Graph eXchange Language

- Overview and Current Status -

Andreas Winter

Contents

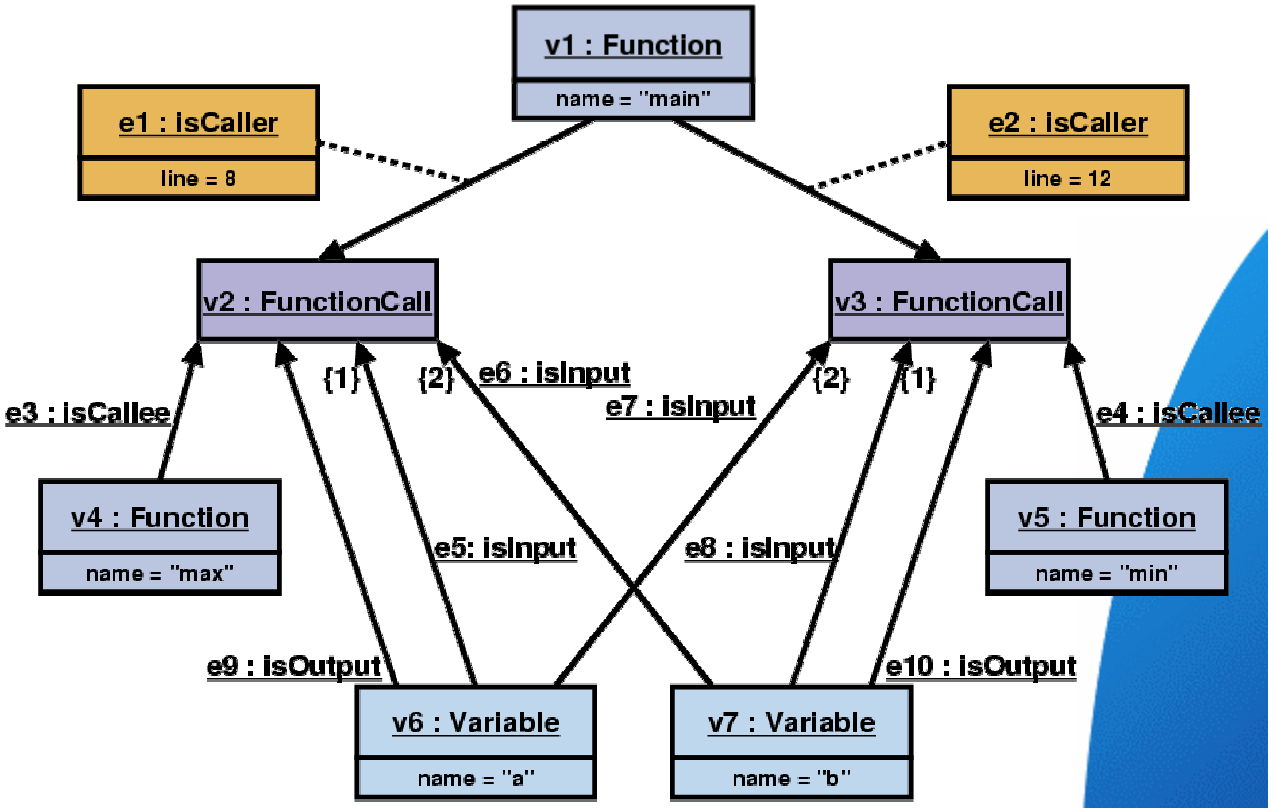
GXL - Example

myDummy.c

File Edit Search Preferences

Shell Macro Windows Help

```
 1 int main()
 2 {
 3         int a;
 4         int b;
 5
 6         ...
 7
 8         a = max(a,b);
 9
10         ...
11
12         b = min(b,a);
13
14         ...
15
16 }
```



GXL - Example

```
<?xml version = "1.0" ?>
<!DOCTYPE gxl SYSTEM "gxl.dtd" >
<gxl>
<graph id = "simpleGraph"
                edgeids = "true">
   <type xlink:href = "schema.gxl" />
   <node id = "v1" >
      <type xlink:href =
      "schema.gxl#Function"/>
      <attr name = "name" >
         <string>main</string>
      </attr>
   <node id = "v2" >
      <type xlink:href =
      "schema.gxl#FunctionCall"/>
   </node>
   <node id = "v3" >
      <type xlink:href =
      "schema.gxl#FunctionCall"/>
   </node>
   <node id = "v4" >
      <type xlink:href =
      "schema.gxl#Function"/>
      <attr name = "name" >
         <string>max</string>
      </attr>
   </node>
   <node id = "v5" >
      <type xlink:href =
      "schema.gxl#Function"/>
      <attr name = "name" >
         <string>min</string>
      </attr>
   </node>
   <node id = "v6" >
      <type xlink:href =
      "schema.gxl#Variable"/>
      <attr name = "name" >
         <string>a</string>
      </attr>
   </node>
   <node id = "v7" >
      <type xlink:href =
      "schema.gxl#Variable"/>
      <attr name = "name" >
         <string>b</string>
      </attr>
   </node>
   <edge id = "e1"
      from = "v2" to = v1"/>
      <type xlink:href =
      "schema.gxl#isCaller"/>
      <attr name = "line" >
         <int>8</int>
      </attr>
   </edge>
   <edge id = "e2"
      from = "v3" to = v1"/>
      <type xlink:href =
      "schema.gxl#isCaller"/>
      <attr name = "line" >
         <int>19</int>
      </attr>
   </edge>
   <edge id = "e3"
      from = "v4" to = v2"/>
      <type xlink:href =
      "schema.gxl#isCallee"/>
   </edge>
   <edge id = "e9"
      from = "v6" to = v2"
      <type xlink:href =
      "schema.gxl#isOutput"/>
   </edge>
   <edge id = "e5"
      from = "v6" to = v2"
      toorder = "1"/>
      <type xlink:href =
      "schema.gxl#isInput"/>
   </edge>
   <edge id = "e6"
      from = "v7" to = v2"
      toorder = "2"/>
      <type xlink:href =
      "schema.gxl#isInput"/>
   </edge>
   <edge id = "e7"
      from = "v6" to = v3"
      toorder = "2"/>
      <type xlink:href =
      "schema .gxl#isInput"/>
   </edge>
   <edge id = "e8"
      from = "v7" to = v3"
      toorder = "1"/>
      <type xlink:href =
      "schema.gxl#isInput"/>
   </edge>
   <edge id = "e9"
      from = "v6" to = v2"
      <type xlink:href =
      "schema.gxl#isOutput"/>
   </edge>
   <edge id = "e10"
      from = "v7" to = v3"
      <type xlink:href =
      "schema.gxl#isOutput"/>
   </edge>
</graph>
</gxl>
```

History

WCRE 1999
AIGra 2000
GROOM 2000

WoSEF 2000

APPLIGRAPH meeting on exchange formats for Graph Transformation 2000

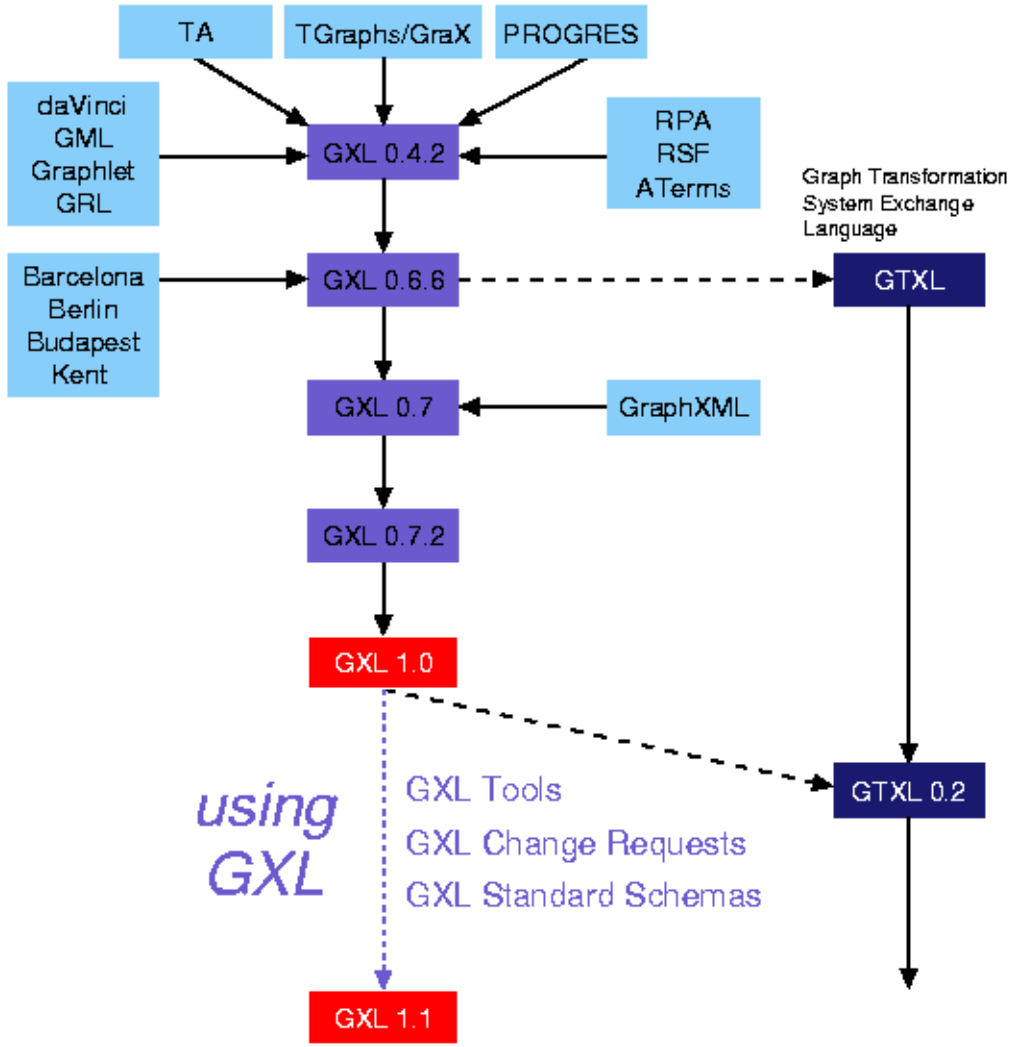
Graph Drawing 2000 workshop on data exchange formats

CASCON 2000
WCRE 2000

Dagstuhl 2001
"Interoperability of Reengineering Tools"

APPLIGRAPH meeting on exchange formats for Graph Transformation 2001

Dagstuhl 2001
"Software Vizualization"
Graph Drawing 2001
WCRE 2001



Requirements of Exchange Formats

Independence

- application independence
  - language independence
    - C/C++, Cobol, Java, JCL, SQL, multi-language, ...
  - abstraction level independence
    - AST, "middle level", Architecture, ...
  - aspect independence
    - data flow, control flow, code structure, ...
- tool independence
  - data structure independence
    - syntax trees, various types of graphs, relational databases, object oriented databases, file and directory structures, ...

Requirements of Exchange Formats

Efficiency

- efficiency in time
- efficiency in space
- efficiency in "building tools"

Extensibility

- extensible for further applications
  - (graph transformation tools, CASE tools, visualization tools, etc)

Universality

- used by others
- standardized format

GXL Objective

Exchanged Data

- instance data
- schemas data

Mathematical Model

- typed, attributed, ordered, directed graphs
- expanded by
  - hypergraphs and hierarchical graphs

Notation

- eXtensible Markup Language (XML)
- Unified Modeling Language (UML)

GXL Graph Model

graph part

typed graphs

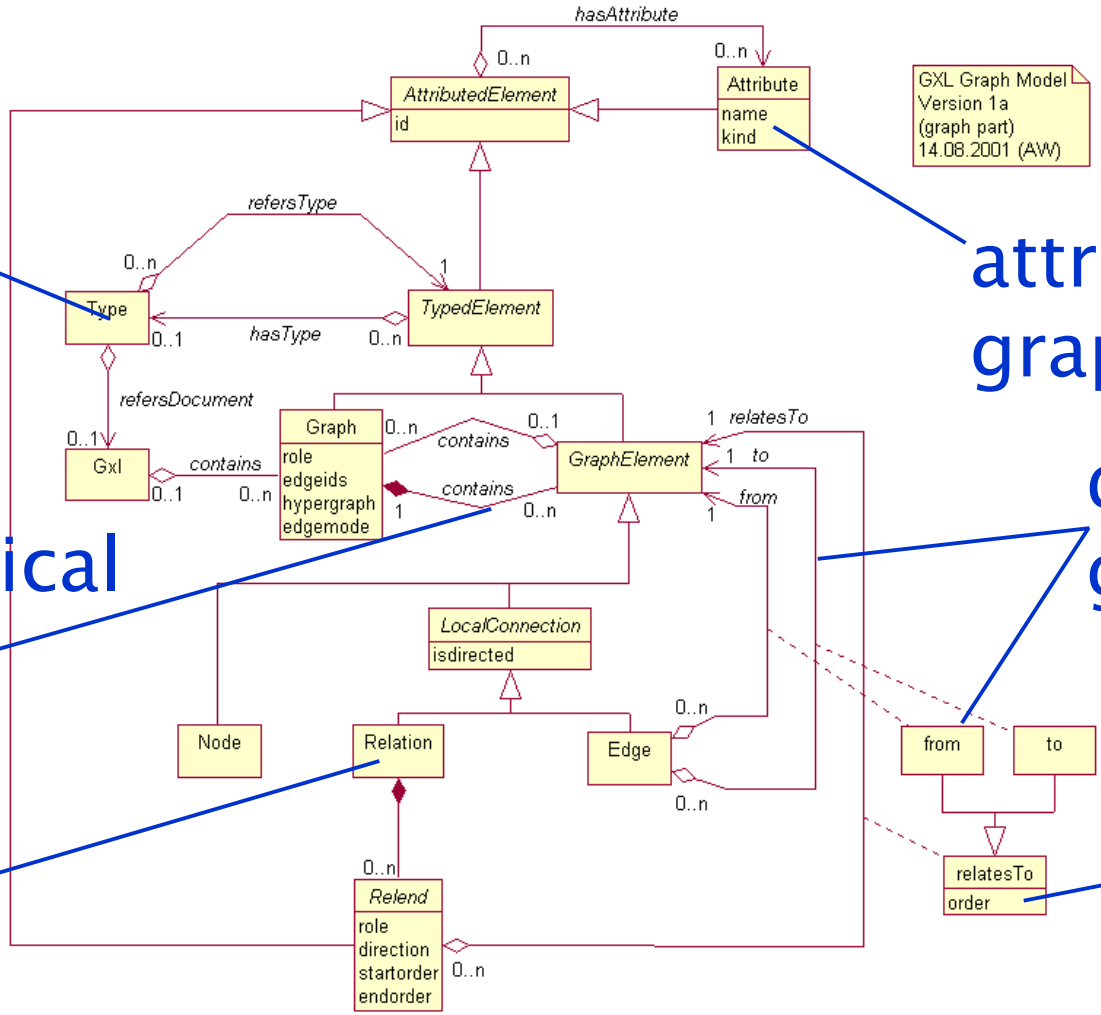
attributed graphs

hierarchical graphs

directed graphs

hyper-graphs

ordered graphs



GXL Document Type Definition (1.0)

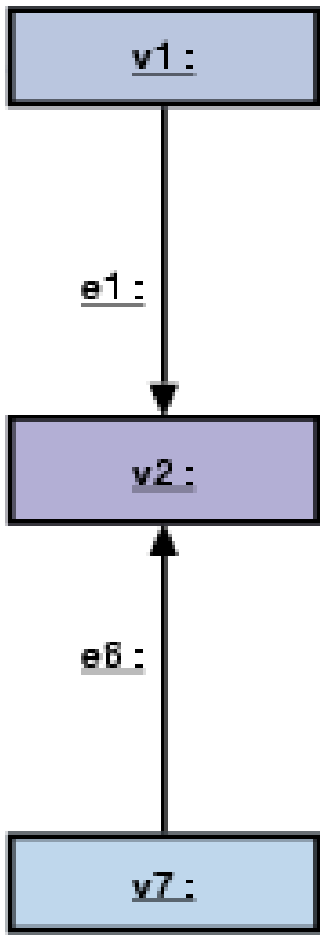
```
<!ENTITY   % *-extension   "" >
<!ENTITY   % *-attr-extension   "" >
<!ELEMENT gxl      (graph* %gxl-extension; ) >
<!ATTLIST gxl      xmlns:xlink      CDATA   #FIXED
                   "www.w3.org/1999/xlink"
                   %gxl-attr-extension>
<!ELEMENT type     EMPTY>
<!ATTLIST type     xlink:type     (simple)   #FIXED "simple"
                   xlink:href     CDATA    #REQUIRED>
<!ELEMENT graph    (type? , attr* , ( node | edge | rel )*
                   %graph-extension; ) >
<!ATTLIST graph    id             ID          #REQUIRED
                   role           NMTOKEN   #IMPLIED
                   edgeids        ( true | false ) "false"
                   hypergraph     ( true | false ) "false"
                   edgemode       ( directed | undirected |
                      defaultdirected | defaultundirected) "directed"
                   %graph-attr-extension;>
<!ELEMENT node     (type? , attr*, graph* %node-extension; ) >
<!ATTLIST node     id             ID         #REQUIRED
                   %node-attr-extension;>
<!ELEMENT edge     (type?, attr*, graph* edge-extension; ) >
<!ATTLIST edge     id             ID         #IMPLIED
                   from           IDREF      #REQUIRED
                   to             IDREF      #REQUIRED
                   fromorder      CDATA      #IMPLIED
                   toorder        CDATA      #IMPLIED
                   isdirected     ( true | false ) #IMPLIED
                   %edge-attr-extension;>
<!ELEMENT rel      (type? , attr*, graph*, relend*
                   %rel-extension;) >
<!ATTLIST rel      id             ID          #IMPLIED
                   isdirected ( true | false ) #IMPLIED
                   %rel-attr-extension;>
<!ELEMENT relend (attr* %relend-extension; ) >
<!ATTLIST relend target       IDREF        #REQUIRED
                   role           NMTOKEN   #IMPLIED
                   direction    ( in | out | none )  #IMPLIED
                   startorder   CDATA      #IMPLIED
                   endorder     CDATA      #IMPLIED
                  %relend-attr-extension; >
<!ELEMENT attr     (type?, attr*, (%val;)) >
<!ATTLIST attr     id             IDREF        #IMPLIED
                   name         NMTOKEN   #REQUIRED
                   kind         NMTOKEN   #IMPLIED >
<!ENTITY   % val " locator | bool | int | float | string | enum |
                   seq | set | bag |tup %value-extension; ">
<!ELEMENT locator  EMPTY >
<!ATTLIST locator  xlink:type  (simple)   #FIXED  "simple"
                      xlink:href  CDATA      #IMPLIED >
<!ELEMENT bool |int | float | string   (#PCDATA) >
<!ELEMENT enum     (#PCDATA) >
<!ELEMENT seq | set | bag | tup  (%val;)* >
```

Exchanging Graphs with GXL

Graph types supported by GXL:

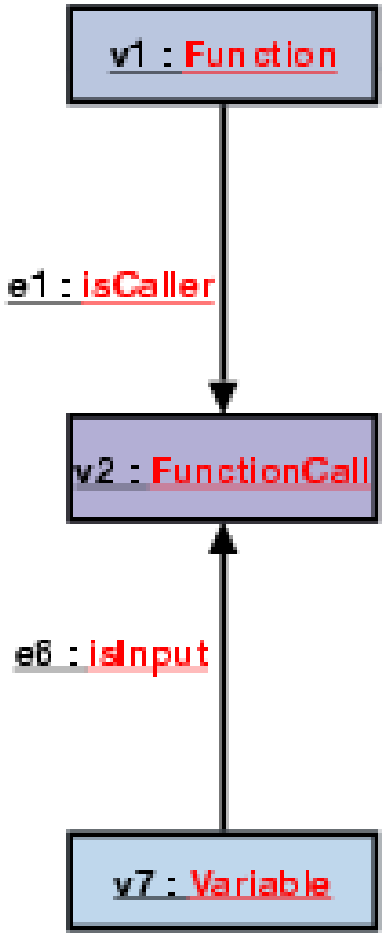
- Typed Graphs
- Attributed Graphs
- Directed Graphs
- Undirected Graphs
- Ordered Graphs
- Hypergraphs
- Hierarchical Graphs
- ...

Directed Graphs




```
<node id = "v1">

                                    <edge id = "e1"
                                          from = "v1"
                                          to = "v2">

</node>

<node id = "v2">

                                    </edge>

</node>
                                    <edge id = "e6
<node id = "v7">                          from = "v7"
                                          to = "v2">

                                    </edge>

</node>
```

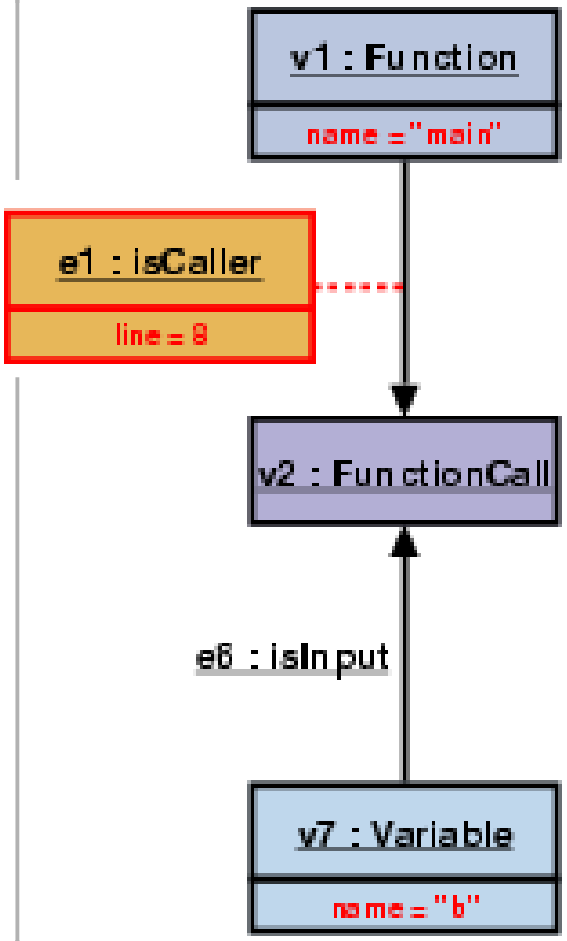
Directed Typed Graphs




```
<node id = "v1">
  <type xlink:href =s.gxl
        #Function" />


  </attr>
</node>

<node id = "v2">
  <type xlink:href ="s.gxl
        #FunctionCall" />
</node>

<node id = "v7">
  <type xlink:href ="s.gxl#
        Variable" />


  </attr>
</node>
```

```
<edge id = "e1"
      from = "v1"
      to = "v2">
  <type xlink:href =s.gxl
        #isCaller" />


  </attr>
</edge>

<edge id = "e6
      from = "v7"
      to = "v2">

  <type xlink:href =s.gxl
        #isInput" />
</edge>
```

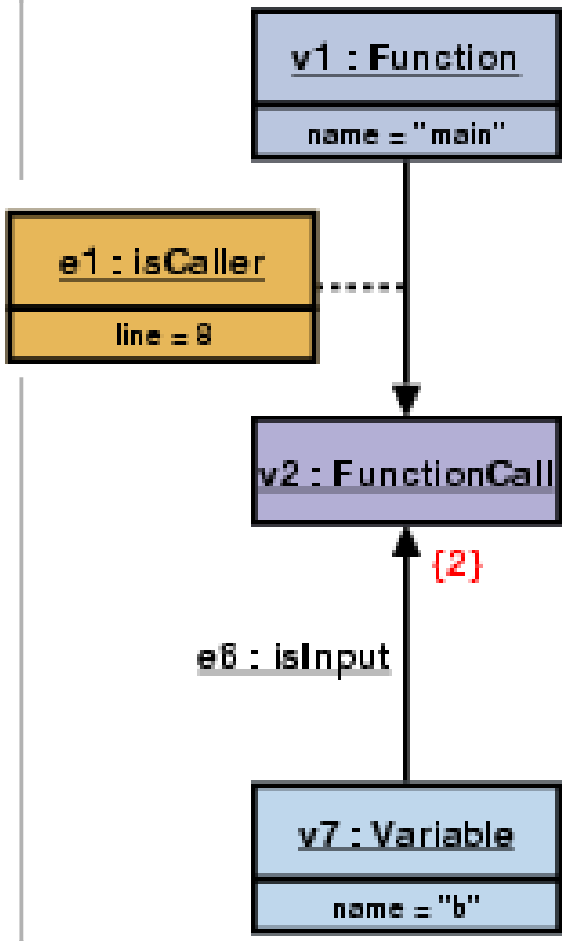
Typed, Attributed, Directed Graphs




```
<node id = "v1">
  <type xlink:href =s.gxl
        #Function" />
  <attr name = "name">
        <string>main</string>
  </attr>
</node>

<node id = "v2">
  <type xlink:href ="s.gxl
        #FunctionCall" />
</node>

<node id = "v7">
  <type xlink:href ="s.gxl#
        Variable" />
  <attr name = "name">
        <string>b</string>
  </attr>
</node>
```

```
<edge id = "e1"
      from = "v1"
      to = "v2">
  <type xlink:href =s.gxl
        #isCaller" />
  <attr name = "line">
        <int>8</int>
  </attr>
</edge>

<edge id = "e6
      from = "v7"
      to = "v2">

  <type xlink:href =s.gxl
        #isInput" />
</edge>
```

Typed, Attributed, Directed, Ordered Graphs




```
<node id = "v1">
  <type xlink:href =s.gxl
        #Function" />
  <attr name = "name">
        <string>main</string>
  </attr>
</node>

<node id = "v2">
  <type xlink:href ="s.gxl
        #FunctionCall" />
</node>

<node id = "v7">
  <type xlink:href ="s.gxl#
        Variable" />
  <attr name = "name">
        <string>b</string>
  </attr>
</node>
```

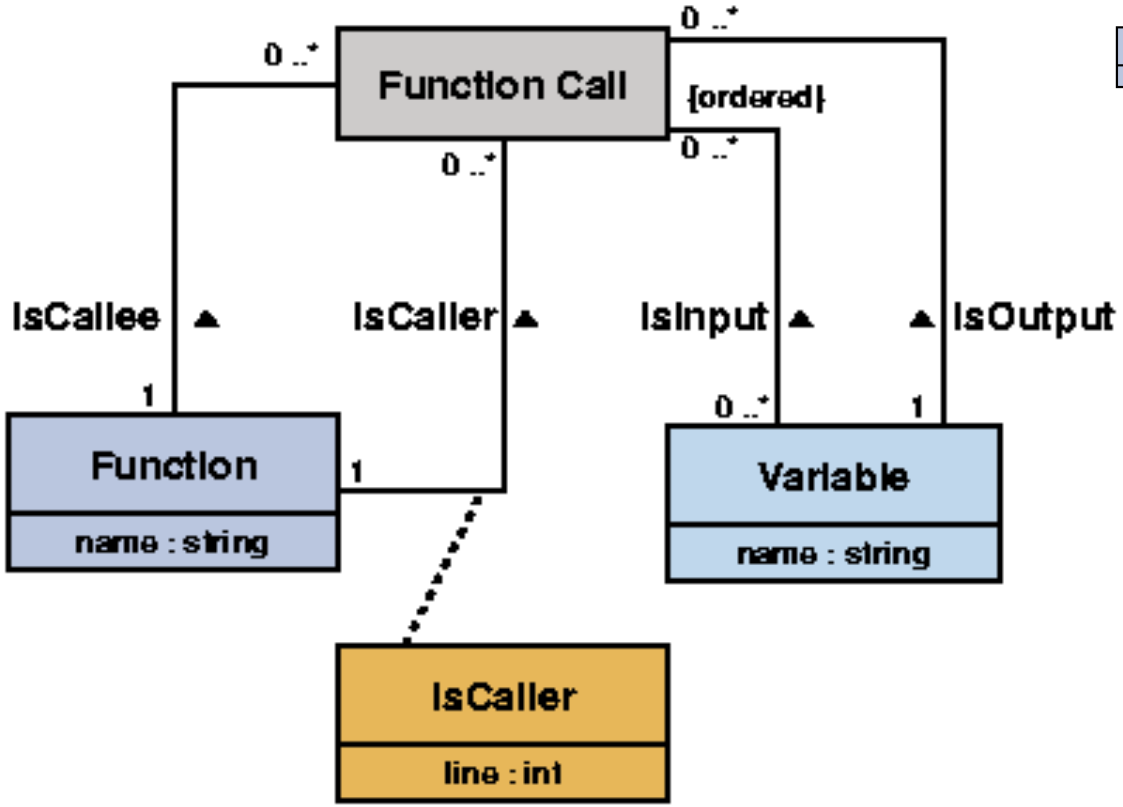
```
<edge id = "e1"
      from = "v1"
      to = "v2">
  <type xlink:href =s.gxl
        #isCaller" />
  <attr name = "line">
        <int>8</int>
  </attr>
</edge>

<edge id = "e6
      from = "v7"
      to = "v2"
      toorder = "2">
  <type xlink:href =s.gxl
        #isInput" />
</edge>
```

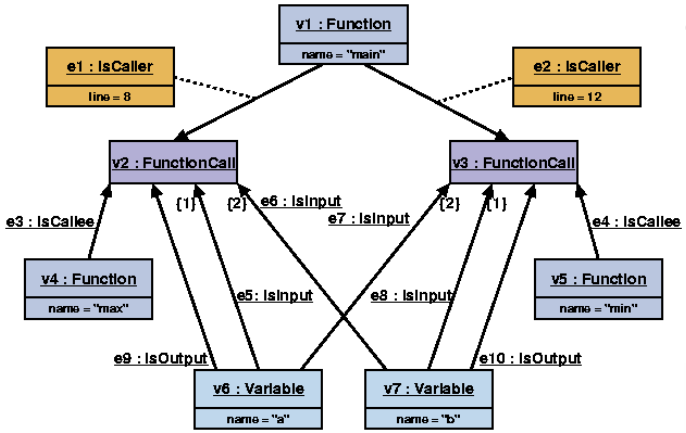
Exchanging Schemas with GXL

GXL Approach:

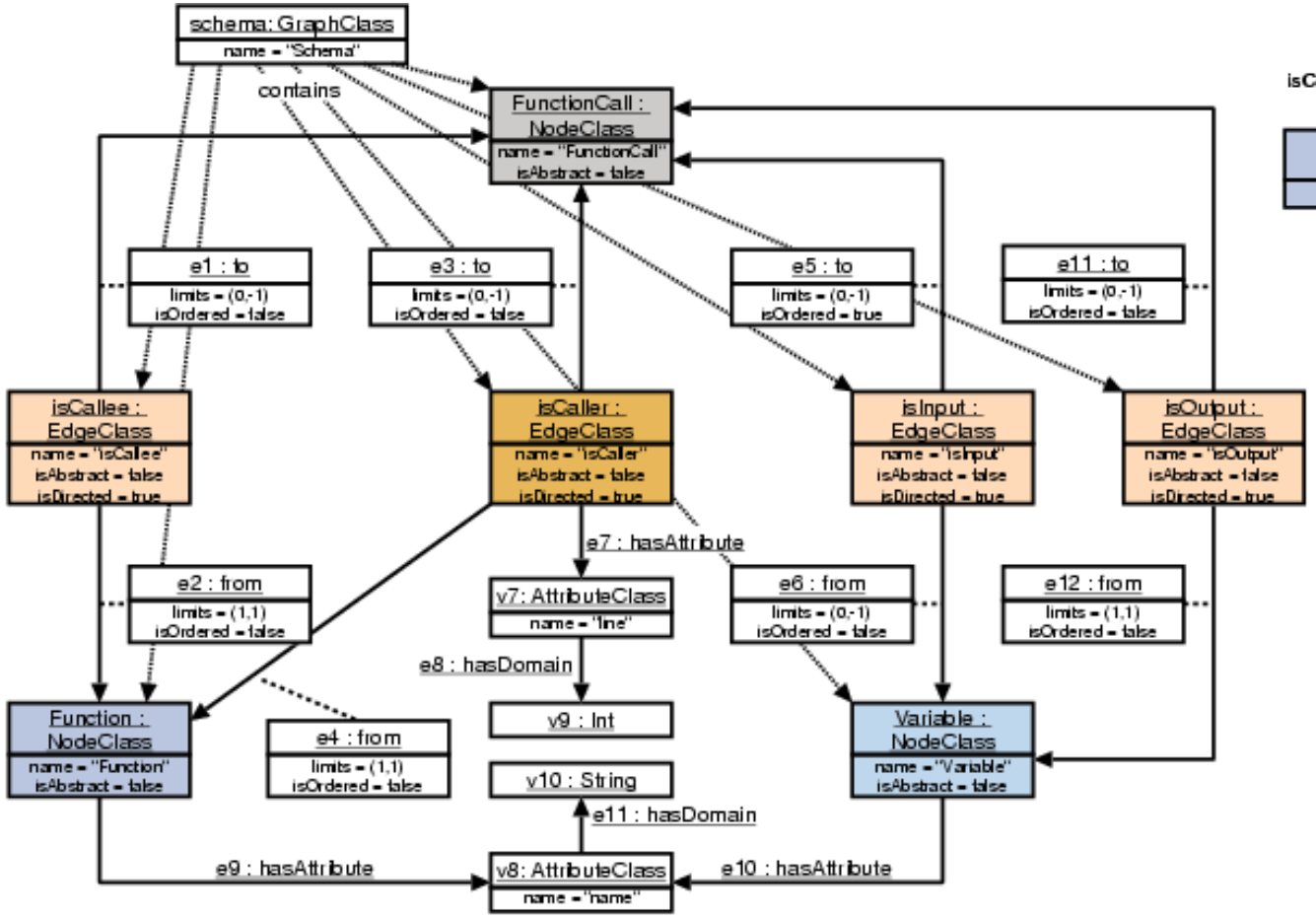
- Defining Graph-Structured (Graph Schemas) by UML Class Diagrams
- Representing UML Class Diagrams by Graphs (Schema Graphs)
- Exchanging Schemas Graphs by GXL Documents

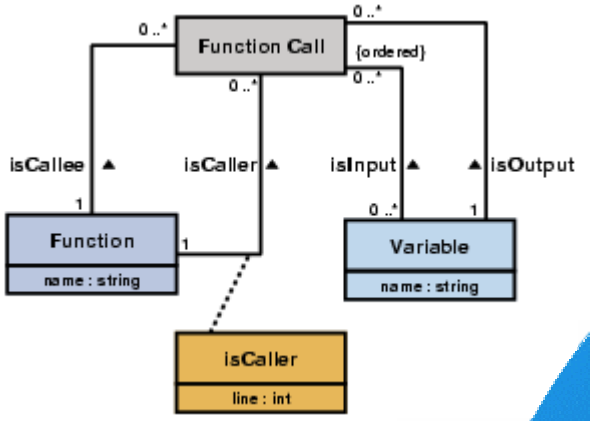
Graph Schemas



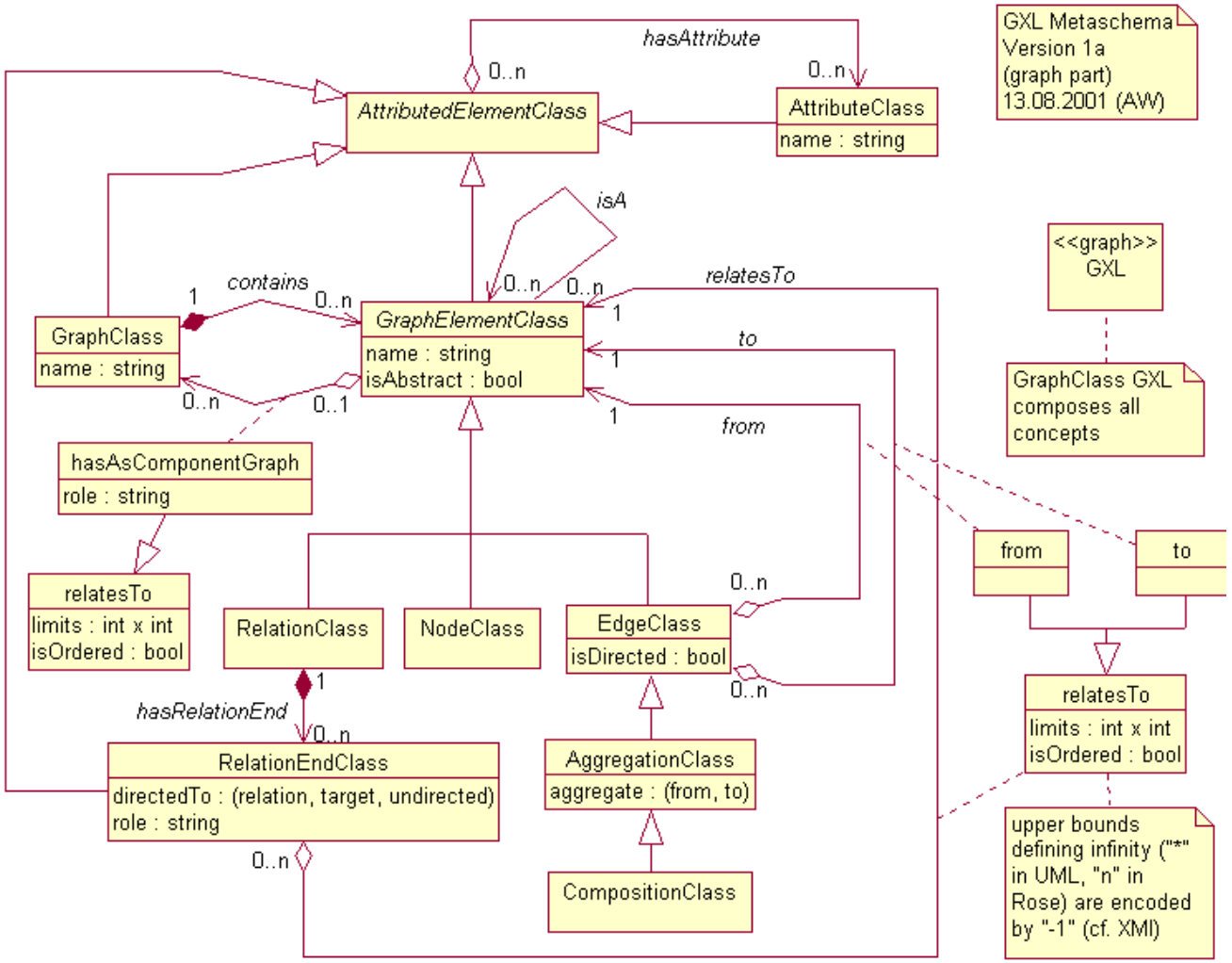


Schema Graphs





GXL Metaschema (Graph Part)



GXL Metaschema - GXL Document

```
<?xml version="1.0"?>
<!DOCTYPE gxl SYSTEM "gxl.dtd">
<gxl>
  <graph id = "gxl">
  <type xlink:href ="gxl.gxl#gxl"/>

<node id = "NodeClass">
  <type xlink:href =
       "gxl.gxl#NodeClass"/>
  <attr name = "name">
    <string>NodeClass</string>
  </attr>
  <attr name = "isAbstract">
    <bool>false</bool>
  </attr>
</node>

...
```

```
<node id = "GraphClassElement">
  <type xlink:href =
       "gxl.gxl#NodeClass"/>
  <attr name = "name">
    <string>GraphClassElement</string>
  </attr>
  <attr name = "isAbstract">
    <bool>true</bool>
  </attr>
</node>
...
<edge
  from = "NodeClass"
  to   = "GraphClassElement">
  <type xlink:href =
       "gxl.gxl#isA"/>
</edge>
...
</gxl>
```

GXL - Current Status

GXL Usage

- Software Reengineering
- Graph Transformation Systems (GTXL)

GXL Current Activities

- Development of GXL Tools
- Definition of GXL Reference Schemas for Reverse-Engineering
- Collection of Change Requests for GXL 1.1

GXL Mailinglist

- 87 Members

GXL Tools and Applications

Filter/Converter

- Bauhaus Resource Graphs (Univ. Stuttgart)
- FAMIX (Nokia)
- GraLab (Univ. Koblenz)
- Progres (RWTH Aachen)
- RPA (Philips, Eindhoven)
- RSF (Univ. Victoria)
- TA (Univ. Waterloo)
- XMI (Univ. BW München)

Reverse Engineering Tools

- Bauhaus (Univ. Stuttgart)
- Columbus (Univ. Szeged)
- CPPX (Univ. Kingston, Univ. Waterloo)
- GUPRO (Univ. Koblenz)
- Missing Link (Merlin, Karlsruhe)
- Rigi (Univ. Victoria)
- PBS/Swagkit (Univ. Waterloo)
- TKSee/SN (Univ. Ottawa)
- Venice (Univ. Helsinki, Nokia)

GXL Tools and Applications

Graph Transformation Systems

- AGG (TU Berlin)
- DiaGen (Univ. Erlangen)
- Fujaba (Univ. Paderborn)
- Genset (Univ. Oregon)
- Progres (RWTH Aachen)

Software Engineering Tools

- Chi-Bel (Univ. Toronto)
- Edinbourgh Concurrency Workbench (Univ. Edinbourgh)
- Upgrade (RWTH Aachen)

Graph Visualizer

- Graph Tool (Univ. Durham)
- Shrimp (Univ. Victoria)
- Touch Graph (Alex Shapiro)
- yFiles (Univ. Tübingen)
- JGraph (Zürich)

Graph Databases

- Gras (RWTH Aachen)

GXL Reference Schemas

C++ Reference Schema

- *Rudi Ferenc*, Tibor Gyimothy (Univ. Szeged)
- Ric Holt, *Andrew Malton* (Univ. Waterloo)

Dagstuhl Middle Level Model

- *Tim Lethbridge*, Sergei Marchenko (Univ. Ottawa)
- Panos Linos (Butler Univ. Indianapolis)
- Erhard Plödereder (Univ. Stuttgart)
- Sander Tichelar (Univ. Bern)

Data Reverse Engineering Reference-Schema

- *Jens Jahnke* (Univ. Victoria)
- Jean Luc Hainault, Jean Henrard (Univ. Namur)
- John Mylopoulos (Univ. Toronto)
- Jörg Wadsack (Univ. Paderborn)

GXL Change Requests

XML Schema
- more powerful schemas

Packages
- attributes - structure (GXL) - GTXL - GraphML ...

Hierarchical Graphs
- graph valued attributes
- references to graphs

Additional Attribute Types
- free type attributes
- structs

References to GXL Schemas
- textual attributes instead of XLINK-references

Namespaces
- better usage of GXL within other XML documents

Superfluous Aspects
- schema related data in instance graphs
- attributes of attributes

Document Information
- header information about GXL-Version, used tools, etc.

Additional features
- implicit nodes, defined by by edges
- classification of attributes (derived,frozen)
- explicitly defined ordering of graph elements
- support for partial graphs
- support to express undefined values and infinity

Next Steps

August 1, 2002

- Deadline for GXL Change Request

December, 2002

- Release of GXL 1.1 *alpha*

January, 2003

- Request for Comments for GXL 1.1 *alpha*

February 14, 2003

- GXL 2nd Birthday
- Release of GXL 1.1

Conclusion

GXL: uniform language for

- exchanging graphs
- exchanging graph schemas

GXL: ratified as standard exchange format

- Software Reengineering community (Dagstuhl, January 26. 2001)
- Usage within GTXL Graph Transformationsystems Exchange Language (APPLIGRAPH Subgroup Meeting, Bremen, March, 1.-2., 2001)

more information

- http: `www.gupro.de/GXL`
- mailto: `gxl@uni-koblenz.de`