a running implementation of an autonomic networking framework

IETF 88, NMRG, Vancouver

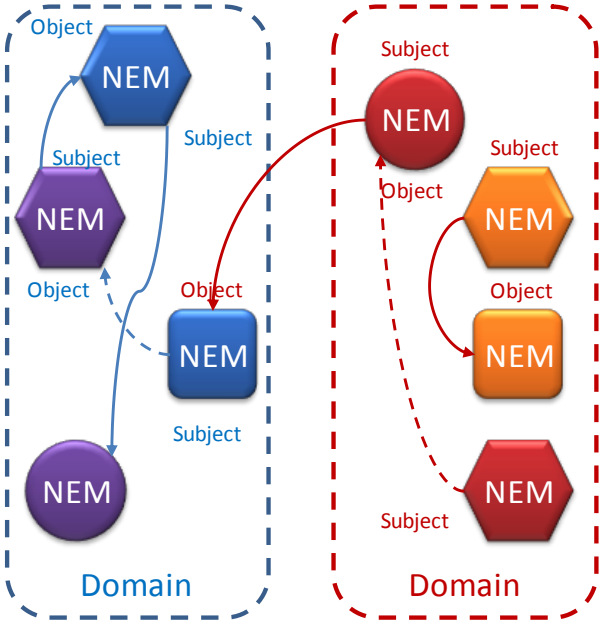
Laurent Ciavaglia

WWW.UNIVERSELF-PROJECT.EU

STARTING POINT

How to cope with networks ecosystem diversity?

- Multiple types of autonomic function
- Multiple technologies
- Multiple roles, interactions, relationships



Propositions:

- Common "model" for autonomic functions
  → **Network Empowerment Mechanism (NEM)**

- Common "utility" functions
  → **Unified Management Framework (UMF)**

- UMF manages NEMs which (autonomously) control network resources

- Put it in practice (cf. project demo track records) and share it

FRAMEWORK AND EMPOWERMENT MECHANISMS

NETWORK EMPOWERMENT MECHANISM (NEM)

NEM = an autonomic function controlling network resources

NEM = use of relevant method to solve a concrete operational problem in a specific networking environment

- e.g. use of genetic algorithm for interference coordination in LTE networks

UNIFIED MANAGEMENT FRAMEWORK (UMF)

NEM SKIN = unified abstraction of NEM

- common set of objects describing its properties and capabilities
  e.g. manifest, mandate, instance description
- common set of interfaces to connect and interact with the UMF and other NEMs

NEM SKIN = vector of unification, re-usable software component, and an accelerator for NEM implementation

FRAMEWORK AND EMPOWERMENT MECHANISMS

NETWORK EMPOWERMENT MECHANISM (NEM)

One framework to manage multiple/any types of NEMs

Ability to cope with NEM ecosystem diversity

- heterogeneity of NEM function/goal
- multiple technology domains
- multiple roles and interactions among NEMs

with same model and interfaces (skin)

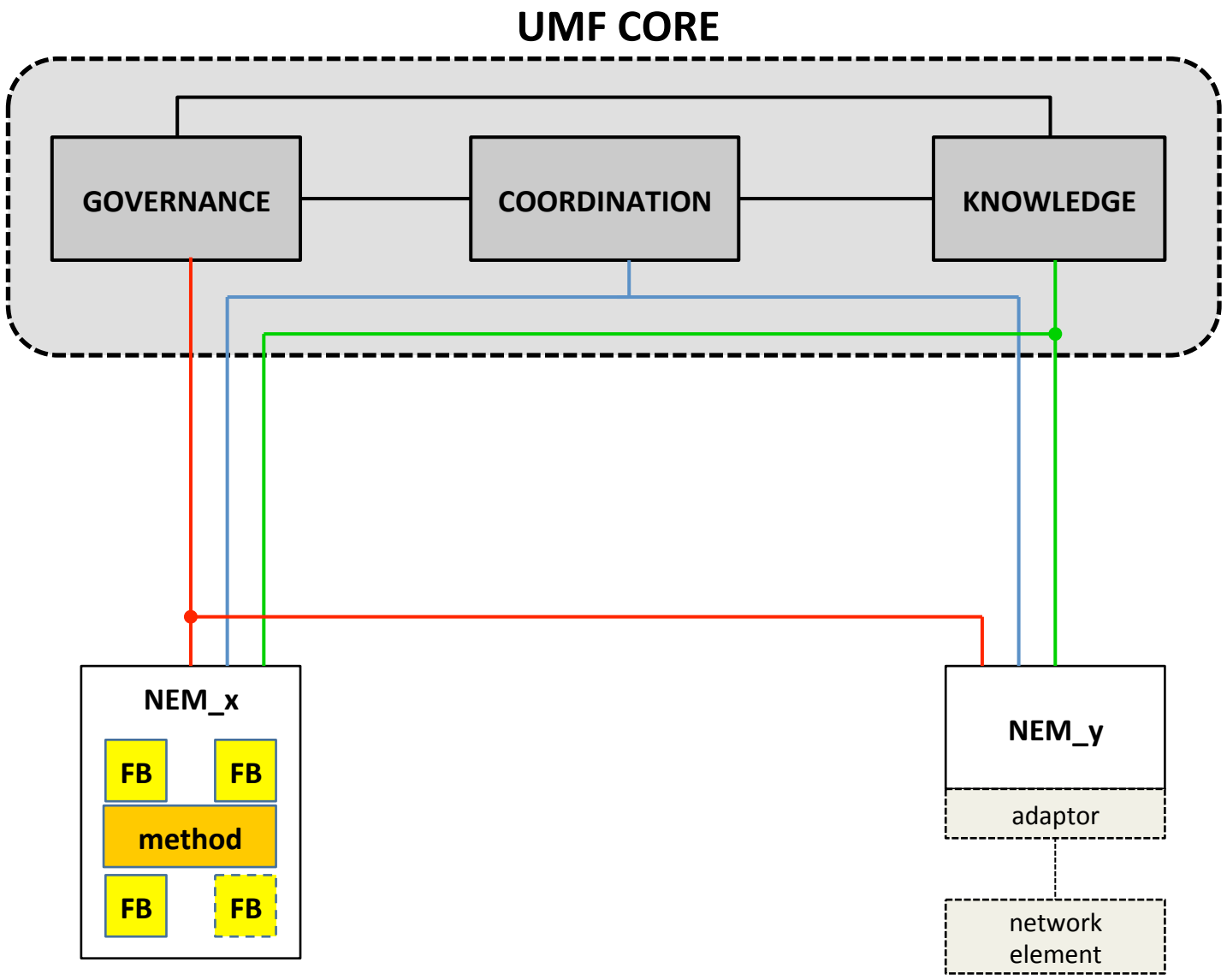
UNIFIED MANAGEMENT FRAMEWORK (UMF)

Specification of core utility functions, workflows and NEM lifecycle

3 core functions:

- governance, coordination, knowledge
- and associated mechanisms e.g. conflict avoidance, data mining…

FRAMEWORK AND EMPOWERMENT MECHANISMS

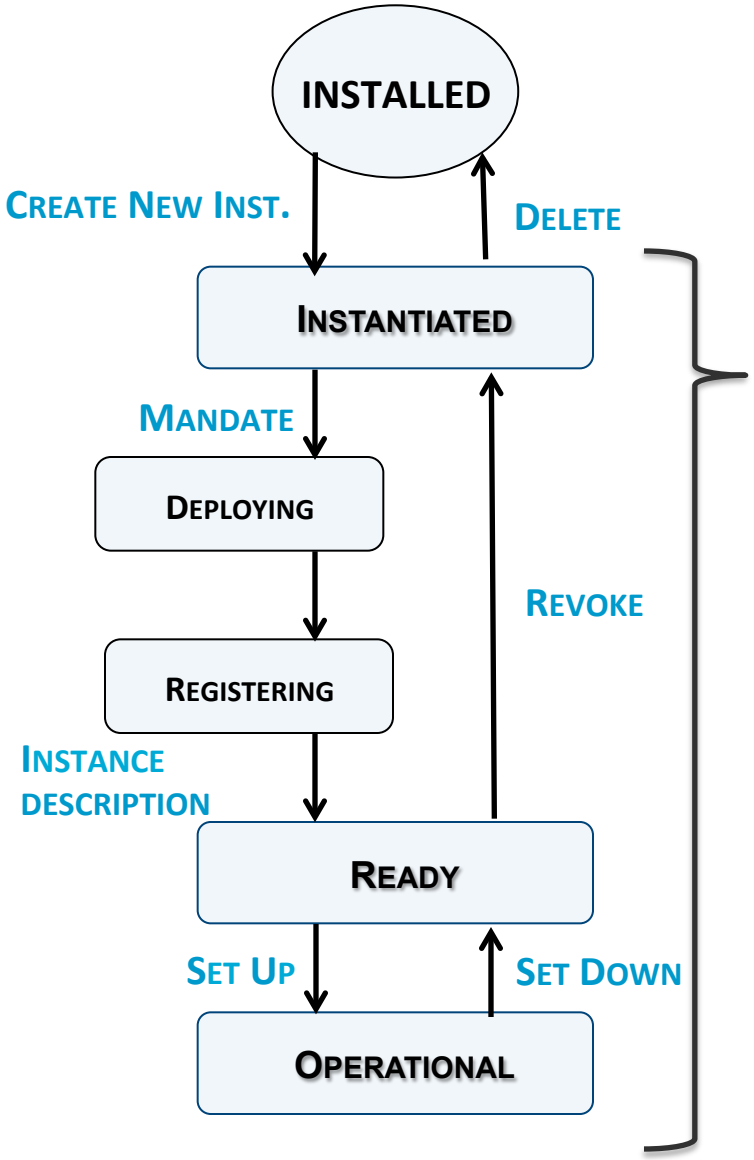


UMF IN A NUTSHELL
UMF CORE FUNCTIONAL BLOCKS

Seamless deployment and trustworthy interworking of NEMs require:

- Tools for the operators to deploy, pilot, control and track progress of NEMs in a unified way
  - **GOVERNANCE functional block**
- Tools to identify/avoid conflicts and ensure stability and performance when several NEMs are concurrently working
  - **COORDINATION functional block**
- Tools to make NEMs find, formulate and share relevant information to enable or improve their operation
  - **KNOWLEDGE functional block**
- APIs to enable NEMs “plug and play” deployment, interoperability and monitoring/configuration
  - **NEM Skin**

UMF IN A NUTSHELL
NEM LIFECYCLE



NEM Class (software)
described by MANIFEST (machine readable)

NEM Instance
described by INSTANCE DESCRIPTION

Life-cycle:

Detail the states and transition of a NEM instance, from its being installed, to it running its MAPE autonomic loop.

Steps include all the management by the UMF core functional blocks.

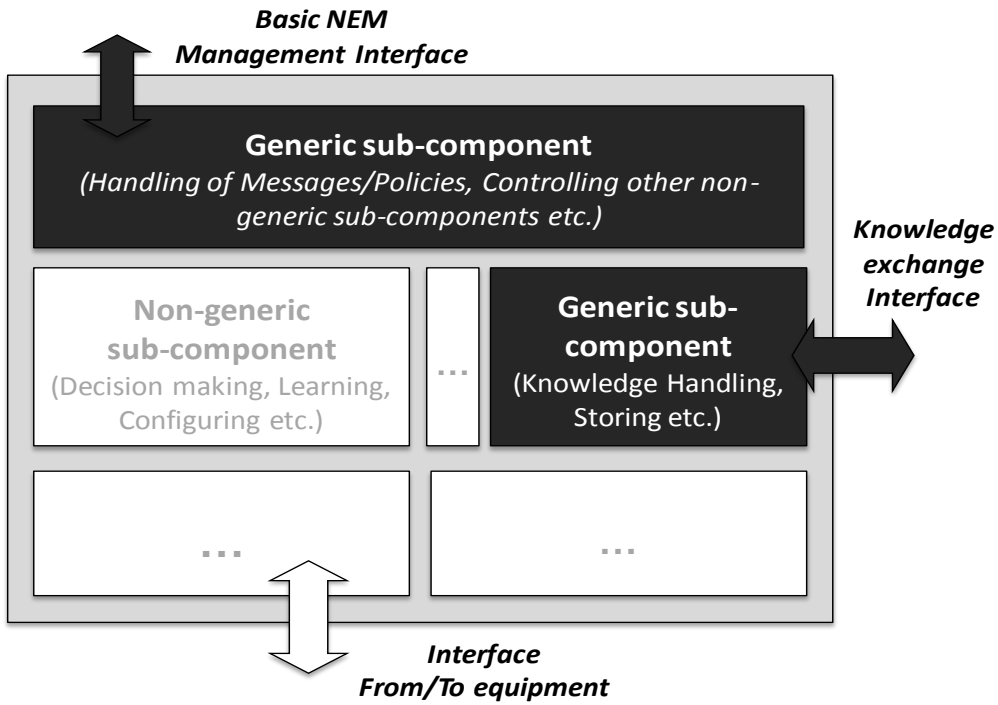
NEM SKIN

Derives from the need for a common base for all NEMs

- The UMF must be able to interact with NEMs for their deployment and operation as well as supply them with required inputs (operator's goals/policies, mandate…)
- Technological heterogeneity is abstracted at the NEM level while the required info/ commands are propagated into the framework in a UMF-compliant way

NEM Skin consists in

- the specifications a NEM must meet (i.e. the interfaces and info exposed to UMF)
- and the means to accomplish this (i.e. a REST-based API targeted for developers)



NEM SKIN Structure Overview

- **Generic NEM functionality**

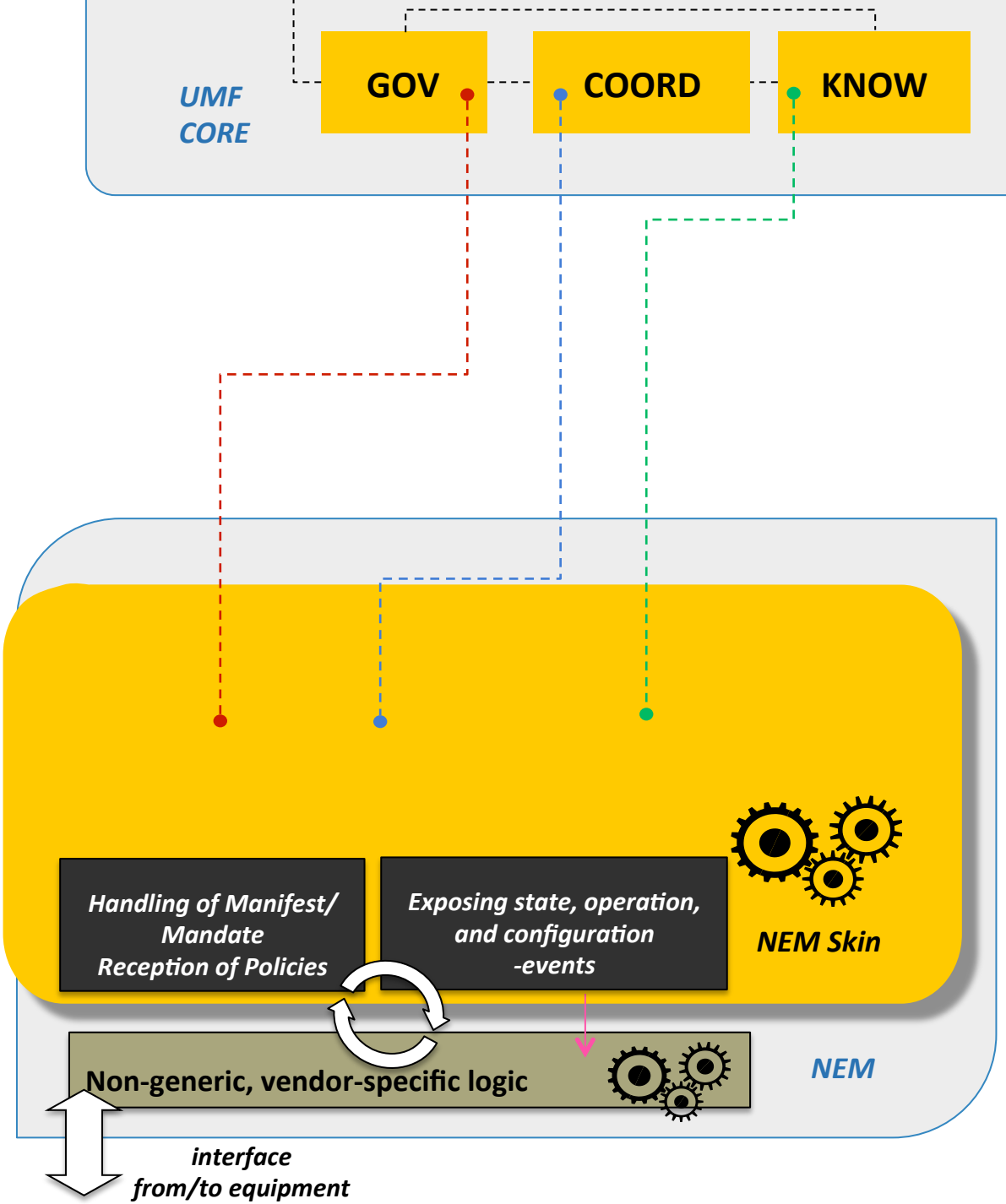
  handling the *manifest*, *mandate*, *registration*, *reception of policies*, *configuration options*, *actions*, *NEM state*

- **UMF compliance**

  logic, interfaces and structure derived from UMF specifications and NEM model specifications

- **NEM lifecycle-related events**

  for the NEM developer and the management from UMF CORE



NEM SKIN Structure Overview

- **Generic NEM functionality**

  handling the *manifest*, *mandate*, *registration*, *reception of policies*, *configuration options*, *actions*, *NEM state*

- **UMF compliance**

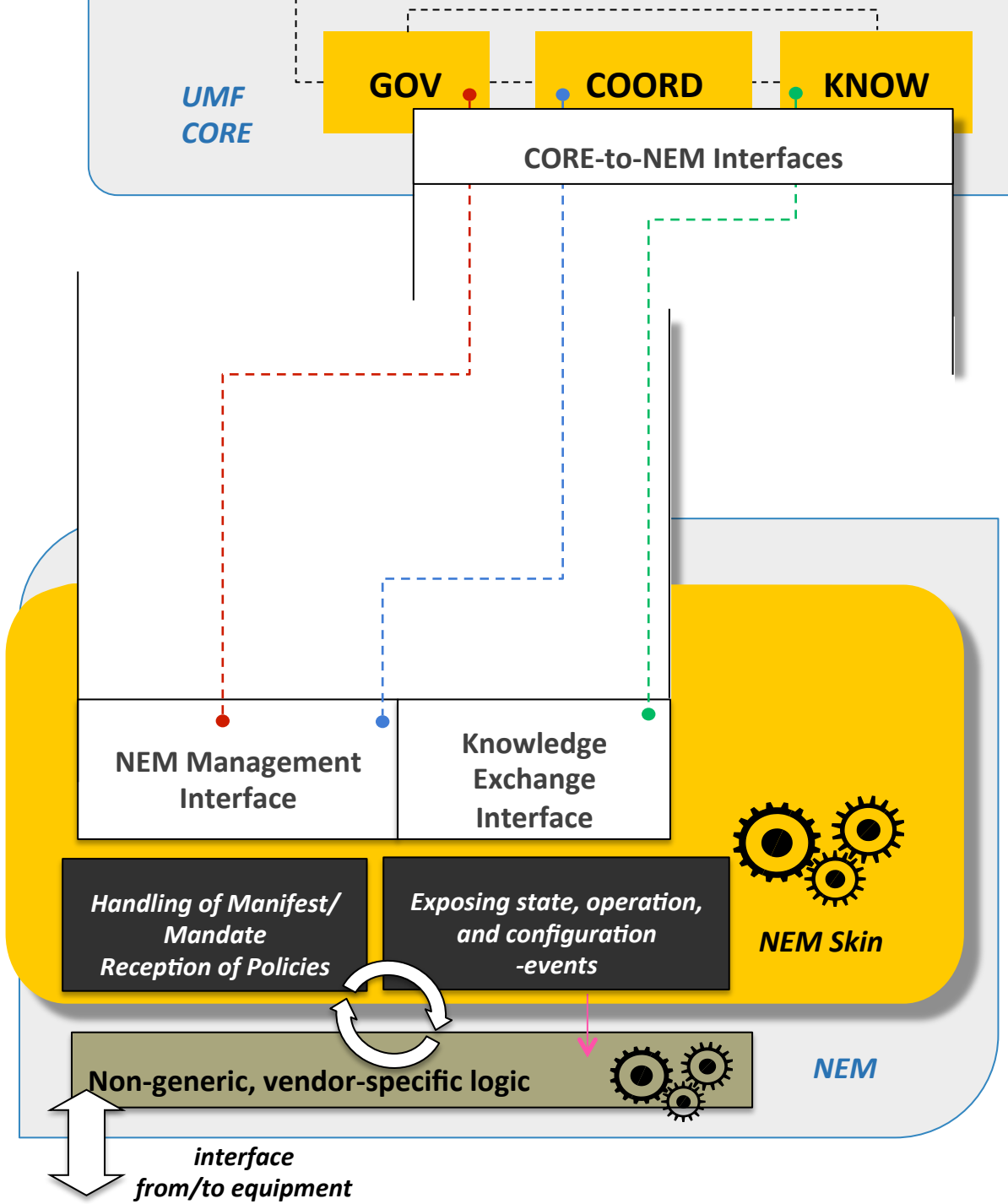
  logic, interfaces and structure derived from UMF specifications and NEM model specifications

- **NEM lifecycle-related events**

  for the NEM developer and the management from UMF CORE

- **Interface definitions**

  method/resource signatures



NEM SKIN Structure Overview

- **Generic NEM functionality**

  handling the *manifest*, *mandate*, *registration*, *reception of policies*, *configuration options*, *actions*, *NEM state*

- **UMF compliance**

  logic, interfaces and structure derived from UMF specifications and NEM model specifications

- **NEM lifecycle-related events**

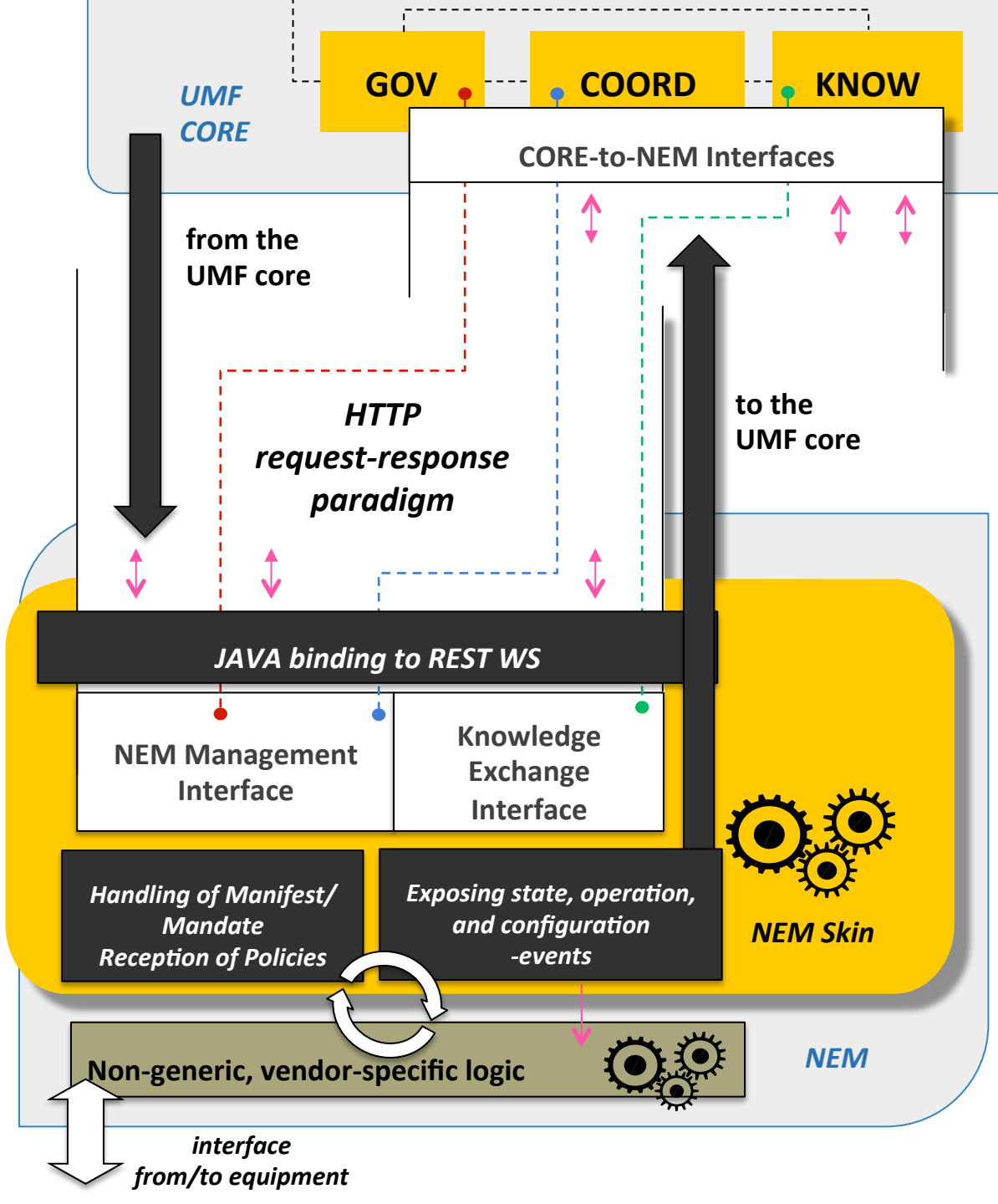
  for the NEM developer and the management from UMF CORE

- **Interface definitions**

  method/resource signatures

- **Mechanism to handle the RESTful interfaces**

- **Not restricted to REST or JAVA**



NEM SKIN Development

A JAVA package that implements:

- the RESTful interfaces to/from the UMF Core, as well as some of the most critical aspects of the NEM model regarding the manifest, the mandate, the actions, information and configuration options.
- the seamless driving of a NEM through it's lifecycle (but with capabilities to intercept it)
- additional abstraction over the HTTP details to provide with a REST-based RPC-invocation mechanism using regular JAVA code and compile-time binding
- seamless publish/subscribe-over-HTTP support using regular JAVA events

The final result is:

- **a single point of updating the UMF-related part of all NEMs**
- **UMF compliance for the NEM developer without having to be aware of any protocol-specific details**
- **an API that might potentially be used with a communication technology other than REST (future design choices might instruct so)**

UMF IN A NUTSHELL
UMF CORE FUNCTIONAL BLOCKS

Seamless deployment and trustworthy interworking of NEMs require:

- Tools for the operators to deploy, pilot, control and track progress of NEMs in a unified way
  - **GOVERNANCE functional block**
- Tools to identify/avoid conflicts and ensure stability and performance when several NEMs are concurrently working
  - **COORDINATION functional block**
- Tools to make NEMs find, formulate and share relevant information to enable or improve their operation
  - **KNOWLEDGE functional block**
- APIs to enable NEMs “plug and play” deployment, interoperability and monitoring/configuration
  - **NEM Skin**

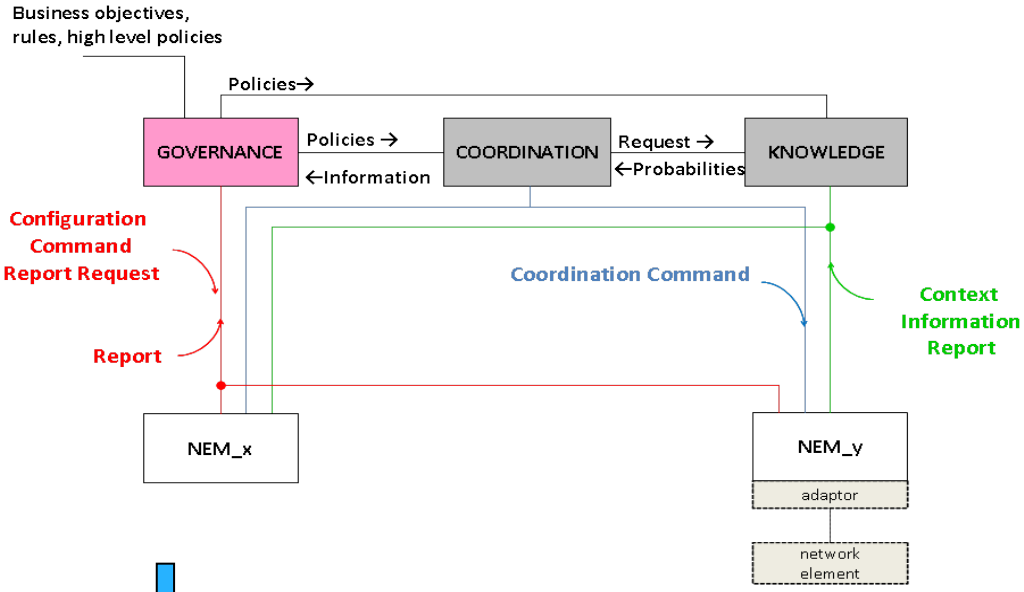
UMF IN A NUTSHELL
UMF CORE FUNCTIONAL BLOCKS

Responsible for:

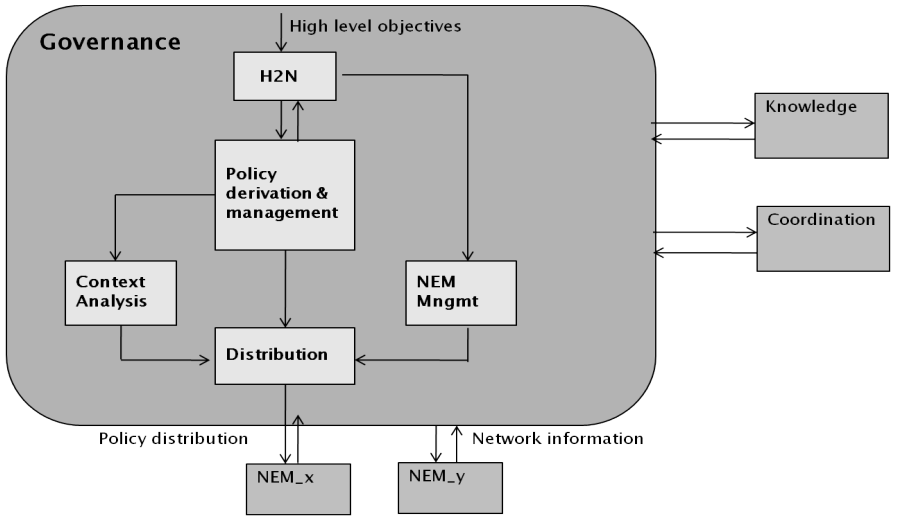
- The interaction between human operator and its network→ express business goals report on critical states of self-managed operations/ devices
- Driving NEMs' behavior→ policy-based framework for translating business-level, service specific goals/requests into low level, policies and configuration commands

GOVERNANCE ←→ NEM:

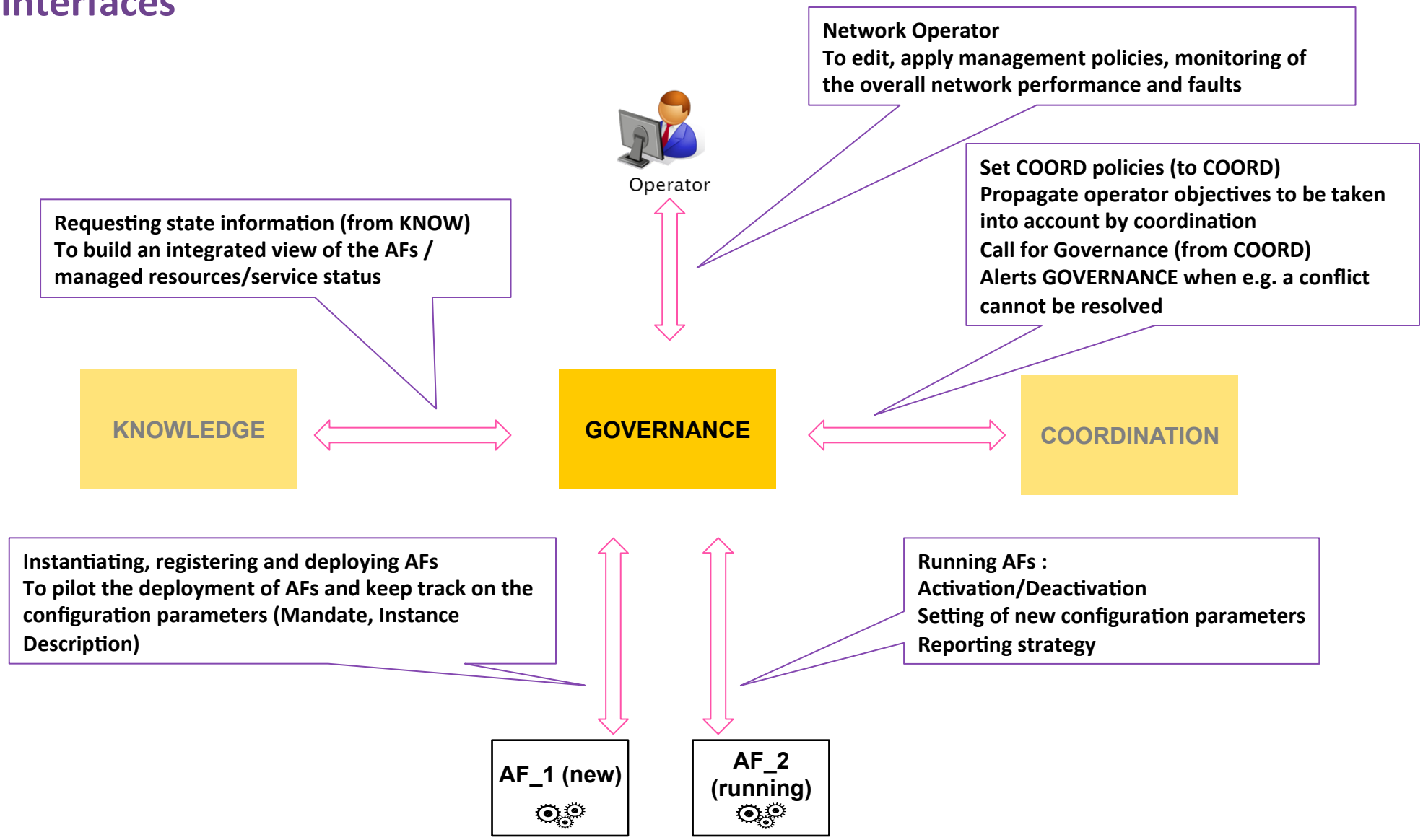
- Commands to set NEM's status/mode (e.g. active, idle, stopped) and configure its operational parameters.
- Report on the NEM's operational conditions and configuration characteristics (e,g. performance indicators, capabilities/ behaviour, interaction with other NEMs).



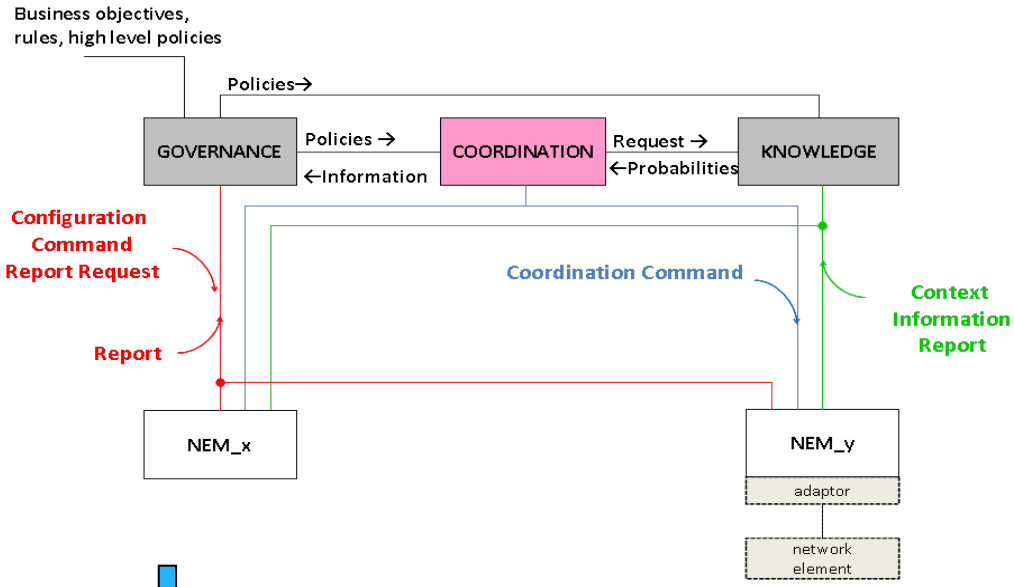
Functional decomposition



UMF / GOVERNANCE Interfaces



UMF IN A NUTSHELL
UMF CORE FUNCTIONAL BLOCKS



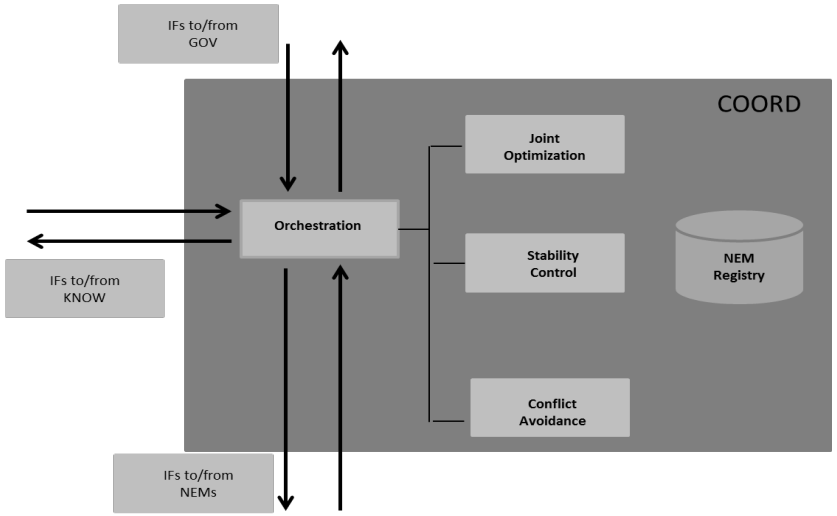
Responsible for:

- Ensuring the proper sequence in triggering of NEMs and the conditions under which they will be invoked taking into account:
  - ✓ Operator and service requirements,
  - ✓ Needs for Conflict avoidance, joint optimization and stability control.

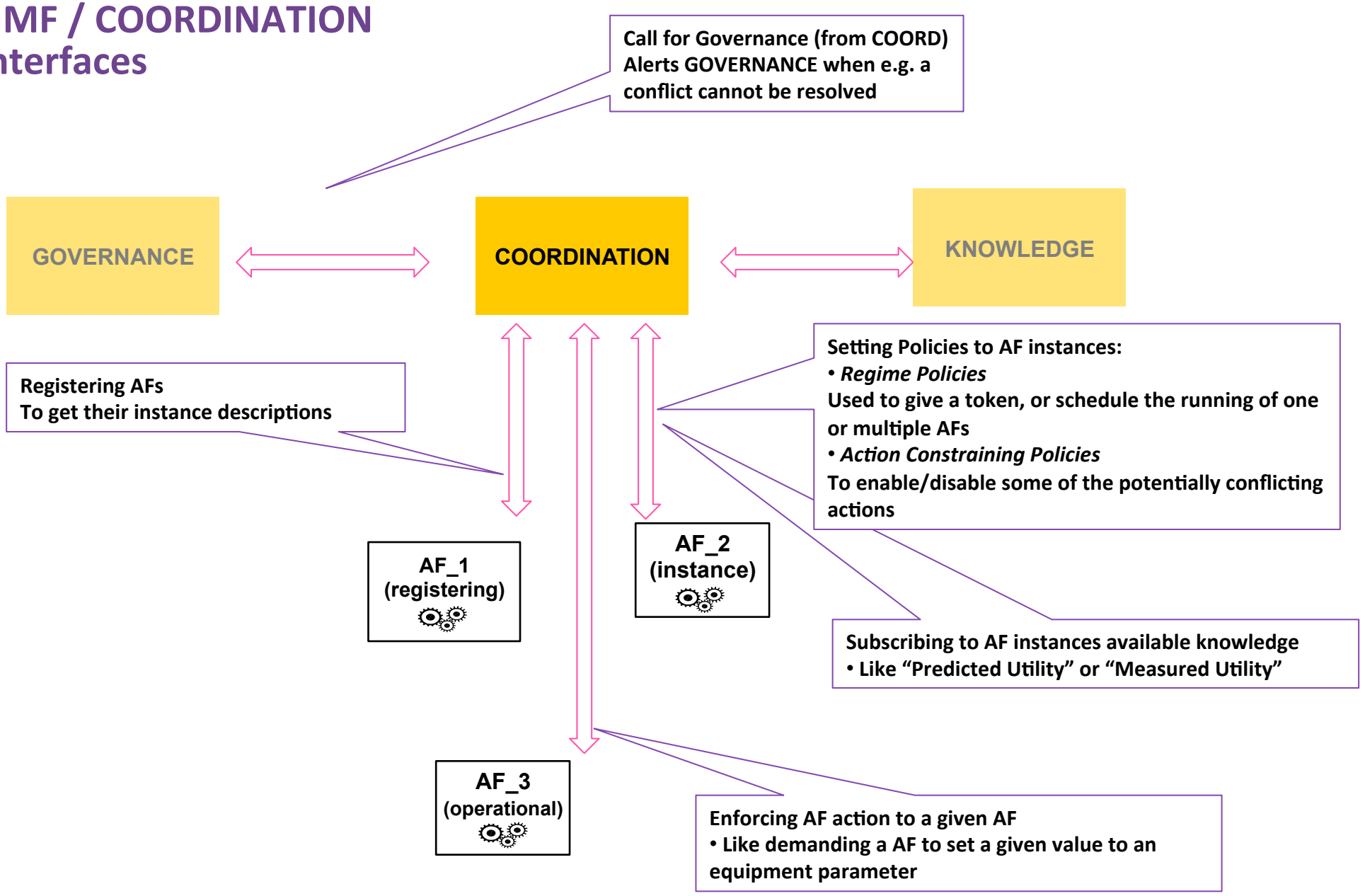
COORDINATION ←→ NEM:

- Commands to drive coordination including: tokens, timing, constraints, status (active/ idle), etc.
- Information on the NEMs operation including: parameters, metrics, scope, utility functions, etc.

Functional decomposition



UMF / COORDINATION Interfaces



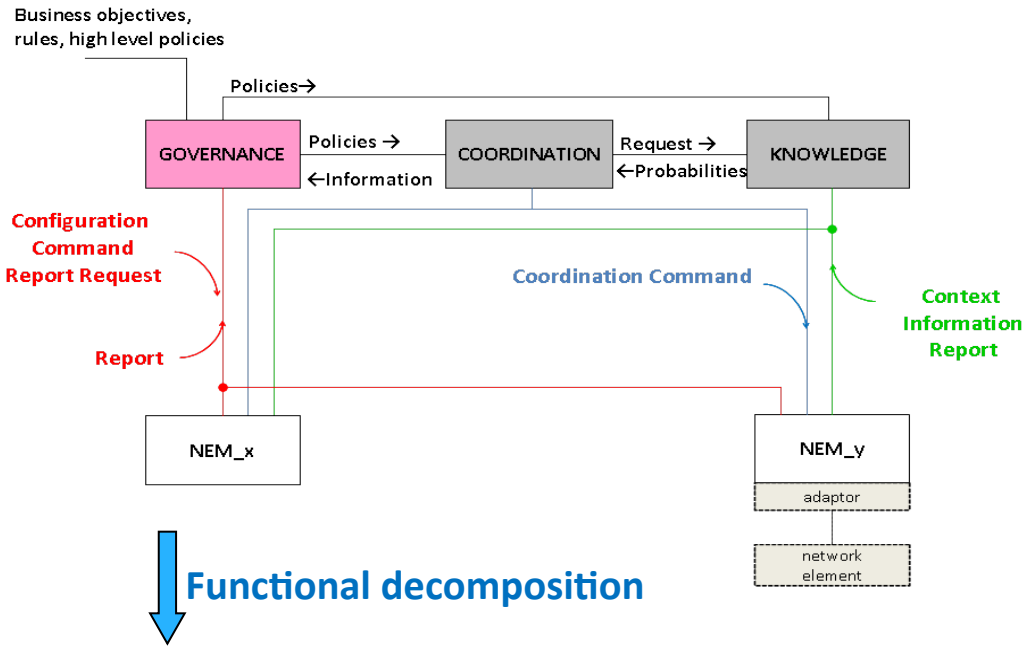
UMF IN A NUTSHELL
UMF CORE FUNCTIONAL BLOCKS

Responsible for:

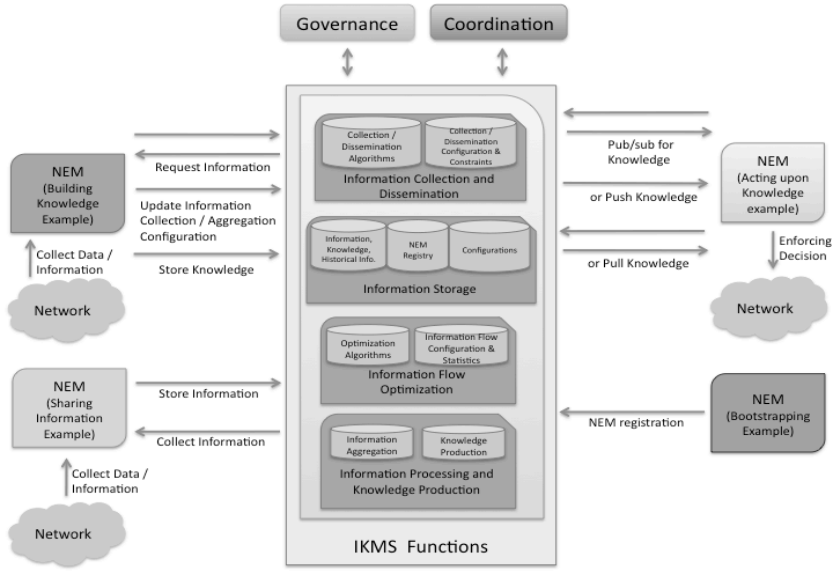
- Providing the suitable probabilistic models methods and mechanisms for derivation and exchange of Knowledge, based on :
  - ✓ Context and configuration information from NEMs,
  - ✓ Policies from Governance,
  - ✓ Information on NEM interactions from coordination

KNOWLEDGE ←→ NEM:

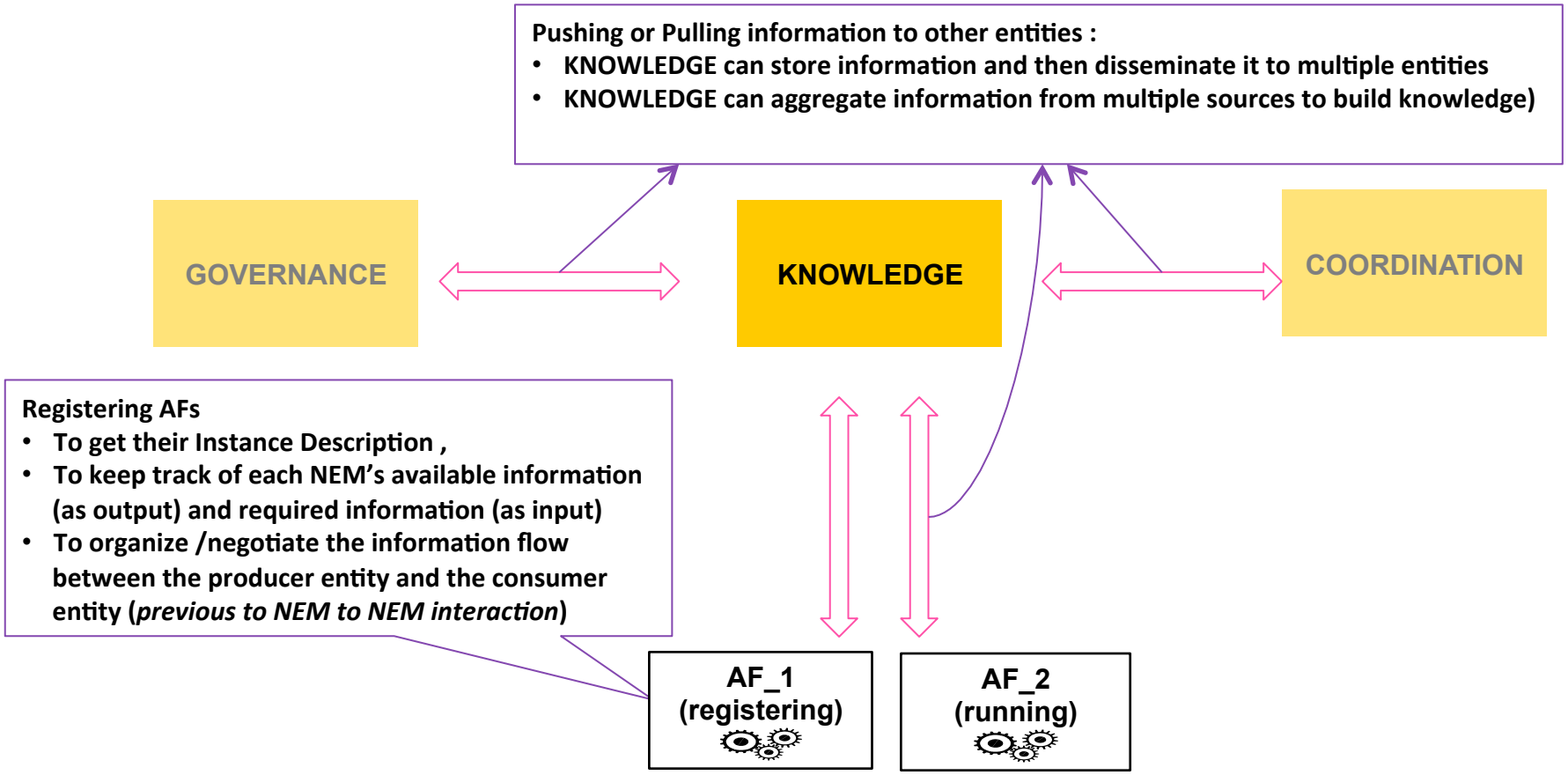
- Commands to retrieve, share, derive and manage knowledge including: publish, subscribe, push, pull, request, store, notify … messages.
- Registration of NEMs.



Functional decomposition



UMF / KNOWLEDGE
Interfaces



SUMMARY

A unified framework to deploy and control self-managing functions

- **Specifications of the UMF core utility functions**
- **Specifications for interoperable and versatile autonomic functions**
- **UMF and NEM APIs (skin) and workflows/sequence charts**
- **Publicly available specifications, developer guidelines**
- **Implemented, tested, modular and re-usable components**
  - NEM skin, RESTful APIs
  - **Available as open (multi-OS) platform for IRTF/IETF and +**
    - Website under construction (should be up and running by end of the month)
    - Building a community, experimenting further with IETF protocols
    - Several NEMs (~10), use cases and data plane technologies available for demo/test
    - Making proof of feasibility, gaining knowledge…

THOUGHTS for the discussion

- **Agree and define design principles and properties of an autonomic network**
- **Identify cross-domain use cases highlighting limitations of current protocols/ practices**

  → how to correlate measurements, to check policy consistency…

  → linking network and service “layers” (chaining aspects)
- **How to make operations scale?**

  → Automatic, adaptive and aware
- **Document guidelines/recommendations to design/enhance IETF protocols with autonomic networking principles**

  → to improve Internet manageability and performance

QUESTIONS & ANSWERS

WWW.UNIVERSELF-PROJECT.EU