NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

GENERATING GRAPHML XML FILES FOR GRAPH VISUALIZATION OF ARCHITECTURES AND EVENT TRACES FOR THE MONTEREY PHOENIX PROGRAM

by

Timothy L. Shields

September 2012

Thesis Advisor: Mikhail Auguston
Second Reader: Terry Norbraten

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Generating GraphML XML Files for Graph Visualization of Architectures and Event Traces for the Monterey Phoenix Program		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Timothy L. Shields			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Architecture, architecture modeling, and testing architecture models are key components of the software design process. The ability to design and visualize architecture models efficiently and accurately has a direct impact on the ability of designers to satisfy the requirements of their stakeholders. The Monterey Phoenix (MP) program is one such application for building executable architecture models. The MP program is used to build and test models based upon inter-related events between the user-environment, program processes and data.

At the heart of architecture analysis in MP is the accurate creation of high-level graphs that depict the details of both software architectures and event traces of resulting architectures. To date, development of these graphs has been a fragmented, manual process that often relies on shoe-horning other applications into tasks for which those programs were never intended to be used.

This thesis implements MPGrapher, an LL1, single-pass compiler that generates XML documents for visualizing MP architectures and event traces. It is based on generating files that conform to the Graph Markup Language (GraphML). MPGrapher compiles well-formed XML files that conform to the yEd GraphML schema. These files will be opened and analyzed using the tools provided by the free yEd graphing application.

14. SUBJECT TERMS XML, GraphML, Monterey Phoenix, software architecture, executable software architecture models, software architecture visualization			**15. NUMBER OF PAGES** 225
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

GENERATING GRAPHML XML FILES FOR GRAPH VISUALIZATION OF ARCHITECTURES AND EVENT TRACES FOR THE MONTEREY PHOENIX PROGRAM

Timothy L. Shields
Captain, United States Marine Corps
B.A., Miami University, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Timothy L. Shields

Approved by: Mikhail Auguston
Thesis Advisor

Terry Norbraten
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Architecture, architecture modeling, and testing architecture models are key components of the software design process. The ability to design and visualize architecture models efficiently and accurately has a direct impact on the ability of designers to satisfy the requirements of their stakeholders. The Monterey Phoenix (MP) program is one such application for building executable architecture models. The MP program is used to build and test models based upon inter-related events between the user-environment, program processes and data.

At the heart of architecture analysis in MP is the accurate creation of high-level graphs that depict the details of both software architectures and event traces of resulting architectures. To date, development of these graphs has been a fragmented, manual process that often relies on shoe-horning other applications into tasks for which those programs were never intended to be used.

This thesis implements MPGrapher, an LL1, single-pass compiler that generates XML documents for visualizing MP architectures and event traces. It is based on generating files that conform to the Graph Markup Language (GraphML). MPGrapher compiles well-formed XML files that conform to the yEd GraphML schema. These files will be opened and analyzed using the tools provided by the free yEd graphing application.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ACM	Association for Computing Machinery
BPMN	Business Process Model and Notation
DAG	Directed Acyclic Graph
DOM	Document Object Model
ERD	Entity Relationship Diagrams
FM	Formal Methods
GML	Graph Modelling Language
GraphML	Graph Markup Language
GXL	Graph eXchange Language
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
ISCA	International Society for Computers and Their Applications
ISCA	International Symposium on Computer Architecture
MP	Monterey Phoenix
PCDATA	Parsed Characted Data
RAM	Random Access Memory
SIAM	Society for Industrial and Applied Mathematics
SODA	Symposium on Discreet Algorithms
SysML	Systems Modeling Language
UML	Unified Modeling Language
W3C	Worldwide Web Consortium
XGMML	eXtensible Graph Markup and Modeling Language
XPath	XML Path Language

XSLT	XML Stylesheet Language Transformations
XML	Extensible Markup Language

ACKNOWLEDGMENTS

To Professor Mikhail Auguston, my advisor for this project. First you taught me that I actually do know something about automata. Then you piqued my interest in building models in the field of software architecture. Then you took me under your wing and helped me to trace a successful series of events to the finish.

To Professor Man-Tak Shing, for your help, your instruction, your prayers, and your genuine concern for me well-being.

Terry Norbraten and John Falby, not only for teaching me Java, but for your willingness to take a break from whatever you were doing to help me out whenever I asked.

To Loren Peitso, for teaching me how operating systems organize memory, and for subsequently helping me to organize my own.

To Commander Alan Shaffer, for helping me on a professional basis, for opening my eyes to worlds of programming that I never realized existed, and for providing a host of inspiration that you never even realized.

To Donna Cuadrez, for giving my project that extra special attention (e.g., some much needed one-on-one help when I couldn't make it to the required briefs, et al.).

Most of all I would like to thank my wife Nallely. You kept me sane, and gave me something to look forward to besides tests and papers and deadlines and writing code. Thank you for the family – I couldn't have done this without you. Perhaps Chicago sang it best, "You're the inspiration."

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT

Software architecture, building models of software architectures, as well as testing architecture models, are all key components of the software design process. The ability to accurately design and visualize architecture models in a timely and efficient manner directly impacts the ability of designers to satisfy the requirements of their stakeholders. The Monterey Phoenix (MP) program is one such application for constructing executable architecture models. The MP program is used to build and test models based upon inter-related events between the user-environment, program processes and data.

At the heart of architecture analysis in MP is the accurate creation of high-level graphs that depict the details of both software architectures and event traces of resulting architectures. To date, development of these graphs has been a fragmented, manual process that often relies on shoe-horning other applications into tasks for which those programs were never intended to be used. For instance, the authors of MP have re-written MP-specific architectures into Alloy-specific formats and fed the resulting files to Alloy solely for the purpose of creating coherent graphs.

This thesis implements MPGrapher, an LL1, single-pass compiler that generates XML documents for visualizing MP architectures and event traces. It is based on generating files that conform to the Graph Markup Language (GraphML). MPGrapher compiles well-formed XML files that conform to the yEd GraphML schema. These files are meant to be opened and analyzed using the free graphing tools, especially the yEd graphing application.

1. Proposed Area of Research

The general area of research for this thesis is to investigate the use of the Graph Markup Language (GraphML) standard to generate visual representations of program event traces. The goal of the proposed research is to demonstrate the potential utility of automatically generating GraphML files from event traces for hundreds of nodes of event

traces (200 nodes average) and 1,000 edges between nodes. GraphML files will be used as input to the yEd Graph Editor (or other 3rd party graphing software) for display.

2. List of Initial Research Questions

- What elements and attributes are necessary for coherent representation of graphs in the GraphML language?
- Are other markup languages available to represent graphs?
- Is yEd suitable for displaying graphs generated by MP?
- Are other graphing applications also well-suited for displaying graphs generated by MP?
- Can the yEd graphing application handle hundreds of nodes and thousands of edges between nodes?

B. BRIEF INTRODUCTION TO FORMAL METHODS

There are a great many definitions and opinions as to what formal methods (FM) are. There are also a great many ways to perform FM. One way to define FM is "any attempt to use mathematics in the development of a software-intensive computer-based system, in order to improve the quality of the resulting software-intensive computer-based systems." (Berry, 1999) Another, multi-part definition might be, "1) a specification written and approved in accordance with established standards, and 2) a specification written in a formal notation, often for use in proof of correctness." (Bowen & Hinchey, 1995) In layman terms, FM are essentially mathematical proofs of whether formal specifications are correct or not.

One also does not have to look very far to find criticisms of FM, nor does one have to search hard to find individuals who are ready to openly argue that FM are a complete waste of time and money.

The answer to the question, "What are formal methods, and why are they important?" can best be answered with answers to additional questions.

1. What Are the Obstacles, Limitations, and Overhead of FM?

In two separate papers, Hall, Bowen and Hinchley summarized the obstacles, limitations and overhead of FM in 14 simple points. The first seven were provided by Hall in 1990 (Hall, 1990).

- FM are all about proving programs
- Because FM are all about proving programs, they require highly-trained mathematicians
- FM delay the development process
- FM lack tools
- Because of the previous four reasons, FM increase the cost of development
- Those who don't fully understand FM believe that it is supposed to guarantee that software is perfect (and this is exactly what they hope for since they believe they will have to pay so much for them)
- Therefore, FM are only useful (i.e., worth the cost) for safety critical systems

An additional seven points were offered later (Bowen & Hinchey, 1995)

- FM are unacceptable to users
- FM are not used on real, large-scale systems
- FM replace traditional engineering design methods
- FM only apply to software
- FM are unnecessary
- FM are not supported
- FM people always use FM

2. What Are the Benefits of Using Formal Methods?

In reality, all of the above obstacles described in section 1 are not real obstacles. The obstacles described in section 1 are in fact counter-arguments to some of the biggest benefits to doing FM. Case in point: since 1990, a number of tools for doing FM have been developed. Specification languages such as OBJ and Z have been defined. Provers such as Larch and Nqthm were written. Type checkers such as Fuzz, Formaliser, Zola, and Cadiz have also been developed. There is also little evidence to support the claim that

FM can only be done by hard-core mathematicians – those who are best at FM will generally confess that the math required for doing effective FM can be learned in less than a week.

3. What Are Formal Specifications and Formal Models?

Essentially, formal models are derived from formal specifications. Formal specifications more or less capture requirements and other necessities. On the other hand, formal models are built on previously defined specifications.

4. What Phases of Software Development Best Support the Use of FM?

The most important time to use formal methods is certainly during the design phase. It is well known that the most egregious software errors are made during the design phase. Therefore, it is most important to catch design errors before any time or personnel are committed to writing code, and that is precisely where formal methods come into play. Additionally, it is also well known that fixing errors at design time is far less costly than fixing errors during the test phase. Therefore it also makes sense that spending money on good, precise formal methods during the design phase is far less costly than fixing egregious errors during the testing phase.

Another time when it would make sense to apply FM is during the maintenance phase, when users and designers come back to the drawing room to discuss changes and additions to the next iteration of the software. In reality, it makes always sense to spend a little money and effort to re-apply FM any time there are changes to requirements and/or test-cases.

C. THE MONTEREY PHOENIX PROGRAM

The Monterey Phoenix program was developed in response to the fourth criticism stated in section B.1, “Formal Methods lack tools.” While it is true that many tools have been developed for doing formal methods on software, the Monterey Phoenix program is designed to work not only for software verification, but for verification of systems and systems of systems as well.

Mikhail Auguston, Clifford Whitcomb, and others have proposed several formal software and system architecture specification models based on the behaviors of those systems. These models aim to solve the provided verification of systems requirements at the earliest stages of the development process.

For these models, the behavior of the system is defined as a set of events (rendered as event traces) with two basic relations: event precedence, and event inclusion. The structures of event traces are specified using event grammars and constraints which are further organized into schemas.

The MP project provides tools and methods which are designed to improve the development of system architectures, and to verify and test requirements and design decisions early in the development process. The main objectives of the project are:

- To develop a new approach to software and system architecture formal specification based on behavior models.
- To make architecture models executable on an abstract machine, so that it becomes possible early in the system development phase to do testing and verification of the top level system design.
- To provide a method and tools for extracting multiple views from the architecture models, and corresponding visualization tools that can be customized to the user's needs.
- To provide the method for system stepwise refinement from the top level architecture models to the detailed design and implementation models, supported by tools for sanity checks and refinement consistency checks.
- To provide formalism for specifying system's environment models, based on the behavior modeling, so that the system architecture can be tested and verified in the interaction with its environment.

Two different kinds of graphs are needed for the Monterey Phoenix program, as the following subsections will demonstrate. The first kind is the architecture graph. The architectures of the systems (or system of systems) that MP analyzes belong in their own category of graphs.

Graphs of architectures require only two types of edges, but those edges may be either directed or undirected, and each edge may have one or more labels. Graphs of the behavior analyses that MP performs on these architectures belong to their own category of graphs. These graphs also require just two types of edges, but these edges must be

completely distinct from the edges defined for architectures. The edges of a trace graph will never have labels, and while architecture edges may be either directed or undirected, trace edges are strictly directed.

Nodes are also a concern for the two types of graphs. Trace graphs will only have two types of nodes – roots and events. Architectures, on the other hand, require three different kinds of nodes: environment, processes and data.

Some examples of MP architectures and traces are presented here. (Auguston & Whitcomb, 2012)

1. Pipe and Filter Example

Architecture designs are fed to MP as text files. An explanation of the grammar used for MP input files is beyond the scope of this thesis.[1] An example of the input file that was used to compose a simple pipe and filter architecture is given in Figure 1.

```
SCHEMA simple_message_flow
ROOT Task_A:  (* send *);
ROOT Task_B:  (* receive *);
COORDINATE  (* $x: send  *) FROM Task_A,
            (* $y: receive *) FROM Task_B ADD $x PRECEDES $y;
```

Figure 1. MP input file for pipe and filter architecture

Figure 2 provides a graphical representation of the architecture described in Figure 1. Input files such as that shown in Figure 1 are difficult to read and understand. A picture, however, is worth a thousand words. Figure 2 is the sort of architecture diagram that the MPGrapher application attempts to reproduce in an automatic manner.

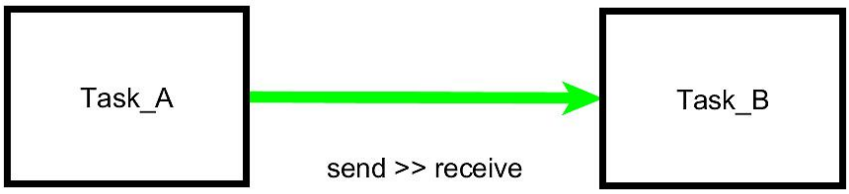

Figure 2. Pipe and filter example: architecture (from Auguston & Whitcomb, 2012)

[1] Auguston & Whitcomb, 2012 provides detailed instructions on how to use the grammar and compose input files for MP.

Figure 2 describes a simple system, whereby messages are sent and received, and send events must precede receive events. Each box describes a separate process. The green arrow between the boxes represents an interaction between the two processes. The event that triggers the interaction on the side of Task_A is called `send`, whereas the event that triggers the interaction on the side of Task_B is called `receive`. In this case, interactions are strictly ordered – `send` events must preceded `receive` events.

Running a trace of three send events and three receive events would result in a graph trace similar to Figure 3. In Figure 3, pairs of events (nodes) that are in a "precedes" relationship with other events are connected with dashed, green, directed edges, while pairs of events which are in an "includes" relationship are connected by solid, black, directed arrows.

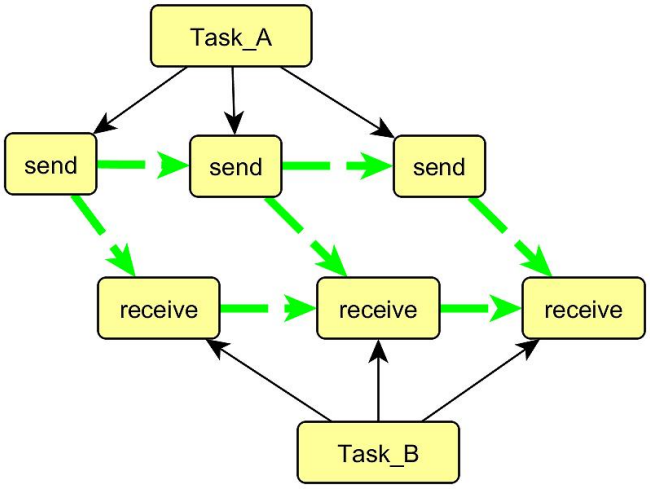

Figure 3. Pipe and filter example: trace (from Auguston & Whitcomb, 2012)

Figure 3 shows the same two processes as Figure 2. The black arrows in Figure 3 indicate that `Task_A` and `Task_B` include those events to which they point. As one would expect, messages must be sent before they can be received, and the first message sent must precede the second and third messages sent, as indicated by the green dashed arrows. The dashed green arrows also indicate that for this particular event trace message one must also be received before messages two and three. This is an indication of a design flaw in the pipe and fileter architecture that was uncovered by MP. As such, it

may also be an issue that the designers of the pipe and filter architecture may want to re-design before coding and testing begin.

2. ATM System Process Example

Figure 4 describes a rather more involved architecture.

```
SCHEMA ATM_withdrawal
ROOT Customer: (*  insert_card
      ( (   identification_succeeds request_withdrawal
          ( get_money | not_sufficient_funds ))| identification_fails)*);
ROOT ATM_system:  (*   read_card      validate_id
      ( id_successful  check_balance
        ((sufficient_balance dispense_money)| unsufficient_balance)| id_failed)*);
ROOT Data_Base:       (* ( validate_id | check_balance ) *);
Data_Base,  ATM_system SHARE ALL validate_id, check_balance ;
COORDINATE  (* $x: insert_card  *)     FROM Customer,
            (* $y: read_card *)        FROM ATM_system        ADD $x PRECEDES $y ;
COORDINATE  (* $x: request_withdrawal *) FROM Customer,
            (* $y: check_balance *)       FROM ATM_system     ADD $x PRECEDES $y ;
COORDINATE  (* $x: identification_succeeds *) FROM Customer,
            (* $y: id_successful *)           FROM ATM_system ADD $y PRECEDES $x ;
COORDINATE  (* $x: get_money  *)       FROM Customer,
            (* $y: dispense_money *)   FROM ATM_system        ADD $y PRECEDES $x ;
COORDINATE  (* $x: not_sufficient_funds *) FROM Customer,
            (* $y: unsufficient_balance *) FROM ATM_system   ADD $y PRECEDES $x ;
COORDINATE  (* $x: identification_fails  *)   FROM Customer,
            (* $y: id_failed  *)              FROM ATM_system ADD $y PRECEDES $x ;
```

Figure 4. MP input file for ATM system architecture

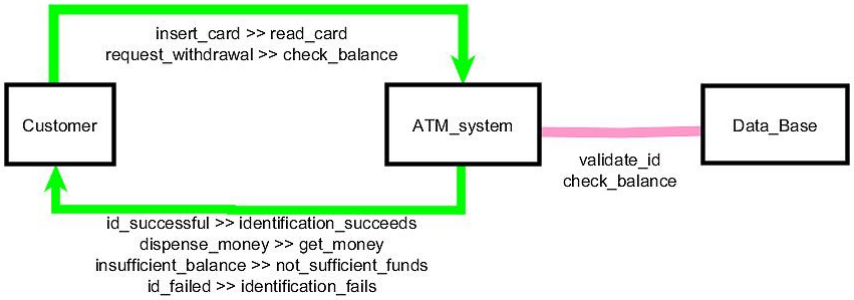

Figure 5. ATM system example: architecture (from Auguston & Whitcomb, 2012)

The first difference one will note is that unlike example 1, the architecture of Figure 5 has two different kinds of edges. As in the pipe and filter example, the green arrows indicate a strict ordering between events: `insert_card` events must precede `read_card` events, `dispense_money` events must precede `get_money` events, and so on. The pink line, however, indicates simple handshake events between the `ATM_System` and `Data_base` processes. In a handshake-like event there is no

ordering. The trace graph which results from running this architecture is also quite a bit more complicated, as Figure 6 demonstrates.

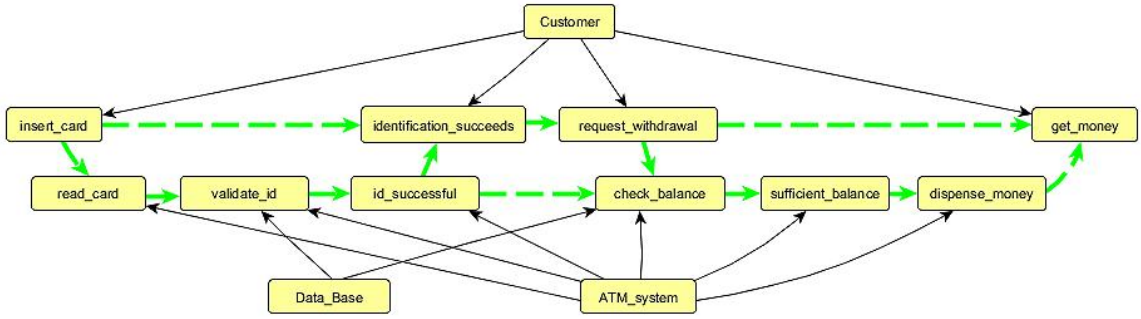

Figure 6. ATM system example: trace (from Auguston & Whitcomb, 2012)

The node-types and edge-types of Figure 6 aren't any different from those of Figure 3; there are just many more of them, which lead to a more complicated graph.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND SELECTION OF YED

A. SOFTWARE DESIGN IS DONE BY GRAPHING

Graphing is ubiquitous in the world of software design. Before one sits down to code a piece of software, it is important to first know what the architecture of the system is to be. While it is always possible to fill volumes of printed material with written words which describe exactly how software is to be put together, pictures are generally preferred, even in the design of software, systems, and systems of systems architectures. These days, industry and academia most often use the Unified Modeling Language (UML) to design software, and the entirety of UML is focused around graphs: Use-Case Diagrams, Class Diagrams, Collaboration Diagrams, Component Diagrams, State Diagrams, Activity Diagrams, and Sequence Diagrams. While there are a number of alternatives to UML diagrams, , they all depend on graphs irregardless of the name of the alternative.

1. Small Subset of Well-established UML Alternatives

a. Systems Modeling Language (SysML)

SysML offers a means to design architectures that is similar to UML but is more compact and theoretically easier to learn. SysML is essentially designed to support the design of systems and systems of systems in addition to software. SysML artchitectures comprise many of the diagrams available to UML as well as requirements diagrams, parametric diagrams, binary decision diagrams, and internal block diagrams. Following are examples of a requirements diagram (Figure 7) and a parametric diagram (Figure 8) from SysML.

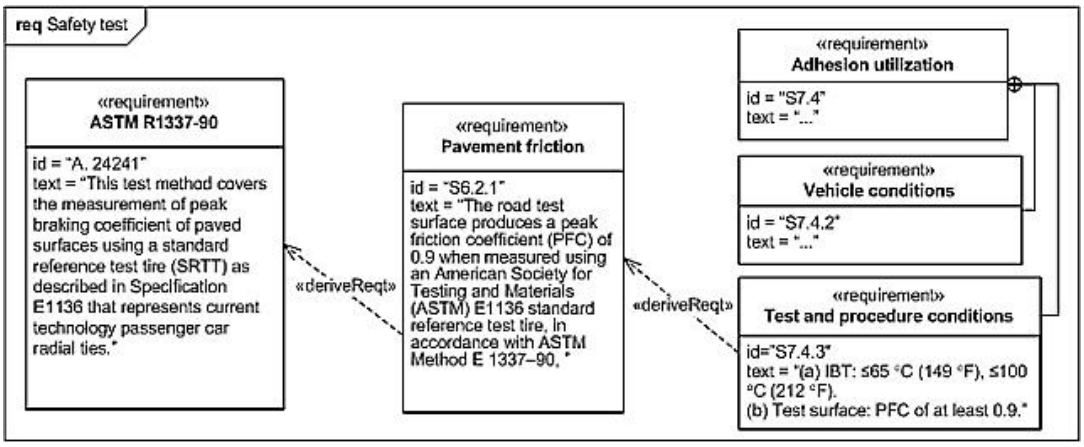

Figure 7. SysML requirements diagram (From FormalMind Website Authors, 2011)

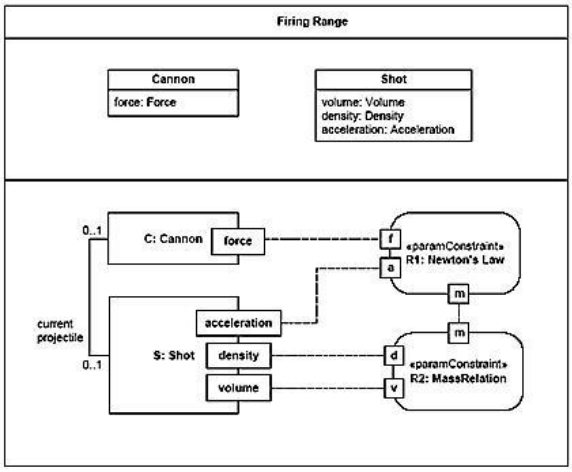

Figure 8. SysML parametric diagram (From EETimes Website Authors, 2011)

b. Business Process Model and Notation (BPMN)

BPMN notation is used to model business processes. BPMN is often used to capture the business logic of an organization as a part of requirements engineering.

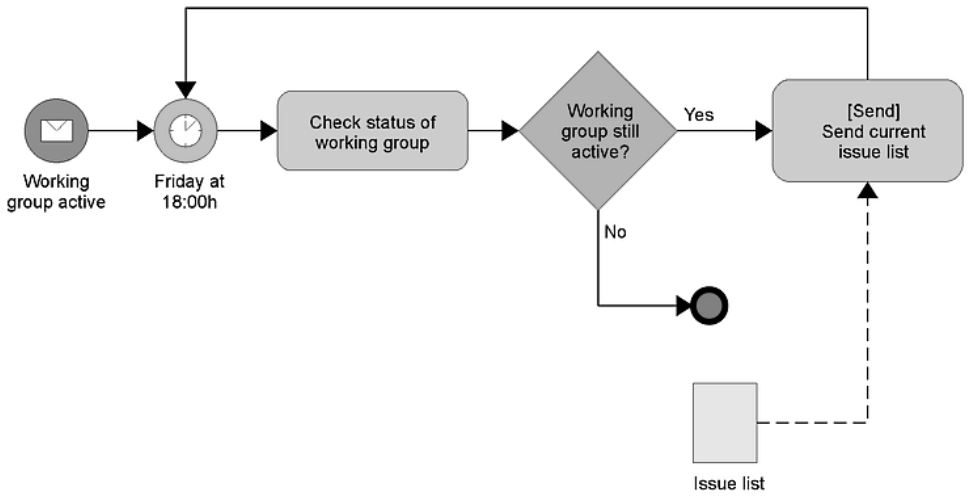

Figure 9. BPMN example diagram (From Wikipedia contributors, 2012)

c. Entity Relationship Diagrams (ERD)

ERDs are graphical representations of data and are frequently used to design databases.

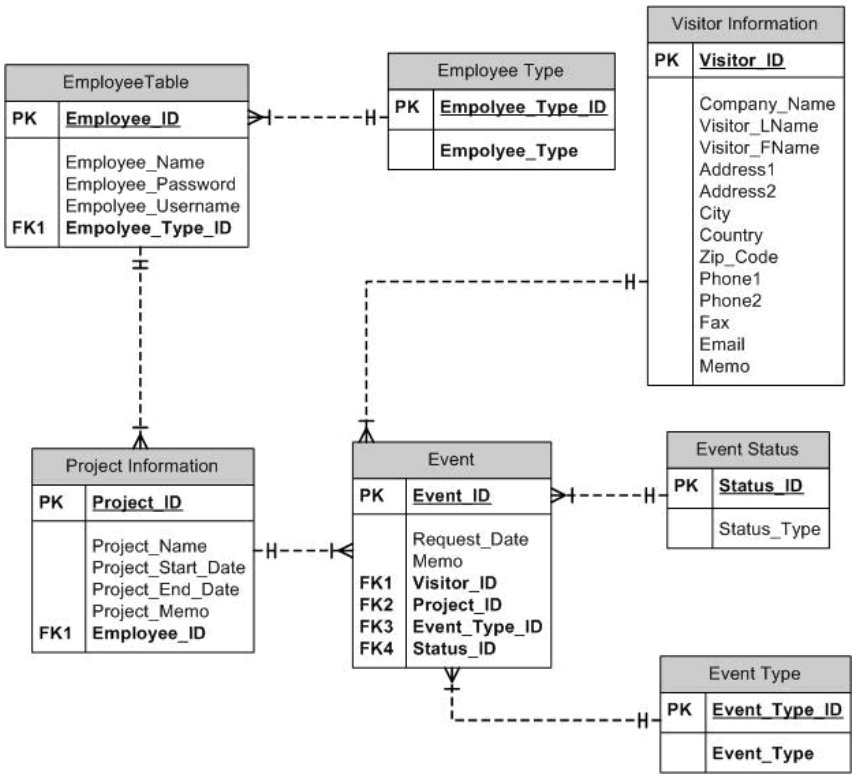

Figure 10. ERD example diagram (From JPMensah website authors, 2005)

d. Data Flow Diagrams

Data Flow Diagrams graph the flow of a piece of data through an information system (or systems).

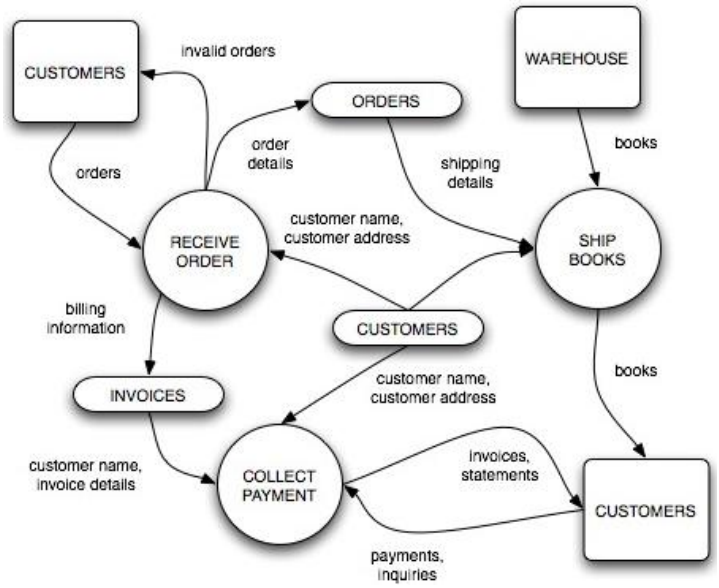

Figure 11. Data flow example diagram (From Yourdon website authors, 2012)

2. Further Discussion of the Definition of the Word "Graph"

Regardless of whether they are called graphs or diagrams, the examples above all have the following in common: they contain nodes, as well as edges which connect pairs of nodes. As such, they may all be produced by graphing software. Additionally, while graphical depictions of architectures have obvious benefits, they all suffer from the same drawbacks of graphs. Namely, it is difficult to represent very large systems within limited real estates (such as computer monitors or printed pieces of paper). Additionally, as the numbers of nodes of a graph becomes high, and the numbers of edges increases, crossings of edges tends to become quite high, which in turn reduces the reader's ability to understand what the architect is attempting to depict (especially when there are labels attached to the edges). Also, it is often possible and beneficial to remove nodes from a graph with no loss of meaning, which in turn increases the readability of the graph. Each of these issues, along with an exploration of the XML graphing language GraphML, will be explored in sections B–E.

The differences in meaning between the words “graph” and “diagram” are often blurred. Many dictionaries actually uses the word diagram to define the word graph and lists each as a synonym for the other. A graph need not contain edges to satisfy the definition of a graph—diagrams contain icons, which are essentially nodes, and can be reproduced by any graphing application. For instance, an engineering diagram of the outside architecture of a building could essentially be thought of as a graph of overlapping nodes but no edges. It is also possible to represent all kinds of charts as graphs. For instance, a bar chart is simply a graph of one or more square nodes, with no edges and no intervening space between the nodes. Even tables of information could be reproduced by graphing applications – each cell is essentially a node (however large) with a label (however lengthy), adjoined to adjacent nodes, with no spaces in between and no edges (unless one wanted to represent the borders of cells as edges). This thesis will treat the two words as synonyms according to the following definition: “a collection of one or more nodes and edges which either join pairs of vertices or represent loops to a single node.”

B. GRAPH VISUALIZATION

One of the goals of the Monterey Phoenix (MP) project is to provide libraries of predefined predicates, functions and tools to extract and visualize views. To date, the ideal way to visualize the event traces of MP (as well as the architectures which provide the input to MP) is with directed acyclic graphs (DAGs). However, given the large number of possible nodes (up to several hundred) and edges (up to more than one thousand), several issues arise in regards to the readability of such graphs. Some of these issues are: simplifying the graph by transitive reduction of unnecessary edges (discussed in section D), improving readability of graphs, to include minimizing the number of crossings between edges (discussed in section E), and especially improving the visualization and cognitive understanding of large graphs. There is a number of methods available for maximizing the cognitive meaning of large graphs within the limited, such as layered, or Sugiyama, graph drawing. In a Sugiyama graph, all the arrows of a directed graph point down and the nodes are arranged into horizontal levels, or layers. Such graphs are easy to comprehend, but are generally not suited well to showing graphs of

more than one type of edge, nor are they suited well to large graphs of hundreds of nodes and edges. One type of graph visualization, which shows some very exciting potential for MP is the force directed graph, as discussed in an article by Hachul and Jünger (2005).

The article by Hachul & Jünger is from the academic field of drawing force directed algorithms. Force directed algorithms provide a means for drawing graphs with roughly equal length edges and as few edge crossings as possible. Demonstrations of the algorithms also produce some very aesthetically pleasing graphs as shown in Figures 12 and 13.

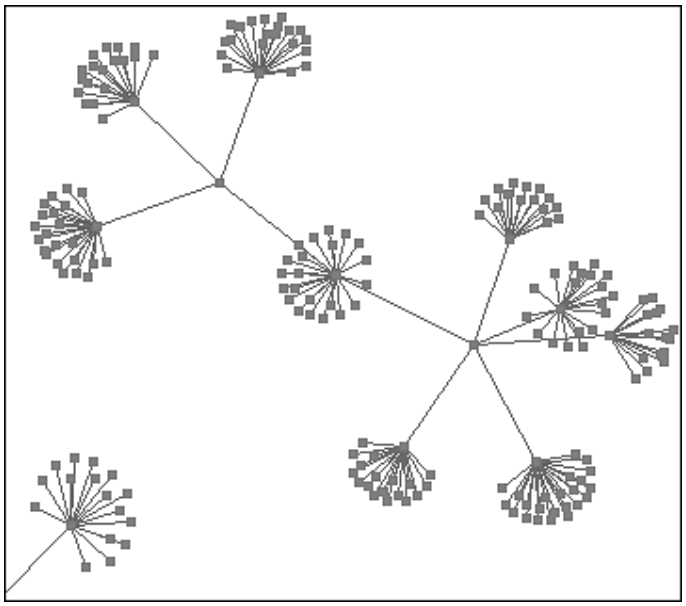

Figure 12. Graph produced from force directed algorithm (1)

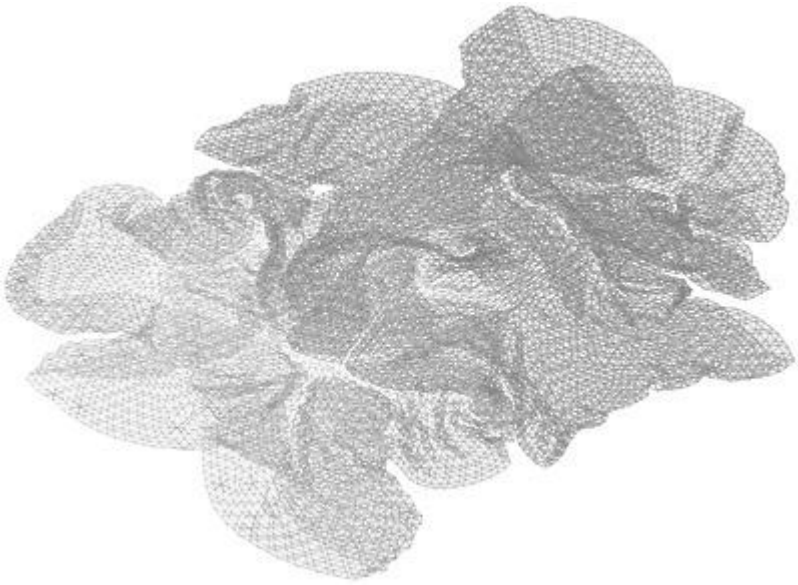

Figure 13. Graph produced from a force directed algorithm (2)

The most common means of achieving the force effect is achieved by the application of Hooke's Law[2] to edges and Coulomb's Law[3] to nodes. In other words, forces are applied to edges as if they were springs, and other forces are applied to the nodes as if they were charged particles. The entire graph is then simulated as if it were a dynamic system, whose "states" are iteratively adjusted until the system comes to equilibrium.

Force directed graphs provide a number of advantages, chief among them being the fact that no special knowledge of graph theory is required. However, chief among the drawbacks is the high running time of the algorithms. Improving running times are the focus of Hachul and Michael Jünger's article. The authors claim that, given a graph `G` with vertices `V` and edges `E` where `G = (V, E)`, the worst-case running time of their algorithm `= O(|V|log|V|+|E|)`, while the running times of traditional force directed graphs are at least quadratic. They further demonstrate that their algorithm draws graphs of 100,000 nodes in less than five minutes with minimal edge crossings (the

[2] Hooke's law of elasticity is an approximation that states that the extension of a spring is in direct proportion with the load applied to it. (Wikipedia contributors, 2012).

[3] Coulomb's inverse-square law is a law of physics describing the electrostatic interaction between electrically charged particles. (Wikipedia contributors, 2014).

authors tested the algorithms in a C++ application on a Linux machine with a 2.8 GHz processor – the amount of available RAM was not given).

The improvement in running time is essentially achieved by splitting a graph into multiple sub-graphs of at most depth three. The authors describe these sub-graphs as solar systems: a central "sun" node, with branching "planet" nodes, with branching "moon" nodes, which have further "satellite" nodes. Rather than iteratively running the force model on the graph as a whole, the models are run concurrently on each of the solar systems, which are then re-combined into a whole graph. When the graph is very deep, the "sun" nodes of the various sub-graphs may in turn become "satellite" nodes of the larger graph, and so on.

The authors provide the math and representative running time for each of the steps in the process: the graph partitioning process (initially random and iterative thereafter), and the force calculation step. Throughout the narrative, the authors discuss their (faster) approximations of the traditional force directed algorithms and explain how accuracy is achieved by convergence via the iterative steps of the process.

The authors provide the C++ source code implementation of their algorithm in a separate paper, which could seemingly be ported over to Java fairly easily. It is especially relevant since the authors are tackling the topic of drawing very large graphs, which are of chief concern for the topic of this thesis: graphs of hundreds of nodes with more than 1000 edges.

There are, however, a couple drawbacks to the algorithm. The first is that the article is rather dated, and it is not known either how widely the algorithm has been accepted (i.e., implemented in other applications or systems), nor if the ideas have been improved or refuted. The second drawback is that while the algorithm does indeed draw some very pretty graphs, navigating those graphs for the purpose of understanding its information (i.e., viewing event traces) is not covered in the paper at all.

C. TRANSITIVE REDUCTION

Aho, Garey, and Ullman (1972) gave an effective definition for the transitive reduction of a graph in their seminal paper:

> Given a directed graph G, G^t is said to be a transitive reduction of G provided that:
>
> (i) there is a directed path from vertex *u* to vertex *v* in G^t if and only if there is a directed path from *u* to *v* in G, and
>
> (ii) there is no graph with fewer arcs than G^t satisfying condition (i).

Given a directed graph with nodes labeled A, B and C, and with edges [A,B], [A,C] and [B,C] as shown in Figure 14, since there exists a path from A to B and from B to C, the edge [A,C] may be removed, thus transitively reducing the graph.

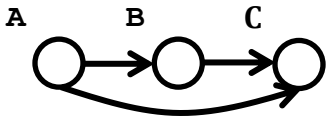

Figure 14. Transitive reduction example

Given both the complexity and size of graphs produced by MP, it is not surprising that there are a number of redundant arcs within those graphs.

1. Relationship Between Transitive Closure and Transitive Reduction

Aho et al. (1972) showed that there is a direct correlation between finding the transitive closure of a directed graph and finding the transitive reduction of a directed graph. After a series of proofs of six lemmas and four theorems, the authors show that the time complexity of the best algorithm for finding the transitive reduction of a graph is the same as the time to compute the transitive closure of a graph (or to perform Boolean matrix multiplication).

2. Complexity

A rather straight-forward algorithm for affecting the transitive closure of a directed graph would be something along the following lines (taken from http://stackoverflow.com/questions/1690953/transitive-reduction-algorithm-pseudocode):

```
foreach x in graph.vertices
   foreach y in graph.vertices
      foreach z in graph.vertices
           delete edge xz if edges xy and yz exist
```

Such an algorithm is neat and simple. However, it does not take much analysis to realize that such simplicity comes at a very heavy cost: the algorithm grows quite rapidly. Simple analysis of the above algorithm gives an approximate growth rate of $O(n^3)$.

3. The Floyd-Warshall Algorithm

A complete solution to the problem is presented in the outstanding text by Cormen, Leiserson, Rivest and Stein (2009). Chapter 25 of the Cormen et al. book introduces the Floyd-Warshall algorithm. The algorithm is designed to find the shortest path between all pairs of nodes in an adjacency matrix, with weighted edges between paths (Cormen et al., 2009, pp. 693–699):

```
n = W.rows
D(0) = W
for k = 1 to n
   let D(k) = d(k)ij be a new n x n matrix
   for i = 1 to n
      for j = 1 to n
         d(k)ij = min d(k−1)ij, d(k−1)ik + d(k−1)kj
return D(n)
```

where W = an n x n weighted-adjacency matrix for a directed graph with n nodes, and $D^{(n)}$ is the adjacency matrix of all-pairs shortest-paths of W.

Although MP produces DAGs with non-weighted edges, the algorithm can still be used for finding the transitive closure of a DAG with only slight modifications. The first step is to assign a weight of 1 to all edges of the graph; with these weights, if there exists a path between i and j, $d_{ij} < n$, and $d_{ij} = \infty$ otherwise. In the second step, the

min() operation is replaced with logical OR (∨) and the + operation is replaced with logical AND (∧); with these changes, if the exists a path between i and j, $t_{ij} = 1$, and $t_{ij} = 0$ otherwise (Cormen et al., 2009):

```
n = |G.V|
let T(0) = t(0)ij be a new n x n matrix
for i = 1 to n
   for j = 1 to n
   if i == j or (i,j) ϵ G.E
       t(0)ij = 1
   else
       t(0)ij = 0
for k = 1 to n
   let T(k) = t(k)ij be a new n x n matrix
   for i = 1 to n
      for j = 1 to n
         t(k)ij = t(k−1)ij ∨ t(k−1)ik ∧ t(k−1)kj
return T(n)
```

where |G.V| = the number of vertices of an n x n DAG, G.E is the set of edges of G, and $T^{(n)}$ is the transitive closure of G.

This algorithm has been proven to grow at a rate of $\Theta(n^3)$, which is quite similar to the algorithm described in section D.2. The primary advantage of this algorithm is that because the transitive-closure algorithm uses Boolean values rather than integers, its space requirement is limited to just the size of a Boolean.

As was discussed earlier, there is a direct correlation between the transitive closure of a graph and its transitive reduction. The algorithm for finding the transitive closure of a graph above is actually the opposite of what we want to find. Rather than finding the shortest path between two points, we want to eliminate them. Exchanging logical ANDs with logical ORs from the algorithm above will yield the desired graph:

```
n = |G.V|
let T(0) = t(0)ij be a new n x n matrix
for i = 1 to n
   for j = 1 to n
   if i == j or (i,j) ϵ G.E
```

```
        t_ij^(0) = 1
    else
        t_ij^(0) = 0
for k = 1 to n
    let T^(k) = t_ij^(k) be a new n x n matrix
    for i = 1 to n
        for j = 1 to n
            t_ij^(k) = t_ij^(k-1) ∧ !(t_ik ∧ t_kj)
return T^(n)
```

where |G.V| = the number of vertices of an n x n DAG, G.E is the set of edges of G, and $T^{(n)}$ is the transitive reduction of G.

D. GRAPH PLANARIZATION

The readability of a graph (and hence the ability to derive meaning from looking at a graph) often boils down to the confusion (or lack thereof) created from edges crossing over each other. Reducing the number of edge-crossings within a graph is crucial to a graph's ability to communicate with the viewer. In mathematical terms, this problem of joining nodes in a graph such that no two edges cross is known as planarization. Given the high number of nodes and edges which the Monterey Phoenix application produces, it is especially important to planarize each graph in order to make the graphs more meaningful than ordinary text files which are organized into rows and columns of a matrix. In particular, since there are many ways to draw any particular graph, is there a way (or several ways) to draw it such that no two of its edges cross? Alternatively, if it can be shown that no graph can be drawn such that no two edges cross, how can the graph be drawn with the minimum number of crossings?

Much work has been done in the field of graph planarization. A recent article, by Chimani and Gutwenger (2011), does much to first summarize the most important advances in planarization methods over the last 40 years, as well as present a new breakthrough in the subject. In this paper the authors review the manipulation of graphs (to include planarization) all the way back to a paper by Tarjan (1972). Later, Batini, Talamo and Tamassia (1984) introduced an algorithm which incidentally dealt with

minimizing the number of edge crossings within a graph: given a graph G = (V, E), the algorithm presented by Batini et al. runs in two phases, in which a planar subgraph G′ = (V, E′) is first found, then the temporarily removed edges are reinserted, one at a time, each time solving a single-edge insertion problem (i.e., reinserting each edge with the minimum number of crossings). The single-edge insertion problem is essentially solved by introducing temporary pseudo-nodes into the graph, looking for connections with minimum crossings, and then removing the pseudo-nodes.

From the point of view of the Chimani and Gutwenger paper, all improvements in the field of graph planarization have essentially been improvements on the 1984 algorithm of Batini et al. While the algorithm of Batini et al. only considered fixed embeddings within planar graphs, Gutwenger, Mutzel and Weiskircher (2005) found a linear-time algorithm which finds the optimal reinsertion path of an edge over all possible planar embeddings of G′ subgraphs. The 2005 algorithm by Gutwenger et al. used so-called SPQR-tree data structures, an important concept for Chimani and Gutwenger's 2011 algorithm. In 2009, Gutwenger et al. further demonstrated a (very complicated) insertion algorithm for inserting pseudo-nodes into G′, which optimally insert a vertex, along with all of its incident edges, and considers all possible embeddings (Chimani, Gutwenger, Mutzel, & Wolf, 2009).

From this background, the 2011 paper by Chimani and Gutwenger presents a simplification of a method presented at the Symposium on Discrete Algorithms by Chuzhoy, Makarychev, and Sidiropoulos (2011) in which, after finding the largest planar subgraph G′ of a graph G = (V, E), multiple edges are re-introduced (not one at a time, as in the single-edge insertion problem) to find a drawing of G with the fewest number of edge crossing and in the least amount of time. Although the authors admit that method described Chuzhoy et al. is more correct, they present their simplification from a practical point of view (since it was shown by Ziegler (2001) that simultaneously inserting an arbitrary number of edges into a planar graph is NP-hard, the authors do not believe that algorithm by Chuzhoy et al. is practical).

S subgraphs represent simple cycles, P subgraphs represent two vertices with multiple edges between them (parallel components), and R subgraphs represent tri-connected graphs. Since the algorithm hinges around reinserting single edges (one or more at a time), the reintroduction of Q edges is trivial and are not considered[4]. The exact mechanics of how to find and construct the S, P and R nodes of a graph are beyond the scope of this paper, as are the steps taken to re-assemble them into a planar SPR-tree. Representative drawings of the respective stages and compositions of an SPR-tree are given below (taken directly from the article). Figure 15 depicts a planar graph in the top left, its SPR-tree in the top right, and its decomposed subgraphs at the bottom. Virtual edges are depicted among the subgraphs by dotted lines:

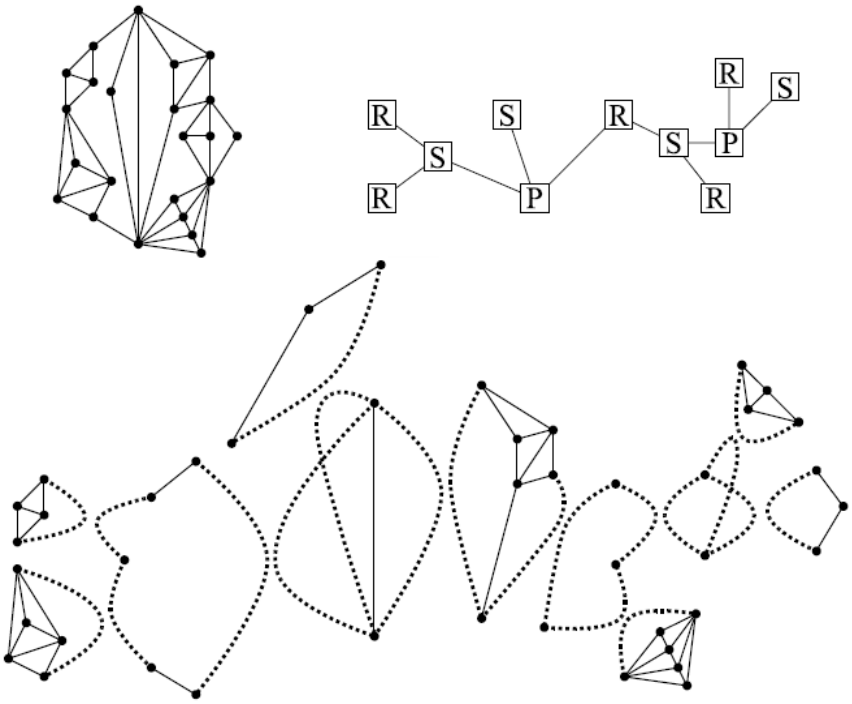

Figure 15. Example of S, P and R nodes

Figure 15 only depicts the final candidate nodes which will be used to re-assemble the planar graph shown at the top left. Many other virtual edges were considered by the algorithm before finding the ideal virtual edges depicted above.

[4] A more complete introduction to SPQR trees is presented in Appendix B.

Finding these ideal virtual edges is the whole point of Chimani & Gutwenger (2011). While algorithms for considering each candidate virtual edge one-at-a-time have been around for quite some time, the approach by Chimani & Gutwenger (2011) speeds things up considerably by considering several candidate edges at a time.

The authors present a number of (rather confusing) charts showing the efficacy of their algorithm. They ran their algorithm head-to-head with eight other algorithms, some of which were faster (but not as correct), and some which were more correct (but not nearly as fast). If their charts are to be interpreted correctly, their results are quite impressive, showing improvements from 300-3100%. All nine algorithms were run against a variety of well-known graph libraries, including the Rome Graphs, the AT&T Graphs, the ISCA Graphs, and the KnownCR Graphs libraries. The tests were run after some optimizations on the various graph sets. For instance, self-loops parallel edges and planar blocks were removed from all the graphs (elements of graphs which have no relevance to planarization).

All the current research can still be applied to graphs produce by Monterey Phoenix, albeit with additional steps added to the process (since graphs produced by MP are strictly acyclic and directed): 1) the directions of each of the edges must be removed (in order to make the graph undirected and to introduce cycles), 1a) the direction of each edge must be remembered, 2) the direction of each edge must be reintroduced after the planarization step is complete.

There is one more complexity which MP graphs introduce – trace graphs produced by MP usually contain edges of two types, one to indicate inclusion, and another to indicate precedence. When planarizing, since it is usually not possible to construct a completely planar graph any way, it would be ideal to give precedence to those edges, which indicate precedence (so that at least the order of events can be easily read and understood).

E. THE GRAPH MARKUP LANGUAGE

After a search of 25 databases[5], two relevant articles were discovered which specifically covered the subject of the Graph Markup Language (GraphML). GraphML is apparently widely used as there were numerous articles that made reference to the technology, as well as a number of reports on systems and applications that use GraphML, but there is next to nothing about the subject itself.

The first article that was discovered was written by Ulrik Brandes, Markus Eiglsperger, Ivan Herman, Michael Himsolt, and M. Scott Marshall, and was titled *GraphML Progress Report: Structural Layer Proposal*. As its title suggests, it is a status report on the construction of the GraphML schema by the authors of the schema. There was not much relevant information in the article as GraphML was still under construction – it is only mentioned here because of the dearth of any other writing on the subject. A second article was written by Brandes and Pich (2005) and is covered in detail in the following paragraphs.

The goal of the Brandes & Pich paper seems to be a plug for the benefits of applying transformations to XML data, especially transformations that result in documents which can be consumed by graphing applications. As one of the authors of the GraphML schema, Ulrik Brandes naturally focuses on GraphML as the target language of transformations from the paper.

The article provides some background information on XML, XSLT and GraphML. This background information is quite brief and could not possibly be used as an acceptable introduction to the aforementioned technologies. Instead, the article focuses on the purposes of XML, XSLT and GraphML, along with their gross architectures, as well as a description of how the three work together to accomplish the goal of producing a graph from an XML document.

Unfortunately, the article is from 2005, which is a bit dated. On January 23, 2007, XSLT 2.0 and XPath 2.0 became W3C recommendations. As the information presented in this article is strictly targeted to XPath 1.0 and XSLT 1.0 transformations, the paper

[5] The list of databases searched is presented as Appendix C.

only mentions the methods XPath uses to traverse the Document Object Model (DOM) of XML documents. XPath 2.0 does not use the DOM. Everything in XPath 2.0 is a sequence, and sequences differ significantly from the data structures provided by the DOM. However the article is still relevant since all web browsers, as well as a majority of all other XML consumers/producers, still rely on XSLT 1.0 and XPath 1.0. The only applications which are purely XSLT/XPath 2.0 compliant are the modern XSLT processors themselves; even the XSLT processors are often run in XSLT/XPath 1.0 mode, or even process 1.0 elements in line with 2.0 documents.

The article also provides some very good insight as to how to use XML, XSLT, and GraphML with Java. Roughly 25% of the paper is dedicated to the discussion of XSLT and GraphML Java bindings. The authors discuss Java classes for graph data structures and algorithms, wrapper classes, and stylesheet (XSLT) the instantiate wrappers and communicate with them.

F. ALTERNATIVES TO THE GRAPH MARKUP LANGUAGE

GraphML is certainly not the only markup language for describing graphs. Alternatives are available, both in XML formats and in plaintext. A few of the more well-known alternatives are presented here. This list is by no means comprehensive, but rather it is intended to reflect the multitude of options available for describing graphs in either textual or XML-based formats.

1. Graph Modelling Language (GML)

The GML file name is quite ambiguous, as the same three letters are also used to describe the Geography Markup Language, the Game Marker Language, and the Graffiti Markup Language (used by law enforcement investigators), all of which have larger followings than the Graph Modeling Language. While the three aforementioned languages all represent branches of XML, the Graph Modelling Language does not – files written to the GML standard are formatted text files. GML is sometimes also referred to as Graph Meta Language.

The Graph Modelling Language was invented by Michael Himsolt of the Universität Paßau in Passau, Germany. It was designed as a portable file format for exchanging graph information (Himsolt, 1995). Like GraphML, generic GML files are quite simple. GML supports graphics and other external data, which is also similar to GraphML. However, unlike GraphML, GML files support the use of comments, as demonstrated in Figure 16.

```
graph [
     comment "This is a sample graph"
     directed 1
     id 42
     label "Hello, I am a graph"
     node [
          id 1
          label "Node 1"
          thisIsASampleAttribute 42
     ]
     node [
          id 2
     ]
     edge [
          source 1
          target 2
          label "Edge from node 1 to node 2"
     ]
]
```

Figure 16. Sample GML document

A short list of applications that support the GML file format (Wikipedia contributors, 2012):

- Clarib
- Cytocsape
- Gephi
- Graph-Tool
- igraph
- NetworkX
- ocamlgraph

- OGDF (Open Graph Drawing Framework)
- Tulip
- yEd

2. eXtensible Graph Markup and Modeling Language (XGMML)

XGMML is an XML file format based on the GML format. It was designed to provide a 1:1 representation of GML to XGMML data elements in order to provide maximum compatibility. Figure 17 provides an example XGMML document, based on the example GML document from section F.2:

```
<?xml version="1.0"?>
<graph directed="1" id="42" label="Hello, I am a graph">
<node id="1" label="node 1" />
<node id="2" />
<edge source="1" target="2" label="Edge from node 1 to node 2" />
</graph>
```

Figure 17. Sample XGMML document

A short list of applications that support the XGMML file format (Wikipedia contributors, 2009):

- Cytoscape
- Biomax BioXM Knowledge Management Environment
- JNets

3. DOT Language

Like GML, DOT language is a text-based graph description language that is not compatible with XML. The DOT language is very simple, even simpler than the GML, as it allows nodes to be defined on the fly, while edges are simply represented as dashes and angle braces. For instance, `a--b` represents a non-directed edge between two nodes, `a` and `b`, while `c -> d` represents a directed edge from a node `c` to a node `d`. The DOT language also supports comments and attributes. A sample DOT document is presented in Figure 18.

```
graph graphname {

       // This attribute applies
       // to the graph itself
       size="1,1";

       // The label attribute can be
       // used to change the label of a node
       a [label="Foo"];

       // Here, the node shape is changed.
       b [shape=box];

       // These edges both have different
       //line properties
       a -- b -- c [color=blue];
       b -> d [style=dotted];
}
```

Figure 18. Sample DOT document

A short list of applications that support the DOT file format (Wikipedia contributors, 2012):

- Graphviz
- Canviz
- Grappa
- Beluging
- Tulip
- OmniGraffe
- ZGRViewer
- VizierFX
- Gephi

It is interesting to note that DOT files often use the .gv file extension as the .dot file extension is used by the Microsoft Office suite of applications to describe template files.

4. Graph eXchange Language (GXL)

GXL is an XML-based language designed to be used for exchanging graph information. It was designed primarily to support the re-engineering of software by 1) describing typed, attributed directed graphs, and 2) by representing instance data as well as data structures. The language was developed and supported by a number of partners, to include (Heinen, 2008):

- Bell Canada (Datrix Group)
- Centrum voor Wiskunde en Informatica, The Netherlands (Interactive Software Development and Renovation and Information Visualization)
- IBM Centre for Advanced Studies, Canada
- Mahindra British Telecom, India
- Merlin Software-Engineering GmbH, Germany
- Nokia Research Center, Finland (Software Technology Laboratory)
- Philips Research, The Netherlands (Software Architecture Group)
- RWTH Aachen, Germany (Department of Computer Science III)
- TU Berlin, Germany (Theoretical CS/Formal Specification Group)
- University of Berne, Switzerland (Software Composition Group)
- University Bw München, Germany (Institute for Software Technology)
- University of Edinburgh, UK, (Edinburgh Concurrency Workbench)
- University of Koblenz, Germany (GUPRO Group)
- University of Oregon, USA (Department of Computer Science)
- University of Paderborn, Germany (AG Softwaretechnik)
- University of Stuttgart, Germany (BAUHAUS Group)
- University of Szeged, Hungary (Research Group on Artificial Intelligence)
- University of Toronto, Canada (Software Architecture Group)
- University of Victoria, Canada (RIGI Group)
- University of Waterloo, Canada (Software Architecture Group)

GXL files are quite a bit more involved than the GraphML alternatives discussed so far. A (simple) sample GXL document is presented in Appendix D. It is not supported by any of the graphing applications that were considered to support either the Monterey Phoenix program or the MPGrapher application.

G. YED AND YED ALTERNATIVES

Like the GraphML file format, which has a multitude of available alternatives, there are also a host of applications that can be used to consume those file formats to produce graphs. A visit to AlternativeTo (AlternativeTo contributors, 2012), a website that collects and lists applications which present similar features, reveals that there are no less than 20 alternatives to the yEd graphing application, and the list is sure to grow larger in the near future. Some of the more well-established programs, especially those that are free, easy to use, and support XML (especially GraphML) files, are discussed in this section. A note on the definition of "easy to use": for the purposes of this study, easy to use will be taken to mean that the software:

- Provides one-click re-arrangement of graph layouts
- Provides a means to navigate and zoom graphs
- Runs on all platforms (Windows, Mac and Linux)
- Uses XML which can be analyzed and reproduced from outside sources

1. LucidChart

LucidChart is a web-based graphing application which is free, provided one subscribes to a one-user license and forgoes most useful features (such as web-support, Visio import/export, and saving files). The primary benefit of LucidChart is that it is a cloud-based, online collaborative tool. For instance, two users, one in San Francisco and one in Stockholm, can open up one document, make changes to it, and (nearly) instantly see the changes that each makes from the other side of the world. LucidChart is quite easy to use in some respects: the authors provide a host of templates that one can start from, and modifying charts is as easy as dragging nodes around the screen. Regretably there is not an option to change the layout to a pre-defined layout with just one click. The application also lacks a navigation pane for easily zooming and panning large graphs.

LucidChart supports export of image files and imports and exports Visio Files (for a fee), but it does not support any known XML file format.

2. Dia

Dia is a free, open-source diagramming tool which was developed as a scaled-down version of Microsoft Visio. Although the Dia website touts its Visio-like features, the application seems to be geared entirely towards producing UML diagrams.

Unlike LucidChart, Dia does not run on the web via cloud services, but it does run on all platforms.

Dia saves files with a .dia extension. Although the .dia format is XML, it is compressed XML which is not readily available for analysis or reproduction.

Dia is rather powerful and quite easy to use, at least in terms of creating new graphs from scratch and modifying existing graphs. However, there is no navigation pane to easily zoom and pan large graph, nor are there any pre-defined layouts available.

One of the major drawbacks of the program is that one cannot simply download an installer to set up the program. One must instead download source files and compile them into an executable.

3. Gephi

According to the Gephi homepage, "Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. Runs on Windows, Linux and Mac OS X. Gephi is open-source and free." Gephi is more of a visualization and analysis tool than a graph production tool. For instance, Gephi has three different modes in which it can run: Overview mode, Data Laboratory mode, and Preview mode. The Overview mode is designed for analyzing networks that are laid out as graphs. Data Laboratory mode lays graphs out in table format and provides all sorts of tools for adding, deleting, copying, merging and modifying attributes of both nodes and edges. Lastly, the Preview mode is designed to actually display graphs.

While it may sound quite powerful, Gephi is actually extremely slow and cumbersome. There are no layouts from which a user may choose – there is just one default layout. Nodes cannot be created by drag-and-drop, nor can edges be created by

dragging between nodes – one must use the Data Laboratory mode to add edges and nodes. Also, it is actually quite difficult to get the program to actually display a graph. From the Preview mode, one must find the right combination of changing properties in propertied window and hitting the refresh button. There is also no navigation pane for panning and zooming large graphs.

One of the most interesting benefits of Gephi is that is supports GraphML. Additionally, unlike yEd, generic GraphML files (i.e., no layout information included in `<data>` elements) opened in Gephi are laid out with default properties, which means that nodes are not stacked one-on-top of the other and all labels are visible.

4. Microsoft Visio

Visio is generally considered to be the granddaddy of all graphing software – every graphing application available today compares itself to Visio. Visio is indeed very powerful, however, like all powerful software, it is also very tedious. There are probably more template libraries available for Visio than for all other graphing applications combined. Also, as part of the Microsoft Office suite of applications, there are very powerful automation and animation tools available to Visio users.

Grand as Visio may be, it does not suit the needs of the current project. First, the product is not free, with legitimate (non-pirated) copies of the most basic version retailing for about $200. Second, while the default file format for Visio documents is XML, a quick look at the resulting XML documents will reveal that the vast majority of the information contained in those XML documents is binary data. Also, Visio does not easily run on all platforms. While separate (nearly identical) versions are available for Mac and Windows, the software will not run on Linux platforms outside of virtual machines.

Despite its drawbacks, Visio does come with some additional benefits that would be very welcome in other graphing applications. First, although it is not obvious how to do so, Visio automatic graph layouts with just a few click of the mouse. Also, not only does the program offer a navigation pane for panning and zooming, it utilizes some very useful additional mouse wheel operations as well:

- To pan up and down, simply roll the mouse wheel up and down
- To zoom, hold down the Ctrl key and roll the mouse wheel up and down
- To pan left or right, hold down the Shift key and roll the mouse wheel, up to pan left, down to pan right

5. The yEd Graphing Application

The program chosen to layout, manage and display graphs for Monterey Phoenix is the yEd graphing application by yWorks. yEd runs on the Java virtual machine and is therefore available to all platforms. yEd offers all of the features which were desired in displaying and manipulating graphs for the Monterey Phoenix program:

- It is free
- With only a few clicks, users can change the layout of any graph to one of 13 different pre-defined layouts
- It provides a navigation pane from which one can pan and zoom
- Its default file format is XML (GraphML specifically)

yEd also offers a fish-eye lens, with which one can zoom individual parts of a graph by simply hovering with the mouse. Additionally, yEd supports built-in algorithms which implement solutions to many of the issues discussed in sections B–D (for example, yEd is very adept at planarizing graphs).

a. Drawbacks of yEd

While yEd supports all of the requirements for displaying and manipulating graphs for MP, it does have a number of shortfalls:

(1) Keyboard shortcuts are not customizable.

(2) It is not possible to alter the staking order of objects – edges may obscure labels, labels may obscure nodes, etc.

(3) Nodes without defined `<y: x="" y="" />` elements are opened one-on-top of each other in a single, overlapping stack.

(4) Clicking and dragging nodes can be quite irritating – rather than dragging a node as one would expect, one instead creates a new edge which zooms out into the middle of nowhere. This behavior might be a boon to the creating of graphs

from scratch, but it is an excruciating nuisance when it comes to manually manipulating graphs for analysis.

b. Brief yEd Tutorial

Figure 19 displays a screenshot of yEd, features of which will briefly be presented in the coming sections.

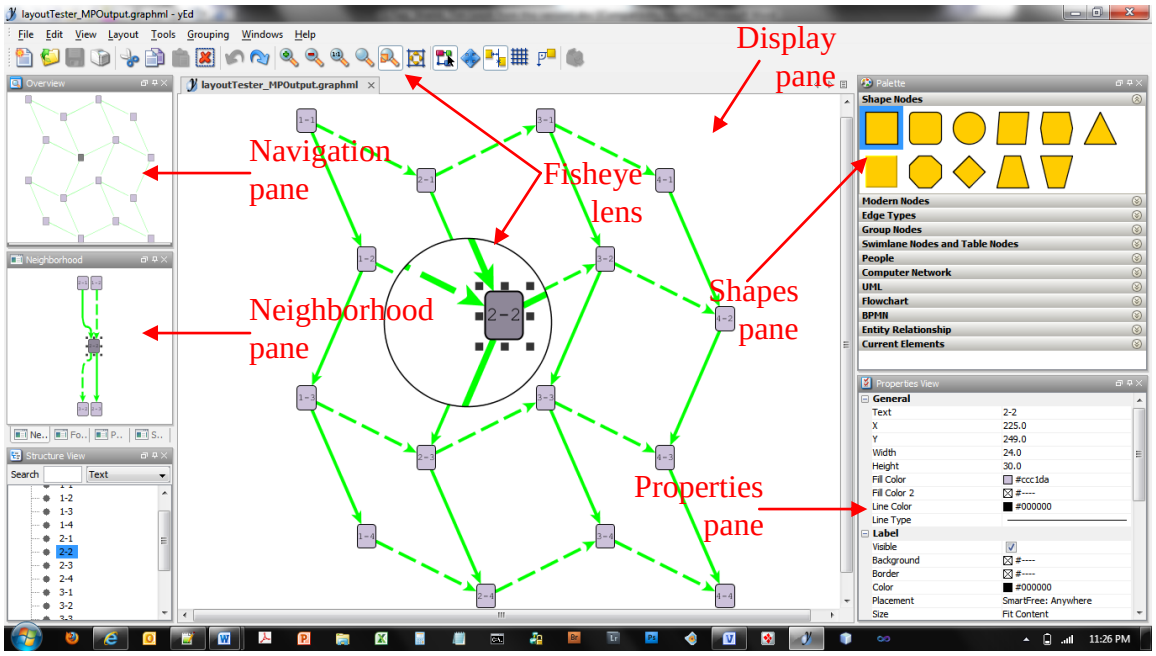

Figure 19. Screenshot of the yEd graphing application

(1) Display Pane. The display pane is where all graphs are displayed. Nodes are selected by double-clicking on them. Edges are selected in the same way. To move a node one must first double-click the node, and then drag it – simply clicking and dragging a node will result in the production of a new edge with that particular node as the source node.

(2) Shapes Pane. yEd provides a "Palette" window, which this thesis refers to as the shapes pane. From this window a user can build graphs from scratch by either clicking on a shape and dragging it to the display pane or by simply double-clicking on a shape in the shape pane – from then on any single click in the display pane will result in the production of a new node of that shape. Similarly, one may create an edge by dragging an edge type to a node, or one may double-click on an edge

type, and any edge created thereafter will be an edge of that particular type. Edges are created by clicking on a source node and dragging to a target node. Figure 20 shows a close-up detail of the shapes pane.

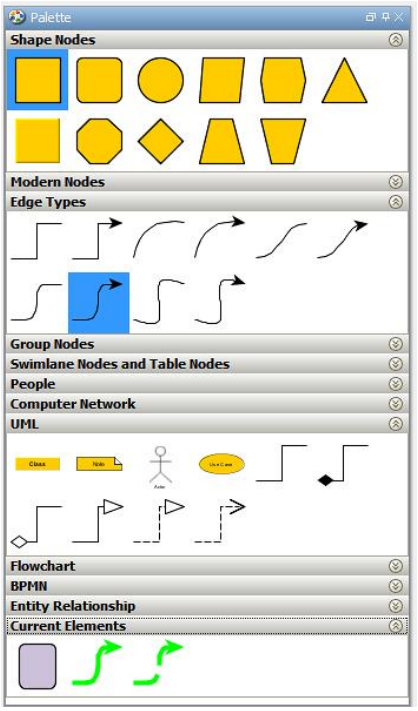

Figure 20. Shapes pane

As can be seen in Figure 20, many different types of nodes and edges are available out-of-the box. In addition to the shape types displayed in Figure 20, yEd also supports importing custom shapes. Custom shapes may be images, Scalable Vector Graphics, or Visio XML shape files.

(3) Properties Pane. yEd provides a "Properties View" window that this thesis will refer to as the properties pane. From the properties pane, most attributes that pertain to nodes, edges, or entire graphs (to include labels) may be manipulated from the properties pane. Sadly, not all possible properties may be set in the properties pane. For instance, while existing edge labels may be manipulated from the properties pane, in order to create a label in the first place one must right-click the edge of interest and select "Add label" from the context menu.

All of the properties that are available in the properties pane are also available by right-clicking on a shape and selecting "Properties" from the context

menu. Additionally, by single-clicking on a shape and pressing the f6 key, one can bring up a similar properties dialog.

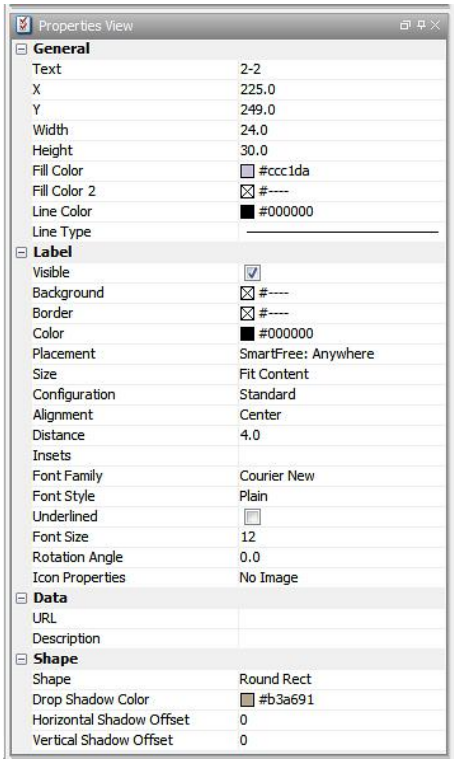

Figure 21. Properties pane

(4) Navigation Pane. yEd provides an "Overview" window, which this thesis refers to as the navigation pane. The navigation pane operates very much like similar tools in other graphing applications (such as the "Pan & Zoom" window in Microsoft Visio). A brief inspection of Figure 22 will reveal that there are two areas to the navigation pane: a larger, gray area, and a smaller, white area. The gray area represents the entire graph that is currently being displayed, regardless of screen clipping due to the current zoom level. On the other hand, the white area reveals those portions of the graph that the program is actually able to display, based on the current zoom level. From this point, panning the graph is a simple matter of clicking in the white portion of the navigation pane and dragging it around – the portion of the graph that yEd displays in the display pane directly correlates to the location and size of the white area of the navigation pane. Additionally, the red arrow in Figure 22 points to a small black dot.

Changing the zoom level of the display pane may be accomplished as simply as clicking on this small black dot and dragging up to zoom in or down to zoom out.

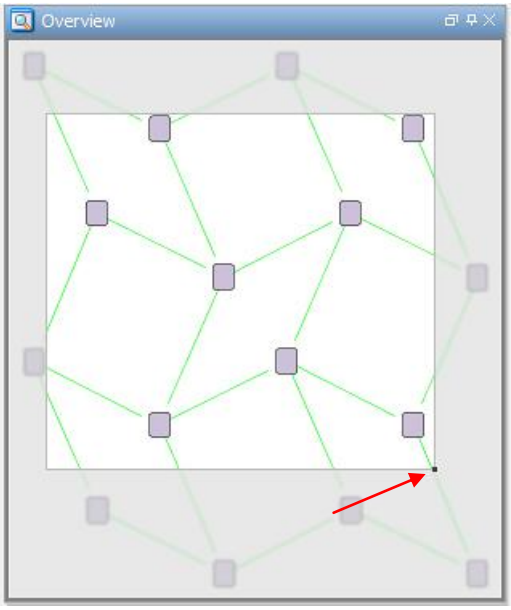

Figure 22. Navigation pane

The navigation pane is an extremely useful tool for panning and zooming around large graphs with hundreds of nodes and thousands of edges.

(5) Neighborhood Pane. yEd also provides a "Neighborhood" window. This tool is another extremely useful tool for navigating very large graphs. The author of this thesis has not seen a tool quite like this in any other graphing application.

By selecting a node in the display pane, that same node is also displayed in the neighborhood pane, along with each node and corresponding edge that is connected to that particular node. Similarly, clicking on an edge in the display pane reveals the edge in the neighborhood pane, along with each node that is attached to that particular edge. Additionally, once displayed in the neighborhood pane, clicking on edges or nodes in the neighborhood pane also selects the same edge or node in the display pane. It is also possible to select multiple edges or multiple nodes.

Unfortunately, selecting multiple nodes reveals only the selected nodes along with additional connected nodes and edges – the neighborhood pane does not reveal the paths between the selected nodes. Also, it is not possible to select a mixture of

nodes and edges to display in the neighborhood pane – one may either select nodes, or edges, but not both.

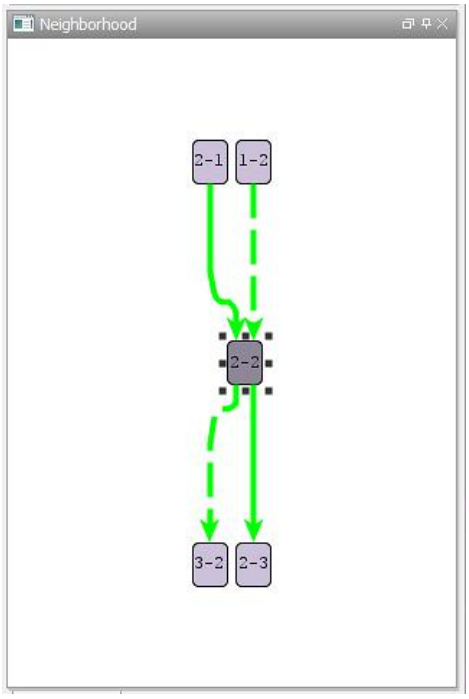

Figure 23. Neighborhood pane

(6) Fisheye Lens. yEd also provides a “Local Zoom” tool, which this thesis refers to as the fisheye lens.

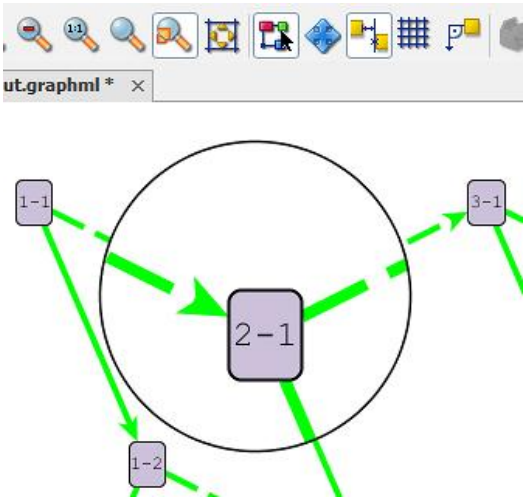

Figure 24. Fisheye lens

As Figure 24 shows, the fisheye lens is basically an additional cursor type, one which happens to display a zoomed-in view of whatever local area the cursor may happen to be over. While the fisheye lens is a great idea in principle, in practice it is not all the useful. The zoom level of the lens is fixed and cannot be modified by the user. When using the lens on a very large graph that is zoomed out to 25%, the lens reveals an area that is only zoomed in to about 40%, which is still too small to be seen.

(7) Graph Types. While all of the aforementioned tools contribute to make yEd an outstanding graphing application, the true power of yEd is borne out by its ability to effortlessly change the layout of any graph with just a few mouse clicks.

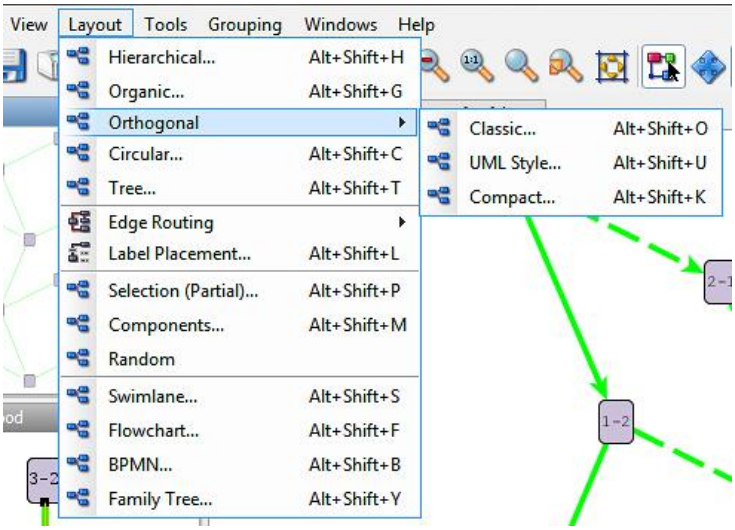

Figure 25. Graph types available in yEd

As shown in Figure 25, yEd offers 12 graph layout patterns (Edge Routing, Label Placement, Selection (Partial) and Components are not actual layouts, but options for other layouts). Clicking on any of the layout styles brings up an options box similar to Figure 26. yEd provides no less than 37 different layout options, which are logically grouped into six different panes. While adjusting these layout options provides the user with rather precise control, accepting the default values is usually sufficient for most graphs.

For the output from the Monterey Phoenix program, the Hierarchical, Orthogonal-Classic, Orthogonal-Compact and Tree layouts generally work

best. Figure 27 shows a sample starting graph that was compiled by the MPGrapher applications. Figures 28-31 show samples of each of the aforementioned layouts, with default layout values, as applied to the graph shown in Figure 27.

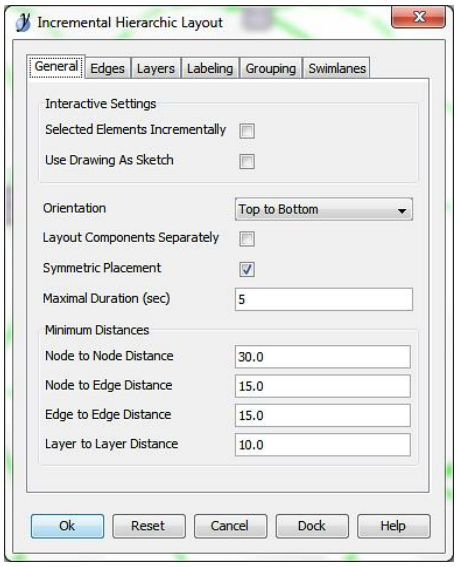

Figure 26. Layout options for the Hierarchal layout style

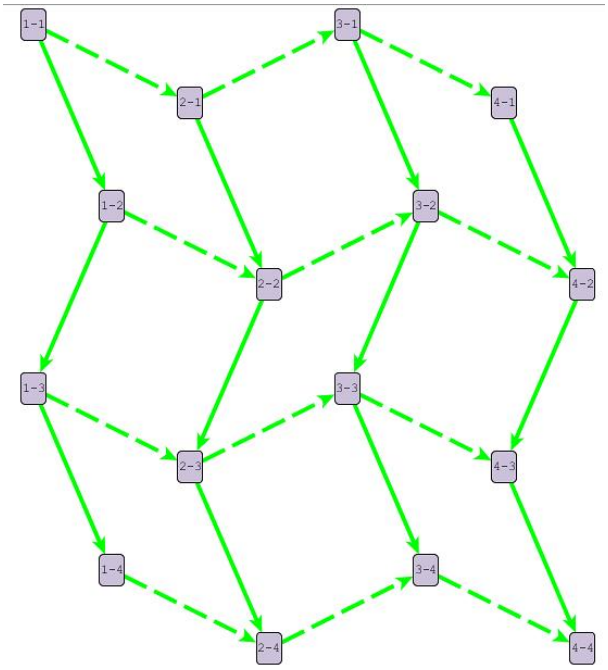

Figure 27. Representative graph

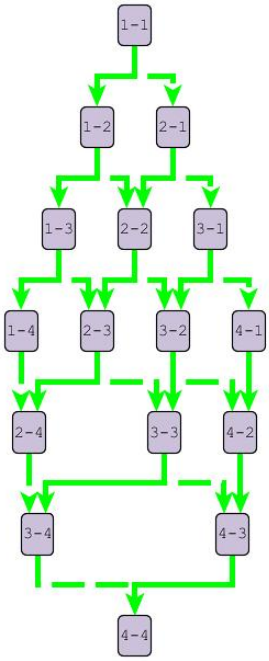

Figure 28. Hierarchical layout example

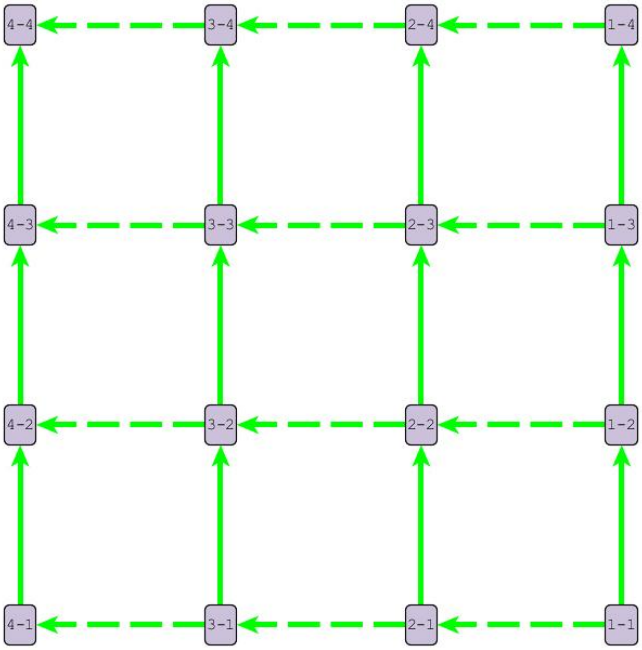

Figure 29. Orthogonal-Classic layout example

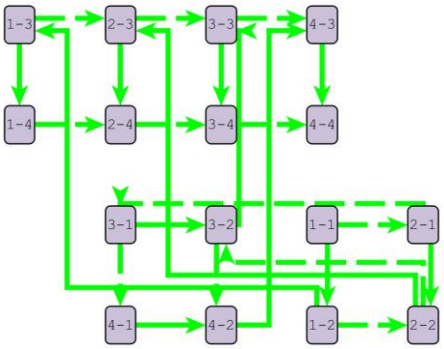

Figure 30. Orthogonal-Compact layout style

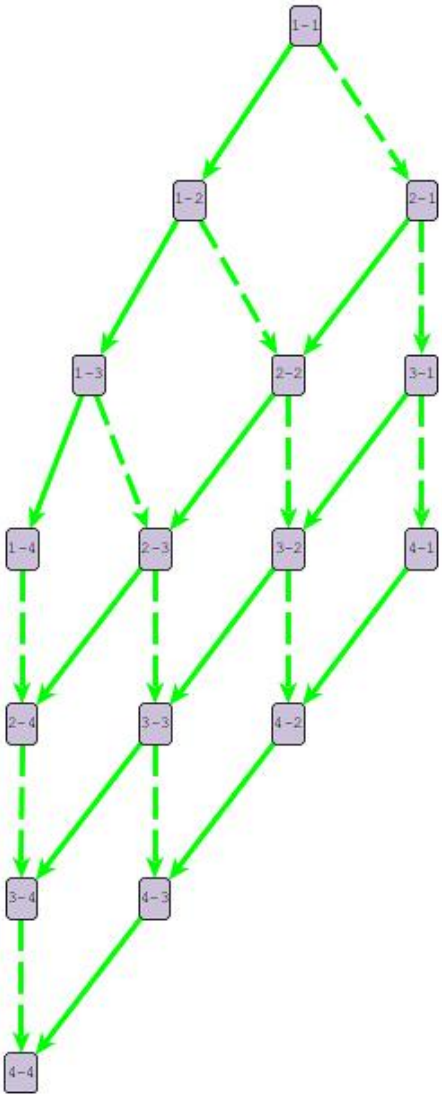

Figure 31. Tree layout example

H. SCANNING AND PARSING TOOLS OF THE STRING CLASS IN JAVA

A background discussion of a compiler application could not be complete without some sort of introduction to the tools used for parsing input documents. Traditional compilers usually use some variant of the age-old, tried-and-true Lex, or the newer Flex, or some other variant which is built upon those excellent parsers. It should come as no surprise that there are a number of parsers available for the Java programming language that are built upon Lex and Flex. One such parser that was investigated for the purposes of this project was the JFlex parser. It was soon discovered that the built-in parsing tools available to the Java String class easily make all Java-based imitations of Lex and Flex unnecessary. JFlex ultimately proved to be obsolete, not well maintained, difficult to install, and essentially more trouble than it was worth. This is probably true of most Java-based imitations[6] of Lex/Flex because built-in parsing tools in Java are so effective. Also, the parsing tools that are built-in to the String class in Java are available by default to all applications and are both well-maintained and very well-documented.

1. The Java Scanner Class

All implementations of Java have provided the `Scanner` class since version 1.5 of the language as part of the `java.util` suite of libraries. The `Scanner` class provides all the basic functionality of traditional Lex-based scanners. The `Scanner` class is able to parse input of all primitive types (integers, floats, Booleans and bytes, as well as `String` objects). A full description of the Scanner class is beyond the scope of this thesis, especially given that such excellent documentation is available online. However, a few remarks on the most often used methods are in order, especially the two that are used by the MPGrapher project.

a. The Scanner.hasNext() Method

The `hasNext()` is necessary for testing whether the scanner has reached the end of a given input file. Without this method, if a Scanner tried to retrieve

[6] JLex and CookCC are two other parser/scanner options available for the Java language.

information beyond the end of an input file, an "Out of bounds" exception would occur. Figure 32 shows an example of the `hasNext()` method:

```
…
File file = new File();
Scanner scan = new Scanner(file);
…
while (scan.hasNext) {
  String lexeme = scan.getNext();
…
```

Figure 32. Example of the `hasNext()` method

b. The Java Scanner.getNext() Method

The `getNext()` method is the workhorse of the class. This method returns the next lexical item of the input file, or an exception if there is no more input to parse. `getNext()` is roughly equivalent to the `yylex()` command in Lex/Flex. Figure 33 shows an example from the MPGrapher application:

```
…
protected final String advance() {

  //curTok is a private field of the GraphML_Compiler object
  //and is instantiated upon construction
  //fetch the next token from the input file and assign it
  //to the curTok variable
  curTok = scan.next();

  //return the current token to the calling method
  return curTok;
}
…
```

Figure 33. Example of the `getNext()` method

c. The Scanner.useDelimeter(String s) Method

The `useDelimeter(s)` method allows developers to precisely control the separators that are allowed between lexical items of a given input document. The method fully supports regular expressions. Figure 34 shows an example that mandates any whitespace character or semicolon as delimeters:

```
…
File file = new File();
Scanner scan = new Scanner(file);
scan.useDelimeter("[;\s]");
…
```

Figure 34. Example of the `useDelimeter(s)` method

2. Parsing Tools Built-in to the Java String Class

The `String` class in Java is a large and robust class, with more than 50 methods and constructors. A complete description of the class is outside the scope of this thesis, especially since is much documentation available online (e.g., docs.oracle.com). However, some of the more useful methods of the class, at least as they pertain to the MPGrapher project, will be introduced here, along with some brief examples from the MPGrapher package.

a. The Java String.endsWith(String s) Method

The `endsWith(s)` method is useful for testing the terminal character of a `String` object. For example (from the `edge()` method of the `GraphML_Compiler` class):

```
…
//test for the end of an edge
if (curTok.endsWith(END_EDGE)) {
  endEdge = true;
…
```

Figure 35. Example of the `endWith(s)` method

b. The Java String.equals(Object o) Method

The `equals(o)` method compares two objects and returns a Boolean true or false, depending on the result. Since all Strings are objects in Java, this is the preferred method for comparing character sequences of a `String` object. The method also fully supports regular expressions, making it a very powerful and often-used tool.

For example (from the `yEdNodes()` method of the `yEd_GraphML_Compiler` class – regular expressions are underlined for emphasis):

```
…
//yEdNode labels may contain multiple tokens, so
interpret
//everything between the current "-symbol and the next
//"-symbol as a single yEdNode label
if (curTok.startsWith("\"") && !curTok.equals("\"\""))
{

  //strip off the leading "-symbol before adding it to
  //the label String
  curLabel = curTok.substring(1, curTok.length());
…
```

Figure 36. Example of the `equals(o)` and `substring(i1, i2)` methods

c. The Java String.length() Method

The `length()` method is another often used tool. Rather than returning the size of a `String` object in terms of bytes, the `lengt+h()` method instead returns the number of characters of a `String` object, to include all whitespace characters. This method is often used, especially in conjunction with the `substring(i1, i2)` method, to build effective sub-Strings. For example, please see the last line of the Figure 36 (`substring(i1, i2)` and `length()` methods are italicized for emphasis).

d. The Java String.startsWith(String s) Method

The `startsWith(s)` method is quite similar to the `endsWith(s)` method, except that is tests the first character (or characters) of a `String` object rather than the last character. Like all the methods of the `String` class, `startsWith(s)` also supports regular expressions. For example (from the `findLabels()` method of the `yEd_GraphML_Compiler` class):

```
…
//if the current token starts with a " and ends with a
";",
//then it must be a one-word label
if (curTok.startsWith("\"")) {
…
```

Figure 37. Example of the `startsWith(s)` method

e. The Java String.substring(int i1, int i2) Method

The `substring(i1, i2)` method is another often-used tool, especially in conjunction with the `length()` method. The method takes two parameters, a start index (`integer`) that indicates where in the `String` the substring shoud start, and a length index (`integer`) that indicates how many characters after the start-character should be included in the substring. For example, please see the last line of Figure 36 (`substring(i1, i2)` and `length()` methods are italicized for emphasis).

THIS PAGE INTENTIONALLY LEFT BLANK

III. ENVIRONMENT, EXPERIMENTATION, AND INVENTION

A. COMPUTER ENVIRONMENT SPECIFICATIONS

All experimentation, design, development, and test were conducted on a Dell Inspiron 1564 laptop with the following configuration:

- Operating system: Windows 7 (with Service Pack 1)
  64-bit
- Processor: Intel i3 M330
  2.13 GHz
  Dual-core
  64-bit
  3 MB Intel Smart-cache
- RAM: 4 GB

B. APPLICATIONS

In addition to yEd, which is described in detail elsewhere, there were two other applications that were used extensively for experimenting with output files generated by yEd and for developing the Java code of the MPGrapher application.

1. Notepad++

Notepad++ is a non-open source text and source-code editing environment that is free to all Windows users – Mac and Linux versions are unfortunately not available. Notepad++ version 5.9.6.2 (UNICODE) was used for all XML development for the MPGrapher project. Notepad++ is designed to be a one-stop-shop for editing source-code of any sort, or from any environment, on the Windows platform. The application offers out-of-the-box line-numbering, syntax highlighting and code coloring for 54 different languages, platforms and styles (presented in Appendix E). All color-coding and syntax highlighting properties are fully customizable. Users are also able to define properties for their own languages and environments in addition to the numerous defaults provided by the program. Depending on the language or environment selected, Notepad++ also offers code hints and code completion (also fully customizable). Notepad++ also comes with a

wide-variety of XML tools, to include a built-in ftp client, XSL transformations, an XPath evaluator, and offers automatic XML validation. Since it is offered only for the Windows platform, and it therefore relatively little-known, the remainder of this section provides explanations and demonstrations of the features of Notepad++ that proved most useful for development of the MPGrapher application.

a. Code Coloring in Notepad++

The code-coloring and syntax highlighting Notepad++ offers a number of useful features that made it especially well-suited for experimenting with XML. The first is code coloring. While Microsoft Word and other text editors are more than capable of displaying XML, Notepad++ colors different parts of XML documents in ways that make it much easier to make sense of the information displayed in an XML document. For instance, Figure 38 shows a snippet of XML as it might appear in an ordinary text editor.

```
1  <?xml version="1.0"?>
2  <graphml xmlns="http://graphml.graphdrawing.org/xmlns"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
5      http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
6      <graph edgedefault="directed" id="G">
7          <node id="blah">
8              This is an example of some plain-text.
9          </node>
10         <node id="halb">
11         </node>
12         <edge id="blah::halb" source="blah" target="halb">
13         </edge>
14     </graph>
15 </graphml>
```

Figure 38. Plain-text XML

From the "Language" drop-down menu in Notepad++, one is able to select from a number of language, to include XML, as demonstrated in Figure 39.

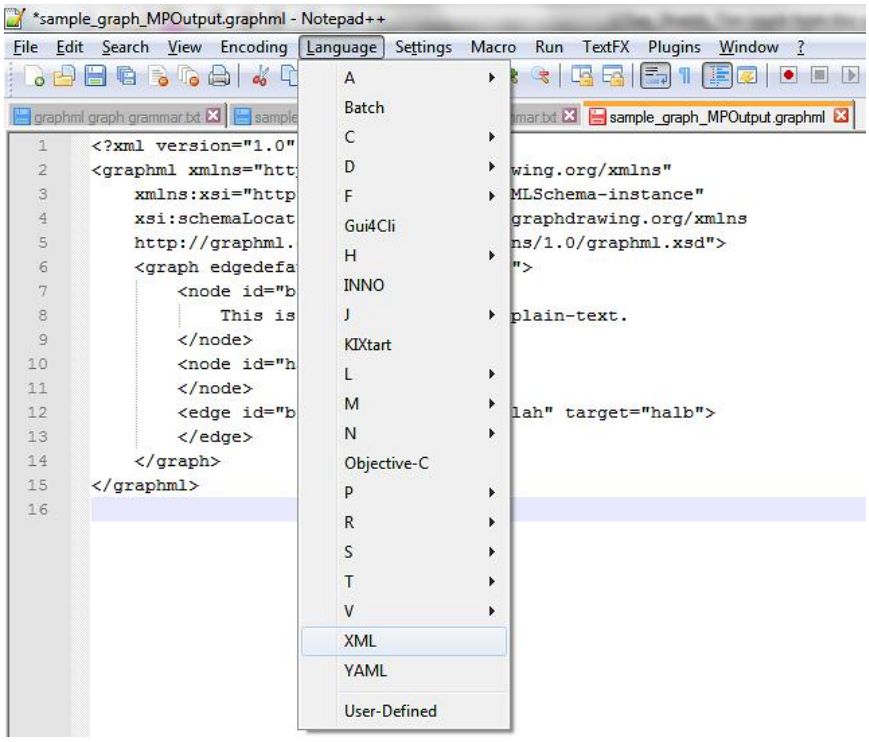

Figure 39. Language drop-down menu in Notepad++

The XML code snippet shown in Figure 38 is shown again in Figure 40 with the default XML style applied by choosing XML from the "Language" drop-down menu.

```
<?xml version="1.0"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
    http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
    <graph edgedefault="directed" id="G">
        <node id="blah">
            This is an example of some plain-text.
        </node>
        <node id="halb">
        </node>
        <edge id="blah::halb" source="blah" target="halb">
        </edge>
    </graph>
</graphml>
```

Figure 40. XML, with XML display-properties applied in Notepad++

b. Word Highlighting in Notepad++

Another useful tool in Notepad++ is word highlighting. By selecting a single word in an open document, Notepad++ highlights all other occurrences of that same word. In Figure 41, the last occurrence of the word graphml was selected, and all other occurrences are colored green. This is a quick and extremely useful search tool.

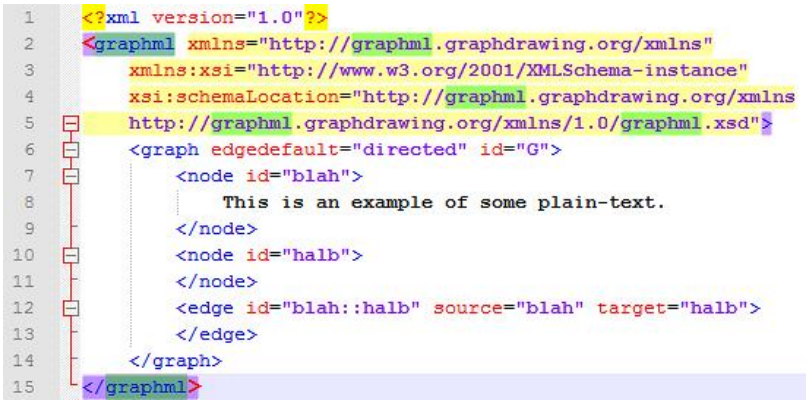

Figure 41. Word highlighting in Notepad++

c. Tagbegin-Tagend Highlighting in Notepad++

In Figure 41, one may also note that leading words of the `<graphml>` elements are highlighted with a purple background. This feature matches the opening and closing tags of any selected XML element. This feature may not seem that useful in a small XML snippet such at figures 41 and 42, but in large documents, where a single element may span several pages, this is an invaluable tool. Figure 42 shows another example of the tool.

Figure 42. Tagbegin-Tagend highlighting in Notepad++

d. The Text-compare Plugin of Notepad++

Probably the single-most useful tool in Notepad++, at least in terms of developing the MPGrapher application, is the Compare plugin. This tool works by comparing two seemingly identical documents and matching them, line by line, according to their differences. The tool is customizable to support formatting differences as well as textual differences. Figure 43 shows an example.

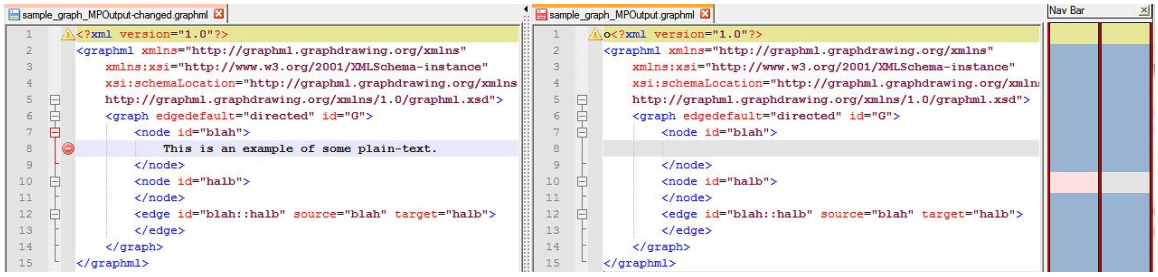

Figure 43. The Compare plugin in Notepad++

The Nav Bar at the right provides a document-wide overview of the areas where two files differ – it allows a user to navigate to the various areas of interest with just a single click. This was an absolute life-saver for finding differences in large XML documents.

2. NetBeans

NetBeans was chosen as the development environment for the MPGrapher application. NetBeans is available for all platforms, is widely available, well-tested, and well-documented. For these reasons the discussion of NetBeans will be sparse, except to point out the reasons why it was chosen over its competitors (such as Eclipse). NetBeans version 7.0.1 was used to develop all the Java code for MPGrapher.

a. Simplicity

Eclipse and other Java Integrated Development Environments (IDEs) are more powerful than NetBeans, with millions of followers writing thousands of plugins every day. However, this is part of the attraction of NetBeans. Because its following is smaller, it is generally easier to install and far easier to use out-of-the-box – other IDEs are cannot even be used until a host of outside plugins are installed. NetBeans is also

intuitive – while there is much documentation available online, the author found that it was largely unneeded as the layout and features of the IDE seemed to "just make sense".

b. Code Completion

Nearly all IDEs offer code completion, but NetBeans implementation tends to be one of the best. Of all IDEs tested by the author, only Microsoft Visual Studio offers better code completion.

c. Aesthetics

NetBeans just looks better than its competitors. This is an important consideration since it tends to reduce stress. Composing software is a stressful task, comprising hours upon hours of staring at a computer screen in frustration. If an IDE is not aesthetically pleasing to the programmer, then it only helps to compound the stressfulness of the situation.

d. Navigation

(1) Navigator Window. NetBeans offers a Navigator window, which is admittedly similar to other IDEs. The Navigator window is shown in Figure 44, with constructors marked by diamonds, methods marked by circles, and fields marked by squares.

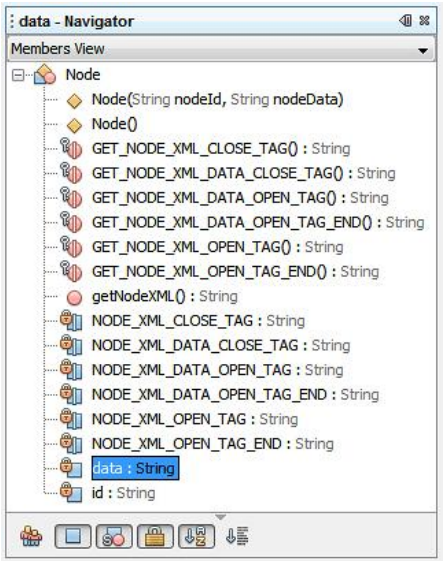

Figure 44. Navigator window in NetBeans

Private members are marked with a padlock, while protected members are marked with a key. In addition to the name of each member, the type and signature of each member is also listed. Double clicking on any of the entries in the Navigator window immediately jumps the cursor to the start of that member in the editor window. In addition to providing useful navigation, the combination of type, signature, name and visibility of all members of a class serve to make the Navigator window a quick, handy overview of a class as a whole, a benefit which sets it apart from similar navigation windows of its competitors.

(2) Errors, Warnings and Highlights. On the far right-side of Figure 45 one will notice a vertical gray bar with red, yellow and olive-colored horizontal stripes.

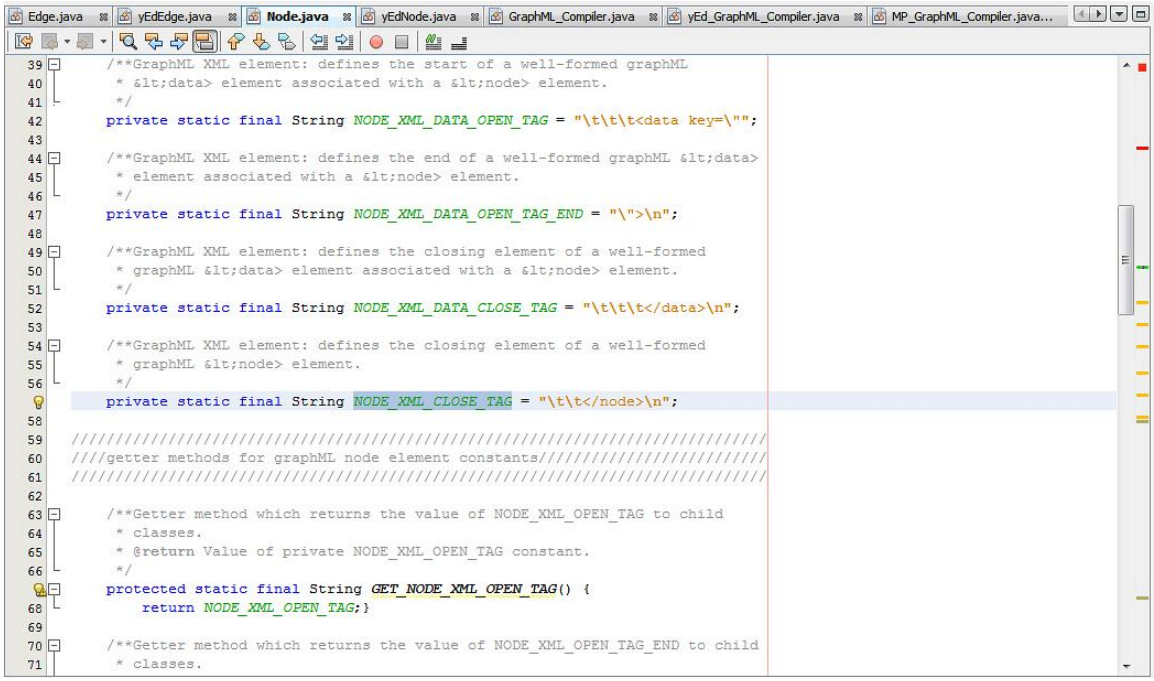

Figure 45. Errors, highlights and warnings in NetBeans

While this region of the editor does not have an actual name, it is an often-used navigation tool. Once again, it provides an overview of an entire document. Within the document, coding errors are marked by red horizontal bars, and warnings are marked by yellow horizontal bars. Additionally, by selecting words in the editor window (such as NODE_XML_CLOSE_TAG in Figure 45), all occurrences of that word are

marked by olive-colored horizontal stripes. Clicking on any of these horizontal bars immediately jumps the editor window to that section of the document. This nameless tool is a unique feature of NetBeans that is unlike any other IDE.

C. EXPERIMENTS WITH GRAPHML OUTPUT FROM YED

Writing a compiler is a rather straight-forward process, provided that one knows precisely what it is that he or she must compile – that was not known at the outset for the MPGrapher compiler. Figuring out what needed to be compiled was a pain-staking process that involved either opening GraphML files in Notepad++, changing one attribute of one element, saving the file and re-opening it in yEd to see what changed, or doing the same thing in reverse by opening a graph in yEd, changing one property, saving the file and re-opening it in Notepad++ to see what changed. Often graphs needed to be saved in yEd with different filenames in order to be compared side-by-side in Notepad++ using the Compare plugin. These experiments are described throughout the rest of this section.

1. Appearance of Generic GraphML Files in yEd

Since yEd uses GraphML as its default file format, the first thing to do was to open a simple GraphML file yEd – the results were bitterly disappointing. Figure 46 lists a simple, well-formed, GraphML schema-compliant XML file that represents a graph with three nodes and two edges.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
    http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
      <graph id="test" edgedefault="undirected">
            <node id="0"/>
            <node id="1"/>
            <node id="2"/>
            <edge ID="e0" source="0" target="1"/>
            <edge ID="e1" source="0" target="2"/>
      </graph>
</graphml>
```

Figure 46. Simple GraphML XML test-file

Figure 47 is a screenshot of the display pane after opening the same document in yEd. All three nodes are stacked one-on-top of the other in a single pile, which means that the edges between the nodes are subsequently hidden – quite a letdown. Selecting "Tree" from the layout (with default properties accepted) drop down menu results in the graph shown in Figure 48 – a remarkable improvement. Despite the improvement, the astute observer will immediately notice that while the GraphML file listed in Figure 46 defined an "undirected" graph, the resulting edges in Figure 48 are nevertheless directed. This phenomenon turned out to be a flaw in yEd: in yEd, each edge must contain its own `<arrows>` element which defines the directions of edges. One would think that if elements are not defined in an edge then the application would apply the value of the `edgedefault=""` attribute defined in the `<graph>` element definition, but that was not the case.

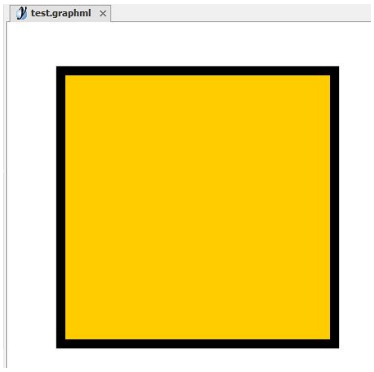

Figure 47. Default layout of a three-node, two edge GraphML document in yEd

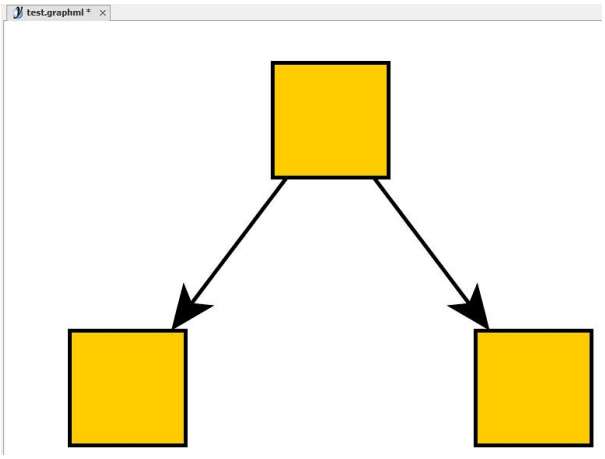

Figure 48. Three-node, two-edge graph after applying the "Tree" layout in yEd

2. Observing Changes in GraphML Files After Applying Layouts

After applying layouts in yEd and saving the results, the next step was to open the file in Notepad++ and note how the file changed from Figure 46. Saving the resulting GraphML file from Figure 48 resulted in the file listing of Appendix F – yEd adds `<data>` and `<key>` elements adding up to two pages of text for a simple three-node graph with only two edges (and no labels).

The first conclusion one must reach is that while yEd fully supports the GraphML schema, the GraphML schema is not sufficient for displaying meaningful graphs in yEd. Upon making this observation, the first question one might ask is, "which elements are necessary to display a graph, and which can be discarded?" While this was a logical question with which to experiment, in actuality it turned out to be more trouble than it was worth. Instead, a better question was, "Assuming that all elements and attributes are necessary, which are necessary to display meaningful graphs, and which can be safely ignored (i.e., provide default values in the MPGrapher application)?" This was accomplished moving nodes around, adding labels, changing various display properties, and comparing the changed files to the unchanged files using the Compare command in Notepad++. After numerous experiments, it became clear that only a handful of elements, and only a few attributes within those elements, needed to be altered for each graph in order to convey meaning in graphs opened by yEd.

A list of those elements and attributes, along with explanations of each, are described in section three. Besides altering these attributes and elements, the rest of the document listed in Appendix F is essentially repeated verbatim as default values for each GraphML file. One notable exception are the `fontFamily=""` and `fontSize=""` attributes of the `<y:edgeLabel>` and `<y:nodeLabel>` elements. These were changed to "Courier New" and "12", respectively, in order to provide fixed font-sizes that could be used to control the widths and heights of labels.

3. Elements and Attributes To Modify In Each GraphML Document

a. <node> Elements

(1) id="" Attribute. `<node>` elements contain only the `id=""` attribute, which must be unique for each node in a GraphML graph.

b. The <y:NodeLabel> Element

(1) PCDATA. The character strings that represent node labels in yEd-specific GraphML documents are actually PCDATA[7]. yEd supports any string of printable characters, to include tabs and carriage returns. Labels may appear anywhere between the opening and closing tags of `<y:NodeLabel>` elements.

c. The <y:Geometry> Element

(1) The x="" Attribute. The `x=""` attribute represents the x-coordinate of where the node should appear in the display pane. Screen coordinates in yEd are measured from the top-most, left-most node of the graph. The left-most node of a yEd graph is given a 0.0 x-coordinate – all nodes to the left of this node receive an x-coordinate less than zero, and all nodes to the right are given x-coordinates greater than zero.

(2) The y="" Attribute. The `y=""` attribute represents the y-coordinate of where the node should appear in the display pane. The top-most node of a yEd graph is given a 0.0 y-coordinate – all nodes above this node receive a y-coordinate less than zero, and all nodes below are given y-coordinates greater than zero.

(3) The height="" Attribute. The `height=""` attribute is a float value that represents the height of a node.

(4) The width="" Attribute. The `width=""` attribute is a float value that represents the width of a node.

[7] PCDATA is an XML term that essentially means "plaintext". PCDATA contrasts CDATA, which stands for Character Data, and is used to define non-parsed character data such as < (the < symbol).

d. The <y:Shape> Element

(1) The type="" Attribute. The `type=""` attribute defines the shape of a node in yEd. Possbile values include rectangle, roundrectangle, and hexagon, just to name a few.

e. The <y:Fill> Element

(1) The color="" Attribute. The `color=""` attribute represents the background color of a node. The value must be a six-digit hexadecimal value, preceded by the # symbol. A-F values may be either upper or lower case characters.

f. The <edge> Element

(1) The id="" Attribute. The `id=""` attribute represents the ID of the `<edge>` element and must be unique for each edge of a GraphML graph.

(2) The source="" Attribute. The `source=""` attribute represents the starting node from which an edge begins.

(3) The target="" Attribute. The `target=""` attribute represents the ending node at which an edge ends. `source=""` and `target=""` attributes must be defined for each edge, regardless of whether the edge is directed or not.

g. The <arrows> Element

Rather than conforming to the `edgedefault=""` attribute defined in the `<graph>` definition element, yEd uses `<arrows>` elements to define direction in edges.

(1) The source="" Attribute. The `source=""` attribute defines the arrow-type that yEd displays at the source-end of an edge. This value is set to `none` for all edges defined in the MPGrapher application.

(2) The target="" Attribute. The `target=""` attribute defines the arrow-type that yEd displays at the target-end of an edge. The MPGrapher application sets this value to `standard` for directed edges and `none` for non-directed edges.

h. The <y:LineStyle> Element

(1) The color="" Attribute. The `color=""` attribute represents the background color of a node. The value must be a six-digit hexadecimal value, preceded by the # symbol. A-F values may be either upper or lower case characters.

(2) The type="" Attribute. The `type=""` attribute indicates the line-style of an edge. The MPGrapher application sets the value of this attribute to either `line` or `dashed`.

(3) The width="" Attribute. The `width=""` attribute controls the weight of an edge. Values are limited to float values, and there is no known limit to how large or small the values must be.

i. The <y:Point> Element

`<y:Point>` elements are used to control the path than an edge may follow. For node-pairs which have two directed edges, one in each direction, MPGrapher defines two such points in order to prevent the resulting edges from being displayed one-on-top of the other, which would represent one bi-directional edge instead of two separate directed edges. Screen coordinates for points use the same coordinate system as described for `<node>` elements in section a.

(1) The x="" Attribute. The `x=""` attribute represents the x-coordinate of where the node should appear in the display pane.

(2) The y="" Attribute. The `y=""` attribute represents the y-coordinate of where the node should appear in the display pane.

j. The <y:EdgeLabel> Element

(1) PCDATA. The text of edge labels are defined by PCDATA is exactly the same way as Node labels (see section b for details).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DISCUSSION OF THE MPGRAPHER APPLICATION

A. CLASSES BUILT FROM SEPARATE GRAPHML GRAMMARS

The MPGrapher application was designed and built in three phases in order to appeal to the broadest audience possible. In phase one, three classes (`GraphML_Compiler`, `Node` and `Edge`) were built which were designed to parse input files which conform to the grammar for generic GraphML files (described in section B) and output GraphML XML files which can be displayed in any application that supports the GraphML standard. The `GraphML_Compiler`, `Node` and `Edge` classes represent a complete GraphML parser / compiler which can be extracted out and used as a stand-alone application for generating GraphML XML files, which can in turn be opened by any application that supports the GraphML file format.

In phase two, three additional classes (`yEd_GraphML_Compiler`, `yEdNode` and `yEdEdge`) were built. These three classes extend the `GraphML_Compiler`, `Node` and `Edge` classes, respectively, in order to provide a complete parser / compiler which can be used for generating GraphML XML files which are specific to the yEd graphing application. Similar to their parent classes, these classes were designed so that they also can be extracted out and used as a stand-alone package for generating yEd-specific GraphML XML files (which is what the MPGrapher application does). The XML files generated by the `yEd_GraphML_Compiler`, `yEdNode` and `yEdEdge` classes provide the minimum XML elements and attributes necessary for displaying graphs in yEd (with nodes arranged into a staggered matrix rather than all in one stack); this design decision was intended to allow future developers to easily alter the `yEd_GraphML_Compiler`, `yEdNode` and `yEdEdge` classes to fit their own needs. Additionally, the `yEd_GraphML_Compiler` class has built-in functionality to recognize input file which represent generic GraphML input files that are not yEd-specific. These files are automatically passed to parent `GraphML_Compiler` objects for parsing and compiling.

In phase three, a third compiler class (`MP_GraphML_Compiler`) was built which extends the `yEd_GraphML_Compiler` class. This class is responsible for defining the `Node` and `Edge` output parameters which are used to display the various types of nodes and edges in yEd. Lastly, an `MPGrapher` wrapper-class was built which opens input files, creates output files, and instantiates `MP_GraphML_Compiler` objects.

B. GRAMMAR AND PARSING PROCESS FOR INPUT FILES

The MPGrapher application is capable of parsing two different types of input files: generic GraphML files and yEd-specific GraphML files. This section describes the details of each grammar.

1. Grammar and Parsing Process for Generic GraphML Input Files

The complete grammar specification which input files must follow for compiling generic GraphML XML documents is presented in Appendix G. The only terminal symbols of the grammar are: `%%nodes%%`, `%%edges%%`, and `%%eog%%`. All text which appears before the `%%nodes%%` token, or after the `%%eog%%` token are ignored and may be used as areas for documenting the input file. The GraphML schema definition does not specifically allow for labeling nodes or edges, therefore the generic GraphML grammar does not define any method to label nodes or edges.

All entries between the `%%nodes%%` token and the `%%edges%%` token define the "nodes" section of an input file. Tokens in the "nodes" section must be separated by whitespace, although the type of whitespace does not matter: spaces, tabs and carriage returns/newline characters all work equally well for separating nodes (mixtures of different whitespace character types may be used to separate nodes into logical orderings). Tokens in the "nodes" section must also be unique – repeats of the same token are not allowed.

All entries between the `%%edges%%` token and the `%%eog%%` define the "edges" section of an input file. Each entry of this line composes a token, followed by whitespace, followed by an additional token, followed by the `;` character (with or without intervening

whitespace). In contrast to the “nodes” section, unique-identifiers are strictly not allowed. However, each token of the “edges” section must precisely match a token of the “nodes” section (with the possible addition of the `;` character).

a. Example Input File

```
%%nodes%%
node1 node2
%%edges%%
node1 node2;
%%eog%%
```

Figure 49. Sample generic GraphML input file

b. Example Input File With Comments

```
This is a graph with
2 nodes & 1 edge:

%%nodes%%
nodeA nodeB
%%edges%%
nodeA nodeB ;
%%eog%%

Notice the whitespace
between nodeB and the
“;” character
```

Figure 50. Sample generic GraphML input file with comments

c. Logical Progression Through the GraphML_Compiler Class

The parser parses tokens until it reaches the `%%nodes%%` token. Upon reaching the nodes token, the compiler continues to parse the input file, token by token, instantiates a Node object using each token as a unique identifier, and adds each Node object to a collection of Nodes. A slightly different process occurs while parsing the “edges” section of an input file. Upon reaching the `%%edges%%` token, the parser continues to parse the input document, token by token. The first token encountered is

stored as the beginning of an identifier which will be used to uniquely identify each edge. Upon encountering the next token, if that token ends with the `;` character, the `;` character is stripped off and the remaining token is added to the identifier which is used to uniquely identify that particular edge (this process forms a `source::target` pair that uniquely identifies each edge); the parser then instantiates an `Edge` object, adds the object to a collection of `Edges`, and continues parsing the rest of the document. If the next token encountered does not end with the `;` character, then the compiler immediately add the token to the identifier of the edge, instantiates an `Edge` object, adds the object to the collection of `Edges`, and continues parsing the input file – in this case, the next token the parser encounters must a stand-alone `;` character, otherwise an error will result.

Upon reaching the `%%eog%%` token, the parser stops parsing and ignores any tokens which may follow. Any lines of text that appear after the `%%eog%%` token may be used as comments.

2. Grammar For yEd-specific GraphML Input Files

The elements and attributes which are necessary to display a graph in the yEd graphing application are considerably more robust than the GraphML schema (although yEd is capable of displaying a generic GraphML file – the nodes are simply displayed one-on-top of the other in a single stack). Therefore, input files to the MPGrapher application, which are meant to be properly displayed in yEd, require a substantially more descriptive grammar. The grammar for compiling yEd-specific GraphML XML documents is presented in Appendix H. It should be immediately apparent that the yEd-specific GraphML grammar is built upon the generic GraphML grammar. This follows the pattern that yWorks followed when they built the schema of yEd around the generic GraphML schema.

While the yEd-GraphML grammar utilizes the same terminal tokens as its parent GraphML grammar (`%%nodes%%`, `%%edges%%`, and `%%eog%%`), it adds one more which is mandatory for any input file which is meant to be displayed in yEd: a `%yEd%` token must appear somewhere before the `%%nodes%%` token.

a. yEd_GraphML_Compiler, "nodes" Section

Classes which call the yEd_GraphML_Compiler class may provide additional tokens which are to be used as sub-sections of the "nodes" section of the input document. This is the only means available for classifying nodes according to their type (and therefore according to how they are to be displayed in yEd). Terminal tokens which are to be used for defining sub-sections of the "nodes" section must represent a single string of printable characters, minus whitespaces and the `%` character, and be contained inside `%` characters (single `%` symbols are meant to differentiate yEd-GraphML grammar terminal tokens from the GraphML terminal tokens which contain double-`%` symbols). For instance, in the MPGrapher application, the MP_GraphML_Compiler uses five such tokens: `%trace_roots%`, `%trace_events%`, `%architecture_environment%`, `%architecture_processes%`, and `%architecture_data%`. Additionally, since nodes are displayed with node labels in yEd, and since identical labels may be repeated (while identical node names are not), input files which follow the yEd-grammar must contain unique identifiers followed by labels in the "nodes" section (to include user-defined sub-sections). Node labels may contain any string of zero or more printable characters, plus whitespace characters, minus quotes, included in quotes. A unique identifier marks the beginning of a node, while a quotes character which follows another quotes character marks the end of a node in the yEd-GraphML grammar.

b. yEd_GraphML_Compiler, "edges" Section

The "edges" section of the yEd-GraphML grammar section is also considerably more complicated than its generic GraphML grammar counterpart.

Upon encountering an `%%edges%%` token in a input document, the `yEd_GraphML_Compiler` begins parsing the "edges" section of the input file and simultaneously instantiates `yEdEdge` objects and adds them to a `yEdEdge` objects collection. As with generic GraphML input files, edge definitions in yEd-specific input files must also provide two unique identifiers, each of which must be an exact match to one of the unique identifiers provided in the "nodes" section, the first of which represents

the source node of the edge, and the second representing to target node of the edge. If there are no labels and no specific display propertied for the edge, then the edge is closed out by a semicolon (the semicolon may either be attached to the target unique identifier or appear by itself at the end of the line).

However, yEd-specific graphs are more than likely to have display properties defined. Following the target unique identifier, if the parser encounters a third token before encountering a semicolon, then this token must represent a user-defined keyword which defines a specific class of edge parameters that the yEd graphing application is to use to display that particular edge. Edge parameters must be defined by the user and passed to the yEd_GraphML_Compiler class as part of the calling signature. The MPGrapher application provides four such definitions, each of which is identified by one of four keywords: `ADD`, `SHARE_ALL`, `INCLUDES`, and `PRECEDES` – each defines several display parameters as listed in section C.2.a. Each edge may provide one and only one user-defined keyword. After processing a user-defined keyword, the parser may either encounter a semicolon or a quotation mark – a semicolon marks the end of the edge definition (the semicolon may either be attached to the keyword or stand-alone), while a quotation mark indicates that there are labels associated with the edge.

If the parser encounters a quotation mark, then the yEd_GraphML_Compiler immediately instantiates a Java String array and begins parsing and compiling edge labels. All tokens within quotation marks, along with intervening whitespace characters, are added to the String array object until the parser encounters a subsequent quotation mark. Each edge may contain an unlimited number of labels. However, the last quotation mark of the last label of the edge must be followed by a semicolon to mark the end of the edge. The `yEd_GraphML_Compiler` attaches label array objects to the `yEdEdge` objects of the `labels` field before it adds the `yEdEdge` object to the `yEdEdge` objects collection.

c. Examples

Some examples of valid input files that conform to the yEd-specific input grammar, at least as pertains to the MPGrapher application, follow.

```
%yEd%
%%nodes%%
%trace_roots%
node1 "node 1" node2 "node 2"
%%edges%%
node1 node2 ADD "this is a label";
%%eog%%
```

Figure 51. Sample yEd-specific GraphML input file

```
%yEd%
%%nodes%%
%architecture_environment%
user "some guy"
%architecture_processes%
file1 "file operation"
file2 "file operation"
%%edges%%
user file1 SHARE_ALL "open file"
"write to file" "delete file" "close file";
User file2 SHARE_ALL "open file"
"write to file" "delete file" "close file" ;
%%eog%%
```

Figure 52. Sample yEd-specific GraphML input file with multiple labels

```
%yEd%
%%nodes%%
%trace_roots%
root "root"
%trace_events%
event1 "event"
event2 "event"
%%edges%%
root event1 INCLUDES  ;
root event2 INCLUDES  ;
event1 event2 PRECEDES;
%%eog%%
```

Figure 53. Sample yEd-specific GraphML input file without edge labels

C. REQUIREMENTS AND SPECIFICATION

1. High-level Goals

- The MPGrapher application will be able to run on any platform.
- The MPGrapher application will parse text files that conform to the GraphML / yEd-GraphML grammars.
- The MPGrapher application will compile and output GraphML XML documents which conform to the yEd XML schema (i.e., the resulting files can be displayed in yEd).
- The MPGrapher application will display graphs of both architectures and traces.
- The MPGrapher application will compile GraphML representations of graphs consisting of hundreds of nodes and thousands of edges in a reasonable amount of time.

2. Measures of Success

There is essentially only one measure of success for the program: successfully open a GraphML XML file in yEd. Of course opening the file in yEd also implies successfully parsing an input file and compiling a GraphML XML output file. Additionally, to "successfully" open the file in yEd implies the following criteria:

- All nodes of the input document are present.
- Nodes are labeled according to the input file
- Nodes are of the same size and large enough to contain the label of the node with the longest label
- All nodes are properly colored and shaped according to the corresponding node-type of the input file
- Nodes are laid out so that each is clearly visible
- All edges from the input file are visible in yEd
- All edges are attached to the proper nodes according to the input file
- Edges between identical nodes do not overlap
- Edges which have labels in the input file have corresponding labels in the output document
- Edges are displayed with the proper weight, color and line-style (dashed for example) according to the edge-type indicated by the input file

a. List of Node Types and Display Properties

Architecture environment-type node parameters:

Node color: #FCD5B5

Node shape: rectangle with rounded corners

Architecture processes-type node parameters:

Node color: #D7E4BD

Node shape: rectangle with rounded corners

Architecture data-type node parameters

Node color: #CCC1DA

Node shape: rectangle with rounded corners

Trace roots-type node parameters

Node color: #D7E4BD

Node shape: hexagon

Trace events-type node parameters

Node color: #FFFF99

Node shape: rectangle with rounded corners

b. List of Edge Types and Display Properties

Trace INCLUDES-type edge parameters

Edge color: #000000

Edge weight: 1.0

Edge direction: directed

Edge line-type: solid line

Trace PRECEDES-type edge parameters

Edge color: #00FF00

Edge weight: 4.0

Edge direction: directed

Edge line-type: dashed

Architecture ADD-type edge parameters

Edge color: #00FF00

Edge weight: 4.0

Edge direction: directed

Edge line-type: solid line

Architecture SHARE_ALL edge parameters

Edge color: #FF99CC

Edge weight: 4.0

Edge direction: non-directed

Edge line-type: solid line

D. ARCHITECTURE AND DESIGN

The MPGrapher package is composed of eight classes which comprise more than 3,600 lines of code (counting code-comments and line-breaks which were added for legibility). MPGrapher is an example of a single-pass, LL1 recursive descent parser. The signatures of the eight classes are presented in Appendix I. The class diagrams of Appendix I follow the traditional UML pattern of beginning public method names with a + symbol, private method names with a - symbol, and protected method names with a # symbol. All class fields within the package are private. Object fields which need to be referenced or set by other objects are done so via protected "getter" and "setter" methods.

The java code of the MPGrapher package is presented in Appendix A. The code in the appendices is thoroughly commented. Although the rest of this section presents an overview of each class, as well as the package as a whole, the reader is referred to the comments in Appendix A for intimate details of the Java code.

An MP-architecture graph of the MPGrapher application is given in Figure 54 as an example.

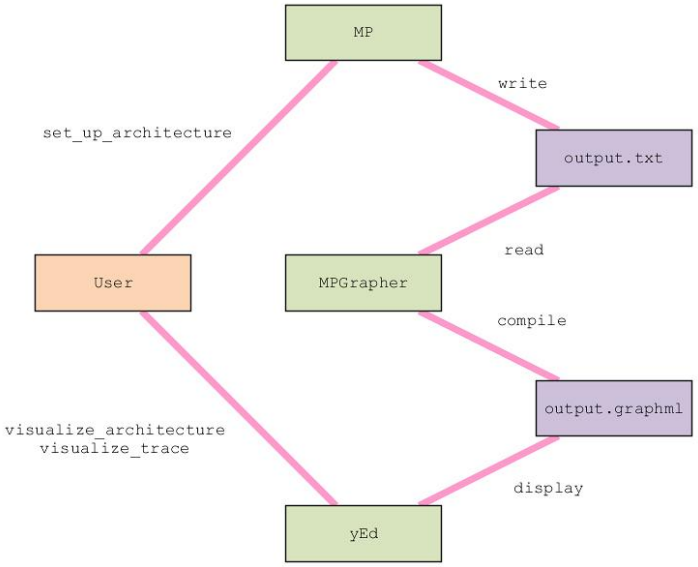

Figure 54. Architecture of the MPGrapher application

1. Class Hierarchy

Figure 55 shows the most abstract view of the architecture of MPGrapher. Boxes in Figure 55 with thick borders indicate the eight classes of the MPGrapher application, while boxes with thin borders represent other constructs such as documents, files, grammar specifications or even ordinary arrays. The yEd graphing application is represented using the traditional UML component symbol.

As indicated in Figure 55, the `GraphML_Compiler` class is the parent of the `yEd_GraphML_Compiler` class, which is in turn the parent of the `MP_GraphML_Compiler` class. Similarly, `Node` is the parent of `yEdNode` and `Edge` is the parent of `yEdEdge`. All child classes within the application inherit all public and protected methods of their parent classes with no overrides.

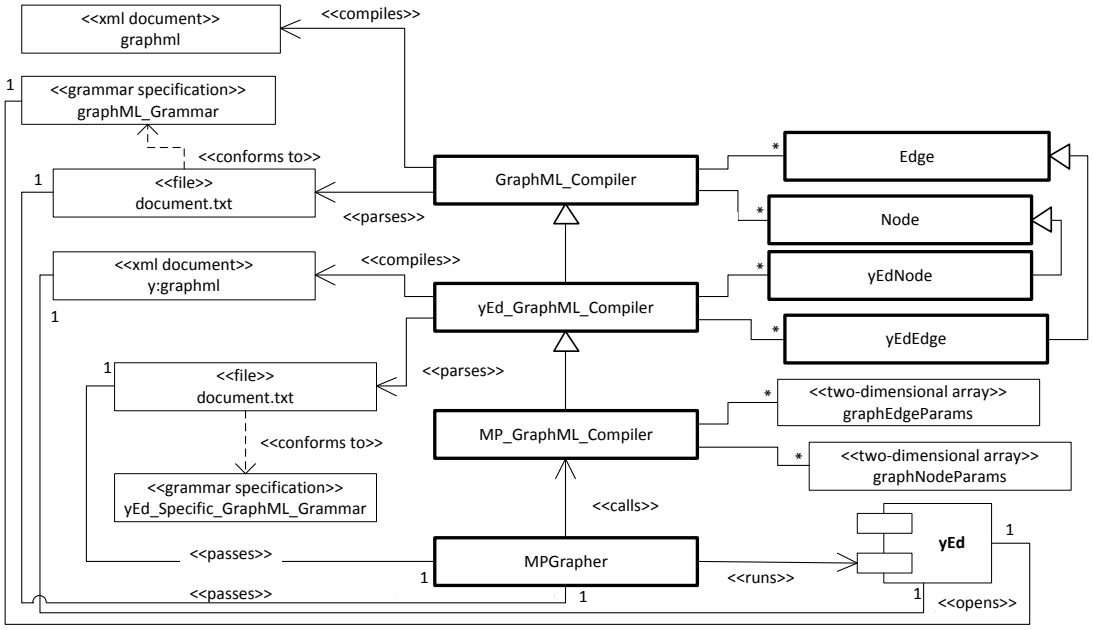

Figure 55. Class hierarchy

The `MPGrapher` class contains the "main" method of the MPGrapher package. The `MPGrapher` class takes one argument, which is the name (and directory) of the file that is to be parsed. The MPGrapher application will graph one and only one file at a time. The type of input file may conform to either the general GraphML grammar or the yEd-specific GraphML grammar. The `MPGrapher` class fetches a GraphML XML document from an `MP_GraphML_Compiler` object, which it then passes on to the yEd graphing application. The `MPGrapher` class will pass one and only one GraphML XML document per run to yEd.

The `MP_GraphML_Compiler` class receives the name (and directory) of the file to be parsed from the `MPGrapher` class and returns a well-formed GraphML XML document. In order to compile GraphML elements and attributes, the `MP_GraphML_Compiler` class first builds a number of String arrays which will be used to control the appearance of edges, nodes and labels in the yEd graphing application. Once built, the arrays are passed to the `yEd_GraphML_Compiler` class along with the name (and directory) of the file to be parsed.

The yEd_GraphML_Compiler class is the workhorse of the MPGrapher package. The yEd_GraphML_Compiler begins the parsing process. If the parser discovers that the input file represents a generic GraphML file, it calls the GraphML_Compiler to complete the parsing and XML compilation process. Otherwise, the yEd_GraphML_Compiler parses the input file, token by token, and composes one or more yEdNode objects and yEdEdge objects. Once the end of the input document is reached, the yEd_GraphML_Compiler class iterates through the yEdNode and yEdEdge object collections and compiles well-formed, yEd-specific GraphML XML elements and attributes.

A similar process is followed by the GraphML_Compiler class as the yEd_GraphML_Compiler class, except that instead of creating and iterating through yEdNode objects and yEdEdge objects, it creates and iterates through generic Node and Edge objects.

2. Logic of the MPGrapher / MP_GraphML_Compiler Classes

While the logic of the GraphML_Compiler, Node, Edge, yEd_GraphML_Compiler, yEdNode and yEdEdge classes have been described in some detail in sections B and D.1, the MPGrapher and MP_GraphML_Compiler classes have largely gone untouched. The application begins when the user (or another application, such as the Monterey Phoenix program) invokes the MPGrapher package. Invoking the MPGrapher package with an input file is done in the usual way. For example:

```
java -jar MPGrapher.jar ..\mpData.txt
```

The user supplies the name (and directory) of a GraphML/yEd-GraphML grammar-compliant text file (and directory) to parse (such as ..\mpData.txt in the example). The name (and directory) of the input file is passed to MPGrapher as args[0]. MPGrapher creates a Java String object (xmlOut) which will ultimately contain the entire GraphML XML document which is to be graphed in yEd. MPGrapher

then instantiates an `MP_GraphML_Compiler` (`mgc`) object and sets `xmlOut` to the return value of the String which is returned by the `getMPGraph` method of `mgc`.

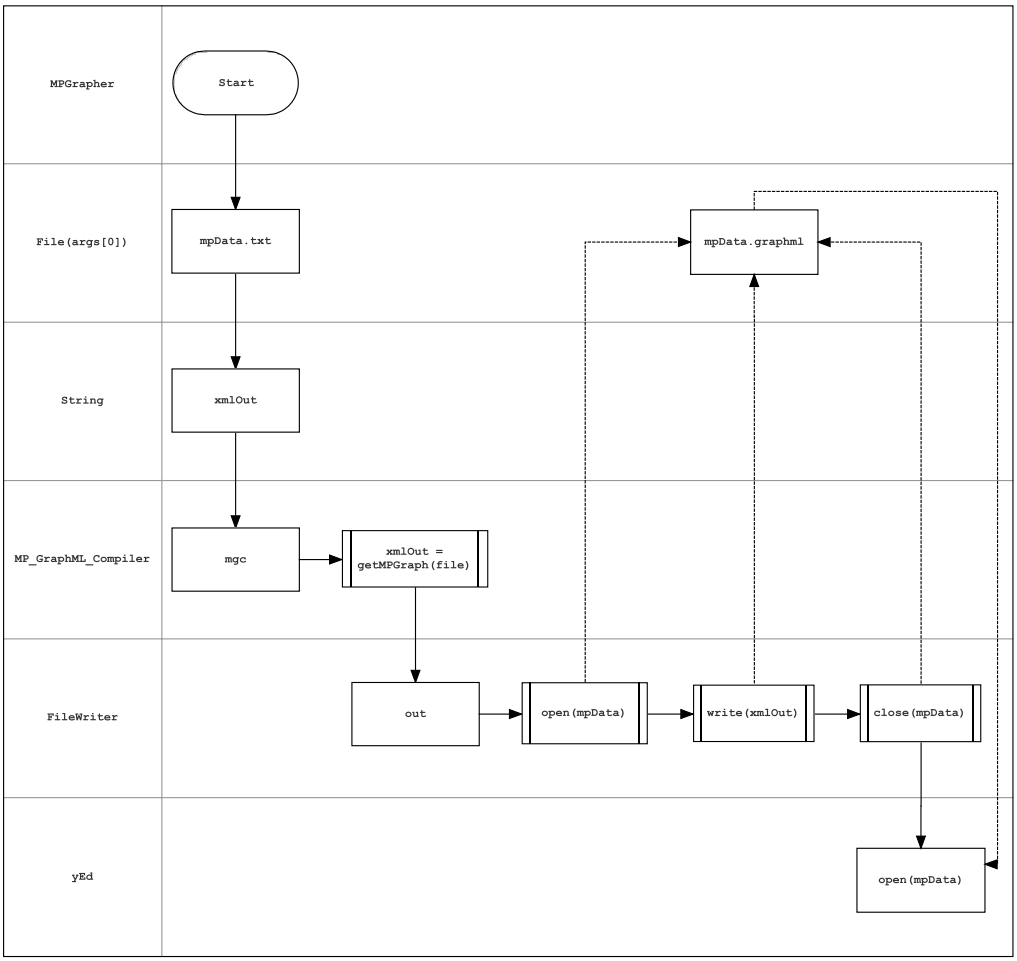

Figure 56. `MPGrapher` class flow chart

Once the GraphML document has been created, `MPGrapher` creates a `Filewriter` object, opens a new file (or overwrites any previous files with the same filename and directory), writes the XML document to the file, and closes the file (the file has the same name as the name given for the input file, with the extension .graphml vice .txt). Lastly, `MPGrapher` opens the newly created GraphML file in yEd.

V. RESULTS AND FUTURE WORK

A. TEST FILES, TESTS, AND TEST RESULTS

Software testing is essentially the process of hunting for bugs. The term "bugs" may not refer to software crashes alone, but may be extended to any "incorrect" area of interest. Unfortunately, due to the undependable nature of software, testing can never guarantee the absence of bugs – it can only test for their presence. The process of ensuring the "correctness" of MPGrapher is described in this section.

Testing of the MPGrapher application was not limited to MPGrapher alone. The yEd graphing application was also subjected to a small number of tests to ensure that it would be able to handle large graphs with hundreds of nodes and thousands of edges without crashing.

1. Test of Speed of Execution of MPGrapher

The last requirement stated in chapter IV, section C.1 states, "The MPGrapher application will compile GraphML representations of graphs consisting of hundreds of nodes and thousands of edges in a reasonable amount of time." In order to test for this requirement, a generic GraphML input document consisting of 500 nodes and 2,491 edges was prepared. MPGrapher was able to parse this file and compile the resulting GraphML document in about 2.2 seconds. While the phrase "reasonable amount of time" might be a bit nebulous, all involved with the MPGrapher program agreed that 2.2 seconds was indeed a reasonable amount of time.

2. Test of Manipulating Large Graphs In yEd

The primary question about the yEd graphing application was, "Can yEd handle large graphs of hundreds of nodes and thousands of edges without crashing?" The GraphML file generated by the previous test (saved as "Big_Graph_test.graphml") was used to answer this question. Opening Big_Graph_test.graphml in yEd and applying the Circular layout (with default properties accepted), yEd was able to successfully apply the layout. The result is shown in Figure 57.

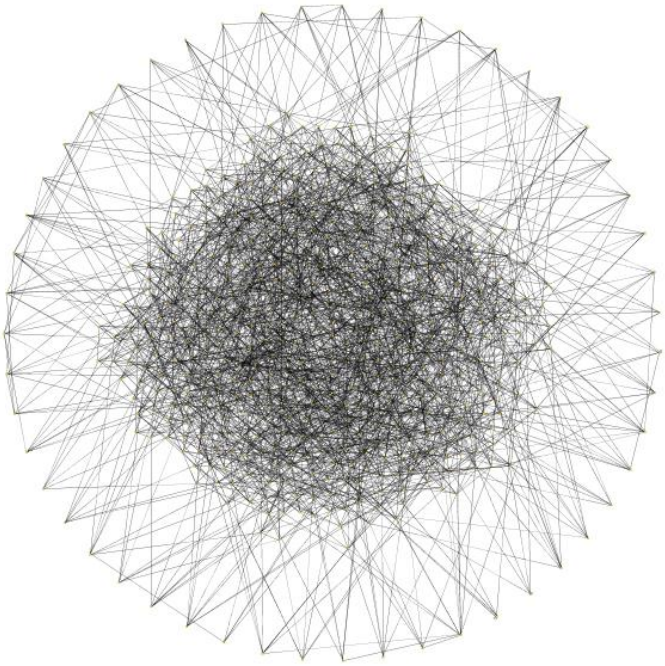

Figure 57. Test result of a large graph in yEd

While this was not a test of speed of execution, a time of 17.6 seconds to perform the test was nevertheless recorded.

3. Layout Test

As stated earlier, graphs opened in yEd without predefined geometry are opened in a single stack. In order to create meaning, MPGrapher is designed to lay out nodes in a matrix, with staggered rows and columns.

```
%yEd% %%nodes%% %architecture_data%
1-1 "1-1" 2-1 "2-1" 2-2 "2-2" 1-2 "1-2"
3-1 "3-1" 3-2 "3-2" 3-3 "3-3" 2-3 "2-3"
1-3 "1-3" 4-1 "4-1" 4-2 "4-2" 4-3 "4-3"
4-4 "4-4" 3-4 "3-4" 2-4 "2-4" 1-4 "1-4" %%edges%%
1-1 1-2 ADD; 1-2 1-3 ADD; 1-3 1-4 ADD;
2-1 2-2 ADD; 2-2 2-3 ADD; 2-3 2-4 ADD;
3-1 3-2 ADD; 3-2 3-3 ADD; 3-3 3-4 ADD;
4-1 4-2 ADD; 4-2 4-3 ADD; 4-3 4-4 ADD;
1-1 2-1 PRECEDES; 2-1 3-1 PRECEDES;
3-1 4-1 PRECEDES; 1-2 2-2 PRECEDES;
2-2 3-2 PRECEDES; 3-2 4-2 PRECEDES;
1-3 2-3 PRECEDES; 2-3 3-3 PRECEDES;
3-3 4-3 PRECEDES; 1-4 2-4 PRECEDES;
2-4 3-4 PRECEDES; 3-4 4-4 PRECEDES; %%eog%%
```

Figure 58. Layout tester input file

Figure 58, an input file of 16 nodes, was created in order to test this layout pattern. The resulting graph, as opened in yEd without any subsequent manipulation, is shown in Figure 59. The resulting GraphML document is presented in Appendix J.

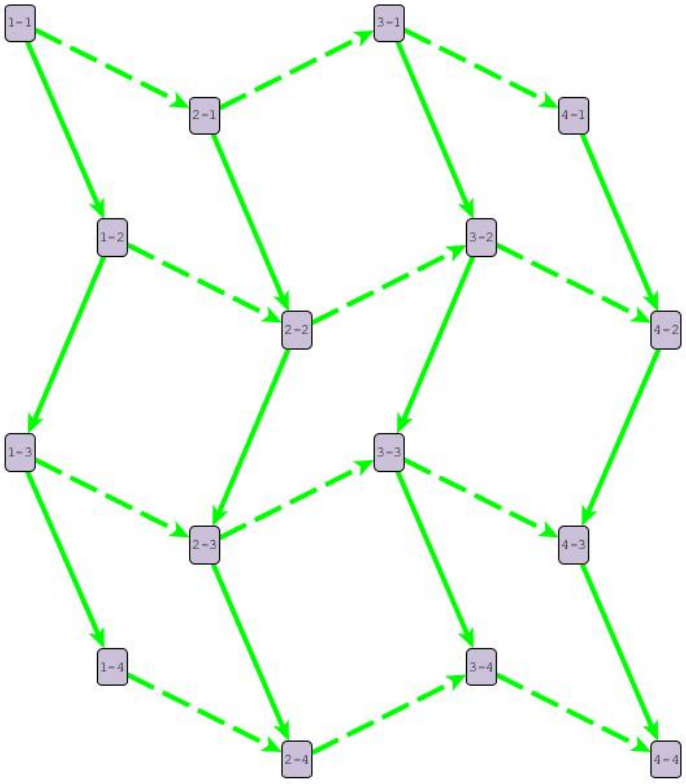

Figure 59. Layout test of MPGrapher

4. Edge Point Test

When a pair of nodes share two edges between them, as in Figure 4, it is necessary to route the two edges along different paths. This is especially important when both edges are directed edges, otherwise, when the two edges are laid on top of each other, the two directed edges will appear as one single, bi-directional edge. MPGrapher is designed both to test for this phenomenon and, when discovered, to automatically insert points into the resulting GraphML file to route the two edges away from each other. Figure 60 lists an input file that was designed to test this feature. It consists of six edges, two that belong to nodes with identical x-coordinates, two that belong to nodes with identical y-coordinates, and two that belong to nodes that diagonal from each other.

```
%yEd% %%nodes%% %architecture_data%
node1 "node 1" node2 "node 2" node3 "node 3"
node4 "node 4" node5 "node 5" node6 "node 6"
node7 "node 7" node8 "node 8" node9 "node 9"
%%edges%% node1 node5 ADD; node5 node1 ADD;
node1 node9 ADD; node9 node1 ADD;
node2 node7 ADD; node7 node2 ADD; %%eog%%
```

Figure 60. Edge points tester input file

Figure 61 shows the resulting graph. The resulting GraphML file is listed in Appendix J.

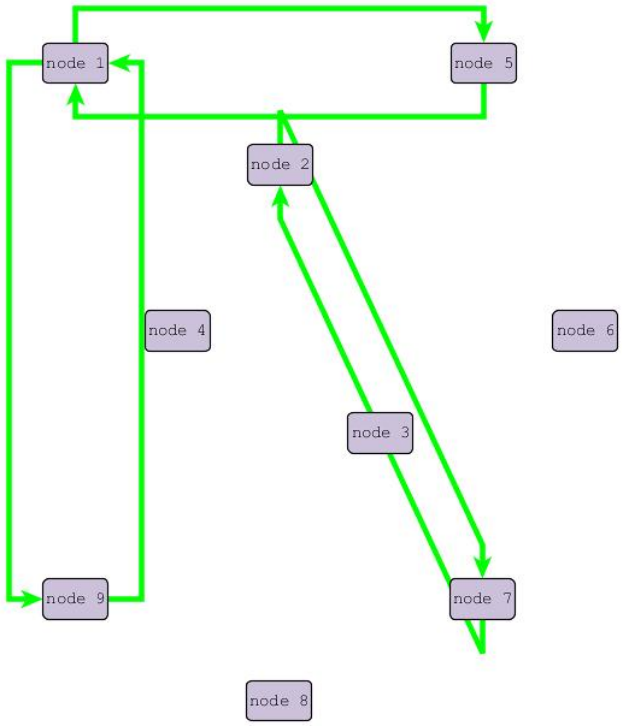

Figure 61. Edge points test of MPGrapher

5. Sample Architecture Graph

Of course, all is for naught if the program is unable to effectively graph architectures and traces. Figure 62 presents an input file designed to produce an architecture graph that mimics that of Figure 4. The resulting graph, as opened in yEd without any manipulation of the layout, is shown in Figure 63. Applying the Orthogonal-classic layout with default properties resulted in Figure 64 – while it is not an exact replica of Figure 4, the meaning of the graph is clear and logical.

```
%yEd% %%nodes%% %architecture_environment% Customer "Customer"
%architecture_processes% ATM_system "ATM System"
%architecture_data% Data_base "Data Base" %%edges%%
ATM_system Data_base SHARE_ALL "validate_id" "check_balance";
ATM_system Customer ADD "dispense_money>>get_money"
"id_successful>>identification_succeeds" "id_failed>>identification_fails"
"insuficient_balance>>not_sufficient_funds";
Customer ATM_system ADD "smart_card>>read_card"
"request_withdrawal>>check_balance"; %%eog%%
```

Figure 62. Input file to test a representative architecture sample

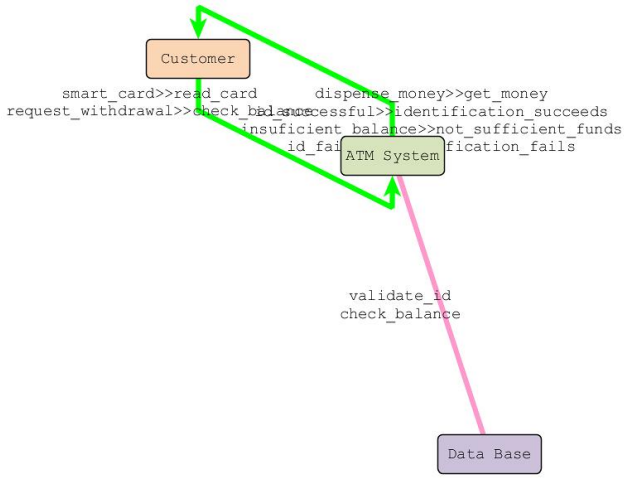

Figure 63. Default graph of sample architecture

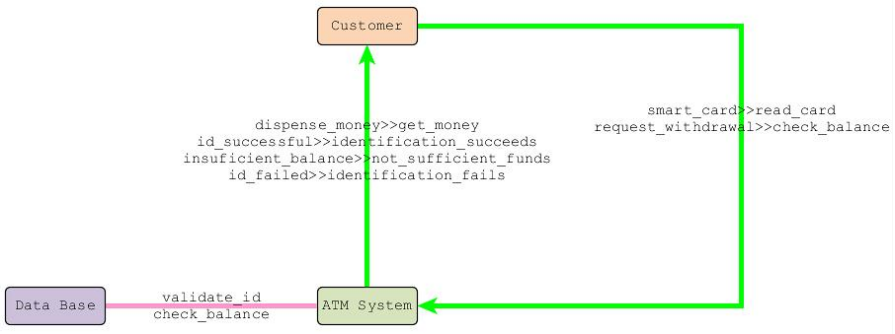

Figure 64. Graph of sample architecture with Orthogonal-Classic layout applied

The GraphML file that was produced for this test is listed in Appendix J.

6. Sample Trace Graph

The only test that remains is to create a representative sample of a graph of a trace. Figure 65 lists an input file that is meant to mimic a trace similar to Figure 3. The

resulting graph, as opened in yEd without any manipulation of the layout, is shown in Figure 66.

```
%yEd% %%nodes%% %trace_roots% Task_A "Task A" Task_B "Task B"
%trace_events% send1 "Send" send2 "Send" send3 "Send"
receive1 "Receive" receive2 "Receive" receive3 "Receive" %%edges%%
Task_A send1 INCLUDES; Task_A send2 INCLUDES; Task_A send3 INCLUDES;
send1 send2 PRECEDES; send2 send3 PRECEDES;
send1 receive1 PRECEDES; send2 receive2 PRECEDES; send3 receive3 PRECEDES;
receive1 receive2 PRECEDES; receive2 receive3 PRECEDES;
Task_B receive1 INCLUDES; Task_B receive2 INCLUDES;
Task_B receive3 INCLUDES; %%eog%%
```

Figure 65. Input file to test a representative trace sample

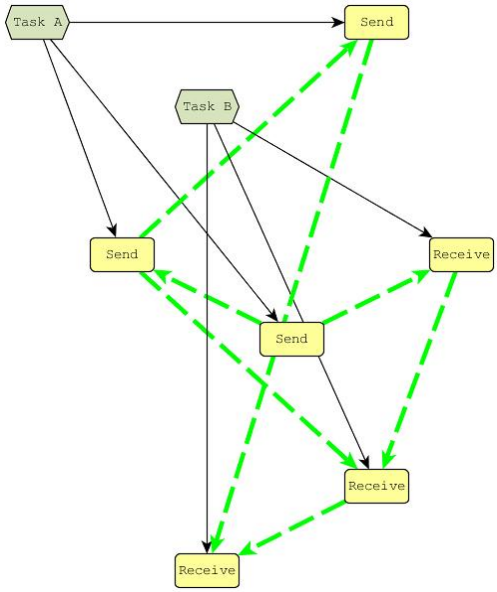

Figure 66. Default graph of sample trace

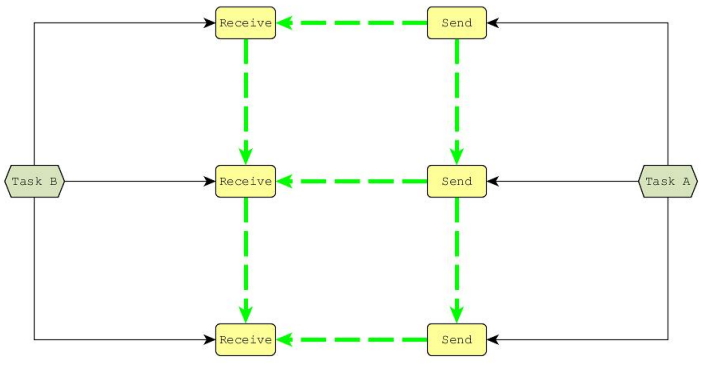

Figure 67. Graph of sample architecture with Orthogonal-Classic layout applied

Figure 67 shows the results of applying the Orthogonal-classic layout with default properties. Once again, while the result is not an exact replica of Figure 3, the meaning is still clear and the trace lines are easy to follow. The GraphML file that was produced for this test is listed in Appendix J.

B. FUTURE WORK

The MPGrapher application is a software project, and, like all projects, it was subject to a schedule and time constraints. Naturally, without unlimited timeframes with which to work, there were a number of refinements that could not be completed. The rest of this chapter suggests some useful additions that could be made to the program, along other, more theoretical improvements.

1. How Best to Represent Graphs

Figure 5 introduced a trace with events laid out in neat rows. An input that attempted to reproduce the graph of Figure 5 was constructed and fed into MPGrapher. The result, with the hierarchical layout applied, is shown in Figure 68.

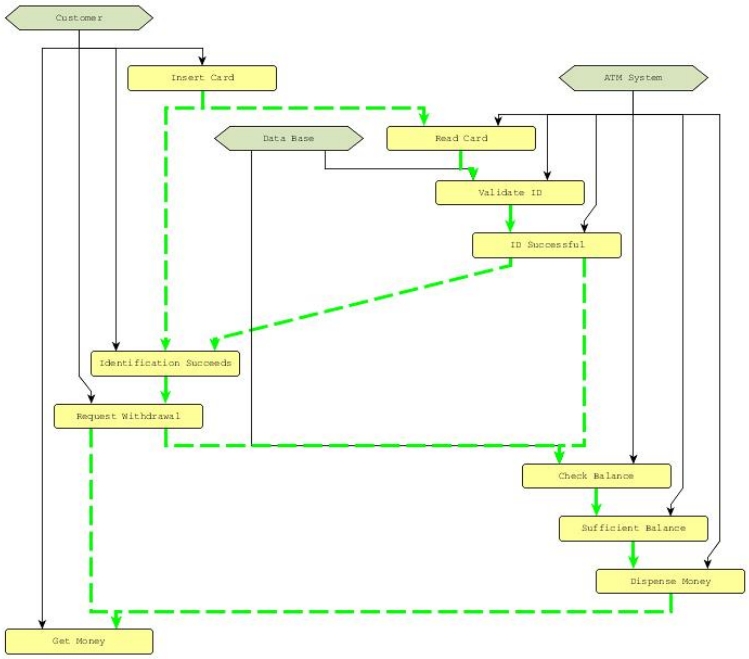

Figure 68. Sample trace with lost threads

All of the nodes, and all of the edges from Figure 5 are present in Figure 68, and all edges are routed to the correct nodes. However, some of the meaning of Figure 5 has nevertheless been lost. The groupings of nodes into rows in Figure 5 was meant to convey that all nodes in a particular row are part of a single thread. There is as yet no way to preserve this sort of information with MPGrapher.

yEd offers swim lanes, both as an editing feature and as a layout. Therefore, if MPGrapher, or a coordinated effort between MPGrapher and MP, could find a way to preserve this information, graphing the result in yEd should be relatively easy.

2. Graphical User Interface

As it stands, MPGrapher supports five types of nodes and four types of edges. Also, the display properties of those nodes and edges is fixed. Not only would changing the appearance of those nodes and edges be a painstaking processes, there is also no easy way of adding new types of nodes and edges in the future.

A graphical user interface would be helpful, especially one which showed examples of the current node and edge types of the program along with options to change those parameters. It would also be helpful if the interface provided a means to define entirely new node and edge types.

3. Type Checking and Input Validation

The MPGrapher currently accepts all input. If the input is correct, then the resulting graph is automatically displayed in yEd. However, if the input is incorrect, then nothing happens at all. It would be very nice if, before beginning the parsing process, the program validated the input file against GraphML and yEd-specific GraphML grammars.

Along with validation and type checking, it would also be quite helpful if the program output some sort of error messages.

4. XML Validation

All of the GraphML files that were output by MPGrapher for this project were validated against the GraphML and yEd schemas. However, this validation was

accomplished in a manual fashion. A true XML compiler should provide built-in validation that automatically validates each and every XML document it produces.

5. Process Multiple Input Files and Output Multiple Graphs

MPGrapher currently takes one and only one input file and produces one and only one output graph. It would be beneficial to enable the program to process multiple input files and produce multiple output graphs.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. CLASSES OF THE MPGRAPHER PACKAGE

A. THE MPGRAPHER CLASS

```
package MPGrapher;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

/**Opens a text-file for parsing, instantiates an MP_GraphML_Compiler object,
 * and initializes the yEd graphing application.
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 28, 2012
 */
public class MPGrapher {

    /**Opens a text-file for parsing, instantiates an MP_GraphML_Compiler
     * object, and initializes the yEd graphing application.
     * @param args
     * args[0] represents the text-file which is to be parsed.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library that is used by the MP_GraphML_Compiler class.
     */
    public static void main(String[] args) throws IOException {

        //String which will be used to represent the path to and name of the
        //output graphML file which the yEd graphing application is to display
        String yEd;

        //String which will contain the XML document that is to be written to
        //the output file
        String xmlOut;

        //object that will write xmlOut to outputFilename
        FileWriter out;

        //String which will be used to represent the current working directory
        //output graphML files will be written to this directory
        String currentDir;

        //String which will contain the name of the output graphML file
        String outputFilename;

        //File object which the application is to parse
        File file = new File(args[0]);

        //create a new MP_GraphML_Compiler object
        MP_GraphML_Compiler mgc = new MP_GraphML_Compiler();

        //fetch a complete, well-formed, yEd-specific graphML XML document via
        //the MP_GraphML_Compiler's getMPGraph method
        xmlOut = mgc.getMPGraph(file);

        //build an output filename, based on the name of the input file
        outputFilename = args[0].substring(0, args[0].length() - 4)
                + "_MPOutput.graphml";
```

```
        //instantiate a new FileWriter, using the filename created above
        out = new FileWriter(outputFilename);

        //write xml to the output file
        out.write(xmlOut);

        //close the output file (necessary to ensure that yEd is able to open
        //the file)
        out.close();

        //instantiate currentDir and set it to the current working directory
        currentDir = new File(".").getAbsolutePath();

        //instantiate yEd with the current directory and the name of the
        //graphML file to display
        yEd = "yEd.exe " + currentDir + "/" + outputFilename;

        //initialize and run yEd
        Runtime.getRuntime().exec(yEd);
    }
}
```

B. THE MP_GRAPHML_COMPILER CLASS

```
package MPGrapher;

import java.io.File;
import java.io.IOException;

/**Parses input files that conform to the MP-graph grammar, and compiles
 * graphML XML files to be displayed in the yEd graphing application (graphML
 * files conform to yWorks' graphML schema as defined by
 * <a href="http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
 * http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd</a>).
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 23, 2012
 */
public class MP_GraphML_Compiler extends yEd_GraphML_Compiler {

////////////////////////////////////////////////////////////////////////////
////variable class fields///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**An array of String values that hold the graphical layout parameters of
     * architecture-type data-nodes in the yEd Graphing application.
     */
    private String[] architectureDataNodeParams;

    /**An array of String values that hold the graphical layout parameters of
     * architecture-type processes-nodes in the yEd Graphing application.
     */
    private String[] architectureProcessesNodeParams;

    /**An array of String values that hold the graphical layout parameters of
     * architecture-type environment-nodes in the yEd Graphing application.
     */
    private String[] architectureEnvironmentNodeParams;

    /**An array of String values that hold the graphical layout parameters of
     * trace-type root-nodes in the yEd Graphing application.
```

```
     */
    private String[] traceRootNodeParams;

    /**An array of String values that hold the graphical layout parameters of
     * trace-type event-nodes in the yEd Graphing application.
     */
    private String[] traceEventNodeParams;

    /**An array of String values that hold the graphical layout parameters of
     * trace-type includes-edges in the yEd Graphing application.
     */
    private String[] edgeTraceIncludesParams;

    /**An array of String values that hold the graphical layout parameters of
     * trace-type precedes-edges in the yEd Graphing application.
     */
    private String[] edgeTracePrecedesParams;

    /**An array of String values that hold the graphical layout parameters of
     * architecture-type add-edges in the yEd Graphing application.
     */
    private String[] edgeArchitectureAddParams;

    /**An array of String values that hold the graphical layout parameters of
     * architecture-type share_all-edges in the yEd Graphing application.
     */
    private String[] edgeArchitectureShareAllParams;

////////////////////////////////////////////////////////////////////////////
////constructor/////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Instantiates a number of arrays and assigns values to them that will
     * be used as parameters for displaying edges and nodes in the yEd
     * graphing application.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library.
     */
    public MP_GraphML_Compiler() throws IOException {

        //instantiate the array to the proper size
        architectureEnvironmentNodeParams = new String[3];

        //enter the token which will signal the start of the architecture-
        //environment area of the input file
        architectureEnvironmentNodeParams[0] = "%architecture_environment%";

        //color all architecture-environment nodes the same color
        architectureEnvironmentNodeParams[1] = "#FCD5B5";

        //make all architecture-environment nodes the same shape
        architectureEnvironmentNodeParams[2] = "roundrectangle";

        //instantiate the array to the proper size
        //enter the token which will signal the start of the architecture-
        //processes area of the input file
        //color all architecture-processes nodes the same color
        //make all architecture-processes nodes the same shape
        architectureProcessesNodeParams = new String[3];

        //enter the token which will signal the start of the architecture-
```

```
//processes area of the input file
architectureProcessesNodeParams[0] = "%architecture_processes%";

//color all architecture-processes nodes the same color
architectureProcessesNodeParams[1] = "#D7E4BD";

//make all architecture-processes nodes the same shape
architectureProcessesNodeParams[2] = "roundrectangle";

//instantiate the array to the proper size
architectureDataNodeParams = new String[3];

//enter the token which will signal the start of the architecture-
//data area of the input file
architectureDataNodeParams[0] = "%architecture_data%";

//color all architecture-data nodes the same color
architectureDataNodeParams[1] = "#CCC1DA";

//make all architecture-data nodes the same shape
architectureDataNodeParams[2] = "roundrectangle";

//instantiate the array to the proper size
traceRootNodeParams = new String[3];

//enter the token which will signal the start of the trace-roots area
//of the input file
traceRootNodeParams[0] = "%trace_roots%";

//color all trace-roots nodes the same color
traceRootNodeParams[1] = "#D7E4BD";

//make all trace-roots nodes the same shape
traceRootNodeParams[2] = "hexagon";

//instantiate the array to the proper size
traceEventNodeParams = new String[3];

//enter the token which will signal the start of the trace-events area
//of the input file
traceEventNodeParams[0] = "%trace_events%";

//color all trace-events nodes the same color
traceEventNodeParams[1] = "#FFFF99";

//make all trace-events nodes the same shape
traceEventNodeParams[2] = "roundrectangle";

//instantiate the array to the proper size
edgeTraceIncludesParams = new String[5];

//enter the token which will indicate that a particular edge is a
//trace-includes type edge
edgeTraceIncludesParams[0] = "INCLUDES";

//color all trace-includes events the same color
edgeTraceIncludesParams[1] = "#000000";

//assign all trace-includes edges the same line thickness
edgeTraceIncludesParams[2] = "1.0";

//trace-includes edges are directed, which is indicated by the term
```

```
//"standard" (meaning standard arrows) in the yEd graphing application
edgeTraceIncludesParams[3] = "standard";

//assign all trace-includes edges the same line appearance
edgeTraceIncludesParams[4] = "line";

//instantiate the array to the proper size
edgeTracePrecedesParams = new String[5];

//enter the token which will indicate that a particular edge is a
//trace-precedes type edge
edgeTracePrecedesParams[0] = "PRECEDES";

//color all trace-precedes events the same color
edgeTracePrecedesParams[1] = "#00FF00";

//assign all trace-precedes edges the same line thickness
edgeTracePrecedesParams[2] = "4.0";

//trace-precedes edges are directed, which is indicated by the term
//"standard" (meaning standard arrows) in the yEd graphing application
edgeTracePrecedesParams[3] = "standard";

//assign all trace-precedes edges the same line appearance
edgeTracePrecedesParams[4] = "dashed";

//instantiate the array to the proper size
edgeArchitectureAddParams = new String[5];

//enter the token which will indicate that a particular edge is an
//architecture-add type edge
edgeArchitectureAddParams[0] = "ADD";

//color all architecture-add events the same color
edgeArchitectureAddParams[1] = "#00FF00";

//assign all architecture-add edges the same line thickness
edgeArchitectureAddParams[2] = "4.0";

//architecture-add edges are directed, which is indicated by the term
//"standard" (meaning standard arrows) in the yEd graphing application
edgeArchitectureAddParams[3] = "standard";

//assign all architecture-add edges the same line appearance
edgeArchitectureAddParams[4] = "line";

//instantiate the array to the proper size
edgeArchitectureShareAllParams = new String[5];

//enter the token which will indicate that a particular edge is an
//architecture-share_all type edge
edgeArchitectureShareAllParams[0] = "SHARE_ALL";

//color all architecture-share_all events the same color
edgeArchitectureShareAllParams[1] = "#FF99CC";

//instantiate the array to the proper size
//assign all architecture-share_all edges the same line thickness
edgeArchitectureShareAllParams[2] = "4.0";

//architecture-share_all edges are non-directed, which is indicated by
//the term "none" (meaning no arrows) in the yEd graphing application
```

```
        edgeArchitectureShareAllParams[3] = "none";

        //assign all architecture-share_all edges the same line appearance
        edgeArchitectureShareAllParams[4] = "line";
    }

////////////////////////////////////////////////////////////////////////////
////public method/////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed, yEd-specific graphML XML document.
     * @param fileToParse
     * String which reflects the "path to" and the "name of" the input file
     * which the application is to parse.
     * @return
     * A well-formed, yEd-specific graphML XML document.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library.
     */
    public String getMPGraph(File fileToParse) throws IOException {

        //instantiate return variable
        String xmlOut = "";

        //instantiate a two-dimensional array to hold various edge parameters
        String[][] graphEdgeParams = new String[4][5];

        //copy each of the arrays that were constructed by the constructor into
        //the two-dimensional graphEdgeParams array
        System.arraycopy(edgeTraceIncludesParams, 0, graphEdgeParams[0], 0, 5);
        System.arraycopy(edgeTracePrecedesParams, 0, graphEdgeParams[1], 0, 5);
        System.arraycopy(edgeArchitectureAddParams, 0, graphEdgeParams[2], 0,
                5);
        System.arraycopy(edgeArchitectureShareAllParams, 0, graphEdgeParams[3],
                0, 5);

        //instantiate a two-dimensional array to hold various node parameters
        String[][] graphNodeParams = new String[5][3];

        //copy each of the arrays that were constructed by the constructor into
        //the two-dimensional graphNodeParams array
        System.arraycopy(traceRootNodeParams, 0, graphNodeParams[0], 0, 3);
        System.arraycopy(traceEventNodeParams, 0, graphNodeParams[1], 0, 3);
        System.arraycopy(architectureEnvironmentNodeParams, 0,
                graphNodeParams[2], 0, 3);
        System.arraycopy(architectureProcessesNodeParams, 0,
                graphNodeParams[3], 0, 3);
        System.arraycopy(architectureDataNodeParams, 0, graphNodeParams[4], 0,
                3);

        //instantiate a new yEd_GraphML_Compiler with the two-dimensional
        //arrays created above
        yEd_GraphML_Compiler ygc = new yEd_GraphML_Compiler(fileToParse,
                graphEdgeParams, graphNodeParams);

        //set the return value to the graphML document returned by the
        //yEd_GraphML_Compiler object
        xmlOut = ygc.get_yEdGraph();
        return xmlOut;
    }
}
```

C. THE YED_GRAPHML_COMPILER CLASS

```
package MPGrapher;

import java.io.File;
import java.util.Arrays;
import java.io.IOException;
import java.util.Comparator;

/**Compiles graphML XML files for the yEd graphing application (files conform
 * to yWorks' graphML schema as defined by
 * <a href="http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
 * http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd</a>).
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 31, 2012
 */
public class yEd_GraphML_Compiler extends GraphML_Compiler {

////////////////////////////////////////////////////////////////////////////
////variable class fields///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field used to track the dimensions of the layout matrix (the matrix is a
     * dims -by- dims square matrix) - used to place nodes on screen in the yEd
     * graphing application (dims is an abbreviation of matrix-"dimensions").
     */
    private int dims;

    /**Field used to track the current row of the layout matrix - used to place
     * nodes on the screen.
     */
    private int curRow;

    /**Field used to track the current column of the layout matrix - used to
     * place nodes on the screen.
     */
    private int curCol;

    /**Field which contains the current token of the input file as returned by
     * the scanner.
     * <p />curTok is an abbreviation and concatenation of the words "current"
     * and "token".
     */
    private String curTok;

    /**Field which is used to place nodes on the screen with uniform space
     * along the X-axis.
     */
    private double colFac;

    /**Field which is used to place nodes on the screen with uniform space
     * along the Y-axis.
     */
    private double rowFac;

    /**Field which is used to track the number of nodes of input/output files
     * (number of node IDs = number of nodes in the graph).
     */
    private int numNodeIDs;
```

```
////////////////////////////////////////////////////////////////////////////
////class fields which correspond to parent class' constants//////////////////
////////////////////////////////////////////////////////////////////////////

    /**GraphML grammar constant which represents the end of the nodes section
     * and the beginning of the edges section (as defined in the
     * GraphML_Compiler class). Used for parsing input documents.
     */
    private String EDGES;

    /**GraphML grammar constant which represents the beginning of the nodes
     * section (as set in the GraphML_Compiler class). Used for parsing
     * input documents.
     */
    private String NODES;

    /**GraphML grammar constant which represents the end of an edge from the
     * input file (as set in the GraphML_Compiler class). Used for parsing
     * input documents.
     */
    private String END_EDGE;

    /**Constant which provides standard increments for array sizes (as set in
     * the GraphML_Compiler class): all arrays within the application are
     * assigned the same size, and grow by the same amount when necessary.
     */
    private int ARRAY_MULTIPLE;

    /**GraphML grammar constant which represents the end of a graph from the
     * input file (as set in the GraphML_Compiler class). All tokens after this
     * are ignored.
     */
    private String END_OF_GRAPH;

////////////////////////////////////////////////////////////////////////////
////yEdEdge fields/////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which stores the total number of possible output yEdEdge types.*/
    private int numEdgeTypes;

    /**The edgeIDs field serves as the key to yEdEdge object values in a
     * key-value pair mapping relationship. Number of edgeIDs = number of
     * yEdEdges.
     * <p /> Used to find objects in a yEdEdge[] array by means of the
     * binarySearch() method (provided by the java.util.Arrays library).
     */
    private String[] edgeIDs;

    /**The edgeTypes field serves as the key to edgeColors, edgeWeights,
     * edgesDirected, and edgeLineStyles object values in a key-value pair
     * mapping relationship.
     * <p /> Used to find objects in edgeColors, edgeWeights, edgesDirected and
     * edgeLineStyle arrays by means of the binarySearch() method (provided by
     * the java.util.Arrays library).
     */
    private String[] edgeTypes;

    /**Field which contains yEdEdge objects.*/
    private yEdEdge[] yEdEdges;

    /**Field used to color lines of edges (according to the yEdEdge's
```

```
     * corresponding edgeType).
     */
    private String[] edgeColors;

    /**Field of weights which are used to control the line thickness of output
     * edges (according to the yEdEdge's corresponding edgeType).
     */
    private double[] edgeWeights;

    /**This field is a reusable integer which is re-set to zero upon the
     * construction of a new yEdEdge and is incremented to reflect the number
     * of labels of a particular yEdEdge.
     */
    private int tempEdgeLabelCount;

    /**Field used to indicate whether a particular edgeType is directed or
     * undirected.
     */
    private String[] edgesDirected;

    /**Field of line-styles (dashed, for example) which are used to control the
     * line-styles of output edges (according to the yEdEdge's corresponding
     * edgeType).
     */
    private String[] edgeLineStyles;

    /**This field is a reusable array of String objects which is used to
     * reflect the various texts of output labels of a particular yEdEdge.
     */
    private String[] tempEdgeLabels;

////////////////////////////////////////////////////////////////////////////
////yEdNode fields//////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which stores the total number of possible output yEdNode types.*/
    private int numNodeTypes;

    /**Field used to uniformly control the width of nodes of output graphs.*/
    private double nodeWidth;

    /**The nodeIDs field serves as the key to yEdNode object values in a
     * key-value pair mapping relationship. Number of nodeIDs = number of
     * yEdNodes.
     * <p /> Used to find objects in a yEdNode[] array by means of the
     * binarySearch() method (provided by the java.util.Arrays library).
     */
    private String[] nodeIDs;

    /**Field used to uniformly control the width of nodes of output graphs.*/
    private double nodeHeight;

    /**The nodeTypes field serves as the key to nodeColors, and nodeShapes
     * object values in a key-value pair mapping relationship.
     * <p /> Used to find objects in nodeColors and nodeShapes arrays by means
     * of the binarySearch() method (provided by the java.util.Arrays library).
     */
    private String[] nodeTypes;

    /**Field which contains yEdNode objects.*/
    private yEdNode[] yEdNodes;
```

```
    /**Field of colors which are used to control the fill color of output nodes
     * (according to the yEdNode's corresponding nodeType).
     */
    private String[] nodeColors;

    /**Field of shapes which are used to control the shape of output nodes
     * (according to the yEdNode's corresponding nodeType).
     */
    private String[] nodeShapes;

    /**Field used to track the String representation of the longest node label
     * (used to control the width of output node shapes according to the
     * yEdNode's corresponding nodeType).
     */
    private int longestLabelName;

////////////////////////////////////////////////////////////////////////////
////class constants//////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Constant which controls the screen-space between output nodes in a
     * uniform manner.
     */
    private static final int NODE_SPACE = 150;

    /**Constant which provides a uniform height to output nodes.*/
    private static final double NODE_HEIGHT = 30.0;

////////////////////////////////////////////////////////////////////////////
////yEd graphML constants////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**yEd graphML grammar constant which is used to indicate whether a given
     * input file represents a generic graphML graph or a graph to be presented
     * in the yEd graphing applications. Used for parsing input documents.
     */
    private static final String YED_GRAPH_TOKEN = "%yEd%";

    /**yEd graphML XML element: provides additional references to the
     * &ltgraphml> element which defines a graphML XML document.
     */
    private static final String YED_GRAPHML_REFS
        = "\txmlns:y=\"http://www.yworks.com/xml/graphml\"\n"
        + "\txmlns:yed=\"http://www.yworks.com/xml/yed/3\"\n";

    /**yEd graphML XML element: defines the yEd-specific graphML schema
     * attribute of a &ltgraph> element.
     */
    private static final String YED_GRAPHML_SCHEMA
        = "\n\thttp://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd";

    /**yEd graphML XML elements: provides yEd-specific graphML &ltkey> elements
     * which are necessary for the proper display of graphs within the yEd
     * graphing application.
     */
    private static final String YED_GRAPHML_KEYS =
        "\t<key for=\"graphml\" id=\"d0\" yfiles.type=\"resources\"/>\n"
        + "\t<key for=\"port\" id=\"d1\" yfiles.type=\"portgraphics\"/>\n"
        + "\t<key for=\"port\" id=\"d2\" yfiles.type=\"portgeometry\"/>\n"
        + "\t<key for=\"port\" id=\"d3\" yfiles.type=\"portuserdata\"/>\n"
        + "\t<key attr.name=\"url\" attr.type=\"string\" for=\"node\" "
            + "id=\"d4\"/>\n"
```

```
        + "\t<key attr.name=\"description\" attr.type=\"string\" for=\"node\" "
            + "id=\"d5\"/>\n"
        + "\t<key for=\"node\" id=\"d6\" yfiles.type=\"nodegraphics\"/>\n"
        + "\t<key attr.name=\"url\" attr.type=\"string\" for=\"edge\" "
            + "id=\"d7\"/>\n"
        + "\t<key attr.name=\"description\" attr.type=\"string\" for=\"edge\" "
            + "id=\"d8\"/>\n"
        + "\t<key for=\"edge\" id=\"d9\" yfiles.type=\"edgegraphics\"/>\n"
        + "\t<key attr.name=\"Description\" attr.type=\"string\" for=\"graph\""
            + " id=\"d10\"/>\n\n";

    /**yEd graphML XML element: provides a value for the key="" attribute of a
     * well-formed graphML graph-&lt;data> element.
     */
    private static final String YED_GRAPH_DATA_TAG =
            "\t\t<data key=\"d10\"/>\n";

    /**yEd graphML XML element: provides a value for the key="" attribute of a
     * well-formed graphML edge-&lt;data> element and defines an additional
     * edge-&lt;data> element as well.
     */
    private static final String EDGE_DATA = "d8\"/>\n"
            + "\t\t\t<data key=\"d9";

    /**yEd graphML XML element: provides a value for the key="" attribute of a
     * well-formed graphML node-&lt;data> element and defines an additional
     * node-&lt;data> element as well.
     */
    private static final String NODE_DATA = "d5\"/>\n"
            + "\t\t\t<data key=\"d6";

    /**yEd graphML XML element: provides a value for the key="" attribute of a
     * well-formed graphML graph-&lt;data> element, as well as an additional
     * yEd-specific &lt;y:Resources/> element.
     */
    private static final String YED_GRAPHML_FOOTER
        = "\t<data key=\"d0\">\n"
        + "\t\t<y:Resources/>\n"
        + "\t</data>\n";

////////////////////////////////////////////////////////////////////////////
////getter methods for yEd graphML constants/////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of YED_GRAPH_DATA_TAG to child
     * classes.
     * @return Value of private YED_GRAPH_DATA_TAG constant.
     */
    private static final String GET_YED_GRAPH_DATA_TAG() {
        return YED_GRAPH_DATA_TAG;}

    /**Getter method which returns the values of GRAPHML_REFS and
     * YED_GRAPHML_REFS to child classes.
     * @return Values of private GRAPHML_REFS and YED_GRAPHML_REFS constants.
     */
    protected static final String GET_YED_GRAPHML_REFS() {

        //instantiates a String object and assigns to it the value of the
        //GRAPHML_REFS constant from the GraphML_Compiler parent-class
        String xmlOut = GET_GRAPHML_REFS();

        //concatenates the value of the YED_GRAPHML_REFS constant to xmlOut
```

```
        xmlOut += YED_GRAPHML_REFS;

        return xmlOut;
    }

    /**Getter method which returns the values of GRAPHML_SCHEMA,
     * YED_GRAPHML_SCHEMA, and GRAPHML_SCHEMA_CLOSE to child classes.
     * @return Values of private GRAPHML_SCHEMA, YED_GRAPHML_SCHEMA and
     * GRAPHML_SCHEMA_CLOSE constants.
     */
    protected static final String GET_YED_GRAPHML_SCHEMA() {

        //instantiates a String object and assigns to it the value of the
        //GRAPHML_SCHEMA constant from the GraphML_Compiler parent-class
        String xmlOut = GET_GRAPHML_SCHEMA();

        //concatenates the value of the YED_GRAPHML_SCHEMA constant to xmlOut
        xmlOut += YED_GRAPHML_SCHEMA;

        //concatenates the value of the GRAPHML_SCHEMA_CLOSE constant from the
        //GraphML_Compiler parent-class to xmlOut to complete a well-formed
        //<graphml> element
        xmlOut += GET_GRAPHML_SCHEMA_CLOSE();

        return xmlOut;
    }

    /**Getter method which returns the values of YED_GRAPHML_KEYS,
     * GRAPHML_GRAPH_OPEN, and YED_GRAPH_DATA_TAG to child classes.
     * @return Values of private GRAPHML_KEYS, GRAPHML_GRAPH_OPEN and
     * YED_GRAPH_DATA_TAG constants.
     */
    protected static final String GET_YED_GRAPHML_GRAPH_OPEN() {

        //instantiates a String object and assigns to it the value of the
        //YED_GRAPHML_KEYS constant which are required for the proper display
        //of graphs in the yEd graphing application
        String xmlOut = YED_GRAPHML_KEYS;

        //concatenates the values of the GRAPHML_GRAPH_OPEN constant from the
        //GraphML_Compiler parent-class and the YED_GRAPH_DATA_TAG to xmlOut to
        //complete a well-formed, yEd-specific <graph> element
        xmlOut += GET_GRAPHML_GRAPH_OPEN() + YED_GRAPH_DATA_TAG;

        return xmlOut;
    }

    /**Getter method which returns the values of GRAPHML_GRAPH_CLOSE,
     * YED_GRAPHML_FOOTER and GRAPHML_FOOTER to child classes.
     * @return Values of private GRAPHML_GRAPH_CLOSE, YED_GRAPHML_FOOTER and
     * GRAPHML_FOOTER constants.
     */
    protected static final String GET_YED_GRAPHML_GRAPH_CLOSE() {

        //instantiates a String object and assigns to it the value of the
        //GRAPHML_GRAPH_CLOSE constant from the GraphML_Compiler parent-class,
        //plus the value of the YED_GRAPHML_FOOTER constant, plus the value of
        //the GRAPHML_FOOTER constant from the GraphML_Compiler parent class.
        String xmlOut = GET_GRAPHML_GRAPH_CLOSE();
        xmlOut += YED_GRAPHML_FOOTER;
        xmlOut += GET_GRAPHML_FOOTER();
```

```
        //the end-result of xmlOut represents a compilation of well-formed
        //yEd-specific graphML elements which are necessary to properly close
        //out an XML document
        return xmlOut;
    }

////////////////////////////////////////////////////////////////////////////
////constructors////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library.
     */
    public yEd_GraphML_Compiler() throws IOException {}

    /**Constructor which opens the input file for parsing - only used for
     * compiling basic graphML graphs.
     * @param fileToParse
     * String which reflects the "path to" and the "name of" the input file
     * which the application is to parse.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library.
     */
    public yEd_GraphML_Compiler(File fileToParse) throws IOException {

        //instantiate various fields to default values
        numNodeIDs = numEdgeTypes = numNodeTypes = 0;
        dims = curCol = curRow = longestLabelName = 0;

        //all nodes are drawn at a standard height by yEd
        nodeHeight = 30.0;

        //open the file for parsing
        openFile(fileToParse);

        //fetch constant values from the GraphML_Compiler parent class
        ARRAY_MULTIPLE = GET_ARRAY_MULTIPLE();
        EDGES = GET_EDGES_TOKEN();
        NODES = GET_NODES_TOKEN();
        END_EDGE = GET_END_EDGE_TOKEN();
        END_OF_GRAPH = GET_END_OF_GRAPH_TOKEN();
    }

    /**Constructor which opens the input file for parsing and compiling graphs
     * - used specifically for compiling graphs to be displayed in the yEd
     * graphing application.
     * @param fileToParse
     * String which reflects the "path to" and the "name of" the input file
     * which the application is to parse. <p />
     * @param edgeTypesParams
     * Two-dimensional array of String objects which represent the 5 parameters
     * which are necessary to display edges in the yEd graphing application.
     *  <p />
     * @param nodeTypesParams
     * Two-dimensional array of String objects which represent the 3 parameters
     * which are necessary to display nodes in the yEd graphing application.
     * @throws IOException
```

```
 * Required by the java.io.File library, which is in turn required by the
 * java.util.Scanner library.
 */
public yEd_GraphML_Compiler(File fileToParse,
        String[][] edgeTypesParams, String[][] nodeTypesParams)
        throws IOException {

    //instantiate various fields to default values
    numNodeIDs = dims = curCol = curRow = longestLabelName = 0;

    //all nodes are drawn at a standard height by yEd
    nodeHeight = 30.0;

    //open the file for parsing
    openFile(fileToParse);

    //fetch constant values from the GraphML_Compiler parent class
    EDGES = GET_EDGES_TOKEN();
    NODES = GET_NODES_TOKEN();
    END_EDGE = GET_END_EDGE_TOKEN();
    END_OF_GRAPH = GET_END_OF_GRAPH_TOKEN();
    ARRAY_MULTIPLE = GET_ARRAY_MULTIPLE();

    //the edgeTypesParams array needs to be sorted in order for the
    //java.util.Array's binarySearch() method to work properly
    //algorithm provided by "Bert F" at "http://stackoverflow.com/
    //questions/4907683/sort-a-two-dimensional-array-based-on-one-column"
    Arrays.sort(edgeTypesParams, new Comparator<String[]>() {
        @Override
        public int compare(final String[] entry1, final String[] entry2) {
            final String type1 = entry1[0];
            final String type2 = entry2[0];
            return type1.compareTo(type2);
        }
    });

    //instantiate a new array to hold edge labels
    tempEdgeLabels = new String[ARRAY_MULTIPLE];

    //the number of possible yEdEdge types is equal to the size of the
    //edgeTypesParams array
    numEdgeTypes = edgeTypesParams.length;

    //instantiate an array to hold edgeTypes and populate it with values
    //from the [x][0] dimension of edgeTypesParams
    edgeTypes = new String[numEdgeTypes];
    for (int i = 0; i < numEdgeTypes; i++)
        edgeTypes[i] = edgeTypesParams[i][0];

    //instantiate an array to hold edgeColors and populate it with values
    //from the [x][1] dimension of edgeTypesParams
    edgeColors = new String[ARRAY_MULTIPLE];
    for (int i = 0; i < numEdgeTypes; i++)
        edgeColors[i] = edgeTypesParams[i][1];

    //instantiate an array to hold edgeWeights and populate it with values
    //from the [x][2] dimension of edgeTypesParams
    edgeWeights = new double[ARRAY_MULTIPLE];
    for (int i = 0; i < numEdgeTypes; i++)
        edgeWeights[i] = Double.valueOf(edgeTypesParams[i][2]);

    //instantiate an array to hold edgesDirected and populate it with
```

```
        //values from the [x][3] dimension of edgeTypesParams
        edgesDirected = new String[ARRAY_MULTIPLE];
        for (int i = 0; i < numEdgeTypes; i++)
            edgesDirected[i] = edgeTypesParams[i][3];

        //instantiate an array to hold edgeLineStyles and populate it with
        //values from the [x][4] dimension of edgeTypesParams
        edgeLineStyles = new String[ARRAY_MULTIPLE];
        for (int i = 0; i < numEdgeTypes; i++)
            edgeLineStyles[i] = edgeTypesParams[i][4];

        //the nodeTypesParams array needs to be sorted in order for the
        //java.util.Array's binarySearch() method to work properly
        //algorithm provided by "Bert F" at "http://stackoverflow.com/
        //questions/4907683/sort-a-two-dimensional-array-based-on-one-column"
        Arrays.sort(nodeTypesParams, new Comparator<String[]>() {
            @Override
            public int compare(final String[] entry1, final String[] entry2) {
                final String type1 = entry1[0];
                final String type2 = entry2[0];
                return type1.compareTo(type2);
            }
        });

        //the number of possible yEdNode types is equal to the size of the
        //edgeTypesParams array
        numNodeTypes = nodeTypesParams.length;

        //instantiate an array to hold nodeTypes and populate it with values
        //from the [x][0] dimension of nodeTypesParams
        nodeTypes = new String[numNodeTypes];
        for (int i = 0; i < numNodeTypes; i++)
            nodeTypes[i] = nodeTypesParams[i][0];

        //instantiate an array to hold nodeColors and populate it with values
        //from the [x][1] dimension of nodeTypesParams
        nodeColors = new String[numNodeTypes];
        for (int i = 0; i < numNodeTypes; i++)
            nodeColors[i] = nodeTypesParams[i][1];

        //instantiate an array to hold nodeShapes and populate it with values
        //from the [x][2] dimension of nodeTypesParams
        nodeShapes = new String[numNodeTypes];
        for (int i = 0; i < numNodeTypes; i++)
            nodeShapes[i] = nodeTypesParams[i][2];
    }

////////////////////////////////////////////////////////////////////////////
////public method///////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed, yEd-specific graphML XML document.
     * @return
     * A well-formed, yEd-specific graphML XML document.
     */
    public String get_yEdGraph() {

        //instantiate return variable
        String xmlOut = "";

        //look-up variable used for searching nodeTypes
        int tempNodeType = 0;
```

```
        //variable used to determine whether the input file represents a basic
        //graphML graph or a yEd-specific graph
        boolean yEdGraph = false;

        //variable used as a stop-value to stop scanning and begin parsing
        boolean foundNodeType = false;

        //step through the input document until either 1) the %yEd% token is
        //discovered, or 2) the %%nodes%% token is discovered
        while (!foundNodeType) {

            curTok = advance();

            //if curTok == %yEd%, the input file represents a yEd-specific
            //graph
            if(curTok.equals(YED_GRAPH_TOKEN)) yEdGraph = true;

            //if the %%nodes%% token is reached without an intervening %yEd%
            //token, then the input file represents a basic graphML graph
            if(curTok.equals(NODES) && !yEdGraph) foundNodeType = true;

            else {

                //search the nodeTypes array for the current token; if found,
                //then the current token represents the start a nodes section
                //of interest within the input file. Begin creating Node
                //objects from this point
                tempNodeType = Arrays.binarySearch(nodeTypes, curTok);
                if (tempNodeType >= 0) foundNodeType = true;
            }
        }

        //if the input file represents an basic graphML graph, then call the
        //getGraph() method of the GraphML_Compiler parent-class (assumes that
        //there will be no <key> elements in the input file).
        if (!yEdGraph) {

            xmlOut = getGraph();

        //otherwise, compile a yEd-specific graph
        } else {

            //compile the proper elements to begin a well-formed, yEd-specific
            //graphML XML output file
            xmlOut = GET_YED_GRAPHML_REFS();
            xmlOut += GET_YED_GRAPHML_SCHEMA();
            xmlOut += GET_YED_GRAPHML_GRAPH_OPEN();

            //calls the method that compiles well-formed, yEd-specific <node>
            //and <edge> graphML XML elements
            xmlOut += statements(yEdGraph, tempNodeType);

            //compile the proper elements to close out a well-formed,
            //yEd-specific graphML XML output file
            xmlOut += GET_YED_GRAPHML_GRAPH_CLOSE();
        }

        return xmlOut;
    }

////////////////////////////////////////////////////////////////////////////
```

```
////private and protected methods/////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////

    /**Scans the input document and compiles and returns well-formed,
     * yEd-specific &lt;node> and &lt;edge> graphML XML elements for output.
     * @param yEdGraph
     * Indicates whether a yEd-specific graph or a basic graphML graph is to be
     * created. <p />
     * @param tempNodeType
     * Re-usable look-up variable which represents the current type of yEdNode
     * object that is to be instantiated, as well as serving as a placeholder
     * to be used for searching nodeTypes for instantiating other types of
     * yEdNode objects.
     * @return
     * Well-formed, yEd-specific &lt;node> and &lt;edge> graphML XML elements
     */
    private String statements(boolean yEdGraph, int tempNodeType) {

        //instantiate return variable
        String xmlOut = "";

        //compile either basic graphML <node> or yEd-specific <node> elements,
        //based on the value of yEdGraph
        if (!yEdGraph) xmlOut = nodes();
        else xmlOut = yEdNodes(tempNodeType);

        //compile either basic graphML <edge> or yEd-specific <edge> elements,
        //based on the value of yEdGraph
        if (!yEdGraph) xmlOut += edges();
        else xmlOut += yEdEdges();

        return xmlOut;
    }

    /**Scans the input document and compiles and returns well-formed,
     * yEd-specific graphML XML &lt;node> elements for output.
     * @param tempNodeType
     * Re-usable look-up variable which represents the current type of yEdNode
     * object that is to be instantiated, as well as serving as a placeholder
     * to be used for searching nodeTypes for instantiating other types of
     * yEdNode objects.
     * @return
     * Well-formed, yEd-specific &lt;node> graphML XML elements
     */
    protected String yEdNodes(int tempNodeType) {

        //local variable which tracks the number of instantiated yEdNode
        //objects
        int localCtr = 0;

        //instantiate the return variable
        String xmlOut = "";

        //re-useable String that holds the current value of a yEdNode label
        String curLabel = "";

        //an array of String objects that will be used to instantiate the final
        //collection of nodeIDs[]
        String[] tempNodeIDs = new String[ARRAY_MULTIPLE];

        //an array of String objects that will be used to instantiate the final
        //collection of yEdNodes[]
```

```
yEdNode[] temp_yEdNodes = new yEdNode[ARRAY_MULTIPLE];

//advance to the id of the first yEdNode
curTok = advance();

//parse and instantiate yEdNode objects until the edges portion of the
//input file is reached
while (!curTok.equals(EDGES)) {

    //the current token of the input file represents the id of a
    //yEdNode
    tempNodeIDs[localCtr] = curTok;

    //advance to the label portion of the current yEdNode
    curTok = advance();

    /**insert non-node label exception here*/

    //yEdNode labels may contain multiple tokens, so interpret
    //everything between the current "-symbol and the next "-symbol as
    //a single yEdNode label
    if (curTok.startsWith("\"") && !curTok.equals("\"\"")) {

        //strip off the leading "-symbol before adding it to the label
        //String
        curLabel = curTok.substring(1, curTok.length());

        //if the current token represents the entire yEdNode label,
        //then strip off the trailing "-symbol as well and advance to
        //the next token
        if (curTok.endsWith("\"")) {

            curLabel = curLabel.substring(0, curLabel.length() - 1);

        //otherwise, process additional tokens as part of the current
        //yEdNode label
        } else {

            curTok = advance();

            //a token which ends with the "-symbol represents the end
            //of the current yEdNode label
            while (!curTok.endsWith("\"")) {

                curLabel += " " + curTok;
                curTok = advance();
            }

            //strip off the trailing "-symbol, add it to the current
            //label String, and advance to the next token
            curLabel += " " + curTok.substring(0, curTok.length() - 1);
        }

        //the screen-width of output nodes will depend upon the length
        //of the longest yEdNode label
        if (curLabel.length() > longestLabelName)
            longestLabelName = curLabel.length();

        //the following if-block checks to see if it is necessary to
        //increase the size of the temp_yEdNodes array in order to
        //avoid "indexOutOfBounds" errors
        if (localCtr % ARRAY_MULTIPLE == 0 && localCtr > 0) {
```

```
            //create a new array equal to the size of the current
            //array, + the size of ARRAY_MULTIPLE
            yEdNode[] tmp = new yEdNode[temp_yEdNodes.length
                    + ARRAY_MULTIPLE];

            //copy the contents of the old array to the new one
            System.arraycopy(temp_yEdNodes, 0, tmp, 0,
                    temp_yEdNodes.length);

            //set the old, smaller array to the new, larger one
            temp_yEdNodes = tmp;
        }

        //instantiate a new yEdNode with the parameters established
        //above and add it to the temp_yEdNodes array
        temp_yEdNodes[localCtr] = new yEdNode(tempNodeIDs[localCtr],
                NODE_DATA, nodeHeight, 0.0, 0.0, 0.0,
                nodeColors[tempNodeType], curLabel,
                nodeShapes[tempNodeType]);
    }

    curTok = advance();

    //the current token must either be the start of a new node, the
    //start of the next nodeType section, or the start of the edges
    //section of the input file
    if (curTok.matches("%.*%")) {

        if (!curTok.equals(EDGES)) {

            tempNodeType = Arrays.binarySearch(nodeTypes, curTok);
            curTok = advance();
        }
    }

    //increment the localCtr to reflect the number of yEdNodes
    //discovered so far
    localCtr++;
}

//set the final number of yEdNodes to the value of the local counter
numNodeIDs = localCtr;

//for the fixed-width, 12 pt, Courier New font used to display node
//labels in the yEd graphing application, the number of characters of
//the longest node label * 8 works best as a uniform node-width
nodeWidth = longestLabelName * 8;

//determine the output screen-placement of each yEdNode
for (int i = 0; i < numNodeIDs; i++) {

    //the constructNodeCoords() method determines values for rowFac and
    //colFac that are used to set the output X and Y screen-coordinates
    //of each yEdNode
    constructNodeCoords();
    temp_yEdNodes[i].setNodeWidth(nodeWidth);
    temp_yEdNodes[i].setNodeX(rowFac);
    temp_yEdNodes[i].setNodeY(colFac);

    //compile well-formed, yEd-specific graphML XML <node> elements
    //based on each yEdNode's properties
```

```
            xmlOut += temp_yEdNodes[i].get_yEdNodeXML();
        }

        //the nodeIDs array is necessary in order to find the correct spelling
        //id="" attributes of <node> elements when building source="" and
        //target="" attributes of yEdEdge objects

        //build a new nodeID[] array of the proper size (not only will the
        //proper size free up unnecessary memory, it is also necessary to
        //avoid the nullPointer exception of the java.util.Array's
        //binarySearch() method)
        nodeIDs = new String[numNodeIDs];

        //copy the appropriate IDs from the old tempNodeIDs array to the new
        //nodeIDs array
        System.arraycopy(tempNodeIDs, 0, nodeIDs, 0, numNodeIDs);

        //sort the new nodeIDs array
        Arrays.sort(nodeIDs);

        //instantiate a the final collection of yEdNodes
        yEdNodes = new yEdNode[numNodeIDs];

        //the following for-block copies yEdNodes from the old temp_yEdNodes
        //array to the final yEdNodes array according to the order established
        //by the sorted nodeIDs array

        //this block establishes a sorted, key-value pair mapping of nodeIDs to
        //yEdNodes
        for (int i = 0; i < nodeIDs.length; i++) {

            int index = Arrays.binarySearch(nodeIDs, tempNodeIDs[i]);
            yEdNodes[index] = temp_yEdNodes[i];
        }

        return xmlOut;
    }

    /**Calculates and sets values for the rowFac and colFac fields, which are
     * used to layout nodes in a matrix pattern in the yEd graphing
     * application.
     */
    private void constructNodeCoords() {

        //every other node is shifted by a standard amount, either to the
        //right,  down, or both, in order to minimize nodes overlapping edges
        //in the output application
        //OFFSET provides the standard offset space
        double OFFSET = NODE_SPACE / 2;

        //determine the proper placement of nodes along the X-axis - we need to
        //account for both the desired space between nodes, as well as the
        //space that's already been taken up by the nodes which have been
        //created so far
        colFac = curCol * NODE_SPACE + curCol * nodeWidth;

        //determine the proper placement of nodes along the Y-axis - if curRow
        //is larger than 0.0, then simply multiply the curRow (the row upon
        //which to place the current node) by NODE_SPACE; otherwise, set
        //rowFac == 0
        if (curRow > 0) rowFac = curRow * NODE_SPACE;
        else rowFac = 0.0;
```

```
//offset every other row
if (curCol % 2 == 1) rowFac += OFFSET;

//stagger every-other column by a standard distance
if (curRow % 2 == 1) colFac += OFFSET;

//the following if-elseif block keeps track of where the next node is
//to be placed in the matrix

//if dims == 0, then no nodes have been placed yet and the current
//matrix size is 1 X 1
if (dims == 0) {

    //the matrix needs to be 2 X 2 in order to place the next node, so
    //increment dims
    dims++;

    //the matrix is filled in left-to-right and bottom-to-top, with
    //(0,0) representing the top-left corner of the bottom and
    //(dims,dims) representing the bottom-right
    //in order to fill the matrix in from the bottom left, we need to
    //set curRow = dims (curCol already == 0 so there's no need to
    //increment it yet).
    curRow = dims;

//continue to fill in the bottom row of the matrix until curCol == dims
} else if (curCol < dims) {

    //increment curCol in order to continue filling the matrix in from
    //left-to-right along the bottom row
    curCol++;

//if curCol == dims and dims == curRow, then we've reached the bottom-
//right corner of the matrix - it's time to start filling in the
//farthest-right column, from bottom-to-top
} else if (curCol == dims && dims == curRow) {

    //decrement curRow in order to fill in the next higher row along
    //the current column
    curRow--;

//continue filling in the farthest-right column until curRow ==
} else if (curRow > 0) {

    //decrement curRow in order to fill in the next higher row along
    //the current column
    curRow--;

//if none of the above cases hold, then the curRow == 0 and
//curCol == dims
//it's time to increase the size of the matrix and begin filling it in
//from left-to-right, bottom-to-top again
} else {

    //increment dims in order to increase the size of the layout-matrix
    dims++;

    //set curRow = dims in order to place the next node on the bottom-
    //most row
    curRow = dims;
```

```
            //set curCol = 0 in order to place the next node on the left-most
            //column
            curCol = 0;
        }
    }

    /**Constructs an array of yEdEdge objects and iteratively calls the
     * get_yEdEdgeXML method of each one in order to build well-formed, yEd-
     * specific graphML XML &lt;edge> elements.
     * @return
     * A String representing zero or more well-formed, yEd-specific graphML
     * XML &lt;edge> elements.
     */
    protected String yEdEdges() {

        //instantiate the return variable
        String xmlOut = "";

        //parse the input document and construct an array of yEdEdge objects
        constructEdges();

        //for each yEdEdge constructed in the command, compile a well-formed,
        //yEd-specific &lt;edge> element and add it to the output String
        for (int i = 0; i < yEdEdges.length; i++)
            xmlOut += yEdEdges[i].get_yEdEdgeXML();

        return xmlOut;
    }

    /**Parses the input file and constructs an array of yEdEdge objects.
     */
    private void constructEdges() {

        //String which will be used to fill the yEdEdge object's id field
        String id;

        //counter which tracks the total number of instantiated yEdEdge objects
        int ctr = 0;

        //re-usable look-up integer used to find yEdNode objects which
        //correspond to "source" and "target" String values
        int index = 0;

        //re-usable String used to populate the source field of yEdEdge objects
        String source;

        //re-usable String used to populate the target field of yEdEdge objects
        String target;

        //re-usable String used to populate the type field of yEdEdge objects
        String type = "";

        //re-usable variable used to populate the numLabels field of yEdEdge
        //objects
        int numEdgeLabels;

        //instantiate a new String array to hold the current yEdEdge's labels
        //--note-- the program assumes that there will not be more than 1000
        //labels for any particular edge
        tempEdgeLabels = new String[1000];

        //a temporary placeholder for building yEdEdge IDs
```

```
//when completed, the edgeIDs[] field will be set to a trimmed-down
//version of this variable
String[] tempEdgeIDs;

//a temporary placeholder for building yEdEdges
//when completed, the yEdEdges[] field will be set to a trimmed-down
//version of this variable
yEdEdge[] temp_yEdEdges = new yEdEdge[1000];

//curTok is still == %%edges%%, so advance to the next token (which
//will be the start of the first yEdEdge)
curTok = advance();

//parse the input file to the end of the graph, instantiate yEdEdge
//objects and add them to the temp_yEdEdges array
while (!curTok.equals(END_OF_GRAPH)) {

    //set numEdgeLabels = 0 a the start of each iteration
    numEdgeLabels = 0;

    //the first token of an edge == the edge's source
    source = curTok;

    //the yEdEdge object's id field == source::target
    id = source + "::";

    //the next token will be the yEdEdge's target
    curTok = advance();

    //check to ensure if the current token represents both the target
    //and the end of the edge
    //if it does, then trim the trailing ";" off the target token
    if (curTok.endsWith(END_EDGE) && !curTok.equals(END_EDGE))
        target = curTok.substring(0, curTok.length() - 1);

    //otherwise, we have not reached the end of the yEdEdge yet
    else {

        //the current token represents the yEdEdge object's target
        target = curTok;

        curTok = advance();

        //the next token may represent the yEdEdge's type (with or
        //without an attached ";"), or it may be a stand-alone ";"
        if (curTok.endsWith(END_EDGE) && !curTok.equals(END_EDGE))
            type = curTok.substring(0, curTok.length() - 1);

        //if curTok != stand-alone ";" then it must represent the type
        //of the current yEdEdge object being built
        else if (!curTok.equals(END_EDGE)) {

            //set the current yEdEdge object's type
            type = curTok;

            curTok = advance();

            //the current token may either be a stand-alone ";" (in
            //which case it's the end of the yEdEdge) or the start of
            //a yEdEdge label
            if (!curTok.equals(END_EDGE)) {
```

```
            //set tempEdgeLabelCount to zero
            tempEdgeLabelCount = 0;

            //build the current yEdEdge object's labels
            findLabels("");

            //set numEdgeLabels to tempEdgeLabelCount (which is
            //calculated by the findlabels() method)
            numEdgeLabels = tempEdgeLabelCount;
        }
    }
}

//add target to the id String
id += target;

//move to the start of the next token before the start of the next
//iteration
curTok = advance();

//the following if-block checks to see if it is necessary to
//increase the size of the temp_yEdEdges array in order to avoid
//"indexOutOfBounds" errors
if (ctr % ARRAY_MULTIPLE == 0 && ctr > 0) {

    //create a new array that's equal to the size of the current
    //array, + the size of ARRAY_MULTIPLE
    yEdEdge[] tmp =
            new yEdEdge[temp_yEdEdges.length + ARRAY_MULTIPLE];

    //copy the contents of the old array to the new one
    System.arraycopy(temp_yEdEdges, 0, tmp, 0,
            temp_yEdEdges.length);

    //set the old, smaller array to the new, larger one
    temp_yEdEdges = tmp;
}

//instantiate a new yEdEdge with the parameters established
//above and add it to the temp_yEdEdges array
temp_yEdEdges[ctr] = new yEdEdge(id, EDGE_DATA, type, "", source,
        target, 0.0, "", "", numEdgeLabels);

//add labels to the yEdEdge object constructed in the previous
//command (if necessary)
if (numEdgeLabels > 0) {

    //instantiate a new String array of the appropriate size
    String[] tmp  = new String[numEdgeLabels];

    //copy the appropriate labels from tempEdgeLabels into the new
    //String array
    System.arraycopy(tempEdgeLabels, 0, tmp, 0, numEdgeLabels);

    //set the just-instantiated yEdEdge object's labels[] field to
    //the newly created/populated String array
    temp_yEdEdges[ctr].setLabels(tmp);

    //find X and Y coordinates for the label object, based on a
    //location that's midway between the yEdEdge's source yEdNode
    //and its target yEdNode
    double[] labelCoords =
```

```
                makeEdgeLabels(temp_yEdEdges[ctr].getEdgeSource(),
                temp_yEdEdges[ctr].getEdgeTarget());

            //set the yEdEdge object's label's X and Y coordinates
            temp_yEdEdges[ctr].setLabelX(labelCoords[0]);
            temp_yEdEdges[ctr].setLabelY(labelCoords[1]);
        }

        //increment the number of yEdEdges
        ctr++;

        //continue parsing the input document to the end of the graph
        if (curTok.equals(END_EDGE) && !curTok.equals(END_OF_GRAPH))
            advance();
    }

    //create a new array of an appropriate size
    edgeIDs = new String[ctr];

    //walk the array of temp_yEdEdges and fill the edgeIDs array
    for (int i = 0; i < ctr; i++)
        edgeIDs[i] = temp_yEdEdges[i].getEdgeID();

    //create a temporary array to hold the current arrangement of edgeIDs
    tempEdgeIDs = new String[ctr];

    //copy edgeIDs into the new array
    System.arraycopy(edgeIDs, 0, tempEdgeIDs, 0, ctr);

    //sort edgeIDs
    Arrays.sort(edgeIDs);

    //instantiate yEdEdges and make it an array of the appropriate size
    yEdEdges = new yEdEdge[ctr];

    //the following for-block fills the contents of yEdEdges such that
    //yEdEdges[] is sorted in ascending order base on each yEdEdge's id,
    //and also establishes a key-value pair mapping between edgeIDs and
    //yEdEdges
    for (int i = 0; i < ctr; i++) {

        //find the old position of tempEdgeID[i] in the new edgeIDs array -
        //this index will correspond to the appropriate index of
        //temp_yEdEdge[i] in the yEdEdges array
        index = Arrays.binarySearch(edgeIDs, tempEdgeIDs[i]);

        //set yEdEdge[index] to temp_yEdedges[i]
        yEdEdges[index] = temp_yEdEdges[i];
    }

    //for each object of yEdEdges, populate the remaining fields
    for (int i = 0; i < yEdEdges.length; i++) {

        //look up the appropriate edgeType of the current yEdEdge
        index = Arrays.binarySearch(edgeTypes, yEdEdges[i].getEdgeType());

        //the index value found in the previous command will correspond to
        //the appropriate edge-display parameters below
        yEdEdges[i].setEdgeColor(edgeColors[index]);
        yEdEdges[i].setEdgeWeight(edgeWeights[index]);
        yEdEdges[i].setEdgeDirected(edgesDirected[index]);
        yEdEdges[i].setEdgeLineStyle(edgeLineStyles[index]);
```

```
    }

    //find points through which to route each edge
    findPoints();

}

/**Parses the portions of the input file which come after an edge's type
 * definition and the end of the edge and compiles labels for the edge. The
 * method operates by recursively calling itself to build up each label one
 * token at a time.
 * @param labelSoFar
 * A String which represents the current label as compiled by previous
 * iterations of the method.
 */
private void findLabels(String labelSoFar) {

    //instantiate the return String and set it to the label which was
    //compiled by previous iterations of the method
    String tempLabel = labelSoFar;

    //it may be that the current token represents the end of the edge
    //if so, then take no action
    if (curTok.equals(END_EDGE)) {

    //if the current token ends with a ";", then take appropriate action
    } else if (curTok.endsWith(END_EDGE)) {

        //if the current token starts with a " and ends with a ";", then it
        //must be a one-word label
        if (curTok.startsWith("\"")) {

            //strip off the leading " and the trailing ";
            tempLabel += curTok.substring(1, curTok.length() - 2);

            //add the token to the array of temporary edge labels
            tempEdgeLabels[tempEdgeLabelCount] = tempLabel;

            //increment the number of labels for the current edge
            tempEdgeLabelCount++;

        //otherwise the current token represents the last word in a multi-
        //word label
        } else {

            //and a space between this token and the previous label and
            //strip off the trailing ";
            tempLabel += " " + curTok.substring(0, curTok.length() - 2);

            //add the token to the array of temporary edge labels
            tempEdgeLabels[tempEdgeLabelCount] = tempLabel;

            //increment the number of labels for the current edge
            tempEdgeLabelCount++;
        }

    //test to see if the current token is a single-word label
    } else if(curTok.matches("\".*\"")) {


        //strip off the leading and trailing quotes and add the token to
        //the array of temporary edge labels
```

```
    tempEdgeLabels[tempEdgeLabelCount] =
            curTok.substring(1, curTok.length() - 1);

    //increment the number of labels for the current edge
    tempEdgeLabelCount++;

    //advance to the next token
    curTok = advance();

    //since the current token is not a stand-alone ";", call
    //findLabels() again with an empty ""
    findLabels("");

//test to see if the current token is the start of a multi-word label
} else if(curTok.startsWith("\"")) {

    //strip off the leading quote and add the token to the array of
    //temporary edge labels
    tempLabel += curTok.substring(1, curTok.length());

    //advance to the next token
    curTok = advance();

    //since the current token is part of a multi-word label, call
    //findLabels() again and pass the label as it's been compiled so
    //far
    findLabels(tempLabel);

//test to see if the current token is the end of a multi-word label
} else if(curTok.endsWith("\"")) {

    //strip off the trailing quote and add the token to the temporary
    //label
    tempLabel += " " + curTok.substring(0, curTok.length() - 1);

    //since the current token ended with a quote, it indicates that
    //this label has been completed - so add it to the temporary arra
    //of labels
    tempEdgeLabels[tempEdgeLabelCount] = tempLabel;

    //increment the temporary count of labels for this edge
    tempEdgeLabelCount++;

    //advance to the next token
    curTok = advance();

    //call findLabels() again with an empty "" and look for more labels
    findLabels("");

//if all the tests above failed, then the current token is part of a
//multi-word label, but is neither the start of the label nor the end
//of a label
} else {

    //add a space between this token and the label as it's been
    //compiled so far
    tempLabel += " " + curTok;

    //advance to the next token
    curTok = advance();

    //since the current token is part of a multi-word label, call
```

```
            //findLabels() again and pass the label as it's been compiled so
            //far
            findLabels(tempLabel);
        }
    }

    /**Constructs X and Y coordinates for a yEdEdge label based on a midpoint
     * between the source and target yEdNode X-Y coordinates of the yEdEdge.
     * @param source
     * The id of the yEdNode associated with this yEdEdge's source.
     * @param target
     * The id of the yEdNode associated with this yEdEdge's target.
     * @return
     * An array of doubles of size two which represent X-Y coordinates of the
     * label.
     */
    private double[] makeEdgeLabels(String source, String target) {

        //a re-usable lookup variable that's used to find the index of yEdNodes
        //in the nodeIDs array
        int nodeIndex;

        //the value that will be returned as labelCoords[0]
        double labelX;

        //the value that will be returned as labelCoords[1]
        double labelY;

        //the X-coordinate of the yEdNode associated with this yEdEdge's source
        double sourceX;

        //the Y-coordinate of the yEdNode associated with this yEdEdge's source
        double sourceY;

        //the X-coordinate of the yEdNode associated with this yEdEdge's target
        double targetX;

        //the Y-coordinate of the yEdNode associated with this yEdEdge's target
        double targetY;

        //instantiates the return variable
        double[] labelCoords = new double[2];

        //find the index of the source nodeID
        nodeIndex = Arrays.binarySearch(nodeIDs, source);

        //find the X and Y-coordinates of the source yEdNode (the nodeIDs and
        //yEdNodes arrays are linked in a key-value pair map relationship)
        sourceX = yEdNodes[nodeIndex].getNodeX();
        sourceY = yEdNodes[nodeIndex].getNodeY();

        //find the index of the target nodeID
        nodeIndex = Arrays.binarySearch(nodeIDs, target);

        //find the X and Y-coordinates of the target yEdNode
        targetX = yEdNodes[nodeIndex].getNodeX();
        targetY = yEdNodes[nodeIndex].getNodeY();

        //find midpoints between the source and target X-Y values
        labelX = (Math.max(sourceX, targetX) - Math.min(sourceX, targetX)) / 2;
        labelY = (Math.max(sourceY, targetY) - Math.min(sourceY, targetY)) / 2;
```

```
    //ensure coordinates are positive values
    if (sourceX > targetX) labelX *= -1;
    if (sourceY > targetY) labelY *= -1;

    //assign the label coordinates
    labelCoords[0] = labelX;
    labelCoords[1] = labelY;

    return labelCoords;
}

/**Iterates through the collection of yEdEdges and determines whether or
 * not it is necessary to make points for each. All non-directed edges and
 * most directed edges will not need points. However, for pairs of directed
 * edges which make a cycle, it is necessary to separate the edges.
 * Otherwise the yEd graphing application will draw one edge directly on
 * top of the other making it appear that there is just one bi-directional
 * edge rather than two directed edges.
 */
private void findPoints() {

    //re-usable place holder to find the index of a yEdEdge's target in the
    //nodeIDs array
    int target;

    //re-usable String which will represent the target::source id to be
    //searched for
    String searchString;

    //walk the array of yEdEdges and test to see if the target of one
    //particular yEdEdge is yEdEdge[i]'s target as well (i.e., they are
    //joined in a cycle)
    for (int i = 0; i < yEdEdges.length; i++) {

        //re-set target
        target = 0;

        //build the search String: yEdEdge's id is "source::target", so
        //test to see if some other yEdEdge has the id "target::source"
        searchString = yEdEdges[i].getEdgeTarget() + "::"
                + yEdEdges[i].getEdgeSource();

        //use the binarySearch tool provided by java.util.Arrays
        target = Arrays.binarySearch(edgeIDs, searchString);

        //if target is less than zero then no match was found
        if (target >= 0) {

            //see if points have already been found for the edge going from
            //the target back to the source
            if (yEdEdges[target].hasPoints() == false)

                //draw points above the source and target yEdNodes
                makeEdgePoints(yEdEdges[i].getEdgeSource(),
                        yEdEdges[i].getEdgeTarget(), true, i);

            else

                //if yEdEdge[target] has points, then points above the
                //yEdNode have already been drawn, so draw this set of
                //points below
                makeEdgePoints(yEdEdges[i].getEdgeSource(),
```

```
                        yEdEdges[i].getEdgeTarget(), false, i);
            }
        }
    }

    /**Creates X-Y coordinate values which will be used as values for x="" and
     * y="" attributes of &lt;y:Point> elements (used to route edges around
     * nodes in the yEd graphing application). All non-directed edges and most
     * directed edges will not need points. However, for pairs of directed
     * edges which make a cycle, it is necessary to separate the edges.
     * Otherwise the yEd graphing application will draw one edge directly on
     * top of the other making it appear that there is just one bi-directional
     * edge rather than two directed edges.
     * @param source
     * The value of the yEdEdge's source field. Will be used in a search of the
     * yEdNodes array.
     * @param target
     * The value of the yEdEdge's target field. Will be used in a search of the
     * yEdNodes array.
     * @param above
     * Indicates whether points will be created above or below the source and
     * target yEdNodes.
     * @param edgeIndex
     * The yEdEdge's array index of the yEdEdge for which this set of points is
     * being constructed.
     */
    private void makeEdgePoints(String source, String target,
            boolean above, int edgeIndex) {

        //the uniform distance, either above or below the source and target
        //yEdNodes, that the points will be drawn
        double distance = 25.0;

        //represents two pairs of X-Y coordinates
        double points[] = new double[4];

        //find the ID's of the source and target yEdNodes (the nodeIDs and
        //yEdNodes arrays are linked in a key-value pair map relationship)
        int sourceIndex = Arrays.binarySearch(nodeIDs, source);
        int targetIndex = Arrays.binarySearch(nodeIDs, target);

        //retrieve the X and Y-coordinates of the source and target yEdNodes
        double sourceX = yEdNodes[sourceIndex].getNodeX();
        double sourceY = yEdNodes[sourceIndex].getNodeY();
        double targetX = yEdNodes[targetIndex].getNodeX();
        double targetY = yEdNodes[targetIndex].getNodeY();

        //if the Y-coordinates of the source and target yEdNodes are the same,
        //then draw the points with Y-coordinates ABOVE the yEdNodes but with
        //the same X-coordinates as the yEdNodes
        if (yEdNodes[sourceIndex].getNodeY() ==
                yEdNodes[targetIndex].getNodeY() && above) {

            //establish X-coordinates that are at the midpoint of the source
            //yEdNode's width
            points[0] = sourceX + nodeWidth / 2;

            //establish Y-coordinates that are a uniform distance above the
            //source yEdNode
            points[1] = sourceY - distance;

            //same for the target yEdNode
```

```
    points[2] = targetX + nodeWidth / 2;
    points[3] = targetY - distance;

//if the Y-coordinates of the source and target yEdNodes are the same,
//then draw the points with Y-coordinates BELOW the yEdNodes but with
//the same X-coordinates as the yEdNodes
} else if (yEdNodes[sourceIndex].getNodeY() ==
        yEdNodes[targetIndex].getNodeY() && !above) {

    //establish X-coordinates that are at the midpoint of the source
    //yEdNode's width
    points[0] = sourceX + nodeWidth / 2;

    //establish Y-coordinates that are a uniform distance below the
    //source yEdNode
    points[1] = sourceY + NODE_HEIGHT + distance;

    //same for the target yEdNode (account for the height of the node)
    points[2] = targetX + nodeWidth / 2;
    points[3] = targetY + NODE_HEIGHT + distance;

//if the X-coordinates of the source and target yEdNodes are the same,
//then draw the points with X-coordinates to the LEFT of the yEdNodes
//but with the same Y-coordinates as the yEdNodes
} else if (yEdNodes[sourceIndex].getNodeX() ==
        yEdNodes[targetIndex].getNodeX() && above) {

    //establish X-coordinates that are a uniform distance to the left
    //of the source yEdNode
    points[0] = sourceX - distance;

    //establish Y-coordinates that are at the midpoint of the source
    //yEdNode's height
    points[1] = sourceY + NODE_HEIGHT / 2;

    //same for the target yEdNode
    points[2] = targetX - distance;
    points[3] = targetY + NODE_HEIGHT / 2;

//if the X-coordinates of the source and target yEdNodes are the same,
//then draw the points with X-coordinates to the RIGHT of the yEdNodes
//but with the same Y-coordinates as the yEdNodes
} else if (yEdNodes[sourceIndex].getNodeX() ==
        yEdNodes[targetIndex].getNodeX() && !above) {

    //establish X-coordinates that are a uniform distance to the right
    //of the source yEdNode (account for the width of the node)
    points[0] = sourceX + nodeWidth + distance;

    //establish Y-coordinates that are at the midpoint of the source
    //yEdNode's height
    points[1] = sourceY + NODE_HEIGHT / 2;

    //same for the target yEdNode
    points[2] = targetX + nodeWidth + distance;
    points[3] = targetY + NODE_HEIGHT / 2;

//otherwise draw the points ABOVE and to the LEFT of the source and
//target yEdNodes
} else if (above) {

    //establish X-coordinates that are a uniform distance to the left
```

```
            //of the source yEdNode
            points[0] = sourceX + nodeWidth / 2;

            //establish Y-coordinates that are a uniform distance above the
            //source yEdNode
            points[1] = sourceY - distance;

            //same for the target yEdNode
            points[2] = targetX + nodeWidth / 2;
            points[3] = targetY - distance;

        //otherwise draw the points BELOW and to the RIGHT of the source and
        //target yEdNodes
        } else if (!above) {

            //establish X-coordinates that are a uniform distance to the right
            //of the source yEdNode (account for the width of the node)
            points[0] = sourceX + nodeWidth / 2;

            //establish Y-coordinates that are a uniform distance below the
            //source yEdNode (account for the height of the node)
            points[1] = sourceY + NODE_HEIGHT + distance;

            //same for the target yEdNode
            points[2] = targetX + nodeWidth / 2;
            points[3] = targetY + NODE_HEIGHT + distance;
        }

        yEdEdges[edgeIndex].setPoints(points);
    }
}
```

D. THE GRAPHML_COMPILER CLASS

```
package MPGrapher;

import java.io.File;
import java.util.Scanner;
import java.io.IOException;

/**Compiles graphML XML files which conform to the GraphML 1.0 schema as
 * defined by
 * <a href="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
 * http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd.</a>
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 23, 2012
 */
public class GraphML_Compiler {

////////////////////////////////////////////////////////////////////////////
////variable class fields///////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which tracks the number of nodes/edges to be graphed*/
    private int ctr = 0;

    /**Field which contains the current token of the input file as returned by
     * the scanner.<p />
     * curTok is an abbreviation and concatenation of the words "current" and
     * "token".
     */
```

```
    private String curTok;

    /**Scanner, provided by the java.util.Scanner library, which scans the
     * input file for tokens.
     */
    private static Scanner scan;

////////////////////////////////////////////////////////////////////////////
////class constants/////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Constant which provides standards increments for array sizes: all arrays
     * within the application are assigned the same size, and grow by the same
     * amount when necessary.
     */
    private static final int ARRAY_MULTIPLE = 1000;

    /**GraphML grammar constant which represents a delimiter between the end of
     * one edge and the start of another edge, or the end of the graph. Used
     * for parsing input documents.
     */
    private static final String END_EDGE = ";";

    /**GraphML grammar constant which represents the end of the nodes section
     * and the beginning of the edges section. Used for parsing input
     * documents.
     */
    private static String EDGES = "%%edges%%";

    /**GraphML grammar constant which represents the beginning of the nodes
     * section. Used for parsing input documents. All tokens before this are
     * ignored.
     */
    private static String NODES = "%%nodes%%";

    /**GraphML grammar constant which represents the end of a graph of the
     * input file. All tokens after this are ignored.
     */
    private static final String END_OF_GRAPH = "%%eog%%";

////////////////////////////////////////////////////////////////////////////
////graphML constants///////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**GraphML XML element: defines the start of a well-formed graphML
     * document.
     */
    private static final String GRAPHML_REFS
        = "<?xml version=\"1.0\"?>\n"
        + "<graphml xmlns=\"http://graphml.graphdrawing.org/xmlns\"\n"
        + "\txmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\n";

    /**GraphML XML element: defines the graphML schema attribute of a
     * &lt;graphml> element (child-classes may include additional schema
     * references here).
     */
    private static final String GRAPHML_SCHEMA
        = "\txsi:schemaLocation=\"http://graphml.graphdrawing.org/xmlns\n"
        + "\thttp://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd";

    /**GraphML XML element: closes the opening graphML element of a well-formed
     * graphML document.
```

```
     */
    private static final String GRAPHML_SCHEMA_CLOSE = "\">\n";

    /**GraphML XML element: defines the opening element of a graph of a
     * well-formed graphML document.
     */
    private static final String GRAPHML_GRAPH_OPEN
        = "\t<graph edgedefault=\"directed\" id=\"G\">\n";

    /**GraphML XML element: defines the closing element of a graph of a
     * well-formed graphML document.
     */
    private static final String GRAPHML_GRAPH_CLOSE = "\t</graph>\n";

    /**GraphML XML element: defines the closing element of a well-formed
     * graphML document.
     */
    private static final String GRAPHML_FOOTER = "</graphml>\n";

////////////////////////////////////////////////////////////////////////////
////getter methods for graphML constants//////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of ARRAY_MULTIPLE to child
     * classes.
     * @return Value of private ARRAY_MULTIPLE constant.
     */
    protected static final int GET_ARRAY_MULTIPLE() {return ARRAY_MULTIPLE;}

    /**Getter method which returns the value of END_EDGE_TOKEN to child
     * classes.
     * @return Value of private END_EDGE_TOKEN constant.
     */
    protected static final String GET_END_EDGE_TOKEN() {return END_EDGE;}

    /**Getter method which returns the value of EDGES to child classes.
     * @return Value of private EDGES constant.
     */
    protected static final String GET_EDGES_TOKEN() {return EDGES;}

    /**Getter method which returns the value of NODES to child classes.
     * @return Value of private NODES constant.
     */
    protected static final String GET_NODES_TOKEN() {return NODES;}

    /**Getter method which returns the value of END_OF_GRAPH to child classes.
     * @return Value of private END_OF_GRAPH constant.
     */
    protected static final String GET_END_OF_GRAPH_TOKEN() {
        return END_OF_GRAPH;}

    /**Getter method which returns the value of GRAPHML_REFS to child classes.
     * @return Value of private GRAPHML_REFS constant.
     */
    protected static final String GET_GRAPHML_REFS() {return GRAPHML_REFS;}

    /**Getter method which returns the value of GRAPHML_SCHEMA to child
     * classes.
     * @return Value of private GRAPHML_SCHEMA constant.
     */
    protected static final String GET_GRAPHML_SCHEMA() {return GRAPHML_SCHEMA;}
```

```
    /**Getter method which returns the value of GRAPHML_SCHEMA_CLOSE to child
     * classes.
     * @return Value of private GRAPHML_SCHEMA_CLOSE constant.
     */
    protected static final String GET_GRAPHML_SCHEMA_CLOSE() {
        return GRAPHML_SCHEMA_CLOSE;}

    /**Getter method which returns the value of GRAPHML_GRAPH_OPEN to child
     * classes.
     * @return Value of private GRAPHML_GRAPH_OPEN constant.
     */
    protected static final String GET_GRAPHML_GRAPH_OPEN() {
        return GRAPHML_GRAPH_OPEN;}

    /**Getter method which returns the value of GRAPHML_GRAPH_CLOSE to child
     * classes.
     * @return Value of private GRAPHML_GRAPH_CLOSE constant.
     */
    protected static final String GET_GRAPHML_GRAPH_CLOSE() {
        return GRAPHML_GRAPH_CLOSE;}

    /**Getter method which returns the value of GRAPHML_FOOTER to child
     * classes.
     * @return Value of private GRAPHML_FOOTER constant.
     */
    protected static final String GET_GRAPHML_FOOTER() {return GRAPHML_FOOTER;}

////////////////////////////////////////////////////////////////////////////
////constructors////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
     * @throws IOException
     * Required by the java.io.File library, which is in  turn required by the
     * java.util.Scanner library.
     */
    public GraphML_Compiler() throws IOException {

        ctr = 0;
        curTok = "";
    }

    /**Constructor which opens the input file for parsing. Currently the only
     * constructor used by the application.
     * @param fileToParse
     * String which reflects the "path to" and the "name of" the input file
     * which the application is to parse.
     * @throws IOException
     * Required by the java.io.File library, which is in turn required by the
     * java.util.Scanner library.
     */
    public GraphML_Compiler(File fileToParse) throws IOException {

        ctr = 0;
        curTok = "";

        //opens fileToParse for the Scanner to parse
        openFile(fileToParse);
    }
```

```
////////////////////////////////////////////////////////////////////////////
////public methods//////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed graphML XML document.
     * @return
     * A well-formed graphML XML document.
     */
    public String getGraph() {

        //instantiates a String object and assigns the opening elements
        String xmlOut = GRAPHML_REFS + GRAPHML_SCHEMA + GRAPHML_SCHEMA_CLOSE
                + GRAPHML_GRAPH_OPEN;

        //compiles and adds additional elements to the document
        xmlOut += statements();

        //add the closing elements of a well-formed graphML XML document
        xmlOut += GRAPHML_GRAPH_CLOSE + GRAPHML_FOOTER;

        return xmlOut;
    }

    /**Compiles and returns a well-formed graphML XML document with &lt;key>
     * elements.
     * <p />&lt;key> elements are defined in the graphML XML schema.
     * @return
     * A well-formed graphML XML document with &lt;key> elements.
     */
    public String getGraph(String keyArgs) {

        //instantiates a String object and assigns the opening elements
        String xmlOut = GRAPHML_REFS + GRAPHML_SCHEMA + GRAPHML_SCHEMA_CLOSE;

        //adds <key> graphML elements to the document
        if (keyArgs != null) xmlOut += keyArgs;

        //adds the opening element for a graphML graph to the document
        xmlOut += GRAPHML_GRAPH_OPEN;

        //compiles and adds additional elements to the document
        xmlOut += statements();

        //add the closing elements of a well-formed graphML XML document
        xmlOut += GRAPHML_GRAPH_CLOSE + GRAPHML_FOOTER;

        return xmlOut;
    }

////////////////////////////////////////////////////////////////////////////
////private and protected methods///////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Instantiates a new Scanner object and opens the input file for parsing.
     * The Scanner class is provided by the java.util suite of libraries.
     * @param fileToParse
     * String which reflects the "path to" and the "name of" the input file
     * which the application is to parse.
     * @throws IOException
     * Required by the java.util.Scanner library.
     */
    protected void openFile(File fileToOpen) throws IOException {
```

```
    scan = new Scanner(fileToOpen);
}

/**Scans the input document and compiles and returns well-formed graphML
 * XML &lt;node> and &lt;edge> elements for output.
 * @return
 * Well-formed &lt;node> and &lt;edge> graphML XML elements
 */
private String statements() {

    //instantiates the String to be returned to the calling method
    String xmlOut;

    //scans the input document for node information
    //compiles &lt;node> elements and adds them to the output String
    xmlOut = nodes();

    //scans the input document for edge information
    //compiles &lt;edge> elements and adds them to the output String
    xmlOut += edges();

    return xmlOut;
}

/**Scans the input document and compiles and returns well-formed graphML
 * XML &lt;node> elements for output.
 * @return
 * Well-formed &lt;node> graphML XML elements
 */
protected String nodes() {

    //instantiates the String to be returned to the calling method
    String xmlOut = "";

    //an array of String objects which will track the id="" attributes of
    //the Node objects which are to be created. nodesIDs serve as the
    //values of id="" attributes within graphML <node> elements.
    String[] nodesIDs = new String[ARRAY_MULTIPLE];

    //parse the input file to the first token of the nodes section of the
    //input file
    while (!curTok.equals(NODES)) curTok = advance();
    curTok = advance();

    //parse the nodes section of the input file and stop at the
    //edges section
    nodesIDs = buildNodesArray(EDGES);

    //compile the String object representing a well-formed graphML XML
    //&lt;node> element to return to the calling method
    for (int i = 0; i < ctr; i++) {

        xmlOut += node(nodesIDs, i);
    }

    return xmlOut;
}

/**Scans the input document and returns an array of String objects which
 * will be used to define the id="" attributes of &lt;node> graphML XML
 * elements.
```

```
 * @param endToken
 * Represents the token which signals the end of the nodes section of the
 * input file and the beginning of the edges section.
 * @return
 * An array of node IDs. This array of String objects values for the id
 * fields of Node objects. nodesIDs serve as the values of id="" attributes
 * of graphML &lt;node> elements.
 */
private String[] buildNodesArray(String endToken) {

    //a new array of String objects to hold the node IDs.
    String[] tempNodeIDs = new String[ARRAY_MULTIPLE];

    while (!curTok.equals(endToken)) {

        //test the current size of the array
        if (ctr % ARRAY_MULTIPLE == 0) {

            //if the array is too large, grow it
            //avoids triggering the "indexOutOfBounds" error
            if (ctr > 0) {

                //create a new, larger array
                String[] tmp = new String[tempNodeIDs.length
                        + ARRAY_MULTIPLE];

                //copy the contents of the old array to the new array
                System.arraycopy(tempNodeIDs, 0, tmp, 0, ctr);

                //set the old array's pointer to the new array
                //Java's garbage collector will take care of destroying the
                //old array
                tempNodeIDs = tmp;
            }
        }

        //add the current token to the array
        tempNodeIDs[ctr] = curTok;

        //get the next token from the input file
        curTok = advance();

        //increment the counter
        ctr++;
    }

    return tempNodeIDs;
}

/**Instantiates new Node objects, sets Node object fields, and adds
 * their properties to the XML output String.
 * @param nodesIDs
 * An array of String objects which is used to set the id="" attribute of
 * &lt;node> elements.
 * @param index
 * An integer which is used to find the appropriate nodeId within the
 * nodeIDs array.
 * @return
 * A well-formed &lt;node> graphML XML element.
 */
private String node (String[] nodesIDs, int index) {
```

```
    //instantiates the String object which will be returned to the calling
    //method
    String xmlOut = "";

    //instantiates a new Node object and sets the Node's "name" and
    //"data" fields
    Node newNode = new Node(nodesIDs[index], "");

    //fetches the &lt;node> graphML XML elements from the Node object
    xmlOut = newNode.getNodeXML();

    return xmlOut;
}

/**Scans the input document and compiles and returns well-formed graphML
 * XML &lt;edge> elements for output.
 * @return
 * Well-formed &lt;edge> graphML XML elements
 */
protected String edges() {

    //re-uses the ctr field and sets it back to zero
    //used to track the number of edges defined in the input/output files
    ctr = 0;

    //instantiates the String to be returned to the calling method
    String xmlOut = "";

    //used to determine if the current token represents the end of one
    //particular edge definition
    boolean endEdge = false;

    //instantiates an array of Strings which are to be used as the
    //source="" attributes of &lt;edge> XML elements
    String[] sources = new String[ARRAY_MULTIPLE];

    //instantiates an array of Strings which are to be used as the
    //target="" attributes of &lt;edge> XML elements
    String[] targets = new String[ARRAY_MULTIPLE];

    //curTok currently == %%edges%%, so begin parsing with the next token
    curTok = advance();

    //parse and compile until the end of the graph is reached.
    while (!curTok.equals(END_OF_GRAPH)) {

        //test the current size of the array
        if (ctr % ARRAY_MULTIPLE == 0) {

            //if the array is too large, grow it
            //avoids triggering the "indexOutOfBounds" error
            if (ctr > 0) {

                //create a new, larger array
                String[] temp1 = new String[sources.length
                        + ARRAY_MULTIPLE];

                //copy the contents of the old array to the new array
                System.arraycopy(sources, 0, temp1, 0, ctr);

                //set the old array's pointer to the new array
                //Java's garbage collector will take care of destroying the
```

```
                //old array
                sources = temp1;

                //create a new, larger array
                String[] temp2 = new String[sources.length
                        + ARRAY_MULTIPLE];

                //copy the contents of the old array to the new array
                System.arraycopy(targets, 0, temp2, 0, ctr);

                //set the old array's pointer to the new array
                //Java's garbage collector will take care of destroying the
                //old array
                targets = temp2;
            }
        }

        //reset endEdge to false
        endEdge = false;

        //set the current token as the edge's source
        sources[ctr] = curTok;
        curTok = advance();

        //test for the end of an edge
        if (curTok.endsWith(END_EDGE)) {

            endEdge = true;

            //strips the ";" off the end of the current token
            curTok = curTok.substring(0, curTok.length() - 1);
        }

        //set the current token as the edge's target
        targets[ctr] = curTok;

        //if the current token ended with ";", the next token will be a
        //stand-alone ";"
        if (!endEdge) curTok = advance();

        //advance to the next token of the input file, which will either be
        //the start of a new edge or the end of the graph
        curTok = advance();

        //increment the counter
        ctr++;
    }

    //compile the String object representing a well-formed graphML XML
    //&lt;edge> element to return to the calling method
    for (int i = 0; i < ctr; i++) {

        xmlOut += edge(sources, targets, i);
    }

    return xmlOut;
}

/**Instantiates new Edge objects, sets Edge object fields, and adds
 * their properties to the XML output String.
 * @param sources
 * An array of String objects which is used to set the source="" attribute
```

```
     * of &lt;edge> elements.
     * @param targets
     * An array of String objects which is used to set the target="" attribute
     * of &lt;edge> elements.
     * @param index
     * An integer which is used to find the appropriate source and target IDs
     * within the source and target arrays.
     * @return
     * A well-formed &lt;edge> graphML XML element.
     */
    private String edge(String[] sources, String[] targets, int index) {

        //instantiates the String object which will be returned to the calling
        //method
        String xmlOut = "";

        //instantiates a new Edge object and sets the Edge's "source" and
        //"target" fields
        Edge newEdge = new Edge(sources[index], targets[index], "");

        //fetches the &lt;node> graphML XML elements from the Node object
        xmlOut = newEdge.getEdgeXML();

        return xmlOut;
    }

    /**Scans and returns the next token of the input document.
     * @return
     * The input document's next token.
     */
    protected final String advance() {

        //curTok is a private field of the GraphML_Compiler object and is
        //instantiated upon construction
        //fetch the next token from the input file and assign it to the curTok
        //variable
        curTok = scan.next();

        //return the current token to the calling method
        return curTok;
    }
}
```

E. THE YEDNODE CLASS

```
package MPGrapher;

/**Constructs yEdNode objects and compiles and returns well-formed,
 * yEd-specific &lt;node> graphML XML elements.
 * @author Capt Timothy L. Shields USMC
 * @since 1.0 last modified July 28, 2012
 */
public class yEdNode extends Node {

////////////////////////////////////////////////////////////////////////////
////variable class fields/////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which represents the ID of a yEdNode object.*/
    private String id;
```

```
    /**Field used to compile well-formed &lt;data> graphML XML elements (child
     * classes may include additional &lt;data> elements here).
     */
    private String data;

    /**Field that holds a String representation of a yEdNode object's label.*/
    private String label;

    /**Field that holds the position of a yEdNode's X-coordinate.*/
    private double nodeX;

    /**Field that holds the position of a yEdNode's Y-coordinate.*/
    private double nodeY;

    /**Field that holds a String representation of a yEdNode object's shape.*/
    private String shape;

    /**Field that holds the width of a yEdNode object.*/
    private double width;

    /**Field that holds the width of a yEdNode object.*/
    private double height;

    /**Field that holds a String representation of a yEdNode object's fill
     * color.
     */
    private String fillColor;

////////////////////////////////////////////////////////////////////////////
////yEd-specific graphML <node> element constants////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**yEd-specific graphML XML element: defines the start of a well-formed,
     * yEd-specific graphML &lt;node> element, up to the height="" attribute of
     * a &lt;y:Geometry> element.
     */
    private static final String YED_NODE_HEIGHT = "\t\t\t\t<y:ShapeNode>\n"
            + "\t\t\t\t\t<y:Geometry height=\"";

    /**yEd-specific graphML XML element: defines the width="" attribute of
     * a well-formed, yEd-specific &lt;y:Geometry> element.
     */
    private static final String YED_NODE_WIDTH = "\" width=\"";

    /**yEd-specific graphML XML element: defines the x="" attribute of
     * a well-formed, yEd-specific &lt;y:Geometry> element.
     */
    private static final String YED_NODE_X_POS = "\" x=\"";

    /**yEd-specific graphML XML element: defines the y="" attribute of
     * a well-formed, yEd-specific &lt;y:Geometry> element.
     */
    private static final String YED_NODE_Y_POS = "\" y=\"";

    /**yEd-specific graphML XML element: closes a &lt;y:Geometry> element and
     * defines the start of a well-formed, yEd-specific graphML &lt;y:Fill>
     * element, up to the color="" attribute.
     */
    private static final String YED_NODE_FILL_COLOR = "\"/>\n"
            + "\t\t\t\t\t<y:Fill color=\"";

    /**yEd-specific graphML XML element: closes a &lt;y:Fill> element, adds a
```

```
     * &lt;y:BorderStyle> element, and defines a well-formed, yEd-specific
     * graphML &lt;y:NodeLabel> element.
     */
    private static final String YED_NODE_LABEL = "\" transparent=\"false\"/>\n"
            + "\t\t\t\t\t<y:BorderStyle color=\"#000000\" type=\"line\" "
                + "width=\"1.0\"/>\n"
            + "\t\t\t\t\t<y:NodeLabel alignment=\"center\" "
                + "autoSizePolicy=\"content\" fontFamily=\"Courier New\" "
                + "fontSize=\"12\" fontStyle=\"plain\" "
                + "hasBackgroundColor=\"false\" hasLineColor=\"false\" "
                + "height=\"44.78125\" modelName=\"custom\" "
                + "textColor=\"#000000\" visible=\"true\" width=\"61.609375\" "
                + "x=\"9.1953125\" y=\"-7.390625\">\n\t\t\t\t\t\t\t";

    /**yEd-specific graphML XML element: defines additional elements which are
     * necessary for the proper display of &lt;node> elements in the yEd
     * graphing application, and begins the start of a well-formed,
     * yEd-specific &lt;y:Shape> element, up to the type="" attribute.
     */
    private static final String YED_NODE_SHAPE =
            "\n\t\t\t\t\t\t<y:LabelModel>\n"
            + "\t\t\t\t\t\t\t<y:SmartNodeLabelModel distance=\"4.0\"/>\n"
            + "\t\t\t\t\t\t</y:LabelModel>\n"
            + "\t\t\t\t\t\t<y:ModelParameter>\n"
            + "\t\t\t\t\t\t\t<y:SmartNodeLabelModelParameter "
                + "labelRatioX=\"0.0\" labelRatioY=\"0.0\" nodeRatioX=\"0.0\" "
                + "nodeRatioY=\"0.0\" offsetX=\"0.0\" offsetY=\"0.0\" "
                + "upX=\"0.0\" upY=\"-1.0\"/>\n"
            + "\t\t\t\t\t\t</y:ModelParameter>\n"
            + "\t\t\t\t\t</y:NodeLabel>\n"
            + "\t\t\t\t\t<y:Shape type=\"";

    /**yEd-specific graphML XML element: defines the closing element of a well-
     * formed, yEd-specific graphML &lt;node> element.
     */
    private static final String YED_NODE_TAG_CLOSE = "\"/>\n"
            + "\t\t\t\t</y:ShapeNode>\n";

////////////////////////////////////////////////////////////////////////////
////getter and setter methods for private yEdNode fields/////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of nodeX to child classes.
     * @return Value of private nodeX variable.
     */
    protected double getNodeX() {return nodeX;}

    /**Getter method which returns the value of nodeY to child classes.
     * @return Value of private nodeY variable.
     */
    protected double getNodeY() {return nodeY;}

    /**Setter method which allows child classes to set the value of nodeX.*/
    protected void setNodeX(double xCoord) {nodeX = xCoord;}

    /**Setter method which allows child classes to set the value of nodeY.*/
    protected void setNodeY(double yCoord) {nodeY = yCoord;}

    /**Setter method which allows child classes to set the value of
     * nodeWidth.
     */
    protected void setNodeWidth(double nodeWidth) {width = nodeWidth;}
```

```
////////////////////////////////////////////////////////////////////////////
////getter methods for yEd-specific graphML node element constants/////////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of GET_YED_NODE_HEIGHT to child
     * classes.
     * @return Value of private GET_YED_NODE_HEIGHT constant.
     */
    private static final String GET_YED_NODE_HEIGHT() {return YED_NODE_HEIGHT;}

    /**Getter method which returns the value of GET_YED_NODE_WIDTH to child
     * classes.
     * @return Value of private GET_YED_NODE_WIDTH constant.
     */
    private static final String GET_YED_NODE_WIDTH() {return YED_NODE_WIDTH;}

    /**Getter method which returns the value of GET_YED_NODE_X_POS to child
     * classes.
     * @return Value of private GET_YED_NODE_X_POS constant.
     */
    private static final String GET_YED_NODE_X_POS() {return YED_NODE_X_POS;}

    /**Getter method which returns the value of GET_YED_NODE_Y_POS to child
     * classes.
     * @return Value of private GET_YED_NODE_Y_POS constant.
     */
    private static final String GET_YED_NODE_Y_POS() {return YED_NODE_Y_POS;}

    /**Getter method which returns the value of GET_YED_NODE_FILL_COLOR to
     * child classes.
     * @return Value of private GET_YED_NODE_FILL_COLOR constant.
     */
    private static final String GET_YED_NODE_FILL_COLOR() {
        return YED_NODE_FILL_COLOR;}

    /**Getter method which returns the value of GET_YED_NODE_LABEL to child
     * classes.
     * @return Value of private GET_YED_NODE_LABEL constant.
     */
    private static final String GET_YED_NODE_LABEL() {return YED_NODE_LABEL;}

    /**Getter method which returns the value of GET_YED_NODE_SHAPE to child
     * classes.
     * @return Value of private GET_YED_NODE_SHAPE constant.
     */
    private static final String GET_YED_NODE_SHAPE() {return YED_NODE_SHAPE;}

    /**Getter method which returns the value of GET_YED_NODE_TAG_CLOSE to child
     * classes.
     * @return Value of private GET_YED_NODE_TAG_CLOSE constant.
     */
    private static final String GET_YED_NODE_TAG_CLOSE() {
        return YED_NODE_TAG_CLOSE;}

////////////////////////////////////////////////////////////////////////////
////constructors////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
```

```
     */
    public yEdNode() {}

    /**Instantiates a yEdNode object and assigns values to fields.
     * @param nodeID
     * Assigned to a yEdNode object's id field. Used as the value of an id=""
     * attribute of a &lt;node> graphML XML element. <p />
     * @param nodeData
     * Assigned to a yEdNode object's data field. Can be used to populate one
     * or more attributes of a &lt;data> graphML XML element, or
     * even additional &lt;data> elements. <p />
     * @param nodeHeight
     * Assigned to a yEdNode object's height field. Used to ensure a uniform
     * height of output nodes. <p />
     * @param nodeWidth
     * Assigned to a yEdNode object's width field. Used to ensure a uniform
     * width of output nodes. <p />
     * @param xParam
     * Assigned to a yEdNode object's nodeX field. Used placing output nodes in
     * a matrix pattern in the yEd graphing application. <p />
     * @param yParam
     * Assigned to a yEdNode object's nodeY field. Used placing output nodes in
     * a matrix pattern in the yEd graphing application. <p />
     * @param color
     * Assigned to a yEdNode object's fillColor field. Used to color output
     * nodes in the yEd graphing application. <p />
     * @param nodeLabel
     * Assigned to a yEdNode object's label field. Used to label output nodes
     * in the yEd graphing application. <p />
     * @param shapeType
     * Assigned to a yEdNode object's shape field. Used to control the shape of
     * output nodes in the yEd graphing application.
     */
    public yEdNode(String nodeID, String nodeData, double nodeHeight,
            double nodeWidth, double xParam, double yParam, String color,
            String nodeLabel, String shapeType) {

        //all field values are taken directly from constructor's parameters
        id = nodeID;
        data = nodeData;
        height = nodeHeight;
        width = nodeWidth;
        nodeX = xParam;
        nodeY = yParam;
        fillColor = color;
        label = nodeLabel;
        shape = shapeType;
    }

////////////////////////////////////////////////////////////////////////////
////public method///////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed, yEd-specific &lt;node> graphML XML
     * element.
     * @return
     * A well-formed, yEd-specific &lt;node> graphML XML element.
     */
    public String get_yEdNodeXML() {

        //instantiate the return String and set it to a well-formed,
        //yEd-specific node> element with attribute values specific to this
```

```
        //yEdNode
        String xmlOut = GET_NODE_XML_OPEN_TAG() + id
                + GET_NODE_XML_OPEN_TAG_END()
                + GET_NODE_XML_DATA_OPEN_TAG() + data
                + GET_NODE_XML_DATA_OPEN_TAG_END()
                + GET_YED_NODE_HEIGHT() + height
                + GET_YED_NODE_WIDTH() + width
                + GET_YED_NODE_X_POS() + nodeX
                + GET_YED_NODE_Y_POS() + nodeY
                + GET_YED_NODE_FILL_COLOR() + fillColor
                + GET_YED_NODE_LABEL() + label
                + GET_YED_NODE_SHAPE() + shape
                + GET_YED_NODE_TAG_CLOSE() + GET_NODE_XML_DATA_CLOSE_TAG()
                + GET_NODE_XML_CLOSE_TAG();

        return xmlOut;
    }
}
```

F. THE NODE CLASS

```
package MPGrapher;

/**Constructs Node objects and compiles and returns well-formed &lt;node>
 * graphML XML elements.
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 23, 2012
 */
public class Node {

////////////////////////////////////////////////////////////////////////////
////variable class fields/////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which represents the ID of a Node object.*/
    private String id;

    /**Field used to compile well-formed &lt;data> graphML XML elements (child
     * classes may include additional &lt;data> elements here).
     */
    private String data;

////////////////////////////////////////////////////////////////////////////
////graphML node element constants////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**GraphML XML element: defines the start of a well-formed graphML
     * &lt;node> element.
     */
    private static final String NODE_XML_OPEN_TAG = "\t\t<node id=\"";

    /**GraphML XML element: defines the end of a well-formed graphML &lt;node>
     * element.
     */
    private static final String NODE_XML_OPEN_TAG_END = "\">\n";

    /**GraphML XML element: defines the start of a well-formed graphML
     * &lt;data> element associated with a &lt;node> element.
     */
    private static final String NODE_XML_DATA_OPEN_TAG = "\t\t\t<data key=\"";
```

```
    /**GraphML XML element: defines the end of a well-formed graphML &lt;data>
     * element associated with a &lt;node> element.
     */
    private static final String NODE_XML_DATA_OPEN_TAG_END = "\">\n";

    /**GraphML XML element: defines the closing element of a well-formed
     * graphML &lt;data> element associated with a &lt;node> element.
     */
    private static final String NODE_XML_DATA_CLOSE_TAG = "\t\t\t</data>\n";

    /**GraphML XML element: defines the closing element of a well-formed
     * graphML &lt;node> element.
     */
    private static final String NODE_XML_CLOSE_TAG = "\t\t</node>\n";

////////////////////////////////////////////////////////////////////////////
////getter methods for graphML node element constants//////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of NODE_XML_OPEN_TAG to child
     * classes.
     * @return Value of private NODE_XML_OPEN_TAG constant.
     */
    protected static final String GET_NODE_XML_OPEN_TAG() {
        return NODE_XML_OPEN_TAG;}

    /**Getter method which returns the value of NODE_XML_OPEN_TAG_END to child
     * classes.
     * @return Value of private NODE_XML_OPEN_TAG_END constant.
     */
    protected static final String GET_NODE_XML_OPEN_TAG_END() {
        return NODE_XML_OPEN_TAG_END;}

    /**Getter method which returns the value of NODE_XML_DATA_TAG_OPEN to child
     * classes.
     * @return Value of private NODE_XML_DATA_OPEN_TAG constant.
     */
    protected static final String GET_NODE_XML_DATA_OPEN_TAG() {
        return NODE_XML_DATA_OPEN_TAG;}

    /**Getter method which returns the value of NODE_XML_DATA_TAG_END to child
     * classes.
     * @return Value of private NODE_XML_DATA_OPEN_TAG_END constant.
     */
    protected static final String GET_NODE_XML_DATA_OPEN_TAG_END() {
        return NODE_XML_DATA_OPEN_TAG_END;}

    /**Getter method which returns the value of NODE_XML_DATA_CLOSE_TAG to
     * child classes.
     * @return Value of private NODE_XML_DATA_CLOSE_TAG constant.
     */
    protected static final String GET_NODE_XML_DATA_CLOSE_TAG() {
        return NODE_XML_DATA_CLOSE_TAG;}

    /**Getter method which returns the value of NODE_XML_CLOSE_TAG to child
     * classes.
     * @return Value of private NODE_XML_CLOSE_TAG constant.
     */
    protected static final String GET_NODE_XML_CLOSE_TAG() {
        return NODE_XML_CLOSE_TAG;}

////////////////////////////////////////////////////////////////////////////
```

```
////constructors/////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
     */
    public Node() {}

    /**Instantiates a Node object and assigns values to the id and data fields.
     * <p />Currently the only constructor used by the application.
     * @param nodeId
     * Assigned to a Node object's id field. Used as the value of an id=""
     * attribute of a &lt;node> graphML XML element.
     * @param nodeData
     * Assigned to a Node object's data field. Can be used to populate one or
     * more attributes of a &lt;data> graphML XML element, or even additional
     * &lt;data> elements.
     */
    public Node(String nodeId, String nodeData) {

        id = nodeId;
        data = nodeData;
    }

////////////////////////////////////////////////////////////////////////////////
////public methods//////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed &lt;node> graphML XML element.
     * @return
     * A well-formed &lt;node> graphML XML element.
     */
    public String getNodeXML() {

        //instantiates a String object, compiles an opening &lt;node> element,
        //and assigns a value to the id="" attribute
        String xmlOut = NODE_XML_OPEN_TAG + id + NODE_XML_OPEN_TAG_END;

        //compiles &lt;data> element(s) when necessary
        //&lt;data> elements must match corresponding &lt;key> elements defined
        //at the top of a graphML XML document
        if (!data.equals("")) xmlOut += NODE_XML_DATA_OPEN_TAG + data
                + NODE_XML_DATA_OPEN_TAG_END;

        //compiles &lt;data> closing element(s) when necessary
        if (!data.equals("")) xmlOut += NODE_XML_DATA_CLOSE_TAG;

        //compiles a &lt;node> closing element
        xmlOut += NODE_XML_CLOSE_TAG;

        return xmlOut;
    }
}
```

G. THE YEDEDGE CLASS

```
package MPGrapher;

/**Constructs yEdEdge objects and compiles and returns well-formed,
 * yEd-specific &lt;edge> graphML XML elements.
```

```
 * @author Capt Timothy L. Shields USMC
 * @since 1.0 last modified July 29, 2012
 */
public class yEdEdge extends Edge {

////////////////////////////////////////////////////////////////////////////
////variable class fields/////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Field which represents the ID of a yEdEdge object.*/
    private String id;

    /**Field used to compile well-formed &lt;data> graphML XML elements (child
     * classes may include additional &lt;data> elements here).
     */
    private String data;

    /**Field which represents the type of a yEdEdge object - the type field is
     * used to control all parameters which affect an edge's appearance in the
     * yEd graphing application.
     */
    private String type;

    /**Field that holds a String representation of a yEdEdge object's line
     * color.
     */
    private String color;

    /**Field that represents the number of labels associated with this yEdEdge
     * object.
     */
    private int numLabels;

    /**Field used to compile source="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private String source;

    /**Field used to compile target="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private String target;

    /**Field used to compile x="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private double labelX;

    /**Field used to compile y="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private double labelY;

    /**Field that holds a String representation of a yEdEdge object's line-
     * thickness.
     */
    private double weight;

    /**Field that holds X-Y coordinates that are used to route edges around
     * nodes in the yEd graphing application.
     */
    private double[] points;
```

```
    /**Field used to indicate whether this yEdEdge is directed or undirected -
     * all edges are assumed to be either non-directed or uni-directed (bi-
     * directional edges are not currently supported).
     */
    private String directed;

    /**Array field of String values which represent the text(s) to be used as
     * this yEdEdge's labels.
     */
    private String[] labels;

    /**Field that holds a String representation of a yEdEdge object's line-
     * style ("dashed" for example).
     */
    private String lineStyle;

    /**Field that indicates whether X-Y point coordinates have been calculated
     * for this yEdEdge object.
     */
    private boolean hasPoints;

////////////////////////////////////////////////////////////////////////////
////yEd-specific graphML <edge> element constants////////////////////////////
////////////////////////////////////////////////////////////////////////////

/*Uncomment these lines to produce different kinds of yEd edges***************/
/*private static final String YED_EDGE_PATH_TAGS = "\t\t\t\t<y:PolyLineEdge>\n"
/*private static final String YED_EDGE_PATH_TAGS = "\t\t\t\t<y:ArcEdge>\n"
/*************************************************************************/

    /**yEd-specific graphML XML element: defines the opening tags of
     * well-formed, yEd-specific graphML &lt;y:QuadEdgeCurve> and &lt;y:Path>
     * elements.
     */
    private static final String YED_EDGE_PATH_TAGS =
            "\t\t\t\t<y:QuadEdgeCurve>\n"
            + "\t\t\t\t\t<y:Path sx=\"0.0\" sy=\"0.0\" tx=\"0.0\" "
                + "ty=\"0.0\">\n";

    /**yEd-specific graphML XML element: defines the opening of a well-formed,
     * yEd-specific graphML &lt;y:Point> element, up to the x="" attribute.
     */
    private static final String YED_EDGE_POINTS_X =
            "\t\t\t\t\t\t<y:Point x=\"";

    /**yEd-specific graphML XML element: defines the y="" attribute of a
     * well-formed, yEd-specific graphML &lt;y:Point> element.
     */
    private static final String YED_EDGE_POINTS_Y = "\" y=\"";

    /**yEd-specific graphML XML element: defines the close of a well-formed,
     * yEd-specific graphML &lt;y:Point> element.
     */
    private static final String YED_EDGE_POINTS_CLOSE_TAGS = "\"/>\n";

    /**yEd-specific graphML XML element: closes the &lt;y:Path> element and
     * begins the definition of a well-formed, yEd-specific graphML
     * &lt;y:LineStyle> element up to the color="" attribute.
     */
    private static final String YED_EDGE_COLOR = "\t\t\t\t\t</y:Path>\n"
            + "\t\t\t\t\t<y:LineStyle color=\"";
```

```
    /**yEd-specific graphML XML element: defines the type="" attribute of a
     * well-formed, yEd-specific graphML &lt;y:LineColor> element.
     */
    private static final String YED_EDGE_LINE_STYLE = "\" type=\"";

    /**yEd-specific graphML XML element: defines the width="" attribute of a
     * well-formed, yEd-specific graphML &lt;y:LineColor> element.
     */
    private static final String YED_EDGE_WIDTH = "\" width=\"";

    /**yEd-specific graphML XML element: defines the close of a well-formed,
     * yEd-specific graphML &lt;y:LineStyle> element and begins the definition
     * of a &lt;y:Arrows> element, up to the target="" attribute.
     */
    private static final String YED_EDGE_TARGET = "\"/>\n"
            + "\t\t\t\t\t<y:Arrows source=\"none\" target=\"";

    /**yEd-specific graphML XML element: defines the close of a well-formed,
     * yEd-specific graphML &lt;y:Arrows> element and also defines a
     * &lt;y:EdgeLabel> element.
     */
    private static final String YED_EDGE_LABEL = "\"/>\n"
            + "\t\t\t\t\t<y:EdgeLabel alignment=\"center\" distance=\"2.0\" "
                + "fontFamily=\"Courier New\" fontSize=\"12\" "
                + "fontStyle=\"plain\" hasBackgroundColor=\"false\" "
                + "hasLineColor=\"false\" height=\"34.0\" "
                + "modelName=\"custom\" preferredPlacement=\"anywhere\" "
                + "ratio=\"0.5\" textColor=\"#000000\" visible=\"true\" "
                + "width=\"40.0\" x=\"0.0\" y=\"0.0\">\n\t\t\t\t\t\t\t";

    /**yEd-specific graphML XML element: defines additional well-formed,
     * yEd-specific graphML elements that are necessary for the proper display
     * of graphs in the yEd graphing application.
     */
    private static final String YED_EDGE_CLOSE =
            "\n\t\t\t\t\t\t<y:LabelModel>\n"
            + "\t\t\t\t\t\t<y:SmartEdgeLabelModel "
                + "autoRotationEnabled=\"false\" "
                + "defaultAngle=\"0.0\" defaultDistance=\"10.0\"/>\n"
            + "\t\t\t\t\t</y:LabelModel>\n"
            + "\t\t\t\t\t<y:ModelParameter>\n"
            + "\t\t\t\t\t<y:SmartEdgeLabelModelParameter angle=\"0.0\" "
                + "distance=\"30.0\" distanceToCenter=\"true\" "
                + "position=\"right\" ratio=\"0.5\" segment=\"0\"/>\n"
            + "\t\t\t\t\t</y:ModelParameter>\n"
            + "\t\t\t\t</y:EdgeLabel>\n"
            + "\t\t\t<y:BendStyle smoothed=\"true\"/>\n"
            + "\t\t</y:QuadEdgeCurve>\n";
/*Uncomment these lines to produce different kinds of yEd edges***************/
/*            + "\t\t\t</y:ArcEdge>\n";
/*            + "\t\t\t</y:PolyLineEdge>\n";
/******************************************************************************/

//////////////////////////////////////////////////////////////////////////////
////getter and setter methods for private yEdEdge fields//////////////////////
//////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of this yEdEdge object's id to
     * child classes.
     * @return Value of private id variable.
     */
```

```
protected String getEdgeID() {return id;}

/**Getter method which returns the value of type to child classes.
 * @return Value of private type variable.
 */
protected String getEdgeType() {return type;}

/**Getter method which returns the value of source to child classes.
 * @return Value of private source variable.
 */
protected String getEdgeSource() {return source;}

/**Getter method which returns the value of target to child classes.
 * @return Value of private target variable.
 */
protected String getEdgeTarget() {return target;}

/**Getter method which returns the value of hasPoints to child classes.
 * @return Value of private hasPoints variable.
 */
protected boolean hasPoints() {return hasPoints;}

/**Setter method which allows child classes to set the value of labelX.
 * @param labelXCoord
 * Double value which is to be used as the label's X-coordinate.
 */
protected void setLabelX(double labelXCoord) {labelX = labelXCoord;}

/**Setter method which allows child classes to set the value of labelY.
 * @param labelYCoord
 * Double value which is to be used as the label's Y-coordinate.
 */
protected void setLabelY(double labelYCoord) {labelY = labelYCoord;}

/**Setter method which allows child classes to set the value of weight.
 * @param edgeWeight
 * Double value which is to be used to control the edge's line-thickness in
 * the yEd graphing application.
 */
protected void setEdgeWeight(double edgeWeight) {weight = edgeWeight;}

/**Setter method which allows child classes to set the value of color.
 * @param edgeLineColor
 * String representation of the color that the yEd graphing application is
 * to use to color the edge.
 */
protected void setEdgeColor(String edgeLineColor) {color = edgeLineColor;}

/**Setter method which allows child classes to set the value of directed.
 * @param edgeDirected
 * Edge-type String which the compiler will interpret as eight directed or
 * non-directed.
 */
protected void setEdgeDirected(String edgeDirected) {
        directed = edgeDirected;}

/**Setter method which allows child classes to set the value of lineStyle.
 * @param edgeLineStyle
 * String value which will be used to control the line-style (dashed for
 * example) of the edge in the yEd graphing application.
 */
protected void setEdgeLineStyle(String edgeLineStyle) {
```

```
            lineStyle = edgeLineStyle;}

    /**Setter method which allows child classes to set the value of this
     * yEdEdge object's labels array.
     * @param edgeLabels
     * An array of String's that will be used as the texts for the yEdEdge
     * object's labels.
     */
    protected void setLabels(String[] edgeLabels) {
            System.arraycopy(edgeLabels, 0, labels, 0, numLabels);}

    /**Setter method which allows child classes to set the value of this
     * yEdEdge object's points array and hasPoints field.
     * @param edgePoints
     * Array of doubles that will be copied to the points field.
     */
    protected void setPoints(double[] edgePoints) {

        System.arraycopy(edgePoints, 0, points, 0, 4);
/*Uncomment this line to produce ArcEdge kinds of yEd edges******************/
/*        System.arraycopy(edgePoints, 0, points, 0, 2);
/*************************************************************************/
        hasPoints = true;
    }

////////////////////////////////////////////////////////////////////////////
////getter methods for yEd-specific graphML edge element constants/////////
////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of GET_YED_EDGE_PATH_TAGS to child
     * classes.
     * @return Value of private GET_YED_EDGE_PATH_TAGS constant.
     */
    private static final String GET_YED_EDGE_PATH_TAGS() {
            return YED_EDGE_PATH_TAGS;}

    /**Getter method which returns the value of GET_YED_EDGE_POINTS_X to child
     * classes.
     * @return Value of private GET_YED_EDGE_POINTS_X constant.
     */
    private static final String GET_YED_EDGE_POINTS_X() {
            return YED_EDGE_POINTS_X;}

    /**Getter method which returns the value of GET_YED_EDGE_POINTS_Y to child
     * classes.
     * @return Value of private GET_YED_EDGE_POINTS_Y constant.
     */
    private static final String GET_YED_EDGE_POINTS_Y() {
            return YED_EDGE_POINTS_Y;}

    /**Getter method which returns the value of GET_YED_EDGE_POINTS_CLOSE_TAG
     * to child classes.
     * @return Value of private GET_YED_EDGE_POINTS_CLOSE_TAG constant.
     */
    private static final String GET_YED_EDGE_POINTS_CLOSE_TAGS() {
            return YED_EDGE_POINTS_CLOSE_TAGS;}

    /**Getter method which returns the value of GET_YED_EDGE_COLOR to child
     * classes.
     * @return Value of private GET_YED_EDGE_COLOR constant.
     */
    private static final String GET_YED_EDGE_COLOR() {return YED_EDGE_COLOR;}
```

```
    /**Getter method which returns the value of GET_YED_EDGE_LINE_STYLE to
     * child classes.
     * @return Value of private GET_YED_EDGE_LINE_STYLE constant.
     */
    private static final String GET_YED_EDGE_LINE_STYLE() {
            return YED_EDGE_LINE_STYLE;}

    /**Getter method which returns the value of GET_YED_EDGE_WIDTH to child
     * classes.
     * @return Value of private GET_YED_EDGE_WIDTH constant.
     */
    private static final String GET_YED_EDGE_WIDTH() {return YED_EDGE_WIDTH;}

    /**Getter method which returns the value of GET_YED_EDGE_TARGET to child
     * classes.
     * @return Value of private GET_YED_EDGE_TARGET constant.
     */
    private static final String GET_YED_EDGE_TARGET() {return YED_EDGE_TARGET;}

    /**Getter method which returns the value of GET_YED_EDGE_LABEL to child
     * classes.
     * @return Value of private GET_YED_EDGE_LABEL constant.
     */
    private static final String GET_YED_EDGE_LABEL() {return YED_EDGE_LABEL;}

    /**Getter method which returns the value of GET_YED_EDGE_CLOSE to child
     * classes.
     * @return Value of private GET_YED_EDGE_CLOSE constant.
     */
    private static final String GET_YED_EDGE_CLOSE() {return YED_EDGE_CLOSE;}

////////////////////////////////////////////////////////////////////////////
////constructors////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
     */
    public yEdEdge() {}

    /**Instantiates a yEdEdge object and assigns values to fields.
     * @param edgeID
     * Assigned to a yEdEdge object's id field. Used as the value of an id=""
     * attribute of a &lt;node> graphML XML element. <p />
     * @param edgeData
     * Assigned to a yEdEdge object's data field. Can be used to populate one
     * or more attributes of a &lt;data> graphML XML element, or even
     * additional &lt;data> elements.<p />
     * @param edgeType
     * Assigned to a yEdNode object's type field. The type field is used to
     * control all parameters which affect an edge's appearance in the yEd
     * graphing application.  <p />
     * @param edgeColor
     * Assigned to a yEdNode object's color field. Used to color output edges
     * in the yEd graphing application. <p />
     * @param edgeSource
     * Assigned to a yEdNode object's source field. Used as the "from" node for
     * drawing edges in all graphing applications that support the graphML
     * markup language. <p />
     * @param edgeTarget
```

```
     * Assigned to a yEdNode object's target field. Used as the "to" node for
     * drawing edges in all graphing applications that support the graphML
     * markup language. <p />
     * @param edgeWeight
     * Assigned to a yEdNode object's weight field. Used to control the
     * thickness of output edges in the yEd graphing application. <p />
     * @param edgeDirected
     * Assigned to a yEdNode object's directed field. All edges are assumed to
     * be either non-directed or uni-directed (bi-directional edges are not
     * currently supported). <p />
     * @param edgeLineStyle
     * Assigned to a yEdNode object's lineStyle field. Used to control the
     * type of line of output edges in the yEd graphing application ("dashed"
     * for example). <p />
     * @param numEdgeLabels
     * Assigned to a yEdNode object's numLabels field. Used to indicate the
     * number of this yEdEdge object's output labels in the yEd graphing
     * application.
     */
    public yEdEdge(String edgeID, String edgeData, String edgeType,
            String edgeColor, String edgeSource, String edgeTarget,
            double edgeWeight, String edgeDirected, String edgeLineStyle,
            int numEdgeLabels) {

        //all field values are taken directly from constructor's parameters
        id = edgeID;
        data = edgeData;
        type =  edgeType;
        color = edgeColor;
        hasPoints = false;
        source = edgeSource;
        target = edgeTarget;
        weight = edgeWeight;
        points = new double[4];
/*Uncomment this line to produce ArcEdge kinds of yEd edges******************/
/*        points = new double[2];
/*****************************************************************************/
        directed = edgeDirected;
        lineStyle = edgeLineStyle;
        numLabels = numEdgeLabels;
        if (numLabels > 0) labels = new String[numLabels];
    }

////////////////////////////////////////////////////////////////////////////
////public methods//////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed, yEd-specific &lt;edge> graphML XML
     * element.
     * @return
     * A well-formed, yEd-specific &lt;edge> graphML XML element.
     */
    public String get_yEdEdgeXML() {

        //instantiate the return String and set it to the start of a
        //well-formed, yEd-specific <edge> element with attribute values
        //specific to this yEdEdge
        String xmlOut = GET_EDGE_XML_OPEN_TAG() + id
                + GET_EDGE_XML_SOURCE() + source
                + GET_EDGE_XML_TARGET() + target
                + GET_EDGE_XML_OPEN_TAG_END()
                + GET_EDGE_XML_DATA_OPEN_TAG() + data
```

```
                + GET_EDGE_XML_DATA_OPEN_TAG_END() + GET_YED_EDGE_PATH_TAGS();

        //if this yEdEdge has points, then construct the appropriate <y:Point>
        //elements and add them to the return String
        if (hasPoints) xmlOut += GET_YED_EDGE_POINTS_X() + points[0]
                + GET_YED_EDGE_POINTS_Y() + points[1]
                + GET_YED_EDGE_POINTS_CLOSE_TAGS()
                + GET_YED_EDGE_POINTS_X() + points[2]
                + GET_YED_EDGE_POINTS_Y() + points[3]
/*Uncomment this line to produce ArcEdge kinds of yEd edges******************/
/*        if (hasPoints) xmlOut += GET_YED_EDGE_POINTS_X() + 0.0
/*                + GET_YED_EDGE_POINTS_Y() + 0.0
/*****************************************************************************/
                + GET_YED_EDGE_POINTS_CLOSE_TAGS();

        //add additional elements and attributes to the return String
        xmlOut += GET_YED_EDGE_COLOR() + color
                + GET_YED_EDGE_LINE_STYLE() + lineStyle
                + GET_YED_EDGE_WIDTH() + weight
                + GET_YED_EDGE_TARGET() + directed
                + GET_YED_EDGE_LABEL();

        //if this yEdEdge has labels, then add them to the return String also
        if (numLabels > 0) {

            //add the first label
            xmlOut += labels[0];

            //if there are additional labels then add them as well, with
            //intervening carriage return characters
            if (numLabels > 1) {

                for (int i = 1; i < numLabels; i++) xmlOut += "\n" + labels[i];
            }
        }

        //add the remaining elements necessary for a well-formed, yEd-specific
        //graphML XML <edge> element
        xmlOut += GET_YED_EDGE_CLOSE() + GET_EDGE_XML_DATA_CLOSE_TAG()
                + GET_EDGE_XML_CLOSE_TAG();

        return xmlOut;
    }
}
```

H. THE EDGE CLASS

```
package MPGrapher;

/**Constructs Edge objects and compiles and returns well-formed &lt;edge>
 * graphML XML elements.
 * @author Capt Timothy L. Shields USMC
 * @since version 1.0 last modified July 23, 2012
 */
public class Edge {

//////////////////////////////////////////////////////////////////////////////
////variable class fields/////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////

    /**Field which represents the ID of an Edge object.*/
```

```
    private String id;

    /**Field used to compile well-formed &lt;data> graphML XML elements (child
     * classes may include additional &lt;data> elements here).
     */
    private String data;

    /**Field used to compile source="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private String source;

    /**Field used to compile target="" attributes of well-formed &lt;edge>
     * graphML XML elements.
     */
    private String target;

////////////////////////////////////////////////////////////////////////////
////graphML edge element constants//////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**GraphML XML element: defines the start of a well-formed graphML
     * &lt;edge> element.
     */
    private static final String EDGE_XML_OPEN_TAG = "\t\t<edge id=\"";

    /**GraphML XML element attribute: defines the source="" attribute of a
     * well-formed graphML &lt;edge> element.
     */
    private static final String EDGE_XML_SOURCE = "\" source=\"";

    /**GraphML XML element attribute: defines the target="" attribute of a
     * well-formed graphML &lt;edge> element.
     */
    private static final String EDGE_XML_TARGET = "\" target=\"";

    /**GraphML XML element: defines the end of a well-formed graphML &lt;edge>
     * element.
     */
    private static final String EDGE_XML_OPEN_TAG_END = "\">\n";

    /**GraphML XML element: defines the start of a well-formed graphML
     * &lt;data> element associated with a &lt;edge> element.
     */
    private static final String EDGE_XML_DATA_OPEN_TAG = "\t\t\t<data key=\"";

    /**GraphML XML element: defines the end of a well-formed graphML &lt;data>
     * element associated with a &lt;edge> element.
     */
    private static final String EDGE_XML_DATA_OPEN_TAG_END = "\">\n";

    /**GraphML XML element: defines the closing element of a well-formed
     * graphML &lt;data> element associated with a &lt;edge> element.
     */
    private static final String EDGE_XML_DATA_CLOSE_TAG = "\t\t\t</data>\n";

    /**GraphML XML element: defines the closing element of a well-formed
     * graphML &lt;edge> element.
     */
    private static final String EDGE_XML_CLOSE_TAG = "\t\t</edge>\n";

////////////////////////////////////////////////////////////////////////////
```

```
////getter methods for graphML edge element constants//////////////////////////
////////////////////////////////////////////////////////////////////////////////

    /**Getter method which returns the value of EDGE_XML_OPEN_TAG to child
     * classes.
     * @return Value of private EDGE_XML_OPEN_TAG constant.
     */
    protected static final String GET_EDGE_XML_OPEN_TAG() {
        return EDGE_XML_OPEN_TAG;}

    /**Getter method which returns the value of EDGE_XML_SOURCE to child
     * classes.
     * @return Value of private EDGE_XML_SOURCE constant.
     */
    protected static final String GET_EDGE_XML_SOURCE() {
        return EDGE_XML_SOURCE;}

    /**Getter method which returns the value of EDGE_XML_TARGET to child
     * classes.
     * @return Value of private EDGE_XML_TARGET constant.
     */
    protected static final String GET_EDGE_XML_TARGET() {
        return EDGE_XML_TARGET;}

    /**Getter method which returns the value of EDGE_XML_OPEN_TAG_CLOSE to
     * child classes.
     * @return Value of private EDGE_XML_OPEN_TAG_CLOSE constant.
     */
    protected static final String GET_EDGE_XML_OPEN_TAG_END() {
        return EDGE_XML_OPEN_TAG_END;}

    /**Getter method which returns the value of EDGE_XML_DATA_OPEN_TAG to child
     * classes.
     * @return Value of private EDGE_XML_DATA_OPEN_TAG constant.
     */
    protected static final String GET_EDGE_XML_DATA_OPEN_TAG() {
        return EDGE_XML_DATA_OPEN_TAG;}

    /**Getter method which returns the value of EDGE_XML_DATA_OPEN_TAG_END to
     * child classes.
     * @return Value of private EDGE_XML_DATA_OPEN_TAG_END constant.
     */
    protected static final String GET_EDGE_XML_DATA_OPEN_TAG_END() {
        return EDGE_XML_DATA_OPEN_TAG_END;}

    /**Getter method which returns the value of EDGE_XML_DATA_CLOSE_TAG to
     * child classes.
     * @return Value of private EDGE_XML_DATA_CLOSE_TAG constant.
     */
    protected static final String GET_EDGE_XML_DATA_CLOSE_TAG() {
        return EDGE_XML_DATA_CLOSE_TAG;}

    /**Getter method which returns the value of EDGE_XML_CLOSE_TAG to child
     * classes.
     * @return Value of private EDGE_XML_CLOSE_TAG constant.
     */
    protected static final String GET_EDGE_XML_CLOSE_TAG() {
        return EDGE_XML_CLOSE_TAG;}

////////////////////////////////////////////////////////////////////////////////
////constructors////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
    /**Basic constructor with no parameters.
     * <p />Not currently used by the application but necessary for defining
     * sub-classes which inherit from this class.
     */
    public Edge () {

        id = "";
        source = "";
        target = "";
        data = "";
    }

    /**Instantiates an Edge object and assigns values to the id, source, target
     * and data fields.
     * Currently the only constructor used by the application.
     * @param edgeSource
     * Assigned to an Edge object's source field. Used as the value of a
     * source="" attribute of an &lt;edge> graphML XML element.
     * @param edgeTarget
     * Assigned to an Edge object's target field. Used as the value of a
     * target="" attribute of an &lt;edge> graphML XML element.
     * @param edgeData
     * Assigned to a Edge object's data field. Can be used to populate one or
     * more attributes of a &lt;data> graphML XML element, or even additional
     * &lt;data> elements.
     */
    public Edge (String edgeSource, String edgeTarget, String edgeData) {

        //uses target, concatenated to source with an intervening "::" as the
        //edge's id
        id = edgeSource + "::" + edgeTarget;
        source = edgeSource;
        target = edgeTarget;
        data = edgeData;
    }

////////////////////////////////////////////////////////////////////////////
////public methods//////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

    /**Compiles and returns a well-formed &lt;node> graphML XML element.
     * @return
     * A well-formed &lt;node> graphML XML element.
     */
    public String getEdgeXML () {

        //instantiates a String object, compiles an opening &lt;edge> element,
        //and assigns value to the id="", source="" and target="" attributes
        String xmlOut = EDGE_XML_OPEN_TAG + id + EDGE_XML_SOURCE + source
                + EDGE_XML_TARGET + target + EDGE_XML_OPEN_TAG_END;

        //compiles &lt;data> element(s) when necessary
        //&lt;data> elements must match corresponding &lt;key> elements defined
        //at the top of a graphML XML document
        if (!data.equals("")) {
            xmlOut += EDGE_XML_DATA_OPEN_TAG + data
                    + EDGE_XML_DATA_OPEN_TAG_END;
        }

        //compiles &lt;data> closing element(s) when necessary
        if (!data.equals("")) xmlOut += EDGE_XML_DATA_CLOSE_TAG;
```

```
        //compiles a &lt;edge> closing element
        xmlOut += EDGE_XML_CLOSE_TAG;

        return xmlOut;
    }
}
```

APPENDIX B. SPQR TREES[8]

A. S TREES

In an S node, the associated graph is a cycle graph with three or more vertices and edges. Individual nodes have a maximum degree of two. The S stands for "series".

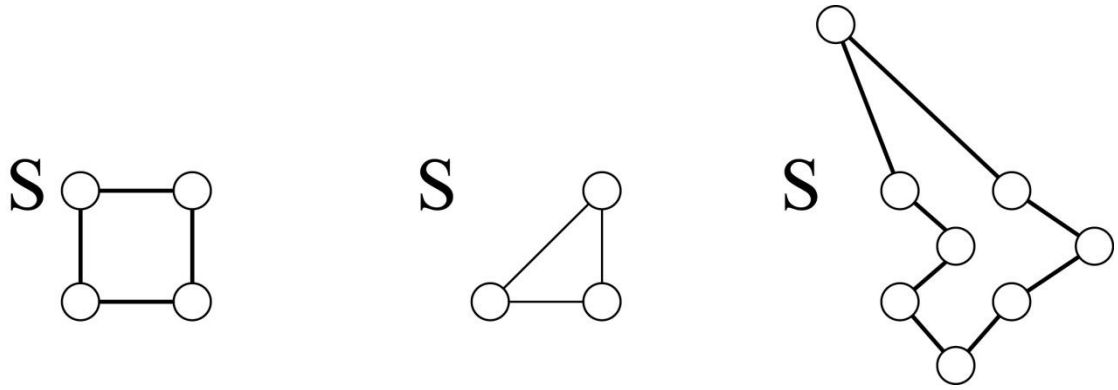

Figure 69. Examples of S trees

B. R TREES

In an R node, the associated graph is a 3-connected graph that is not a cycle or dipole. In the application of SPQR trees to planar graph embedding, the associated graph of an R node has a unique planar embedding. The R stands for "rigid".

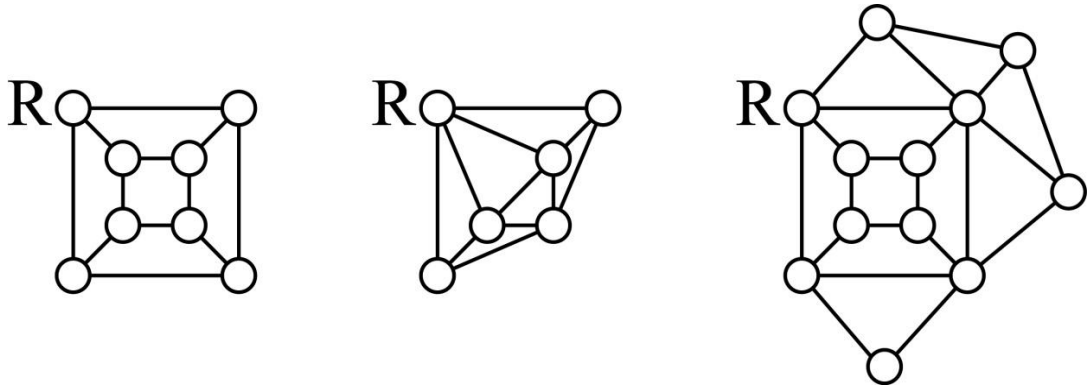

Figure 70. Examples of R trees

[8] Material for Appendix 2 borrows heavily from (Wikipedia contributors, 2012).

C. P TREES

In a P node, the associated graph is a dipole graph, a multigraph with two vertices and three or more edges, or the planar dual to a cycle graph. This case is analogous to parallel composition in series-parallel graphs. The P stands for "parallel".

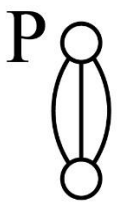

Figure 71. Example of a P tree

D. Q TREES

In a Q node, the associated graph has a single edge. This trivial case is necessary to handle the graph that has only one edge, but does not appear in the SPQR trees of more complex graphs. Q nodes are used to join groups of S, P and R groups of nodes into a larger graph.

APPENDIX C. LIST OF SEARCHED DATABASES

ACM Digital Library (ACM)
BOSUN (Dudley Knox Library)
Compendex (Engineering Village)
Computer and Information Systems Abstracts (ProQuest)
COMPUTER-SCIENCEnetBASE (CRCnetBASE)
Computing Reviews (Computing Reviews)
DTIC Online (Defense Technical Information Center)
ebrary (ebrary)
Engineering Village 2 (Engineering Village 2)
ENGnetBASE: Engineering Handbooks Online (CRCnetBASE)
IEEE/IET Electronic Library (IEEE Xplore)
IGI Global (IGI Global)
Internet and Personal Computing Abstracts (EBSCO)
ITECHnetBASE (CRCnetBASE)
Navy Modeling and Simulation Office (NMSO)
Networked Computer Science Technical Reference Library (NCSTRL)
NTIS (National Technical Information Service)
ProQuest (ProQuest)
ProQuest Computing (ProQuest)
Science Citation Index (ISI Web of Knowledge)
SPIE Digital Library (SPIE)
SpringerLink (Springer)
Synthesis Digital Library of Engineering and Computer Science (Morgan & Claypool)
TelecommunicationsnetBASE (CRCnetBASE)
WorldCat (OCLC)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. SAMPLE GXL DOCUMENT

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gxl SYSTEM "http://www.gupro.de/GXL/gxl-1.0.dtd">
<gxl xmlns:xlink=" http://www.w3.org/1999/xlink">
  <graph id="simpleExample-instance" edgeids=" true" edgemode=" directed"
hypergraph=" false">
    <type
xlink:href="http://www.gupro.de/GXL/examples/schema/gxl/simpleExample/simpleExa
mpleSchema.gxl#simpleExampleSchema" xlink:type=" simple"/>
    <node id="p">
      <type
xlink:href="http://www.gupro.de/GXL/examples/schema/gxl/simpleExample/simpleExa
mpleSchema.gxl#Proc" xlink:type=" simple"/>
      <attr name=" file">
        <string> main.c</string>
      </attr>
    </node>
    <node id="q">
      <type
xlink:href="http://www.gupro.de/GXL/examples/schema/gxl/simpleExample/simpleExa
mpleSchema.gxl#Proc" xlink:type=" simple"/>
      <attr name=" file">
        <string> test.c</string>
      </attr>
    </node>
    <edge id="c" to="q" from="p">
      <type
xlink:href="http://www.gupro.de/GXL/examples/schema/gxl/simpleExample/simpleExa
mpleSchema.gxl#Call" xlink:type=" simple"/>
      <attr name=" line">
        <int> 42</int>
      </attr>
    </edge>
  </graph>
</gxl>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. LANGUAGES SUPPORTED BY NOTEPAD++

ADA	ASP
Assembly	AutoIt
Batch	C
C#	C++
Caml	Cmake
COBOL	CSS
D	Diff
Flash actionscript	Fortran
Gui4Cli	Haskell
HTML	INNO
Java	Javascript
JSP	KIXtart
LISP	Lua
Makefile	Matlab
MS INI file	MS-DOS Style
Normal Text	NSIS
Objective-C	Pascal
Perl	PHP
Postscript	PowerShell
Properties	Python
R	Resource File
Ruby	Shell
Scheme	Smalltalk
SQL	TCL
Tex	Visual Basic
VHDL	Verilog
XML	YAML

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. GRAPHML FILE WITH TREE LAYOUT IN YED

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:y="http://www.yworks.com/xml/graphml"
  xmlns:yed="http://www.yworks.com/xml/yed/3"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
    http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
  <key for="graphml" id="d0" yfiles.type="resources"/>
  <key for="port" id="d1" yfiles.type="portgraphics"/>
  <key for="port" id="d2" yfiles.type="portgeometry"/>
  <key for="port" id="d3" yfiles.type="portuserdata"/>
  <key attr.name="url" attr.type="string" for="node" id="d4"/>
  <key attr.name="description" attr.type="string" for="node" id="d5"/>
  <key for="node" id="d6" yfiles.type="nodegraphics"/>
  <key attr.name="url" attr.type="string" for="edge" id="d7"/>
  <key attr.name="description" attr.type="string" for="edge" id="d8"/>
  <key for="edge" id="d9" yfiles.type="edgegraphics"/>
  <graph edgedefault="directed" id="G">
    <node id="n0">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="30.0" x="52.5" y="0.0"/>
          <y:Fill color="#FFCC00" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
            fontFamily="Dialog" fontSize="12" fontStyle="plain"
            hasBackgroundColor="false" hasLineColor="false" hasText="false"
            height="4.0" modelName="custom" textColor="#000000" visible="true"
            width="4.0" x="13.0" y="13.0">
            <y:LabelModel>
              <y:SmartNodeLabelModel distance="4.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartNodeLabelModelParameter labelRatioX="0.0"
               labelRatioY="0.0" nodeRatioX="0.0" nodeRatioY="0.0"
               offsetX="0.0" offsetY="0.0" upX="0.0" upY="-1.0"/>
            </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="rectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="n1">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="30.0" x="0.0" y="70.0"/>
          <y:Fill color="#FFCC00" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
            fontFamily="Dialog" fontSize="12" fontStyle="plain"
            hasBackgroundColor="false" hasLineColor="false" hasText="false"
            height="4.0" modelName="custom" textColor="#000000" visible="true"
            width="4.0" x="13.0" y="13.0">
            <y:LabelModel>
              <y:SmartNodeLabelModel distance="4.0"/>
```

```
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartNodeLabelModelParameter labelRatioX="0.0"
               labelRatioY="0.0" nodeRatioX="0.0" nodeRatioY="0.0"
               offsetX="0.0" offsetY="0.0" upX="0.0" upY="-1.0"/>
            </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="rectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="n2">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="30.0" x="105.0" y="70.0"/>
          <y:Fill color="#FFCC00" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
           fontFamily="Dialog" fontSize="12" fontStyle="plain"
           hasBackgroundColor="false" hasLineColor="false" hasText="false"
           height="4.0" modelName="custom" textColor="#000000" visible="true"
           width="4.0" x="13.0" y="13.0">
            <y:LabelModel>
              <y:SmartNodeLabelModel distance="4.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartNodeLabelModelParameter labelRatioX="0.0"
               labelRatioY="0.0" nodeRatioX="0.0" nodeRatioY="0.0"
               offsetX="0.0" offsetY="0.0" upX="0.0" upY="-1.0"/>
            </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="rectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <edge id="e0" source="n0" target="n1">
      <data key="d8"/>
      <data key="d9">
        <y:PolyLineEdge>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0"/>
          <y:LineStyle color="#000000" type="line" width="1.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:BendStyle smoothed="false"/>
        </y:PolyLineEdge>
      </data>
    </edge>
    <edge id="e1" source="n0" target="n2">
      <data key="d8"/>
      <data key="d9">
        <y:PolyLineEdge>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0"/>
          <y:LineStyle color="#000000" type="line" width="1.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:BendStyle smoothed="false"/>
        </y:PolyLineEdge>
      </data>
    </edge>
  </graph>
  <data key="d0"> <y:Resources/> </data>
</graphml>
```

APPENDIX G. GRAPHML GRAMMAR

```
input_description ::=    "%%nodes%%" graph "%%eog%%" ;

graph ::=                ( + unique_id + )
                         "%%edges%%"
                         ( * (unique_id unique_id ";") * ) ;

unique_id ::=            any string of printable characters, minus whitespace
                         characters
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. YED-GRAPHML GRAMMAR

```
input_description ::=    "%yEd%" graph "%%eog%%" ;

graph ::=                "%%nodes%%" ( trace | architecture ) ;

trace ::=                "%trace_roots%" ( + unigue_id label + )
                         "%trace_events%" ( * unique_id label * )
                         "%%edges%%"
                            ( * unique_id unique_id
                               ( "INCLUDES" | "PRECEDES" )
                               ";"
                            * ) ;

architecture ::=         "%architecture_environment%" ( + unique_id label + )
                         "%architecture_processes%" ( + unique_id label + )
                         "%architecture_data%" ( * unique_id label * )
                         "%%edges%%"
                            ( * unique_id unique_id
                               ( "SHARE_ALL" | "ADD" )
                               ( * label * )
                               ";"
                            * ) ;

label ::=                any string of printable characters, plus whitespace
                         characters, minus quotes, contained inside quotes

unique_id ::=            any string of printable characters, minus whitespace
                         characters, not repeated anywhere else in the input
                         file
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. CLASS DIAGRAMS

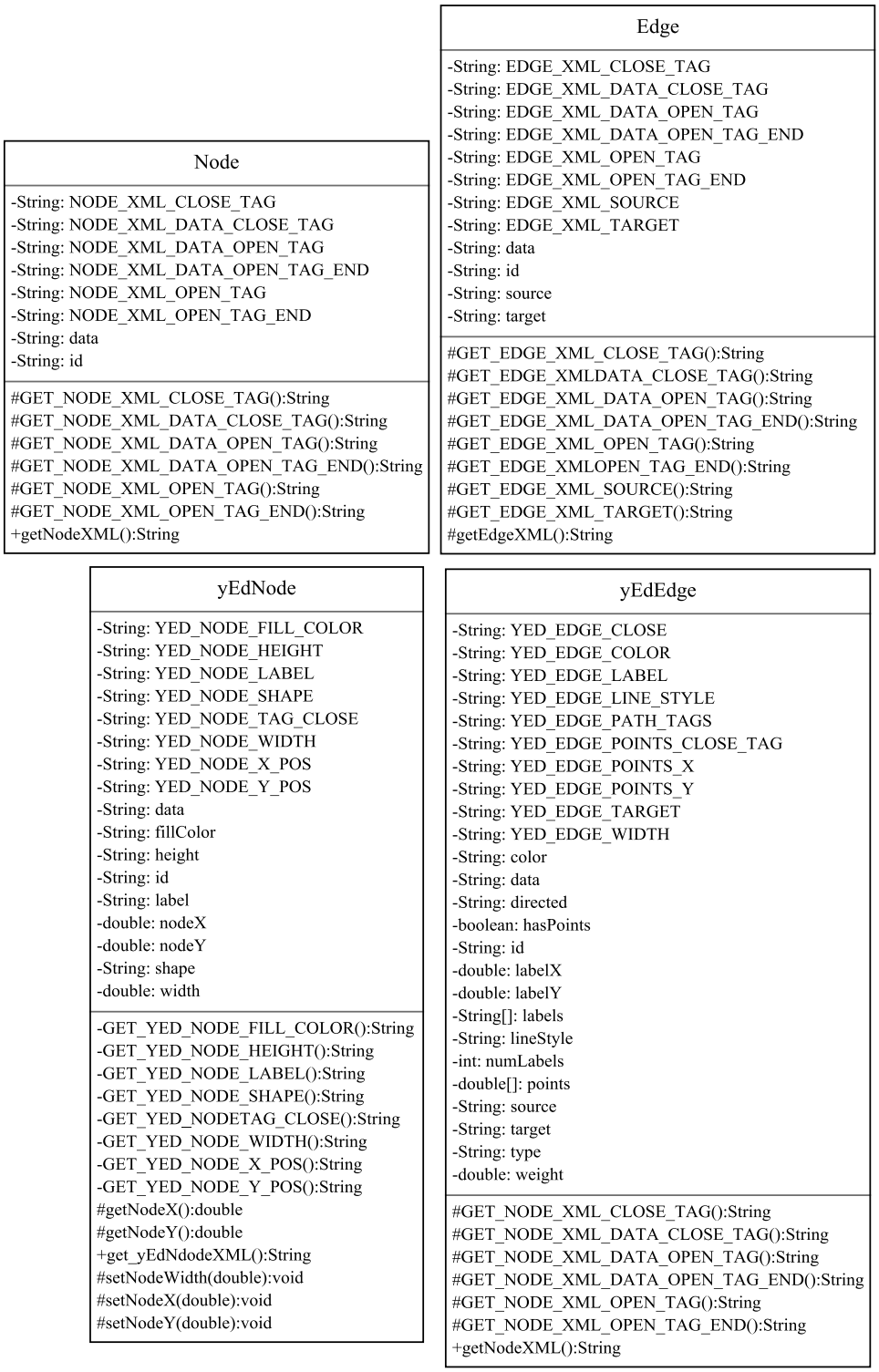

Figure 72. `Node`, `yEdNode`, `Edge` and `yEdEdge` class diagrams

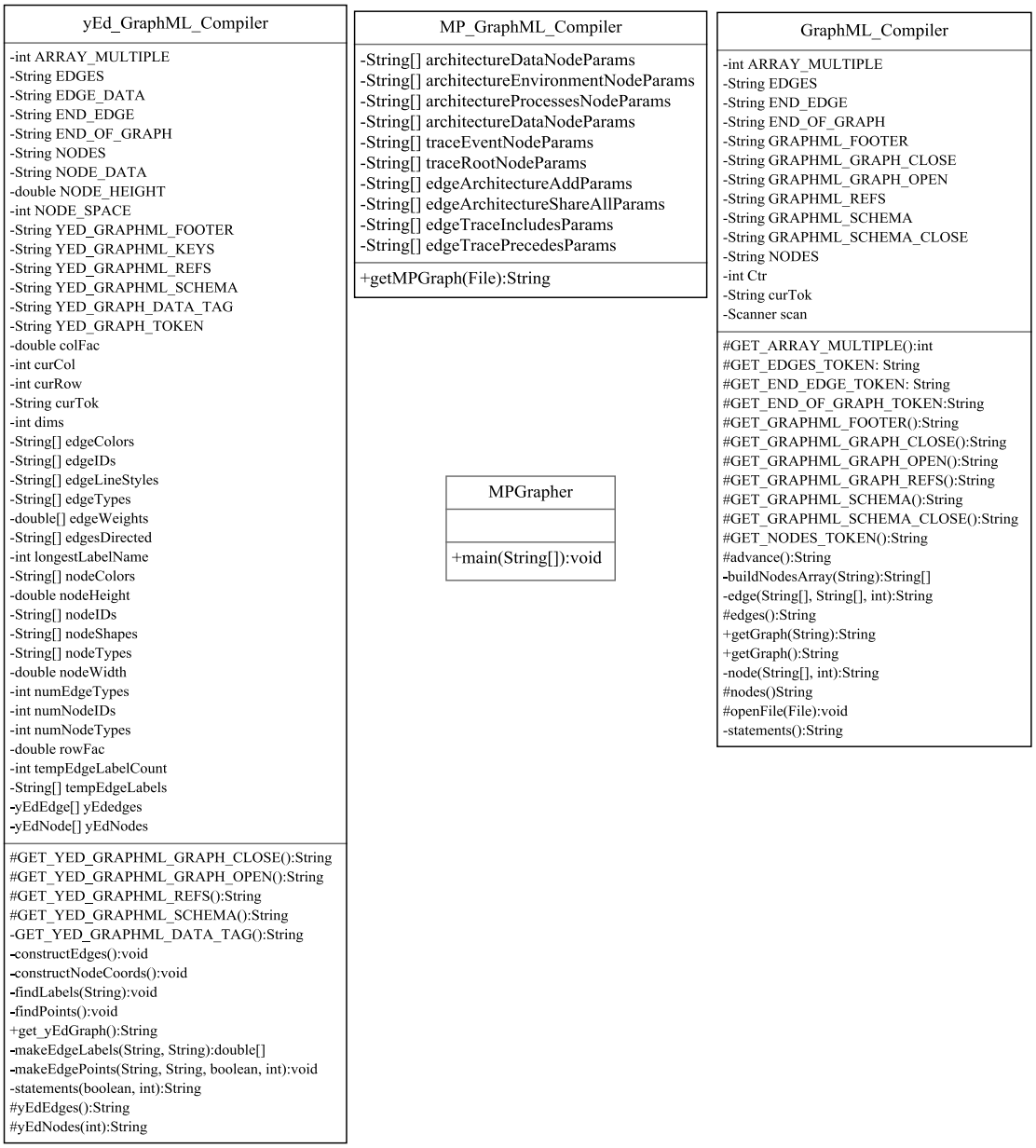

Figure 73. Compilers and `MPGrapher` class diagrams

APPENDIX J. MPGRAPHER OUTPUT FILES

A. LAYOUTTESTER_MPOUTPUT.GRAPHML

```
<?xml version="1.0"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:y="http://www.yworks.com/xml/graphml"
  xmlns:yed="http://www.yworks.com/xml/yed/3"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd
  http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
  <key for="graphml" id="d0" yfiles.type="resources"/>
  <key for="port" id="d1" yfiles.type="portgraphics"/>
  <key for="port" id="d2" yfiles.type="portgeometry"/>
  <key for="port" id="d3" yfiles.type="portuserdata"/>
  <key attr.name="url" attr.type="string" for="node" id="d4"/>
  <key attr.name="description" attr.type="string" for="node" id="d5"/>
  <key for="node" id="d6" yfiles.type="nodegraphics"/>
  <key attr.name="url" attr.type="string" for="edge" id="d7"/>
  <key attr.name="description" attr.type="string" for="edge" id="d8"/>
  <key for="edge" id="d9" yfiles.type="edgegraphics"/>
  <key attr.name="Description" attr.type="string" for="graph" id="d10"/>
  <graph edgedefault="directed" id="G">
    <data key="d10"/>
    <node id="1-1">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="24.0" x="0.0" y="0.0"/>
          <y:Fill color="#CCC1DA" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            1-1
            <y:LabelModel>
              <y:SmartNodeLabelModel distance="4.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
            </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="roundrectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="2-1">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="24.0" x="150.0" y="75.0"/>
          <y:Fill color="#CCC1DA" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
```

```
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           2-1
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="2-2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="225.0" y="249.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           2-2
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="1-2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="75.0" y="174.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           1-2
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
```

```
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="3-1">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="300.0" y="0.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          3-1
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="3-2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="375.0" y="174.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          3-2
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="3-3">
```

```
     <data key="d5"/>
     <data key="d6">
       <y:ShapeNode>
         <y:Geometry height="30.0" width="24.0" x="300.0" y="348.0"/>
         <y:Fill color="#CCC1DA" transparent="false"/>
         <y:BorderStyle color="#000000" type="line" width="1.0"/>
         <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           3-3
           <y:LabelModel>
             <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
           </y:ModelParameter>
         </y:NodeLabel>
         <y:Shape type="roundrectangle"/>
       </y:ShapeNode>
     </data>
   </node>
   <node id="2-3">
     <data key="d5"/>
     <data key="d6">
       <y:ShapeNode>
         <y:Geometry height="30.0" width="24.0" x="150.0" y="423.0"/>
         <y:Fill color="#CCC1DA" transparent="false"/>
         <y:BorderStyle color="#000000" type="line" width="1.0"/>
         <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           2-3
           <y:LabelModel>
             <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
           </y:ModelParameter>
         </y:NodeLabel>
         <y:Shape type="roundrectangle"/>
       </y:ShapeNode>
     </data>
   </node>
   <node id="1-3">
     <data key="d5"/>
     <data key="d6">
       <y:ShapeNode>
         <y:Geometry height="30.0" width="24.0" x="0.0" y="348.0"/>
         <y:Fill color="#CCC1DA" transparent="false"/>
         <y:BorderStyle color="#000000" type="line" width="1.0"/>
         <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
```

```
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            1-3
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="4-1">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="450.0" y="75.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            4-1
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="4-2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="525.0" y="249.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            4-2
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
```

```
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="4-3">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="450.0" y="423.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          4-3
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="4-4">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="525.0" y="597.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          4-4
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="3-4">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
```

```
        <y:Geometry height="30.0" width="24.0" x="375.0" y="522.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           3-4
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="2-4">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="225.0" y="597.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           2-4
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="1-4">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="24.0" x="75.0" y="522.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           1-4
          <y:LabelModel>
```

```
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <edge id="1-1::1-2" source="1-1" target="1-2">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="1-1::2-1" source="1-1" target="2-1">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
```

```
    <edge id="1-2::1-3" source="1-2" target="1-3">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="1-2::2-2" source="1-2" target="2-2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="1-3::1-4" source="1-3" target="1-4">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
```

```
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="1-3::2-3" source="1-3" target="2-3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="1-4::2-4" source="1-4" target="2-4">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
```

```
    </edge>
    <edge id="2-1::2-2" source="2-1" target="2-2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="2-1::3-1" source="2-1" target="3-1">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="2-2::2-3" source="2-2" target="2-3">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
```

```
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="2-2::3-2" source="2-2" target="3-2">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="2-3::2-4" source="2-3" target="2-4">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
```

```
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="2-3::3-3" source="2-3" target="3-3">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="2-4::3-4" source="2-4" target="3-4">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="3-1::3-2" source="3-1" target="3-2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
```

```
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="3-1::4-1" source="3-1" target="4-1">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="3-2::3-3" source="3-2" target="3-3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
```

```
        </y:EdgeLabel>
       <y:BendStyle smoothed="false"/>
     </y:QuadEdgeCurve>
     </data>
   </edge>
   <edge id="3-2::4-2" source="3-2" target="4-2">
     <data key="d8"/>
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
         <y:Arrows source="none" target="standard"/>
         <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
           <y:LabelModel>
             <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
           </y:ModelParameter>
         </y:EdgeLabel>
       <y:BendStyle smoothed="false"/>
     </y:QuadEdgeCurve>
     </data>
   </edge>
   <edge id="3-3::3-4" source="3-3" target="3-4">
     <data key="d8"/>
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#00FF00" type="line" width="4.0"/>
         <y:Arrows source="none" target="standard"/>
         <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
           <y:LabelModel>
             <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
           </y:ModelParameter>
         </y:EdgeLabel>
       <y:BendStyle smoothed="false"/>
     </y:QuadEdgeCurve>
     </data>
   </edge>
   <edge id="3-3::4-3" source="3-3" target="4-3">
     <data key="d8"/>
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
```

```
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="3-4::4-4" source="3-4" target="4-4">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="4-1::4-2" source="4-1" target="4-2">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
```

```
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="4-2::4-3" source="4-2" target="4-3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="4-3::4-4" source="4-3" target="4-4">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
 </graph>
 <data key="d0">
   <y:Resources/>
 </data>
</graphml>
```

B. EDGEPOINTTESTER_MPOUTPUT.GRAPHML

```
<?xml version="1.0"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:y="http://www.yworks.com/xml/graphml"
  xmlns:yed="http://www.yworks.com/xml/yed/3"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd
  http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
  <key for="graphml" id="d0" yfiles.type="resources"/>
  <key for="port" id="d1" yfiles.type="portgraphics"/>
  <key for="port" id="d2" yfiles.type="portgeometry"/>
  <key for="port" id="d3" yfiles.type="portuserdata"/>
  <key attr.name="url" attr.type="string" for="node" id="d4"/>
  <key attr.name="description" attr.type="string" for="node" id="d5"/>
  <key for="node" id="d6" yfiles.type="nodegraphics"/>
  <key attr.name="url" attr.type="string" for="edge" id="d7"/>
  <key attr.name="description" attr.type="string" for="edge" id="d8"/>
  <key for="edge" id="d9" yfiles.type="edgegraphics"/>
  <key attr.name="Description" attr.type="string" for="graph" id="d10"/>
  <graph edgedefault="directed" id="G">
    <data key="d10"/>
    <node id="node1">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="48.0" x="0.0" y="0.0"/>
          <y:Fill color="#CCC1DA" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            node 1
           <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
           </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="roundrectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="node2">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="48.0" x="151.0" y="75.0"/>
          <y:Fill color="#CCC1DA" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
```

```
            node 2
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node3">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="225.0" y="273.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            node 3
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node4">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="75.0" y="198.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            node 4
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
```

```
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node5">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="301.0" y="0.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          node 5
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node6">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="376.0" y="198.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          node 6
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node7">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="300.0" y="396.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
```

```
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           node 7
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node8">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="150.0" y="471.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           node 8
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="node9">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="48.0" x="0.0" y="396.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           node 9
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
```

```
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <edge id="node1::node5" source="node1" target="node5">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          <y:Point x="24.0" y="-25.0"/>
          <y:Point x="325.0" y="-25.0"/>
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="node1::node9" source="node1" target="node9">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          <y:Point x="-25.0" y="15.0"/>
          <y:Point x="-25.0" y="411.0"/>
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
```

```
      </data>
    </edge>
    <edge id="node2::node7" source="node2" target="node7">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
            <y:Point x="175.0" y="50.0"/>
            <y:Point x="324.0" y="371.0"/>
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="true"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="node5::node1" source="node5" target="node1">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
            <y:Point x="325.0" y="55.0"/>
            <y:Point x="24.0" y="55.0"/>
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="true"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="node7::node2" source="node7" target="node2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
            <y:Point x="324.0" y="451.0"/>
```

```
          <y:Point x="175.0" y="130.0"/>
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="node9::node1" source="node9" target="node1">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          <y:Point x="73.0" y="411.0"/>
          <y:Point x="73.0" y="15.0"/>
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="396.0" y="396.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
 </graph>
 <data key="d0">
   <y:Resources/>
 </data>
</graphml>
```

C. SAMPLEARCHITECTURE_MPOUTPUT.GRAPHML

```
<?xml version="1.0"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:y="http://www.yworks.com/xml/graphml"
  xmlns:yed="http://www.yworks.com/xml/yed/3"
```

```
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd
  http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
  <key for="graphml" id="d0" yfiles.type="resources"/>
  <key for="port" id="d1" yfiles.type="portgraphics"/>
  <key for="port" id="d2" yfiles.type="portgeometry"/>
  <key for="port" id="d3" yfiles.type="portuserdata"/>
  <key attr.name="url" attr.type="string" for="node" id="d4"/>
  <key attr.name="description" attr.type="string" for="node" id="d5"/>
  <key for="node" id="d6" yfiles.type="nodegraphics"/>
  <key attr.name="url" attr.type="string" for="edge" id="d7"/>
  <key attr.name="description" attr.type="string" for="edge" id="d8"/>
  <key for="edge" id="d9" yfiles.type="edgegraphics"/>
  <key attr.name="Description" attr.type="string" for="graph" id="d10"/>
  <graph edgedefault="directed" id="G">
    <data key="d10"/>
    <node id="Customer">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="80.0" x="0.0" y="0.0"/>
          <y:Fill color="#FCD5B5" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            Customer
           <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
           </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="roundrectangle"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="ATM_system">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="80.0" x="150.0" y="75.0"/>
          <y:Fill color="#D7E4BD" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            ATM System
           <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
```

```
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="Data_base">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="80.0" x="225.0" y="305.0"/>
        <y:Fill color="#CCC1DA" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          Data Base
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <edge id="ATM_system::Customer" source="ATM_system" target="Customer">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          <y:Point x="190.0" y="50.0"/>
          <y:Point x="40.0" y="-25.0"/>
        </y:Path>
        <y:LineStyle color="#00FF00" type="line" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          dispense_money>>get_money
id_successful>>identification_succeeds
insuficient_balance>>not_sufficient_funds
id_failed>>identification_fails
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="false"/>
    </y:QuadEdgeCurve>
    </data>
```

```
    </edge>
    <edge id="ATM_system::Data_base" source="ATM_system" target="Data_base">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#FF99CC" type="line" width="4.0"/>
          <y:Arrows source="none" target="none"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            validate_id
check_balance
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="Customer::ATM_system" source="Customer" target="ATM_system">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
            <y:Point x="40.0" y="55.0"/>
            <y:Point x="190.0" y="130.0"/>
          </y:Path>
          <y:LineStyle color="#00FF00" type="line" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            smart_card>>read_card
request_withdrawal>>check_balance
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
        <y:BendStyle smoothed="false"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
  </graph>
  <data key="d0">
    <y:Resources/>
  </data>
</graphml>
```

D. SAMPLETRACE_MPOUTPUT.GRAPHML

```
<?xml version="1.0"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:y="http://www.yworks.com/xml/graphml"
  xmlns:yed="http://www.yworks.com/xml/yed/3"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
  http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd
  http://www.yworks.com/xml/schema/graphml/1.1/ygraphml.xsd">
  <key for="graphml" id="d0" yfiles.type="resources"/>
  <key for="port" id="d1" yfiles.type="portgraphics"/>
  <key for="port" id="d2" yfiles.type="portgeometry"/>
  <key for="port" id="d3" yfiles.type="portuserdata"/>
  <key attr.name="url" attr.type="string" for="node" id="d4"/>
  <key attr.name="description" attr.type="string" for="node" id="d5"/>
  <key for="node" id="d6" yfiles.type="nodegraphics"/>
  <key attr.name="url" attr.type="string" for="edge" id="d7"/>
  <key attr.name="description" attr.type="string" for="edge" id="d8"/>
  <key for="edge" id="d9" yfiles.type="edgegraphics"/>
  <key attr.name="Description" attr.type="string" for="graph" id="d10"/>
  <graph edgedefault="directed" id="G">
    <data key="d10"/>
    <node id="Task_A">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="56.0" x="0.0" y="0.0"/>
          <y:Fill color="#D7E4BD" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            Task A
           <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
           </y:LabelModel>
           <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
           </y:ModelParameter>
          </y:NodeLabel>
          <y:Shape type="hexagon"/>
        </y:ShapeNode>
      </data>
    </node>
    <node id="Task_B">
      <data key="d5"/>
      <data key="d6">
        <y:ShapeNode>
          <y:Geometry height="30.0" width="56.0" x="150.0" y="75.0"/>
          <y:Fill color="#D7E4BD" transparent="false"/>
          <y:BorderStyle color="#000000" type="line" width="1.0"/>
          <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
```

```
            Task B
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="hexagon"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="send1">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="225.0" y="281.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            Send
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="send2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="75.0" y="206.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
            Send
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
```

```
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="send3">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="300.0" y="0.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          Send
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="receive1">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="375.0" y="206.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
          Receive
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="receive2">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="300.0" y="412.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
```

```
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           Receive
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <node id="receive3">
    <data key="d5"/>
    <data key="d6">
      <y:ShapeNode>
        <y:Geometry height="30.0" width="56.0" x="150.0" y="487.0"/>
        <y:Fill color="#FFFF99" transparent="false"/>
        <y:BorderStyle color="#000000" type="line" width="1.0"/>
        <y:NodeLabel alignment="center" autoSizePolicy="content"
fontFamily="Courier New" fontSize="12" fontStyle="plain"
hasBackgroundColor="false" hasLineColor="false" height="44.78125"
modelName="custom" textColor="#000000" visible="true" width="61.609375"
x="9.1953125" y="-7.390625">
           Receive
          <y:LabelModel>
            <y:SmartNodeLabelModel distance="4.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartNodeLabelModelParameter labelRatioX="0.0" labelRatioY="0.0"
nodeRatioX="0.0" nodeRatioY="0.0" offsetX="0.0" offsetY="0.0" upX="0.0" upY="-
1.0"/>
          </y:ModelParameter>
        </y:NodeLabel>
        <y:Shape type="roundrectangle"/>
      </y:ShapeNode>
    </data>
  </node>
  <edge id="Task_A::send1" source="Task_A" target="send1">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#000000" type="line" width="1.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">

          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
```

```
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="Task_A::send2" source="Task_A" target="send2">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#000000" type="line" width="1.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="Task_A::send3" source="Task_A" target="send3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#000000" type="line" width="1.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="Task_B::receive1" source="Task_B" target="receive1">
    <data key="d8"/>
```

```
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#000000" type="line" width="1.0"/>
         <y:Arrows source="none" target="standard"/>
         <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
           <y:LabelModel>
             <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
           </y:ModelParameter>
         </y:EdgeLabel>
       <y:BendStyle smoothed="true"/>
     </y:QuadEdgeCurve>
     </data>
   </edge>
   <edge id="Task_B::receive2" source="Task_B" target="receive2">
     <data key="d8"/>
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#000000" type="line" width="1.0"/>
         <y:Arrows source="none" target="standard"/>
         <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
           <y:LabelModel>
             <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
           </y:LabelModel>
           <y:ModelParameter>
             <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
           </y:ModelParameter>
         </y:EdgeLabel>
       <y:BendStyle smoothed="true"/>
     </y:QuadEdgeCurve>
     </data>
   </edge>
   <edge id="Task_B::receive3" source="Task_B" target="receive3">
     <data key="d8"/>
     <data key="d9">
       <y:QuadEdgeCurve>
         <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
         </y:Path>
         <y:LineStyle color="#000000" type="line" width="1.0"/>
         <y:Arrows source="none" target="standard"/>
         <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
           <y:LabelModel>
```

```
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="receive1::receive2" source="receive1" target="receive2">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="receive2::receive3" source="receive2" target="receive3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
```

```
    <edge id="send1::receive1" source="send1" target="receive1">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="true"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="send1::send2" source="send1" target="send2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
            <y:LabelModel>
              <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
            </y:LabelModel>
            <y:ModelParameter>
              <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
            </y:ModelParameter>
          </y:EdgeLabel>
        <y:BendStyle smoothed="true"/>
      </y:QuadEdgeCurve>
      </data>
    </edge>
    <edge id="send2::receive2" source="send2" target="receive2">
      <data key="d8"/>
      <data key="d9">
        <y:QuadEdgeCurve>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
          </y:Path>
          <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
          <y:Arrows source="none" target="standard"/>
          <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
```

```
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="send2::send3" source="send2" target="send3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
  </edge>
  <edge id="send3::receive3" source="send3" target="receive3">
    <data key="d8"/>
    <data key="d9">
      <y:QuadEdgeCurve>
        <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0">
        </y:Path>
        <y:LineStyle color="#00FF00" type="dashed" width="4.0"/>
        <y:Arrows source="none" target="standard"/>
        <y:EdgeLabel alignment="center" distance="2.0" fontFamily="Courier New"
fontSize="12" fontStyle="plain" hasBackgroundColor="false" hasLineColor="false"
height="34.0" modelName="custom" preferredPlacement="anywhere" ratio="0.5"
textColor="#000000" visible="true" width="40.0" x="0.0" y="0.0">
          <y:LabelModel>
            <y:SmartEdgeLabelModel autoRotationEnabled="false"
defaultAngle="0.0" defaultDistance="10.0"/>
          </y:LabelModel>
          <y:ModelParameter>
            <y:SmartEdgeLabelModelParameter angle="0.0" distance="30.0"
distanceToCenter="true" position="right" ratio="0.5" segment="0"/>
          </y:ModelParameter>
        </y:EdgeLabel>
      <y:BendStyle smoothed="true"/>
    </y:QuadEdgeCurve>
    </data>
```

```
    </edge>
  </graph>
  <data key="d0">
    <y:Resources/>
  </data>
</graphml>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Aho, A., Garey, M., & Ullman, J. (1972). The transitive reduction of a directed graph. *Society for Industrial and Applied Mathematics (SIAM)*, 131-137.

AlternativeTo contributors. (2012, August 30). *yEd graph editor alternatives and similar software*. Retrieved from AlternativeTo: http://alternativeto.net/software/yed/

Auguston, M., & Whitcomb, C. (2012). Behavior models and composition for software and systems architecture. *ICSSEA*, 1-15.

Batini, C., Talamo, M., & Tamassia, R. (1984). Computer aided layout of entity relationship diagrams. *Journal of Sytems and Software*, 163-173.

Berry, D. (1999). Formal methods: the very idea, some thoughts about why they work when they work. Waterloo, Ontario, Canada: Computer Science Department, University of Waterloo.

Binder, R. (2000). *Testing Object-Oriented Systems.* Addison-Wesley.

Bowen, J., & Hinchey, M. (1995). Seven more myths of formal methods. *Software, IEEE*, 34-41.

Brandes, U., & Pich, C. (2005). GraphML transformation. *Lecture Notes in Computer Science*, 89-99.

Chimani, M., & Gutwenget, C. (2011). Advances in the planarization method: effective multiple edge insertions. *Proceedings of the 19th International Conference on Graph Drawing*, 87-98.

Chimani, M., Gutwenger, C., Mutzel, P., & Wolf, C. (2009). Inserting a vertex into a planar graph. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA09)*, 375-383.

Chuzhoy, J., Makarychev, Y., & Sidiropoulos, A. (2011). On graph crossing number and edge planarization. *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1050-1069.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms - 3rd Edition.* Cambridge, Massachusetts: The MIT Press.

EETimes website authors. (2011, 8 30). *SysML - The Systems Modeling Language*. Retrieved from EETimes: http://www.eetimes.com/design/embedded/4006704/SysML--The-Systems-Modeling-Language

FormalMind website authors. (2011, 12 21). *Formal Mind - Requirements + UML = SysML*. Retrieved from FormalMind: Science for Systems Engineering: http://www.formalmind.com/en/blog/requirements-uml-sysml

GraphML Working Group. (2009, December 15). *GraphML Specification*. Retrieved February 28, 2012, from graphdrawing.org: http://graphml.graphdrawing.org/specification.html

Gutwenger, C., Mutzel, P., & Weiskircher, R. (2005). Inserting an edge into a planar graph. *Algorithmica*, 289-308.

Hachul, S., & Jünger, M. (2005). Drawing large graphs with a potential-field-based multilevel algorithm. *Lecture Notes in Computer Science*, 285-295.

Hall, A. (1990). Seven myths of formal methods. *IEEE Software*, 11-19.

Heinen, O. (2008, February 15). *GXL - Graph eXchange Language*. Retrieved from GXL Introduction: http://www.gupro.de/GXL/Introduction/background.html

Himsolt, M. (1995). GML: A Portable Graph File Format. Passau, Germany: Universität Paßau.

Hopcroft, J., Motwani, R., & Ullman, J. (2006). *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley.

Hunter, D., Rafter, J., Fawcett, J., Van der Vlist, E., Ayers, D., Duckett, J., . . . McKinnon, L. (2007). *Beginning XML, 4th edition.* Indianapolis: Wrox Publishers.

Huth, M., & Ryan, M. (2004). *Logic in Computer Science.* Cambridge University Press.

Jackson, D. (2006). *Software Abstractions.* The MIT Press.

JPMensah website authors. (2005, 2 7). *ITEC485 Template*. Retrieved from JPMensah: http://www.jpmensah.com/ITEC485/matrix.htm

Levine, J. (2009). *Flex & Bison: Text Processing Tools.* O'Reilly.

Louden, K. (1997). *Compiler Construction.* PWS Publishing Company.

Rosen, K. (2007). *Discrete Mathematics and Its Application.* McGraw Hill.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 146-160.

Taylor, R., Medvidovic, N., & Dashofy, E. (2009). *Software Architecture: Foundations, Theory, and Practice.* Wiley.

Wikipedia contributors. (2009, May 14). *XGMML*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=XGMML&oldid=289822037

Wikipedia contributors. (2012, 7 17). *Business Process Model and Notation*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Business_Process_Model_and_Notation&oldid=502708942

Wikipedia contributors. (2012, August 24). *Coulomb's law*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Coulomb%27s_law&oldid=508929239

Wikipedia contributors. (2012, August 19). *DOT language*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=DOT_language&oldid=508100908

Wikipedia contributors. (2012, July 20). *Graph Modelling Language*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Graph_Modelling_Language&oldid=503323864

Wikipedia contributors. (2012, August 20). *Hooke's law*. Retrieved from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/w/index.php?title=Hooke%27s_law&oldid=508244019

Wikipedia contributors. (2012, May 30). *SPQR tree*. Retrieved May 30, 2012, from Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/SPQR_tree

Wu, C. (2007). *A Comprehensive Introduction to Object-Oriented Programming with Java.* McGraw-Hill.

Yourdon website authors. (2012, 5 30). *Chapter 9 - Structured Analysis Wiki:*. Retrieved from yourdon: http://yourdon.com/strucanalysis/wiki/index.php?title=Chapter_9

Ziegler, T. (2001). Crossing minimization in automatic graph drawing. *(PhD thesis, Saarland University)*. Germany.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California

4. Director, Training and Education, MCCDC, Code C46
   Quantico, Virginia

5. Director, Marine Corps Research Center, MCCDC, Code C40RC
   Quantico, Virginia

6. Marine Corps Tactical Systems Support Activity (Attn: Operations Officer)
   Camp Pendleton, California

7. Auguston, Mikhail
   Naval Postgraduate School
   Monterey, California

8. Norbraten, Terry
   Naval Postgraduate School
   Monterey, California