5-1-2003

CK Metrics for DAML Documents

S. Daniel Daugherty II

UNIVERSITY OF ALBAMA IN HUNTSVILLE

CK Metrics for DAML Documents

A Thesis submitted in partial satisfaction of
the requirements
of the Honor's Program for the degree of

Batchelor of Science

in

Computer Science

by

S. Daniel Daugherty II

May 2003

ABSTRACT OF THE THESIS

CK Metrics for DAML Documents

by

S. Daniel Daugherty II

Batchelor of Science, Undergraduate Program in Computer Science

University of Alabama in Huntsville, May 2003

Professor Anthony Mark Orme, Advisor

The purpose of my honors senior project will be to work on the development of a tool to collect object oriented metrics from DARPA Agent Markup Language + Ontology Inference Layer (DAML+OIL) documents. Throughout the project I will be overseeing the development a graphical user interface that will allow a user to specify the URI of a DAML document and then compute the class-oriented metrics that are similar to the MOOD Suite and CK Metrics Suite. Having this tool that specifically focuses on cohesion, coupling, and complexity of ontologies in DAML will allow future researches to be able to answer many questions currently unanswered. How many relations are there in connected graphs of ontologies? Are all ontologies completely connected? How cohesive are ontologies in DAML? And what is the coupling between ontologies in DAML?

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to describe the architecture and design of the DAML metric system. This document will describe the purposed module breakdown of the DMS software product. The relationships of the modules to outside data sources will be decomposed and described at the functional level. Also, the relationships between the modules will be described.

This document will follow from the software requirements document that was composed during the requirements phase of this project. The modules and data structures will be mapped into the data flow diagrams of the requirements documentation.

1.2 Scope

The scope of this document is the breakdown of the DAML system into an object and functional description. All data structures that pass into the system, function within the system and are passed out of the system will be generally detailed. As implied the use of a module breakdown will identify the design entities of the DMS software. This document will describe the internal aspects of each design entity. A description will be provided for how the design entities internally process the data structures of the DMS system. Also, this document will provide a general description of the algorithms that will be used to provide a solution to each design entities task.

The description of the graphical user interface will be provided as prescribed in the requirements documentation. The interface between the internal design entities and the GUI will be explicitly detailed.

1.3 Definitions, acronyms, and abbreviations

DARPA	Defense Advanced Research Projects Agency
DAML	DARPA Agent Markup Language
DAML+OIL	DAML's Current Markup Language
GUI	Graphical user interface
OIL	Ontology Inference Layer
URL	Uniform Resource Locator - a link to a specific entity or location on the Web
URI	Uniform Resource Identifier - similar to, but more general than a URL
RDF	Resource Description Framework - a statement about URI entities that often expresses a relationship between them
HTML	Hypertext Markup Language
XML	Extensible Markup Language - a coding system that isolates data elements in a Web page

2. Overall description

2.1 Product perspective

The DMS is a self-contained, stand-alone software product, which is comprised of a single GUI with highly limited user options. The DMS provides a software tool for generating object-oriented metrics (i.e., CK and MOOD metrics) for analyzing Web content (specifically, DAML+OIL coded Web content). The DMS displays metrics data associated with objects within a given DAML document in a treelike structure similar to the Microsoft's Windows file manager format. The tree can be

expanded at specific nodes to reveal the specific classes encountered in the analysis and parsing of the DAML documents.

2.2 Product functions

The primary function of the DMS software is to read and parse DAML documents, which is initiated by the URI input provided by the user. The URI may be selected from a drop down menu (for instance, in the event that it references a DAML document on the local system) or input manually by the user (similar to using a Web browser. The system analyzes each of the DAML documents it encounters, building the treelike hierarchy of nodes, and then generates appropriate and corresponding metric data associated with each class inserted as a node in the tree.

2.3 User characteristics

The intended user of the DMS software is assumed to be familiar with Microsoft's Windows Operating Systems, as well as Internet Explorer. It is also assumed that the user has at least a basic understanding of the DAML+OIL coding scheme, as well as the specific metrics (CK and MOOD metrics) the software generates, and has a specific application or interest in these metrics with respect to the coupling and cohesion of Web content.

2.4 Constraints

The primary constraint placed on the software is that the user only reference well-formed DAML documents as input at the user interface. It is assumed therefore that the DAML tags will adhere to the DARPA standards given in the DAML+OIL ontology markup language. Also, because Microsoft's .NET programming environment is being used to develop the DMS and relies on non-deterministic garbage collection, it may be necessary for the user to perform a fresh reboot of the software or monitor system resources with repeated use of the software over a

lengthy period of time. Although any instance of critically depleted system resources is highly unlikely as a result of standard use of this software product, it is necessary to mention the possibility of such an occurrence.

2.5 Assumptions and dependencies

It is highly recommended that the DMS software be run on the Windows 2000 operating system or more recent versions such as Windows XP. Microsoft's .NET framework should be loaded on the system running the DMS, as well as Internet Explorer (Version 6.0 is highly preferred, with Version 5.0 as a recommended minimum requirement). Also, In the event that additional input data other than DAML documents are required to be included as input for the software, the specific software requirements specified in this SRS will no longer be valid and will require appropriate revision.

3. Specific requirements

3.1 External Interface requirements

- The system should be Windows compatible and be running C#.
- The system should also have a screen resolution of at least 640x480.

3.2 Functional requirements

- The system will accept a URI as input. The URI will represent the location of the desired DAML document.
- The system will validate the URI. If the URI is determined to be invalid, the software will notify and prompt the user.
- The system will retrieve well-formed DAML documents.
- The DMS software will verify the DAML document. If there is an error in the document the system will notify and prompt the user.

- The system will parse the DAML document and calculate metrics based on coupling and cohesion between the classes.
- The metrics will be displayed to the user and the user will be able to browse the resulting data based on the specific document and the classes within the document.

3.2.1 Information flows

3.2.1.1 Context diagram

The following context diagram summarizes the overall flow of information in the DMS system.

See Figure 1: Context diagram (Level 0) of the DMS

The context diagram in Figure 1 represents the most general description of the flow of data in the DMS. The URI entered by the user references a specific, well-formed DAML document that is read, parsed, and analyzed by the DMS software. The system then generates a data tree with corresponding metrics data.

3.2.1.2 Data flow diagram 1

The following diagram represents the first level of data flow within the DMS System.

See Figure 2: Level 1 of the DMS

3.2.1.2.1 Data entities

Well-formed (i.e., correctly formatted) DAML documents are the only valid input to the DMS system. DMS output will be comprised of the data tree and the corresponding metrics that have been calculated for each node. The destination for this output will be the display window.

3.2.1.2.2 Pertinent processes

The user will enter the URI of a DAML document. The computer that the software is running on will need to be connected to a network if the DAML document is located either on the Internet or the network itself. Once the software verifies the DAML document, it will fetch and parse the document. After verifying and parsing the document, the software will calculate the metrics for the classes in the document. Once this is done the application will display the results in the client area of the application.

3.2.1.3 Data flow diagram 2

The following diagram represents the second level of data flow within the DMS System.

See Figure 4: Level 2 of the DMS

3.2.1.3.1 Data entities

Well-formed (i.e., correctly formatted) DAML documents are the only valid input to the DMS system. DMS output will be comprised of the data tree and the corresponding metrics that have

been calculated for each node. The destination for this output will be the display window.

3.2.1.3.2 Pertinent processes

The user will enter the URI of a DAML document. The computer that the software is running on will need to be connected to a network if the DAML document is located either on the Internet or the network itself. Once the software verifies the DAML document, it will fetch and parse the document.

3.2.1.4 Traceability matrix

The following traceability matrix summarizes the information of the product.

Traceability Matrix

Classes (across) / Requirements (down)	DMSMainForm	DamlReader	MetricContainer	PersistentStorageManager
Accept URI as input	X			
Validate URI		X		
Retrieve well-formed DAML document		X		
Verify well-formed DAML document		X		
Parse DAML document		X	X	
Calculate Metrics		X	X	
Display Results	X			X

4. Module Decomposition Description

The DMS system is broken down into five objects and one user defined data type. The entry point for the system is the DMSMainForm object. This is the

applications main form and is point where the user starts to enter the URI's that describes the location of the DAML documents that the metrics will be calculated on. The main form makes use of the class DMSReport. DMSreport will interface with the classes that actually do the computational work for the system. Also, DMSReport is the form where the results from the metric computation will be displayed. The rest of this section will present the module decomposition of DMS and describe the attributes and methods that will be used for this solution.

4.1 Class DMS::DMSMainForm

4.1.1 Attributes

Name: Components
Type: Container
Visibility: Private
Description: Contains references to the visual components of the main form.

Name: isMidicontainer
Type: Bool
Visibility: Private
Description: This attribute indicates if the multiple document interface functionality is enabled.

4.1.2 Methods

4.1.2.1 DMSMainForm()

Input: Void

Returns: Void

Visibility: Private

Description: This is the constructor method for this object. This method will initialize the main visual form where URI input will be accepted and user events will be processed.

4.1.2.2 InitializeComponent()

Input: Void

Returns: Void

Visibility: Private

Description: This method is responsible for ensuring that all objects are initialized to a satisfactory state when the application is started.

4.1.2.3 Main()

Input: Void

Returns: Void

Visibility: Protected

Description: This method will process the events that will occur in the application. Main will have control of the user interaction with the system. Main will also manage interaction with multiple report documents if needed.

4.1.2.4 ExecuteNewDamlQuery ()

Input: Void

Returns: Void

Visibility: Private

Description: This method will be executed by Main () and will instantiate a DMSReport object (Please see below). As stated, main will have control of the execution of this

method. The method will be started when the user clicks a button on the visual form that is controlled by DMSMainForm. The DMSReport object will be created and its DMSReport() method will be called passing it the URI that the user has input.

DMSReport will not be created if there is no URI input.

4.1.2.5 Dispose()

Input: Void

Returns: Void

Visibility: Protected

Description: This method will perform the final clean up for the application upon exiting.

4.2 Class DMS::DMSReport

4.2.1 Attributes

Name: resultsMgr

Type: PersistentStorageManager

Visibility: Private

Description: This is an instance of a PersistentStorageManger object. This is the one an only object of this type for the

application. This object will be used to place result set into a persistent storage area for future use. Also, this object will be used to read past result sets.

Name: baseURI

Type: String

Visibility: Private

Description: This is the member that holds the URI input that was input on the main form.

Name: components

Type: container

Visibility: Private

Description: Contains references to the visual components of the report form.

4.2.2 Methods

4.2.2.1 DMSReport()

Input: Void

Returns: Void

Visibility: Private

Description: Constructor that initializes an instance of this class.

4.2.2.2 InitializeComponent()

Input: Void

Returns: Void

Visibility: Private

Description: This method will initialize the reports visual form

4.2.2.3 CheckForPreviousResults()

Input: Void

Returns: Bool

Visibility: Private

Description: This method will use the PersistentStorageManager object to check the storage location for results that have been gathered in relation to the URI contained in the baseURI attribute.

4.2.2.4 DisplayResults()

Input: Void

Returns: Void

Visibility: Private

Description: This method will display results that are gathered from the persistent storage lookup or the GetStatics() method call for a given DAML document at the specified URI in the baseURI attribute.

4.2.2.5 GetStatistics()

Input: Void

Returns: Bool

Visibility: Private

Description: This method will instantiate a DamlReader object. The DamlReader object will work to parse the DAML documents and use the metric container to calculate the desired metric for the objects in the daml documents.

4.2.2.6 Dispose()

Input: Void

Returns: Void

Visibility: Protected

Description: This method is will perform the final clean up for this object instance

4.3 Class DMS::PersistentStorageManager

4.3.1 Attributes

Name: storageMgr

Type: PersistentStorageManager

Visibility: Private (for instances of the implementing class)

Description: This attribute is the member for this class that holds a reference to the PersistentStorageManager instantiated in the DMS::DMSReport.

4.3.2 Methods

4.3.2.1 PersistentStorageManager()

Input: void

Returns: void

Visibility: Public (for all objects within the system)

Description: Constructor function that instantiates and initializes the class.

4.3.2.2 DoesResultSetExist()

Input: baseURI as a string

Returns: Boolean value

Visibility: Public (for all objects within the system)

Description: Returns a Boolean value that indicates that a set of results already exists.

4.3.2.3 SaveResultSet()

Input: baseURI as a string and resultSet as a DataSet

Returns: Boolean value

Visibility: Public (for all objects within the system)

Description: Returns a Boolean value that indicates that the set of results was successfully saved.

4.3.2.4 GetResultSet()

Input: baseURI as a string

Returns: DataSet

Visibility: Public (for all objects within the system)

Description: Returns a DataSet according to the given input search parameters.

4.3.2.4 PersistResultSetToDisk()

Input: resultSet as DataSet

Returns: Boolean value

Visibility: Private (for instances of the implementing class)

Description: Writes the results associated with the PersistentStorageManager to disk.

4.3.2.5 GetPersistentStoragePath

Input: void

Returns: string

Visibility: Private (for instances of the implementing class)

Description: Returns a string that represents the path to the PersistentStorageManager.

4.4 Class DMS:: DamlReader

4.4.1 Attributes

Name: uriBrowser

Type: WebClient

Visibility: Private (for instances of the implementing class)

Description: This attribute is an instance of the webclient object that will be used to retrieve the daml documents

Name: uriData

Type: Stream

Visibility: Private (for instances of the implementing class)

Description: This attribute will used by for document parsing.

4.4.2 Methods

4.4.2.1 DamlReader()

Input: void

Returns: void

Visibility: Public (for all objects within the system)

Description: Constructor function that initializes the class.

4.4.2.2 Read()

Input: URI as string

Returns: Boolean value

Visibility: Public (for all objects within the system)

Description: Reads DAML documents line by line and returns TRUE if the document was successfully read.

4.4.2.3 GetMetric

Input: metricType as MetricTypes

Returns: DataSet

Visibility: Public (for all objects within the system)

Description: Gets the specific metrics data types to be used in the DAML analysis. This class is dependent on the enumeration associated with the metricTypes class (which returns 0 for coupling and 1 for cohesion, as s elected by the user at the GUI).

4.5 Class DMS:: MetricContainer

4.5.1 Attributes

Name: baseURI

Type: String

Visibility: Private

Description: This attribute holds the original URI.

Name: uriData

Type: Stream

Visibility: Private

Description: This attribute holds the formatted DAML file data.

4.5.2 Methods

4.5.2.1 MetricContainer()

Input: void

Returns: void

Visibility: public

Description: Constructor function that initializes the class.

4.5.2.2 Calculate()

Input: void

Returns: void

Visibility: public

Description: This function is called and to start calculating the the desired metric. The method will instantiate and use the metric objects DMS::Coupling and DMS::Cohesion. This function passes the uriData attribute to the metric classes.

4.6 Class DMS:: Cohesion

4.6.1 Attributes

Name: uriData

Type: Stream

Visibility: Private

Description: This attribute holds the formatted DAML file data.

4.6.2 Methods

4.6.2.1 Cohesion

Input: void

Returns: void

Visibility: public

Description: Constructor function that initializes the class.

4.6.2.2 ComputeCohesion

Input: void

Returns: void

Visibility: public

Description: This method computes the cohesion metric for the information contained in the uriData attribute.

4.7 Class DMS:: Coupling

4.7.1 Attributes

Name: uriData

Type: Stream

Visibility: Private

Description: This attribute holds the formatted DAML file data.

4.7.2 Methods

4.7.2.1Coupling

Input: void

Returns: void

Visibility: public

Description: Constructor function that initializes the class.

4.7.2.2 ComputeCoupling

Input: void

Returns: void

Visibility: public

Description: This method computes the coupling metric for the information contained in the uriData attribute.

4.8 Enumeration DMS:: metricTypes

4.8.1 Attributes

Name: Coupling

Type: Integer

Value: 0

Description: Specifies to compute the metric based on coupling

Name: Cohesion

Type: Integer

Value: 1

Description: Specifies to compute the metric based on cohesion

5 Dependency Description

5.1 Internal entity dependency

5.1.1 Class DMS::DMSMainForm

- The method Main() is dependent on the Components attribute. Main() is responsible for processing events that occur from user interaction with objects on the applications main form.

5.1.2 Class DMS::DMSReport

- The method CheckForPreviousResults() is dependent on the resultsMgr and baseURI attributes.
- The method DisplayResults() is dependent on the results obtained from the call to the method GetStatistics().

5.1.3 Class DMS::PersistentStorageManager

- The method GetResultSet() is dependent on the results from the call to the method GetPersistentStoragePath().

5.1.4 Class DMS:: DamlReader

- The method Read() is dependent on the uriBrowser attribute.
- The method GetMetric() is dependent on the operations of the method Read()
- The attribute uriData is filled by the operations of the read() method.

5.1.5 Class DMS:: MetricContainer

- The Calculate() method is dependent on the data contained in the uriData attribute.

5.1.6 Class DMS:: Cohesion

- The method ComputeCohesion() is dependent on the uriData attribute.

5.1.7 Class DMS:: Coupling

- The method ComputeCoupling() is dependent on the uriData attribute.

5.1.8 Enumeration DMS:: metricTypes

There are no internal dependencies for this type.

5.2) Class Dependency

5.2.1 Class DMS::DMSMainForm

- DMSMainForm is dependent on the DMSReport class. An instance of the DMSReport class is instantiated in the execution of the DMSMainForm ::ExecuteNewDamlQuery() method.

5.2.2 Class DMS::DMSReport

- This class is dependent on the use of the PersistentStorageManager class. The DMSReport::CheckForPreviousResults() method uses the resultsMgr attribute to check for previous results based on the baseURI attribute.
- The DMSReport:: GetStatistics() method uses a DAMLReader object.

5.2.3 Class DMS::PersistentStorageManager

- This class is only dependent on the DMSReport::baseURI attribute.

5.2.4 Class DMS:: DamlReader

- DamlReader is dependent on the DMSReport::baseURI attribute
- DamlReader is dependent on the metrictypes enumeration
- DamlReader uses the MetricContainer class in the DamlReader:: GetMetric() method call.
- DamlReader is dependent on the PersistentStorageManager class for use of the persistent store to store results.

5.2.5 Class DMS:: MetricContainer

- MetricContainer is dependent on the DamlReader::uriData attribute.
- MetricContainer is dependent on the use of the Cohesion class
- MetricContainer is dependent on the use of the Coupling class
- Metric Container is dependent on the dmsReport::baseURI attribute

5.2.6 Class DMS:: Cohesion

- Cohesion is dependent MetricContainer::uriData attribute.

5.2.7 Class DMS:: Coupling

- Coupling is dependent MetricContainer::uriData attribute.

6 Interface Description

6.1Class interface

6.1.1 MainForm::ExecuteNewDamlQuery()

- This method instantiates a DMSReport object and passes it the baseURI and control. This operation is performed in the internal call to the ExecuteNewDamlQuery() method.

6.1.2 DMSReport:: CheckForPreviousResults()

- This method is called internally and in turn instantiates a PersistentStorageManager object and calls the method PersistentStorageManager:: DoesResultSetExist().

6.1.3 DMSReport::GetStatistics()

- This method is called internally and in turn instantiated a DamlReader object.
- This method calls DamlReader::Read().

- This method calls DamlReader::GetMetric().

6.1.4 DamlReader::GetMetric()

- This method Will instantiate a MetricContainer object.
- GetMetric will call MetricContainer::Calculate.

6.1.4 MetricContainer:: Calculate()

- Calculate will instantiate a coupling and cohesion object.
- Calculate will call the Cohesion:: ComputeCohesion() method
- Calculate will call the Coupling:: ComputeCoupling() method.

7 Data Detailed Design

a. See UML Diagram Figure 5.

8 Conclusion

a. I feel that the product that has developed will help answer many of those questions. With my position being project manager I have also gained a greater access of how to relate to people and what it takes to have a successful work group. We followed a typical life cycle in our program and with great timing we were able to produce a product not only on time, but even early.

9 Acknowledgements

a. First and foremost I would like to thank my project advisor, Tony Orme, for giving me the inspiration and guidance to complete this product.
b. I would also like to thank all of the people that contributed to the product either with their programming skills or documentation: Steven Geary, Matt Spivey, Norman Raley, Hermeet Sethi, and also David Evans.
c. Beyond the scope of this product I would like to thank Dr. Phil Richards, former chairperson of the computer science department, Dr. Jack Fix,

Dean of the College of Science. Without them none of this one have been possible.

d. I would also like to thank Dr. Modlin, the former director of the Honor's Program, and Dr. Mebane, the current director of the Honor's program for giving me the courage to complete this and make it through the honors program.

10 References

http://www.daml.org

McGuinness, D. L.; Fikes, R.; Hendler, J.; Stein, L. A., "DAML+OIL: An Ontology Language for the Semantic Web," IEEE Intelligent Systems, IEEE, vol. 17(5), Sep/Oct 2002, pp. 72-80.

http://www.ieee.org

Mahalingam, K.; Huhns, M. N., "A tool for organizing Web information," IEEE Project Reports, IEEE, vol. 30(6) , June 1997, pp. 80-83.

Cherry, S. M., "Weaving a Web of Ideas," IEEE Spectrum, IEEE, vol. 39(9), Sep 2002, pp. 65-69.

11 Appendix

a. Figure 1: Context Diagram Level 0
b. Figure 2: Context Diagram Level 1
c. Figure 3: Context Diagram Level 2
d. Figure 4: UML Diagram

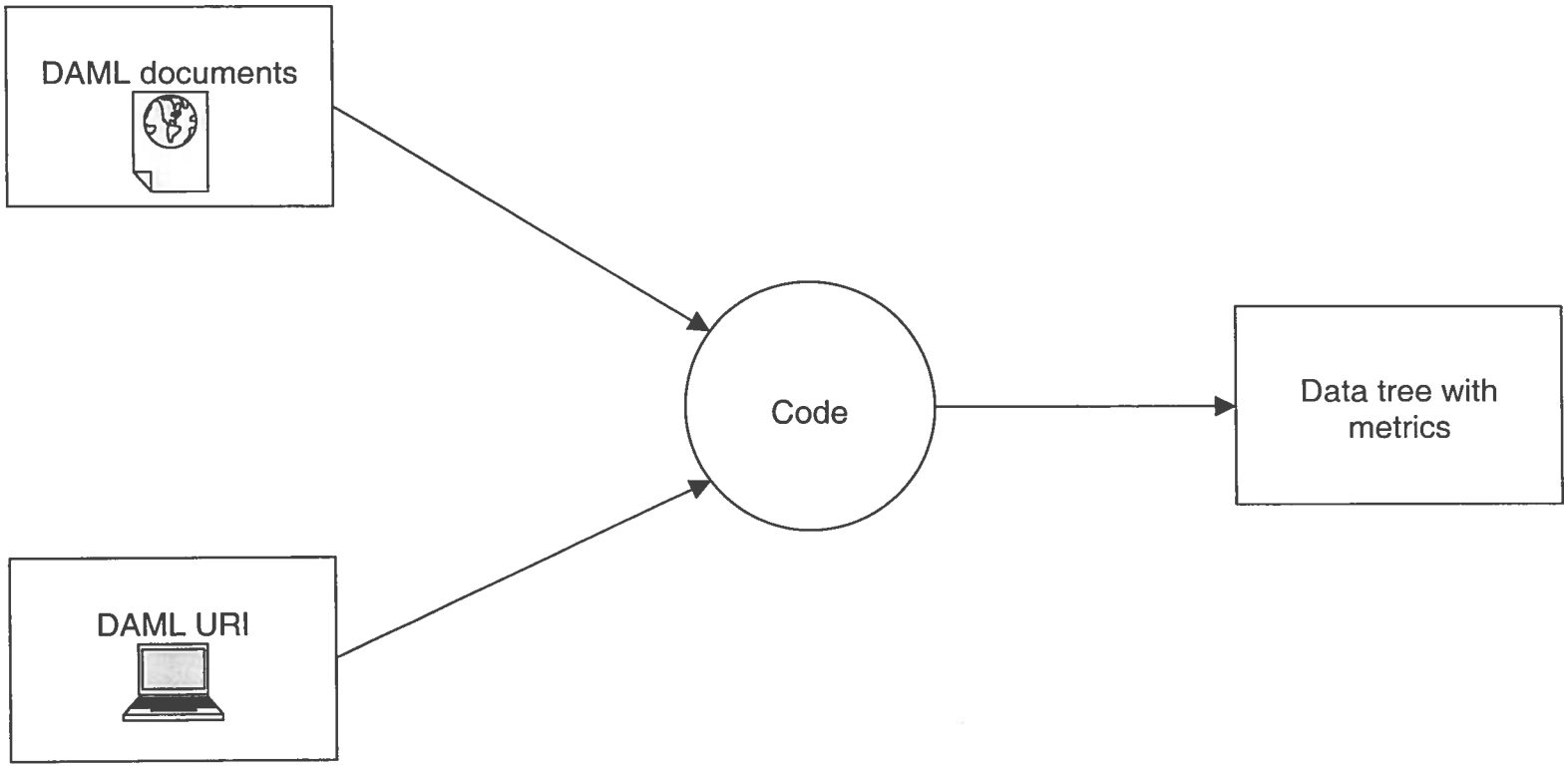


Figure 1 Context Level 0

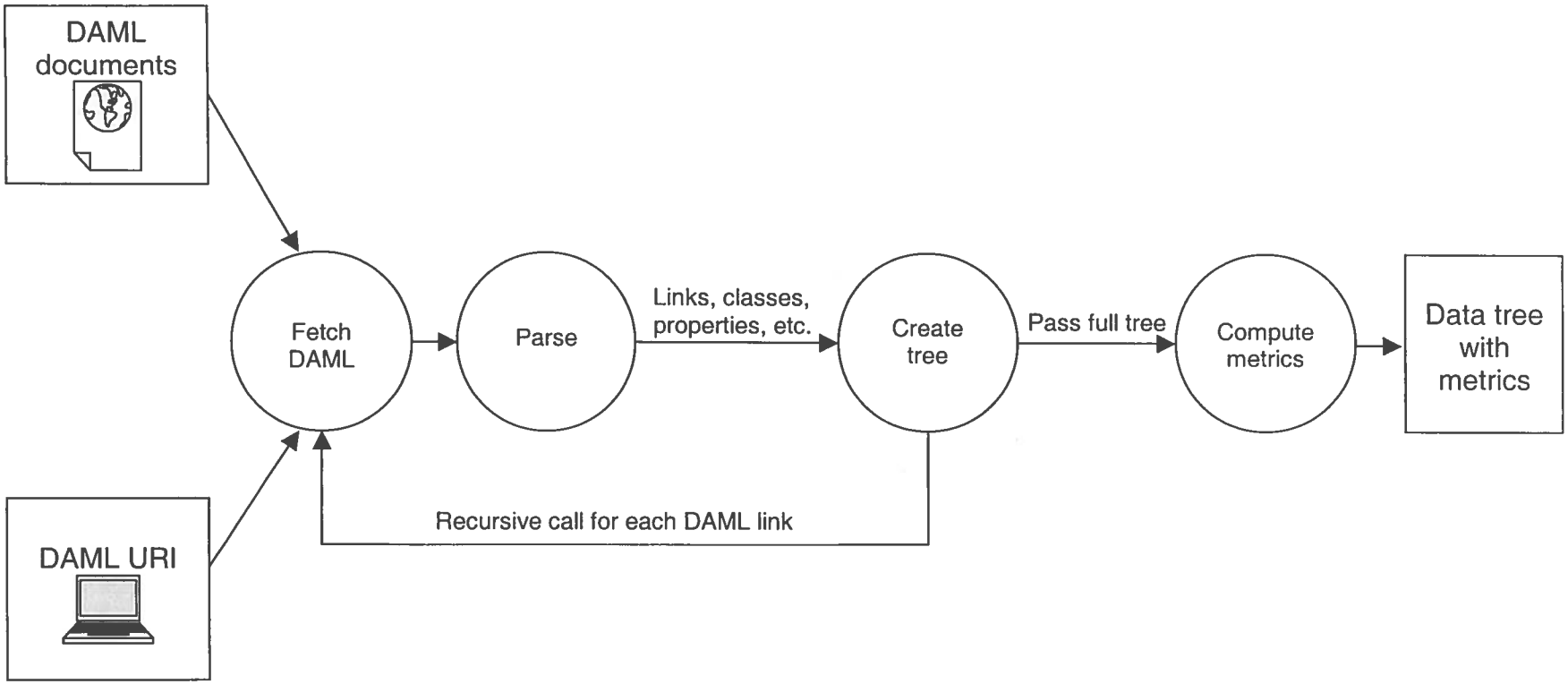


Figure 2 Context Level 1

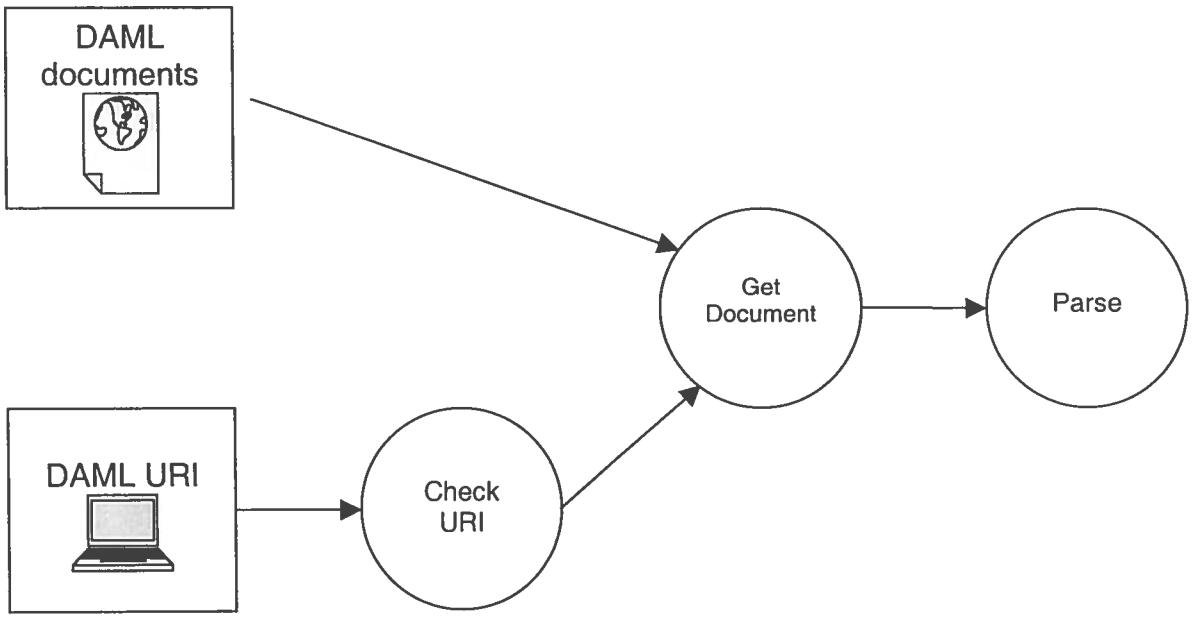
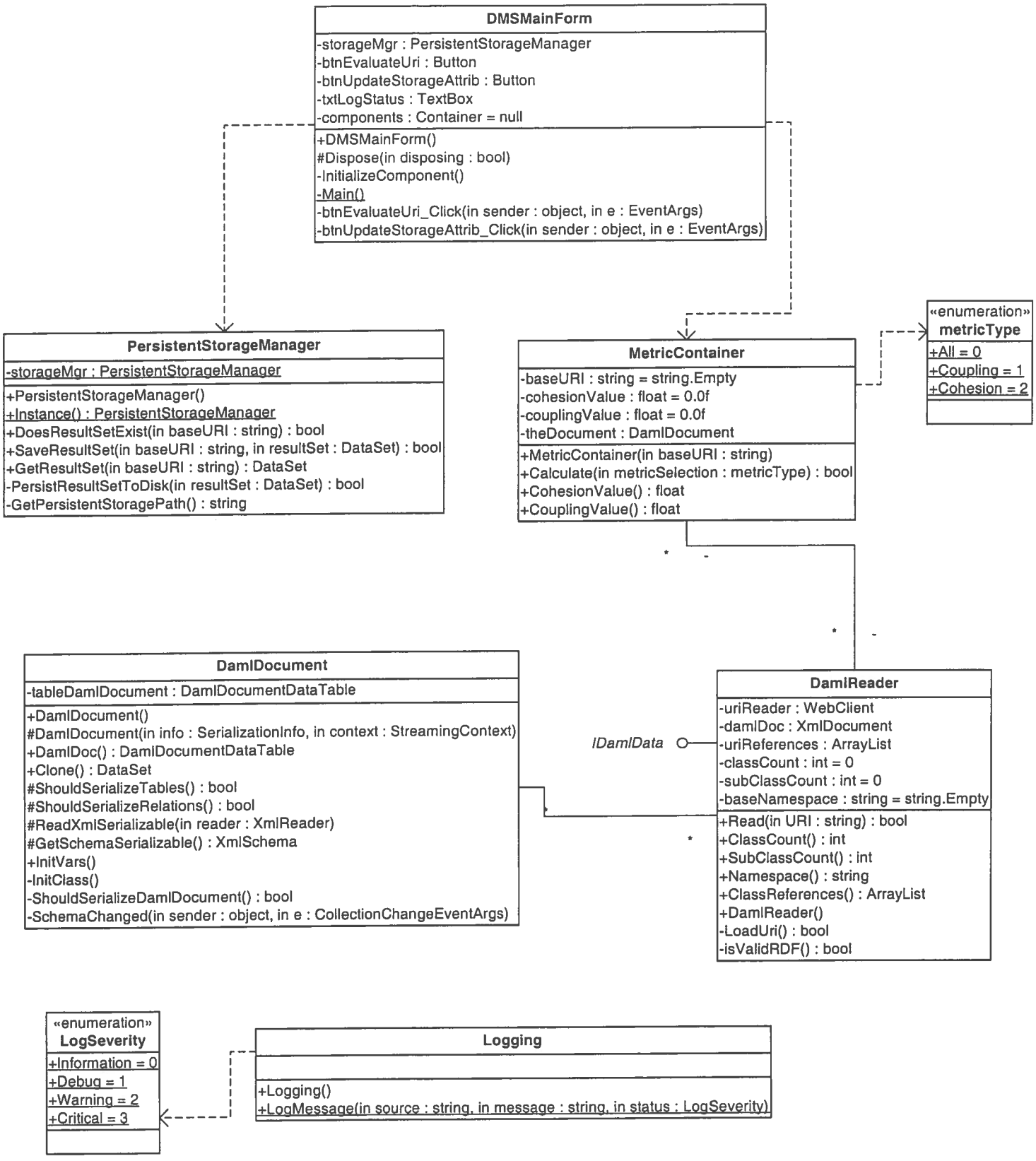


Figure 3 Context Level 1


DMSMainForm
-storageMgr : PersistentStorageManager
-btnEvaluateUri : Button
-btnUpdateStorageAttrib : Button
-txtLogStatus : TextBox
-components : Container = null
+DMSMainForm()
#Dispose(in disposing : bool)
-InitializeComponent()
-Main()
-btnEvaluateUri_Click(in sender : object, in e : EventArgs)
-btnUpdateStorageAttrib_Click(in sender : object, in e : EventArgs)
PersistentStorageManager
-storageMgr : PersistentStorageManager
+PersistentStorageManager()
+Instance() : PersistentStorageManager
+DoesResultSetExist(in baseURI : string) : bool
+SaveResultSet(in baseURI : string, in resultSet : DataSet) : bool
+GetResultSet(in baseURI : string) : DataSet
-PersistResultSetToDisk(in resultSet : DataSet) : bool
-GetPersistentStoragePath() : string
MetricContainer
-baseURI : string = string.Empty
-cohesionValue : float = 0.0f
-couplingValue : float = 0.0f
-theDocument : DamlDocument
+MetricContainer(in baseURI : string)
+Calculate(in metricSelection : metricType) : bool
+CohesionValue() : float
+CouplingValue() : float
«enumeration»
metricType
+All = 0
+Coupling = 1
+Cohesion = 2
DamlDocument
-tableDamlDocument : DamlDocumentDataTable
+DamlDocument()
#DamlDocument(in info : SerializationInfo, in context : StreamingContext)
+DamlDoc() : DamlDocumentDataTable
+Clone() : DataSet
#ShouldSerializeTables() : bool
#ShouldSerializeRelations() : bool
#ReadXmlSerializable(in reader : XmlReader)
#GetSchemaSerializable() : XmlSchema
+InitVars()
-InitClass()
-ShouldSerializeDamlDocument() : bool
-SchemaChanged(in sender : object, in e : CollectionChangeEventArgs)
IDamlData
DamlReader
-uriReader : WebClient
-damlDoc : XmlDocument
-uriReferences : ArrayList
-classCount : int = 0
-subClassCount : int = 0
-baseNamespace : string = string.Empty
+Read(in URI : string) : bool
+ClassCount() : int
+SubClassCount() : int
+Namespace() : string
+ClassReferences() : ArrayList
+DamlReader()
-LoadUri() : bool
-isValidRDF() : bool
«enumeration»
LogSeverity
+Information = 0
+Debug = 1
+Warning = 2
+Critical = 3
Logging
+Logging()
+LogMessage(in source : string, in message : string, in status : LogSeverity)

UML Figure 4.

2003 - 2.0

Honors Senior Project
Approval

Form 3 – Submit with completed thesis. All signatures must be obtained.

Name of candidate: S. Daniel Daugherty II

Department: Computer Science

Degree: B.S. CS

Full title of project: CK Metrics for DAML Documents

Approved by:

Project Advisor — 05/05/2003 — Date

H-S. Lamparter

Department Chair — 05/08/2003 — Date

John L. Mellone

Honors Program Director for Honors Council — 5/8/2003 — Date