Observatoire de PARIS
OV France Workflow Day
PDL and its framework:
Concepts, Client, Server,...

Carlo Maria Zwölf
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 $\rho$ in $cm^{-3}$

**Constraints**

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

Constraints

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

- 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 \textit{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

<table>
<thead>
<tr>
<th>Description needs</th>
<th>Interoperability needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe physical properties of parameters</td>
<td>Interaction of two services has sense if the parameter sent by the first and expected by the second have same</td>
</tr>
<tr>
<td>Nature</td>
<td>Meaning</td>
</tr>
<tr>
<td>Physical constraints</td>
<td>Arbitrary Conditions</td>
</tr>
</tbody>
</table>
Motivations

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

<table>
<thead>
<tr>
<th>Description needs</th>
<th>Interoperability needs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe physical properties of parameters</td>
<td>Interaction of two services has sense if the parameter sent by the first and expected by the second have same</td>
</tr>
<tr>
<td>Nature</td>
<td>Meaning</td>
</tr>
<tr>
<td>Describe complex relations involving parameters</td>
<td>Interaction of two services has sense if all preconditions of second service are satisfied by output of first one</td>
</tr>
<tr>
<td>Physical constraints</td>
<td>Arbitrary Conditions</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Nature</th>
<th>Meaning</th>
<th>Unit</th>
<th>Precision</th>
<th>Range</th>
</tr>
</thead>
</table>

- Describe physical properties of parameters

<table>
<thead>
<tr>
<th>Physical constraints</th>
<th>Arbitrary Conditions</th>
<th>Mathematical Conditions</th>
</tr>
</thead>
</table>

- Describe complex relations involving parameters

### 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 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:
- $p_1$ is a m/s vector speed and $\|p_1\| < c$
- $p_2$ 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

\[ p_2 \in \{2; 4; 6\}, p_3 \in [-1, +1] \text{ and } (|\sin(p_1)p_2 - p_3|)^{1/2} < 3/2. \]

if \( p_1 \in ]\pi/2, \pi] \) then

\[ 0 < p_2 < 10, p_3 > \log(p_2) \text{ and } (p_1 \cdot p_2) \text{ must belong to } \mathbb{N}. \]

Output:

\[ p_4, p_5 \in \mathbb{R}^3 \]

Always \[ \frac{\|p_5\|}{\|p_4\|} \leq 0.01. \]
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.

Two double values required:
- double Temp
- double Dens

Temp should be temperature? Which unit?
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.

Sending: Temp = -4; Dens = -10
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.

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 \([10 ; 10^4]\).

OK!
Everything is clear

Sending: Temp = -4 ; Dens = -10

PDL checker
PDL and interoperability

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

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

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

Service 1:
Inputs a, b reals
Outputs c real and
\[ c = -\text{abs}(a-b) \]
Always \( c < 0 \)

Service 2:
Inputs a, b reals
Outputs c real and
\[ c = +\text{abs}(a-b) \]
Always \( c > 0 \)

Service 3:
Inputs c reals
Always \( c > 0 \)
Outputs d real and
\[ d = \sqrt{c} \]
Always \( d > 0 \)
PDL and interoperability

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

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

Service 3:
Inputs $c$ real
Always $c > 0$
Outputs $d$ real and $d = \sqrt{c}$
Always $d > 0$
PDL and interoperability

Let

- \( S_1 \) and \( S_2 \) be two services.
- \( p^j(S_i) \) be the \( j \)th parameter of \( S_i \).
- \( \mathcal{I}(S_i) \) (resp. \( \mathcal{O}(S_i) \)) be the set of input (resp. output) parameters of \( S_i \).
- \( C^p_{\mathcal{I}(S_i)} \) (resp. \( C^p_{\mathcal{O}(S_i)} \)) 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 \( C^p_{\mathcal{O}(S_1)} \) \( \implies \) \( p^k(S_2) \) satisfies \( C^p_{\mathcal{I}(S_2)} \)

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 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
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

Diagram:
- Dynamic ‘intelligent’ client calls PDL Server (exposing every code as a UWS service)
- PDL Server uses PDL CORE (the grammar)
- PDL CORE uses Interoperability Checker
- Taverna Plugin interacts with the system
The Dynamic client

XML

PDL

Generic client code base

Specific Client

Configures

Becomes
The Dynamic client

Generic client code base

Specific Client

Configures

Becomes

Service description:

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

Configures

Generic client code base

Becomes

Specific Client

Service description:

- \( p_1 \in \mathbb{R}, p_2 \in \mathbb{N} \) and \( p_3 \) is boolean.
- if \( p_1 > 0 \) \( \Rightarrow \) \( p_2 \in \{2; 4; 6\} \) and \( p_3 \) must be false.
- if \( p_1 < 0 \) \( \Rightarrow \) \( p_3 \) must be true.
The PDL Server: deploy a UWS compliant service in few clicks

PDL Server layer (based on JSP) → UWS service

Configures → Becomes

- Pattern File
- XML

- [XML]
  - <PDL>
  - XML
  - Pattern File
The PDL Server: deploy a UWS compliant service in few clicks

PDL Server layer (based on JSP)

Configures

Generic server routines
Getting param values from user requests:

Param1 = 10
Param2 = 12.4
Param3 = toto
Param4 = true

Pattern File

Becomes

UWS service
Of the specific code

FILE PATTERN1

Param1;Param2
Param3
Param4

./myProcessing -o Param1Param2
Tar -zcvf result Param3.tar
./myPostProcessing Param3 Param4

Processed FILE1

10;12.4
toto
toto
true

FILE PATTERN2

Processed FILE 2
PDL Server: a distributed architecture

- **Data Base** (holding user data and configurations)
- **Web Frontal** Serving both human and other SI requests
- **Input File Hub**
- **Output File Hub**
- **Worker node**
  - Connected to SshFS or NFS

SshFS or NFS
PDL Server: a distributed architecture

Data Base (holding user data and configurations)

Web Frontal
Serving both human and other SI requests

JDBC

Input File Hub

Output File Hub

1) Processed File from pattern
Is put here

Worker node

Worker node

Worker node

SshFS or NFS
PDL Server: a distributed architecture

1) Processed File from pattern Is put here

2) Every node retrieves the work he can do and start processing

3) The worker node store the computed results on the Output Hub
PDL Server: a distributed architecture

1) Processed File from pattern Is put here

Data Base (holding user data and configurations)

Web Frontal Serving both human and other SI requests

JDBC

1) Processed File from pattern Is put here

Input File Hub

Output File Hub

Worker node

2) Every node retrieves the work he can do and start processing

SshFS or NFS
PDL Server: a distributed architecture

1) Processed File from pattern
   Is put here

2) Every node retrieves the work he can do and start processing

3) The worker node store the computed results on the Output Hub
PDL Server: a distributed architecture

1) Processed File from pattern
   Is put here

2) Every node retrieves the work he can do and start processing

3) The worker node store the computed results on the Output Hub

4) The frontal check if new results are available and notifies it to the user who asked the job

Web Frontal
Serving both human and other SI requests

Data Base
(holding user data and configurations)

Input File Hub

Output File Hub

Worker node

Worker node

Worker node

JDBC

SshFS or NFS