Information technology for distributed engineering systems

Jochen Hanff
Institut für Bauingenieurwesen
FG Theoretische Methoden der Bau- und Verkehrstechnik
TU Berlin

- Distributed environments and applications
- Exchange of data
  - XML
- Remote objects
  - RMI
  - CORBA
- Object structures and algorithms in engineering
  - Dependencies of objects
  - Sequence of modification
- Conclusion

Object oriented method

- An application consists of **structured sets of objects**
- Each object consists of **methods** and **attributes**
  - The methods describe the behaviour of an object
  - The attributes contain the data which is stored in an object

Distributed environments

- Exchange of **data**
  - Remote method invocation
  - Exchange of files
- Using remote **functionality**
  - Remote method invocation

Exchange of Data

Extensible Markup Language (XML)

- XML is a document processing standard proposed by the World Wide Web Consortium (W3C)
- XML is a meta-language that allows you to create your own document markups
- XML documents are similar to HTML documents (in fact, HTML can be defined in XML)

Overview of an XML document

- XML Document (required)
- Document Type Definition (DTD) (optional)
- Stylesheet (XSL, XSLT) (optional)

Example of a simple XML document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE objects SYSTEM "objects.dtd">
<!-- this is a comment  -->
<objects>
   <point id="x1" x="1.0"  y="1.0"/>
   <point id="x2" x="3.0"  y="1.5"/>
   <point id="x3" x="2.0"  y="2.0"/>
   <triangle id="x4">
      <p1>x1</p1>
      <p2>x2</p2>
      <p3>x3</p3>
   </triangle>
</objects>
```

Example of a simple XML document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE objects SYSTEM "objects.dtd">
<!-- this is a comment  -->
<objects>
   <point id="x1" x="1.0"  y="1.0"/>
   <point id="x2" x="3.0"  y="1.5"/>
   <point id="x3" x="2.0"  y="2.0"/>
   <triangle id="x4">
      <p1>x1</p1>
      <p2>x2</p2>
      <p3>x3</p3>
   </triangle>
</objects>
```

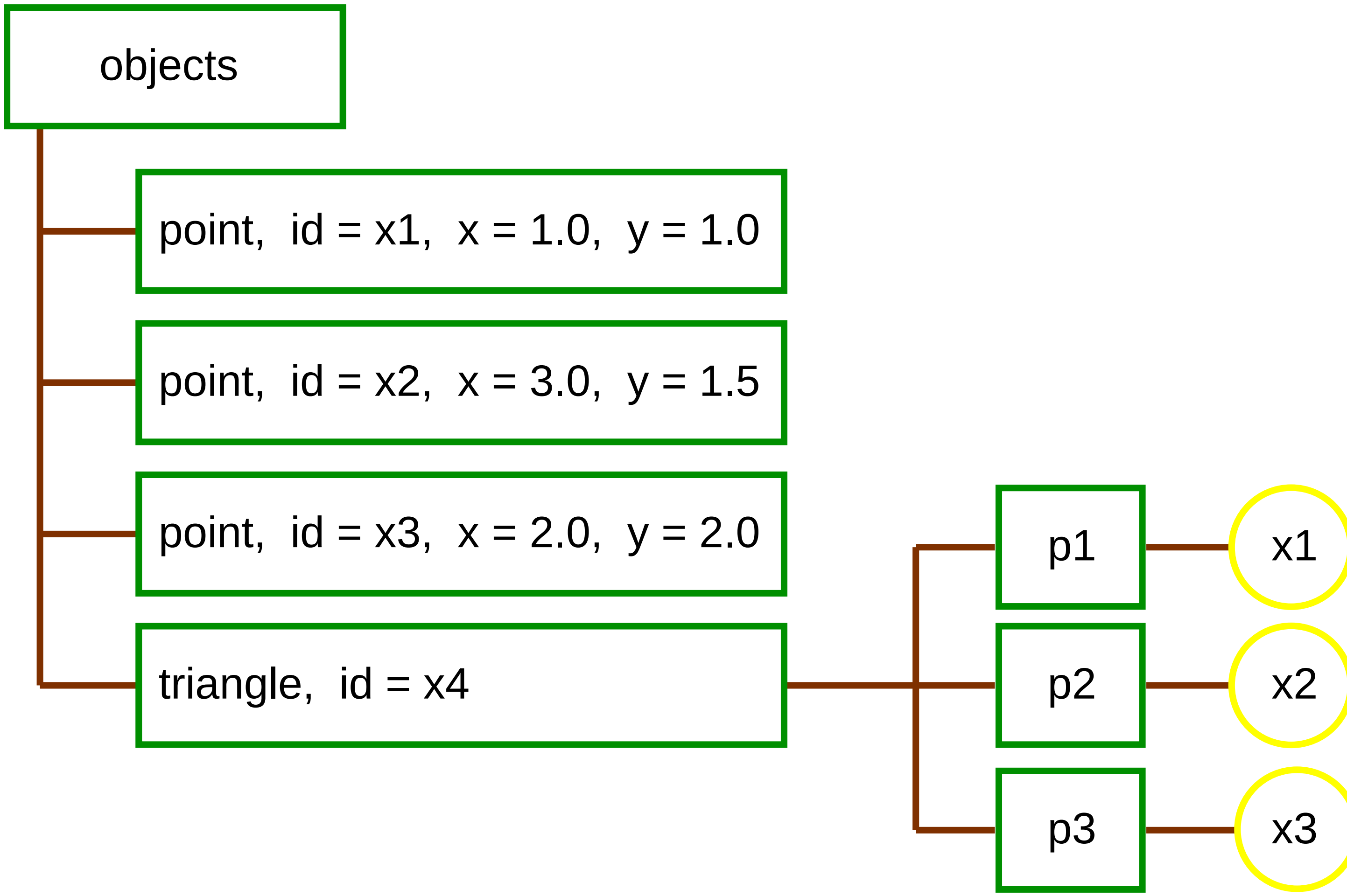
Example of a simple XML document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE objects SYSTEM "objects.dtd">
<!-- this is a comment  -->
<objects>
   <point id="x1" x="1.0"  y="1.0"/>
   <point id="x2" x="3.0"  y="1.5"/>
   <point id="x3" x="2.0"  y="2.0"/>
   <triangle id="x4">
      <p1>x1</p1>
      <p2>x2</p2>
      <p3>x3</p3>
   </triangle>
</objects>
```

Example of a simple XML document

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE objects SYSTEM "objects.dtd">
<!-- this is a comment  -->
<objects>
   <point id="x1" x="1.0"  y="1.0"/>
   <point id="x2" x="3.0"  y="1.5"/>
   <point id="x3" x="2.0"  y="2.0"/>
   <triangle id="x4">
      <p1>x1</p1>
      <p2>x2</p2>
      <p3>x3</p3>
   </triangle>
</objects>
```

Hierarchy :


objects
point, id = x1, x = 1.0, y = 1.0
point, id = x2, x = 3.0, y = 1.5
point, id = x3, x = 2.0, y = 2.0
triangle, id = x4
p1
p2
p3
x1
x2
x3

Document Type Definition (DTD) : 'objects.dtd'

```
<!ELEMENT objects (triangle*,point*)>

<!ELEMENT point EMPTY>
<!ATTLIST point id  ID    #REQUIRED
                x   CDATA #REQUIRED
                y   CDATA #REQUIRED>

<!ELEMENT triangle (p1,p2,p3)>
<!ATTLIST triangle id ID #REQUIRED>

<!ELEMENT p1 (#PCDATA)>
<!ELEMENT p2 (#PCDATA)>
<!ELEMENT p3 (#PCDATA)>
```

Extensible Stylesheet Language (XSL)
Extensible Stylesheet Language Transformations (XSLT)

- XSL is used to describe how an XML source document is transformed into another document
- XSL documents are themselves XML documents

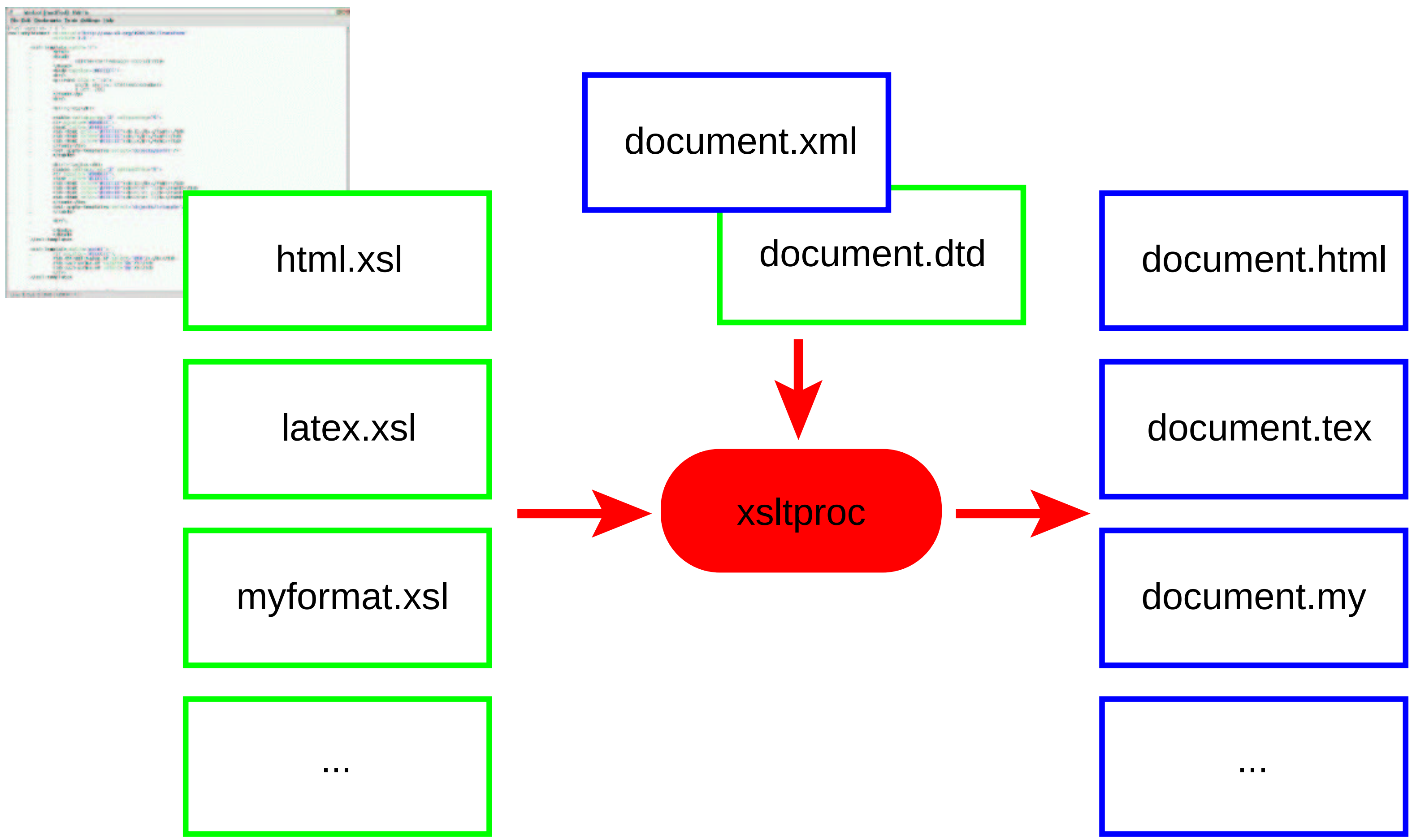
html.xsl [modified] - KWrite

File Edit Bookmarks Tools Settings Help

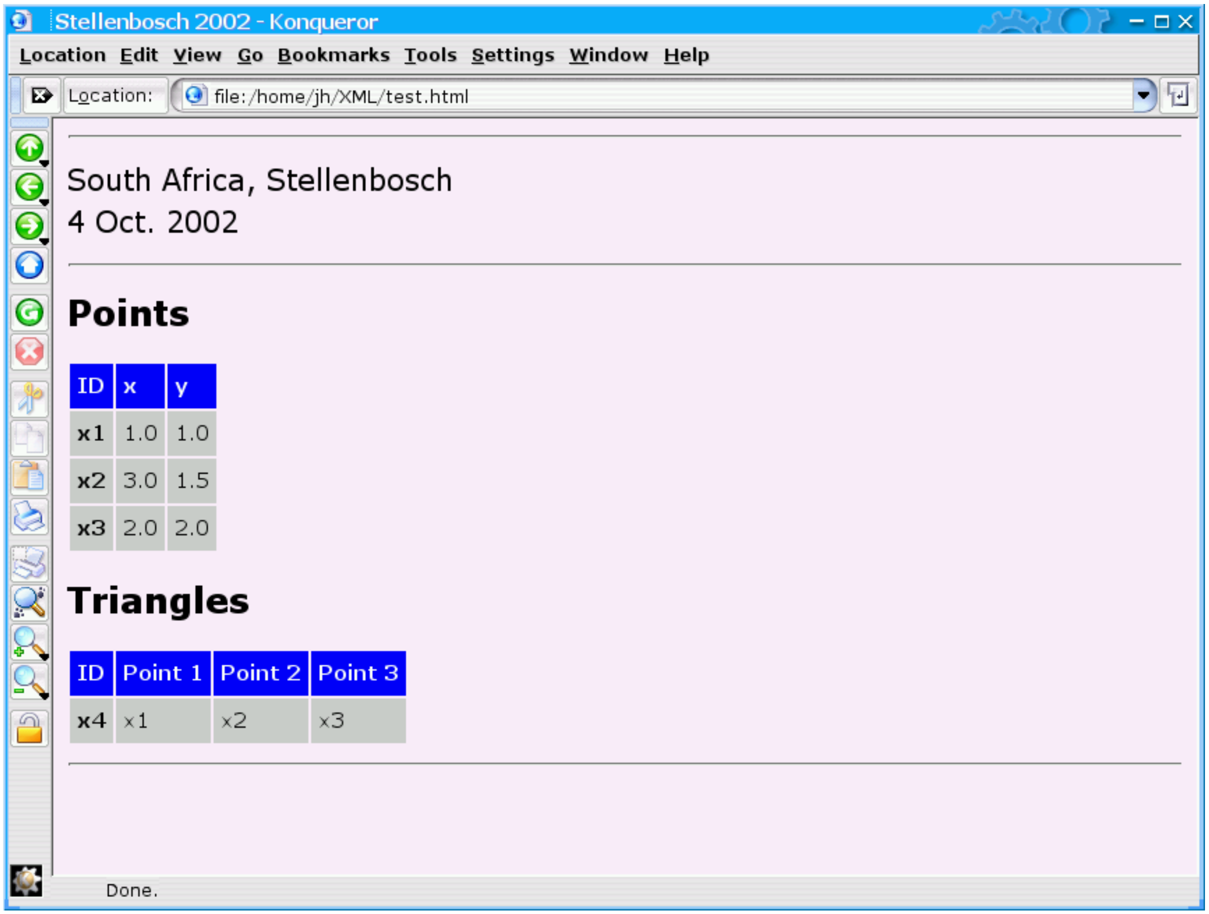
```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version='1.0'>

        <xsl:template match="/">
                <html>
                <head>
                        <title>Stellenbosch 2002</title>
                </head>
                <body bgcolor="#FFEEFF">
                <hr/>
                <p><font size = "+2">
                        South Africa, Stellenbosch<br/>
                        4 Oct. 2002
                </font></p>
                <hr/>

                <h1>Points</h1>

                <table cellspacing="2" cellpadding="5">
                <tr bgcolor="#0000FF">
                <font color="#FFFFFF">
                <td><font color="#FFFFFF"><b>ID</b></font></td>
                <td><font color="#FFFFFF"><b>x</b></font></td>
                <td><font color="#FFFFFF"><b>y</b></font></td>
                </font></tr>
                <xsl:apply-templates select="objects/point"/>
                </table>

                <h1>Triangles</h1>
                <table cellspacing="2" cellpadding="5">
                <tr bgcolor="#0000FF">
                <font color="#FFFFFF">
                <td><font color="#FFFFFF"><b>ID</b></font></td>
                <td><font color="#FFFFFF"><b>Point 1</b></font></td>
                <td><font color="#FFFFFF"><b>Point 2</b></font></td>
                <td><font color="#FFFFFF"><b>Point 3</b></font></td>
                </font></tr>
                <xsl:apply-templates select="objects/triangle"/>
                </table>

                <hr/>

                </body>
                </html>
        </xsl:template>

        <xsl:template match="point">
                <tr bgcolor="#CCCCCC">
                <td><b><xsl:value-of select="@id"/></b></td>
                <td><xsl:value-of select="@x"/></td>
                <td><xsl:value-of select="@y"/></td>
                </tr>
        </xsl:template>
```

Line: 1 Col: 0 INS NORM *

Extensible Stylesheet Language Transformations (XSLT)



Extensible Stylesheet Language (XSL)



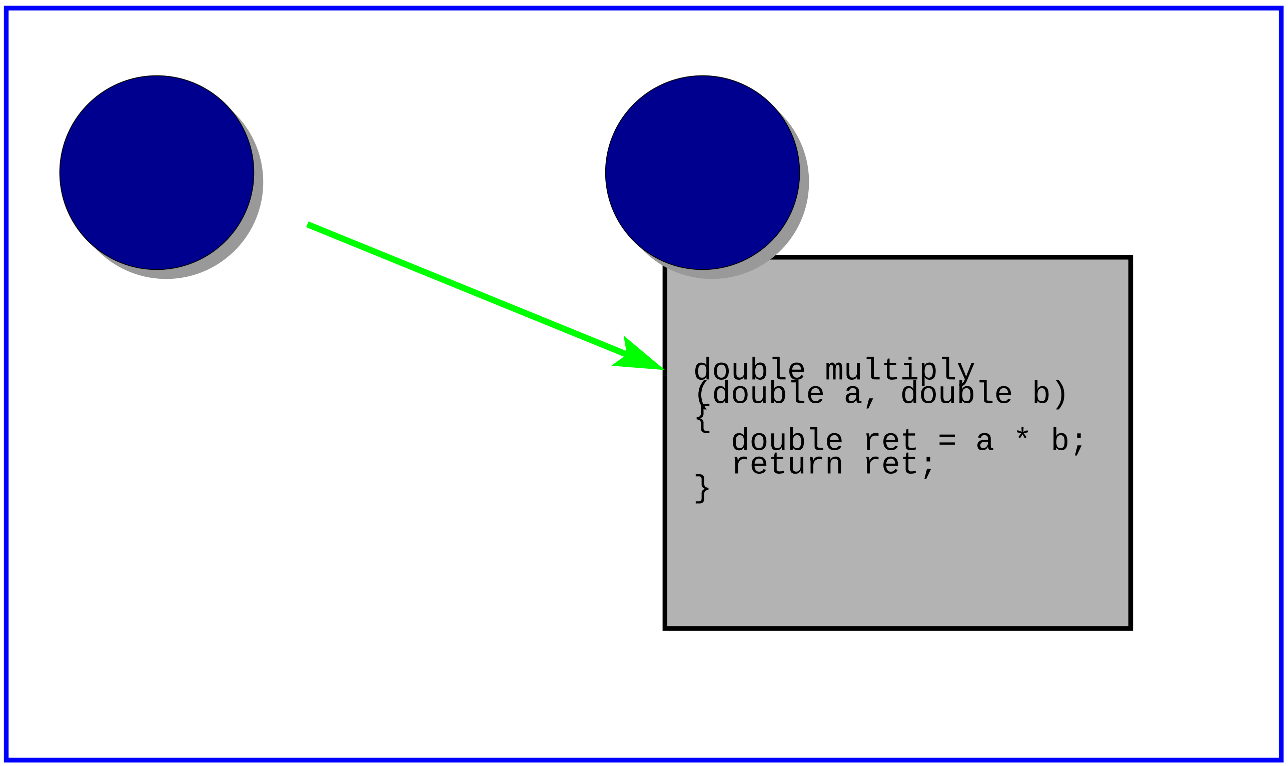
Advantages of XML

- structured data
- standardized description of structure
- tools available
  (parser, xslt processor, xml validator ...)

Disadvantages of XML

- File size
- Increased processing time

Using remote functionality



Client/server model

Using remote functionality

JAVA Remote method invocation (RMI)

- Available since JAVA 1.1
- RMI is a mechanism that enables an object
  on one Java virtual machine to invoke methods on an object
  in another Java virtual machine

JAVA Remote method invocation (RMI)

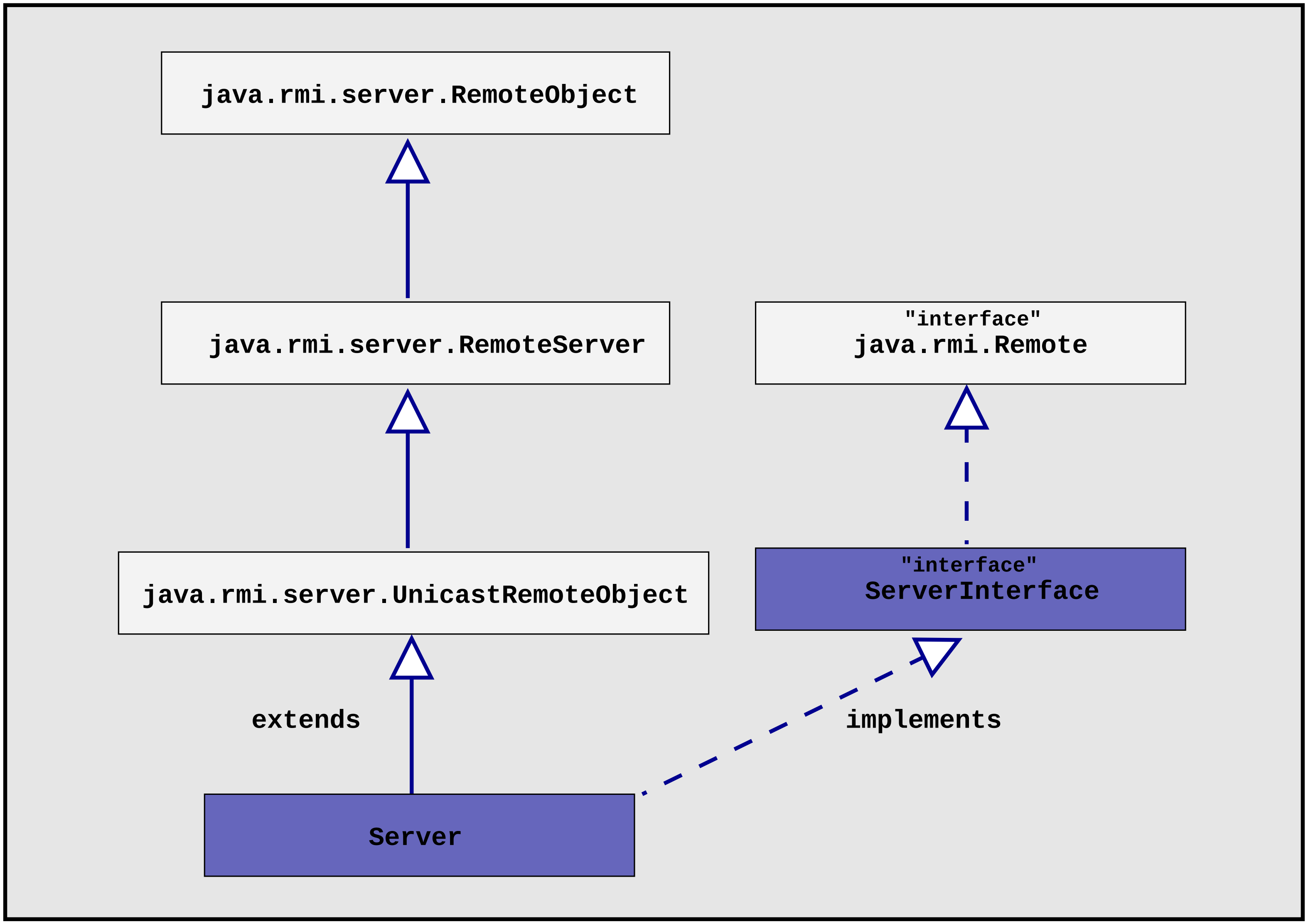
1. Define a JAVA interface which inherits `java.rmi.Remote`

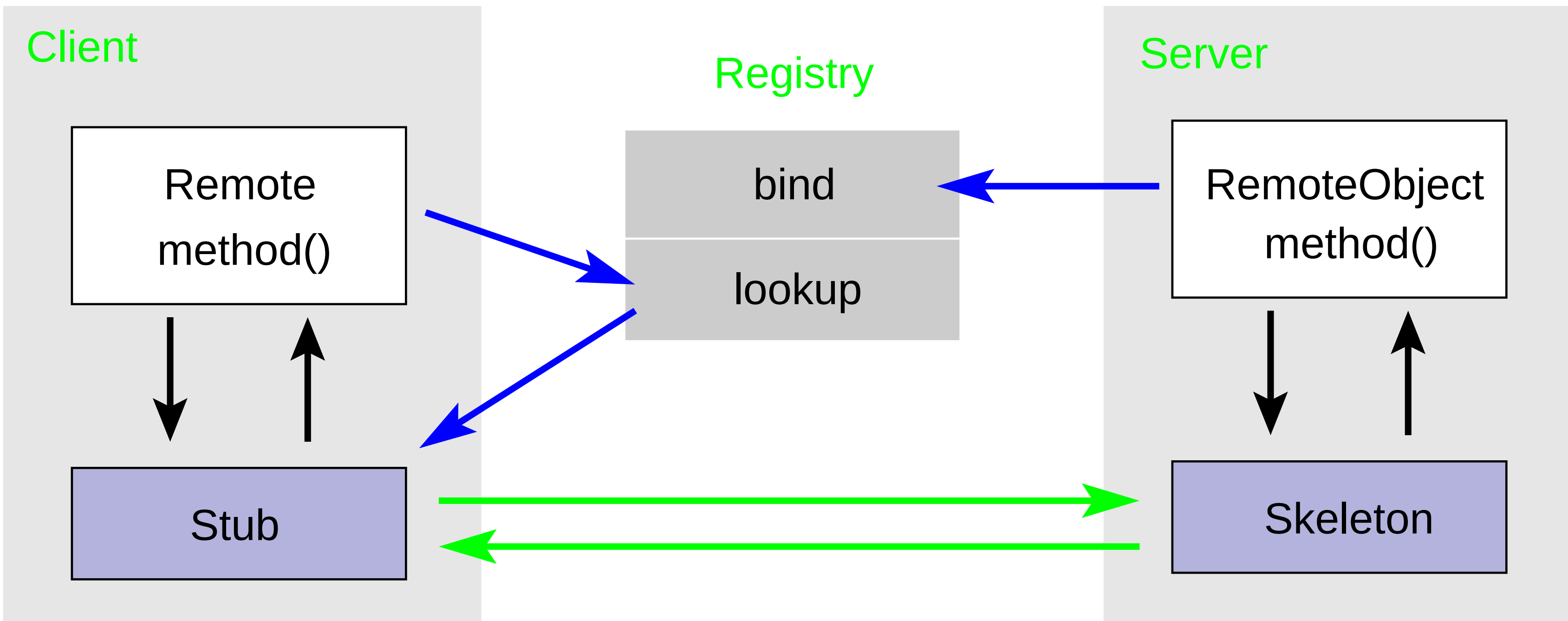
3. Define a server class which inherits `java.rmi.server.UnicastRemoteObject` and which implements your interface

4. Generate stubs/skeletons using `rmic`

5. Bind object to JAVA registry

Class hierarchy for remote objects




Client
Registry
Server
Remote
method()
bind
lookup
RemoteObject
method()
Stub
Skeleton

Advantages of JAVA RMI

- Easy to use
- No additional interface description language

Disadvantages of JAVA RMI

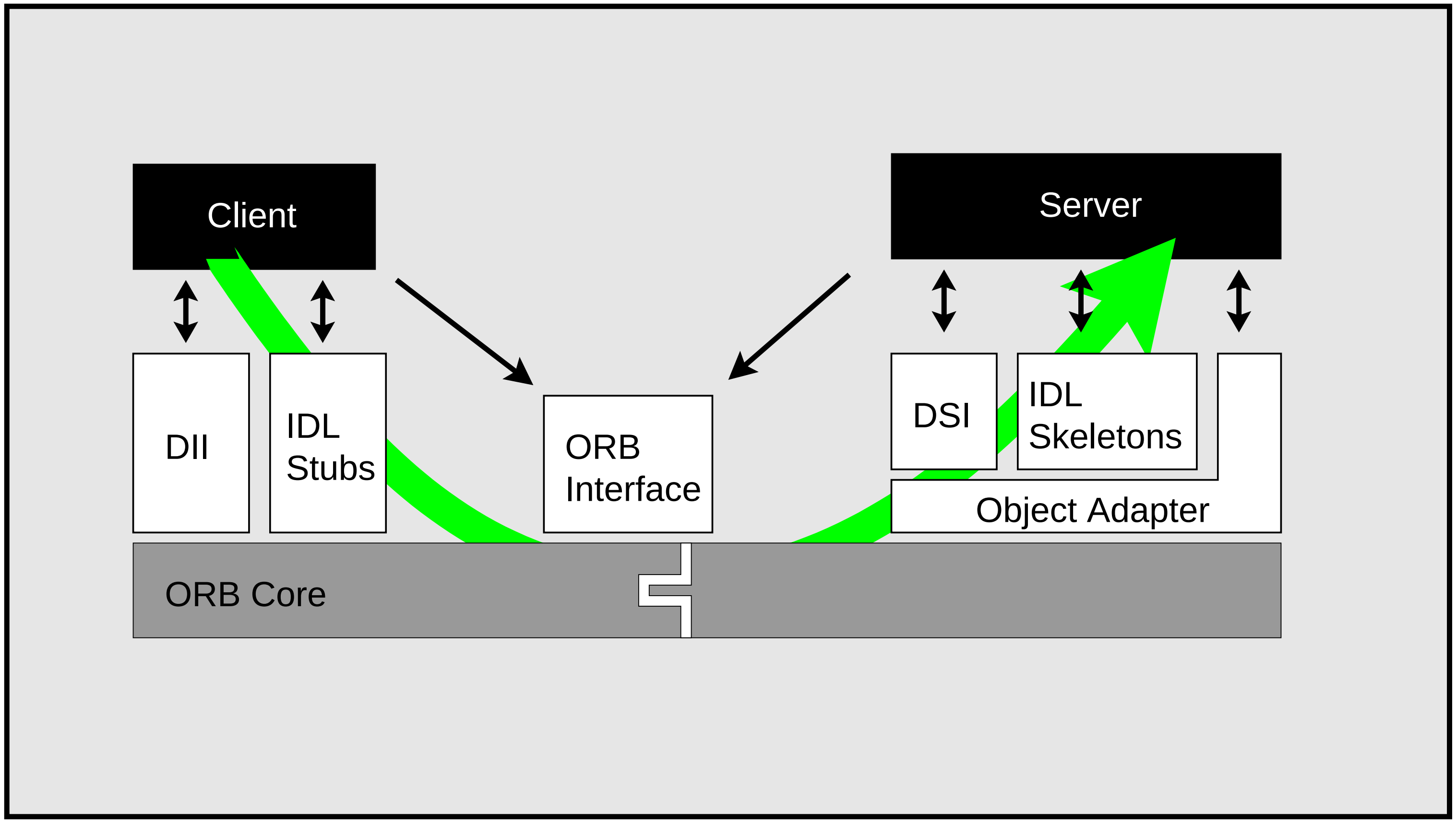
- Limited to JAVA platform

Common Object Request Broker Architecture (CORBA)

- Specified by the Object Management Group (OMG)
- Platform for distributed object oriented applications in heterogeneous environments
- Language bindings for C++, JAVA, Smalltalk ...
- Commercial and free implementations

Parts of CORBA

- Interface Definition Language (IDL)
- ORB Core
- ORB Interface
- Static interfaces (stubs, skeletons)
- Dynamic Interfaces (Dynamic Skeleton Interface, Dynamic Invocation Interface)
- Object Adapters (Basic Object Adapter, Portable Object Adapter)
- Inter-ORB protocols


Client
Server
DII
IDL
Stubs
ORB
Interface
DSI
IDL
Skeletons
Object Adapter
ORB Core

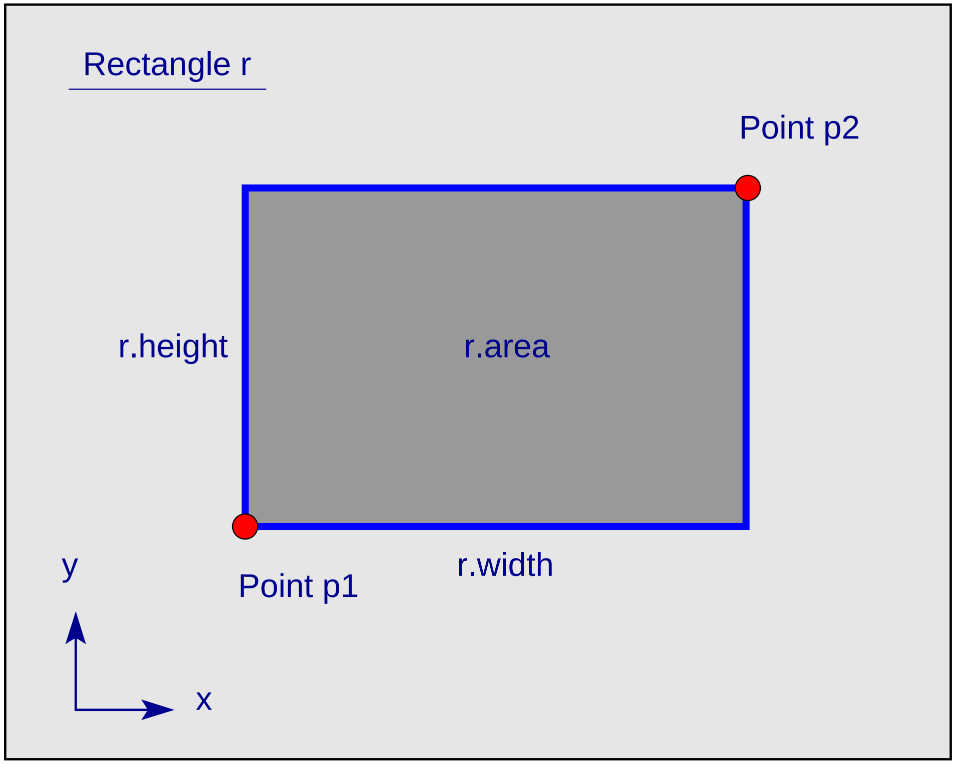
Advantages of CORBA

- Platform independent
- Language bindings for several languages
- Standardized services (notification, life cycle, trading ...)

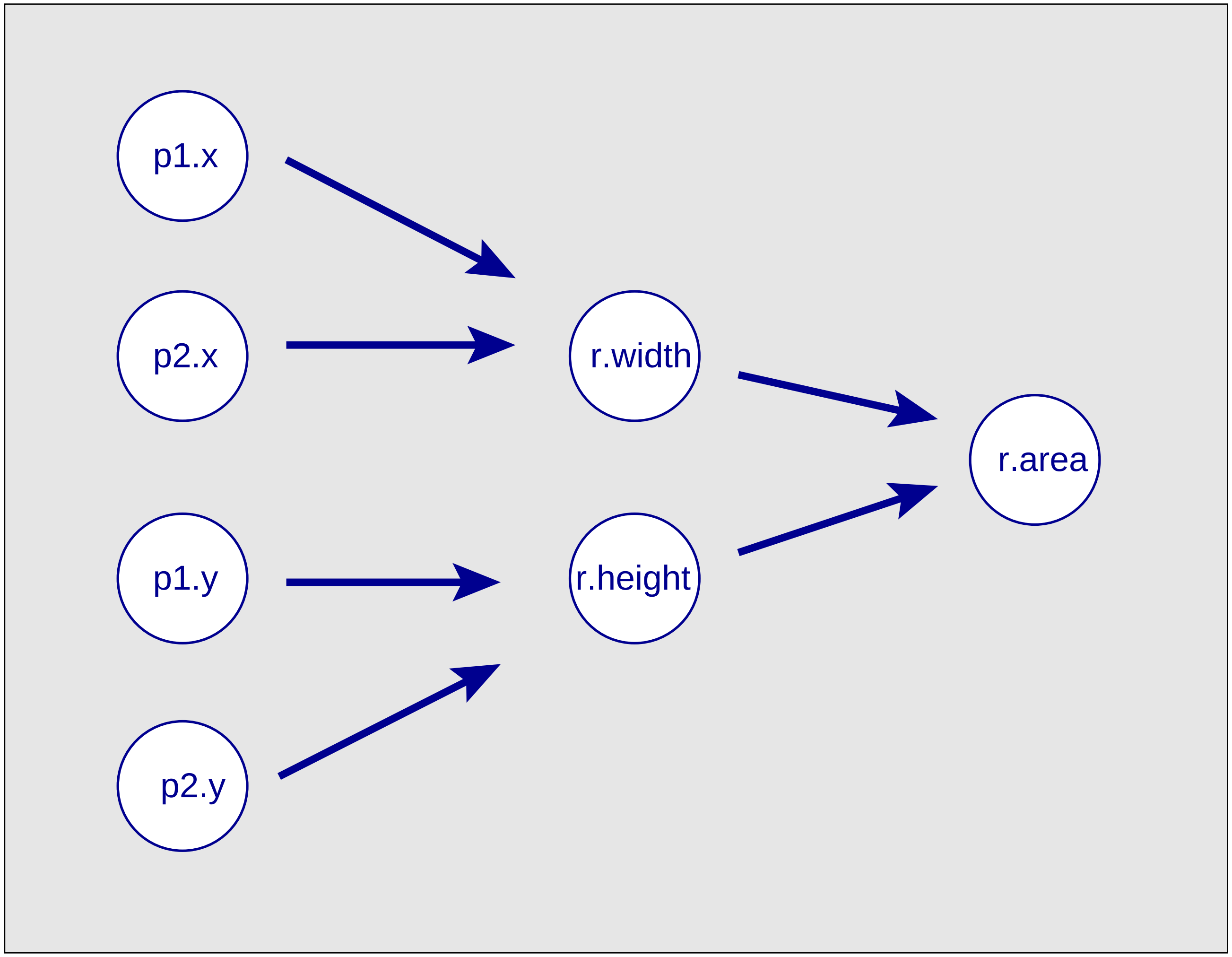
Disadvantages of CORBA

- Increased overhead
- Performance (?)

Dependencies of objects



$$
\begin{aligned}
r.area &= r.height \cdot r.width \\
r.height &= p2.y - p1.y \\
r.width &= p2.x - p1.x
\end{aligned}
$$


p1.x
p2.x
p1.y
p2.y
r.width
r.height
r.area

Object structures and algorithms in engineering

- High complexity of objects
- Relations to other objects are manifold
- Complex algorithms with high computational effort

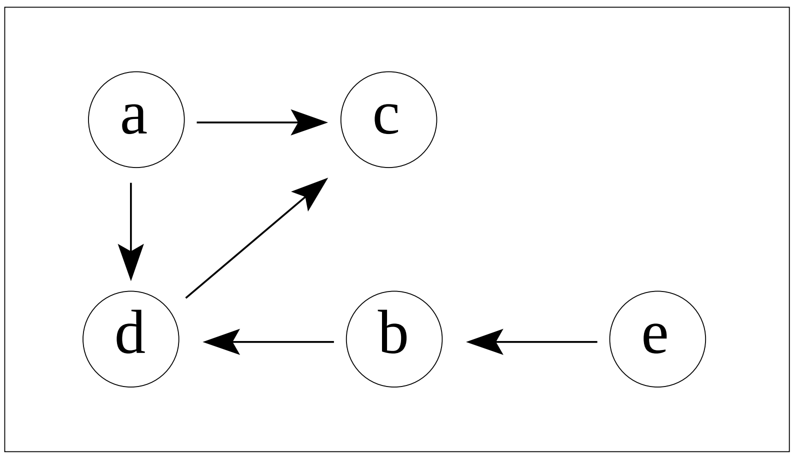
This leads to :

- Delayed updates
- Correct sequence of modification of attributes has to be determined

$\Rightarrow$ **Graph theory**

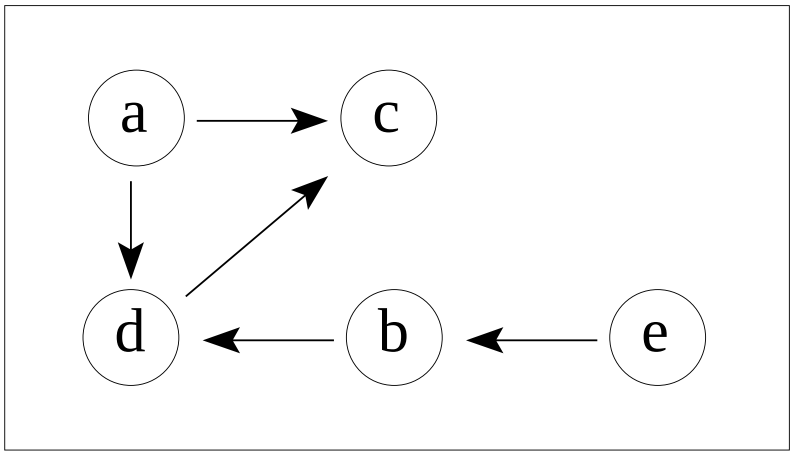
$$\boxed{\mathsf{B} := \{\ (a,b) \in \mathsf{A} \times \mathsf{A} \mid a \ \textsf{binds}\ b\ \}}$$

$$\mathsf{B} = \{ \quad (a,d),(a,c),(d,c),\\ (b,d),(e,b) \quad \}$$



$$\mathsf{B} = \begin{array}{cc} \left[\begin{array}{ccccc} \circ & \circ & \bullet & \bullet & \circ \\ \circ & \circ & \circ & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \bullet & \circ & \circ \\ \circ & \bullet & \circ & \circ & \circ \end{array}\right] & \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} \\ \begin{array}{ccccc} a & b & c & d & e \end{array} & \end{array}$$

$$\boxed{\mathsf{H} := \{\ (a,b) \in \mathsf{A} \times \mathsf{A} \mid b \text{ depends on } a\ \}}$$

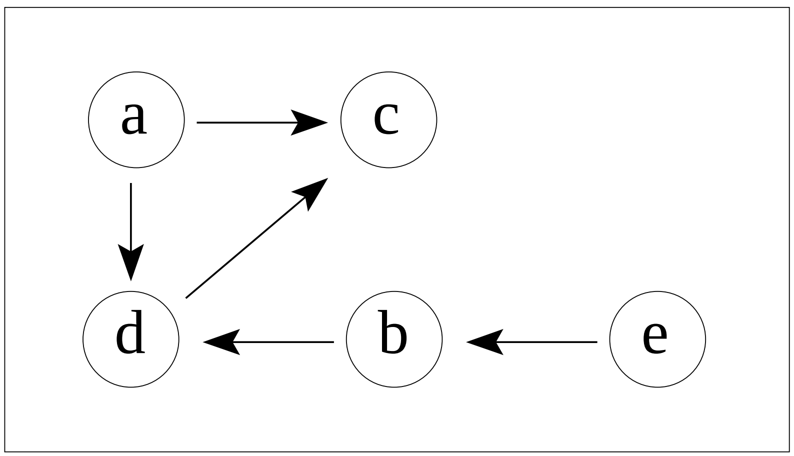


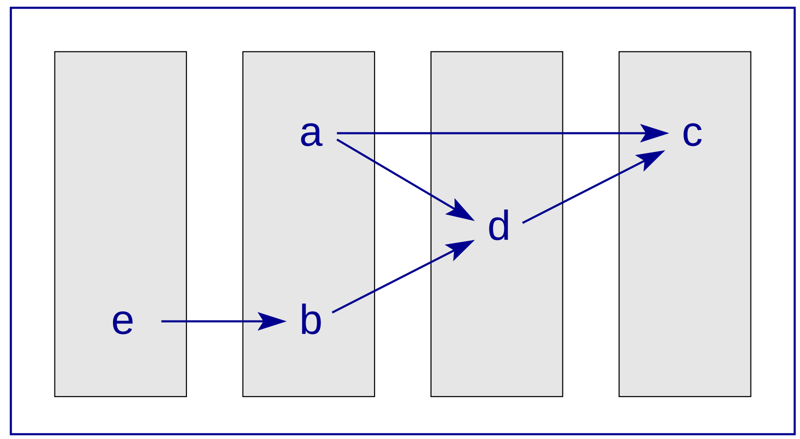
$$\mathsf{H} = \mathsf{B} \cup \mathsf{B}^2 \cup \ldots \cup \mathsf{B}^{n-1}$$

$$\mathsf{H} = \begin{array}{cc} \begin{bmatrix} \circ & \circ & \bullet & \bullet & \circ \\ \circ & \circ & \bullet & \bullet & \circ \\ \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \bullet & \circ & \circ \\ \circ & \bullet & \bullet & \bullet & \circ \end{bmatrix} & \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} \\ \begin{array}{ccccc} a & b & c & d & e \end{array} & \end{array}$$

Sequence of modification

Topological sorting





Attribute c has to be updated.

Correct sequence of modification : $< e, a, b, d, c >$

System

1. Elements — E : Set of elements
2. Properties — F : Set of properties
3. Algorithms — $(f_1, \cdots, f_m) \longrightarrow \boxed{\mathsf{A}} \longrightarrow (h_1, \cdots, h_n)$

   A : Set of algorithms
4. Complete binding relation — $\mathsf{B} := (\mathsf{A}, \mathsf{F}, R_e, R_a)$

   $R_e \subseteq \mathsf{F} \times \mathsf{A}$ Input parameters, algorithms

   $R_a \subseteq \mathsf{A} \times \mathsf{F}$ Algorithms, output parameters
5. System — $\mathsf{S} := (\mathsf{E}, \mathsf{A}, \mathsf{F}; \mathsf{B})$

Update of a system

Choosing properties which have to be updated :

$$\mathsf{Z} := \{a \in \mathsf{F} \mid a \text{ is choosen for an update}\}$$

Domain of properties :

$$\mathsf{D_F} := \{a \in \mathsf{F} \mid a \text{ has to be updated}\}$$

Domain of algorithms :

$$\mathsf{D_A} := \{A \in \mathsf{A} \mid \quad \text{Algorithm } A \text{ must be invoked for updating an element in } \mathsf{D_F}\ \}$$

Domain of update :

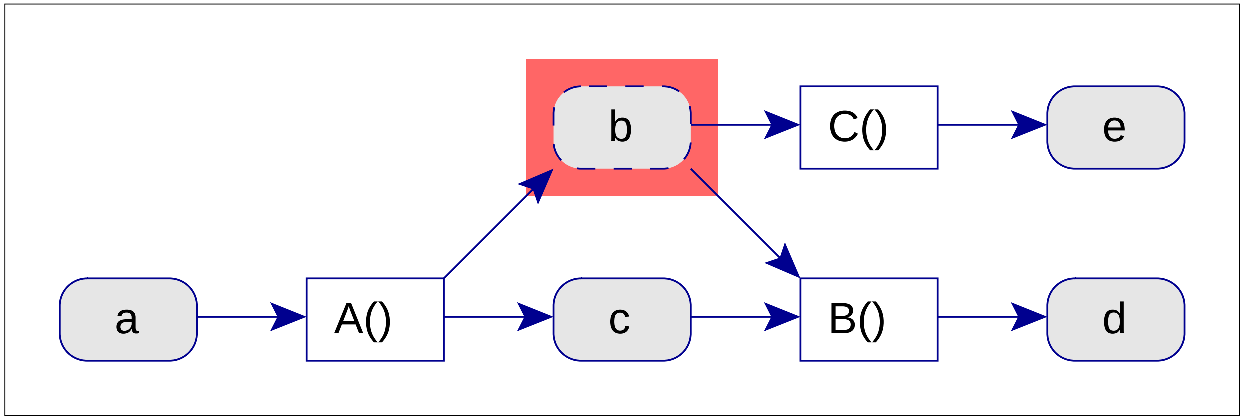
$$\mathsf{D} = \mathsf{D_A} \cup \mathsf{D_F}$$

Reduced binding relation

$$\bar{\mathsf{B}} = \Gamma^T \mathsf{B} \Gamma$$

$$\bar{\mathsf{B}} := \{(a,b) \in \mathsf{D} \times \mathsf{D} \mid (a,b) \in \mathsf{B}\}$$

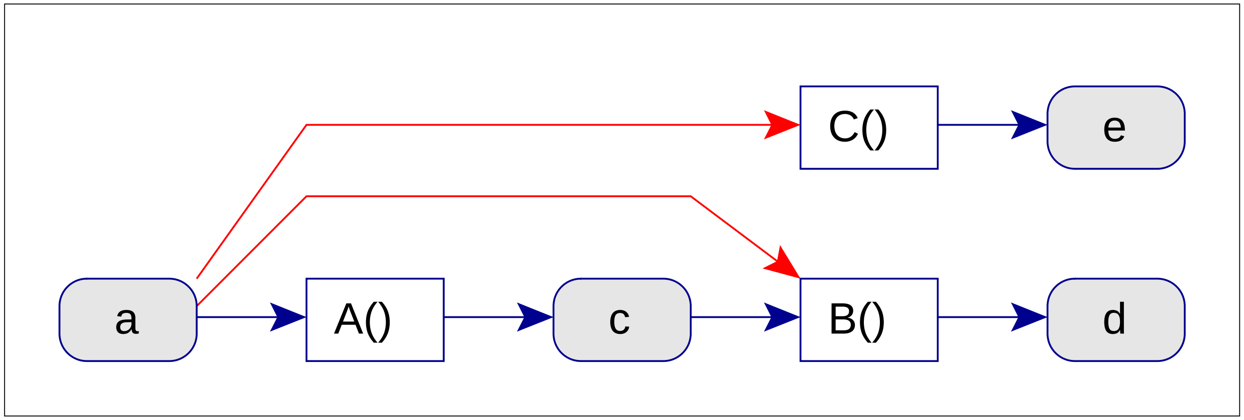
Removing virtual properties



$$\bar{\mathsf{B}} = \{(a,A),(A,b),(A,c),(b,C),(b,B),(c,B),(C,e),(B,d)\}$$

$$\mathsf{D} = \{a,A,b,c,C,B,e,d\}$$

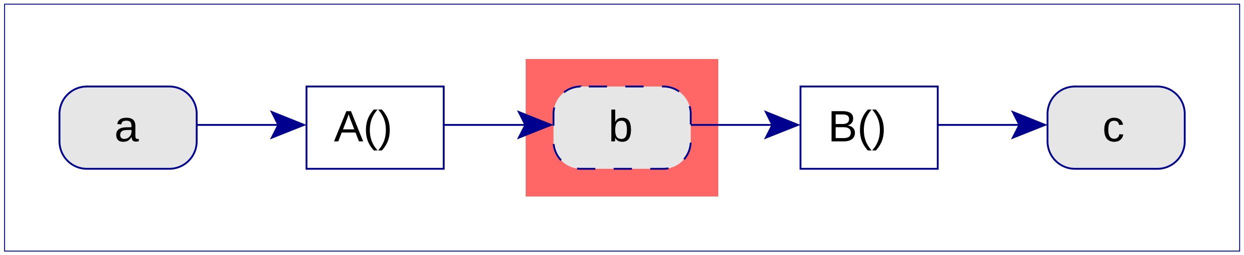
Removing virtual properties



$$
\begin{aligned}
\bar{\mathsf{B}}_{\mathsf{v}} &= \bar{\mathsf{B}} \setminus \{(A,b),(b,C),(c,B),(b,B)\} \cup \{(a,C),(a,B)\} \\
&= \{(a,A),(a,C),(a,B),(A,c),(c,B),(C,e),(B,d)\}
\end{aligned}
$$

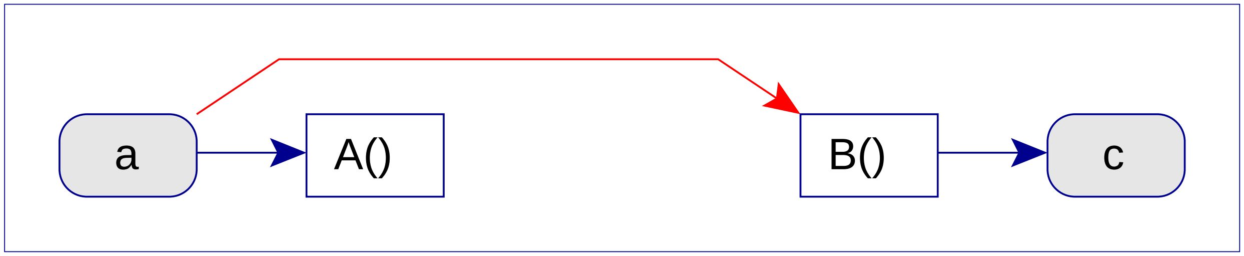
$$
\mathsf{D}_{\mathsf{v}} = \{a,A,c,C,B,e,d\}
$$

Removing methods without successors



$$\bar{\mathsf{B}} = \{(a, A), (A, b), (b, B), (B, c)\}$$
$$\mathsf{D} = \{a, A, b, B, c\}$$

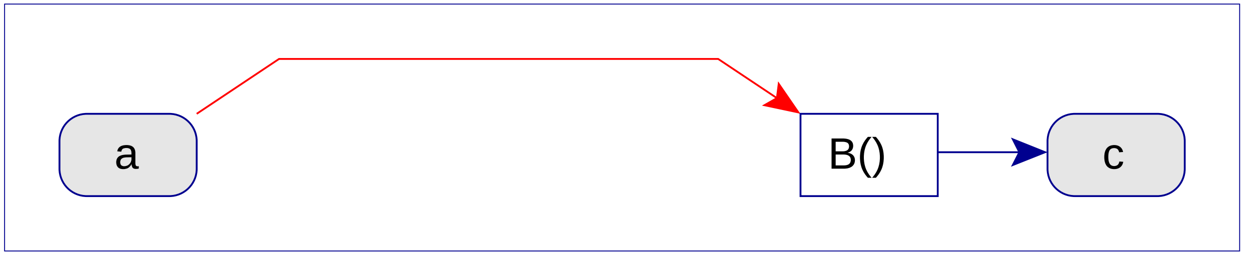
Removing methods without successors



$$\bar{\mathsf{B}}_{\mathsf{v}} = \{(a, A), (a, B), (B, c)\}$$

$$\mathsf{D}_{\mathsf{v}} = \{a, A, B, c\}$$

Removing methods without successors

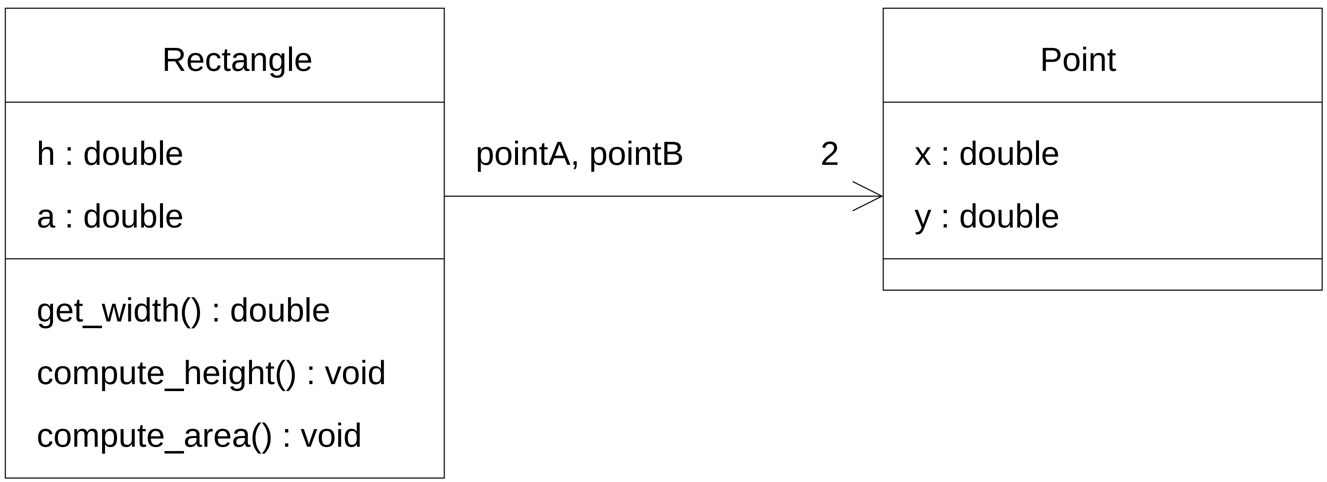


$$\bar{\mathsf{B}}_{\mathsf{v,m}} = \{(a, B), (B, c)\}$$
$$\mathsf{D}_{\mathsf{v,m}} = \{a, B, c\}$$

$\boxed{\text{Sequence of modification}}$

Calculation of $\bar{\mathsf{B}}_{\mathsf{A}}$	$\bar{\mathsf{B}}_{\mathsf{A}} = \Gamma_A^T \, \bar{\mathsf{B}}_{\mathsf{v,m}}^2 \, \Gamma_A$
Topological sorting	$\bar{\mathsf{B}}_{\mathsf{A}} \quad \longrightarrow \quad a$
Invoking methods	$a =< B, D, A, ... >$

Example


Rectangle
h : double
a : double
get_width() : double
compute_height() : void
compute_area() : void
pointA, pointB
2
Point
x : double
y : double

Objects $\quad \mathsf{E} = \{ \quad r, p1, p2 \}$

Properties $\quad \mathsf{F} = \{ \quad p1.x, p1.y, p2.x, p2.y, r.pointA, r.pointB, r.width, r.h, r.a \}$

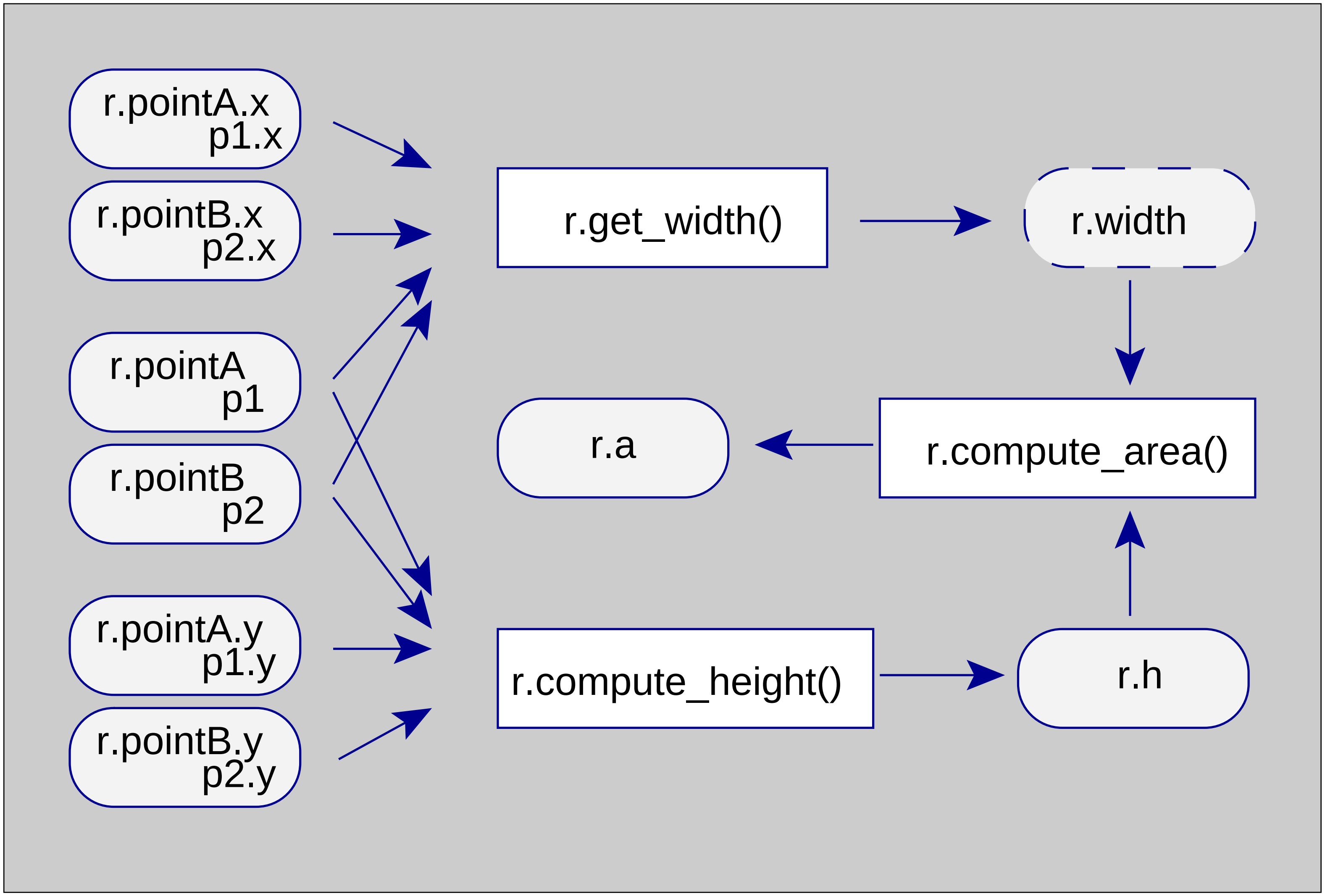
Algorithms $\quad \mathsf{A} = \{ \quad r.get\_width(), r.compute\_height(), r.compute\_area() \}$

$$(r.pointA.x, r.pointB.x) \longrightarrow \boxed{\text{r.get\_width()}} \longrightarrow (r.width)$$

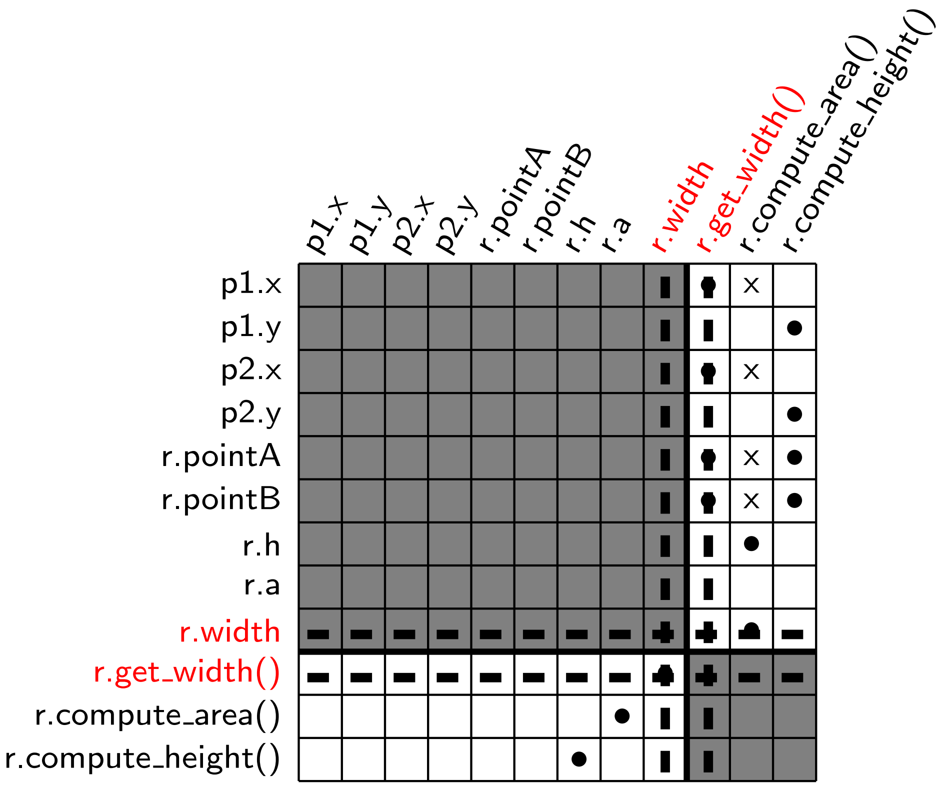
$$(r.pointA.y, r.pointB.y) \longrightarrow \boxed{\text{r.compute\_height()}} \longrightarrow (r.h)$$

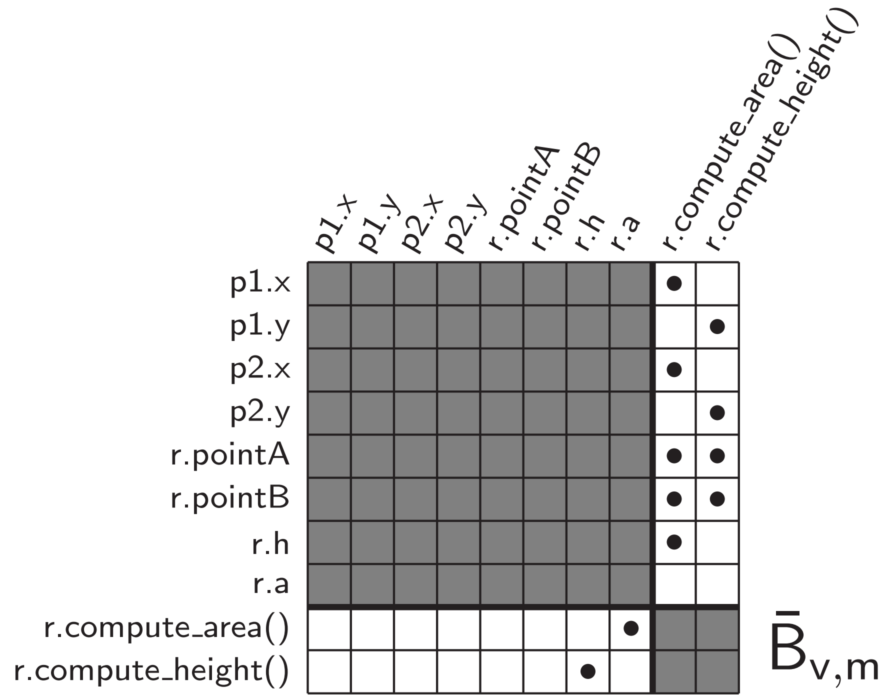
$$(r.h, r.width) \longrightarrow \boxed{\text{r.compute\_area()}} \longrightarrow (r.a)$$

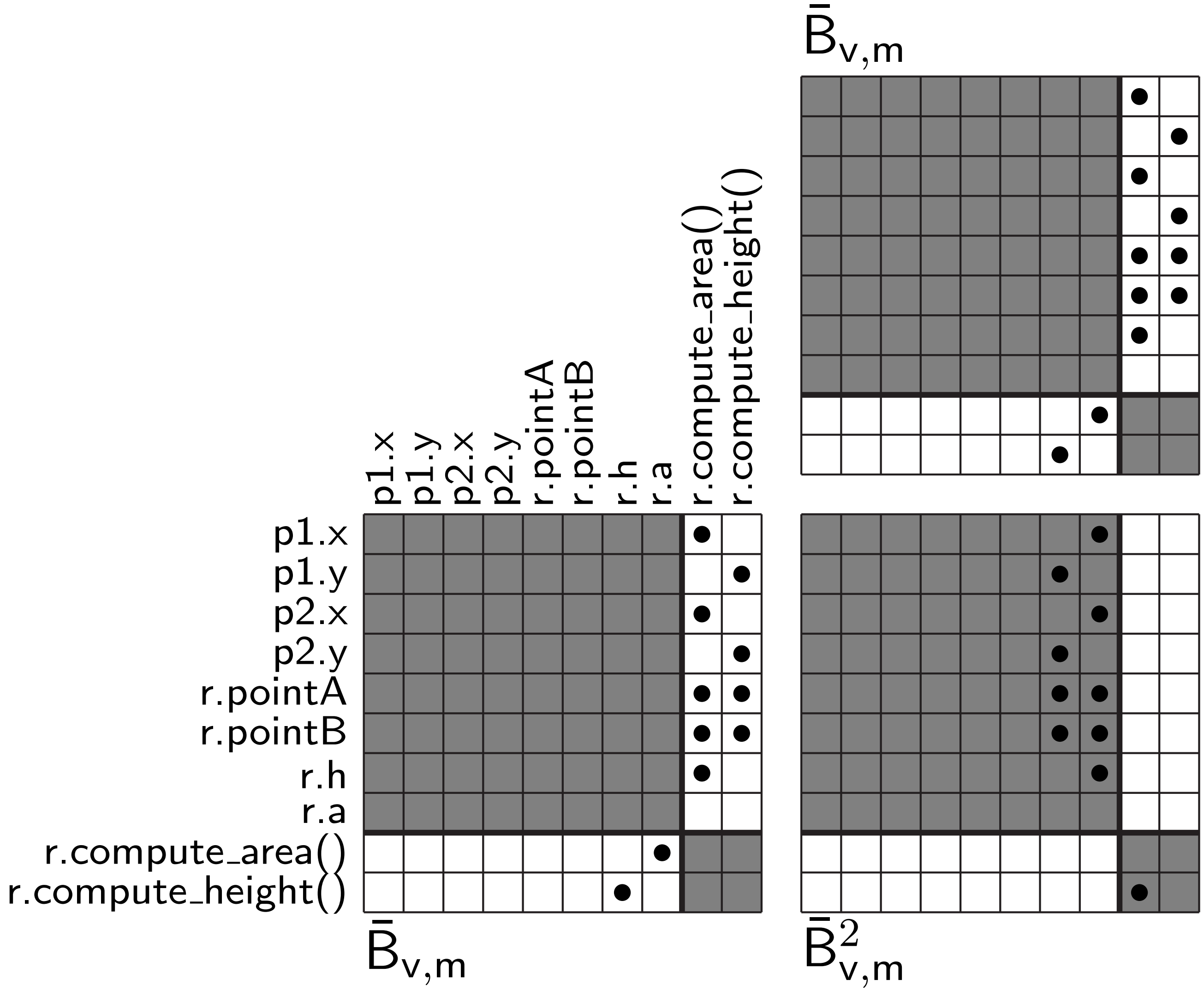
	p1.x	p1.y	p2.x	p2.y	r.pointA	r.pointB	r.h	r.a	r.width	r.get_width()	r.compute_area()	r.compute_height()
p1.x										●		
p1.y												●
p2.x										●		
p2.y												●
r.pointA										●		●
r.pointB										●		●
r.h											●	
r.a												
r.width											●	
r.get_width()									●			
r.compute_area()								●				
r.compute_height()							●					


r.pointA.x
p1.x
r.pointB.x
p2.x
r.pointA
p1
r.pointB
p2
r.pointA.y
p1.y
r.pointB.y
p2.y
r.get_width()
r.width
r.a
r.compute_area()
r.compute_height()
r.h

$$
\begin{aligned}
\mathsf{Z} &= \{r.a\} \\
\mathsf{D_F} &= \{r.a, r.breite, r.h, r.punktA, r.punktB, p1.x, p1.y, p2.x, p2.y\} \\
\mathsf{D_A} &= \{r.berechne\_flaeche(), r.breite(), r.berechne\_hoehe()\} \\
\mathsf{D} &= \mathsf{D_F} \cup \mathsf{D_A} \\
&= \{r.a, r.breite, r.h, r.berechne\_flaeche(), r.breite(), r.berechne\_hoehe(), \\
&\quad r.punktA, r.punktB, p1.x, p1.y, p2.x, p2.y\}
\end{aligned}
$$

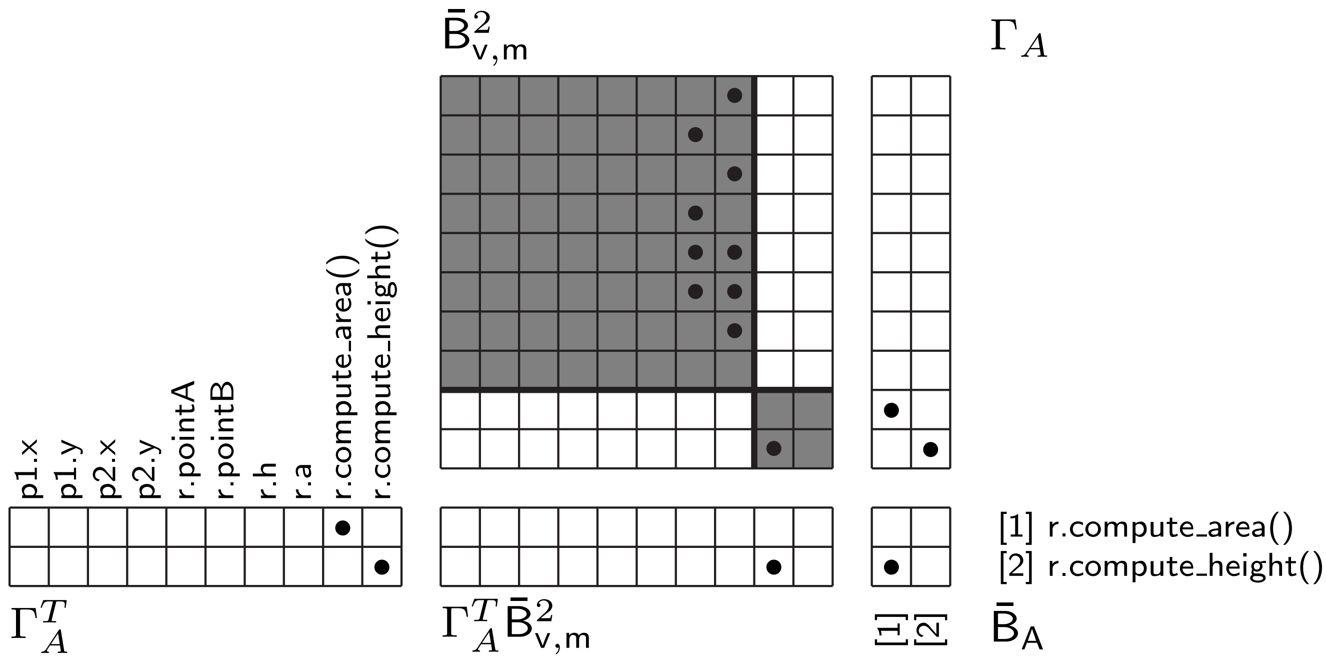

p1.x
p1.y
p2.x
p2.y
r.pointA
r.pointB
r.h
r.a
r.width
r.get_width()
r.compute_area()
r.compute_height()


p1.x
p1.y
p2.x
p2.y
r.pointA
r.pointB
r.h
r.a
r.compute_area()
r.compute_height()
$\bar{B}_{v,m}$

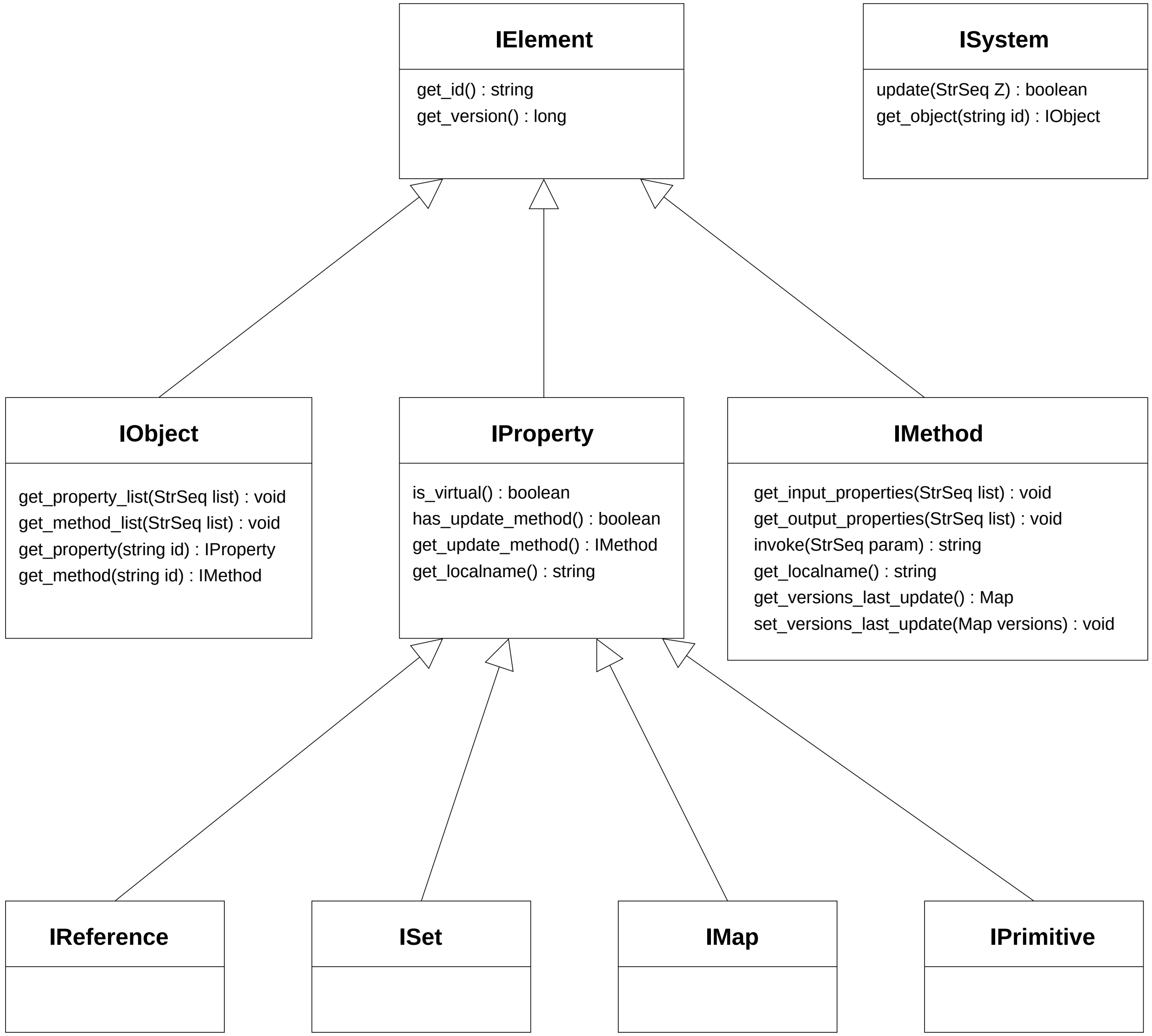


$\bar{B}_{v,m}$
p1.x
p1.y
p2.x
p2.y
r.pointA
r.pointB
r.h
r.a
r.compute_area()
r.compute_height()
$\bar{B}_{v,m}$
$\bar{B}^2_{v,m}$

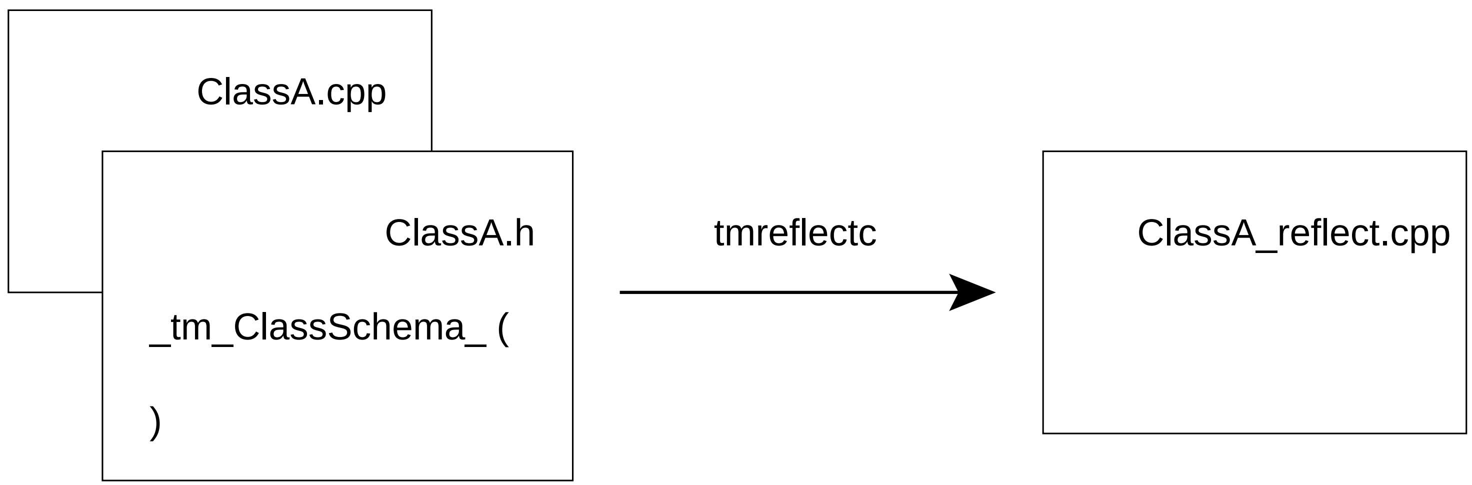
$$\bar{\mathsf{B}}_{\mathsf{A}} = \Gamma_A^T \bar{\mathsf{B}}^2_{\mathsf{v,m}} \Gamma_A \tag{1}$$



Topological sorting : $\bar{\mathsf{B}}_{\mathsf{A}} \quad \rightarrow \quad < r.compute\_height(), r.compute\_area() >$


IElement
get_id() : string
get_version() : long
ISystem
update(StrSeq Z) : boolean
get_object(string id) : IObject
IObject
get_property_list(StrSeq list) : void
get_method_list(StrSeq list) : void
get_property(string id) : IProperty
get_method(string id) : IMethod
IProperty
is_virtual() : boolean
has_update_method() : boolean
get_update_method() : IMethod
get_localname() : string
IMethod
get_input_properties(StrSeq list) : void
get_output_properties(StrSeq list) : void
invoke(StrSeq param) : string
get_localname() : string
get_versions_last_update() : Map
set_versions_last_update(Map versions) : void
IReference
ISet
IMap
IPrimitive

Metacompiler



```
class Point : public tm_core::tmObject
{
public:

  _tm_ClassSchema_
  ( <class name="Point">

    <superclass>tm_core::tmObject</superclass>

    <primitive name="X">
       <datatype>double</datatype>
       <read>get_x</read>
       <write>set_x</write>
       <activate>activate_x</activate>
       <info>"Coordinate x."</info>
    </primitive>

    <primitive name="Y">
       <datatype>double</datatype>
       <read>get_y</read>
       <write>set_y</write>
       <activate>activate_y</activate>
       <info>"Coordinate y."</info>
    </primitive>

    </class> );

  Point(double x, double y, tm_core::tmRepository* repo = 0);
```

```
  Point(tm_core::tmRepository* repo = 0);
  double get_x() const { return _x; }
  bool   set_x(double value);
  double get_y() const { return _y; }
  bool   set_y(double value);
  void   set_xy(double x, double y);

protected:

  /* --  this is needed for persistence  ------------------------------ */
  TM_OBJECT
  Point(const QString& id, tm_core::tmRepositoryP* repo = 0)
    : tm_core::tmObject(id,repo) {
    init();
  }
  void activate_x(double x) { _x = x; }
  void activate_y(double y) { _y = y; }
  /* ------------------------------------------------------------------ */

private:

  void init();
  double _x, _y;

};
```

```
class Rectangle : public tm_core::tmObject
{
public:

  _tm_ClassSchema_
  ( <class name="Rectangle">
    <superclass>tm_core::tmObject</superclass>

    <primitive name="Width" virtual="true">
       <datatype>double</datatype>
       <read>get_width</read>
       <info>"Width of rectangle."</info>
    </primitive>

    <primitive name="Height">
       <datatype>double</datatype>
       <read>get_height</read>
       <activate>activate_height</activate>
       <info>"Height of rectangle."</info>
    </primitive>

    <primitive name="Area">
       <datatype>double</datatype>
       <read>get_area</read>
       <activate>activate_area</activate>
       <info>"Area of rectangle."</info>
    </primitive>
```

```
<reference name="point_A">
   <datatype>Point</datatype>
   <read>get_point_A</read>
   <activate>activate_Point_A</activate>
</reference>

<reference name="point_B">
   <datatype>Point</datatype>
   <read>get_point_B</read>
   <activate>activate_Point_B</activate>
</reference>

<method name="compute_height">
   <returntype>void</returntype>
   <input>point_A.Y</input>
   <input>point_B.Y</input>
   <output>Height</output>
</method>

<method name="compute_area">
   <returntype>void</returntype>
   <input>Height</input>
   <input>Width</input>
   <output>Area</output>
</method>

<method name="get_width">
    <returntype>double</returntype>
```

```
        <input>point_A.X</input>
        <input>point_B.X</input>
        <output>Width</output>
      </method>

      </class> );

    Rectangle(Point* p1, Point* p2, tm_core::tmRepository* repo = 0);
    double  get_width()      { return _point_B->get_x() - _point_A->get_x(); }
    double  get_height()     { return _height; }
    double  get_area()       { return _area; }
    void    compute_height() { _height = _point_B->get_y() - _point_A->get_y();}
    void    compute_area()   { _area = _height * get_width(); }
    QString get_point_A()    { return _point_A->get_ID(); }
    QString get_point_B()    { return _point_B->get_ID(); }

  private:

    void init();

    double _height, _area;

    Point* _point_A;
    Point* _point_B;

  };
```

- Besides direct dependencies of attributes, indirect dependencies are of special interest.
- The sequence in which the attributes are updated is vital.
- Due to the continuous change of source and target data by the users of the related applications, the links between the attributes are not static.
- In a distributed environment, it is unlikely that all objects are known at a central location.
- The number of attributes is high.

Conclusion

- Standards for exchanging data and remote method invocation are available
- On their own, these standards do not solve the problems of engineering in distributed environments
- The development of methods for distributed engineering in computer networks require mathematical rigour,
  i.e. application of graph theory