Autonomic Functions Life-cycle and Management

draft-peloso-anima-autonomic-function-01
draft-ciavaglia-anima-coordination-01

IETF96 – Berlin

Pierre Peloso, Laurent Ciavaglia

Motivations

- Converge towards **common guidelines to deploy and operate autonomic functions**
  - because different types of AFs in a same AN
  - because AFs/ASAs from different vendors/developers
  - because AFs interworking among themselves and with ANI functions

Benefits

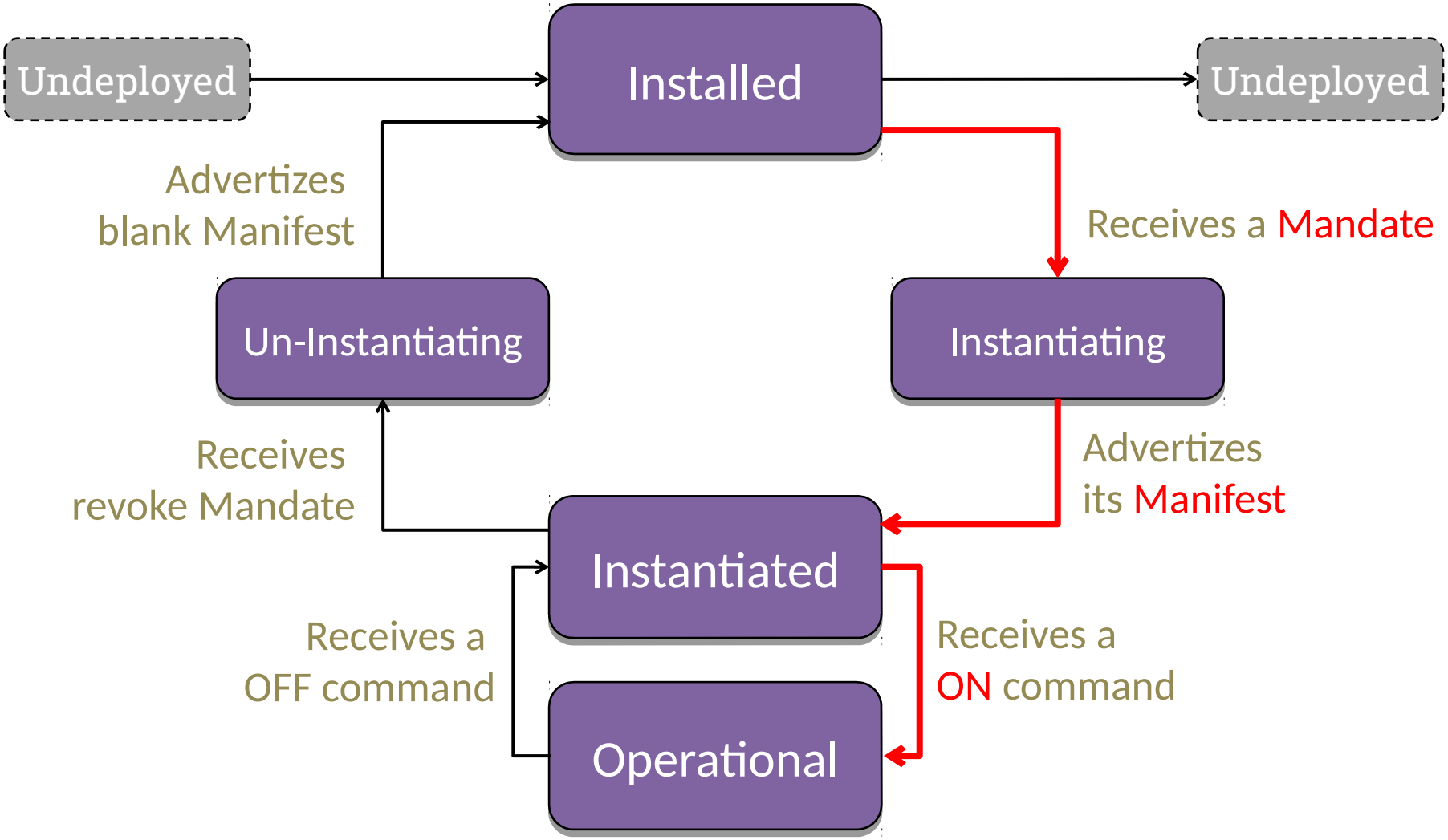
- Help Operators to **concentrate on achieving the goals** of the functions deployed, **not how** to plan, deploy and operate them.

- Help Developers to **concentrate on the core functionality**, not how to interface with the network infrastructure.

Proposal

- Main elements:
  - The lifecycle
  - The information types, flow and data structures
    - ASA Class Manifest
    - ASA Instance Mandate
    - ASA Instance Manifest

ASA Life Cycle



States explained

Undeployed

the Autonomic Function Agent is a mere piece of software, which may not even be available on any host.

Installed

the Autonomic Function Agent is available on a (/multiple) host(s), and after having shared its ASA class Manifest (which describes in a generic way independently of the deployment how the ASA would work). In this state the ASA is waiting for an ASA Instance Mandate, to determine which resources it manages (when the ASA is strictly coupled to resources [e.g. part of a Node OS], there is no need to wait for an instance mandate, the target resources being intrinsically known).

Instantiated

the Autonomic Function Agent has the knowledge of which resources it is meant to manage. In this state the ASA is expecting a "on" message in order to start executing its autonomic loop. From this state on the ASA can share an Instance Manifest (which describes how the ASA instance is going to work).

Operational

In this state, ASAs are executing their autonomic loop, hence acting on network, by modifying resources parameters. A "off" message will turn back the ASA in an Instantiated state

ASA Class Manifest

- An ASA class is a piece of software that contains the computer program of an Autonomic Function Agent.

- In order to install and instantiate appropriately an autonomic function in its network, the operator needs to know which are the characteristics of this piece of software.

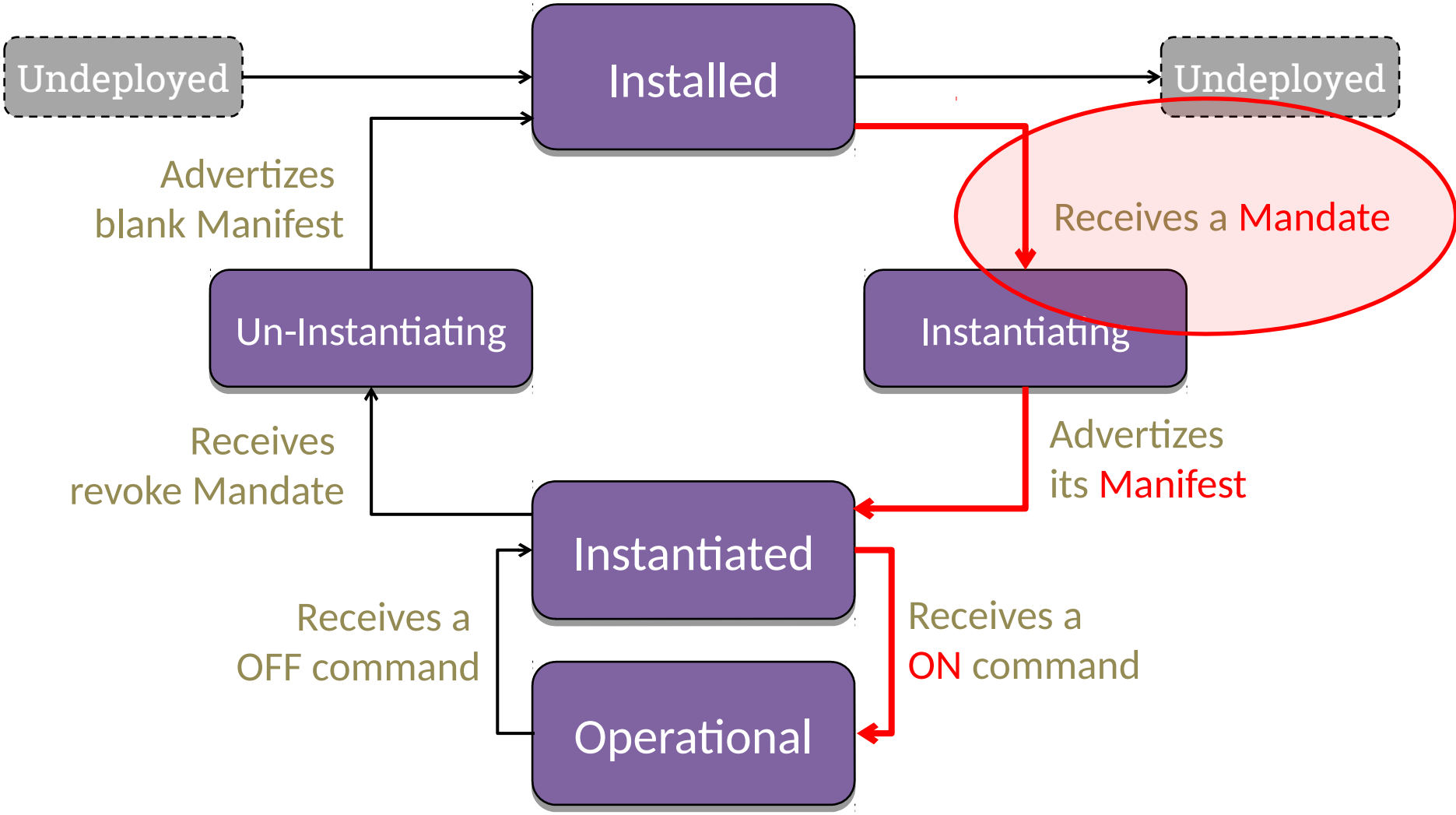
- An ASA class manifest, which is (a machine-readable) description of both the autonomic function and the piece of code that executes the function.

ASA Class Manifest

		is dynamically installable)
Network Segment	NetSegment...	Lists the network segments on which the autonomic function is applicable (e.g. IP backbone versus RAN).
Manageable Equipments	Equipments...	Lists the nodes/resources that this piece of code can manage (e.g. ALU 77x routers, Cisco CRS-x routers, Huawei NEXE routers).
Autonomic Loop Type	Enum	States what is the type of loop MAPE-K and whether this be halted in the course execution.
Acquired Inputs	Raw InfoSpec...	Lists the nature of info that an ASA agent may a the managed resource(s) links load).
External Inputs	Raw InfoSpec...	Lists the nature of info that an ASA agent may r from other ASA in the e that could provide such information/knowledge.
Possible Actions	Raw ActionSpec	Lists the nature of act an ASA agent may enforc the managed resource(s)

```
            <contentType>Numeric</contentType>
            <informationUsage>External Optional</informationUsage>
            <type>Knowledge</type>
         </ManagementInfoSpecification>
         <ManagementInfoSpecification>
            <descriptor>Prediction of router interface load</descriptor>
            <contentType>Numeric</contentType>
            <informationUsage>External Optional</informationUsage>
            <type>Knowledge</type>
         </ManagementInfoSpecification>
      </OptionalExternalInputs>
      <PossibleActions>
         <ManagementActionSpecification>
            <descriptor>Switch ON/OFF Ethernet port</descriptor>
            <contentType>Boolean</contentType>
            <controlFlexibility>{Enabled, Disabled, Intercepted}</controlFlexibility>
         </ManagementActionSpecification>
         <ManagementActionSpecification>
            <descriptor>Switch ON/OFF Ethernet port</descriptor>
            <contentType>Boolean</contentType>
            <controlFlexibility>{Enabled, Disabled, Intercepted}</controlFlexibility>
         </ManagementActionSpecification>
         <ManagementActionSpecification>
            <descriptor>Switch ON/OFF IP interface</descriptor>
            <contentType>Boolean</contentType>
            <controlFlexibility>{Enabled, Disabled}</controlFlexibility>
         </ManagementActionSpecification>
         <ManagementActionSpecification>
            <descriptor>Change metric of IP interface</descriptor>
            <contentType>Numeric</contentType>
            <controlFlexibility>{Enabled, Disabled, Constrained}</controlFlexibility>
         </ManagementActionSpecification>
      </PossibleActions>
      <ConfigurationOptions>
         <SpecificNEMPolicySpec>
            <Name>Condition_00001_MPLS_TE</Name>
            <Description>Load Threshold reached to trigger MPLS TE</Description>
            <Event>OnGovPolicyUpdate</Event>
            <Conditions>
               <ConditionAtomic>
                  <ConditionVariable>CurrentLoad</ConditionVariable>
                  <Operator opType="2"/>
                  <DefaultValue>0.2</DefaultValue>
               </ConditionAtomic>
```

ASA Life Cycle



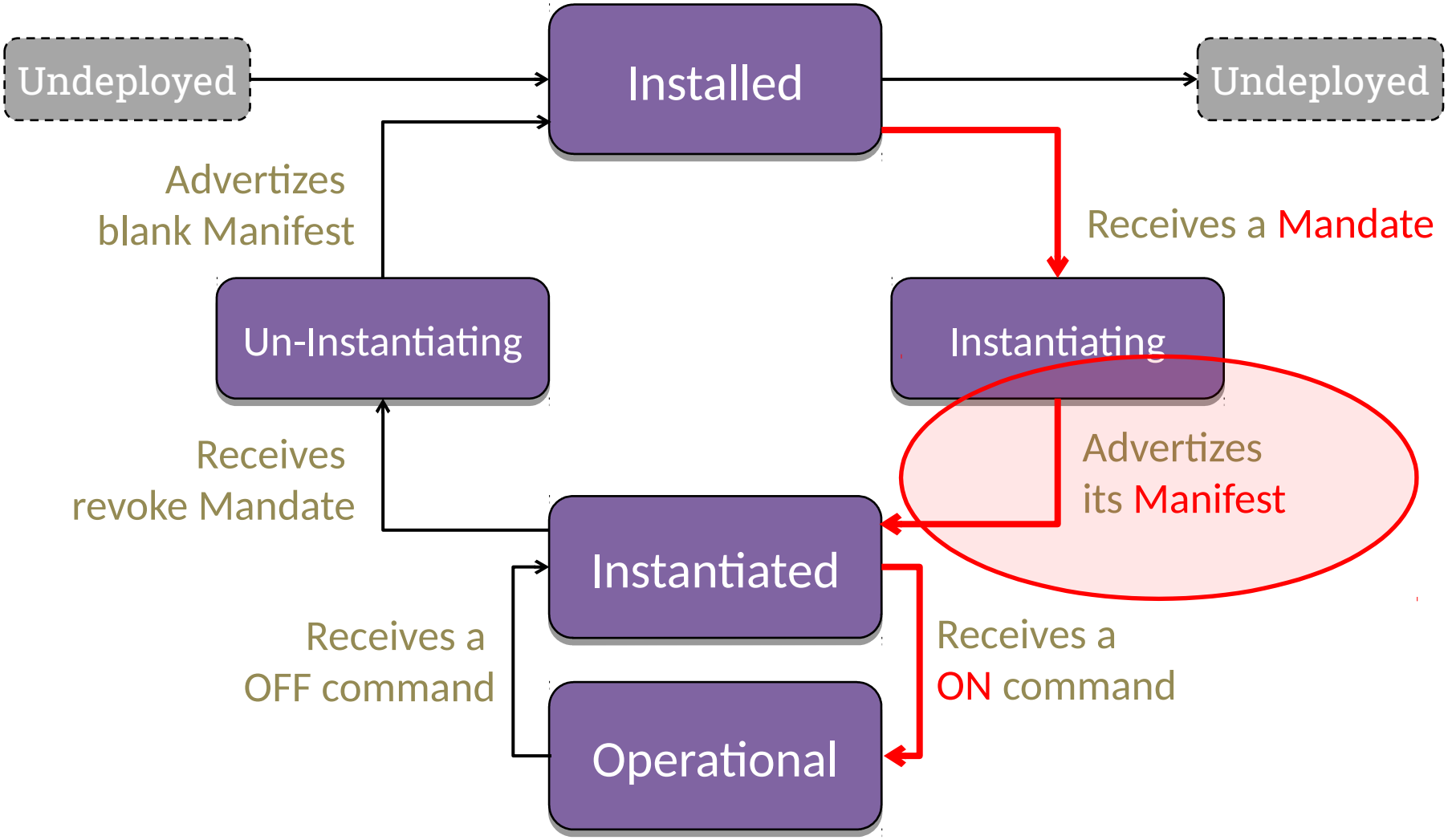
ASA Instance Mandate

- An ASA instance is the ASA agent: a running piece of software of an ASA class. A software agent is a persistent, goal-oriented computer program that reacts to its environment and interacts with other elements of the network.
- In order to install and instantiate appropriately an autonomic function in its network, the operator may specify to ASA instances what they are supposed to do: in term of which resources to manage and which objective to reach.

- An ASA Instance Mandate is (a machine-readable) set of instructions sent to create autonomic functions out of ASAs.

- An ASA instance mandate could be either:
  - sent to a targeted ASA In which case, the receiving Agent will have to manage the specified list of resources,
  - broadcast to all ASA In which case, the ASAs would collectively determine which agent would handle which resources from the list, and if needed (and feasible) this could also trigger the dynamic installation/instantiation of new agents (ACP should be capable of bearing such scenarios).

ASA Life Cycle



ASA Instance Manifest

- Once the ASAs are properly instantiated, the operator and its managing system need to know which are the characteristics of these ASAs.

- An ASA instance manifest is (a machine-readable) description of either an ASA or a set of ASAs grouped as an autonomic function.

Control when an ASA runs

- To control when an ASA runs (and possibly how it runs), the operator needs the capacity to start and stop ASAs.
- Additionally this type of message could also be used to specify how the ASA is meant to run, e.g. whether its control loop is subdued to some constraints in terms of pace of execution or rhythm of execution (once a second, once a minute, once a day…)

Know what an ASA does to the network

- To know what an ASA does to the network, the operator needs to have the information of the elements either monitored or modified by the ASA, hence this ASA should disclose those.
- The disclosing should take the form of a ASA Instance Manifest.

Decide which ASA control which equipment

- To determine which ASA controls which equipment (or vice-versa which equipments are controlled by which ASAs), the operators needs to be able to instruct ASA before the end of their bootstrap procedure.

- These instructions sent to ASA during bootstrapping should take the format of an ASA Instance Mandate.

Additional “users”

- The lifecycle and related information conveyed is also useful for
  - The coordination function (cf previous presentation)
  - The exchange of information, knowledge creation and knowledge database
  - Other functions of the ANI…?

Summary

- Converge towards **common guidelines to deploy and operate autonomic functions**
- Achieved thanks to **Lifecycle and information types, flow and data structure**
- Help Operators to **concentrate on achieving the goals** and Developers **on the core functionality**
- **Requires better modeling of AF/ASA and interactions with ANI functions**