Observatoire de PARIS

OV France Workflow Day

PDL and its framework:

Concepts, Client, Server,...

Carlo Maria Zwölf

LERMA

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

PDL: Why and What is it?

Scientific real use case : Service for broadening computations

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

Constraints

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

PDL: Why and What is it?

Scientific real use case : Service for broadening computations

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

Constraints

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

- Existing solutions (Wadl, WSDL) for describing services does not fit the scientific needs:
  - There is no description of algorithms, physics and utility behind a given service (one has to know *a priori* the service for using it)
  - There is no description about the physical meaning of parameters and units
  - Descriptions are only in a computer science sense.
  - Interoperability is understood only in a basic computer science way.

Motivations

- PDL aim is to answer to two major issues in scientific services

Description needs				
Describe physical properties of parameters				
Nature	Meaning	Unit	Precision	Range
Describe complex relations involving parameters				
Physical constraints	Arbitrary Conditions	Mathematical Conditions		

Interoperability needs		
Interaction of two services has sense if the parameter sent by the first and expected by the second have same		
Computer type	Physical concept	Unit
Interaction of two services has sense if all preconditions of second service are satisfied by output of first one		

Motivations

- PDL aim is to answer to two major issues in scientific services

Description needs

Describe physical properties of parameters

Nature	Meaning	Unit	Precision	Range

Describe complex relations involving parameters

Physical constraints	Arbitrary Conditions	Mathematical Conditions

Existing solutions (*WADL* & *WSDL*) don't fit this fine scientific need

Interoperability needs

Interaction of two services has sense if the parameter sent by the first and expected by the second have same

Computer type	Physical concept	Unit

Interaction of two services has sense if all preconditions of second service are satisfied by output of first one

Existing workflow engines (*Babel, Taverna, OSGI, OPalm, GumTree*) implements interoperability only in a “basic” computer way

Motivations

- PDL aim is to answer to two major issues in scientific services

Description needs				
Describe physical properties of parameters				
Nature	Meaning	Unit	Precision	Range
Describe complex relations involving parameters				
Physical constraints		Arbitrary Conditions	Mathematical Conditions	

Interoperability needs		
Interaction of two services has sense if the parameter sent by the first and expected by the second have same		
Computer type	Physical concept	Unit
Interaction of two services has sense if all preconditions of second service are satisfied by output of first one		

PDL is a rigorous grammar for
- Finely describing the set of parameters (**inputs** & **outputs**) in a way that
  - Can be understood easily by humans
  - Can be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

PDL description capabilities meet:

- The "*scientific*" description needs
- The "*scientific*" workflow needs

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.
- All the rules and specifications are detailed into the Working Draft

Get the PDL working draft → [**pdl.obspm.fr**](pdl.obspm.fr)

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.
- All the rules and specifications are detailed into the Working Draft

Get the PDL working draft → **pdl.obspm.fr**

Examples of description capabilities

Input:

- $\boldsymbol{p}_1$ is a m/s vector speed and $\|\boldsymbol{p}_1\| < c$
- p_2 is a Kelvin temperature and $p_2 > 0$
- p_3 is a kg mass and $p_3 \geq 0$

Output:

- p_4 is a Joule Energy and $p_4 \geq 0$

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.
- All the rules and specifications are detailed into the Working Draft

Get the PDL working draft → pdl.obspm.fr

Examples of description capabilities

Input:

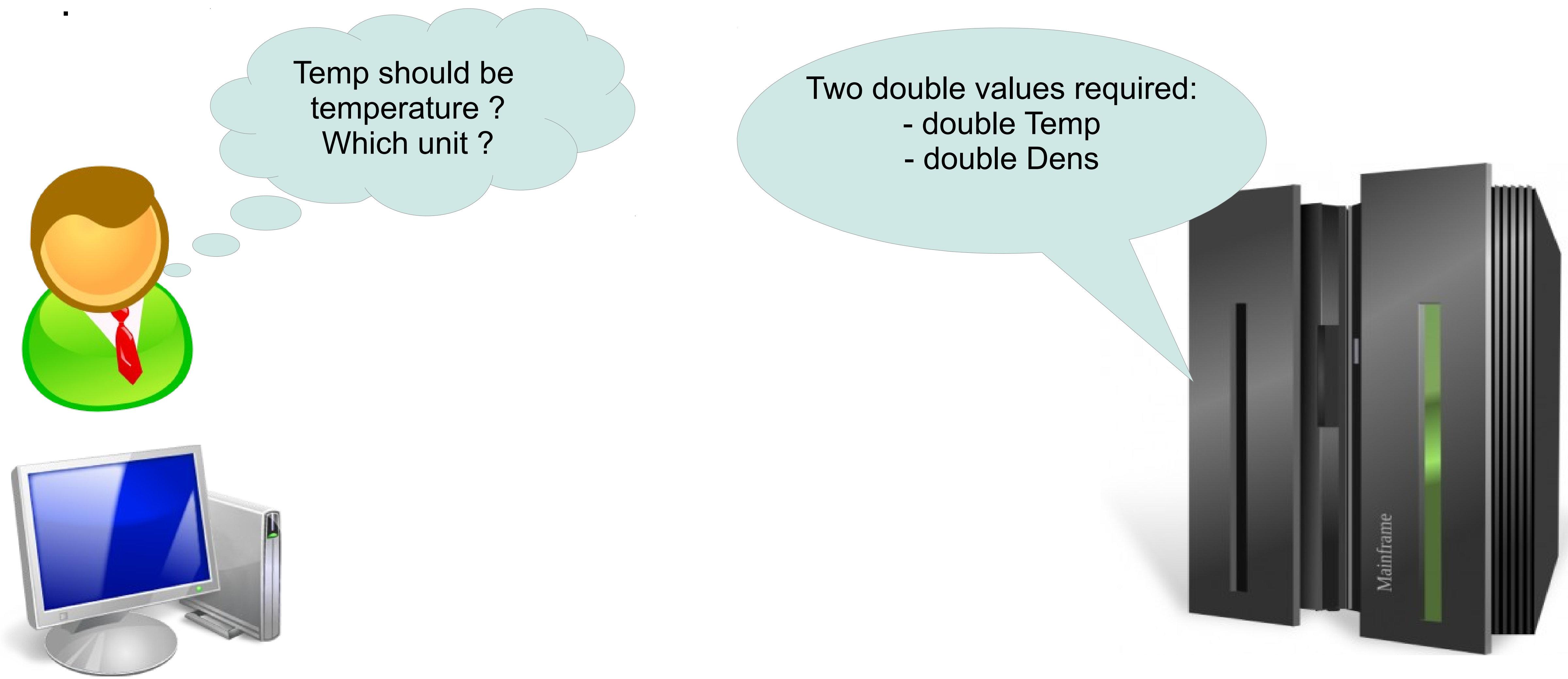
- $\mathbb{R} \ni p_1 > 0;\ p_2 \in \mathbb{N};\ p_3 \in \mathbb{R}$
- if $p_1 \in]0, \pi/2]$ then
  $p_2 \in \{2; 4; 6\}$, $p_3 \in [-1, +1]$ and $(|\sin(p_1)^{p_2} - p_3|)^{1/2} < 3/2$.
- if $p_1 \in]\pi/2, \pi]$ then
  $0 < p_2 < 10$, $p_3 > \log(p_2)$ and $(p_1 \cdot p_2)$ must belong to $\mathbb{N}$.

Output:

- $\boldsymbol{p}_4, \boldsymbol{p}_5 \in \mathbb{R}^3$
- Always $\dfrac{||\boldsymbol{p}_5||}{||\boldsymbol{p}_4||} \leq 0.01$.

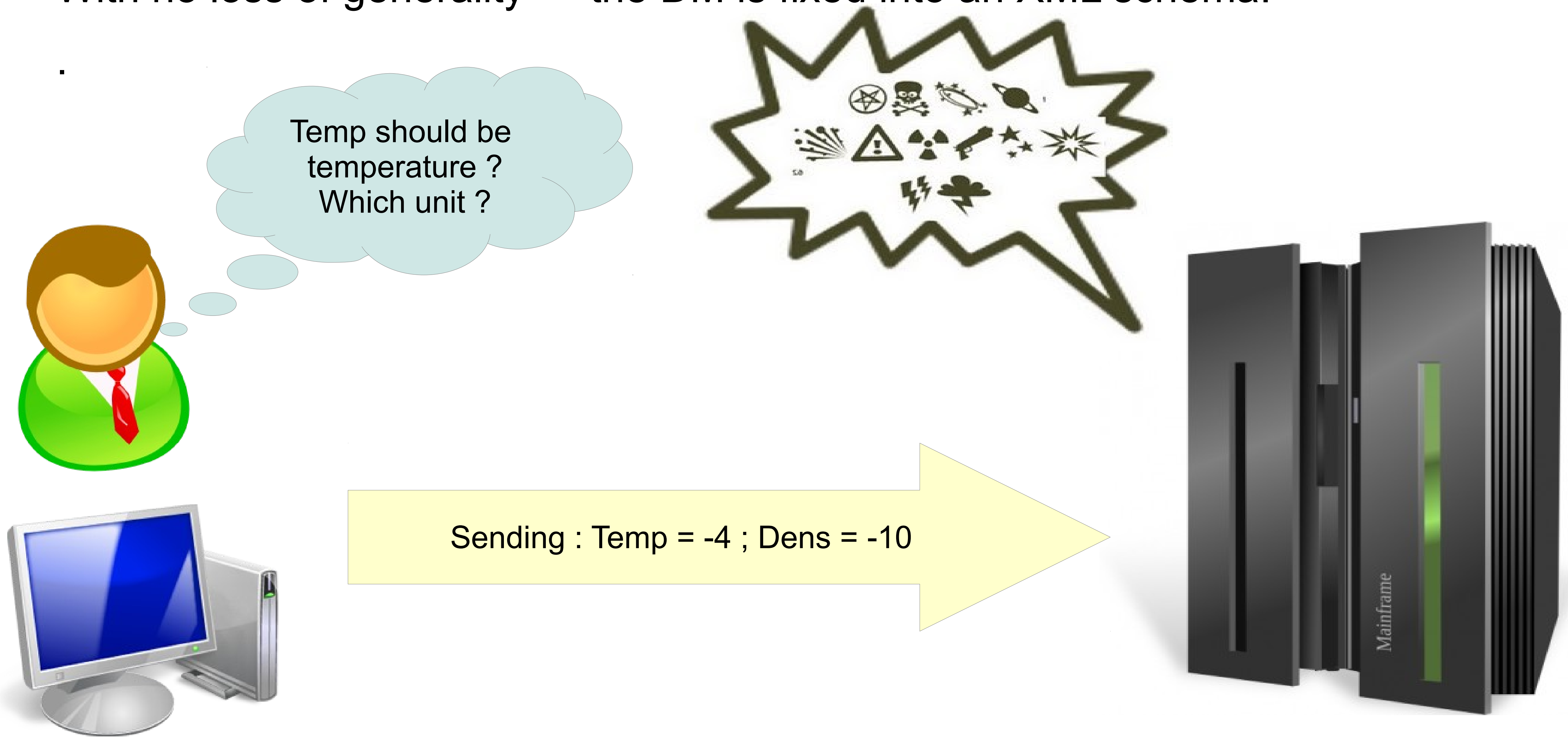
PDL Principles

- 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.



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.



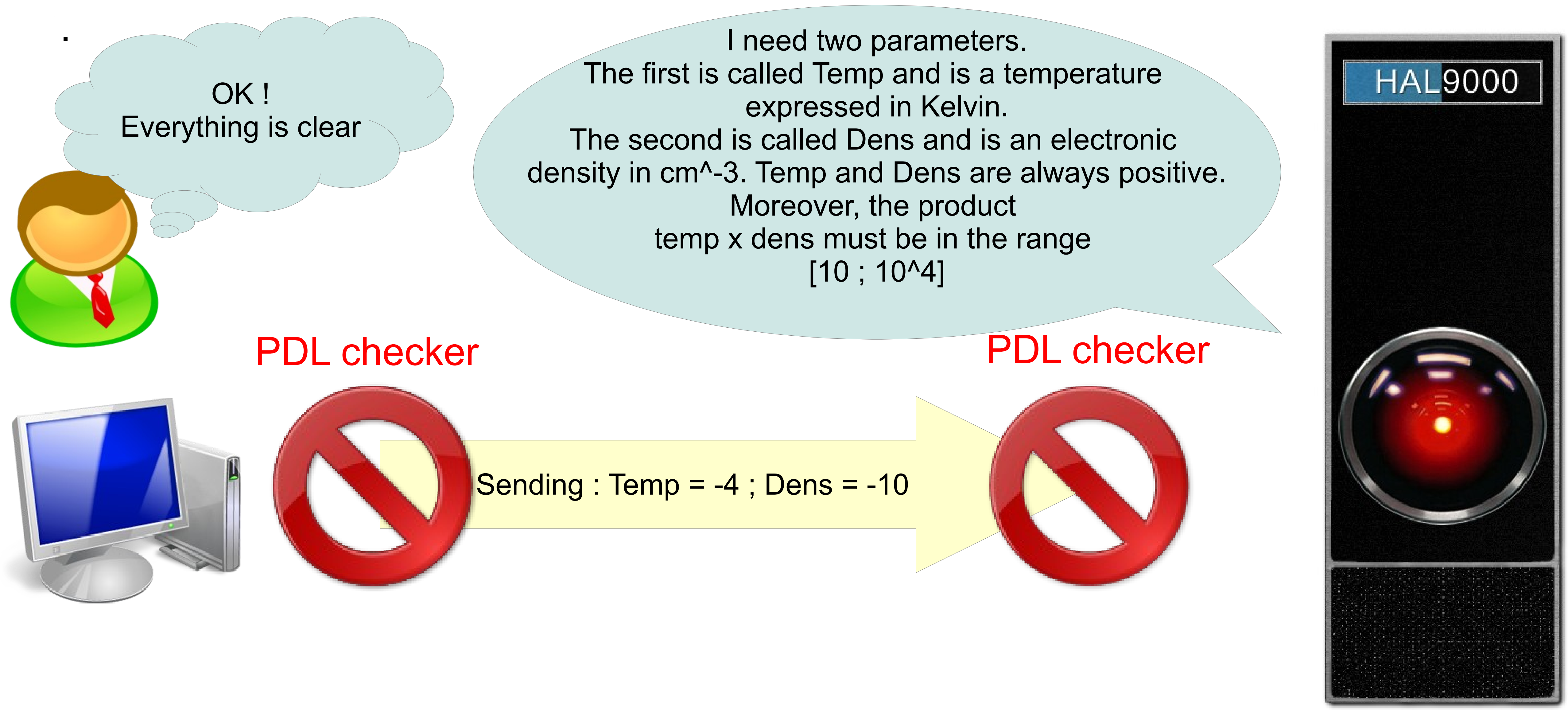
PDL Principles

- 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.



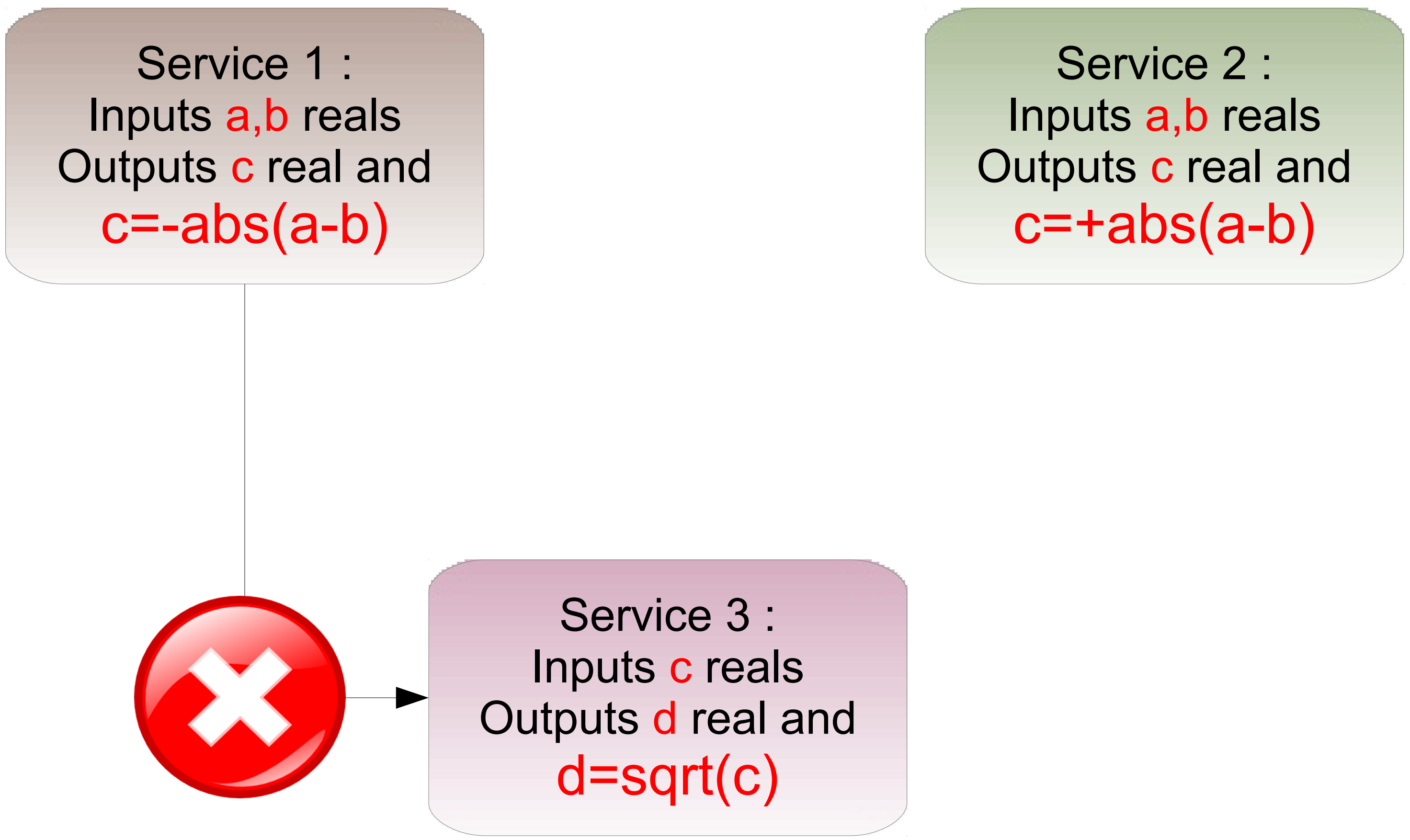
PDL and interoperability

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

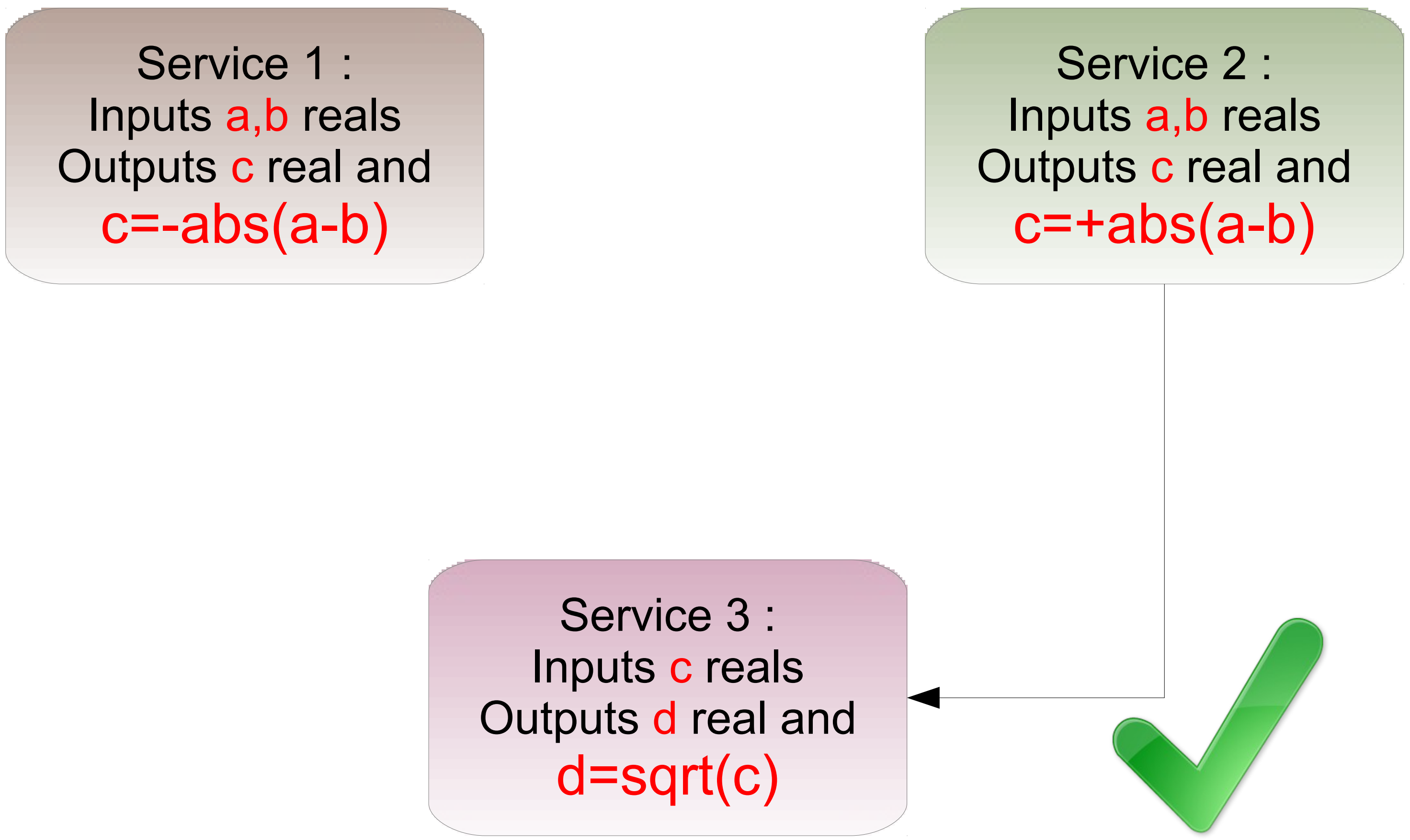
Service 2 :
Inputs a,b reals
Outputs c real and
$c=+abs(a-b)$

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

PDL and interoperability



PDL and interoperability



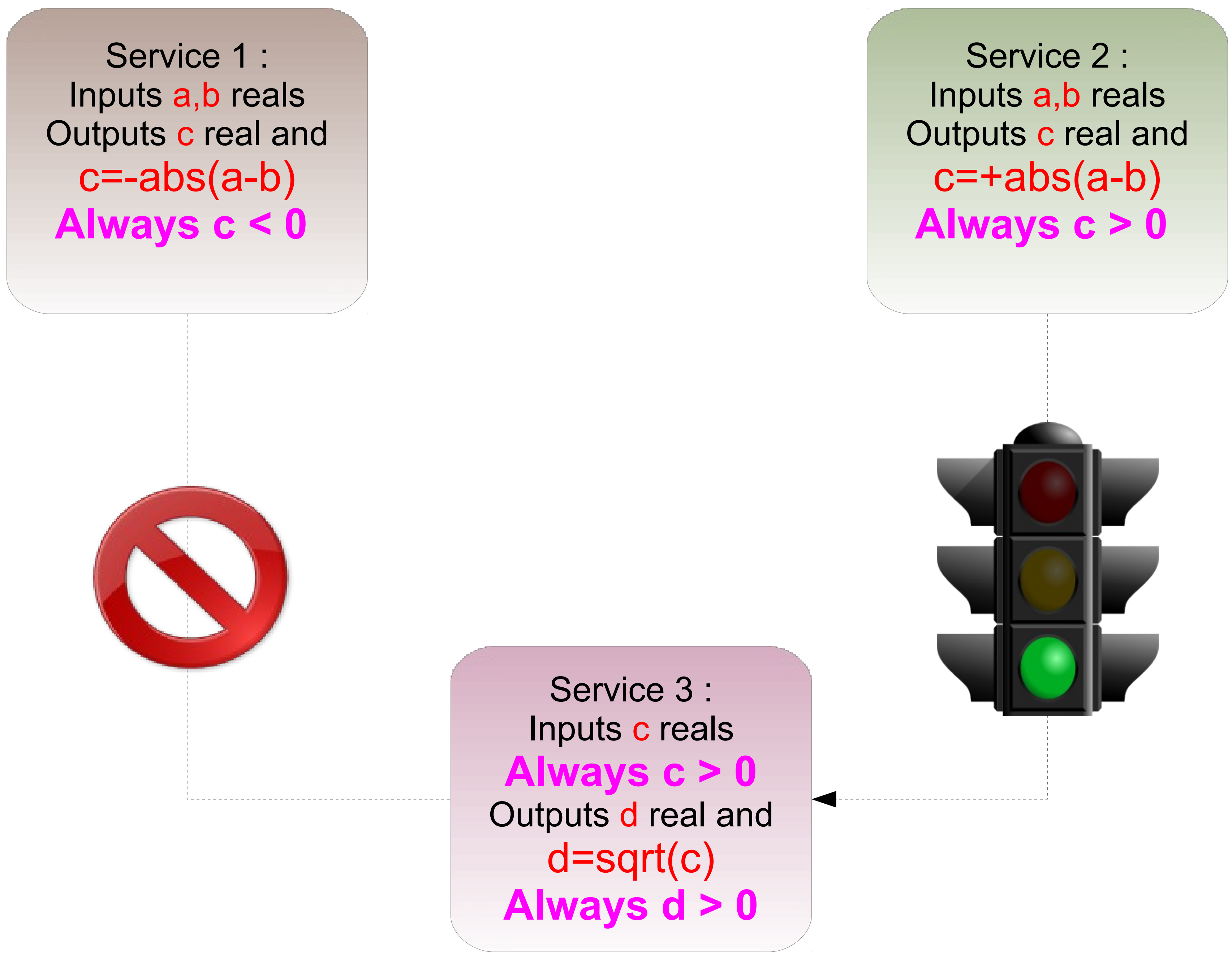
PDL and interoperability

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

PDL and interoperability



PDL and interoperability

Let

- S_1 and S_2 be two services.
- $p^j(S_i)$ be the jth parameter of S_i.
- $\mathcal{I}(S_i)$ (resp. $\mathcal{O}(S_i)$) be the set of input (resp. output) parameters of S_i.
- $\mathcal{C}^{p^j}_{\mathcal{I}(S_i)}$ (resp. $\mathcal{C}^{p^j}_{\mathcal{O}(S_i)}$) the set of all constraints on $\mathcal{I}(S_i)$ (resp. $\mathcal{O}(S_i)$) involving p^j.

S_2 could follow S_1 into a workflow iff $\forall p^k(S_2) \in \mathcal{I}(S_2)\, \exists\, p^l(S_1) \in \mathcal{O}(S_1)$ such that:

- $p^k(S_2) = p^l(S_1)$
- $p^l(S_1)$ satisfies $\mathcal{C}^{p^l}_{\mathcal{O}(S_1)} \implies p^k(S_2)$ satisfies $\mathcal{C}^{p^k}_{\mathcal{I}(S_2)}$

- The equality is in the sense that parameters have same
  - UCDs
  - UTypes
  - SkossConcepts
  - Units

PDL main corollaries

Since parameters and constraints are finely described with fine grained granularit,y many possibilities are open:

- Generic elements could be automatically generated
- These generic elements are “configured” by a specific PDL description instance

PDL main corollaries

Since parameters and constraints are finely described with fine grained granularity, many possibilities are open:

- Generic elements could be automatically generated
- These generic elements are "configured" by a specific PDL description instance

Dynamic 'intelligent' client

Taverna Plugin

PDL Server
(exposing every code as a UWS service)

Automatic Generation of
Checking algorithms

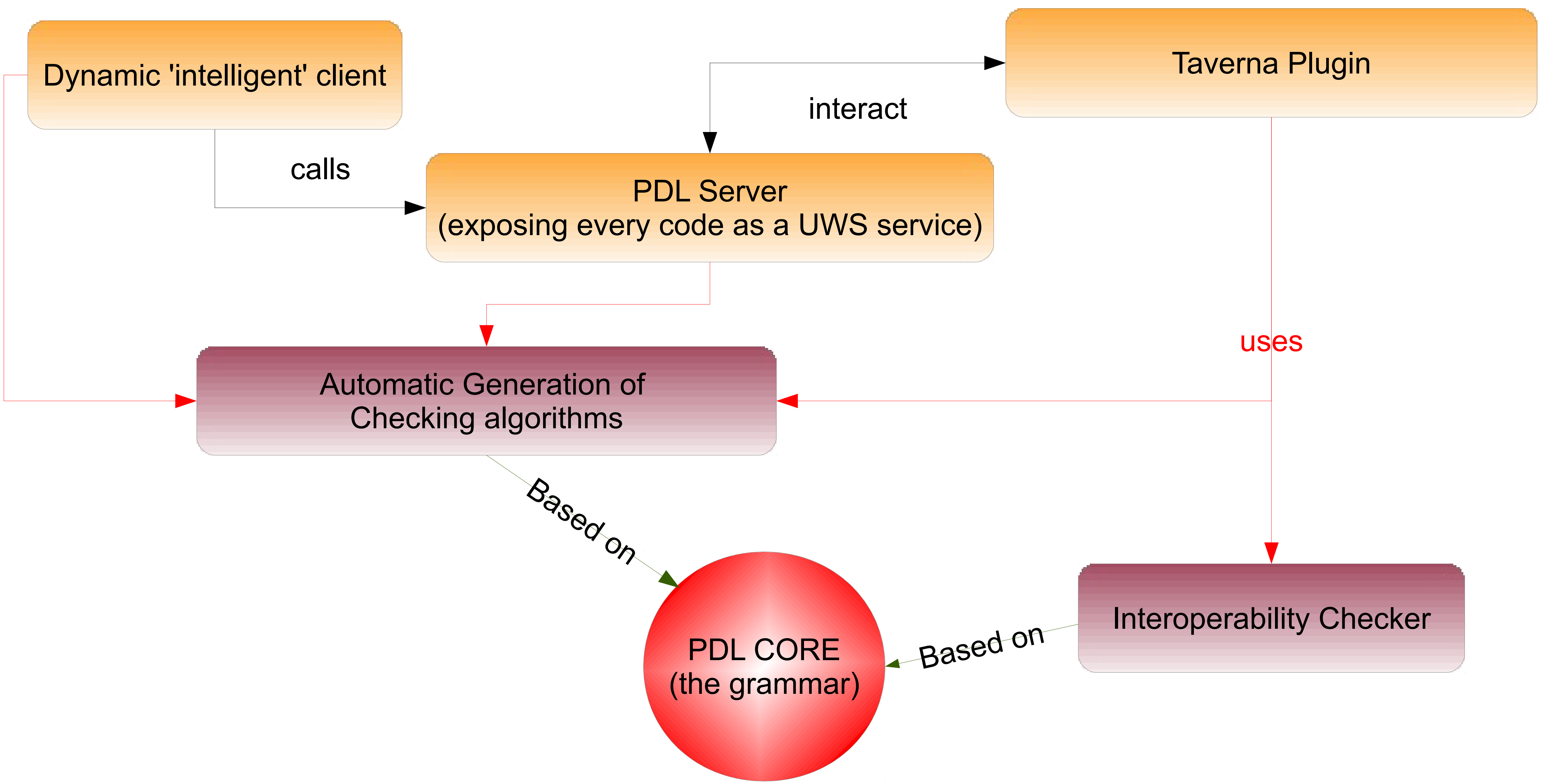
PDL CORE
(the grammar)

Interoperability Checker

PDL main corollaries

Since parameters and constraints are finely described with fine grained granularity, many possibilities are open:

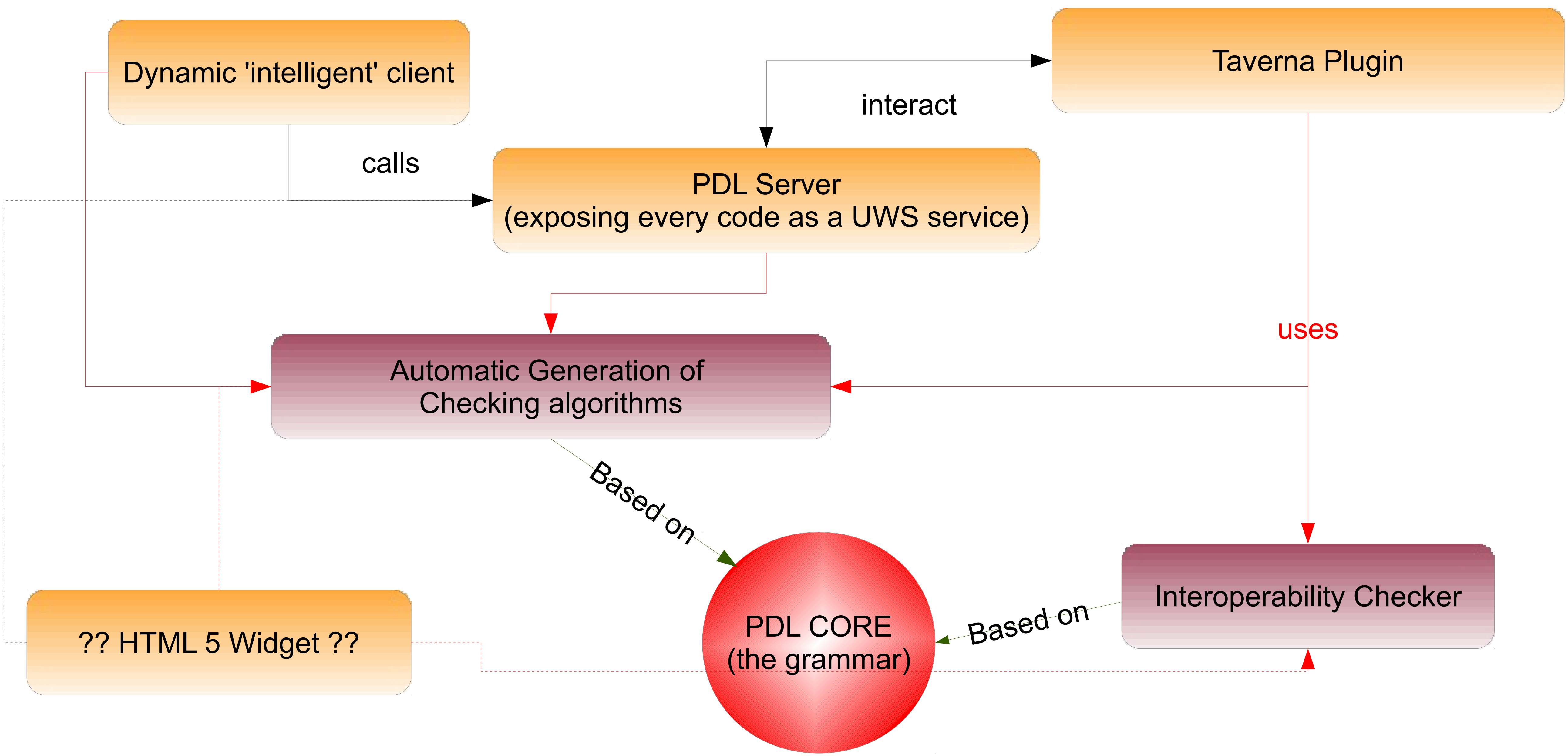
- Generic elements could be automatically generated
- These generic elements are “configured” by a specific PDL description instance



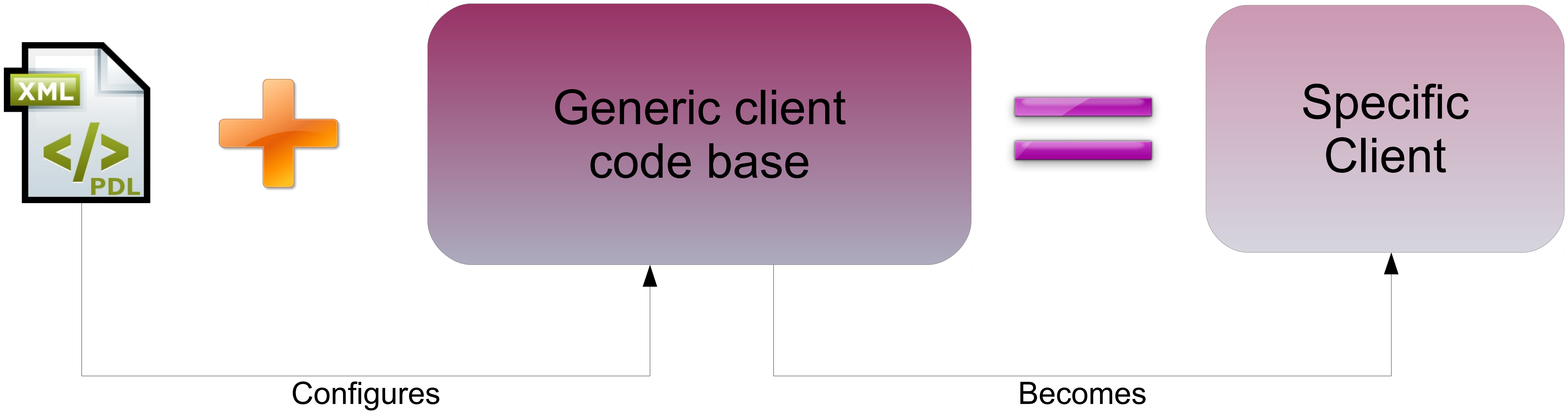
PDL main corollaries

Since parameters and constraints are finely described with fine grained granularity, many possibilities are open:

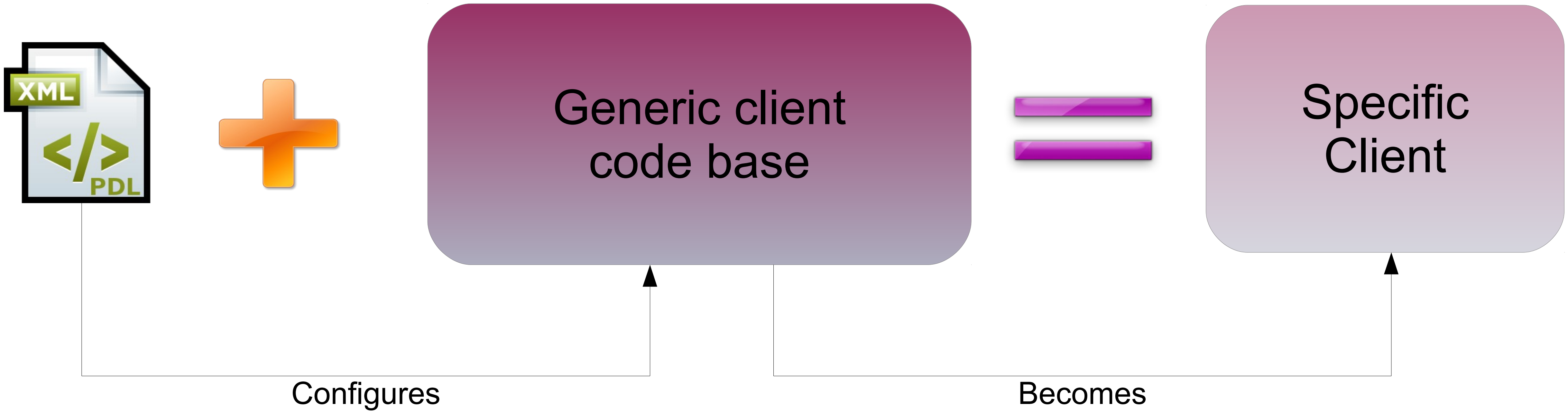
- Generic elements could be automatically generated
- These generic elements are “configured” by a specific PDL description instance



The Dynamic client

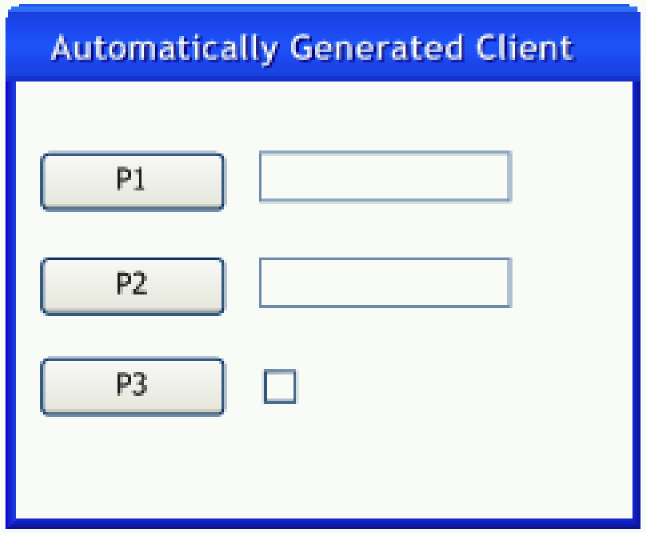


The Dynamic client

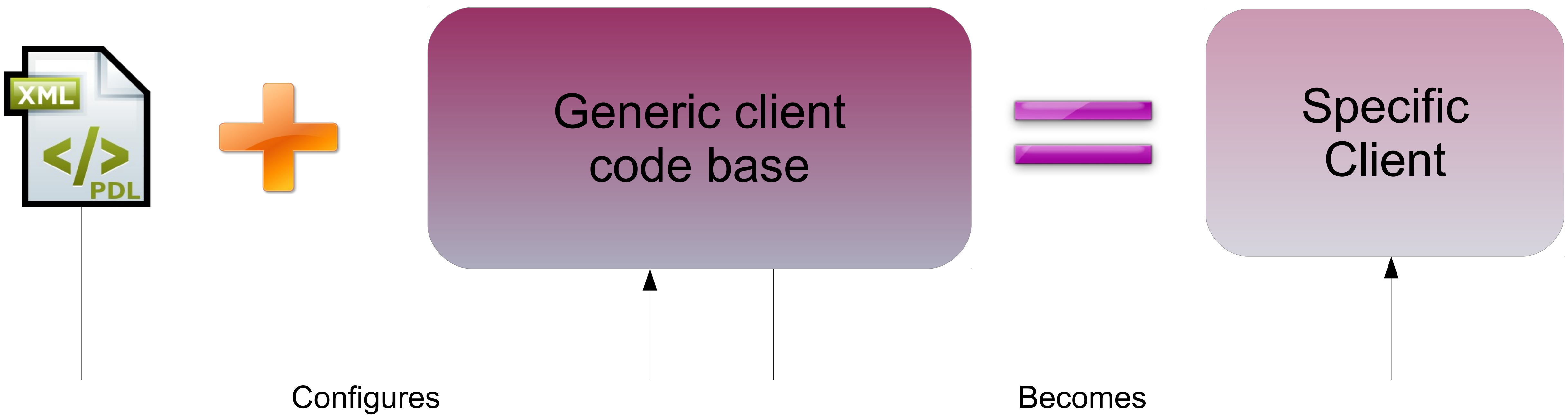


Service description:

- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ and p_3 is boolean.
- if $p_1 > 0 \implies p_2 \in \{2; 4; 6\}$ and p_3 must be false.
- if $p_1 < 0 \implies p_3$ must be true.

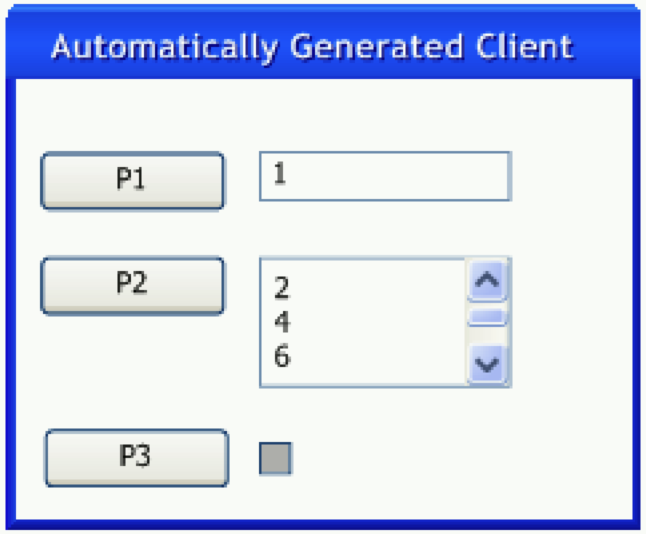


The Dynamic client

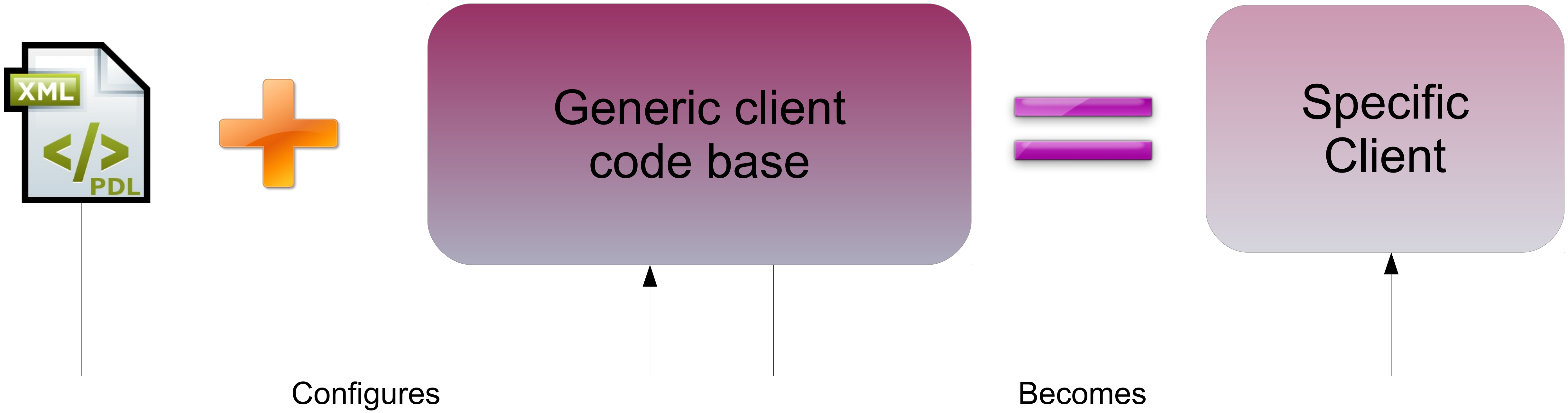


Service description:

- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ and p_3 is boolean.
- if $p_1 > 0 \implies p_2 \in \{2; 4; 6\}$ and p_3 must be false.
- if $p_1 < 0 \implies p_3$ must be true.

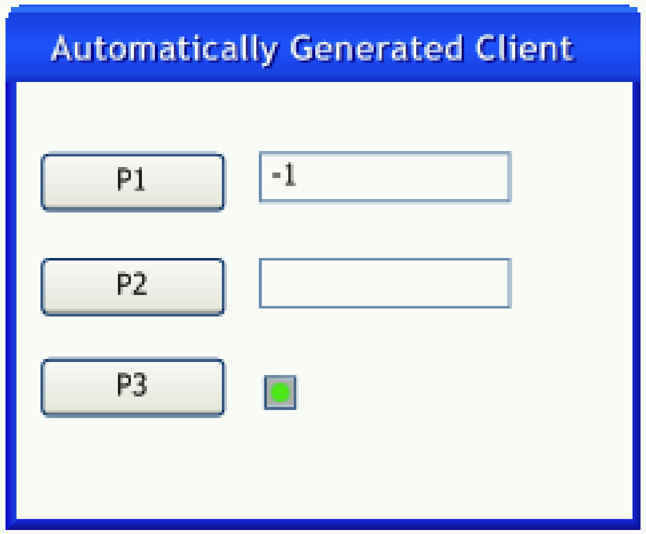


The Dynamic client

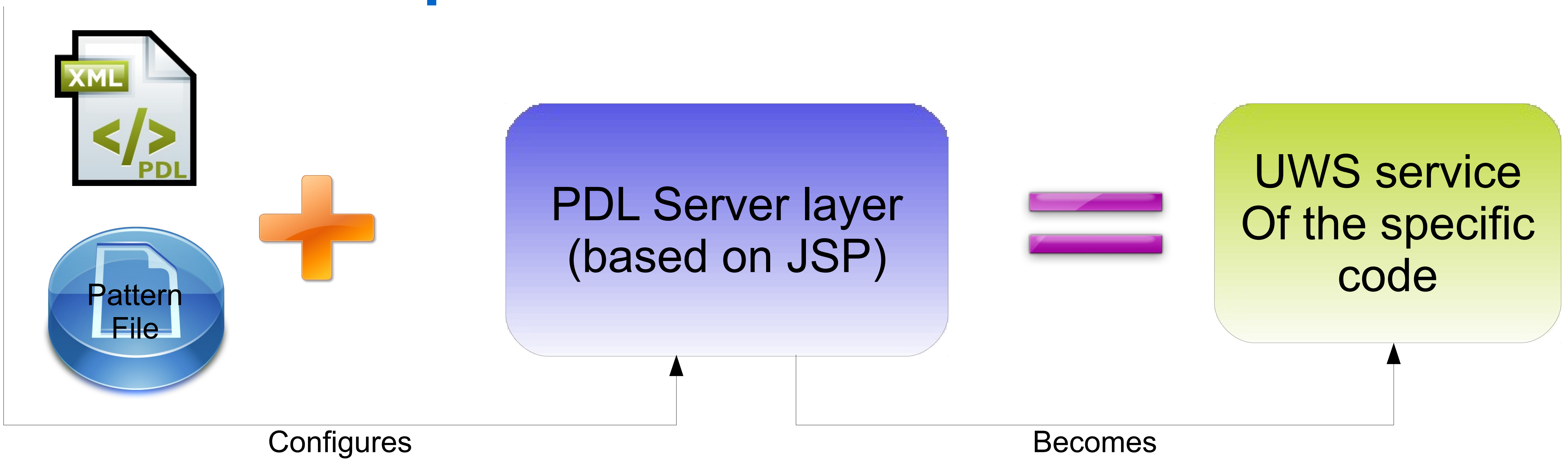


Service description:

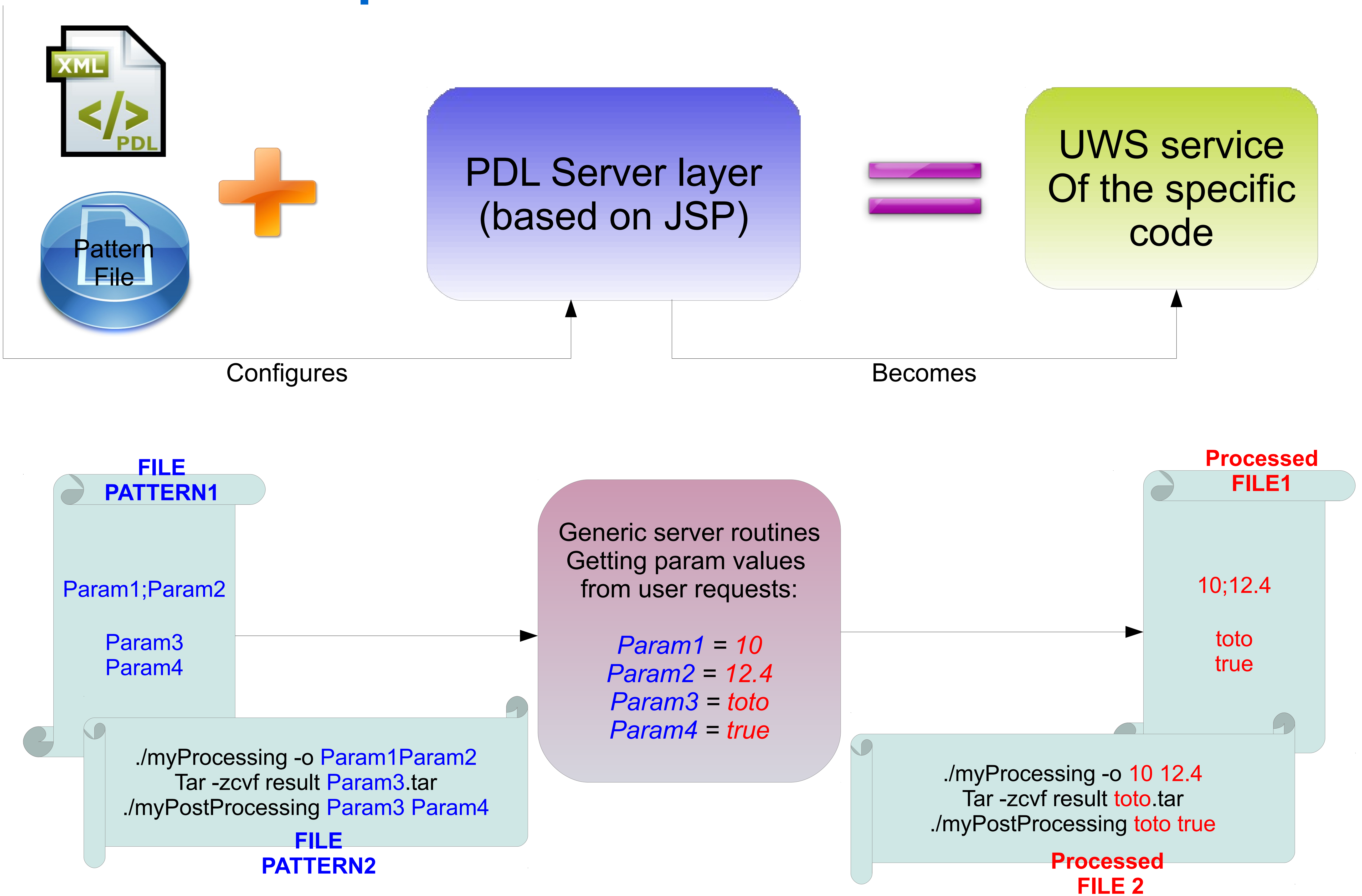
- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ and p_3 is boolean.
- if $p_1 > 0 \implies p_2 \in \{2;4;6\}$ and p_3 must be false.
- if $p_1 < 0 \implies p_3$ must be true.



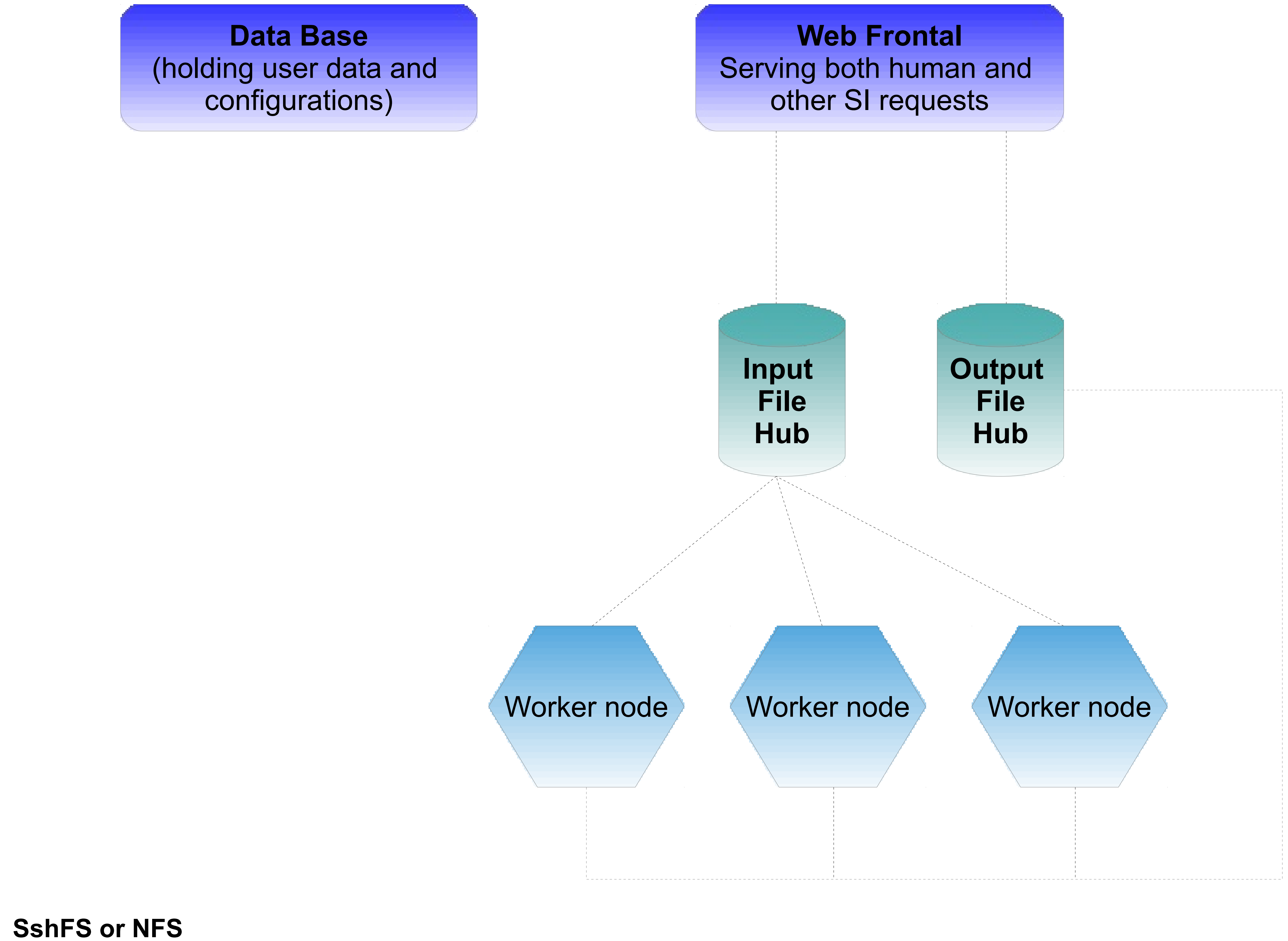
The PDL Server : deploy a UWS compliant service in few clickes



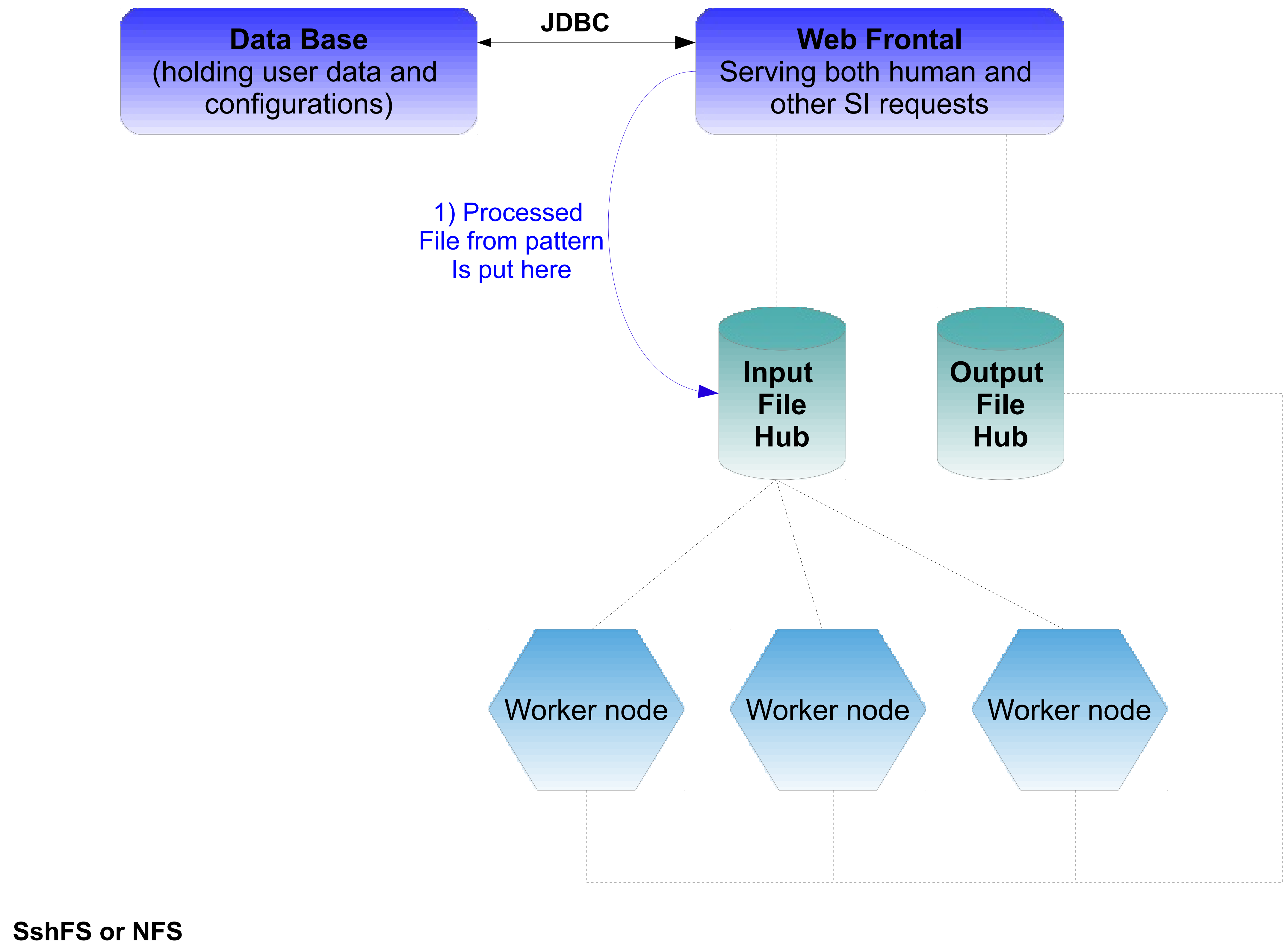
The PDL Server : deploy a UWS compliant service in few clickes



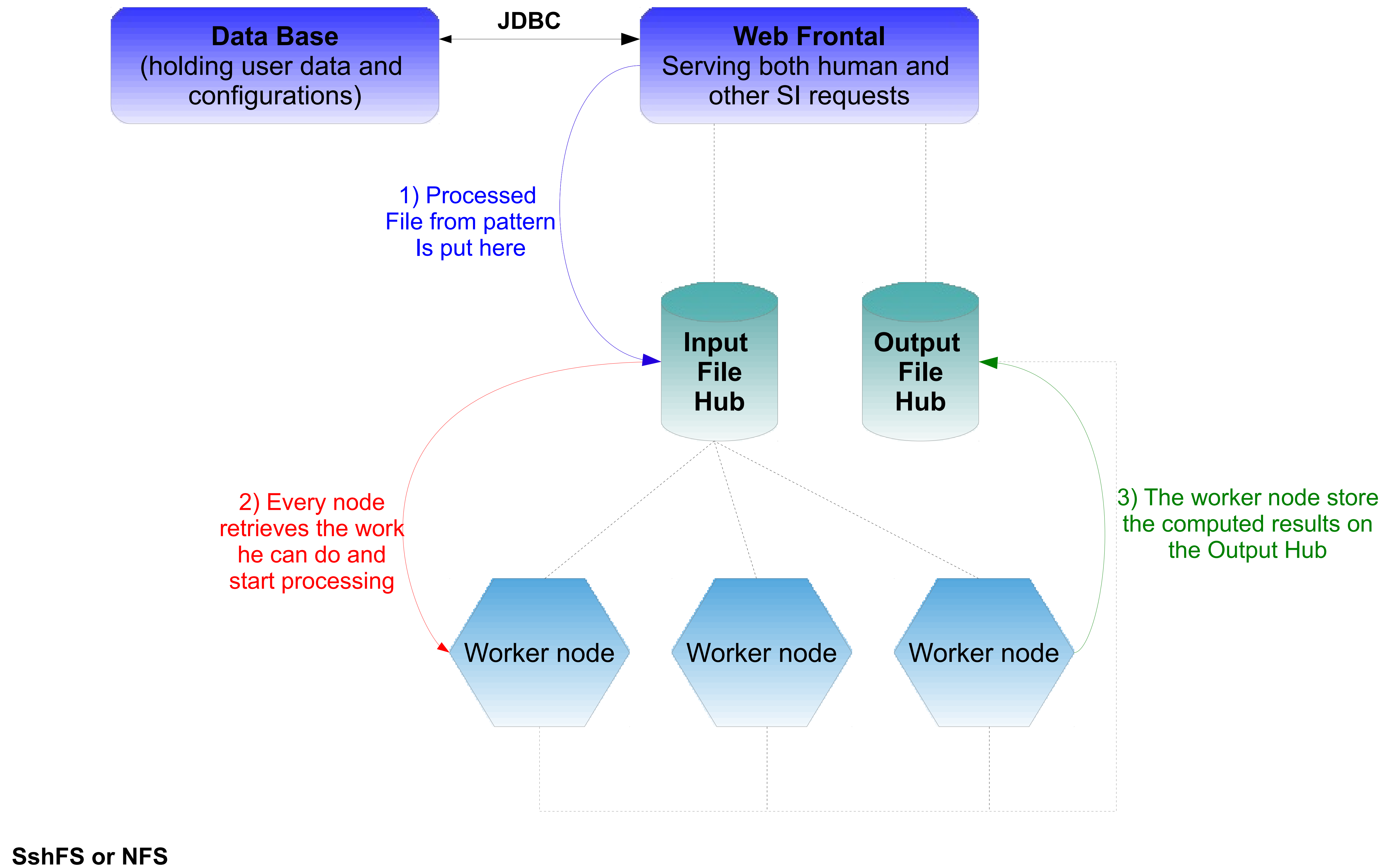
PDL Server: a distributed architecture



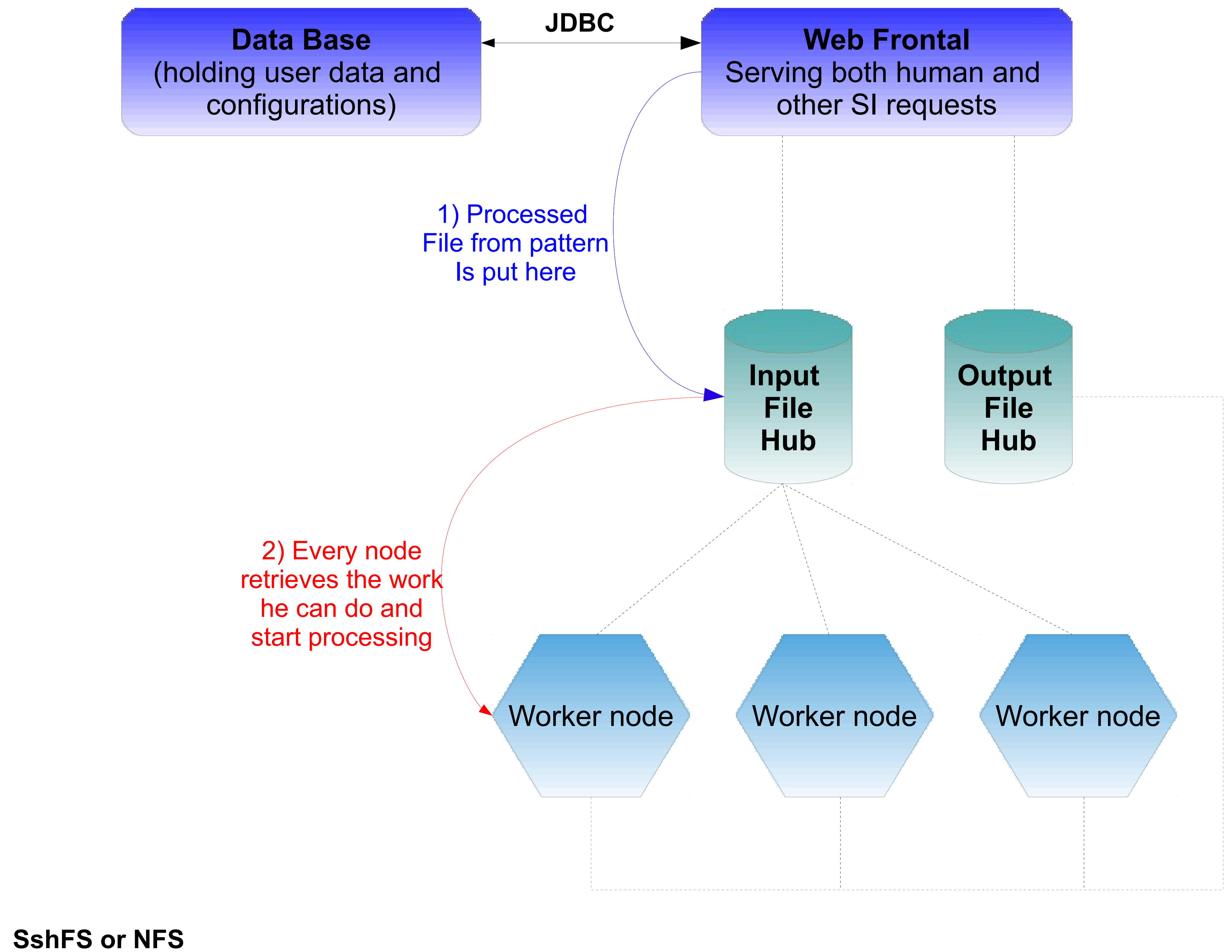
PDL Server: a distributed architecture



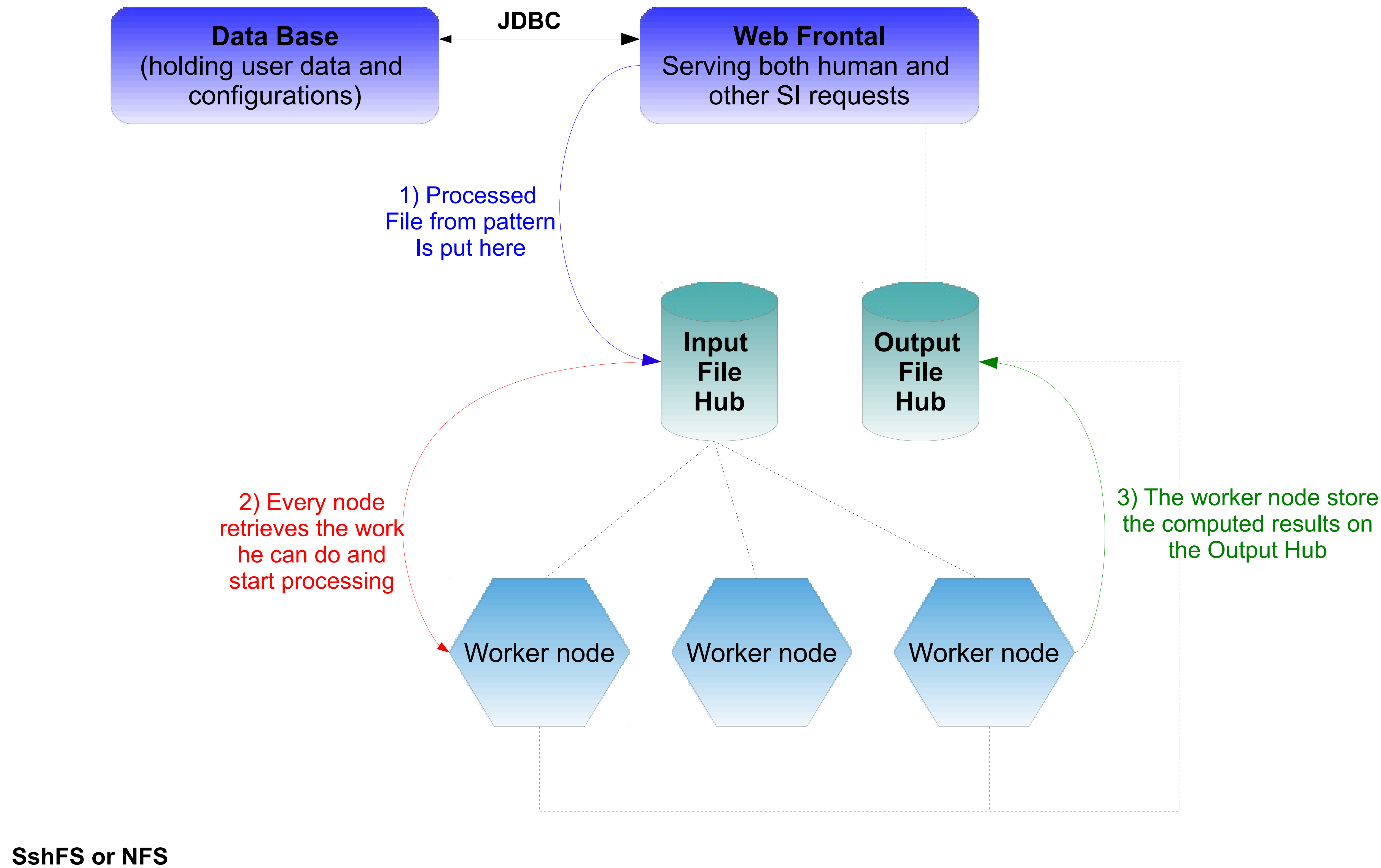
PDL Server: a distributed architecture



PDL Server: a distributed architecture



PDL Server: a distributed architecture



PDL Server: a distributed architecture