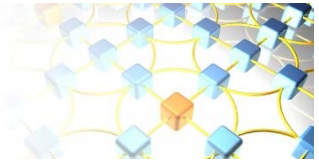
POLICY-BASED MANAGEMENT FOR NETWORK SLICING

Dr. John Strassner

Orchestration Area Director, MEF

Policy Definition

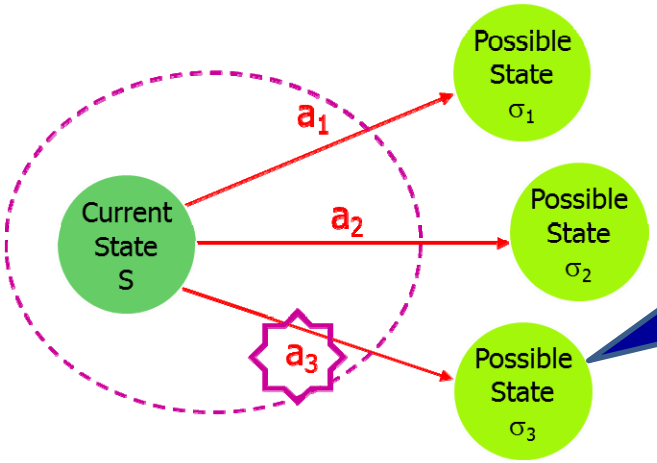


- **Policy is Old…**
  - "Policies are rules governing the choices in behavior of a system"

    Sloman, 1994 [1]
  - "Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects." Strassner, 2003 [2]
- **Why We Care** Ref [3]
  - Reduces OpEx and TTM
  - Devices and systems will not, in general, be autonomic – but with appropriate management and orchestration, devices and systems can **appear** to function autonomically

Types of Policy Rules (1)

- **Imperative: Event-Condition-Action (ECA)**
  - IF the clause of Events evaluates to TRUE
    - IF the clause of Conditions evaluates to TRUE
      - THEN execute the clause of Actions
  - Explicit programming of state (rationality is compiled into the policy!)



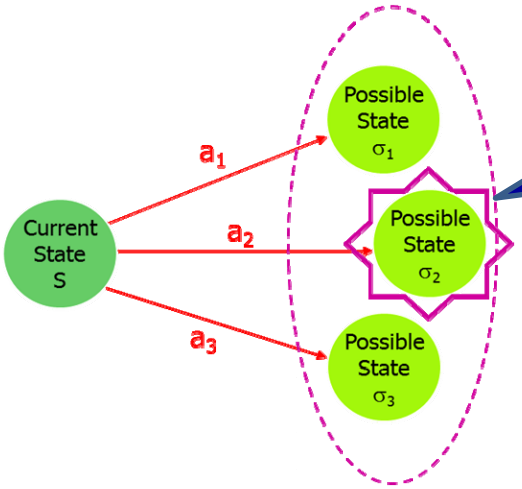
Advantages:
- Can be simple; system knows exactly what to do

Disadvantages:
- Explosion of policies
- Conflict detection and resolution can be very difficult

Ref [4][12]

Types of Policy Rules (2)

- **Declarative (or Goal-based)**
  - Express ***what*** should be done, ***not how to do it***
  - Specifies criteria for choosing a set of states
  - Each state has a binary value
  - Rationality is generated by optimizer/planner
  - *Typically implemented as a logic program*



Advantages:
- More abstract, and potentially more flexible, than ECA policies

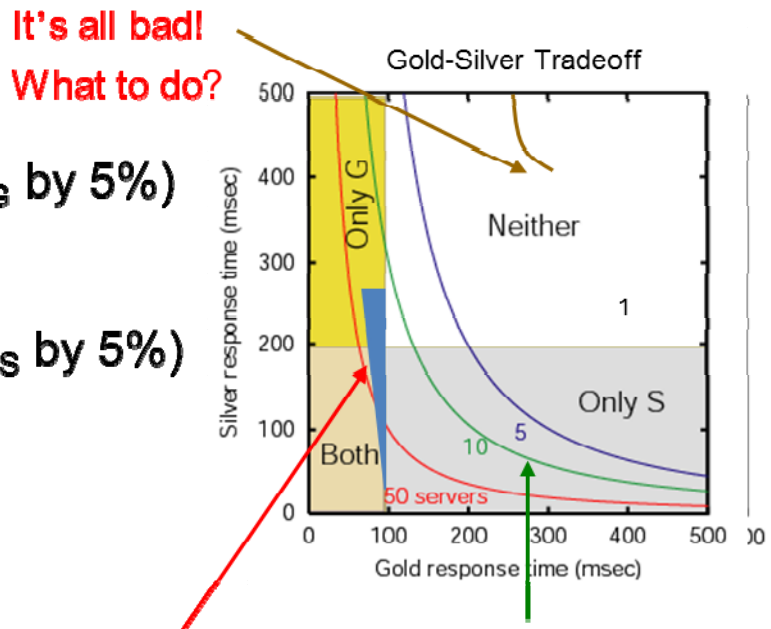
Disadvantages:
- Requires sophisticated translation and optimization modules

Ref [4][12]

Why Multiple Types of Policy Rules?

Gold: IF (RT_G > 100 msec)
THEN (Increase CPU_G by 5%)

Silver: IF (RT_S > 200 msec)
THEN (Increase CPU_S by 5%)



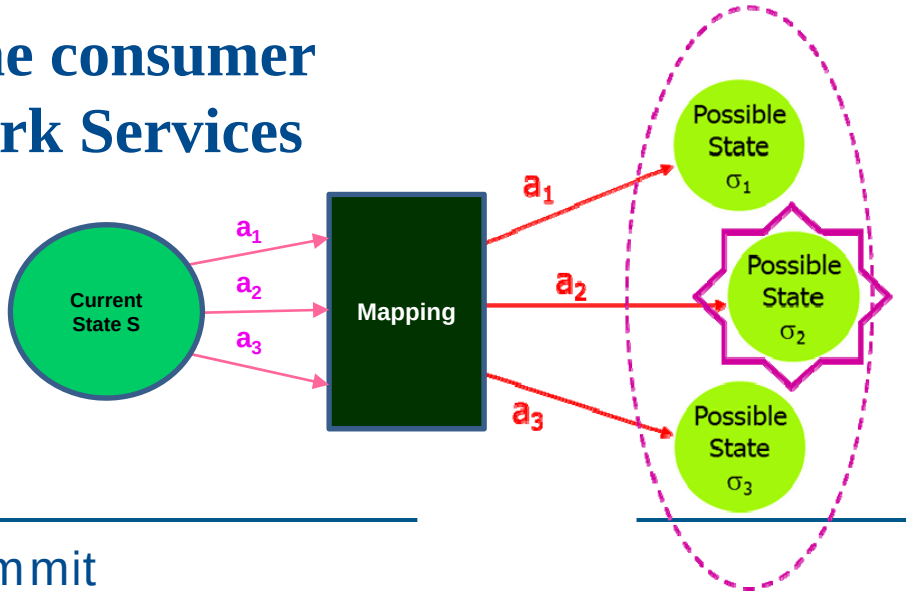
Do we always want to satisfy Gold at all expense?

- Better to partially satisfy all classes?
- Better to satisfy both Silver and other service classes at expense of Gold?

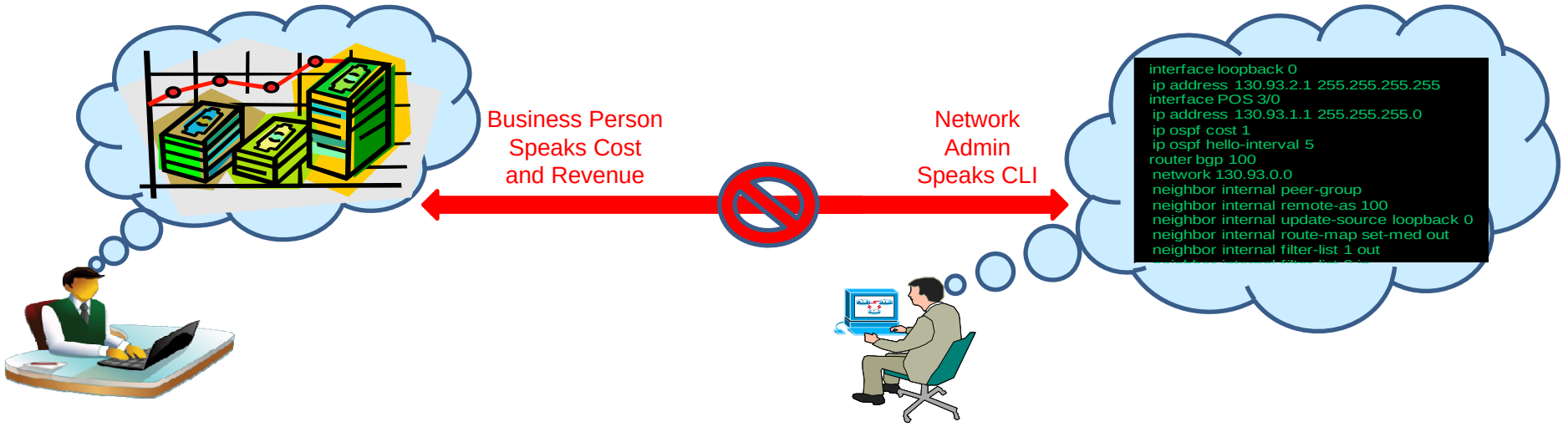
Ref [4]

The Reinvention of Intent

- **Policy Management is HARD**
  - People want simpler solutions
- **Many Different Constituencies Want Intent**
  - End Users who aren't technical want to define policies to control behavior
  - Application Developers want to build Network Services, but existing network interfaces don't help them do this
  - Operators want more abstract and powerful ways to define Network Services
- **Intent offers the ability to define consumer abstractions that invoke Network Services**
- **Intent is a *Declarative Policy*, but *not necessarily logic-based***
- ***Intent requires a Mapping***



Different Meanings of Policy



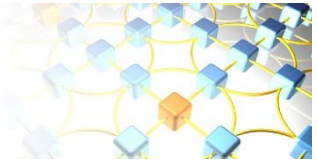
Ref [4][5][12]

Constituencies: The Policy Continuum

Ref [6][12]

Policy Management is HARD

- **Most of the literature covers imperative policies**
  - These are simple in theory
  - Difficult in practice, due to conflict detection and resolution
- **Virtually no literature covers combining different types of policy paradigms**
  - This is the focus of SUPA [11] and MEF [12]
- **Scalable Policy Architectures are hard**
  - Being explored in the MEF now
- **Unless Policy is integrated into the models used for control and management, it will fail**



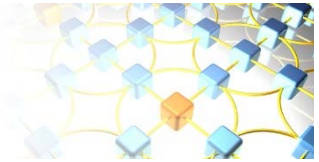
Slice Definition

- **A Slice is a collection of resources that supports a set of services**
  - It has technical features, but is driven by business needs

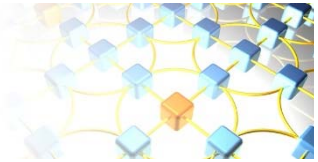
- **Slice characteristics**

- Resources may be physical, logical, or virtual
- Resources provide compute, storage, and networking functions
- Resources are chosen to satisfy the features and behavior of, and the constraints on, the service(s) supported
- Resources in the slice are orchestrated to provide a set of services
- Resources in the slice may be fully or partially isolated from other slices
- APIs should be designed as model-driven, modular, reusable building blocks

Slicing Features



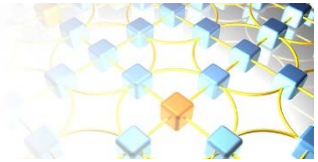
- **Devices need different QoS and QoE from the network**
- **Slices should be able to be dynamically provisioned**
- **Slices might need special characteristics**
  - Isolation (e.g., for emergencies)
  - Large events might require large numbers of HD streams
- **Multiple ways to govern slices**
  - UE perspective groups by similar user characteristics
  - Network perspective groups by similar resource allocation
  - SLA perspective groups by similar SLA requirement



What Does This Tell Us?

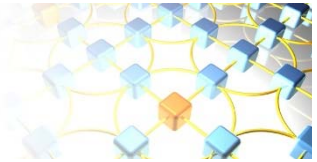
- **Need a mechanism to select the right resources**
- **Must understand the underlying business reason for a slice**
- **Need to understand the context that the slice is operating in**

- **MEF Approach**
  - Management and Orchestration must be *model-driven*
  - The model needs to do as good a job modeling business concepts as it does modeling resources and services
  - We need a mechanism to *advertise capabilities* and *impose constraints*
  - Orchestration uses Policy to make its decisions
  - Orchestration is multi-layer and distributed (not a single God-Box)



Policy in the MEF

- **A Policy could be used to build and modify ACLs**
  - A Policy is NOT the ACL itself
- **For North-South, or hierarchies in general:**
  - Policies *manage* behavior, and must be *hierarchical in scope*
- **For East-West:**
  - Policies *negotiate* (e.g., request and offer, but not *control*) behavior
- **We need a *Common Management Abstraction***
  - Policies are selected based on a 3-tuple:{Context, Capabilities, Constraints}
  - Metadata can be attached to each element in the 3-tuple
  - Context selects policies based on applicability
  - Capabilities describe what the policy does
  - Constraints restrict the capabilities offered and/or the behavior of the policy



Summary

- **Policy is still misunderstood**
  - *The building and modification of ACLs, NOT the ACL*
- **Policy has been used like hammer**
  - *Sometimes you need more than code*
- **Policy must be adaptive, like slices**
  - *So why do we still talk APIs?*
- **We need a management abstraction for Policy**
  - *Then we can work on abstractions for Orchestration*

References

[1] M. Sloman, “Policy Driven Management for Distributed Systems”, JNSM, v2, No 4, 1994

[2] J. Strassner, “Policy Based Network Management”, Morgan Kaufman, ISBN 978-1558608597, Sep 2003

[3] IETF ANIMA WG and FOCALE (e.g., [4], [8])

[4] J. Strassner, “Autonomic Management”, NOMS 2010 Tutorial

[5] J. Strassner, “Business-Driven Policy Management”,

[6] S. Davy, B. Jennings, J. Strassner, “The Policy Continuum –Policy Authoring and Conflict Analysis”, Computer Communications Journal, Elsevier, Volume 31, Issue 13, pages 2981-2995, August 2008

[7] J. Strassner, N. Agoulmine, E. Lehtihet, “FOCALE –A Novel Autonomic Networking Architecture”, International Transactions on Systems, Science, and Applications (ITSSA) Journal, Vol. 3, No 1, pp 64-79, May, 2007

[8] K. Barrett, S. Davy, J. Strassner, B. Jennings, S. van der Meer, “Model Based Generation of Integrated Suites of Languages and Tools for Policy Specification, Analysis and Deployment”, IEEE Global Information Infrastructure Symposium, 2007

[9] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, J. Strassner, M. Ó Foghlú, W. Donnelly, J. Strassner, “Towards Autonomic Management of Communication Networks”, IEEE Communications Magazine, Vol 45., No 10, pp 112-121, Oct 2007

[10] J. Strassner, S. van der Meer, D. O’Sullivan, S. Dobson, “The Use of Context-Aware Policies and Ontologies to Facilitate Business-Aware Network Management”, JNSM (17), pp 255-284, 2009

[11] J. Strassner, J. Haplern, S. van der Meer, “Generic Policy Information Model for Simplified Use of Policy Abstractions (SUPA)”, draft-ietf-supa-generic-policy-info-model-02, Jan 2017

[12] MEF, Policy-Driven Orchestration Project

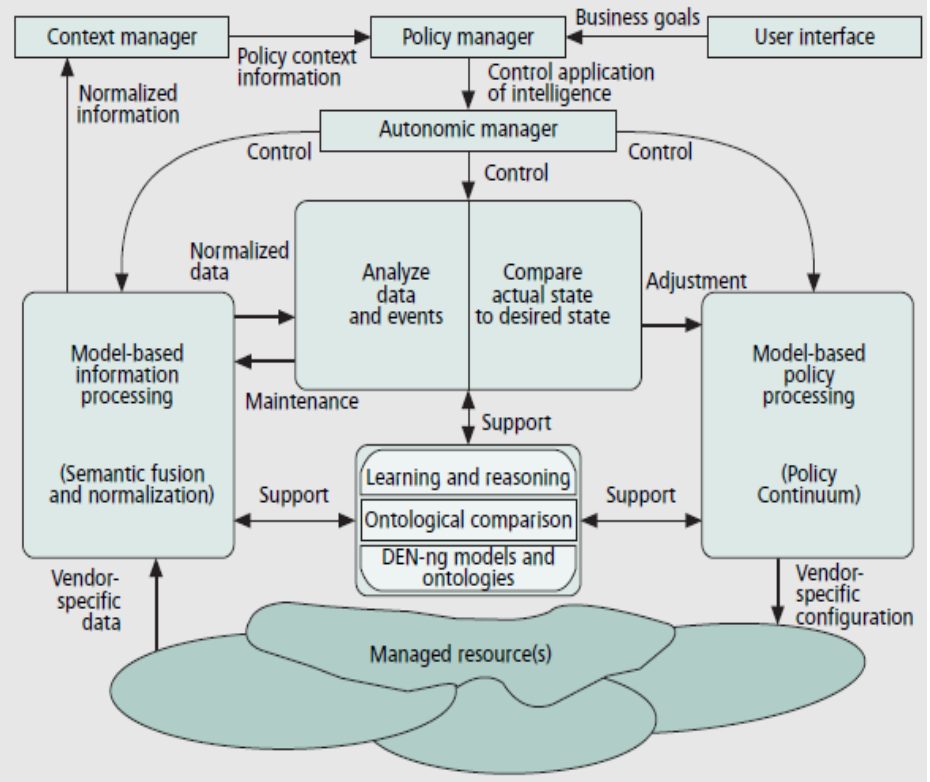
Questions?

“Create like a god. Command like a king. Work like a slave”

- Constantin Brancusi

The FOCALE Autonomic Architecture

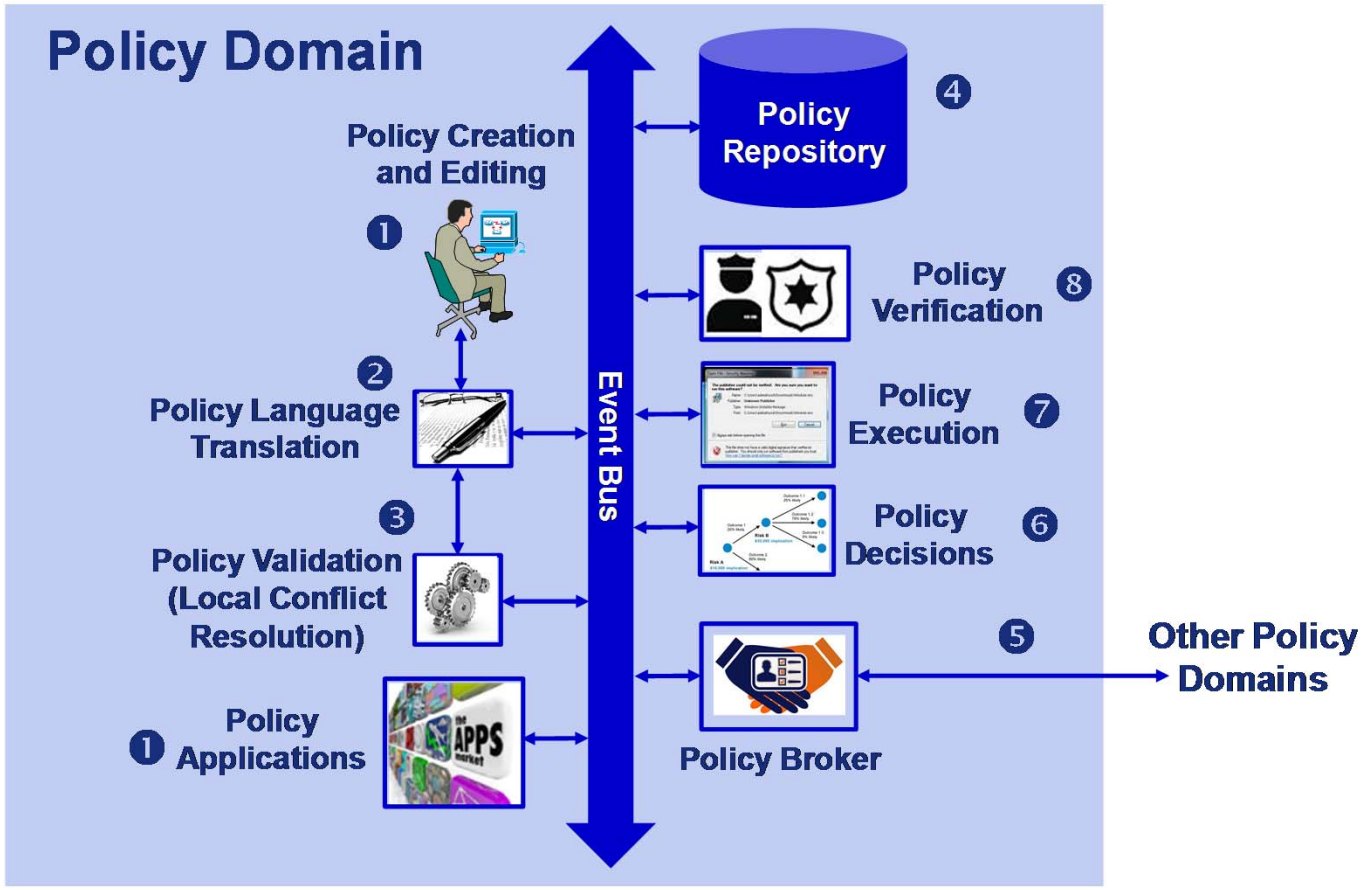
- **FOCALE Was Designed for Autonomic Policy Management**
  - FOCALE abstracts the functionality and behavior of the system being managed using models, ontologies, and logic
  - Context selects policies, which define behavior; as context changes, policies change
  - Input state is extracted/inferred from OAMP data and context and compared to desired state
    - If they are equal, continue monitoring
    - If not, determine set of actions to return to desired state
  - Machine learning observes actions taken and dictated by admins to continuously improve knowledge base



Ref [7][9]

Exemplary Policy Elements

Note: This is a FEDERATED Policy Model; Policy Domains are present at each architectural plane



Ref [4][6][7][8][9][10]