Puna InterOp 20/10/2011

A new working grammar for describing parameters (and their constraints) within the Virtual Observatory

Carlo Maria Zwölf, Franck Le Petit, Paul Harrison.

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

Laboratoire Univers et Théories

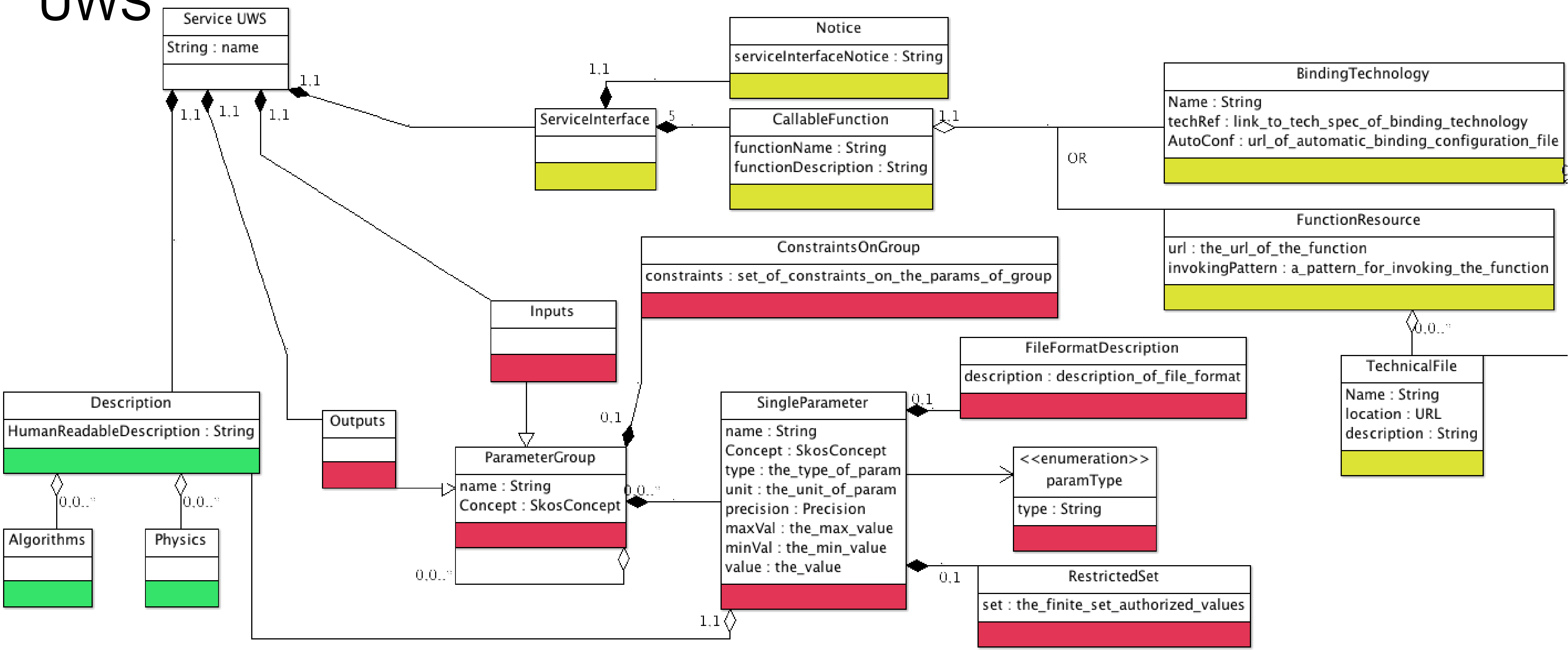
The University of Manchester

Recall the context

- In Naples we presented some ideas fur further developments in the UWS

Recall the context

- In Naples we presented some ideas fur further developments in the UWS



Grouped by:

What the service do (and how)

How to interact with the service (medium)

Explain the 'language' used for interacting with service

Which service should be chosen to meet given needs...

How to reach the service functions...

How to build the arguments for calling service functions and understanding the output results

Recall the context

- In Naples we presented some ideas fur further developments in the UWS
- We have considered that starting work on the *Parameter description layer* is more profitable (and useful through the VO, beyond the UWS scope)

Recall the context

- In Naples we presented some ideas fur further developments in the UWS
- We have considered that starting work on the *Parameter description layer* is more profitable (and useful through the VO, beyond the UWS scope)
- This layer
  - Could work as an additional (and optional) component of UWS
  - Makes interoperability straightforward by checking automatically if two (or more) services could be 'piped' into a given workflow

Recall the context

- In Naples we presented some ideas fur further developments in the UWS
- We have considered that starting work on the *Parameter description layer* is more profitable (and useful through the VO, beyond the UWS scope)

- This layer
  - Could work as an additional (and optional) component of UWS
  - Makes interoperability straightforward by checking automatically if two (or more) services could be 'piped' into a given workflow

- Our needs comes from the requirements of Theory Group: they would like to deploy online codes with complex sets of in(out)put data .

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

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 $p_4 \geq 0$

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

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 $\frac{\|\boldsymbol{p}_5\|}{\|\boldsymbol{p}_4\|} \leq 0.01$.

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

Existing products (ex. Apache Wadl) do not have this fine required descriptive capabilities.

Building our solution

Our goal is

- Finely describe the set of parameters (**inputs** & **outputs**) of a given service in a way that
  - Could be understood easily by humans
  - Could be interpreted and handled by a computer
- Describe complex relations and constraints on and between parameters

Existing products (ex. Apache Wadl) do not have this fine required descriptive capabilities.

Our ideas lead to a working solution fitting our requirements.

The working solution

- The grammar and syntax for building parameters description are fixed in a XML schema:

  http://code.google.com/p/vo-param/source/browse/trunk/model/src/schema/UWS2-V1.1.xsd

The working solution

- The grammar and syntax for building parameters description are fixed in a XML schema:

  http://code.google.com/p/vo-param/source/browse/trunk/model/src/schema/UWS2-V1.1.xsd

- In practice:
  - Service providers (unless they want) don't need to handle directly the XSD file

    ↓

    A GUI will be provided for composing the service description in few clicks

    (This development is a part of a course that will take place during the first half 2012)

The working solution

- The grammar and syntax for building parameters description are fixed in a XML schema:

  http://code.google.com/p/vo-param/source/browse/trunk/model/src/schema/UWS2-V1.1.xsd

- In practice:
  - Service providers (unless they want) don't need to handle directly the XSD file

    ↓

    A GUI will be provided for composing the service description in few clicks

    (This development is a part of a course that will take place during the first half 2012)

  - With our protocol, one could easy express
    - All the possible mathematical expressions involving parameters
    - All the possible conditional sentences (provided they have a logical sense)

The working solution

- The grammar and syntax for building parameters description are fixed in a XML schema:

  http://code.google.com/p/vo-param/source/browse/trunk/model/src/schema/UWS2-V1.1.xsd

- In practice:
  - Service providers (unless they want) don't need to handle directly the XSD file

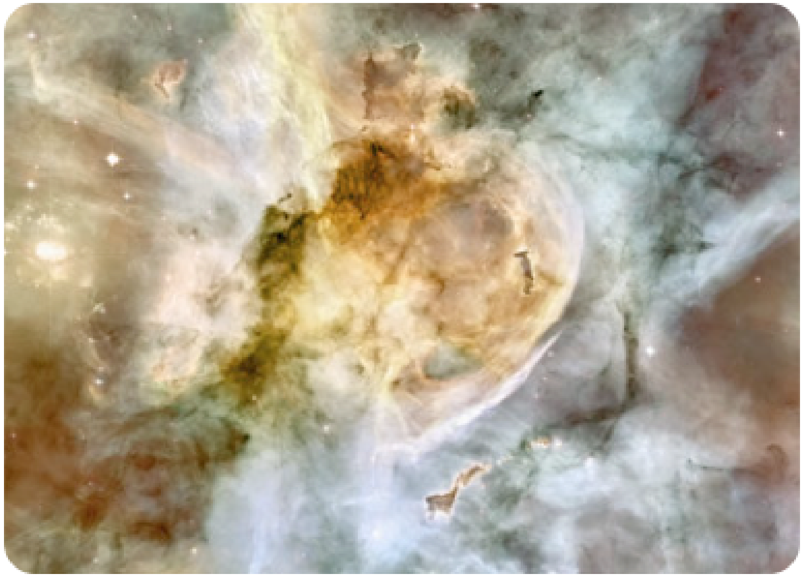
    ↓

    A GUI will be provided for composing the service description in few clicks

    (This development is a part of a course that will take place during the first half 2012)

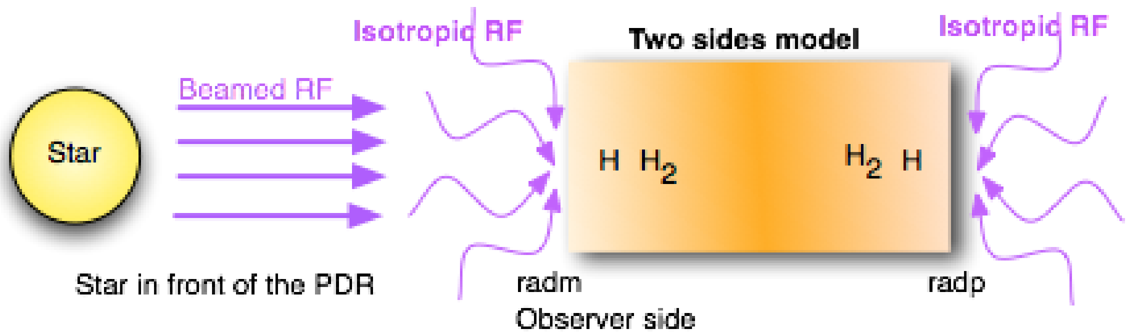
  - With our protocol, one could easy express
    - All the possible mathematical expressions involving parameters
    - All the possible conditional sentences (provided they have a logical sense)
  - All the following examples are automatically generated from Java code using the JaxB Api.

A working example : the PDR code



- Code modeling the micro-physics of interstellar clouds (used to interpret HERSCHEL observations)

- Already Implemented in Astrogrid (CEA) in 2007.



- Incident radiation field
- observer and back side
- ISRF intensity
- Type of stellar spectrum
- distance of the star
- ...

- State equation
- isochore (density)
- isobare (pressure)
- specific user density profile
- ...

- Grains properties
- R min and max
- Extinction properties
- ...

Non trivial relationships between parameters

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<UWS_Service xmlns="http://www.ivoa.net/xml/Parameter/v0.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.ivoa.net/xml/Parameter/v0.1 UWS2-V1.1.xsd">
      <ServiceId>PDR_ONLINE</ServiceId>
      <serviceName>PDR-1D</serviceName>
      <Description>Description layer of the PDR code</Description>
      <ParameterList>
            <parameter>
                  <Name>F_ISRF</Name>
                  <ParameterType>integer</ParameterType>
                  <Unit>None</Unit>
                  <Precision>0</Precision>
            </parameter>
            <parameter>
                  <Name>radm</Name>
                  <ParameterType>real</ParameterType>
            </parameter>
            <parameter>
                  <Name>radp</Name>
                  <ParameterType>real</ParameterType>
            </parameter>
            <parameter>
                  <Name>d_sour</Name>
                  <ParameterType>real</ParameterType>
            </parameter>
            <parameter>
                  <Name>srcpp</Name>
                  <ParameterType>string</ParameterType>
            </parameter>
            <parameter>
                  <Name>srcpp_spectrum</Name>
                  <ParameterType>Spectrum</ParameterType>
            </parameter>
            ...
```

Parameter list

```xml
<ParameterGroup>
    <Name>RadiationFieldAndGeometry</Name>
    <parameterRef parameterName="F_ISRF" />
    <parameterRef parameterName="radm" />
    <parameterRef parameterName="radp" />
    <parameterRef parameterName="d_sour" />
    <parameterRef parameterName="srcpp" />
    <parameterRef parameterName="srcpp_spectrum" />
</ParameterGroup>
```

Parameter groups

```xml
<conditionalStatement xsi:type="IfThenConditionalStatement">
    <if>
        <Criterion xsi:type="Criterion">
            <expression xsi:type="AtomicParameterExpression">
                <parameterRef parameterName="d_sour" />
            </expression>
            <conditionType xsi:type="ValueDifferentOf">
                <Value>0</Value>
            </conditionType>
            <logicalConnector xsi:type="and">
                <criterion xsi:type="Criterion">
                    <expression xsi:type="AtomicParameterExpression">
                        <parameterRef parameterName="srcpp" />
                    </expression>
                    <conditionType xsi:type="BelongToSet">
                        <Value>spectro1</Value>
                        <Value>spectro2</Value>
                        <Value>spectroN</Value>
                    </conditionType>
                </criterion>
            </logicalConnector>
        </Criterion>
    </if>
    <then>
        <Criterion xsi:type="Criterion">
            <expression xsi:type="AtomicParameterExpression">
                <parameterRef parameterName="srcpp_spectrum" />
            </expression>
            <conditionType xsi:type="IsNull" />
        </Criterion>
    </then>
</conditionalStatement>
```

Constraints

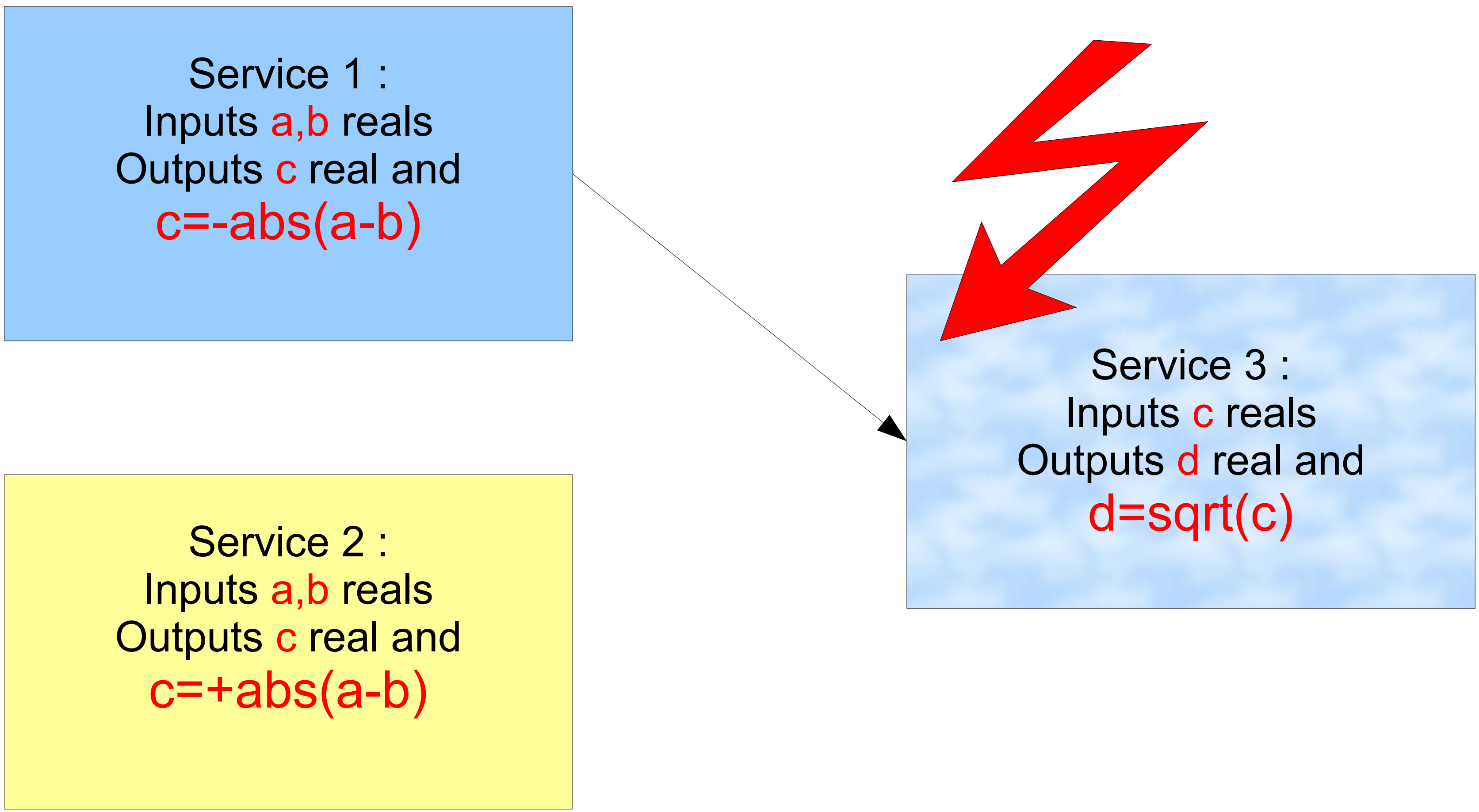
About the interoperability

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

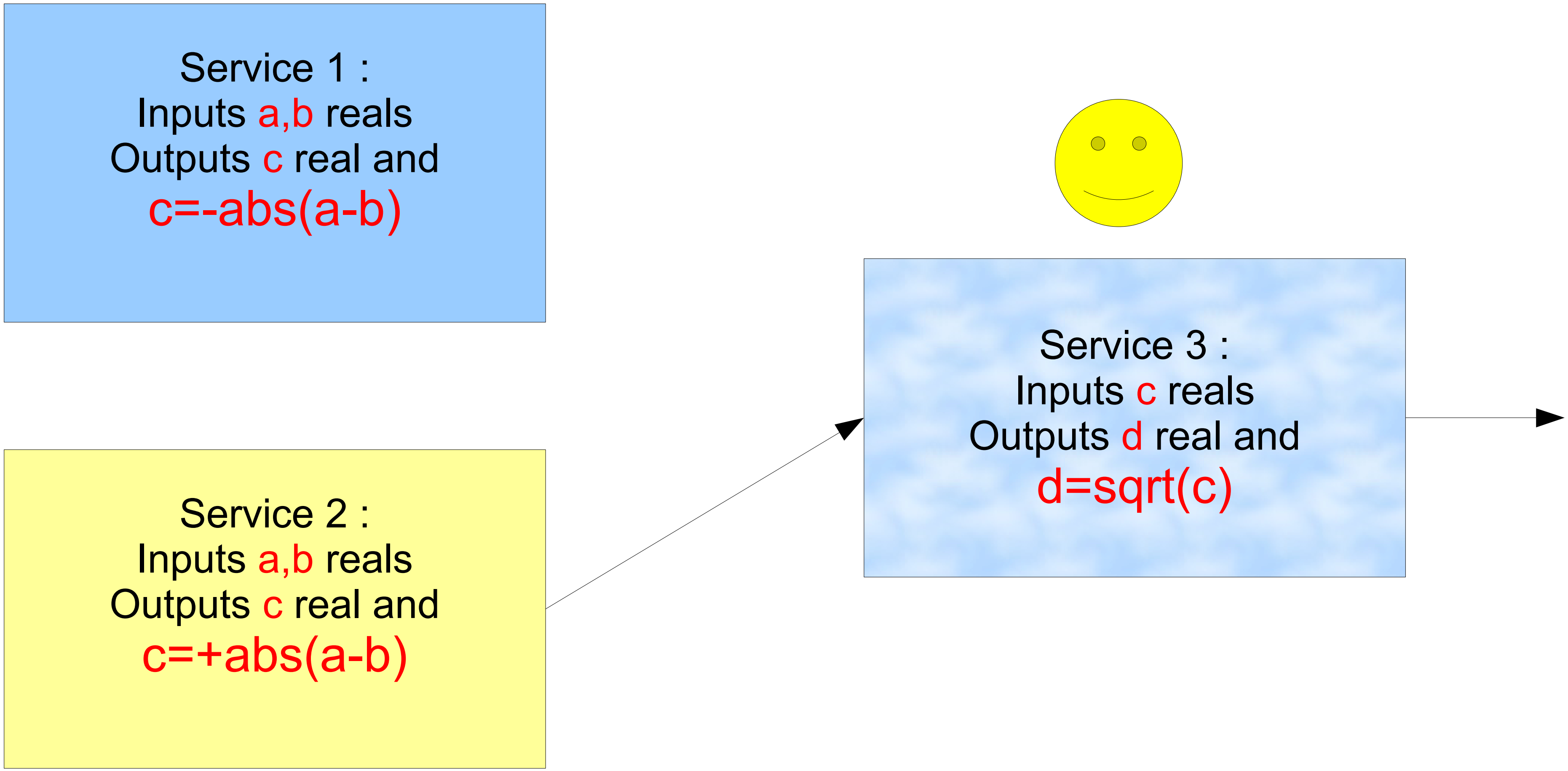
Service 2 :
Inputs a,b reals
Outputs c real and
c=+abs(a-b)

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

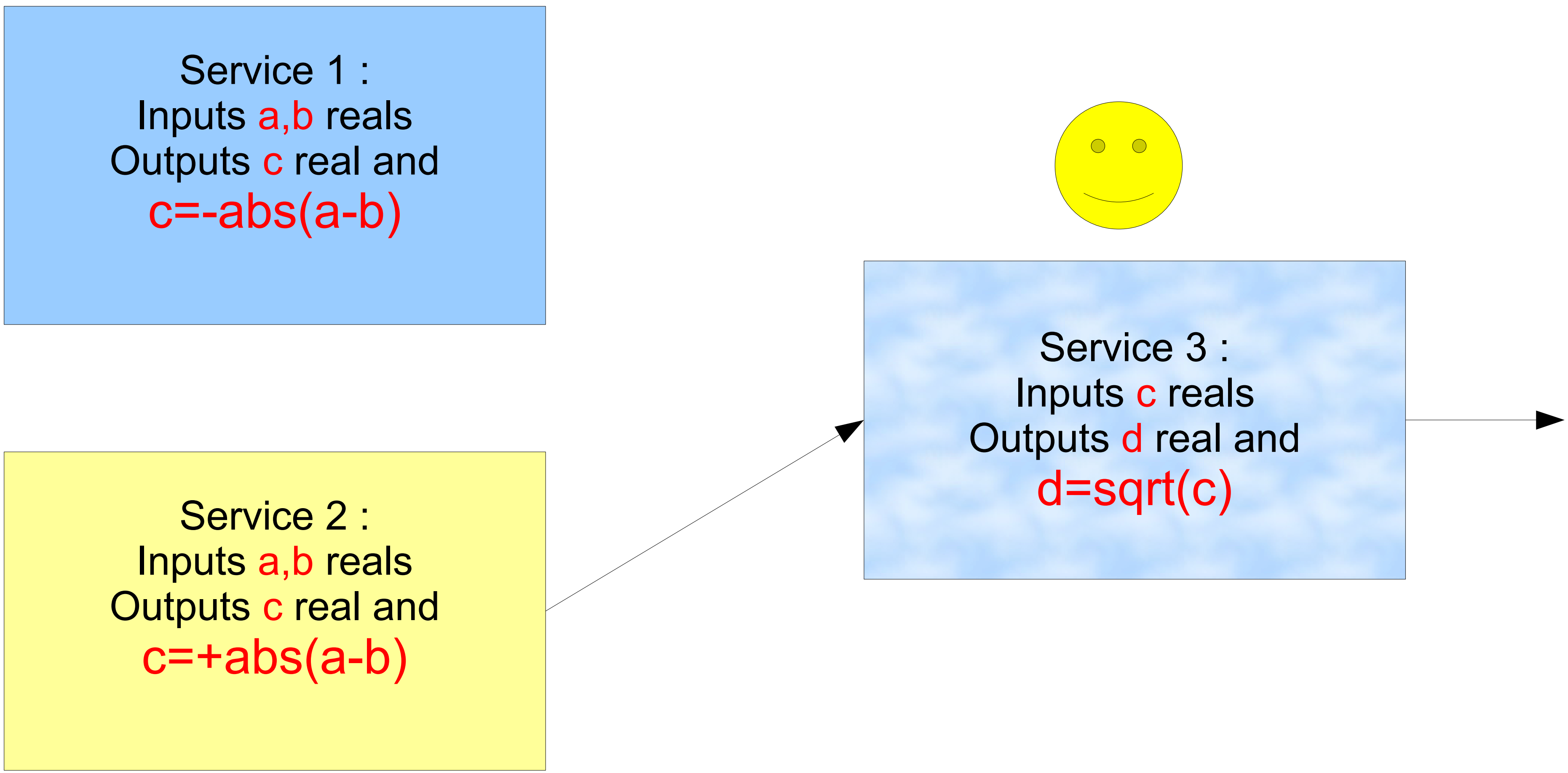
About the interoperability



About the interoperability



About the interoperability



Adding the constraints with our formalism...

About the interoperability

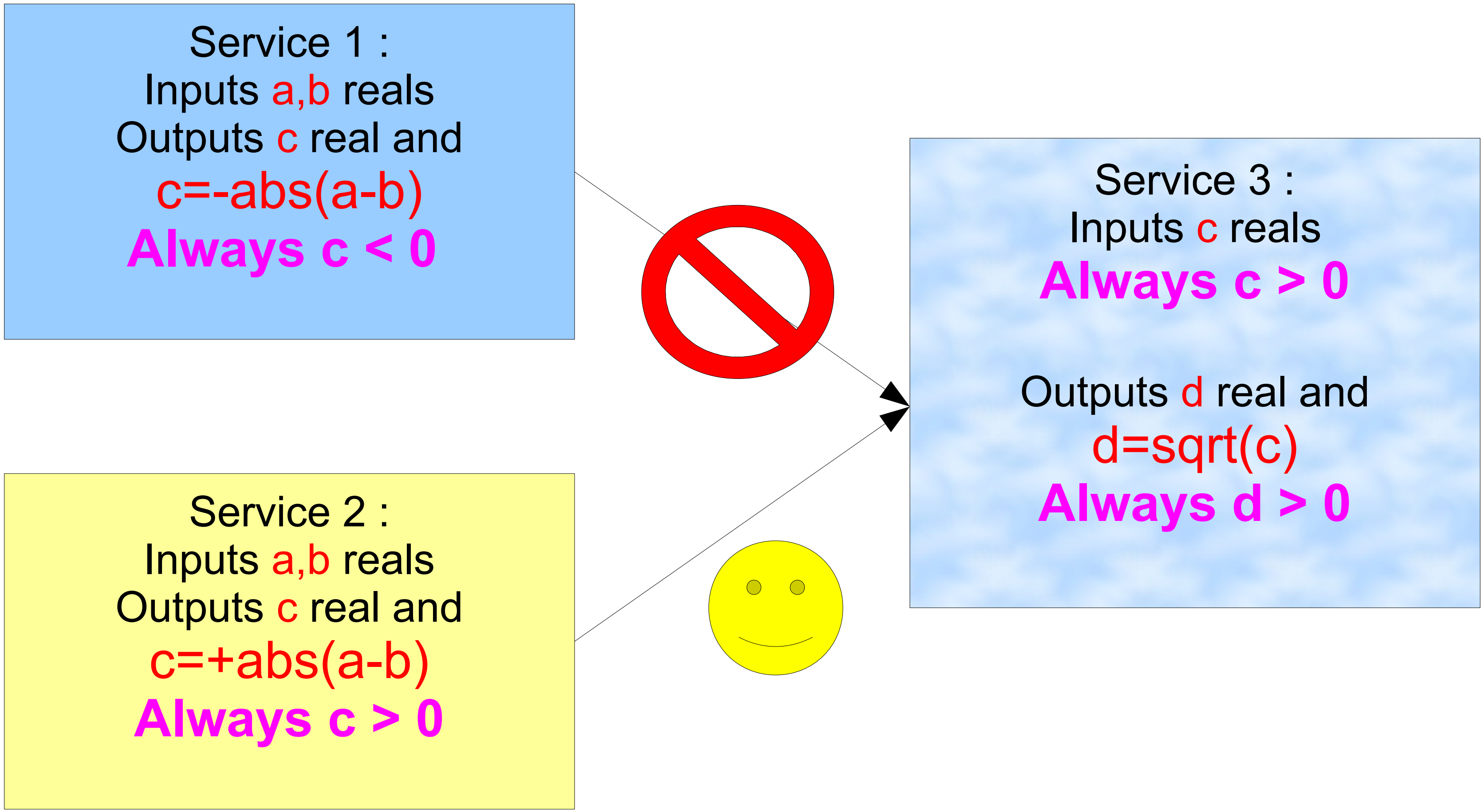
Service 1 :
Inputs a,b reals
Outputs c real and
c=-abs(a-b)
Always c < 0

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

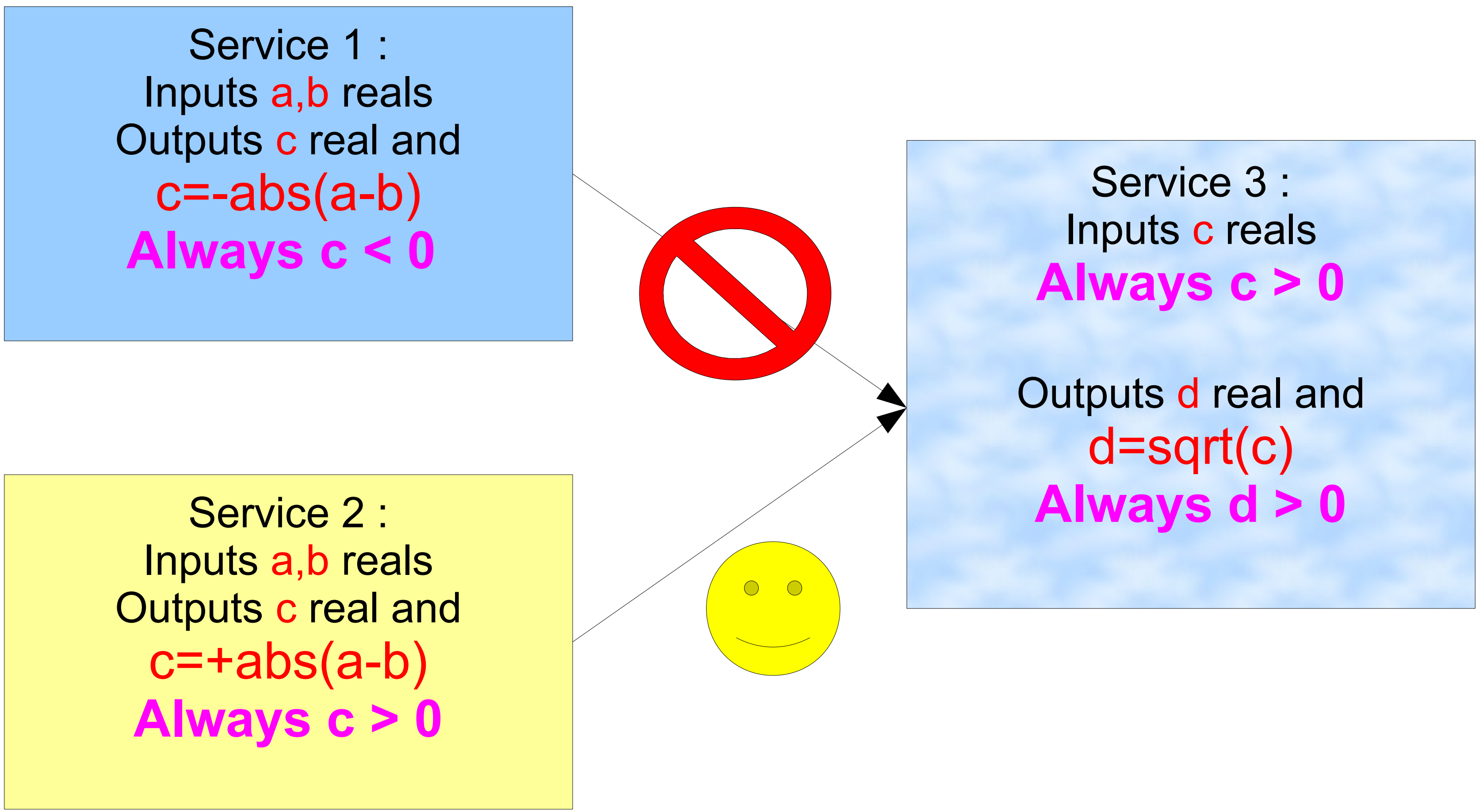
Service 3 :
Inputs c reals
Always c > 0

Outputs d real and
d=sqrt(c)
Always d > 0

About the interoperability



About the interoperability



More generally we can formalize this as following...

About the 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.

About the 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 $\mathcal{C}^{p^l}_{\mathcal{O}(S_1)} \Longrightarrow p^k(S_2)$ satisfies $\mathcal{C}^{p^k}_{\mathcal{I}(S_2)}$

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

About the 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.

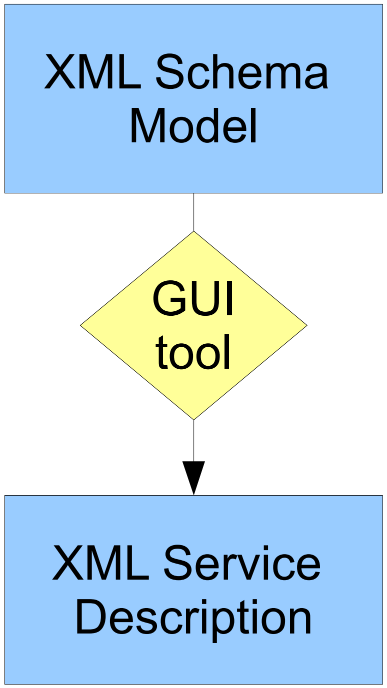
S_2 could follow S_1 into a workflow iff $\forall p^k(S_2) \in \mathcal{I}(S_2)\, \exists\, p^l(S_1) \in \mathcal{O}(S_1)$ such that:

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

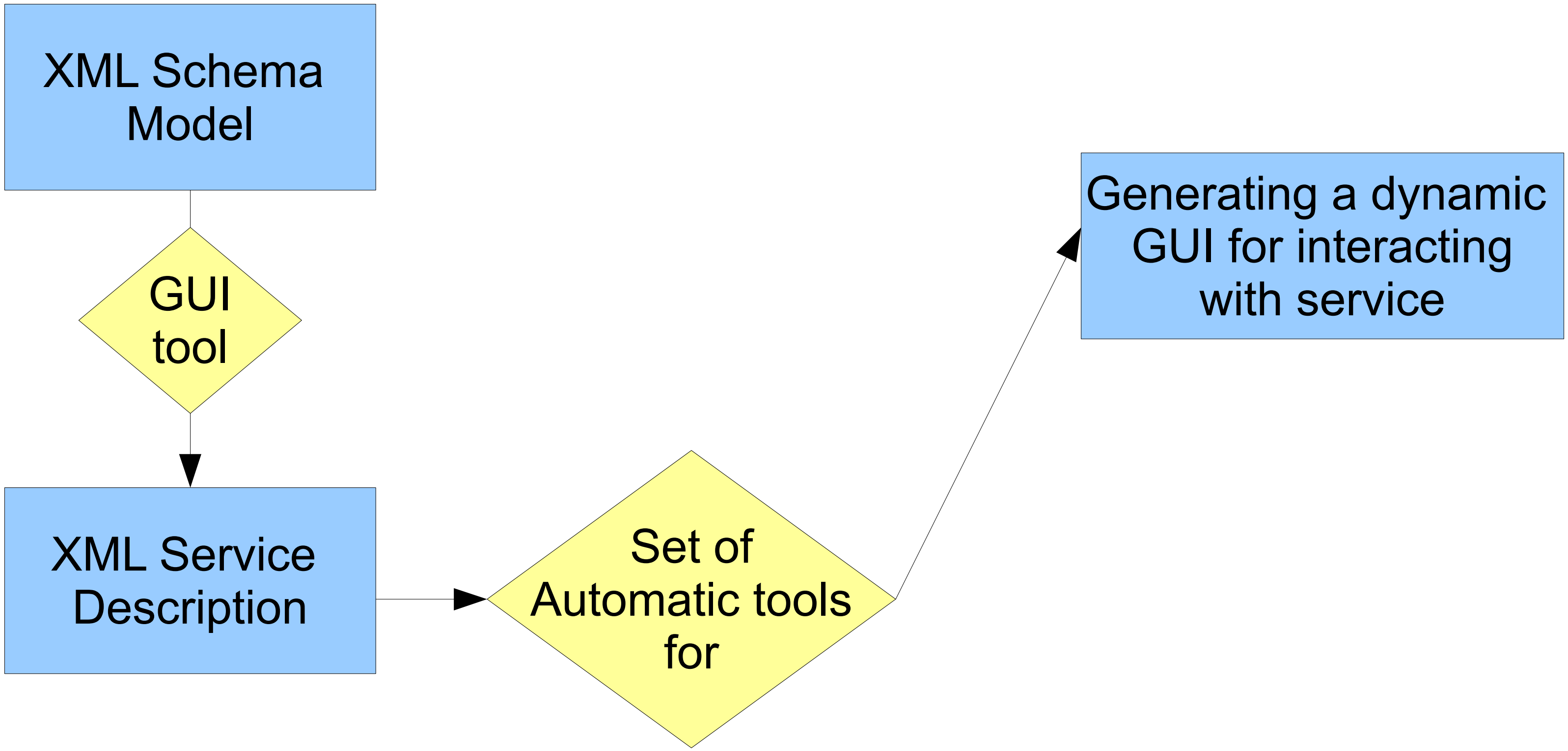
- The equality is in the sense that parameters have same
  - UCDs
  - UTypes
  - SkossConcepts
  - Units

- If the difference is on units, the services are still **compatibles**: we can build a third service performing the unit change, making interoperability possible.

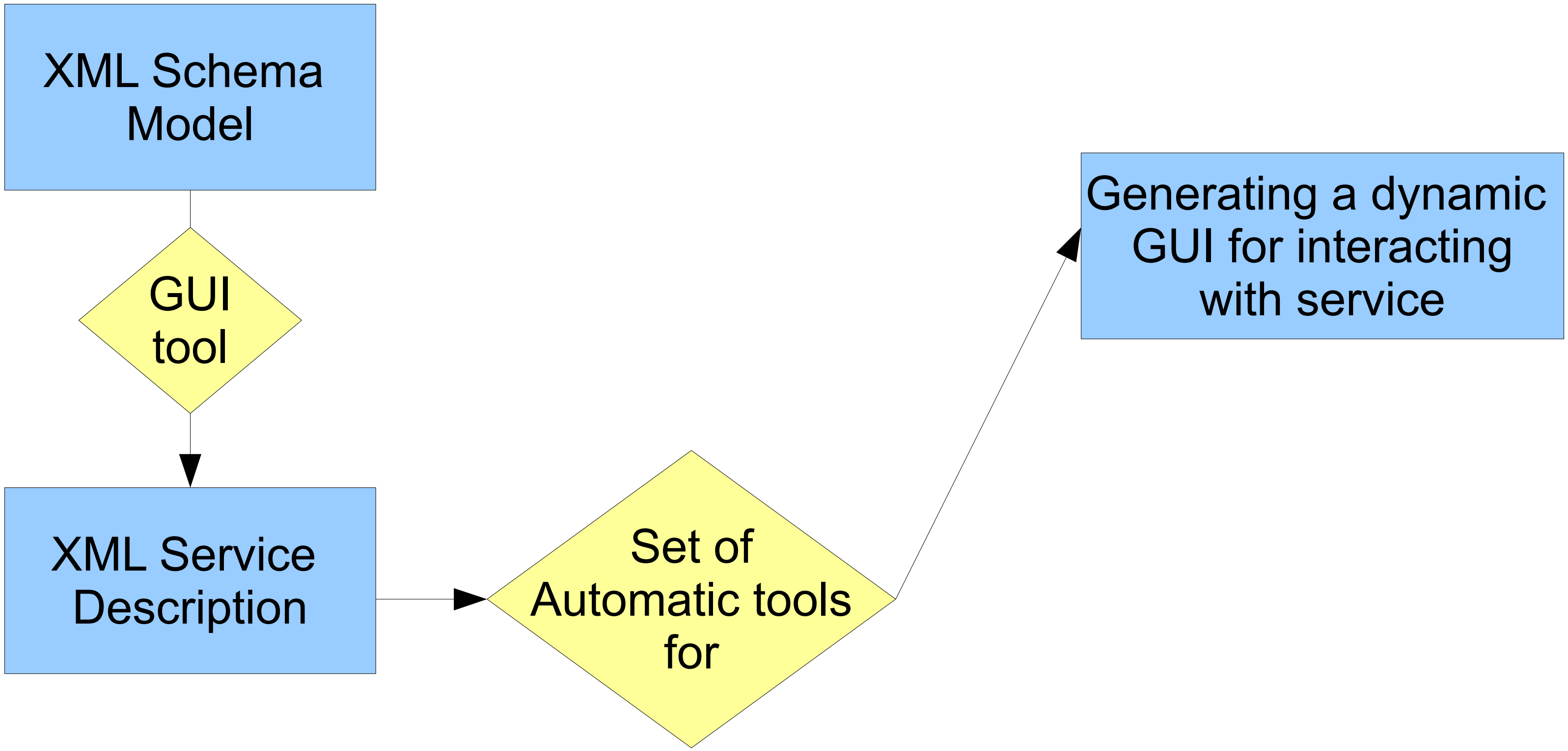
Sketch of the integral solution



Sketch of the integral solution



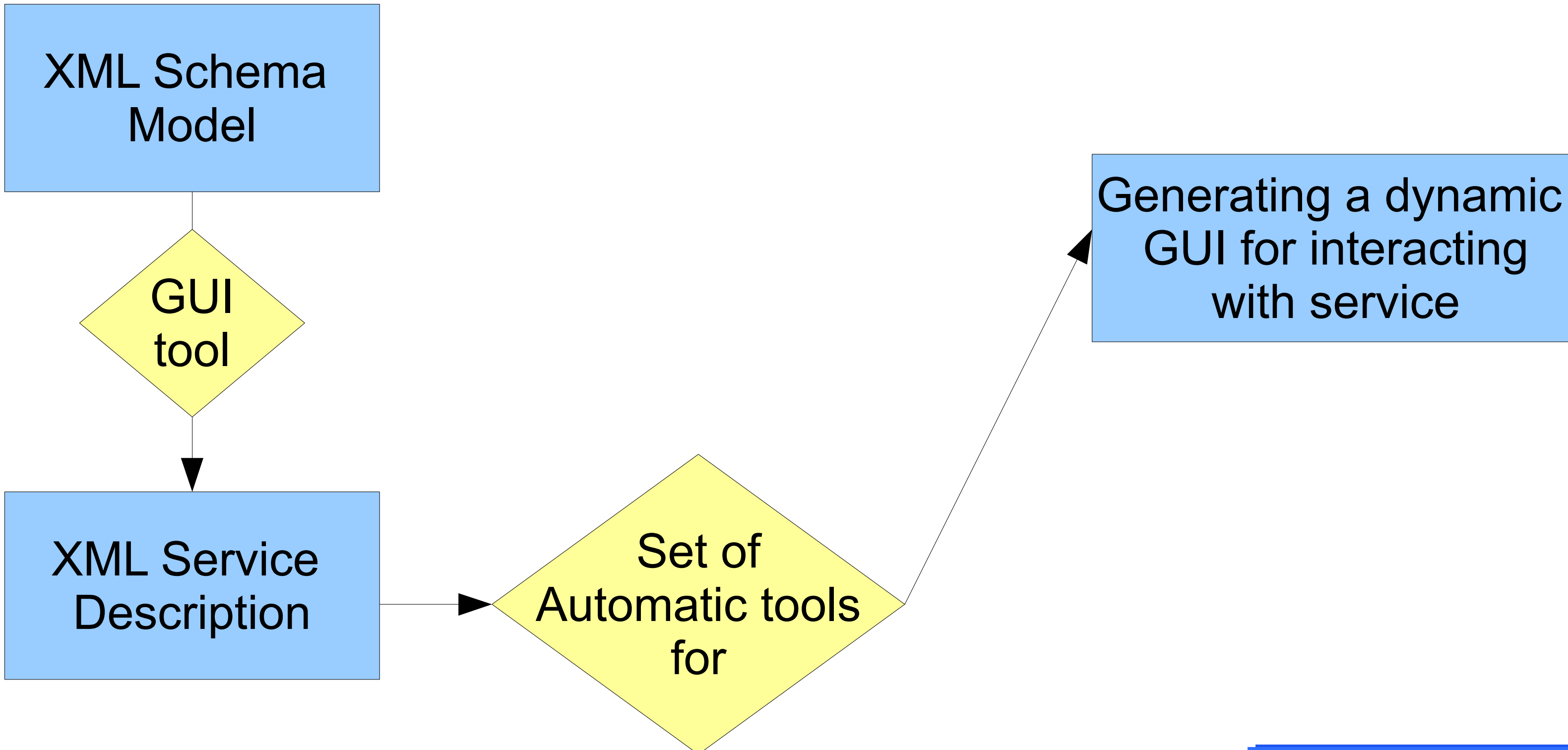
Sketch of the integral solution

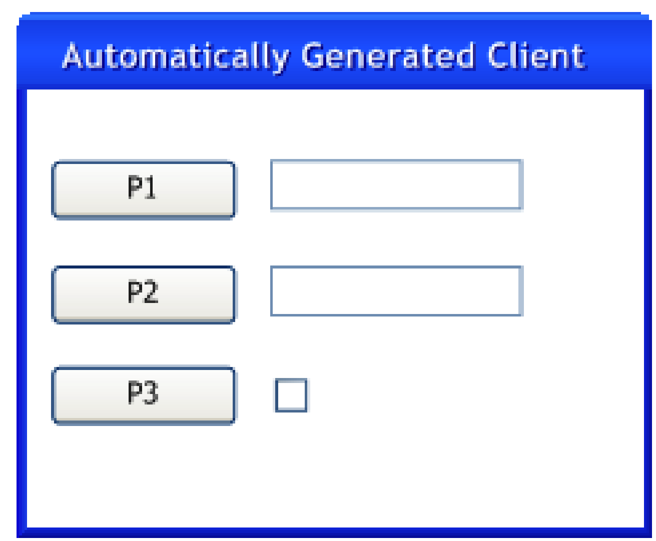


Service description:

- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ 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.

Sketch of the integral solution

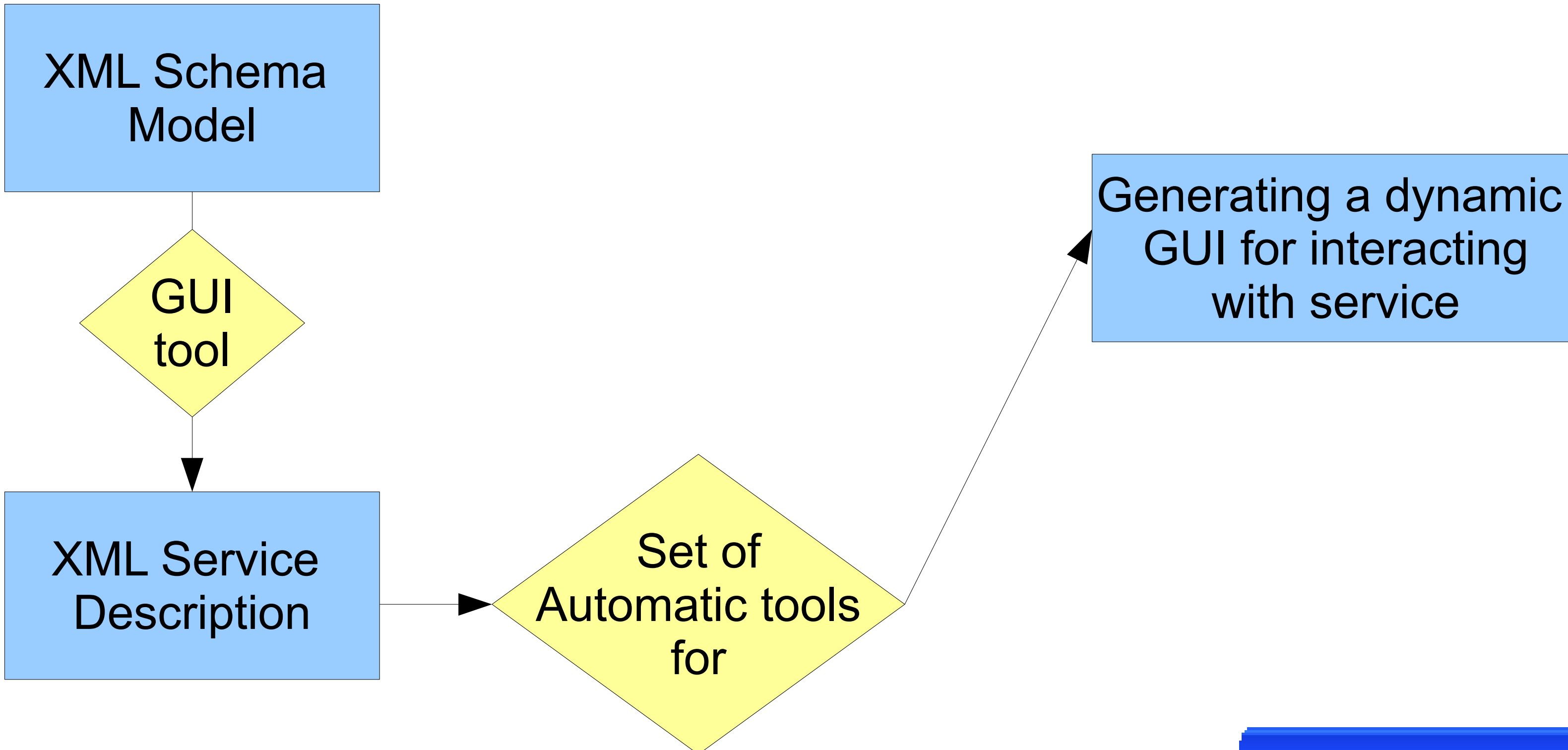




Service description:

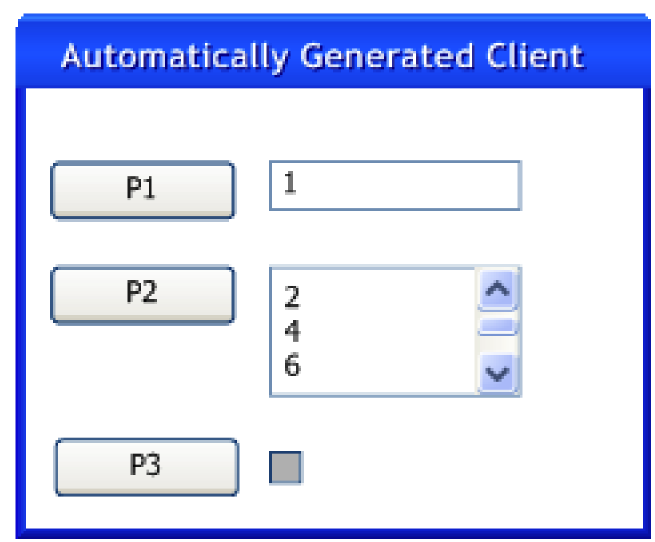
- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ and p_3 is boolean.
- if $p_1 > 0 \implies p_2 \in \{2; 4; 6\}$ and p_3 must be false.
- if $p_1 < 0 \implies p_3$ must be true.

Sketch of the integral solution

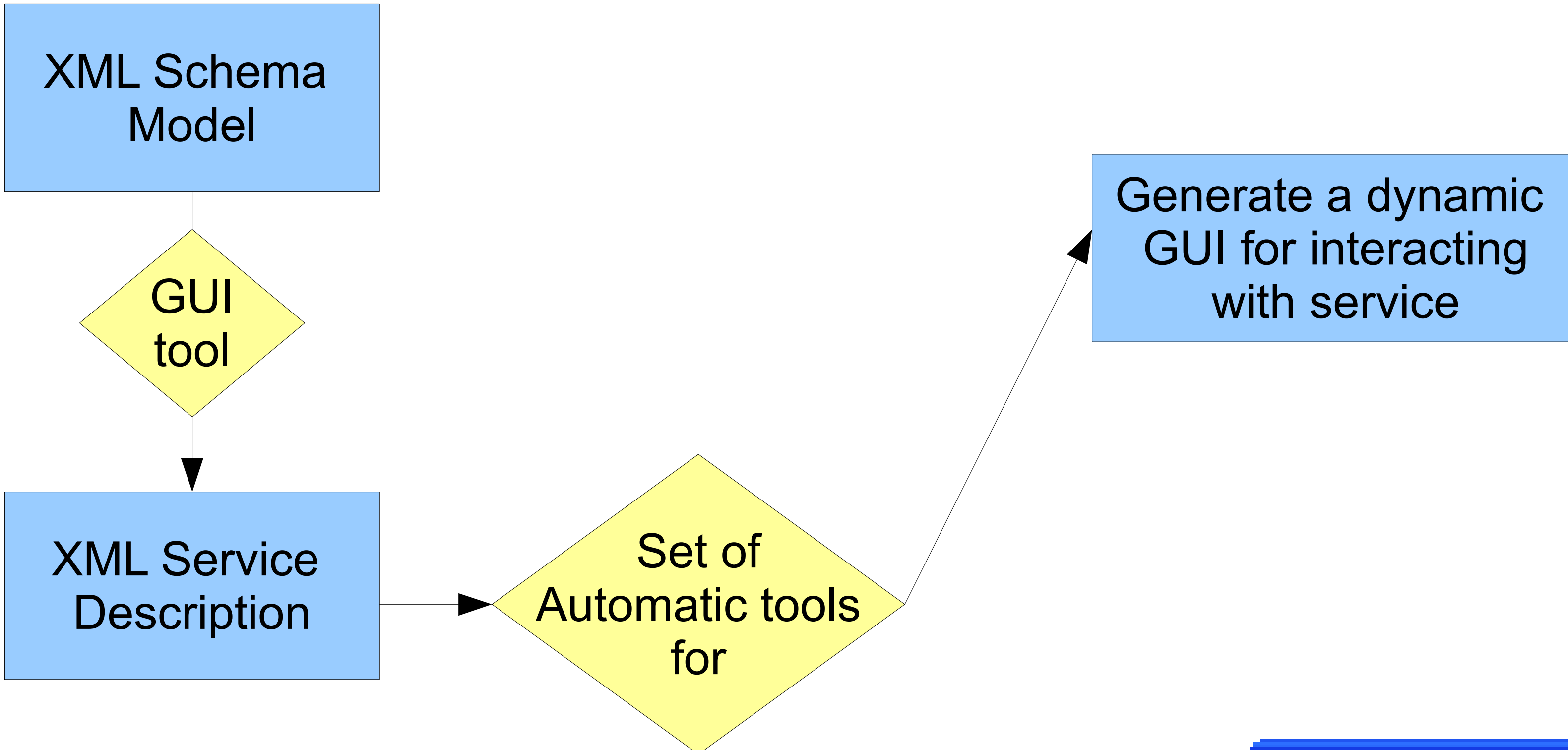


Service description:

- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ 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.

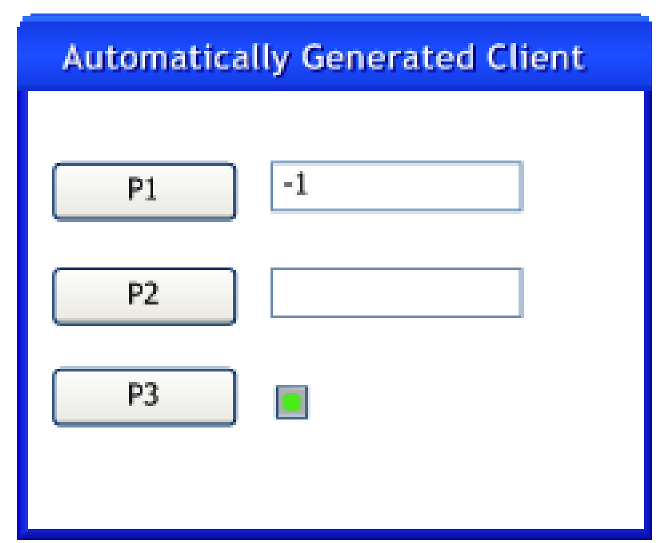


Sketch of the integral solution

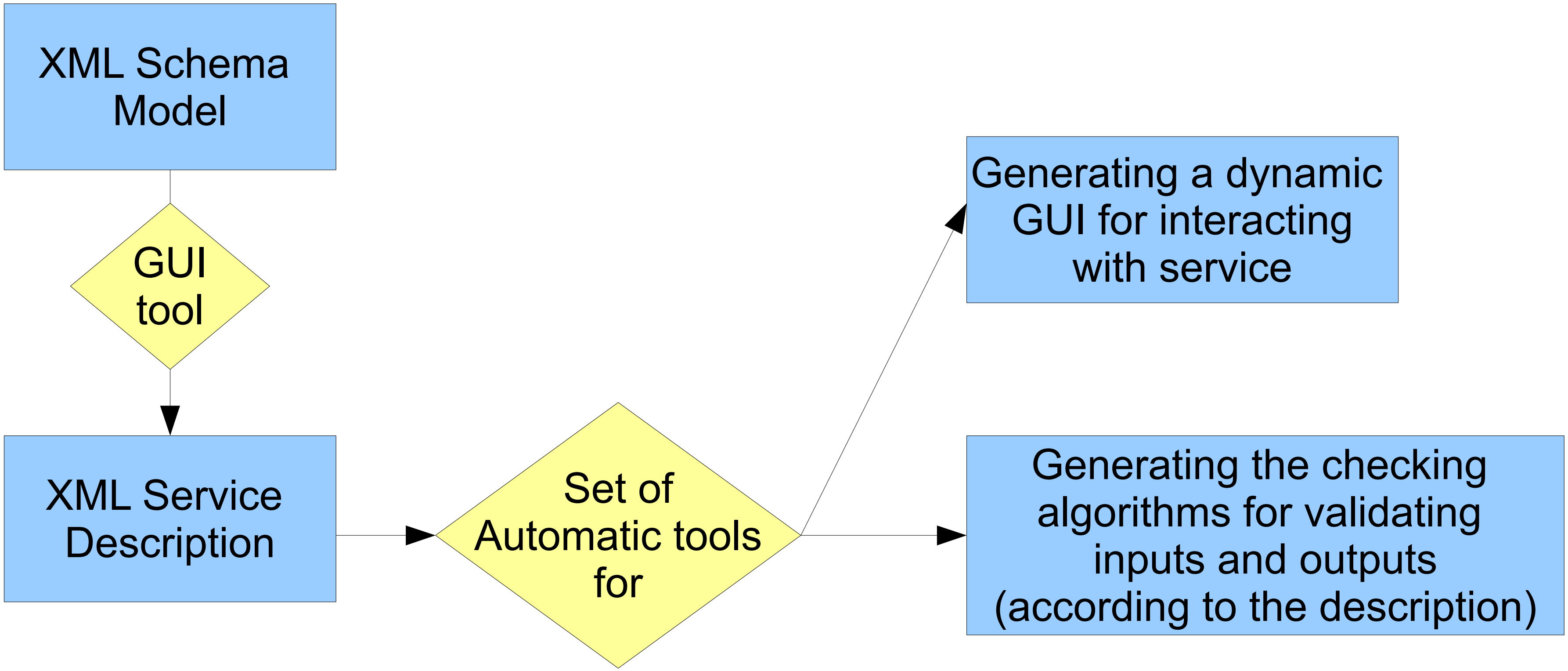


Service description:

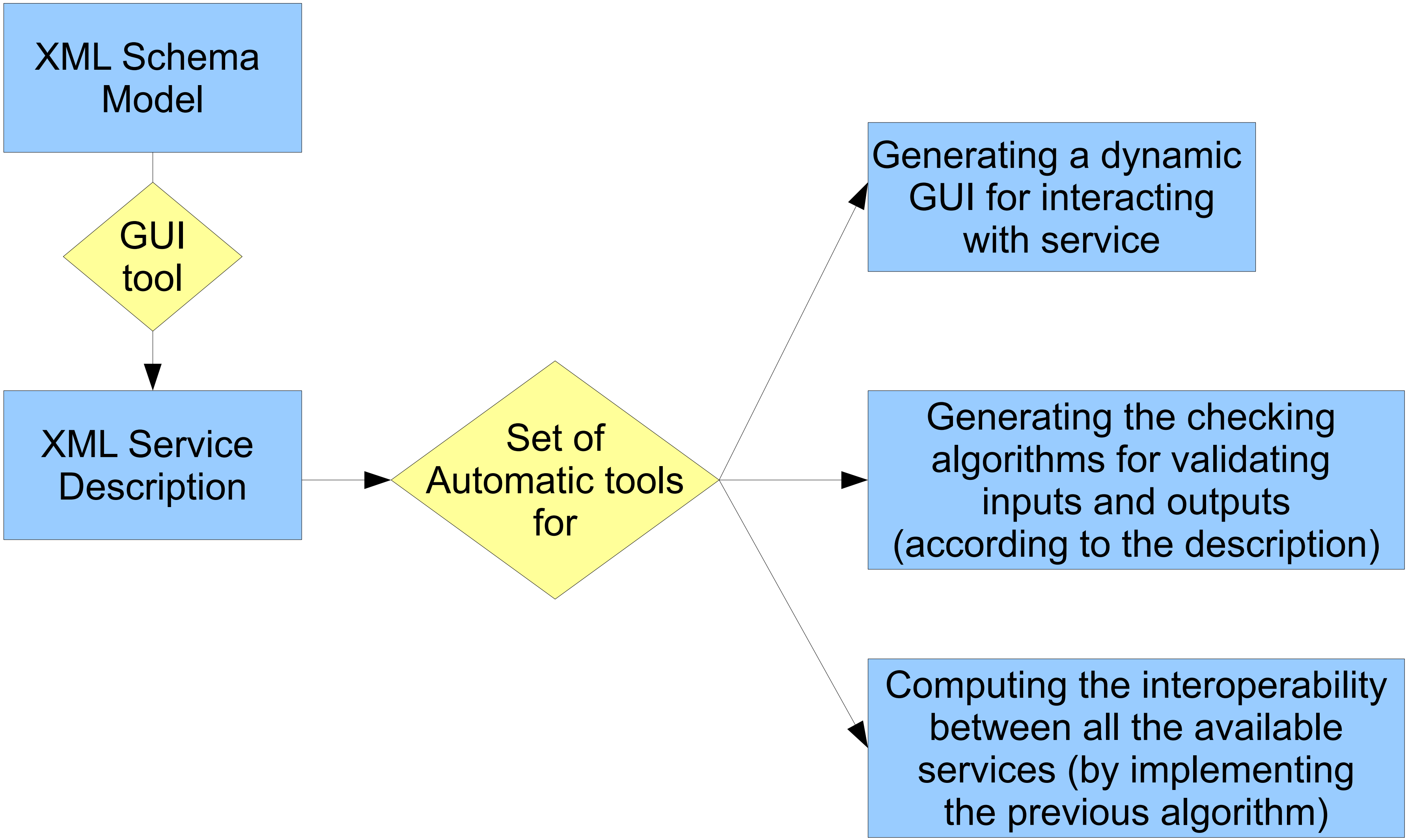
- $p_1 \in \mathbb{R}$, $p_2 \in \mathbb{N}$ and p_3 is boolean.
- if $p_1 > 0 \implies p_2 \in \{2; 4; 6\}$ and p_3 must be false.
- if $p_1 < 0 \implies p_3$ must be true.



Sketch of the integral solution



Sketch of the integral solution



Concluding remarks

With our formalism:

- Users can easily describe parameters and overall their constraints in a **unified way**
- Descriptions are human readable and could be understood by computers.

Interoperability graphs connecting services can be computed *a priori* automatically

It is a consistent step towards a real and integrated interoperability in the VO.

Concluding remarks

With our formalism:

- Users can easily describe parameters and overall their constraints in a **unified way**
- Descriptions are human readable and could be understood by computers.

Interoperability graphs connecting services can be computed *a priori* automatically

It is a consistent step towards a real and integrated interoperability in the VO.

Thank you for your kind attention.