



ESnet

ENERGY SCIENCES NETWORK

SURF

Orchestrator Domain Models

SURF, ESnet

TNC 2023



U.S. DEPARTMENT OF
ENERGY
Office of Science



Topics

1. Domain model basics
2. Example domain models
3. Tips for building domain models
4. Updating existing domain models

Domain Model Basics

- What is a domain model
- Components of a domain model
 - Product
 - Product Blocks
 - Resources
 - Workflows

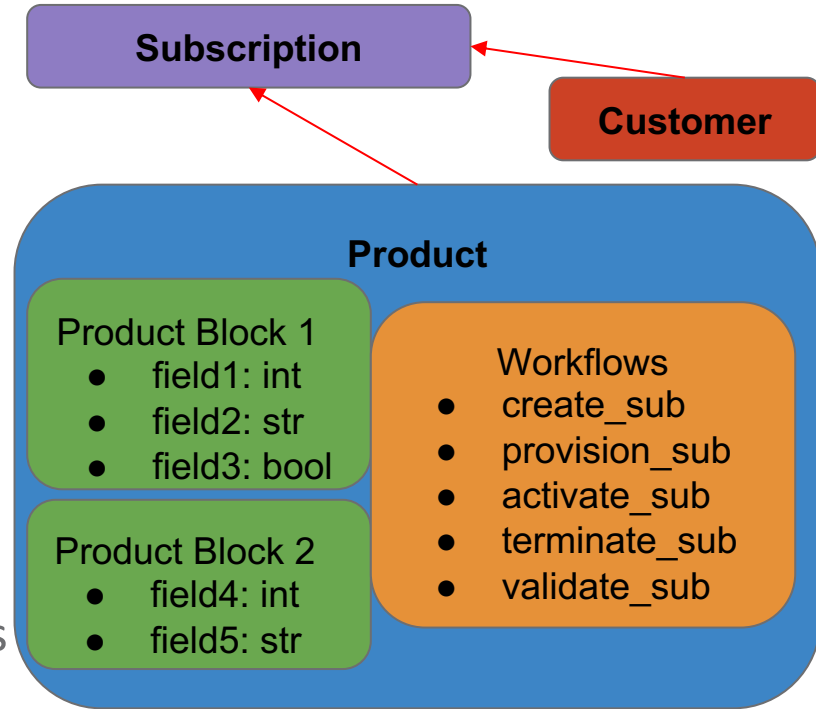
Domain Model Basics

What is a domain model?

- A representation of a product offering, the reusable components of that product, and the actions an engineer can take to manage a customer's subscription to the product
 - Product offering: **Product**
 - Components of product: **Product Block** and its **Resources**
 - Management actions: **Workflows**

Domain Model Basics

- **Subscription:** associates a customer with a product
- **Product:** a collection of product blocks and workflows
- **Product block:** a reusable collection of fields
- **Workflow:** an action that can
 - modify the content of product blocks
 - update the state of the subscription
 - effect change in an external system



Domain Model Basics

Product

- Defines an offering provided to users
- Composed of one or more product blocks and a set of workflows (typically at least 3)

Product Block

- Reusable collection of resources

Resource

- A pydantic field on a product block

Workflow

- A script-like process to manage a subscription

Domain Model Basics

- Kinds of data that are good to place product blocks
 - Resource IDs
 - Resource names
 - Configurable values relevant to the customer's subscription
 - i.e., speed, bandwidth, object state
 - References to other domain models
- Kinds of data that are not great for product blocks
 - Duplicate data from an external system that frequently changes
 - Anything not used to identify a resource or configure the network
 - Will require constant updating, else the subscription will become out-of-sync

Domain Model Basics

Workflows

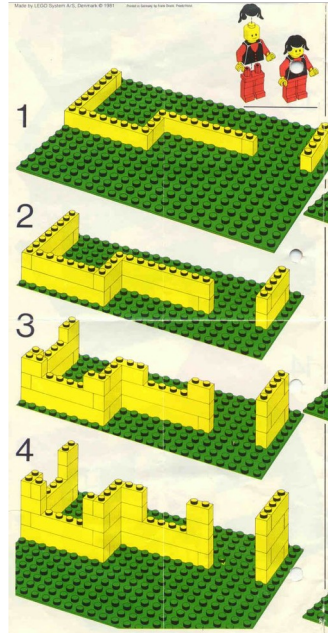
- What are workflows
- Role of workflows with other constructs
- Examples

What are workflows?

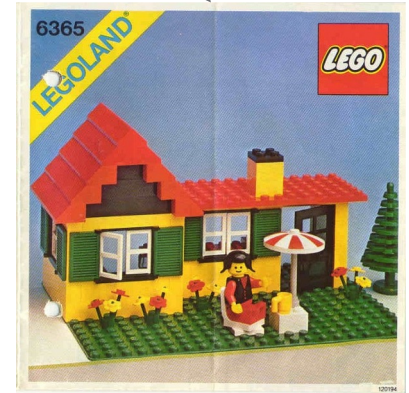
Finite resources exist in other systems



Workflows gather resources from those systems and associates them together to create a subscription



Orchestrator contains the final product



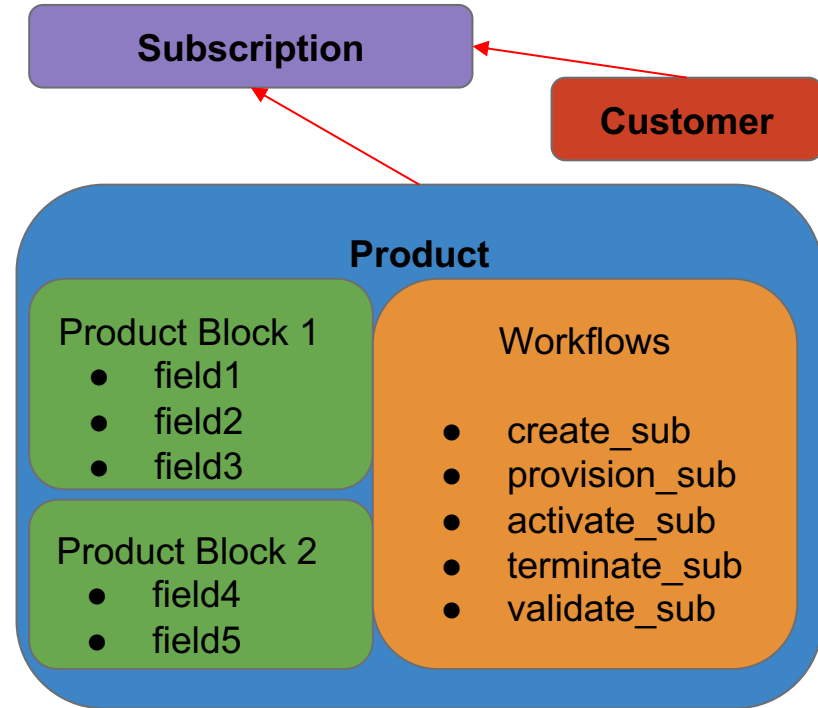
What are workflows?

- Workflows are actions that can be taken on a product subscription
- Workflows instantiate, modify, and terminate subscriptions
- Typical kinds of workflows:
 - `create_subscription` (instantiates a subscription to the product)
 - `modify_subscription` (toggle change in an external system)
 - `terminate_subscription` (ends a subscription to the product)
 - `validate_subscription` (checks the data in external systems is in sync with the domain model)

Role of workflows with other constructs

- Workflows belong to a product
- Workflows instantiate and modify a customer's subscription to a product
- Workflows make changes to the product blocks associated with a product
- Workflows manage the lifecycle of a subscription:

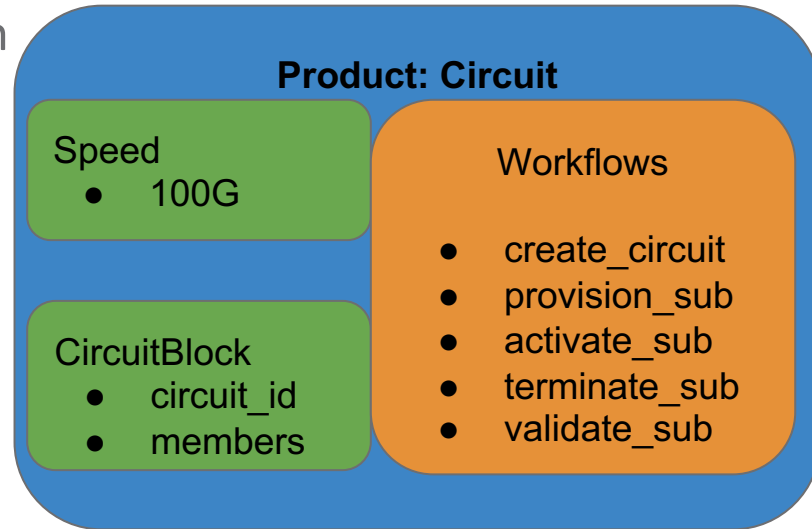
inactive -> provisioning -> active -> terminated



Using Domain Models

- Define a set of Product Blocks, each with a set of Resources
- Write workflows to
 - instantiate and modify a subscription to the product
 - Make changes to external systems
- Create a Product that is composed of product blocks and workflows
- Workflows carry a subscription through a lifecycle:

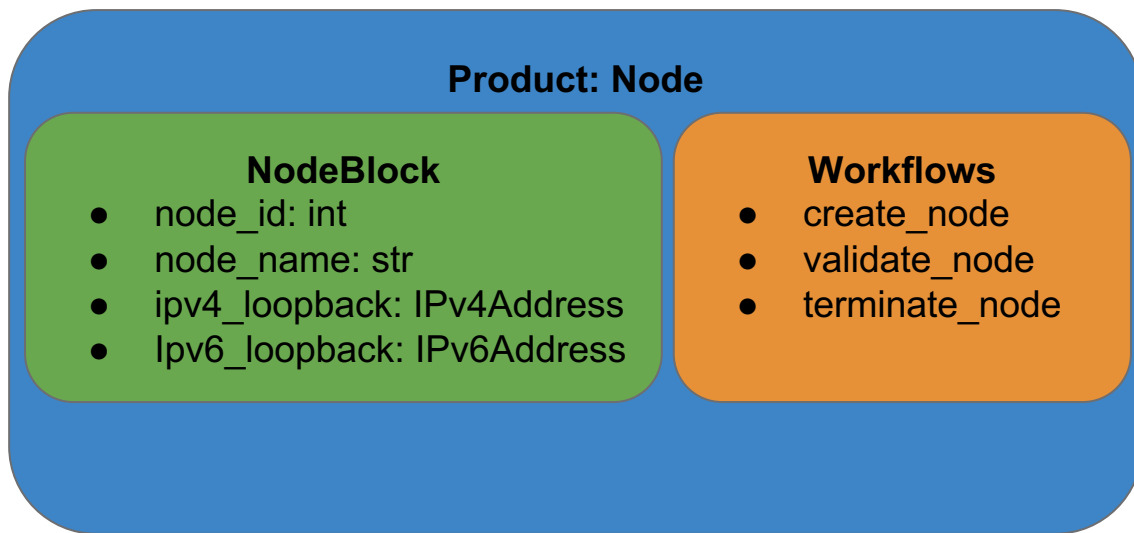
inactive -> provisioning -> active -> terminated



Example domain models: diagrams

Node: represents a router, switch, or transponder

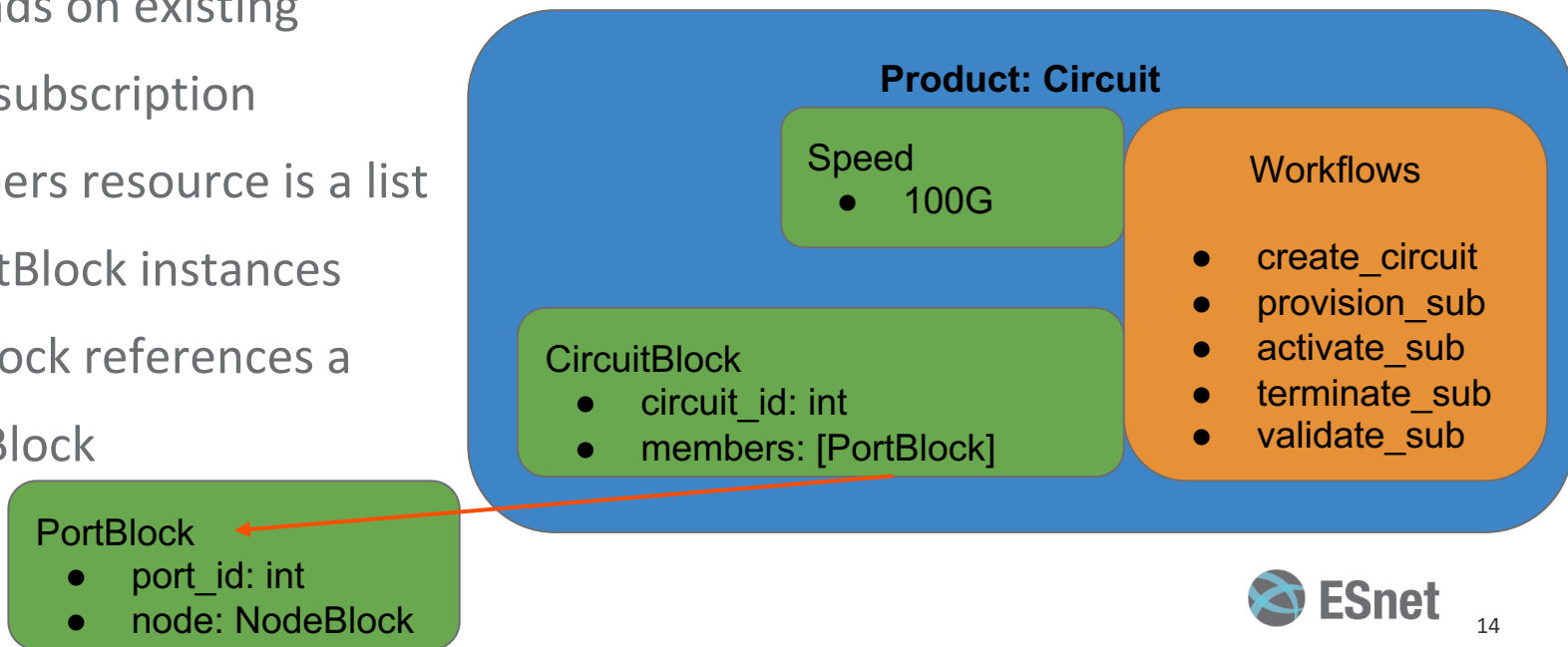
- Customer subscribes to node product
- Engineer uses workflows to instantiate, manage, and terminate subscription



Example domain models: diagrams

Circuit: represents a layer 3 connection to a customer

- Depends on existing Node subscription
- members resource is a list of PortBlock instances
- PortBlock references a NodeBlock



Example domain models: the code

Node

Product → `class Node(NodeProvisioning, lifecycle=[SubscriptionLifecycle.ACTIVE]):`
Product block → `node: NodeBlock`

Product block → `class NodeBlock(NodeBlockProvisioning, lifecycle=[SubscriptionLifecycle.ACTIVE]):`
Fields → `"""Node with optional fields to use in the active lifecycle state."""`
`node_id: int`
`node_name: str`
`ipv4_loopback: IPv4Address`
`ipv6_loopback: IPv6Address`

Circuit Example

Product

Product blocks

Product block

Fields

```
class PortPair(SubscriptionInstanceList[T]):  
    min_items = 2  
    max_items = 2
```

```
class Layer3InterfaceInactive(  
    ProductBlockModel, product_block_name="Layer 3 Interface"  
):  
    port: PortInactive  
    v6_ip_address: IPv6Interface | None = None
```

```
class PortInactive(ProductBlockModel, product_block_name="Port"):  
    port_id: int | None = None  
    port_description: str | None = None  
    port_name: str | None = None  
    node: NodeBlock | None = None
```

```
class Speed(strEnum):  
    HUNDREDG = "100G"
```

```
class CircuitInactive(SubscriptionModel, is_base=True):  
    # Equipment state is planned  
    speed: Speed  
    circuit: CircuitBlockInactive
```

```
class CircuitProvisioning(  
    CircuitInactive, lifecycle=[SubscriptionLifecycle.PROVISIONING]  
):  
    speed: Speed  
    circuit: CircuitBlockProvisioning
```

```
class CircuitBlockInactive(ProductBlockModel, product_block_name="Circuit"):  
    """Object model for a Circuit as used by  
    Backbone Link Service"""  
    members: PortPair[Layer3InterfaceInactive]  
    circuit_id: int | None = None  
    under_maintenance: bool | None = None
```

```
class CircuitBlockProvisioning(  
    CircuitBlockInactive, lifecycle=[SubscriptionLifecycle.PROVISIONING]  
):  
    """Circuit with fields for use in the provisioning lifecycle"""  
    members: PortPair[Layer3InterfaceProvisioning]  
    circuit_id: int  
    under_maintenance: bool
```


Tips for building your own domain models

- Try to think in real-world, reusable blocks
 - Node, Server, Circuit are basic, reusable concepts
 - NokiaServer, ECMP_Circuit, ManagementPort are less reusable constructs that can be constructed from simpler blocks
- Enumerate types of a basic component
 - Most offerings can be encapsulated by some combination of Circuit, Node, and Port
 - Adding enumerations for things like Speed, CircuitType allows for making several products with the same blocks
- If something changes, you can always modify the domain model via a migration

Updating existing domain models

- Domain models are stored in a postgres database
- They are created, updated, and removed by postgres migrations
- To change a domain model:
 - make the change in its representation in its `product_type` and `product_block` files
 - use migration tool to generate a migration to effect the change:

```
> python main.py db migrate-domain-model "migration_name"
```