

## Chapter 8

# DyNetML: A Robust Interchange Language for Rich Social Network Data

Current state of the art in social network data representation presents a fairly bleak picture. Each of analysis and simulation packages uses its own, proprietary and incompatible data format. Some of the file formats do not even have a specification document, making the files unreadable without the software that produced them.

Data formats that were designed for interoperability (such as DL) are rarely expressive enough to fully represent the datasets.

As a result, most researchers are forced to deal with data interchange in a makeshift fashion, at best increasing the workload and at worst resulting in loss of data integrity.

## 8.1 Requirements for Social Network Data Interchange

1. The data interchange format shall be contained in human-readable text files that are at the same time easily parsable by computers.
2. The data interchange format shall allow an entire dataset, complete with all computed measurements, to be stored in one file.
3. The data interchange format shall provide maximum expressive power to its users, allowing:
  - Typed nodes (types may include “person”, “resource”, “organization”, “knowledge”, etc)
  - Multiple sets of nodes of the same type (to express multiple units within the company, etc)
  - Multiple typed attributes per node
  - Typed edges
  - Multiple typed attributes per edge

- Multiple graphs (sets of edges) expressed within the same file
  - Dynamic network data expressed in a single file
4. The data interchange format shall allow developers to extend it in a fashion that will not break existing software.
  5. The data interchange format shall be flexible enough to be used as both input and output of analysis tools.

Current social network data formats have a number of deficiencies:

**Binary files** are very difficult to read if exact specification of the file format is not provided. Significant extra efforts are required to keep compatibility with other tools or between versions of the same tool.

**Multiple files** used for specification of rich data or saving analysis output present a number of problems. First of all, there is a significant potential for data loss due to misplaced or corrupted files (for example, while sent through email). Secondly, a consistent naming scheme for all files and a file catalogue are required to prevent data loss - which requires a certain amount of discipline on the part of the researcher (as these features are not included in the analysis software)

**Raw Data** file such as binary matrices or edge lists lack the expressiveness required to represent multiple relations between nodes or evolution of social networks over time.

**Human-Readable Data** in text files or spreadsheets solves the expressivity problem but requires extensive post-processing by hand or with post-processing scripts. However, these programs often represent the weakest link in the software chain (due to hasty design and dependance on outside tools such as Perl or Awk).

**General-purpose Graph XML formats** such as GraphXML or GXL provide a good approximation of fulfilling the requirements for a data interchange format. They are both human- and machine-readable, and offer extensible capabilities to allow some amount of customization. However, each of the general-purpose formats in existence has a number of individual drawbacks that make it difficult to adapt it for use with social network data.

Table 8.1 shows a comparison of a number of existing formats, both plain-text and XML, in terms of the features required for an expressive interchange format for social network data.

### 8.1.1 Existing Data formats

The *DL* format supported by UCINET[S.P. Borgatti and Freeman, 2002] is a flexible and human readable data format, and it can contain multiple matrices in a single file. However, the matrices in a DL file must be of a single type.

Format	Parseability	Multi-Mode	Dynamic Graph	Node Attributes	Edge Attributes	Graph-Level Attributes	Attribute Typing
UCINET Native	Proprietary, poorly documented	No	No	In separate files	No	No	Numbers
UCINET DL	Ad-Hoc	No	No	No	Single value	No	Numbers
UCINET VNA	Ad-Hoc	No	No	Extensible	Single value	No	Numbers
PAJEK NET	Rigorous Grammar	No	No	Extensible	Extensible	No	Weak
GraphML	XML	No	Delta	Fixed Set, Graph Drawing	Fixed Set, Graph Drawing	No	Fixed
GraphXML	XML	No	Delta	Extensible	Extensible	No	Weak String only
GXL	XML	Fixed Node and Edge Types	No	Extensible	Extensible	No	Strong
ODL	XML	Fixed Node Types, Untyped Edges	No	Extensible	Extensible	No	Strong
DyNetML	XML	Extensible Node and Edge Types	Delta or Full Graph	Extensible	Extensible	Extensible	Strong

Table 8.1: Comparison of Social Network Data Formats

For example, a single DL file cannot contain one matrix containing Agent by Agent data, and another with Agent by Knowledge data. Thus to represent an entire Meta-Matrix, multiple DL files must be used, which increases the likelihood of data inconsistencies.

However, one of the largest drawbacks of DL format is the fact that it is defined in an ad-hoc fashion and lacks a stable grammar. This results in subtle and difficult to detect incompatibilities between tools that use DL as an interchange format.

Moreover, DL files make it very difficult to communicate rich social network data as DL matrices do not support attributes embedded within the data files. While it is possible to communicate dynamic social network data in DL, its handling is somewhat arcane and unstable.

PAJEK[Batagelj and Mrvar, 2003] comes closest to defining a universal interchange format for social network data. PAJEK *.net* format is defined using a rigorous and stable grammar and allows for arbitrary rich data in both nodes and edges.

However, PAJEK files do not have a ready facility for expressing dynamic network data, or for communicating multi-mode and multi-plex networks.

Thus, both of the dominant data formats for social network software have a number of obvious drawbacks that make them unsuitable for communicating large amounts of rich data.

### 8.1.2 XML-Based Graph Data Formats

The first issue that I address in assessing alternative data formats is that of a rigorous yet expressive grammar. A further requirement is that the new format must be easily implementable for support in various software products.

The XML[XML.org, a] language provides powerful facilities for building expressive and stable data formats. Through use of DTDs (Document Type Definitions) and XML Schemata, it is possible to create a data format that would be both easy to implement and use and provide ample expressive power.

The nature of rich social network data is that it combines traditional attribute based datasets with relational, or graph-based data.

A number of data formats for managing graph-based data, including GraphXML[Herman and Marshall, 2000], GraphML[Brandes, Eiglsperger, Herman, Himsolt, and Marshall, 2002] and GXL[Holt, Winter, and Schürr, 2000], propose XML-based languages for treating graph and relational data.

GraphXML[Herman and Marshall, 2000] is an early XML-based graph language designed to fill the need of graph interchange languages in the graph drawing community. It is essentially a single-purpose language that communicates the graph structure and semi-fixed sets of attributes (such as color, shape and location of nodes). GraphXML is the least flexible of the currently available languages and has been largely superseded by GraphML.

GraphML[Brandes, Eiglsperger, Herman, Himsolt, and Marshall, 2002] is an XML format for generalized graph structures. Its characteristic feature is ability to add modules that implement specific extensions or additional data, such as information related to graph drawing. These modules can be combined or stripped without altering the underlying graph structure, which affords GraphML a large degree of flexibility. GraphML is also the only published format that supports manipulation of dynamic graph data.

The main disadvantage of GraphML is that it requires significant modifications in order to support multi-mode and multi-plex data. Due to its orientation towards the graph drawing community, most of the extensions written to the day are related to graphical representation of graphs and do not contribute to GraphML's ability to express social network data.

Additional advantage of GraphML is that it is supported in large number of graph analysis and drawing software, including yFiles products[yWorks], Ashwood libraries[ObjectStyle, 2005] and in future version of PAJEK[Batagelj and Mrvar, 2003]

GXL[Holt, Winter, and Schürr, 2000][Taentzer, 2001][Winter, 2001] was developed to enable interoperability between software reengineering tools and components. GXL facilitates process of combining single-purpose tools for parsing, source code extraction, architecture recovery, data flow analysis,

pointer analysis, program slicing, query techniques, source code visualization, object recovery, restructuring, refactoring into a single reengineering workbench.

The conceptual data model of GXL is a typed, attributed, directed graph. It can also be used to represent instance data as well as schemas for describing the structure of the data. Moreover, the schema can be explicitly stated along with instance data.

GXL comes very close to fulfilling requirements for social network data interchange format. However, the advantage of enforcing strong typing and directionality of the graph edges comes with a drawback of forcing a particular model of social structure into the data - as many social network measures are not defined on directed graphs.

### 8.1.3 XML Data Formats for Social Network Data

ODL[Stacy, 2004] (Organizational Design Language) is an XML-derivative language for representing organizational structures developed by Aptima. ODL was designed approximately at the same time as DyNetML, and answers to a very similar set of requirements. ODL represents rich social network data as a collection of typed nodesets, and a set of graphs connecting these nodes with edges.

A characteristic feature of ODL is its use of bindings to represent unknown or probabilistic edges. A binding element acts as a placeholder for zero or more concrete elements. For example, if the sender of a message is known, but the recipient is not, the recipient of the message can be represented by an empty binding element. As a list of probable recipients emerges, they can be added to the binding element complete with their probability of being to the communication.

While ODL is designed for a purpose that is very similar to that of DyNetML, it possesses a number of shortcomings that limit its use for analysis of rich dynamic networks. First of all, ODL does not have a ready representation of dynamic data. Dynamic representations can be achieved by including multiple graphs, each representing a time period, but that technique does not provide a rigorous encapsulation of a multi-mode network changing over time.

A further shortcoming is due to the fact that ODL operates with a fixed set of concept types (Agent, Knowledge, Resource, Task, Event, Communication, Location and Organization). These concept types are not readily extensible, and the methods for handling them are not consistent between types. This places an undue burden on the developer and maintainer of the tools and may cause inconsistencies in data interpretation if ODL was used for data interchange between various software tools.

While use of ODL as an interchange standard may be difficult (both due

to its design and due to the fact that the language is proprietary), translation between ODL and DyNetML can be easily accomplished using XSLT transform mechanism. Such translator was designed by Aptima and is used to bridge their simulation tools to ORA[Carley and Reminga, 2004] and other tools developed at the CASOS lab.

#### 8.1.4 Advantages and Disadvantages of XML-based Data Interchange

The main advantage of expressing complex data as XML is the ability of XML to import hierarchical and object-oriented structures into a human-readable text file. This solves a major representational challenge of other text-based data formats, which are more suitable for representation of flat attribute tables or matrix-based structures.

The representation of complex data structures enabled by XML would be a much lesser advantage if XML did not provide facilities for specifying the schemata for data contained in XML files. However, both DTD and XML Schema facilitate creation of strong grammars that dictate file contents.

The combination of a rigorous markup grammar and data schema facilities reduces the error rate in creation and parsing of XML documents, and provides significant improvement in structure and implementation of third-party code for use with the data format.

Another major advantage of all XML formats is that the rigorous definition of XML grammars allows for easy transformation between various graph formats. Brandes and Lerner[Brandes, Lerner, and Pich, 2004] demonstrate this by applying an XSLT[Wadler, 1999] transformation to convert between GXL and GraphML formats described above.

An XSLT conversion also exists between the DyNetML format described below and the ODL format developed at Aptima. **TODO: MORE HERE**

The major drawback of XML-derived data formats stems from the size and complexity of XML files. The growth in size is dictated by needs for rigorous markup, as every data element requires a number of delimiter tags to describe its function to the parser. While XML files get quite large, they compress very well with any of the currently available algorithms, resulting in average of 90% compression rates.

Furthermore, the memory requirements of XML (and especially DOM[XML.org, b]) parsers are significantly higher than these for parsing simple text-based formats. Use of SAX[XML.org, c] parsers to process large XML files provides significant memory and processing time savings, but adds burden on the software developer as SAX parsers are significantly more difficult to use.

## 8.2 Extensibility and Modularity of XML Data Formats

A major portion of discussion regarding the future of XML-based data formats for social network data is centered around extensibility of the format for use in and inter-operation of a variety of applications.

A data format specification designed as a common standard, pursues a number of conflicting goals[Stacy, 2004]. The first goal is providing a common language for all tools to be integrated. This goal argues for a least common denominator approach, that is, for a subset of network organizational representations shared by all current and future integrated applications.

The second goal is to provide a comprehensive means for each tool to represent all the data it requires, whether or not any other tool can use that data. This goal argues for a least common multiple approach, that is, for a large set of representations that cover the needs of all integrated applications.

The approach DyNetML takes is a hybrid of the two approaches outlined above. First, DyNetML provides a mechanism for specifying an arbitrary number of typed, named properties at node, edge, node-type, graph, and dataset level. This mechanism is generic and does not violate the overall graph-oriented structure of the data, and thus can be handled by existing tools in a backwards-compatible fashion. This extensibility mechanism serves well for adding custom data points within the graph (e.g. specifying the probability density function of an edge; specifying demographic information on a node).

If the custom data does not fit with the existing graph model (e.g. Anthropac questionnaire data), DyNetML allows for modular extension using auxiliary schemata. This modularity specification follows the W3C XML Schema recommendations[Consortium, 2004]. A developer extending DyNetML with modular data must include the specification of an extension module with the XSD schema of DyNetML and specify its role in the DyNetML structure via XSD entities.

## 8.3 DyNetML: A Data Interchange For Rich Social Network Data

To enable interchange and transmittal of rich social structure data, I have designed DyNetML, an XML-based language that fits the requirements outlined above. While DyNetML is still a work in progress, it is slowly gaining industry acceptance as a data interchange format. It is supported by all tools developed at the CASOS laboratory at Carnegie Mellon University, and is a part of ORA, a MetaMatrix-based organizational network analysis tool.

```

<node id="Mohammed_Rashed_Daoud_al-0whali">
  <properties>
    <property name="knowledgeAccumulated" type="double" value="
      0.7500" />
    <property name="placeOfBirth" type="string" value="Kabul,␣
      Afganistan" />
  </properties>
</node>

<edge source="Mohammed_Rashed_Daoud_al-0whali" target="
  Usama_Bin_Ladin"
type="double" value="1.000000" />
  <properties>
    <property name="observedOn" type="date" value="01/01/2000" /
    >
    <property name="edgeProbability" type="double" value="0.12"
    />
    <property name="edgeStrength" type="double" value="0.12" />
  </properties>

```

Figure 8.1: Examples of custom properties for a node and an edge

As of August 2004, DyNetML will be natively supported by UCINET, the premier software package for social network analysis. Native support is also under development for a number of software tools at the Department of Defence. Through the use of translation tools, DyNetML is also used by Aptima, University of Connecticut and a number of other companies and institutions.

## 8.4 DyNetML: an XML-Derived Social Network Language

To address the needs of data interchange and requirements outlined in section 8.1, we have designed DyNetML: an XML-derived language for expressing rich social network data.

The following example illustrates use of DyNetML for representing simple social network datasets (also illustrated on figure 8.2). This and further examples are derived from the Tanzania Embassy Bombing dataset.

### 8.4.1 DyNetML Format Overview

DyNetML represents dynamic network data as sets of time-slices. Each of the time-slices is a descriptive snapshot of the organization at a given time.

Listing 8.1: A simple network in DyNetML

```

<?xml version = "1.0" encoding = "UTF-8"?>
<DynamicNetwork
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation = "DyNetML.xsd">
<MetaMatrix>
<nodes>
  <nodeset id="agent" type="agent">
    <node id="Mohammed_Rashed_Daoud_al-0whali"/>
    <node id="Mohammed_Sadiq_Odeh"/>
    <node id="Fazul_Abdullah_Mohammed"/>
    <node id="Wadih_al_Hage"/>
    <node id="Usama_Bin_Ladin"/>
    <node id="Ali_Mohammed"/>
  </nodeset>
</nodes>
<networks>
  <graph id="social_network" sourceType="agent" targetType="
    agent" isDirected="true">
    <edge source="Mohammed_Rashed_Daoud_al-0whali" target="
      Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Mohammed_Sadiq_Odeh" target="Wadih_al_Hage"
      type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Wadih_al_Hage
      " type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="
      Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Mohammed_Sadiq_Odeh"
      type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Fazul_Abdullah_Mohammed
      " type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="
      Mohammed_Rashed_Daoud_al-0whali" type="double" value="1"/
      >
    <edge source="Usama_Bin_Ladin" target="
      Fazul_Abdullah_Mohammed" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Wadih_al_Hage" type="
      double" value="1"/>
    <edge source="Ali_Mohammed" target="Wadih_al_Hage" type="
      double" value="1"/>
    <edge source="Ali_Mohammed" target="Usama_Bin_Ladin" type="
      double" value="1"/>
  </graph>
</networks>
</MetaMatrix>
</DynamicNetwork>

```

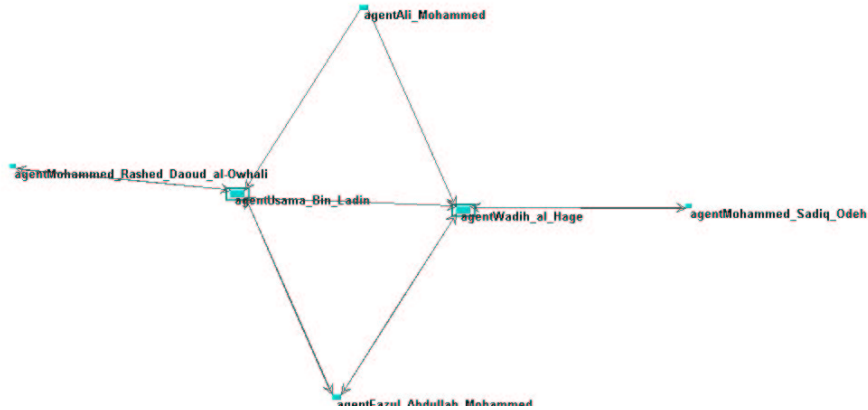


Figure 8.2: A simple network in DyNetML

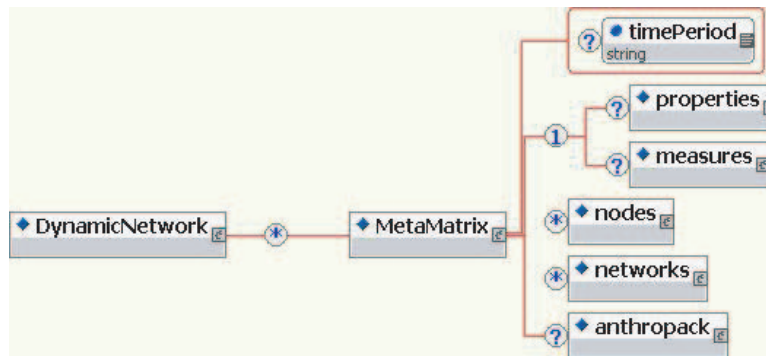


Figure 8.3: Dynamic Networks in DyNetML

Figure 8.3 shows the top-level hierarchy of DyNetML files. A **Dynamic-Network** element is defined as a sequence of **MetaMatrix** elements, each representing a snapshot of the organization for one time period.

Each of the **MetaMatrix** elements consists of:

- an optional **TimePeriod** attribute that allows clear identification of each timeslice.
- a set of **properties** and **measures**, representing data about the whole of the timeslice (see section 8.7 for a complete definition).
- a **nodes** element, containing one or more nodesets (section 8.5.1).
- a **networks** element, containing all networks in this timeslice (section 8.6).
- an **anthropack** element that facilitates linking of network data to anthropological data.

## 8.5 Representing Multiple Node and Relation Types

While designed predominantly for dealing with social network data, DyNetML format is shaped as a generalized graph data interchange framework.

DyNetML represents graphs as sets of nodes (vertices) and relationships (edges) between them. The node specification allows for detailed specification of each vertex, as well as addition of rich data related to it.

### 8.5.1 Specifying Individuals and Nodes

Nodes are organized into **nodesets**, which should be thought about as logical groupings of nodes (by type, by affiliation, etc).

Each nodeset has to be identified with a unique **id** attribute (see figure 8.4), and a **type** attribute. For more detailed description of node types, see section 8.5.1.

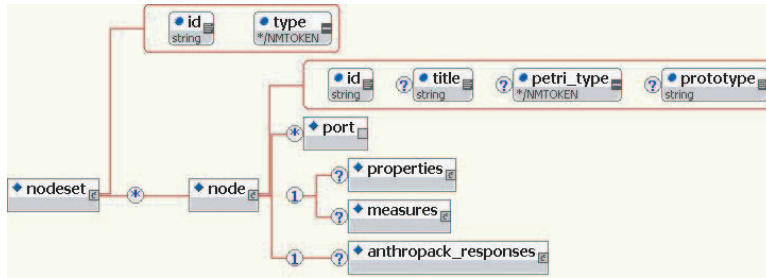
DyNetML allows for an arbitrary number of nodesets (and arbitrary number of nodesets of each type) and an arbitrary number of nodes in each nodeset.

A node specification consists of the following (see figure 8.4):

- **id**: a unique ID *note: it is advisable for ease of searching to use node IDs that do not contain spaces or special characters and are limited in length to 32 characters.*
- **title**: a human-readable title of the vertex (free of restrictions posed on node ID field).
- **prototype**: an optional attribute specifying a parent class of a node. Node prototype can be used to specify additional details about the node.
- Element **port** allows the user to specify inflows and outflows of each node by allowing multiple connection points within each node.
- **Properties** and **Measures** elements allow specification of arbitrary rich data for each node. They are described in more detail in section 8.7.
- **Anthropac** element provides a vehicle for connecting anthropological data with social network data.

### Ports and Multiple Connection Points

In order to implement multiple types of connections within the same graph, and to enable use of graphs as nodes of other graphs, we have implemented a system of ports.



```

<node id="Mohammed_Rashed_Daoud_al-0whali">
  <port name="inbox" port_type="input"/>
  <port name="outbox" port_type="output"/>
  <properties>
    <property name="knowledgeAccumulated" type="double" value="
      0.7500" />
  </properties>
  <measures>
    <measure name="taskExclusivity" type="double" value="0.5017"
      >
      <input id="social_network" />
    </measure>
    <measure name="cognitiveLoad" type="double" value="0.1354">
      <input id="social_network" />
    </measure>
  </measures>
</node>

```

Figure 8.4: Specification of Vertices in DyNetML

A port can be viewed as a point where an edge attaches to a vertex. Thus, a directed edge connecting a port specified as input to another node’s port specified as output represents a resource or information flow across the edge.

Since one can specify multiple input and output ports for every node, it is possible to represent a number of distinct flows along every edge while maintaining clear separation between different types of links.

A port is defined as follows (see figure 8.4):

- attribute **name** specifies a port ID that is unique for this node
- attribute **port\_type** is a multiple choice, with possible values “input”, “output” and “general”

### Node Types in DyNetML

DyNetML has been designed to assist the flow of information between software tools by not only enforcing a consistent structured format upon the

data, but also by specifying a constant vocabulary. Since the language has been designed in service of the Social Network Analysis community, we specify a set of standard node types that could be used to express a majority of rich social network data.

The standard node types are: **agent**, **organization**, **knowledge**, **resource**, **task**, **location**, and **graph**.

While the architecture of DyNetML allows developers to easily add node types, to ensure inter-operability of tools using DyNetML it is advisable to refrain from expanding the vocabulary unless absolutely required. To provide a more fine-grained node type mechanism, it is best to use the **prototype** attribute of nodes to specify arbitrary subtypes.

## 8.6 Representing Relations in DyNetML

DyNetML format allows the user to specify multiple graphs within a single framework, including graphs that share vertices with other graphs.

An example for use of such system is the case where a number of individuals are engaged in multiple relationship types - such as the formal network, informal advice network, or familial ties network.

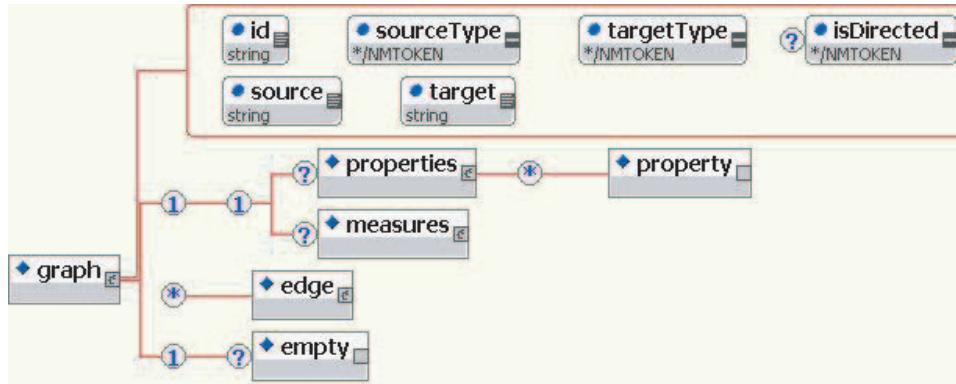
Each graph is specified as follows: (see figure 8.5)

- **id** attribute is the graph's unique ID.
- **source** attribute specifies the nodeset from which the source nodes are taken.
- **sourceType** attribute specifies the type of nodes contained in the source nodeset.
- **target** attribute specifies the nodeset from which the target nodes are taken.
- **targetType** attribute specifies the type of nodes contained in the target nodeset.
- **isDirected** attribute specifies whether the edges of this graph are directed; the attribute can only take values of "true" or "false".

The graph then includes **properties** and **edges** elements (see 8.7), followed by a set of **edge** elements that comprise the actual graph.

### 8.6.1 Edges

Edges (see figure 8.6) of the graph include the following attributes:



```

<networks>
  <graph sourceType="agent" targetType="agent" id="
    social_network">
    <edge source="Mohammed_Rashed_Daoud_al-0whali" target="
      Usama_Bin_Ladin" type="double" value="1.000000" />
    <properties>
      <property name="observedOn" type="string" value="
        01/01/2000" />
    </properties>
    <measures>
      <measure name="probability" type="double" value="0.10">
        <input id="social_network" />
      </measure>
    <edge source="Mohammed_Rashed_Daoud_al-0whali" target="
      Jihad_Mohammed_Ali_(Azzam)" type="double" value="6.000000
      " />
    ...more edges...
  </graph>
  ...more graphs...
</networks>

```

Figure 8.5: Specification of Graphs in DyNetML

- **source** and **sourcePort** attributes specify the source node and port that an edge originates from. The source node should be a part of the nodeset specified in the **source** attribute of the graph. **source** attribute is required; **sourcePort** is optional if no ports have been defined for the source node.
- **target** and **targetPort** attributes specify the source node and port that an edge connects to. The target node should be a part of the nodeset specified in the **target** attribute of the graph. **target** attribute is required; **targetPort** is optional if no ports have been defined for the target node.

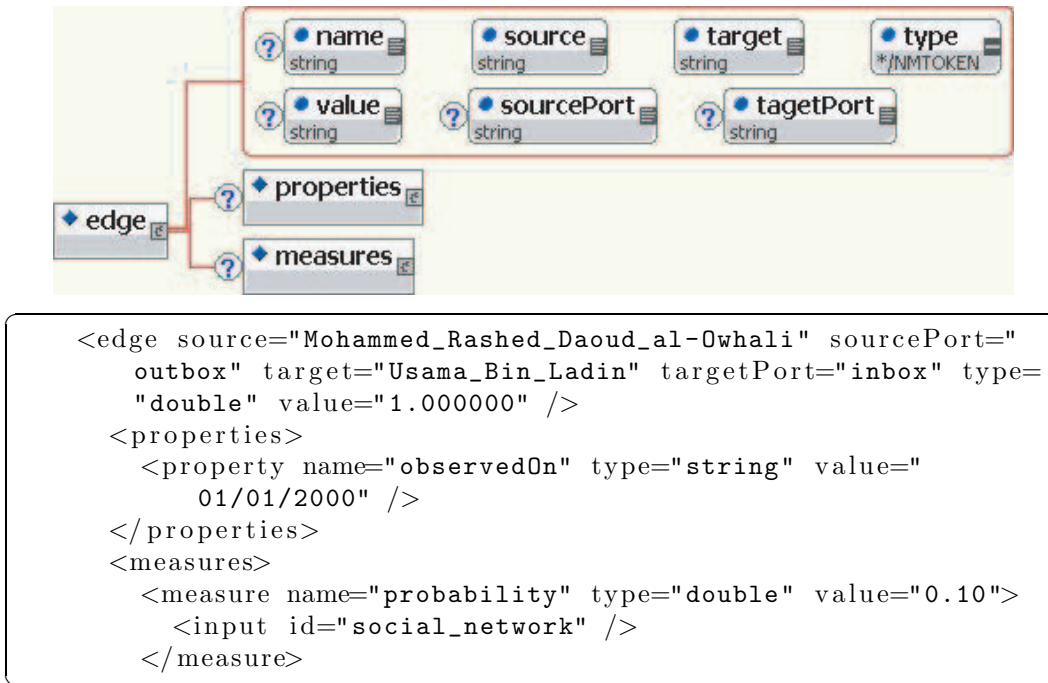


Figure 8.6: Specification of Edges in DyNetML

- **type** attribute is required for every edge. If the edge is unweighted, the type attribute should be set to “binary”; other acceptable edge value types are “double” and “string”
- **value** attribute specifies the edge weight or value; the type of the value should match the type specified in **type** attribute.
- **name** attribute is an optional string that allows the user to add a human-readable title to an edge.
- **properties** and **measures** elements are optional and allow addition of rich data to edge-level specification. A complete description of these elements can be found in section 8.7.

## 8.7 Representing Graph, Node and Edge Attributes

One of the important facilities of DyNetML is its ability to attach rich data or attributes to every element of the structure.

The rich data, specified as **properties** and **measures**, can be added to the **MetaMatrix**, **node**, **graph** and **edge** objects. The mechanisms for handling the rich data are identical for all objects.

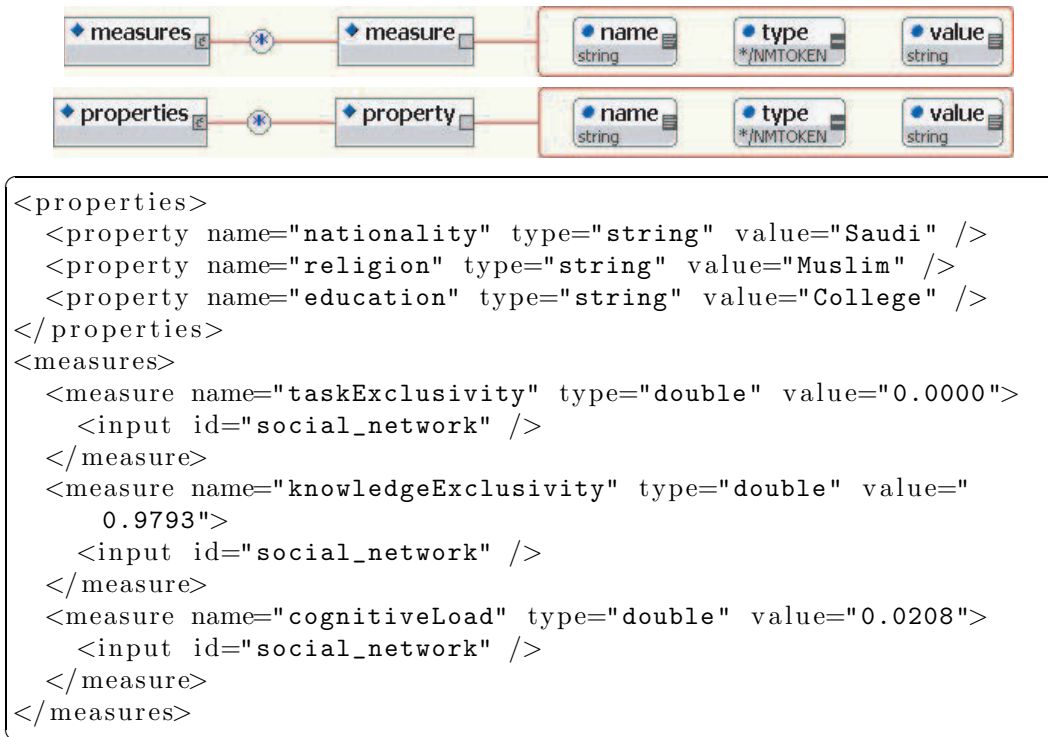


Figure 8.7: Specification of Properties and Measures

**Properties** and **Measures** objects are syntactically similar (see figure 8.7) and consist of a set of name-value pairs. The main distinction between them is that **Properties** should be thought of as attributes inherent to the subject, such as information obtained from a questionnaire or otherwise known about the subjects.

Measures, on the other hand, are computed by analysis tools and inserted into the dataset during processing.

The guidelines for naming properties and measures are following:

- Names should be descriptive of the nature of data contained within.
- Measure names should include the name of the tool that generated them.

For example, the measure of Freeman centrality computed by NetStat tool should look as:

```

<measure name="netstat_freeman_centrality" type="double"
value="3.14"/>

```

## 8.8 Complex Social Networks in DyNetML

The basic use of DyNetML is specification of rich social network data, including properties and measures attached to objects within the network. DyNetML also allows for an arbitrary number of network superimposed upon each other, and specification of network data over time.

The example below is a heavily commented small dataset containing two types of nodes (people and facts), and three networks (friendship, advice and knowledge). It is derived from a full MetaMatrix dataset on the terrorist bombing of the U.S. embassy in Tanzania.

```
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE MetaMatrix SYSTEM "DyNetML.dtd">
```

```
<MetaMatrix timePeriod="1997">
```

```
<nodes>  
  <nodeset id="agent" type="agent">  
    <node id="Mohammed_Rashed_Daoud_al-Owhali"/>  
    <node id="Khalfan_Khamis_Mohamed"/>  
    <node id="Mohammed_Sadiq_Odeh"/>  
    <node id="Ahmed_the_German"/>  
    <node id="Fazul_Abdullah_Mohammed"/>  
    <node id="Wadih_al_Hage"/>  
    <node id="Usama_Bin_Ladin"/>  
    <node id="Ali_Mohammed"/>  
    <node id="Ahmed_Khalfan_Ghailani"/>  
    <node id="Mohammed_Salim"/>  
    <node id="al-Fadl"/>
```

## Header Information

Time periods can be labelled with arbitrary string labels

Nodes are broken up into nodesets by type (e.g. agent, knowledge, resource, task, etc)

```

<node id="al-Fawwaz" />
  <properties>
    <property name="knowledge" type="double" value="0.2500" />
  </properties>
  <measures>
    <measure name="taskExclusivity" type="double" value="0.0000">
      <input id="social_network" />
    </measure>
    <measure name="knowledgeExclusivity" type="double" value="0.9793">
      <input id="social_network" />
    </measure>
    <measure name="cognitiveLoad" type="double" value="0.0208">
      <input id="social_network" />
    </measure>
  </measures>

```

```

<node id="Jihad_Mohammed_Ali_(Azzam)" />
<node id="abouhalima" />
<node id="Abdullah_Ahmed_Abdullah_(Saleh)" />
<node id="Abdal_Rahman" />
</nodeset>

```

```

<nodeset id="type" type="task">
  <node id="surveillance" />
  <node id="weapon_training" />
  <node id="driving_training" />
  <node id="bomb_prep" />
  <node id="bombing" />
</nodeset>

```

This is a more complex node with properties and attached measures

"TASK" nodeset outlines the types of tasks that the group performs

```

<nodeset id="resource" type="resource">
  <node id="building_for_bombmaking"/>
  <node id="money"/>
  <node id="bomb_material"/>
  <node id="truck"/>
</nodeset>

<nodeset id="knowledge" type="knowledge">
  <node id="religious_extremism"/>
  <node id="weapons_expertise"/>
  <node id="surveillance_expertise"/>
  <node id="media_consultant"/>
</nodeset>

</nodes>

```

"RESOURCE" nodeset:  
tangible resources for  
functioning of organization

"KNOWLEDGE": non-  
tangible or mental resources  
of the organization

Now we specify the graphs  
that comprise the metama-  
trix.

```

<networks>
  <graph id="social_network" sourceType="agent" targetType="agent" isDirected="true">
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Jihad_Mohammed_Ali_(Azzam)" type="double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Abdullah_Ahmed_Abdullah_(Saleh)" type="double" value="1"/>
    <edge source="Mohammed_Rashed_Daoud_al-Owhali" target="Abdal_Rahman" type="double" value="1"/>
    <edge source="Mohammed_Sadiq_Odeh" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Ahmed_the_German" target="Abdullah_Ahmed_Abdullah_(Saleh)" type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Fazul_Abdullah_Mohammed" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Mohammed_Sadiq_Odeh" type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Fazul_Abdullah_Mohammed" type="double" value="1"/>
    <edge source="Wadih_al_Hage" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Mohammed_Rashed_Daoud_al-Owhali" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Fazul_Abdullah_Mohammed" type="double" value="1"/>
    <edge source="Usama_Bin_Ladin" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Ali_Mohammed" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Ali_Mohammed" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="al-Fawwaz" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="al-Fawwaz" target="Usama_Bin_Ladin" type="double" value="1"/>
    <edge source="Jihad_Mohammed_Ali_(Azzam)" target="Mohammed_Rashed_Daoud_al-Owhali" type="double" value="2"/>
    <edge source="abouhalima" target="Wadih_al_Hage" type="double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Mohammed_Rashed_Daoud_al-Owhali" type="double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Mohammed_Sadiq_Odeh" type="double" value="1"/>
    <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="Abdal_Rahman" type="double" value="1"/>
    <edge source="Abdal_Rahman" target="Mohammed_Rashed_Daoud_al-Owhali" type="double" value="1"/>
    <edge source="Abdal_Rahman" target="Abdullah_Ahmed_Abdullah_(Saleh)" type="double" value="1"/>
  </graph>

```

This graph specifies the social network of the organization, i.e. who knows or speaks to whom.

NOTE: source and target of each edge should be a valid node; however, it's up to the software developer to ensure that, or to check consistency in any code that imports this data

```

<graph id="knowledge_network" sourceType="agent" targetType="knowledge">
  <properties>
    <property name="title" type="string" value="this_graph_specifies_who_knows_what">
    </property>
  </properties>
  <measures>
    <measure name="centralization" type="double" value="0.023">
    </measure>
    <measure name="density" type="double" value="0.45">
    </measure>
  </measures>
  <edge source="Mohammed_Rashed_Daoud_al-0whali" target="religious_extremism" type="double" value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="weapons_expertise" type="double" value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="surveillance_expertise" type="double" value="1"/>
  <edge source="Mohammed_Sadiq_Odeh" target="religious_extremism" type="double" value="1"/>
  <edge source="Fazul_Abdullah_Mohammed" target="religious_extremism" type="double" value="1"/>
  <edge source="Wadih_al_Hage" target="religious_extremism" type="double" value="1"/>
  <edge source="Ali_Mohammed" target="surveillance_expertise" type="double" value="1"/>
  <edge source="Mohammed_Salim" target="religious_extremism" type="double" value="1"/>
  <edge source="al-Fadl" target="religious_extremism" type="double" value="1"/>
  <edge source="Jihad_Mohammed_Ali_(Azzam)" target="religious_extremism" type="double" value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="surveillance_expertise" type="double" value="1"/>
  <edge source="Usama_Bin_Ladin" target="media_consultant" type="double" value="1"/>
  <edge source="al-Fawwaz" target="media_consultant" type="double" value="1"/>
</graph>

```

Specifies who in the organization knows or has trained in which areas of knowledge. This graph has a number of associated measures and a text property.

```

<graph id="resource_network" sourceType="agent" targetType="resource" isDirected="
  true">
  <edge source="Khalfan_Khamis_Mohamed" target="building_for_bombmaking" type="
    double" value="1"/>
  <properties>
    <property name="edgeDescription" type="string" value="propertyOwnership">
  <properites>
  <measures>
    <measure name="probability" type="double" value="0.85">
  <measures>
  <edge source="Mohammed_Sadiq_Odeh" target="building_for_bombmaking" type="double"
    value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="building_for_bombmaking" type="
    double" value="1"/>
  <properties>
    <property name="edgeDescription" type="string" value="propertyLease">
  <properites>
  <measures>
    <measure name="probability" type="double" value="0.4">
  <measures>
  <edge source="Fazul_Abdullah_Mohammed" target="building_for_bombmaking" type="
    double" value="1"/>
  <edge source="Wadih_al_Hage" target="bomb_material" type="double" value="1"/>
  <edge source="Usama_Bin_Ladin" target="money" type="double" value="1"/>
  <edge source="Ahmed_Khalfan_Ghailani" target="truck" type="double" value="1"/>
  <edge source="Mohammed_Salim" target="money" type="double" value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="building_for_bombmaking"
    type="double" value="1"/>
  <edge source="Abdal_Rahman" target="bomb_material" type="double" value="1"/>
</graph>

```

Relationship between people and tangible resources of the organization  
 This graph illustrates use of edge properties to represent arbitrary rich data

```

<graph id="people_task" sourceType="agent" targetType="task">
  <edge source="Mohammed_Rashed_Daoud_al-0whali" target="surveillance" type="double"
    value="1"/>
  <edge source="Khalfan_Khamis_Mohamed" target="weapon_training" type="double"
    value="1"/>
  <edge source="Mohammed_Sadiq_Odeh" target="weapon_training" type="double" value="
1"/>
  <edge source="Ahmed_the_German" target="bombing" type="double" value="1"/>
  <edge source="Wadih_al_Hage" target="weapon_training" type="double" value="1"/>
  <edge source="Ali_Mohammed" target="surveillance" type="double" value="1"/>
  <edge source="Ahmed_Khalfan_Ghailani" target="bomb_prep" type="double" value="1"/
  >
  <edge source="Mohammed_Salim" target="weapon_training" type="double" value="1"/>
  <edge source="al-Fadl" target="weapon_training" type="double" value="1"/>
  <edge source="Jihad_Mohammed_Ali_(Azzam)" target="surveillance" type="double"
    value="1"/>
  <edge source="Abdullah_Ahmed_Abdullah_(Saleh)" target="bomb_prep" type="double"
    value="1"/>
  <edge source="Abdal_Rahman" target="weapon_training" type="double" value="1"/>
</graph>

```

Assignment of people to tasks

```

<graph id="resource_task" sourceType="resource" targetType="task">
  <edge target="weapon_training" source="weapons_expertise" type="double" value="1"
  />
  <edge target="driving_training" source="weapons_expertise" type="double" value="1"
  "/>
  <edge target="bomb_prep" source="weapons_expertise" type="double" value="1"/>
  <edge target="bombing" source="religious_extremism" type="double" value="1"/>
  <edge target="bombing" source="weapons_expertise" type="double" value="1"/>
</graph>

```

Resource requirements for tasks

```

<graph id="knowledge_task" sourceType="knowledge" targetType="task">
  <edge source="religious_extremism" target="surveillance" type="double" value="1"/>
  <edge source="weapons_expertise" target="surveillance" type="double" value="1"/>
  <edge source="surveillance_expertise" target="surveillance" type="double" value="1"/>
  <edge source="media_consultant" target="surveillance" type="double" value="1"/>
  <edge source="weapons_expertise" target="weapon_training" type="double" value="1"/>
  <edge source="surveillance_expertise" target="weapon_training" type="double" value="1"/>
  <edge source="weapons_expertise" target="driving_training" type="double" value="1"/>
  <edge source="weapons_expertise" target="bomb_prep" type="double" value="1"/>
  <edge source="surveillance_expertise" target="bombing" type="double" value="1"/>
  <edge source="surveillance_expertise" target="bombing" type="double" value="1"/>
</graph>

```

```

<graph id="task_network" sourceType="task" targetType="task" isDirected="true">
  <edge source="bomb_prep" target="weapon_training" type="double" value="1"/>
  <edge source="bombing" target="driving_training" type="double" value="1"/>
  <edge source="bombing" target="bomb_prep" type="double" value="1"/>
  <edge source="bombing" target="surveillance" type="double" value="1"/>
</graph>

```

```

</networks>
</MetaMatrix>

```

Knowledge requirements for tasks

Precedence relationship of tasks in the grand task of the organization (i.e. large-scale attack)

End of example