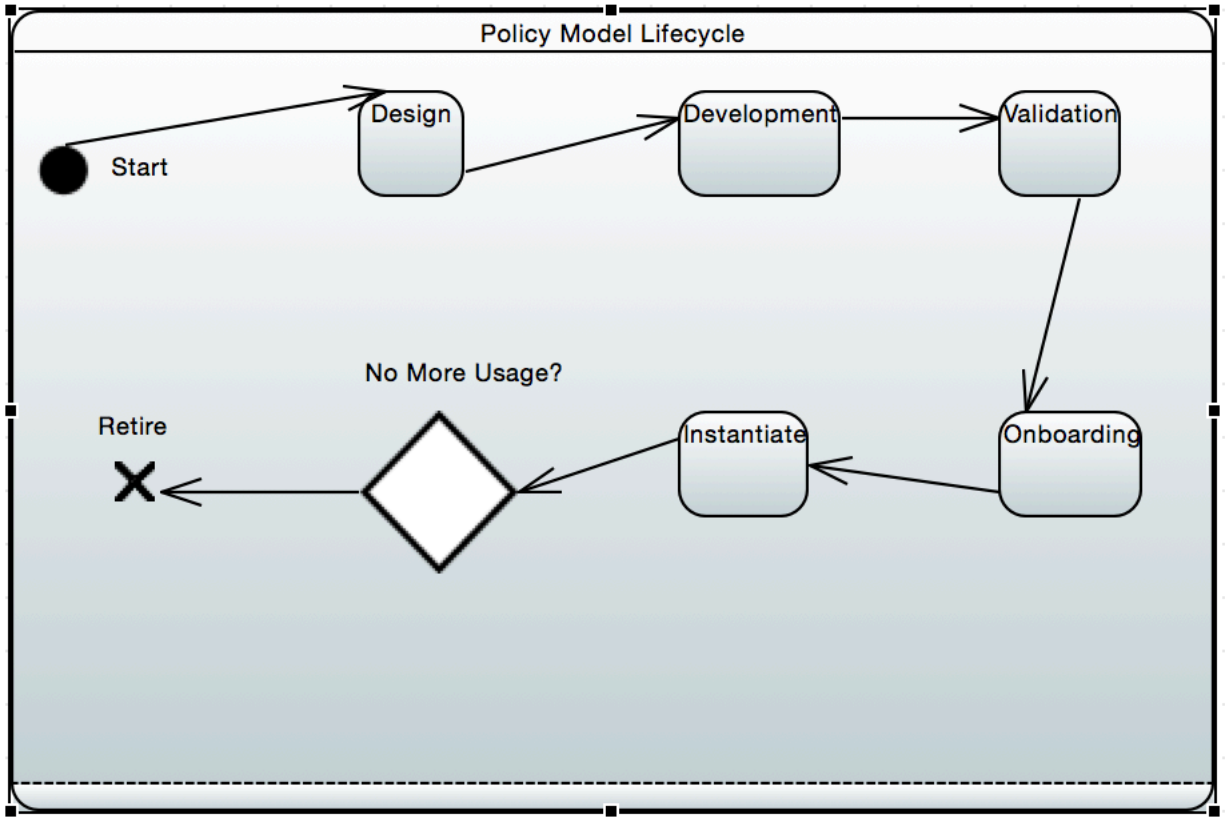
Panel Presentations (TBD – alphabetical order)

- Alex
- Andy
- Pam

Open Panel Discussion

Platform Policy Lifecycle

- Platform Policy Models can have a clear lifecycle, from an initial Design Phase to a potential Retirement Phase
- Here is an example State Diagram Model for Policy Lifecycle



Platform Policy Lifecycle – Design Phase

Example: Operational Locking Policy

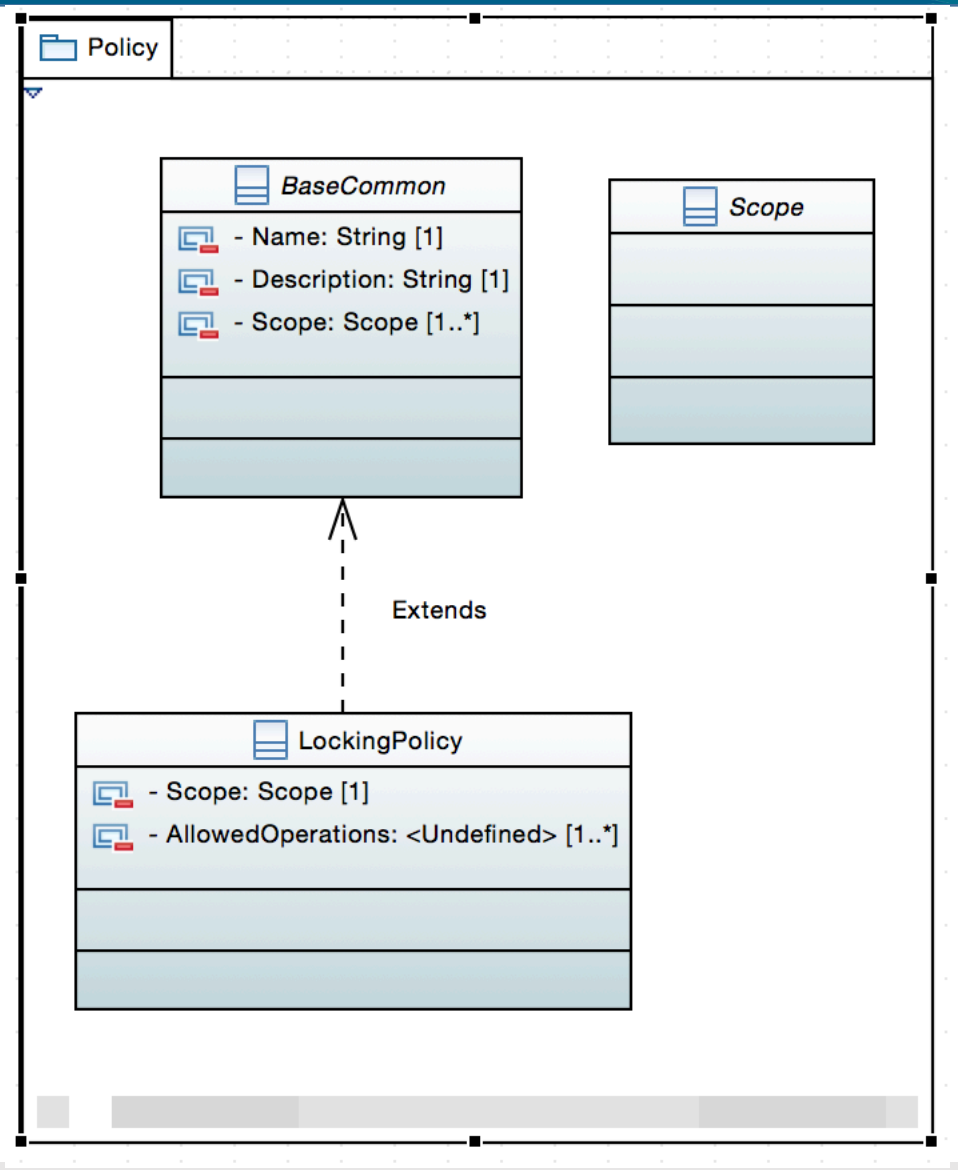
User Story: As an ONAP Operator, I would like to be able to specify a policy that can lock a specific network, service, or resource so I can perform an operation on either an element within the area that is locked or an element located elsewhere.

Use Case #1: A VNF that has several VM's running. An operator may want to lock all the VM's for a particular VNF from other actions such as (VM Restart/Reboot) when a particular operation (eg. Rebuild) is being performed on one of its VM's. However, it should allow other operations to be performed (eg. Health Check, Connectivity Check).

- Architectural Diagrams designed
- Flow Diagrams created (eg. sequence diagram)
- Class Models - Should the models be developed in this phase? This could overlap with Development Phase

Platform Policy Lifecycle – Development Phase

- Auto-Generation of Models into code is possible – but limited
- Still require development to fill in methods (Eg. serialization/deserialization) and possibly development is needed to integrate with other existing infrastructure models
- Can provide simple validation phase requirements such as developer best practices, CI/CD requirements, Release Versioning, etc.
- How can we provide guidelines/processes that can speed up the development process?
- Does the model development process have to align with ONAP releases?



Platform Policy Lifecycle – Validation/On-Boarding Phases

- Validation (beyond development phase)
  - How does a Policy Model truly Perform in the platform?
    - Conflict Detection Tools – still not 100% guaranteed to find conflicts
    - Setting up test environments and build functional tests that can satisfy Sunny Sky and/or Rainy Day scenarios
      - May be limited on resources
      - Could utilize logs captured from real-world network issues
      - Could run in an environment that would utilize existing production logs to see how a policy would perform
  - Time and Resources are always a consideration
    - Need the Ops teams to be able to be highly responsive to occurring network conditions
    - Additional strategies could be implemented to ensure policy(s) behave appropriately and do not bring any harm to the platform or network.
- On-Boarding
  - Is this simply using the CI/CD process to release artifacts?
  - For released code, may require Change Management for ONAP Platform components to be able to utilize new Policy Models

Platform Policy Lifecycle – Instantiation Phase

- Platform Policy CRUD
  - Policies are now available to CRUD via Ops Teams, Service Designers, Resource Developers, etc.
  - Governance Policies can also be modeled, CRUD, and in-place before any resource/service/network policy can be created.
- Policy Deployment
  - Strategy 1: Deploy a policy in a self-contained portion of the network.
    - Scope of the policy can be fine-grained (eg. Cloud Region, Service, Service Instance, etc.)
  - Strategy 2: Deploy a policy in ”Safe Mode” – requires oversight from Operation Personnel before an “automated” operation can be executed.

Platform Policy Lifecycle – Retirement Phase

- What constitutes Retirement for a Policy Model?
  - Lack of use? Maybe applicable to services/resources that are being deprecated. However, there are some policies that are in place to serve as Safe Guards - those would not be candidates for retirement.
  - Need appropriate models for both scenarios

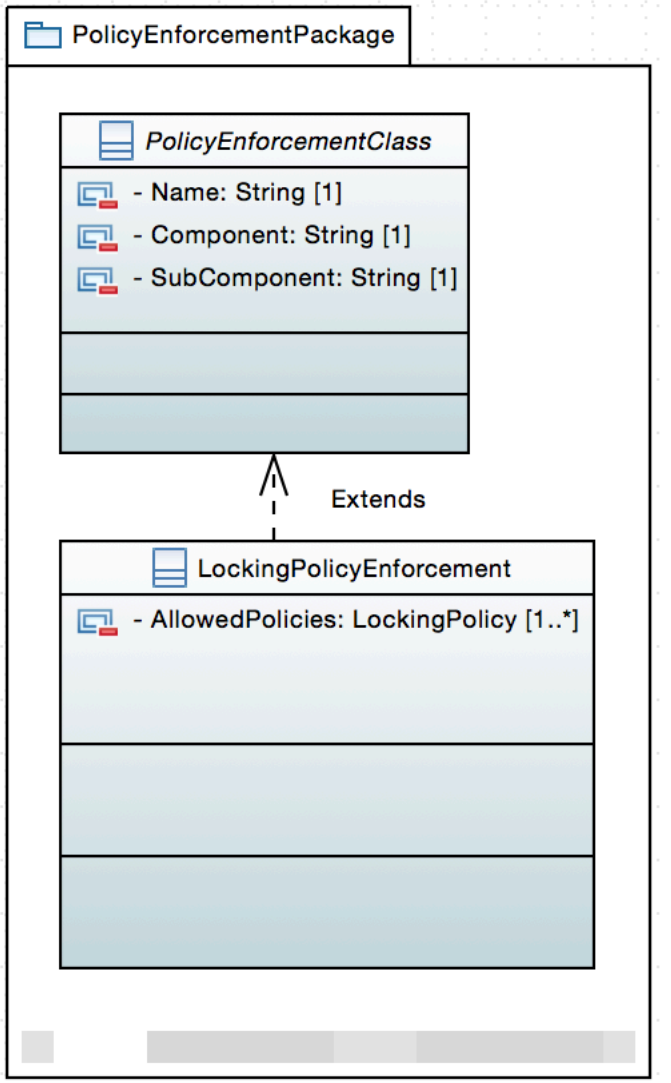
How to Enforce Model Drive Platform Policy

Enforcement of Policy happens in many parts of the ONAP Platform – Policy Enforcement Points (PEP). Some examples:

- DCAE Micro services are Enforcement Point for Control Loop Configuration Policy
- Policy Drools PDP is the Enforcement Point for Control Loop Operational Policy
- For Beijing the OOF will be the Enforcement Point for Optimization Policies

Models for Enforcement Points are needed to ensure consistent tracking and monitoring of Policy within the Platform

- Models for Enforcement Point themselves
  - Need to ensure the PEP only enforces what it is allowed to
- Models for Tracking the Enforcement Point
  - Need to ensure the PEP is actually enforcing the policy as desired by the policy writer
- Models for Learning Capability for Enforcement Point
  - PEP may be behaving correctly, but how could we learn from their behavior and improve the system?



Panel Discussion

Thank You!