ASOV- Days – January 2014

Workflows for astronomy using PDL

Carlo Maria Zwölf,   Julian Garrido.

Laboratoire d'Etude du Rayonnement
et de la Matière en Astrophysique

News from Workflow activity

- April 2013: publication of an IVOA note on scientific Workflow (A. Schaaff, J.E. Ruiz et al)

- May 2013: interop@heidelberg → Workflow meeting as a branch of GWS working group
  - WF4ever, ER-flow and PDL community assembled

- November 2013: Workflow working group meeting
  - Details on http://www.france-ov.org/twiki/bin/view/GROUPEStravail/WorkflowReunion9
  - Shaped as a PDL tutorial (for using from scratch the framework implemented)

- 21st of January 2014 : PDL ended its second IVOA community RFC period
  - http://wiki.ivoa.net/twiki/bin/view/IVOA/PDL1RFC/
  - Immediately started TGC review

PDL: a quick overview

- Parameter Description Language (PDL) is intended to be a lingua franca of parameters:
  - Describes params in a sufficient detail to allow workflow tools to check if parameters can be "piped" between services
    - Physical Properties (Nature, Meaning, unit, precision,...)
    - Computing (Numerical Type, UCD, SKOS concept)
  - Also has capabilities do describe constraints on parameters
    - Physical constraints
    - Arbitrary (including mathematical) constraints
- **Not** a description of parameters "values" (cf. UWS).
- PDL is an overlay completely independent from the technology used by the described services.

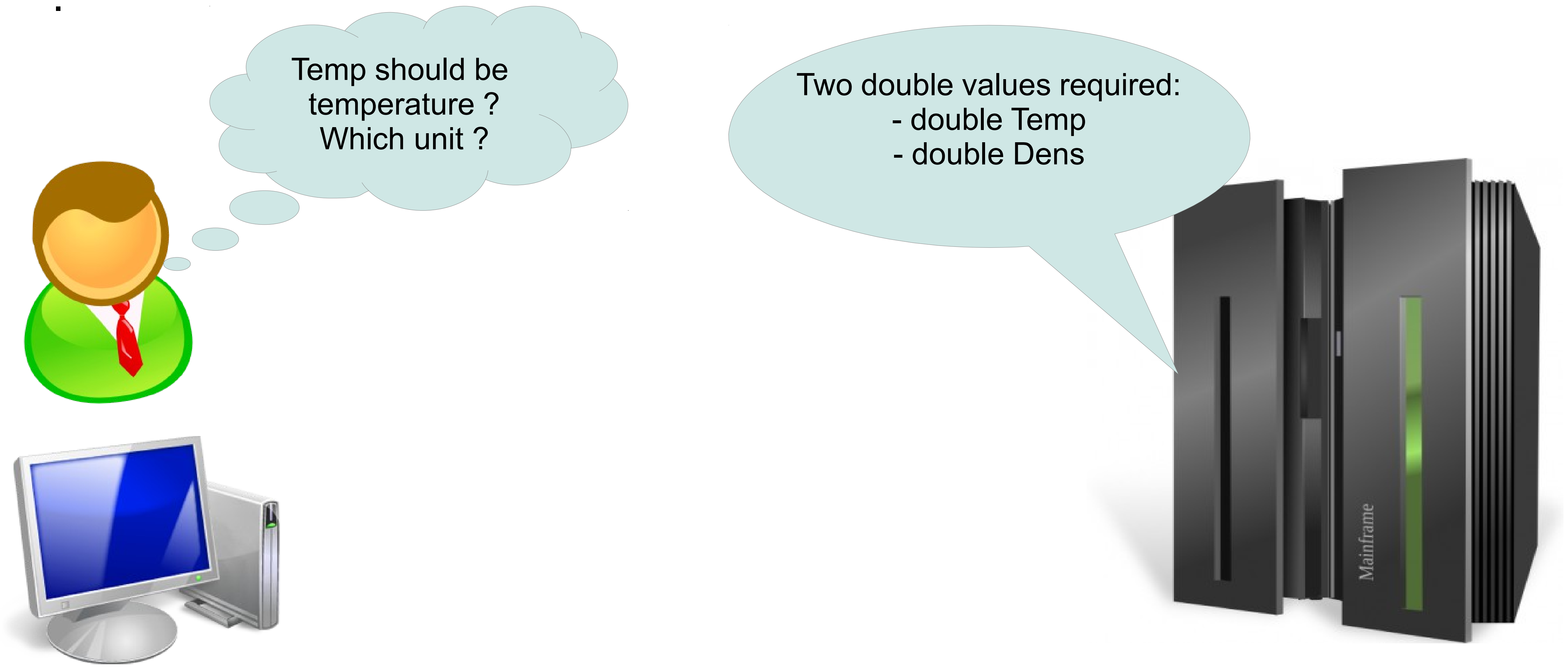
PDL: A QUICK OVERVIEW

- Parameter Description Language (PDL) is intended to be a lingua franca of parameters:
  - Describes params in a sufficient detail to allow workflow tools to check if parameters can be "piped" between services
    - Physical Properties (Nature, Meaning, unit, precision,...)
    - Computing (Numerical Type, UCD, SKOS concept)
  - Also has capabilities do describe constraints on parameters
    - Physical constraints
    - Arbitrary (including mathematical) constraints
- **Not** a description of parameters "values" (cf. UWS).
- PDL is an overlay completely independent from the technology used by the described services.

PDL implementations based on				
Generic software components can be 'configured' by a PDL description for creating quickly fully interoperable new services				
Server exposing services as web services	User Interface (for interaction with PDL services)	Auto Generation of checking algorithms from description	Workflow plugin (for WF interaction with PDL services)	A priori computation of interoperability graphs

PDL Principles

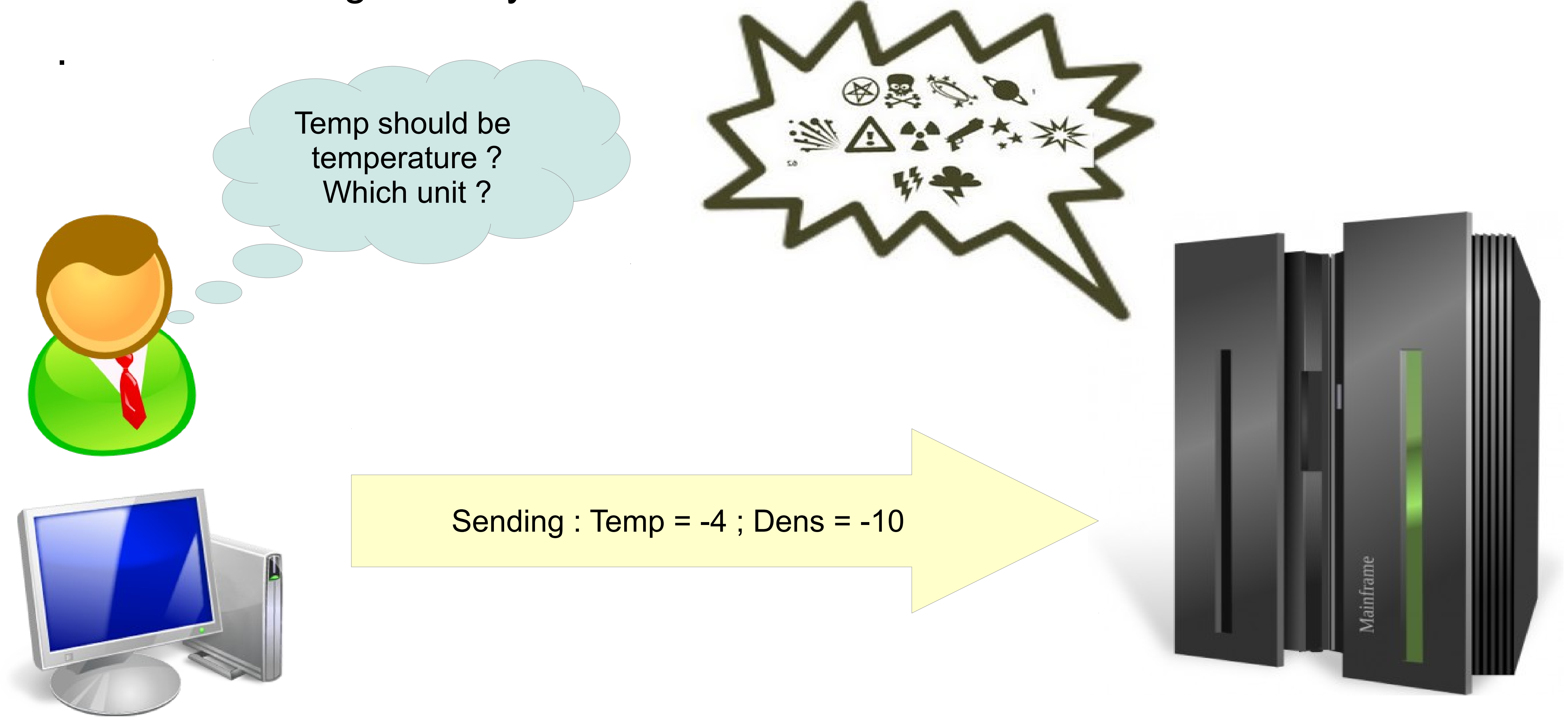
- The language is based on a *Data Model;*
- Each object of the DM corresponds to a syntactic element:
  - Sentences are made by building object-structures;
  - Each sentence is interpreted by a computer by parsing the sentence-related object-structure;
  - With no loss of generality → the DM is fixed into an XML schema.


Temp should be temperature ? Which unit ?
Two double values required: - double Temp - double Dens
Mainframe

PDL Principles

- The language is based on a *Data Model;*
- Each object of the DM corresponds to a syntactic element:
  - Sentences are made by building object-structures;
  - Each sentence is interpreted by a computer by parsing the sentence-related object-structure;
  - With no loss of generality → the DM is fixed into an XML schema.

.


Temp should be temperature ? Which unit ?
Sending : Temp = -4 ; Dens = -10
Mainframe

PDL Principles

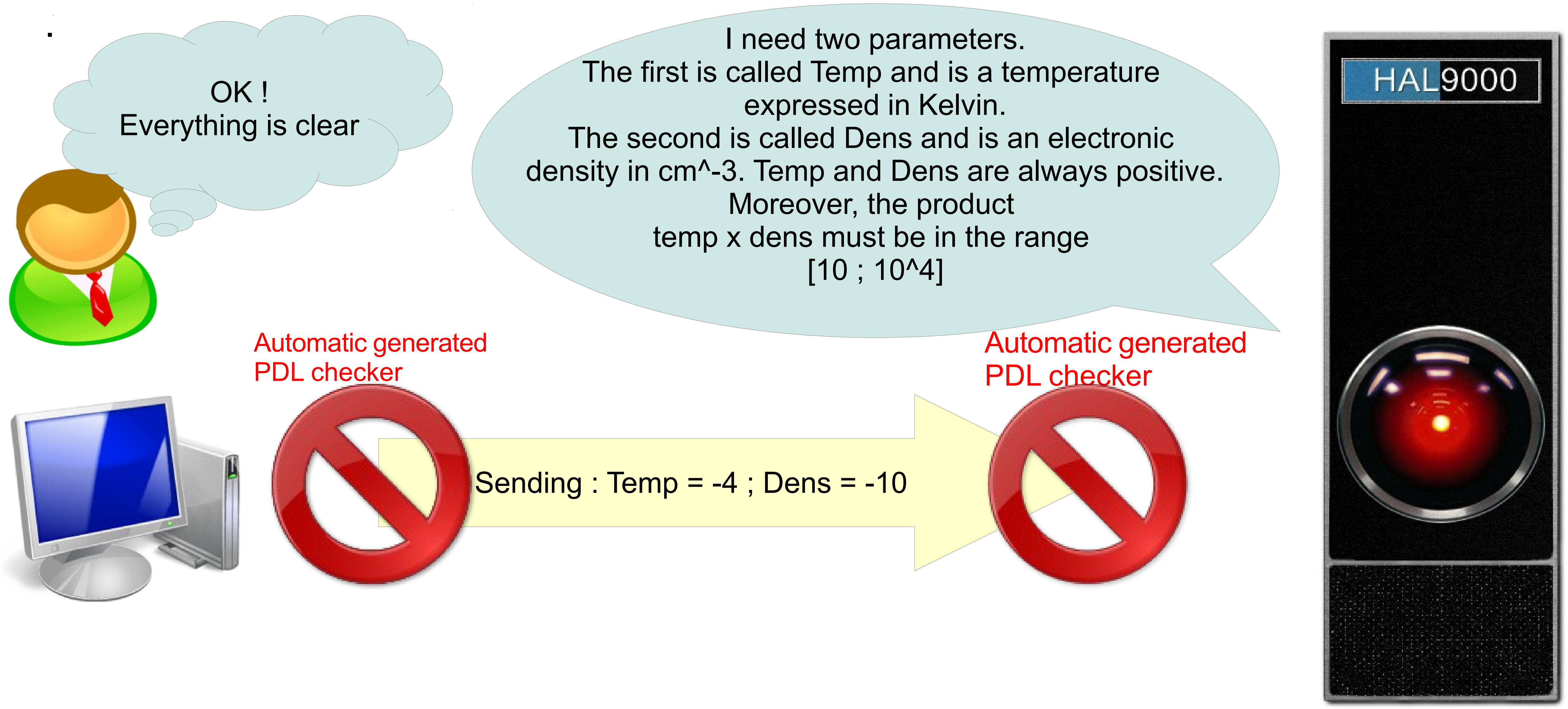
- The language is based on a *Data Model;*
- Each object of the DM corresponds to a syntactic element:
  - Sentences are made by building object-structures;
  - Each sentence is interpreted by a computer by parsing the sentence-related object-structure;
  - With no loss of generality → the DM is fixed into an XML schema.

.

I need two parameters.
The first is called Temp and is a temperature expressed in Kelvin.
The second is called Dens and is an electronic density in cm^-3. Temp and Dens are always positive.
Moreover, the product temp x dens must be in the range [10 ; 10^4]

PDL Principles

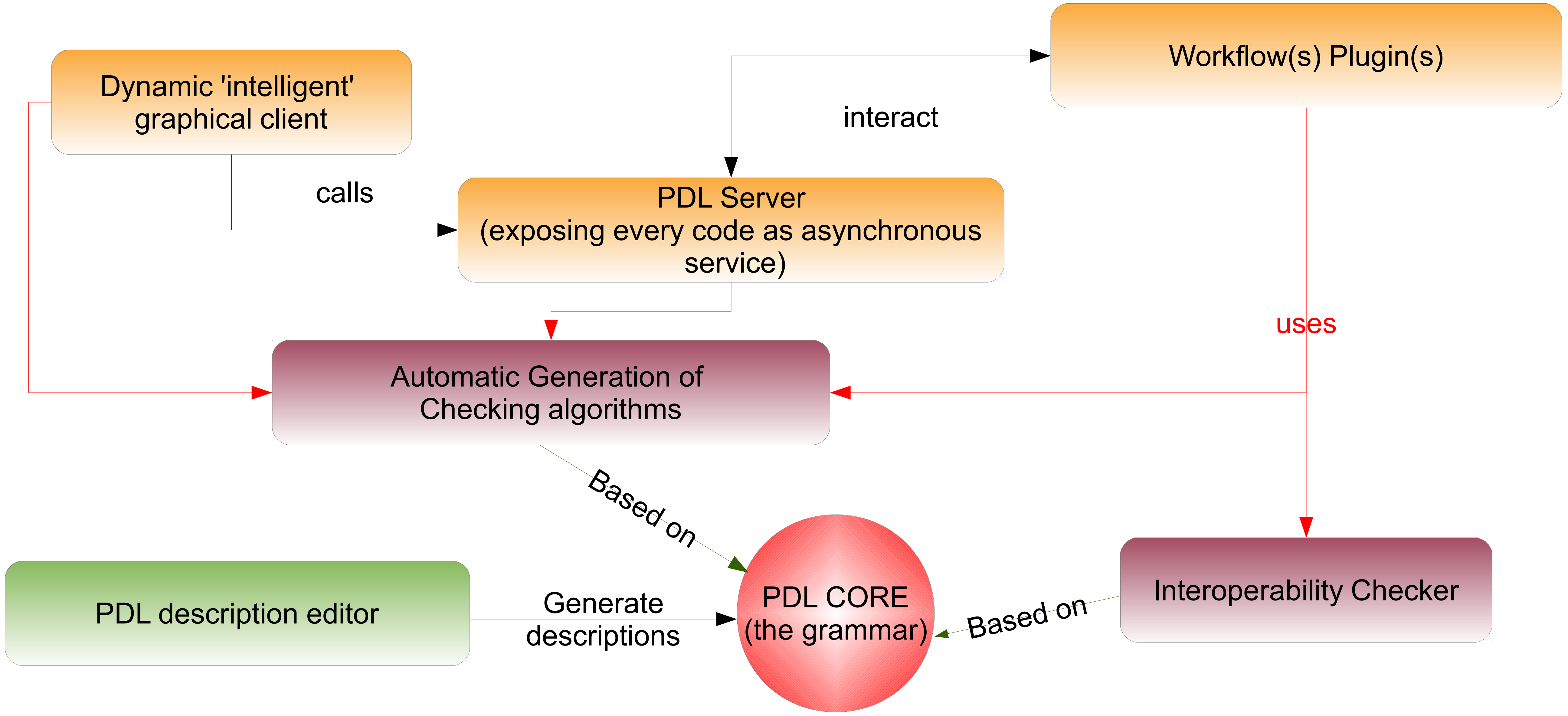
- The language is based on a *Data Model;*
- Each object of the DM corresponds to a syntactic element:
  - Sentences are made by building object-structures;
  - Each sentence is interpreted by a computer by parsing the sentence-related object-structure;
  - With no loss of generality → the DM is fixed into an XML schema.


OK ! Everything is clear
I need two parameters. The first is called Temp and is a temperature expressed in Kelvin. The second is called Dens and is an electronic density in cm^-3. Temp and Dens are always positive. Moreover, the product temp x dens must be in the range [10 ; 10^4]
HAL9000
Automatic generated PDL checker
Sending : Temp = -4 ; Dens = -10
Automatic generated PDL checker

Software components based on PDL (PDL Framework)

Since parameters and constraints are finely described with fine grained granularity:

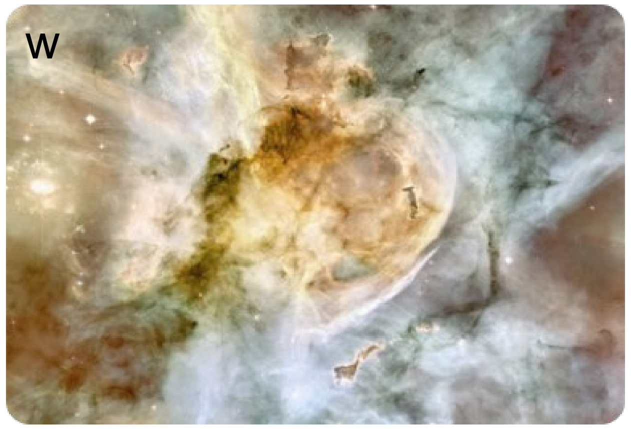
- Generic software elements could be automatically “configured” by a specific PDL description instance:
  - Services containers
  - Graphical User Interfaces
  - Workflow Plugins
- Checking algorithms and interoperability checker between service are automatically generated from descriptions

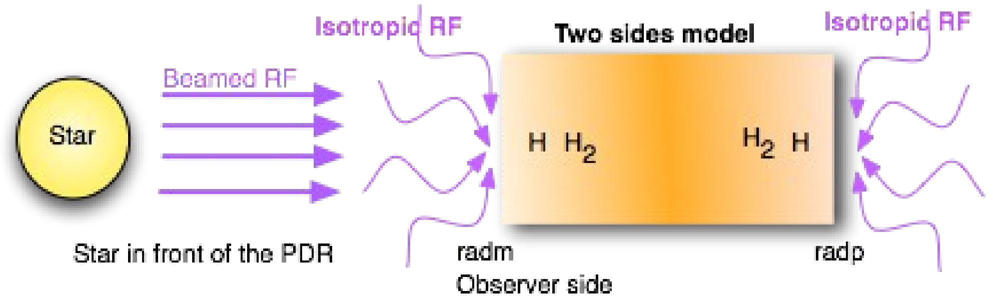


Why does PDL improve Interoperability ?

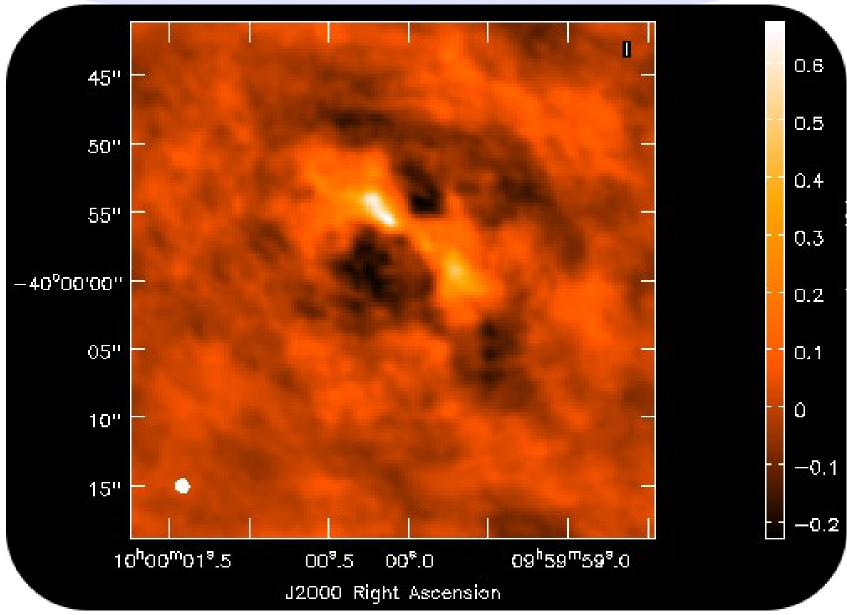
- PDL allows horizontal (i.e. between heterogeneous DM and/or Standards) interoperability.

Code PDR

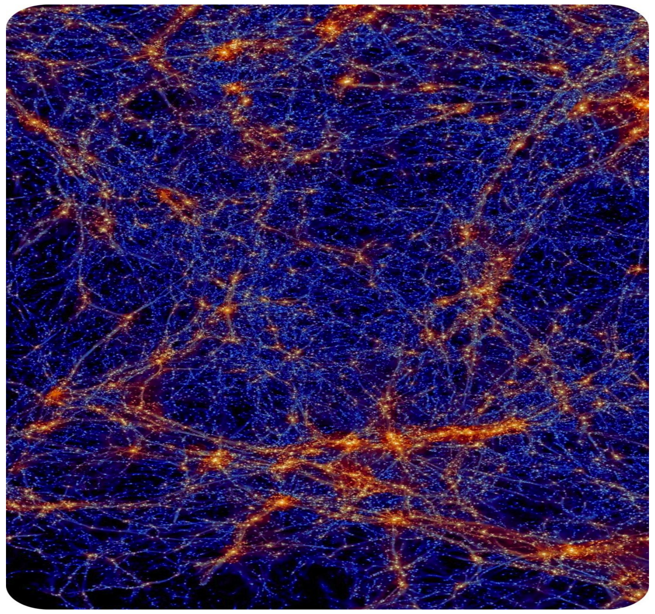




Simulateur de Telescope (Alma)

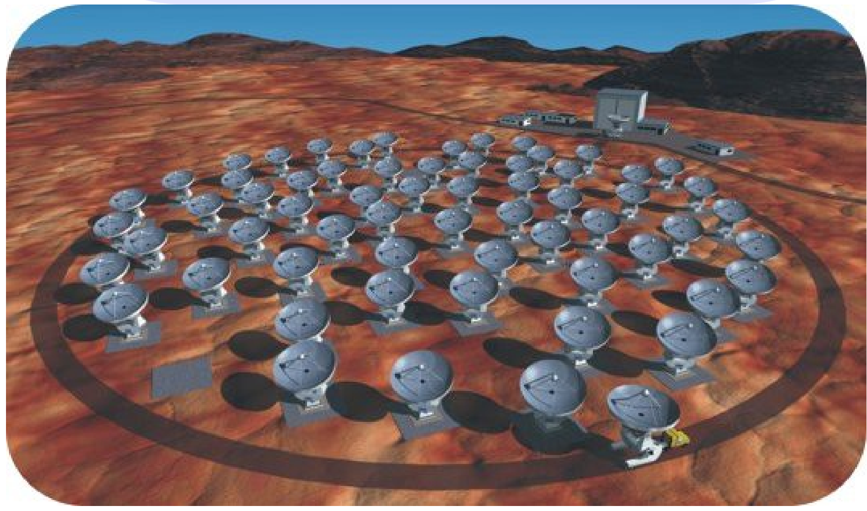


Simulations Ramses



Code Dustem

Donnée d'observations



Why does PDL improve Interoperability ?

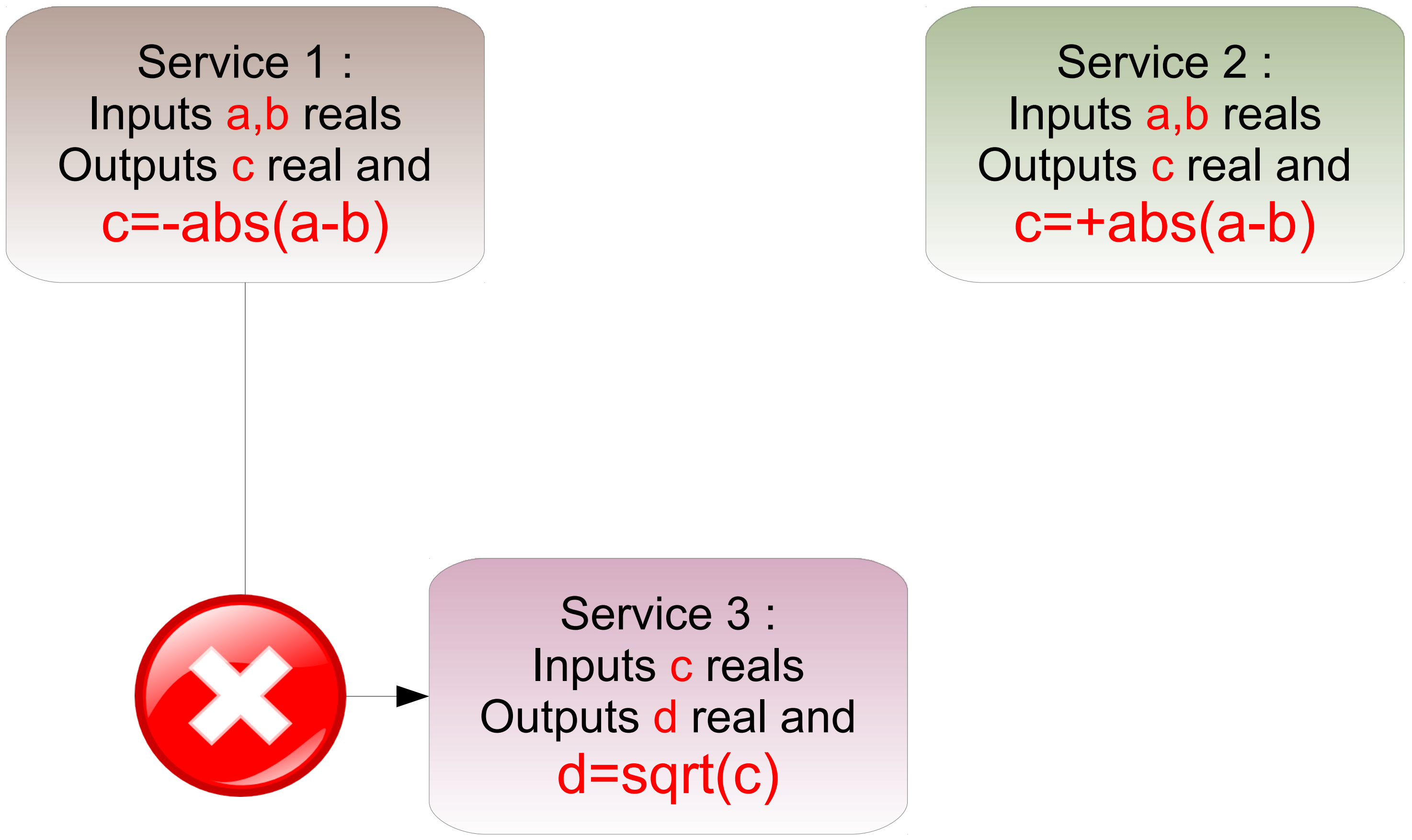
Service 1 :
Inputs a,b reals
Outputs c real and
c=-abs(a-b)

Service 2 :
Inputs a,b reals
Outputs c real and
c=+abs(a-b)

Service 3 :
Inputs c reals
Outputs d real and
d=sqrt(c)

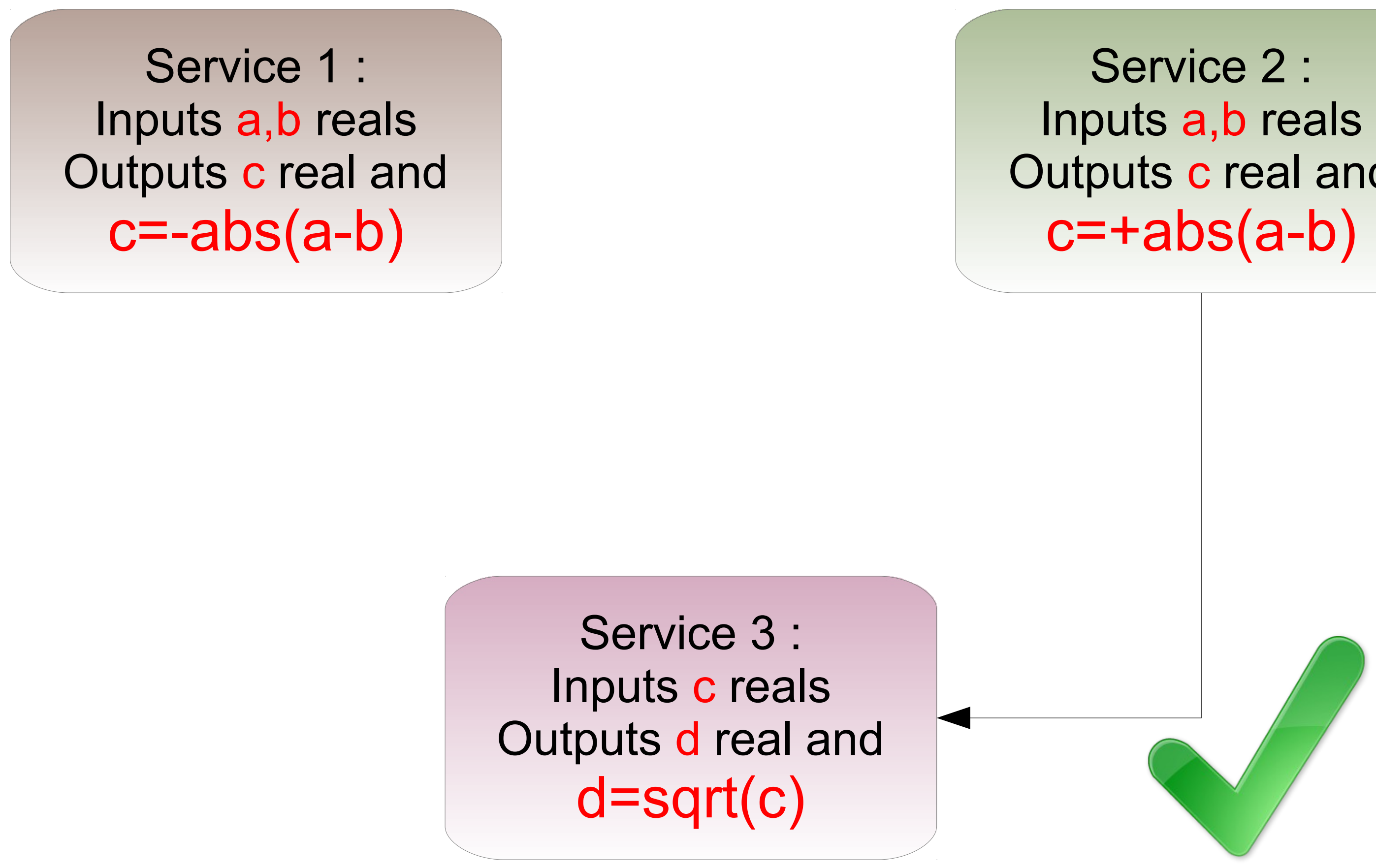
Eventual problems are detected only during the workflow execution

Why does PDL improve Interoperability ?



Eventual problems are detected only during the workflow execution

Why does PDL improve Interoperability ?



Eventual problems are detected only during the workflow execution

Why does PDL improve Interoperability ?

Adding a description layer

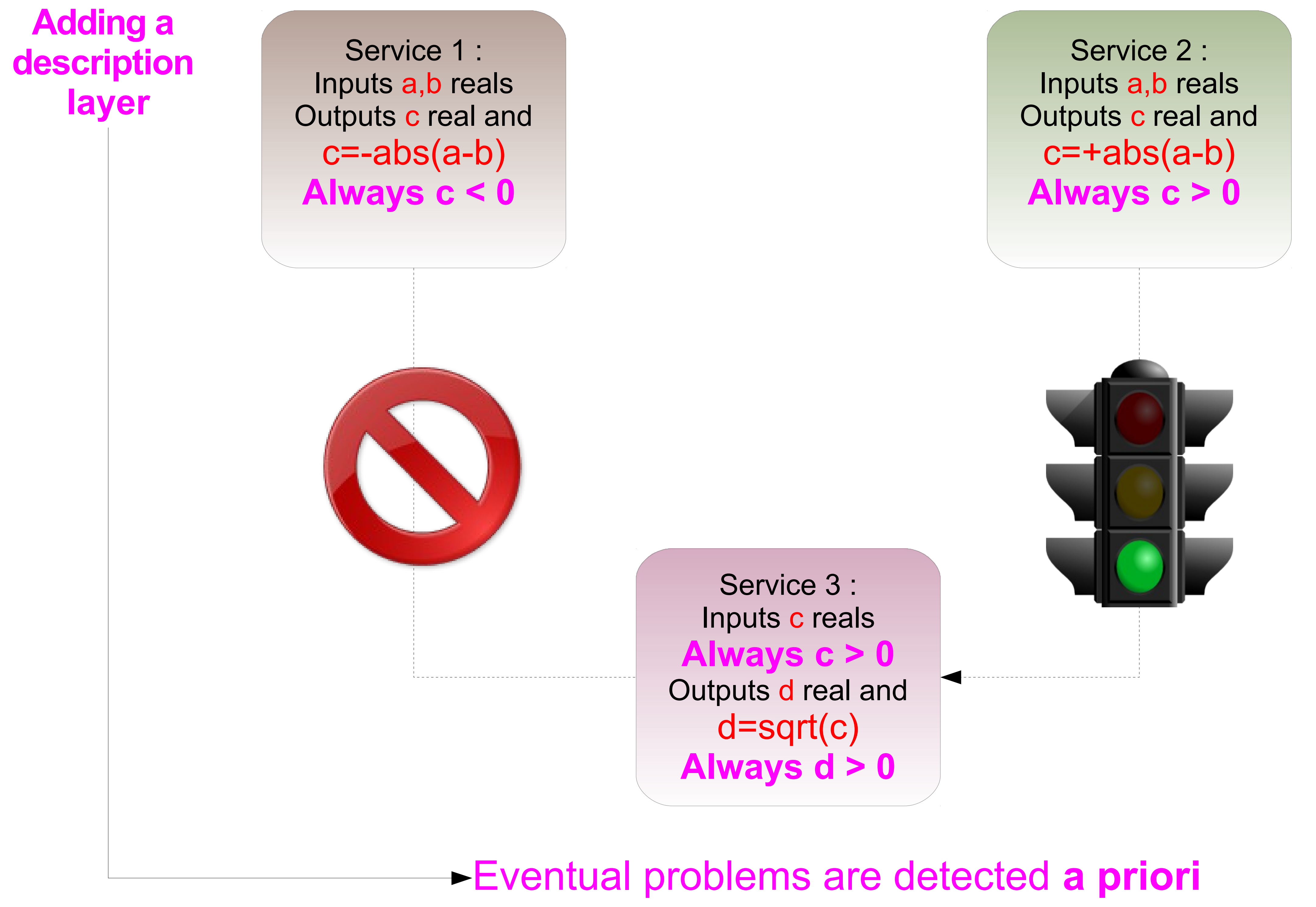
Service 1 :
Inputs a,b reals
Outputs c real and
c=-abs(a-b)
Always c < 0

Service 2 :
Inputs a,b reals
Outputs c real and
c=+abs(a-b)
Always c > 0

Service 3 :
Inputs c reals
Always c > 0
Outputs d real and
d=sqrt(c)
Always d > 0

►Eventual problems are detected **a priori**

Why does PDL improve Interoperability ?



Adding a description layer

Service 1 :
Inputs a,b reals
Outputs c real and
c=-abs(a-b)
Always c < 0

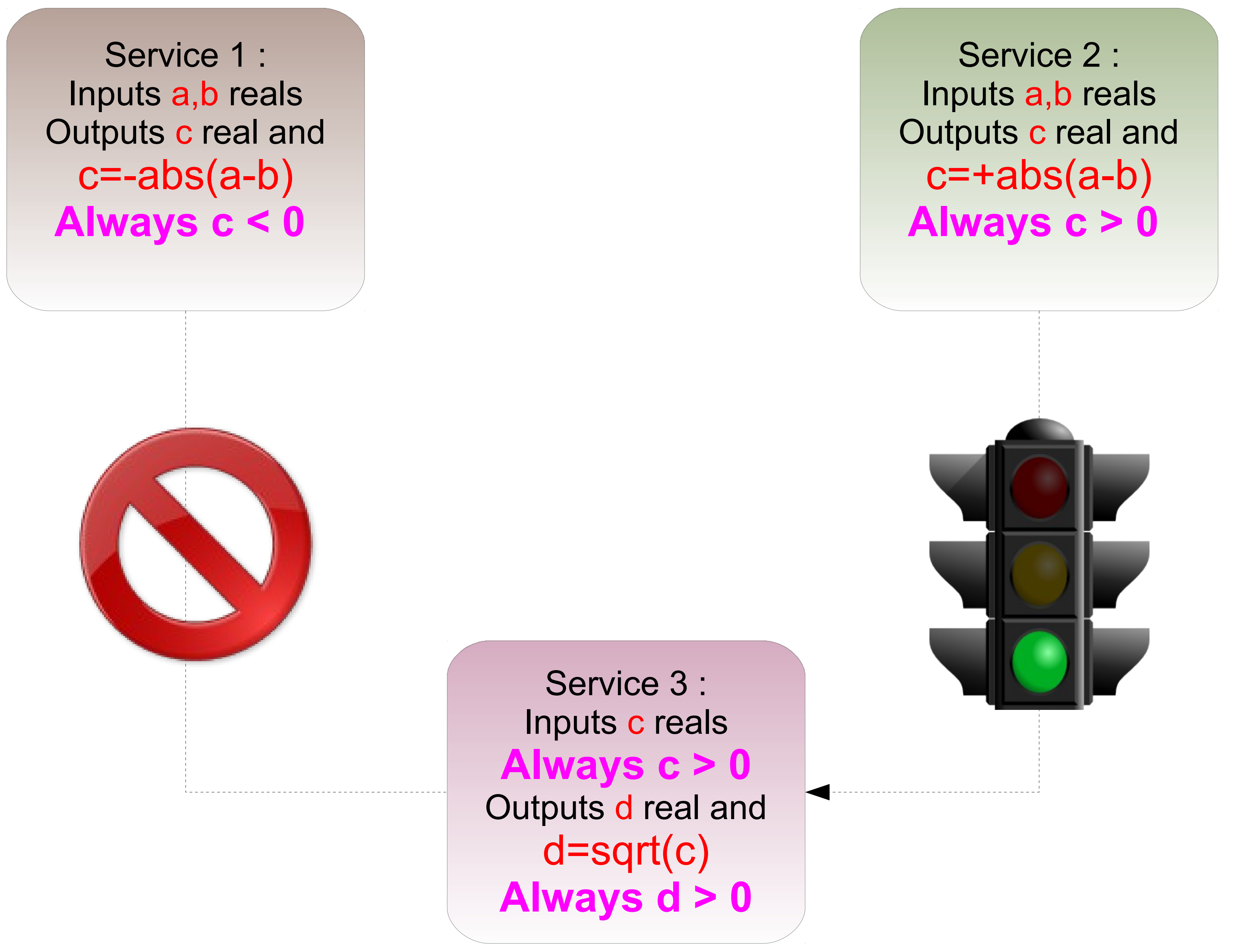
Service 2 :
Inputs a,b reals
Outputs c real and
c=+abs(a-b)
Always c > 0

Service 3 :
Inputs c reals
Always c > 0
Outputs d real and
d=sqrt(c)
Always d > 0

Eventual problems are detected **a priori**

WHY DOES PDL IMPROVE INTEROPERABILITY ?

Adding a description layer



Service 1 :
Inputs a,b reals
Outputs c real and
c=-abs(a-b)
Always c < 0

Service 2 :
Inputs a,b reals
Outputs c real and
c=+abs(a-b)
Always c > 0

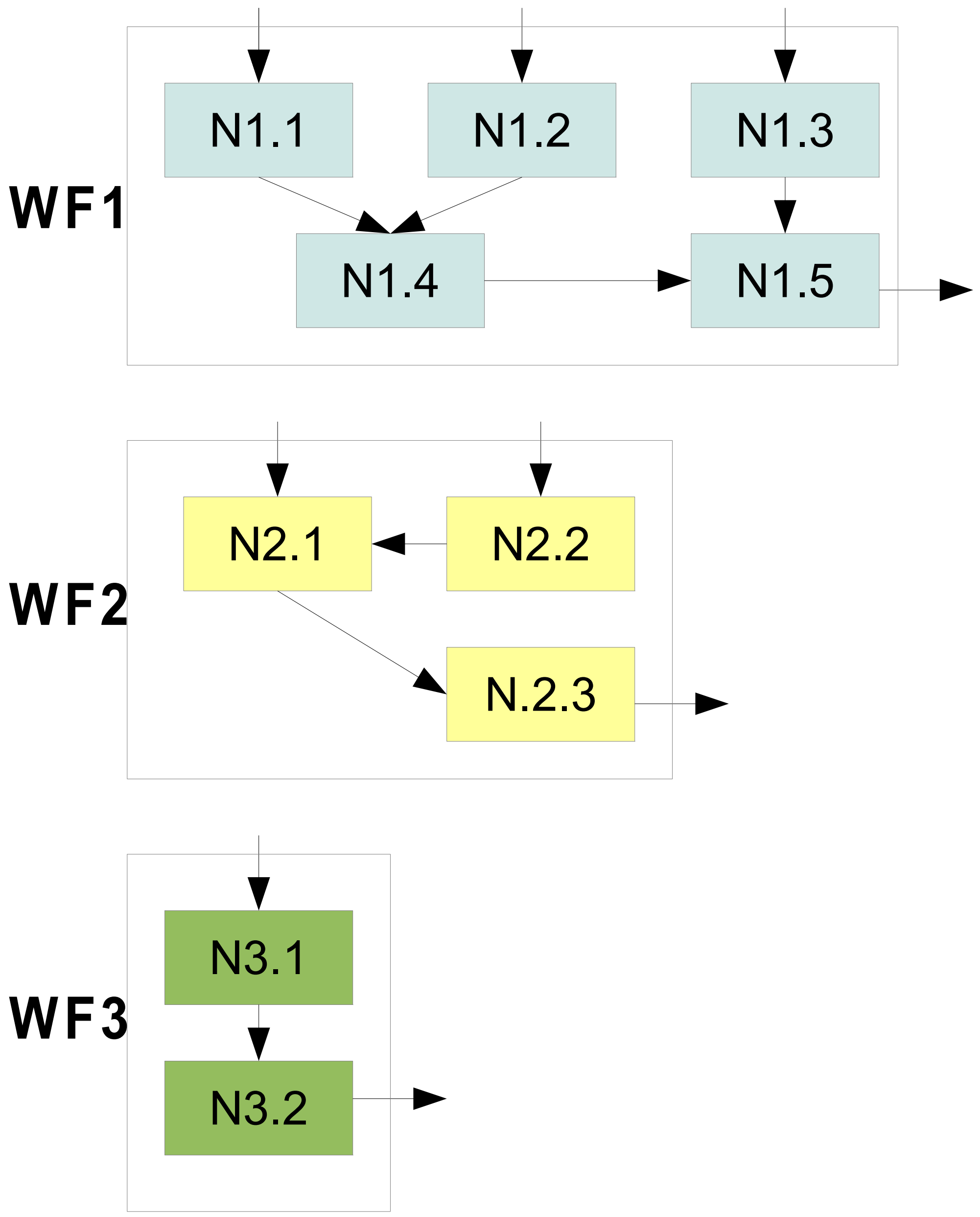
Service 3 :
Inputs c reals
Always c > 0
Outputs d real and
d=sqrt(c)
Always d > 0

►Eventual problems are detected **a priori**

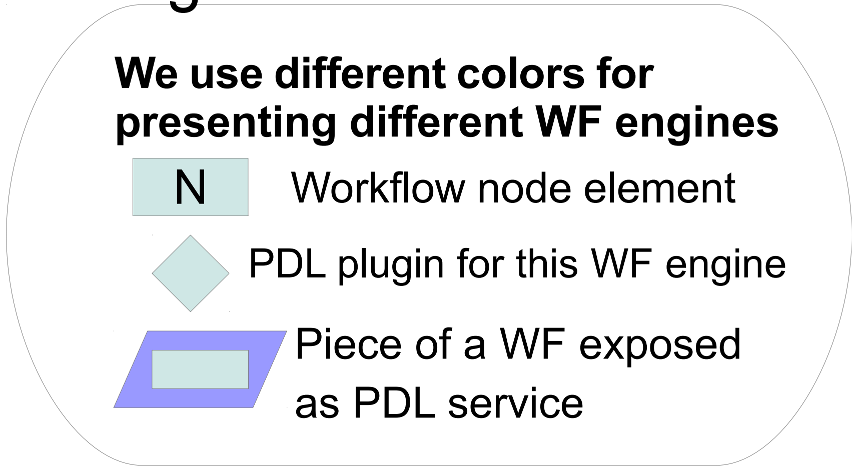
→ PDL generalize these concepts

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:



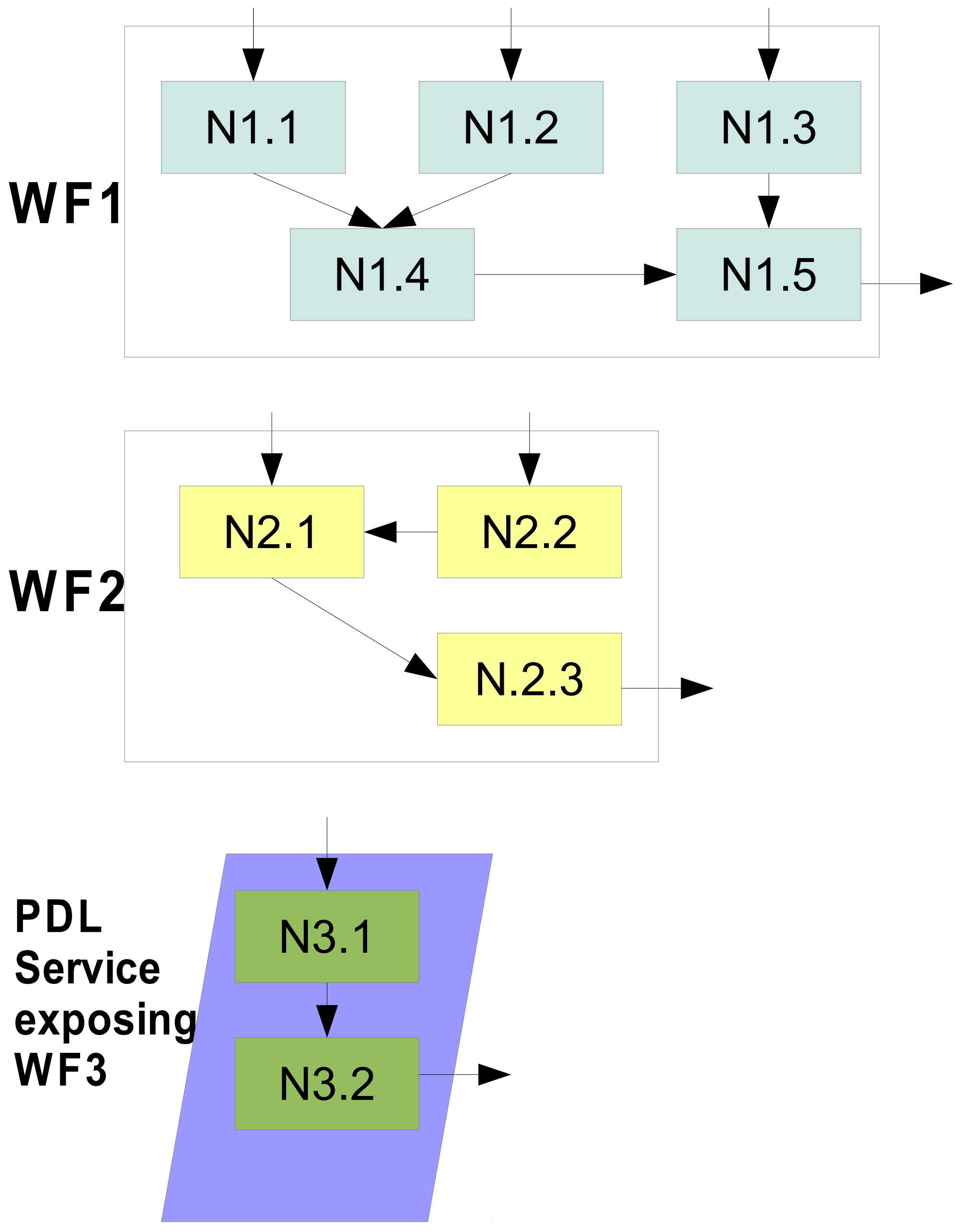
legend

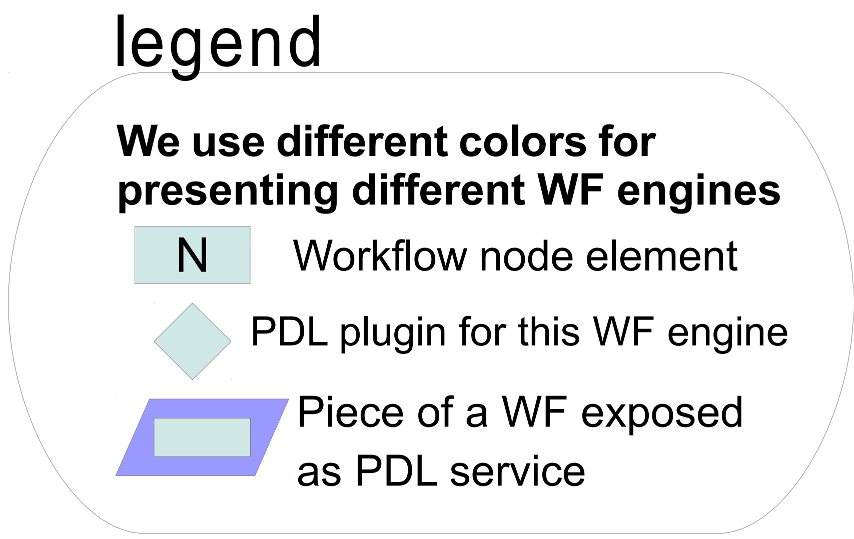

We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:

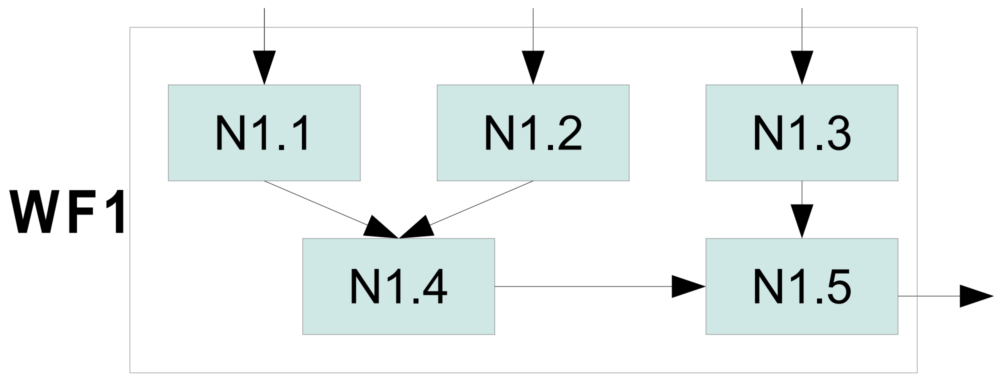


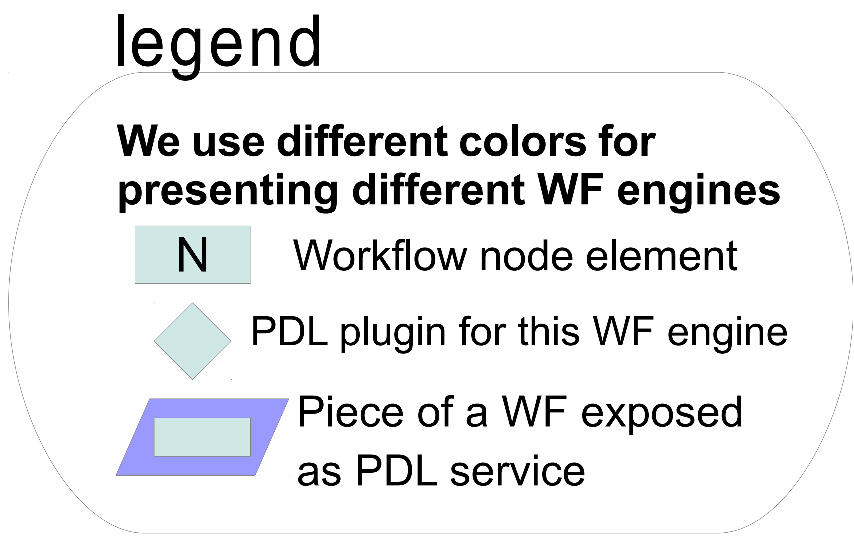

legend
We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

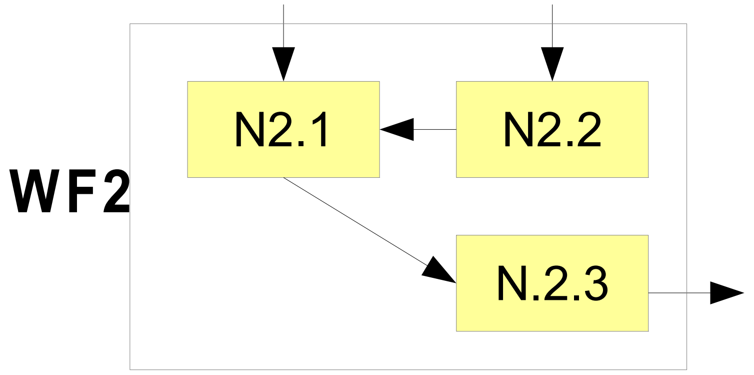
- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**

PDL workflows with heterogeneous workflows engines

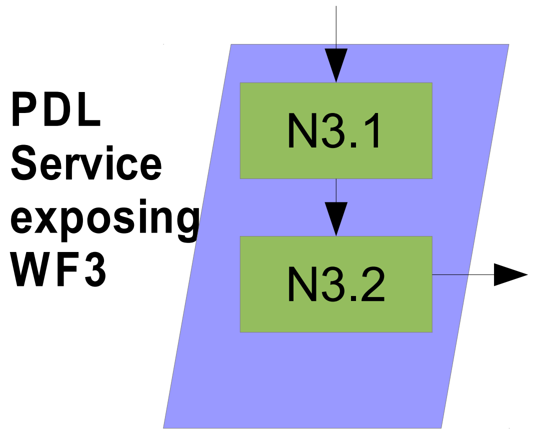
- PDL allow easy cross communication for workflows using different engines:




legend
We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service



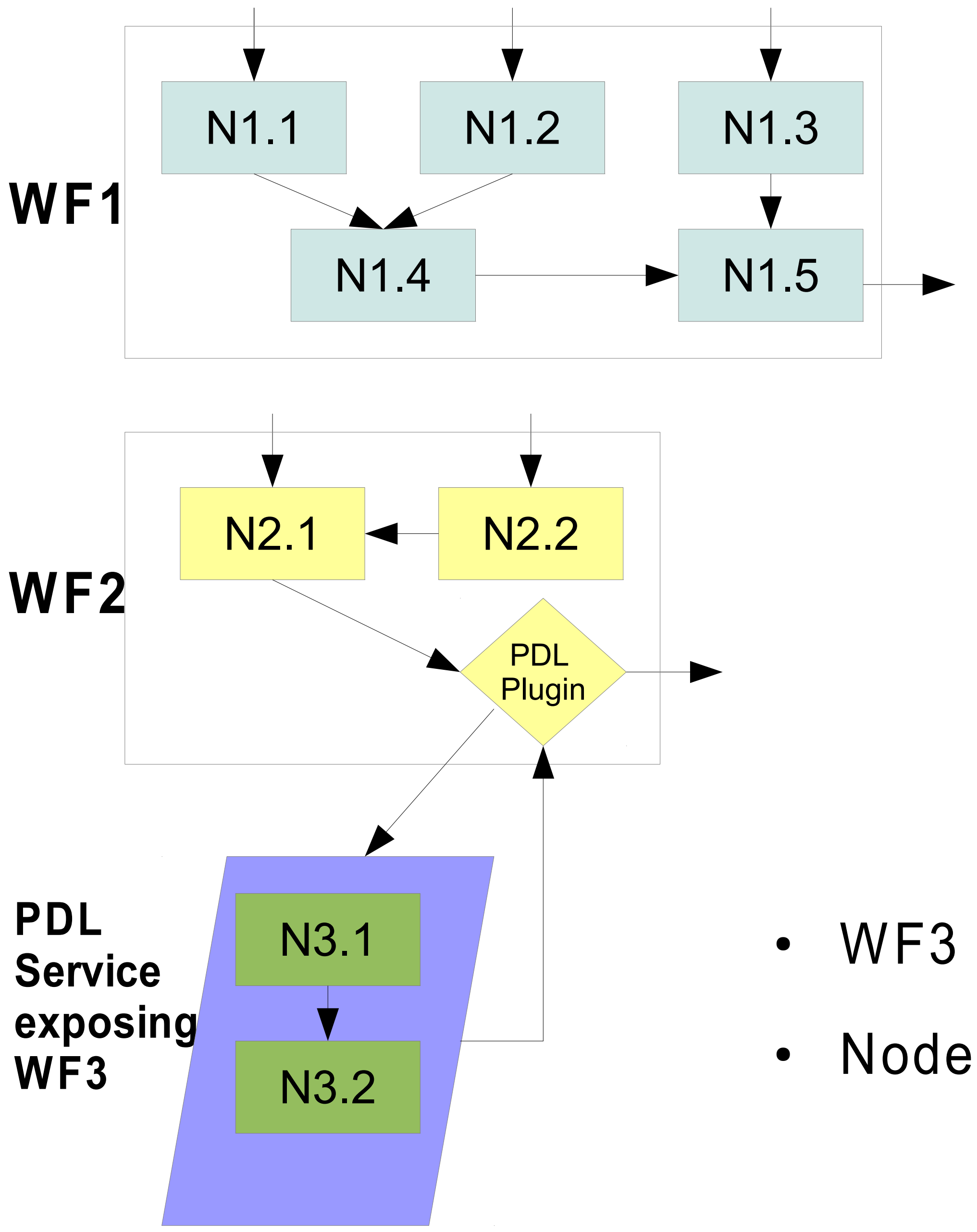
- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**



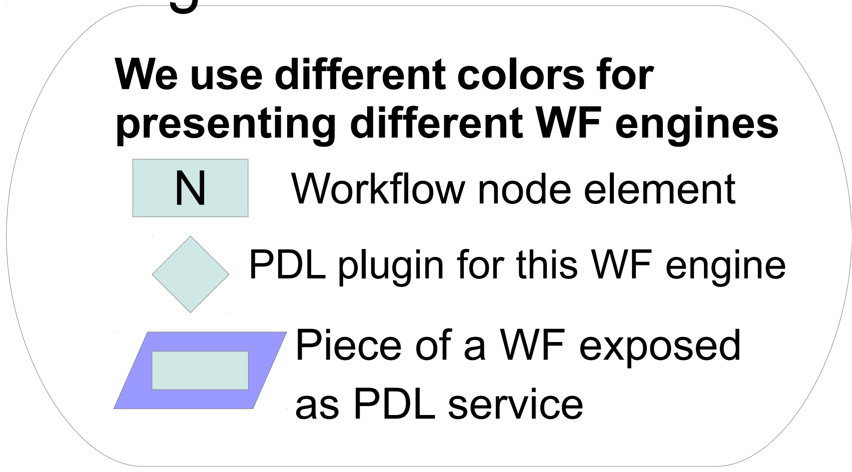
- WF3 is exposed as a PDL Service

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:



legend



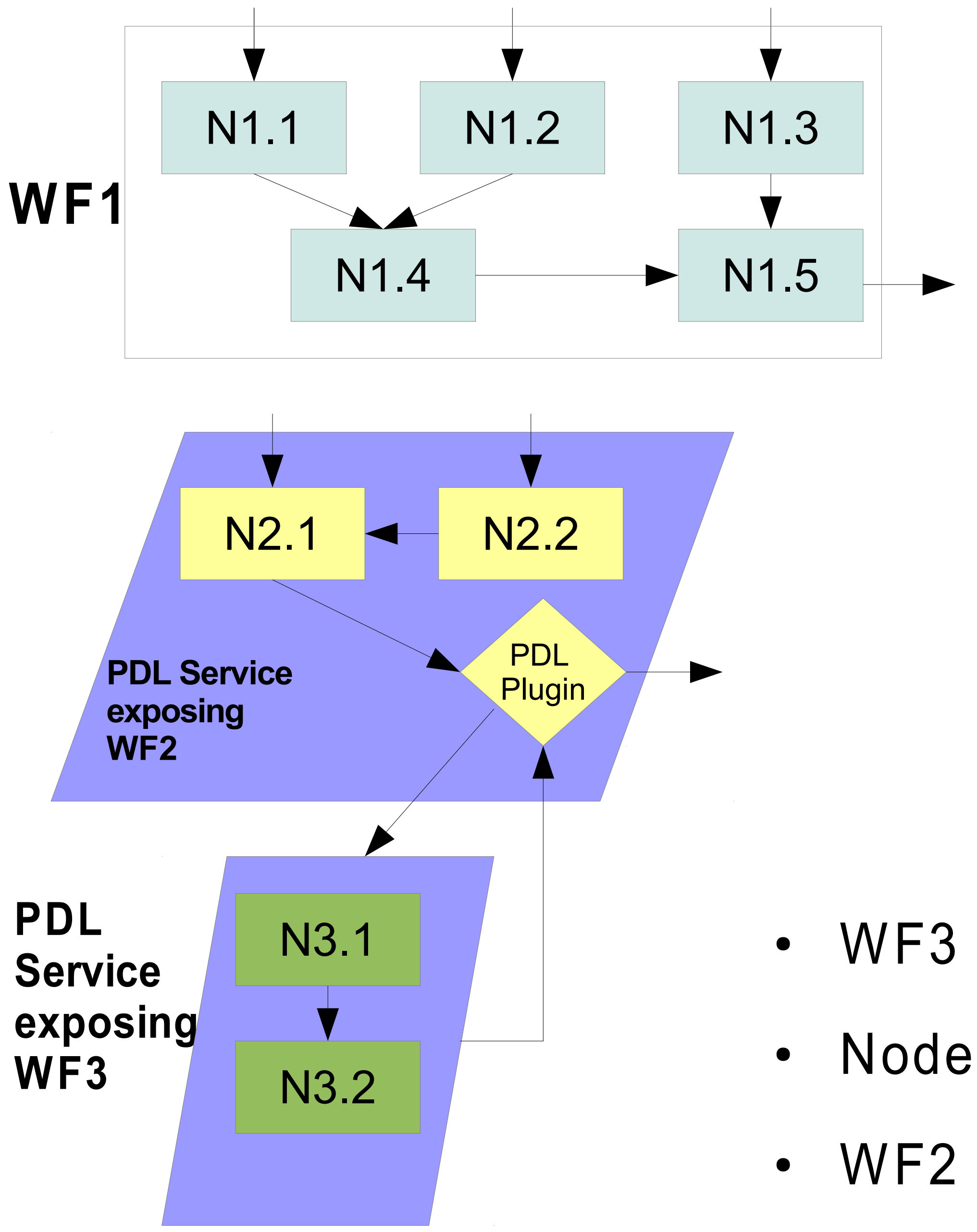
We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

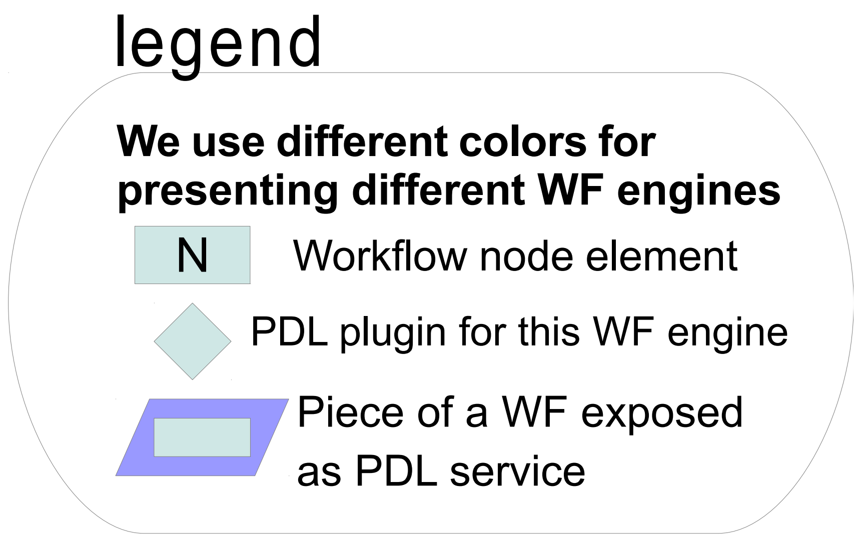
- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**

- WF3 is exposed as a PDL Service
- Node 2.3 calls (using the plugin) the PDL service of WF3

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:





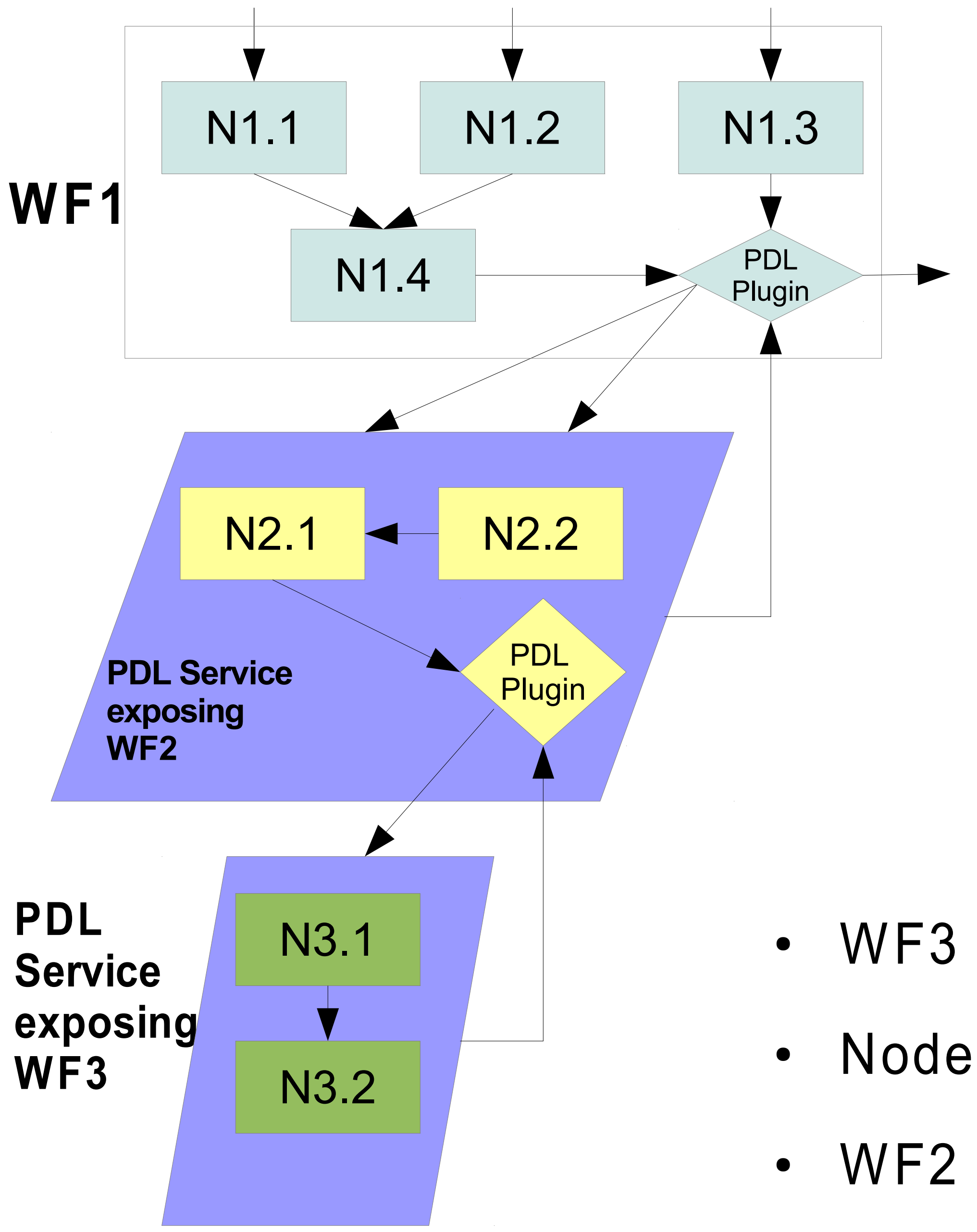
legend
We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**

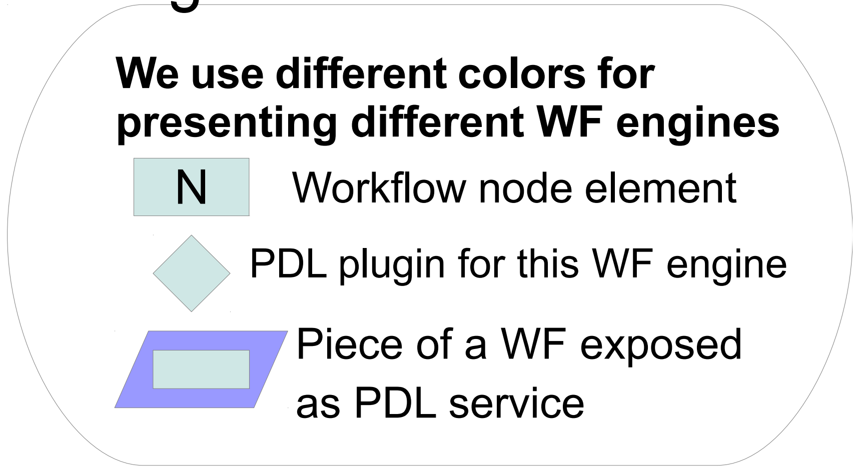
- WF3 is exposed as a PDL Service
- Node 2.3 calls (using the plugin) the PDL service of WF3
- WF2 is exposed as a PDL Service

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:



legend

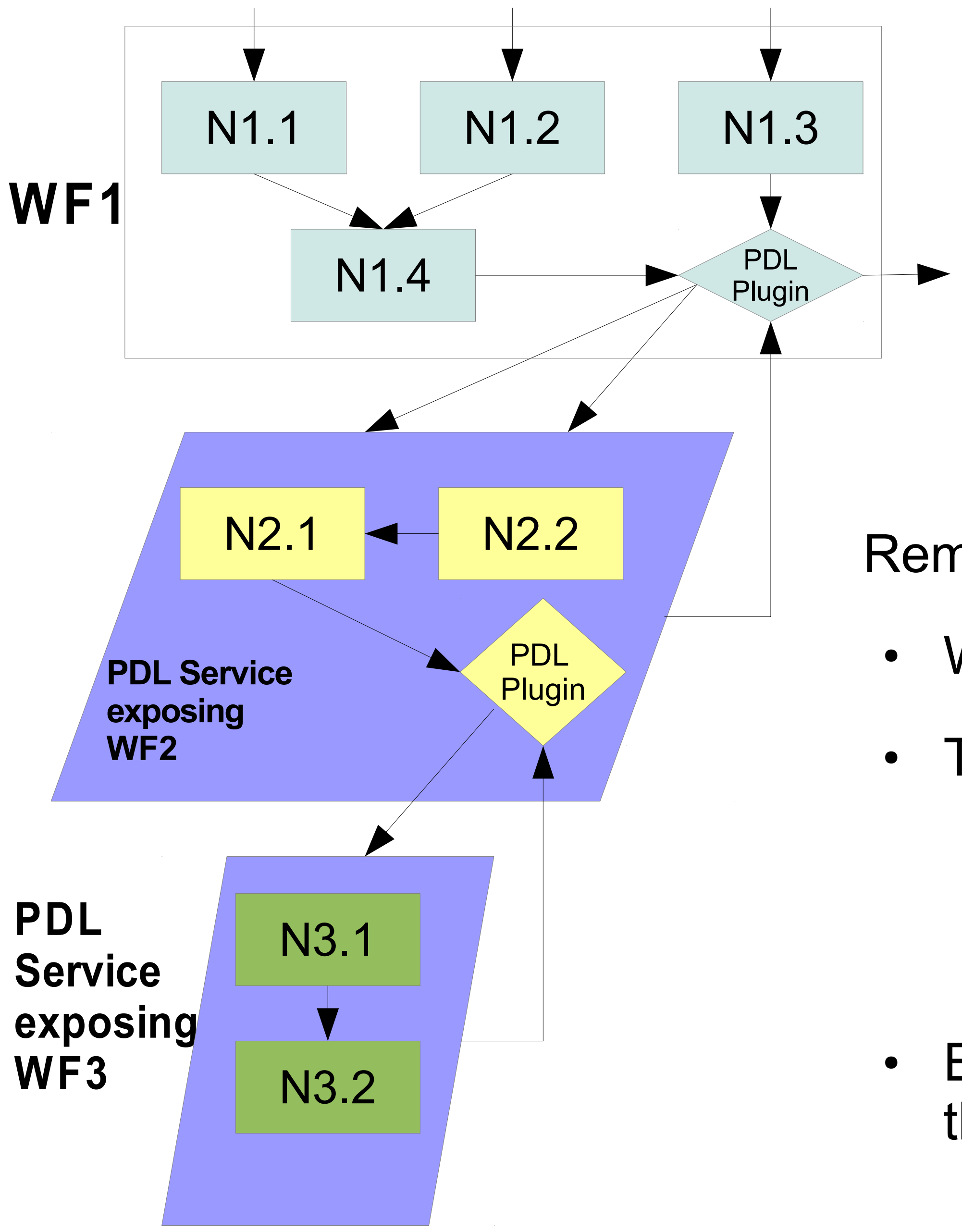

We use different colors for presenting different WF engines
N Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

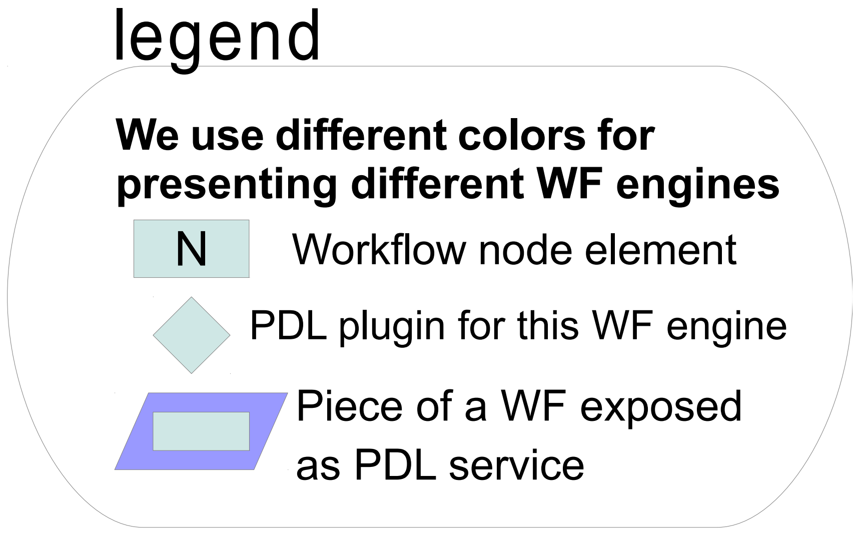
- Assume that we want to use
  - The entire **WF2** as node **N1.5** of **WF1**
  - The entire **WF3** as node **N2.3** of **WF2**
- **Let us see how to perform this with PDL**

- WF3 is exposed as a PDL Service
- Node 2.3 calls (using the plugin) the PDL service of WF3
- WF2 is exposed as a PDL Service
- Node 1.5 calls (using the plugin) the PDL service of WF2

PDL workflows with heterogeneous workflows engines

- PDL allow easy cross communication for workflows using different engines:





legend
We use different colors for presenting different WF engines
N
Workflow node element
PDL plugin for this WF engine
Piece of a WF exposed as PDL service

Remarks

- WF1 can finally call easily element of other Workflow engines
- The resulting WF benefits from PDL advantages
  - Strength scientific oriented interoperabilty
  - Check of interoperabilty graph coherence
- Extracting a PDL service from a piece of Workflow is quick and the procedure could be automatized