



HEIDELBERG INTEROP 2013

PDL STATUS

CARLO MARIA ZWÖLF, FRANCK LE PETIT, PAUL HARRISON.



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



Laboratoire Univers et Théories



The University of Manchester

PDL OVERVIEW

- Parameter Description Language (PDL) is intended to be a lingua franca of parameters:
 - Describes params in a sufficient detailed granularity to allow
 - To generate automatically ad-hoc software from generic elements (client, server,...)
 - To generate verification layers (does parameter satisfy described constraints?)
 - 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 SHORT HISTORY

PDL needs come from scientific services: exemplum from basic service for H2 broadening effect

- Initial level $I \in \mathbb{N}$
- Final level $F \in \mathbb{N}$
- Temperature T in Kelvin
- Electron density ρ in cm^{-3}

Constraints

- $I < F$
- $\frac{9 \rho^{5/3}}{100 T^{1/2}} < 1$

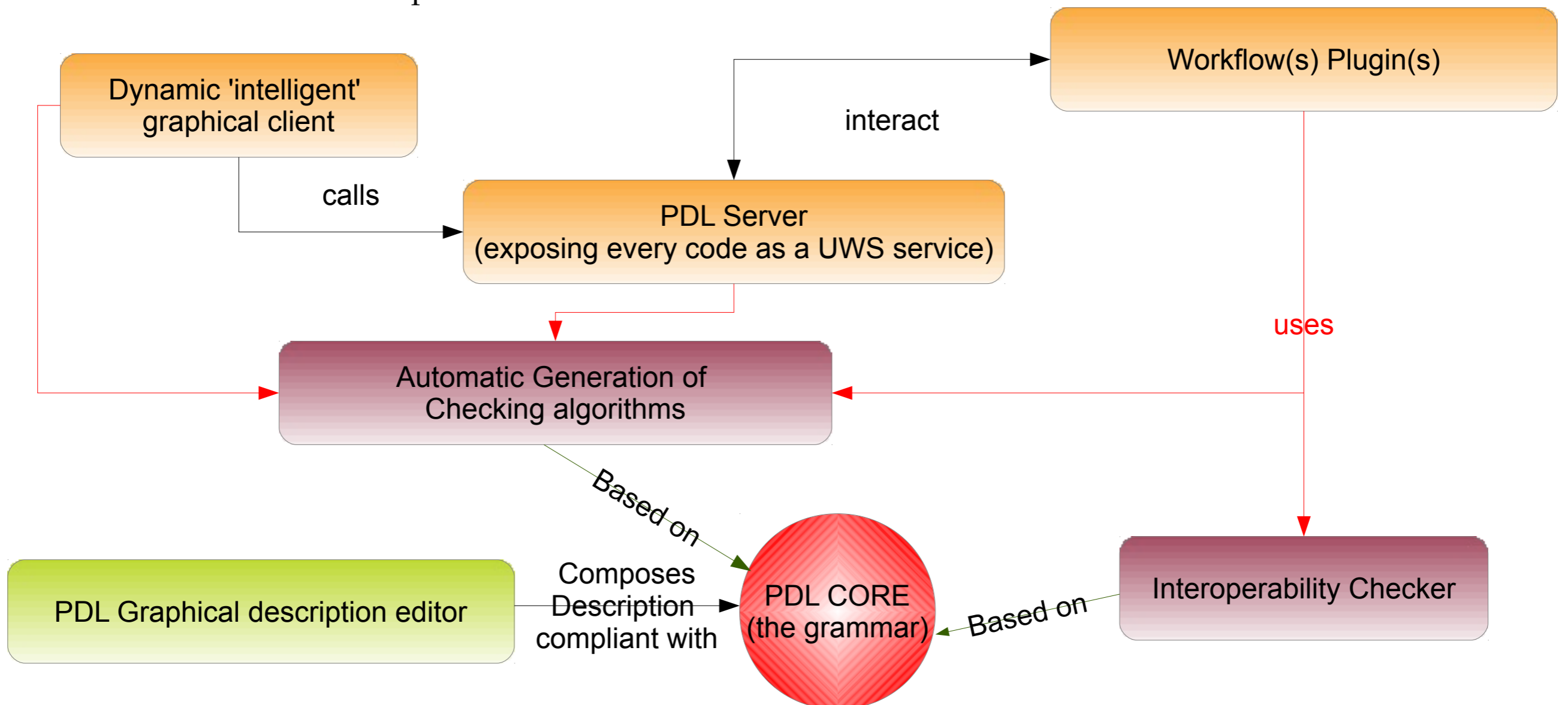
- PDL concepts and core grammar was presented in GWS@Puna,
- First implementation (PDL-dynamic client and core libraries, by Zwölf and Harrison) was presented in [Apps@Urbana](#),
- First IVOA Working Draft presented in [GWS@Urbana](#),
- Implementing note of First implementation available from summer 2012.
- Second implementation (PDL-plugging for Taverna, by Garrido and Ruiz) was presented in GWS@Sao-Paolo
- Third implementation (PDL-Server, by Zwölf) was presented in [Apps@Heidelberg](#)
- Fourth implementation (PDL-Graphical description editor, by Savalle) is under development, see you @Hawaii

Architecture of PDL complete solution



Parameters and constraints are finely described with fine grained granularity:

- Generic software elements are 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



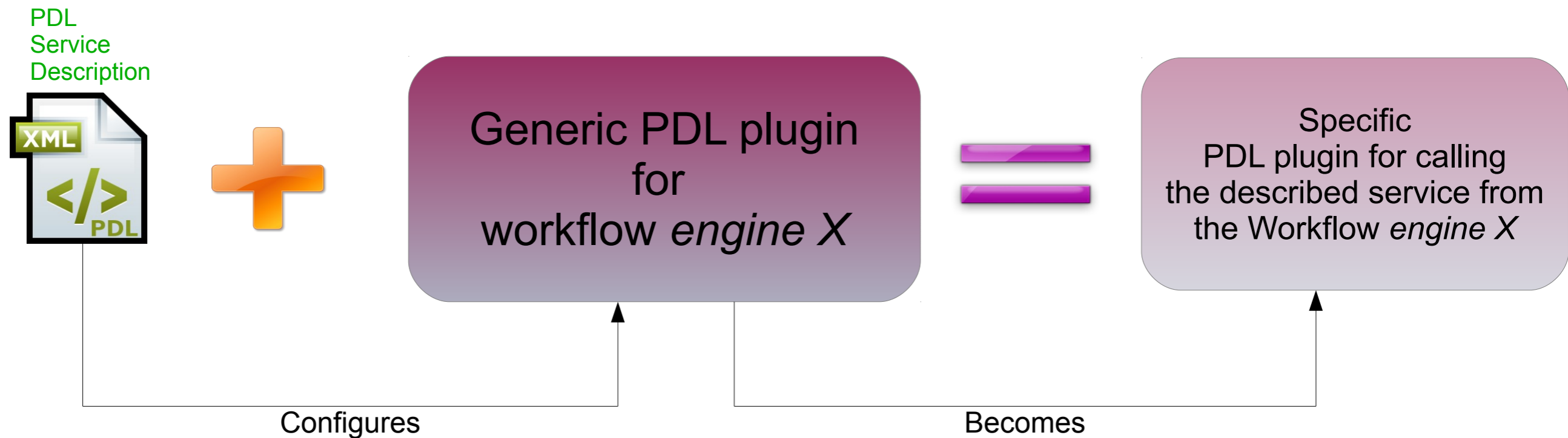
Client and Server

Find client and server presentation (with nice screen cast) at URL:

<http://wiki.ivoa.net/twiki/bin/view/IVOA/InterOpMay2013Applications>

(Application 2, Carlo Maria Zwölf, PDL service for Paris-Durham MHD Shock Code)

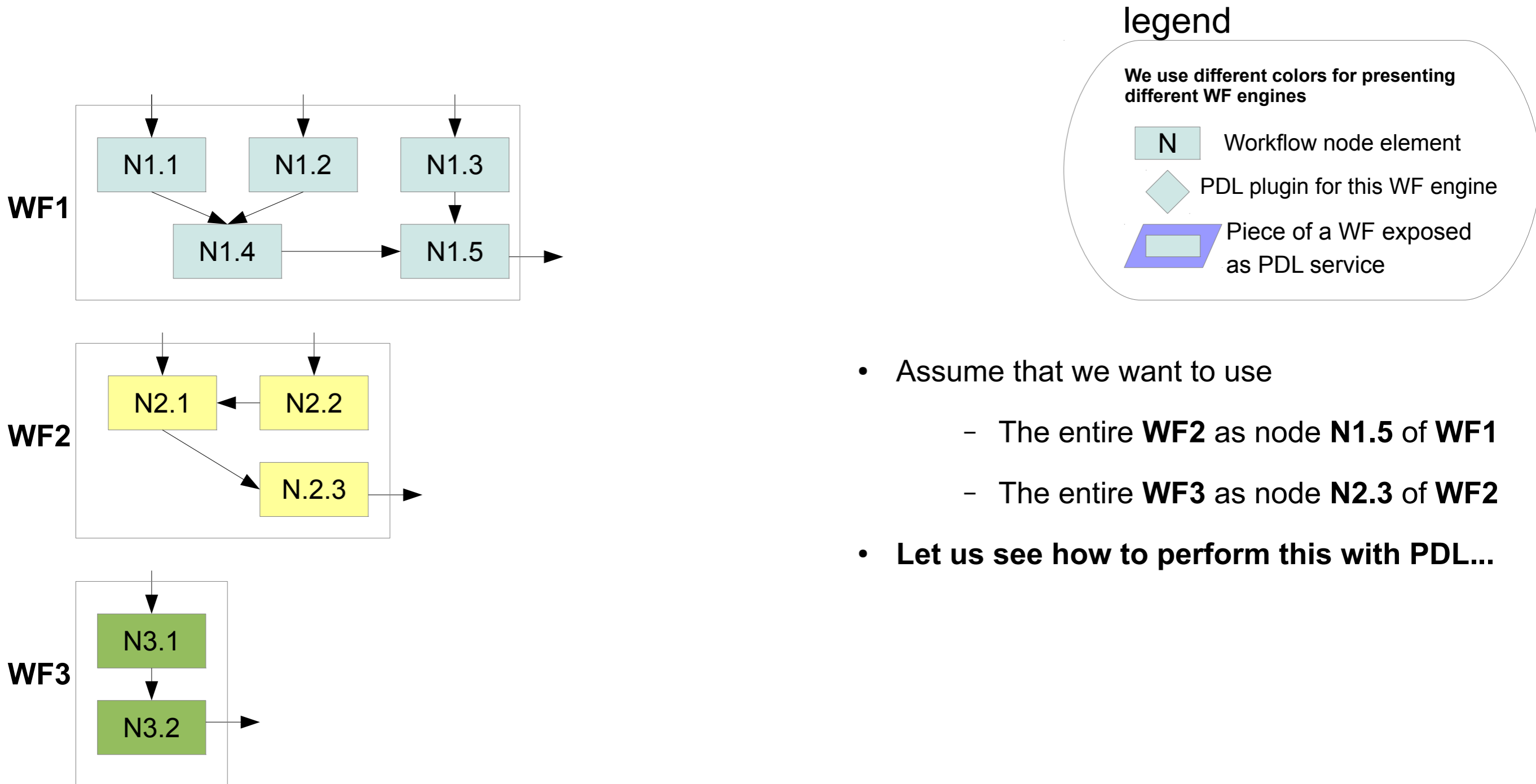
PDL and Workflows : the plugin



- In current developments (Garrido & Ruiz from WF4ever) *engine X* is Taverna
- For a given workflow engine X
 - Generic plugin is written once and for all
 - All the services will use the Plugin configured by the ad hoc description file
 - Maintenance is strongly reduced due to this industrialization.

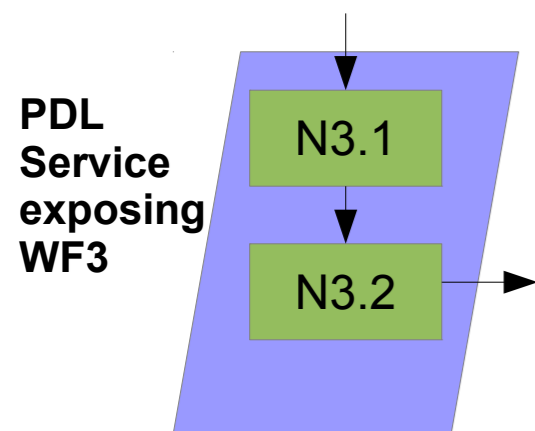
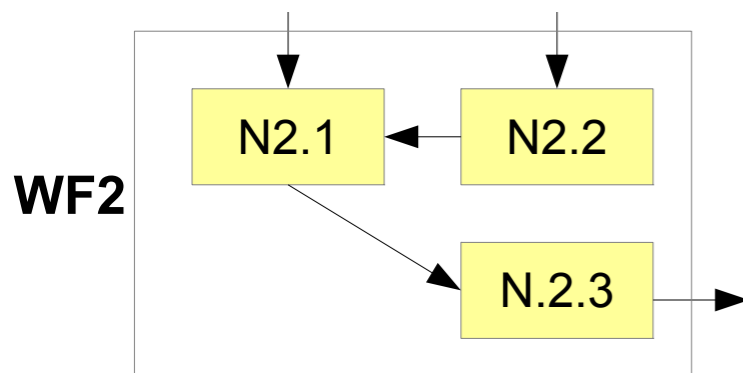
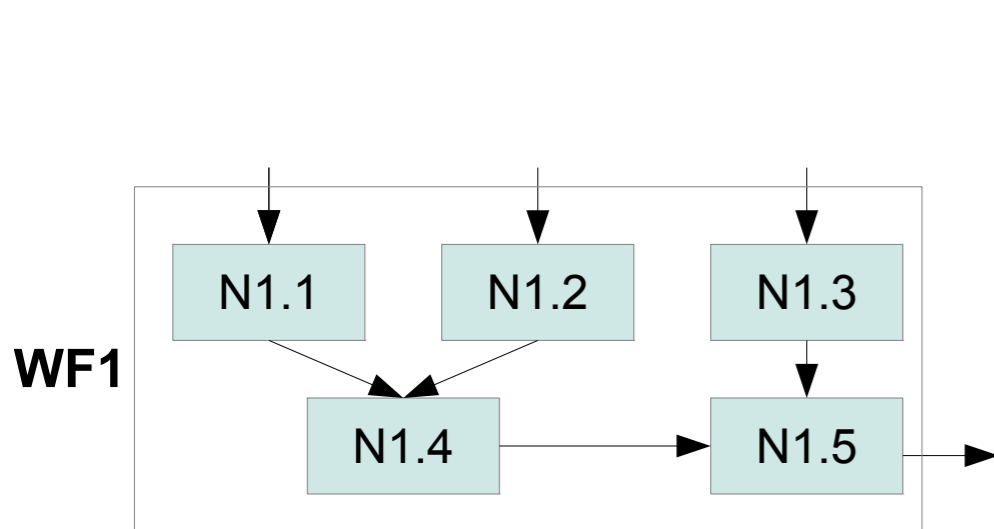
PDL workflows with heterogeneous workflows engines

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:






PDL workflows with heterogeneous workflows engines

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:



legend

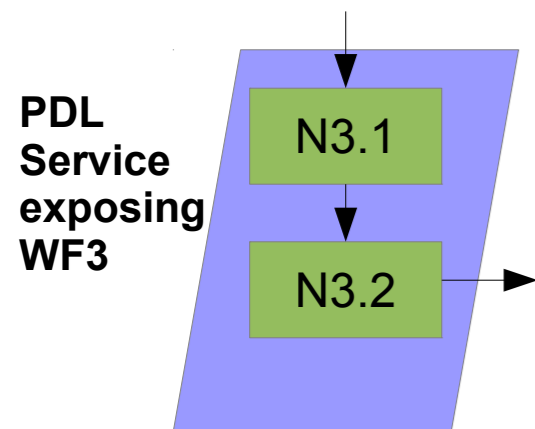
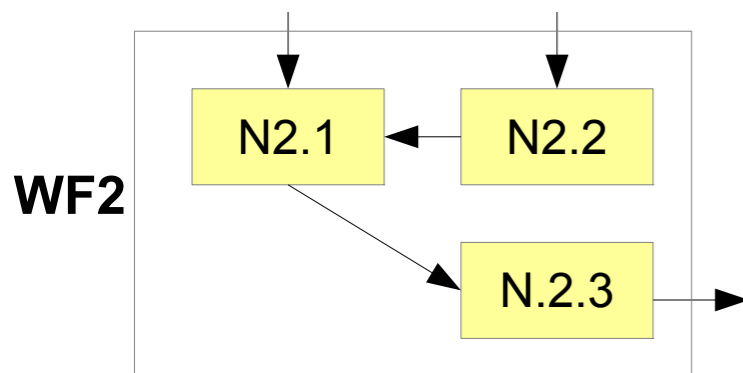
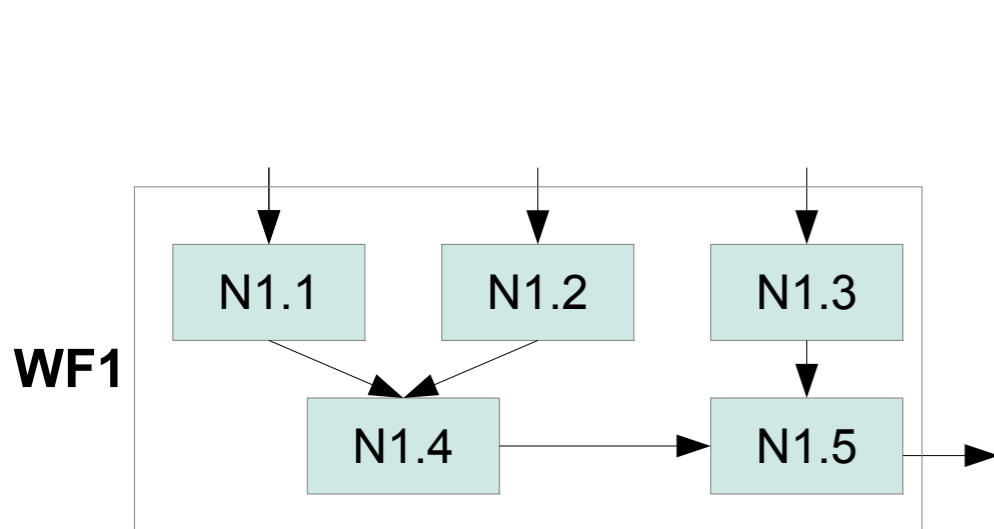
We use different colors for presenting different WF engines

-  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

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:



legend

We use different colors for presenting different WF engines

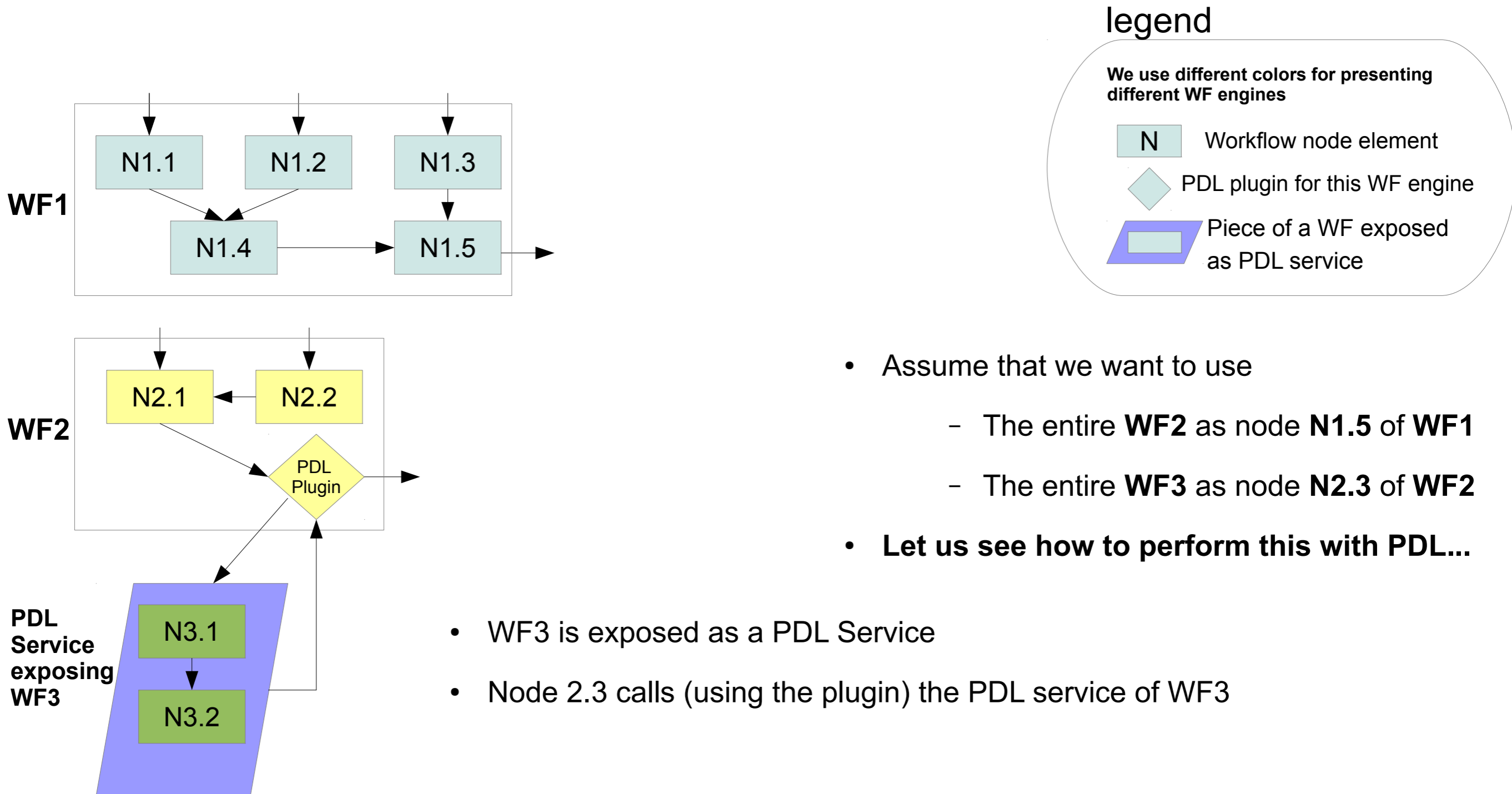
-  Workflow node element
-  PDL plugin for this WF engine
-  Piece of a WF exposed as PDL service

- 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

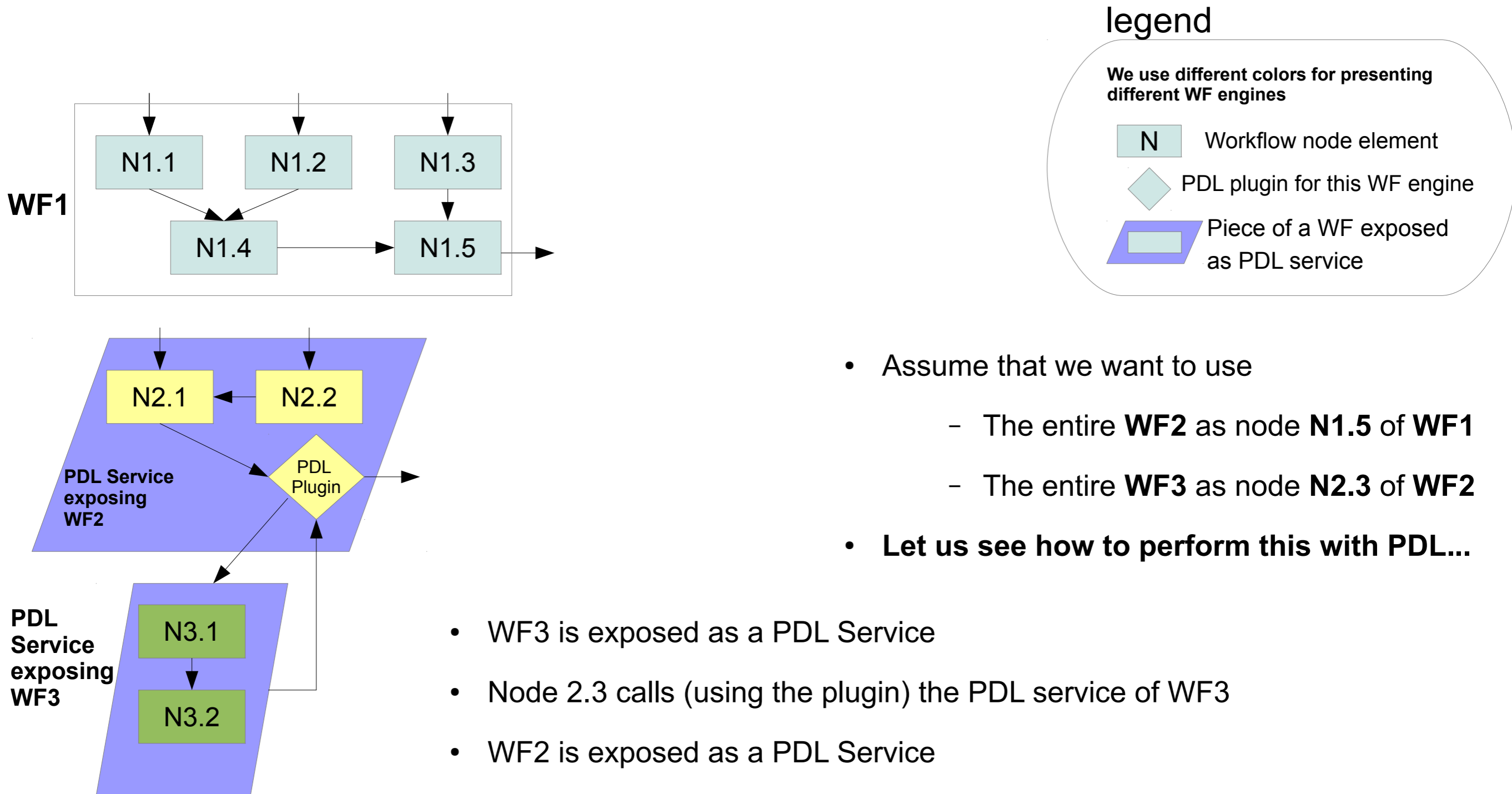
PDL workflows with heterogeneous workflows engines

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:



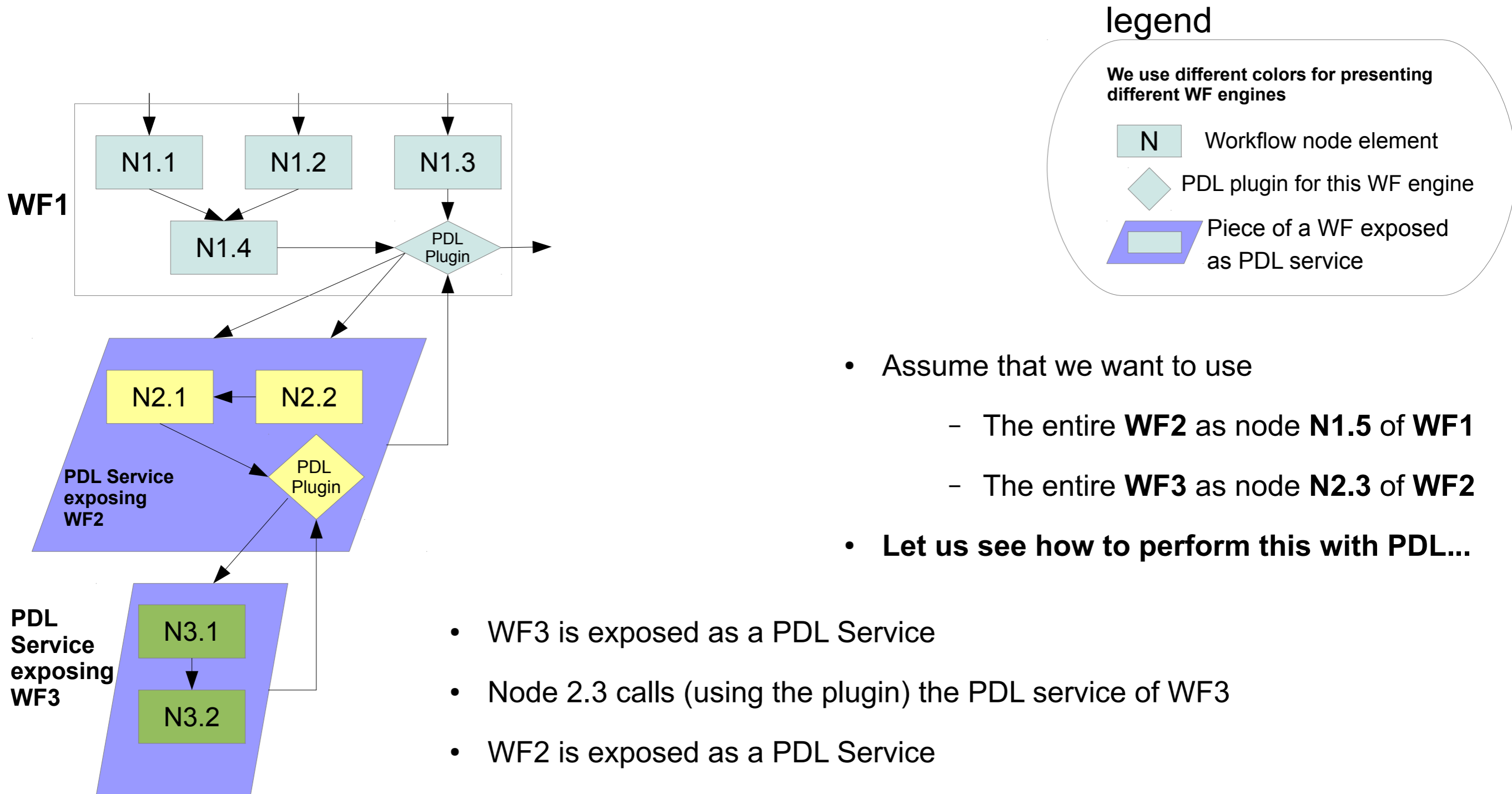
PDL workflows with heterogeneous workflows engines

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:



PDL workflows with heterogeneous workflows engines

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:



- 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

- Unexpected feature when we started PDL, it appeared on the road
- PDL allow easy cross communication for workflows using different engines:

