CHAPTER 4

The Proposed Tool “DEMOS”

In this chapter, I will introduce approach that my work will relay on. Moreover, I will explain how this approach can address the two main issues concerning the design metrics implementation. Finally, I will show how I developed this approach to reach my new ideas and implementation improvements.

In software engineering field, the validation of software design is a so consuming effort, because the expert has to do his measures manually based on a will formed quality model with associated metrics. So, as a solution we need to automate this phase.

When we reviewed published work in this field we found that a lot of authors present many different design quality assessment frame work and different formal definition for design metrics.

A major problem of many of these metrics is their lack of precise definition which can lead to ineffective results, because the actual measurement depends on the understanding of metrics definition by the person who measures [2].

Also one more problem, some metrics depend on the code. So, we need to measure the design in its early stages to eliminate any extra effort and cost [2].

As a result, the main issue we should consider in this research is how to come up with a well formed model and precise metrics definition that not depend on the code.

In this research project, my work will be based on Mohammed ALWakeel novel approach [3]. He provides a framework to design an automatic tool that can measure only class diagram aspects of OO. He calls it, “Design Metrics

Crawler". The basic idea of this tool is to view the design document as data and metrics as queries on this data. In the following section I will explain it more.

Past Framework & Approaches

To design a framework model for OO design measurement, there is a need to focus in two problems. First, formal definition is a prerequisite for serious measurement. Second, design measurement should be possible without code.

Ralf Reissing suggests a solution to address these problems by given the requirement to the solutions and based on these requirements a formal model of OO design is outlined.

The requirements are **[2]:**

1. The definition of metrics shall be precise and ambiguous.
2. The metric should be based on design artifacts that are typically constructed in the design phase.
3. The metric shall be suited for automatic measurement as far as possible.
4. Measurement shall be possible even if the design is incomplete or not detailed.

There are several researches presented to address the previous problems. Here we will introduce three models:

I. The FLAME (Formal Library for Aiding Metrics Extraction)

This model presented by Baroni and Abreu **[1].** They describe OOD metrics based on OCL (Object Constraints Language) statements based on UML class diagrams.

This approach depends on standardized notations that are both precise and formal. However, it can leads to complicated definitions of metrics since OCL intended to describe constraints, not queries.

II. The ODEM (Object-Oriented Design Model)

This model presented by Ressing [1]. This approach based on introducing a new formal model that captures UML class diagrams elements.

This approach makes metrics definition easier to read and understand but the precision required, since the proposed approach consist of plain English expressions.

III. DM Crawler (Design Metric Crawler)

This tool proposed by Mohammed ALWakeel [2]. Its idea is based on using XQuery to represent metrics. This not only solves the problem of expressing metrics, but also eases building tools that extract measures from design artifacts.

This is the most important advantage of using XQuery in comparison with the other approaches. A tool that uses XQuery expressions for measures extraction could be easily modified to support new metrics,

just by expressing them in XQuery expressions, then adding them to the library of metrics supported by the tool.

The developed tool “Design-Metrics Crawler” is an implementation of the idea proposed above, with XQuery expressions stored in a database, extracting measures requires only retrieving the metric definition from the database, then running it against an XMI file, then collecting the resultant measure.

An addition that distinguishes “Design-Metrics Crawler” from other tools that extract measures from XMI files is the post-extraction functions. The developed tool can be used to detect design bad smells, detect structural flaws, identify classes with outliers measures, and to predict the fault-proneness of classes.

Adopted Approach

Many UML diagrams can be interpret as XMI files. This interpretation can lead us to an easy way to navigate and investigate the diagrams element and their associated properties by Xquery language since XMI is used to provide a query-able file format.

DM Crawlers basic idea is to express design documents by XMI schema. Then, metrics express as queries that can be run into these XMI files.

When files are expressed as XML format, there is XPath language help to inquiry the file data. This language offers two types of languages: XSLT and XQuery. For our research purpose, XQuery is best because XSLT was not intended primarily to be a query language, see chapter 3.

DM Crawlers Advantages

The main question here is “Did DM Crawler addresses the main two problems with design metrics which explained in chapter2”. The answer is yes and here the explanation:

The author considers the XQuery language to express the design metrics. So, when using XQuery to express matrices, we will gain these advantages [4]:

- **Preciseness:** XQuery enjoys a well-defined and standardized data model and formal syntax, hence metrics expressed using it will not exhibit the ambiguity usually found in metrics expressed using plain English.

- **Capability to express complex metrics:** Besides being a query language, XQuery offers programming languages-like capabilities such

as variables, looping, and branching. Thus allowing users to express what so ever metrics regardless of their complexity.

- **Easy to convert into automated code:** XQuery expressions are executable by themselves, so there is no need to convert them to any other format before execution.

- **Standardized:** XMI is an OMG standard, and XQuery is a W3C standard.

- **Wide applicability:** XMI is not intended to represent UML diagrams only, hence metrics that target diagrams other than UML ones could be expressed using XQuery. Also, XMI is used to represent reverse-engineered object-oriented code, such as Java and Smalltalk, allowing extracting measures from such code after being converted to XMI.

DM Crawlers Functionality

In the developed tool “Design-Metrics Crawler”, XQuery expressions are store in a database, so, extracting measures requires only retrieving the metric definition from the database, then running it against an XMI file, then collecting the resultant measure.

An addition that distinguishes “Design-Metrics Crawler” from other tools that extract measures from XMI files is the post-extraction functions. The developed tool can be used to detect design bad smells, detect structural flaws, identify classes with outliers measures, and to predict the fault-proneness of classes.

DM Crawlers Structures

In the developed tool “Design-Metrics Crawler”, there are three processing modules and one data model:

- **DM Crawler Data Model**

  Store both extracted measures and metrics definitions in relational database.

- **Metrics Manipulation Module**

  It is the part of DM Crawler that allows the user to add, modify, and remove metrics definition from database. He use 65 metrics associated with class diagram.

- **Design Document Manipulation And Measures Extraction Module**

  It is the part of DM Crawler that handles design documents stored as XMI files.

- **Measures analysis module**

  This is the GUI part of DM Crawler that helps the user analyzes the extracted measures.

  Design structure and functionality details are described in **[1].**

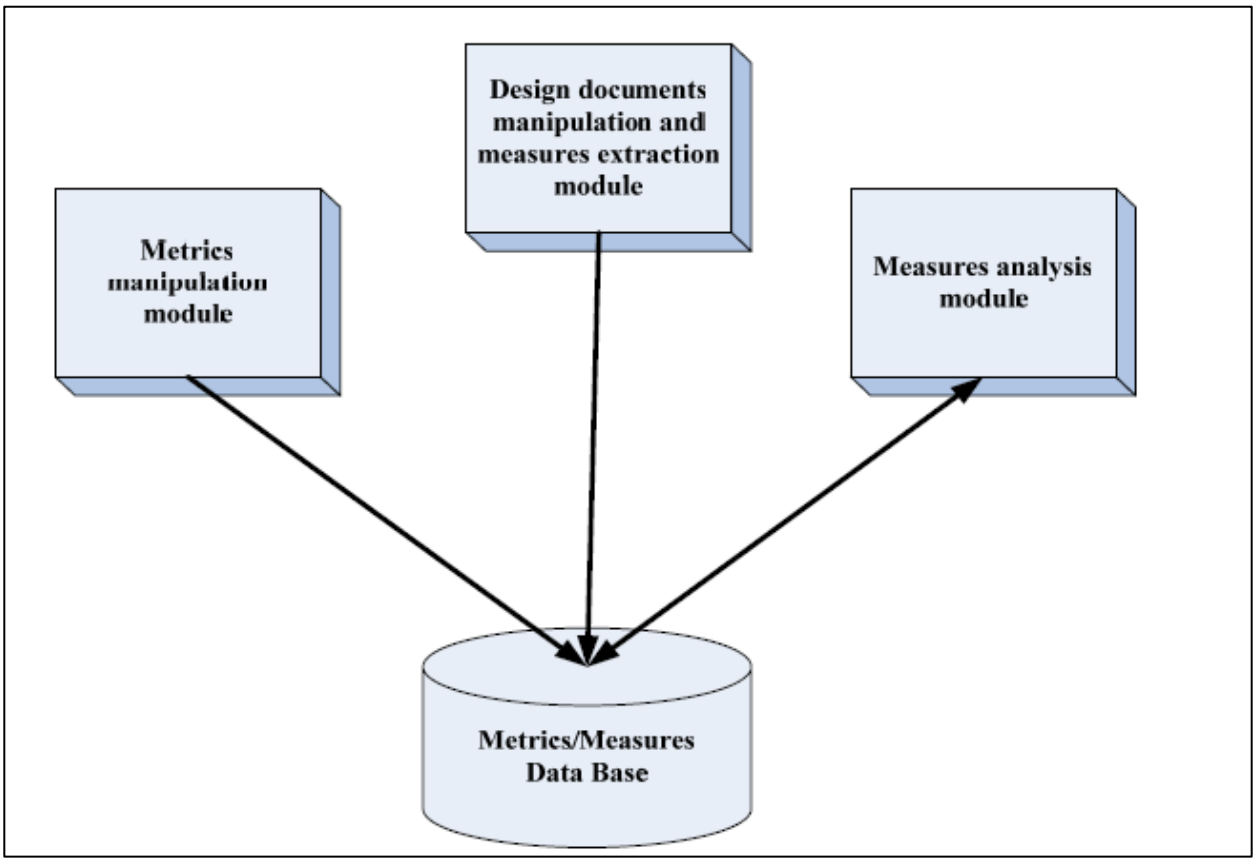

Figure 1: DM Crawler Structure

Presented DEMOS Tool

In DM Crawler approach, the model excludes the dynamic behaviors of OO design and concentrate in static features only. So, in my research project I will improve the DM Crawler tool by adopt other UML diagrams that represent dynamic behaviors, i.e. state diagram and activity diagram.

My proposed tool has the name DEMOS, stand for Design Measurement Object Oriented Metrics. This tool will treat design documents that expressed as XMI files by applying metrics that expressed as Xquery expressions. Then extracted results from these metrics will be re-stored in database for further display options and analysis.

After the extraction process, there is a post extraction phase, which allow user to display the stored results as a table, statistical table, bar chart or design detection report.

The following figure illustrates the basic structure of DEMOS.

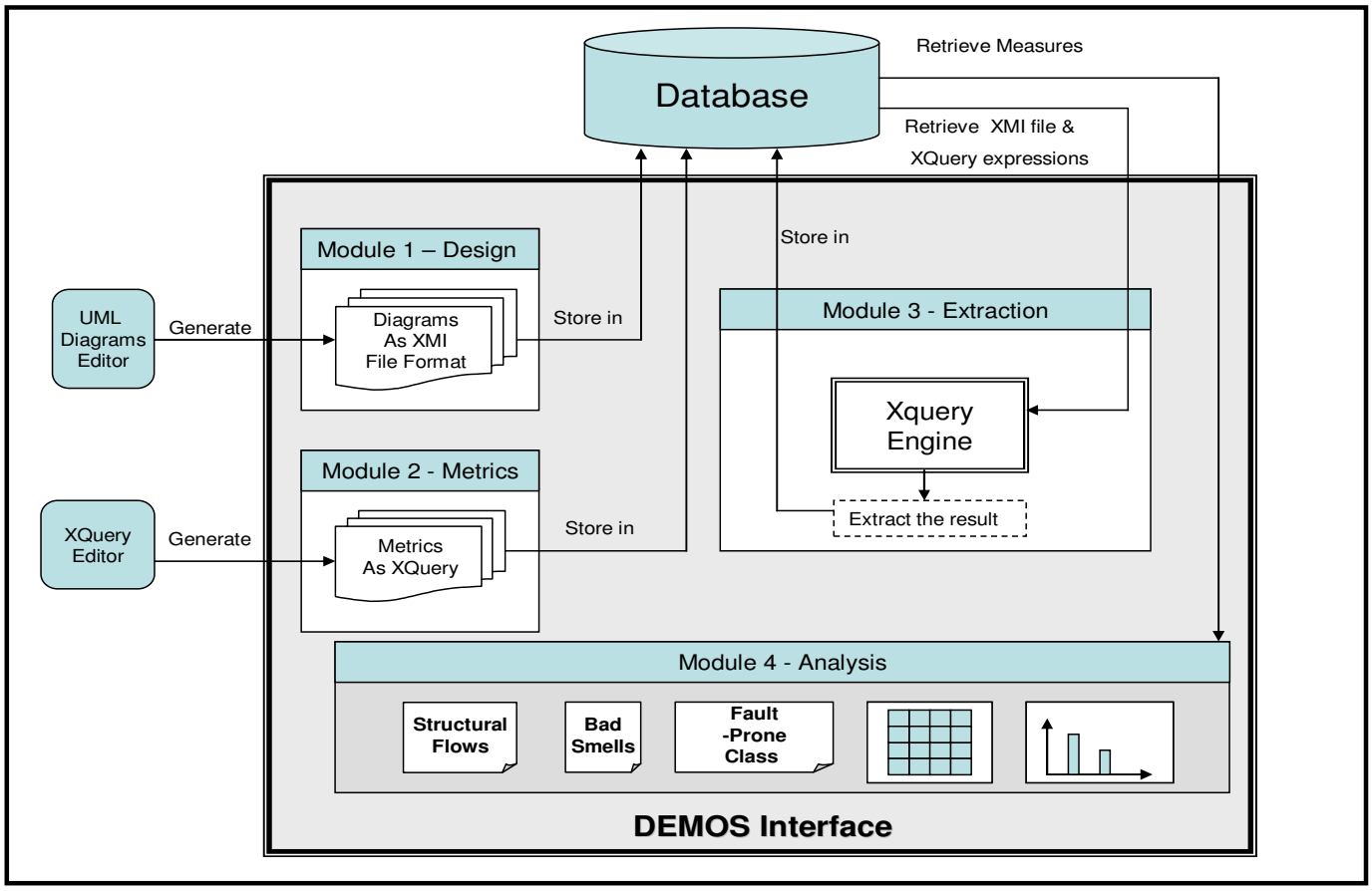

Figure 2: DEMOS Structure

DEMOS Design and Implementation

DEMOS Functional Modules

There are four functional modules:

OO design module, metrics module, measurement module, and analysis module.

- **OO design module**

This module will take care of UML design XMI file that are represented in UML diagram in XMI file format. Each design document will has its associated XMI diagrams. This gives the ability to have more than one version of design, so comparing between diagrams are allowed.

Design documents will not be saved in database, only its path will be saved. This module will give the ability to add, delete or even update the design documents.

User can select any of these following options:

1. **Add a design project**

   This option selected when we want to add a new design project. After add a design document name, a new empty design project will be exist for adding new associated diagrams.

2. **Delete a design project**

   This option will remove the design project from the database and all associated design diagrams and metrics results that belong to it.

3. **Edit existed design project**

   This option will allow user to apply any needed modification to the design name or description without affecting the associated diagrams.

4. **Add a design diagram**

This option selected when we want to add a new design diagram. A user will add a diagram XMI file path to be stored in database for later processing.

User will select to add class, state chart or activity diagrams.

All the added diagrams will be added to the open or selected design project.

5. **Delete a design diagram**

This option will remove the design diagram from the associated design project and the diagram path will be removed from the database.

6. **Edit existed design diagram**

This option will allow user to apply any needed modification to the design diagram. Either to modify the diagram information or to modify the diagram XMI file path.

- **Metrics module**

This module will take care of Static and Dynamic software Metrics. Each metric will have its specific diagrams types and it will be saved in a database.

This module will give the user the ability to add, delete or even edit any metrics stored in database.

User can select any of these following options:

1. **Add a design metric**

   This option selected when we want to add a new design metric. A user will add metrics related information, which diagram it measure and the corresponding Xquery expression.

   There are three metrics types: Class Metrics, State Metrics, and Activity Metrics

2. **Delete a design metric**

   This option will remove the design metrics from the database.

3. **Edit existed design metric**

   This option will allow user to apply any needed modification to the design metric.

There are three types of metrics:

a. **The Static Metrics**

   For static diagram I will use same metrics that are presented by DM Crawler.

   Here I will list them [2]:

Metric Acronym	Metric Name	Metric Acronym	Metric Name
AASP	Average associations per class	AHAGG	Average height of aggregation
AHF	Attribute hiding factor	AIF	Attribute inheritance factor
ANA	Average numbers of ancestors	ANPM	Average number of parameters per method
CAM	Cohesion among methods	CC	Children account
CF	Coupling factor	CLD	Class to leaf depth

DAC	Data abstraction coupling	DAM	Data access metrics
DCC	Direct class coupling	DIT	Depth of inheritance tree
ECM	Export coupling measure	FEF	Factoring effectiveness
HAGG	Height of aggregation	LCC	Loose class cohesion
MAA	Measure of attribute abstraction	MFA	Measure of functional abstraction
MHAGG	Maximum height of aggregation	MHF	Method hiding factor
MIF	Method inheritance factor	NAD	Number of abstract data type
NASC	Number of associations	NCM	Number of class methods in class
NDA	Number of overridden attributes	NDIN	Number of dependencies IN
NDOUT	Number of dependencies out	NIA	Number of inherited attribute
NIM	Number of inherited method	NKA	Number of package attributes
NKM	Number of package method	NNA	Number of new attributes
NNM	Number of new method	NOA	Number of ancestors
NOC	Number of children	NOD	Number of descendants
NODP	Number of direct parts	NOH	Number of hierarchies
NOM	Number of overridden methods	NOP	Number of parents
NP	Number of parts	NPA	Number of public attributes
NPM	Number of public methods	NRA	Number of protected attributes

NRM	Number of protected methods	NVA	Number of private attributes
NVM	Number of private methods	NW	Number of wholes
RER	Reuse ratio	PC	Parents count
PF	Polymorphism ratio	SIX	Specialization ratio
SPR	Specialization index	TCC	Tight class cohesion
TNA	Total number of attributes	TNC	Total number of classes
TNG	Total number of generalization	TNI	Total numbers of interface
TNM	Total number of method	TNR	Total number of relationship
TNC	Total number of associations	WMC	Weighted method count
PPD	Percentage of public data		

Table 1: Class Diagram Metrics

b. The Dynamic Metrics

For dynamic diagram metrics there are two metrics group, one for State Diagram and another for Activity Diagram.

Here I will list the State Metrics, some of these metrics proposed by [23] and the rest I proposed them:

Metric Acronym	Metric Name	Metric Acronym	Metric Name
GNSR [23]	Get Number of Region	GNS [23]	Get Number of States
GNST [23]	Get Number of Transition	GNSTG [23]	Get Number of Transition With Guard
GNSTE [1]	Get Number of Transition With Effect	GNSTT [23]	Get Number of Transition With Trigger
GNSTD [1]	Get Number of Transition With Do Activity	GNSTN [1]	Get Number of Transition With Entry Action
GNSTE [1]	Get Number of Transition With Exit Action	GNSTA [23]	Get Number of Transition with activity
GSCC [23]	Get Cyclomatic-Complexity		

Table 2: State Diagram Metrics

[1] Presented newly in this project

Here I will list the Activity Metrics, some of these metrics proposed by [22] and the rest I proposed them:

Metric Acronym	Metric Name	Metric Acronym	Metric Name
GNA [22]	Get Number of Actions	GNON [22]	Get Number of Object Node
GNIP [22]	Get Number of Node Input Pin	GNOP [22]	Get Number of Node Output Pin
GNP [22]	Get Number of Node Pin	GNIC [22]	Get Number of Initial Control Node
GNFC [22]	Get Number of Final Control Node	GNFFC [22]	Get Number of Flow Final Control Node
GNJC [22]	Get Number of Join Control Node	GNFOC [22]	Get Number of Fork Control Node
GNDC [22]	Get Number of Decision Control Node	GNMC [22]	Get Number of Merge Control Node
GNCN [2]	Get Number of Control Node	GNAP [2]	Get Number of Activity Partition
GNAMP[22]	Get Number of Activity Main Partition	GNASP [2]	Get Number of Activity Sub Partition
GNAG [22]	Get Number of Activity Group	GNCF [2]	Get Number of Control Flow
GNOF [22]	Get Number of Object Flow	GNF [22]	Get Number of Flow
GNG [22]	Get Number of Guard	GNEH [22]	Get Number of ExcHandler

Table 3: Activity Diagram Metrics

[2] These Metrics Presented newly in this project

- **Measurement module**

This is the important module. Its main job is to apply stored metrics definition that stored in database as XQuery expressions on design documents that stored as XMI files. Then, results of this measure will be stored in database for further analysis.

There are four different extraction processes. All the extraction processes has the same technique unless metrics that are related to class diagrams, because there is a need to extract all class element from the class diagram.

User can choose from the menu one of the four extraction module as follows:

1. **All design metrics**

This extraction option will run all metrics stored in database into the selected design diagrams.

2. **Static design metrics**

This extraction option will run static metrics stored in database into the selected class diagram.

3. **Dynamic design metrics**

This extraction option will run dynamic metrics stored in database into the selected state and activity diagrams.

4. **Specific design metrics**

This extraction option will run selected metrics stored in database to the selected design diagrams.

- **Analysis module**

This module called “post-extraction process”. This module can be called after applying measurement module.

There are three different outputs from this module:

1. **Table View**

   This module will present a list of measures that extracted by previous phase as a table.

2. **Statistical View**

   This module will present a list of measures with there associated statistics values such as variance, minimum and maximum.

3. **Design Detection Report**

   This tool can be used to give some additional information about the design. These detections taken from DM Crawler [3]:

   a. Identifying Outliers
   b. Detect Design Bad Smells
   c. Detect Design Structural Flaws
   d. Predicting Fault-Proneness

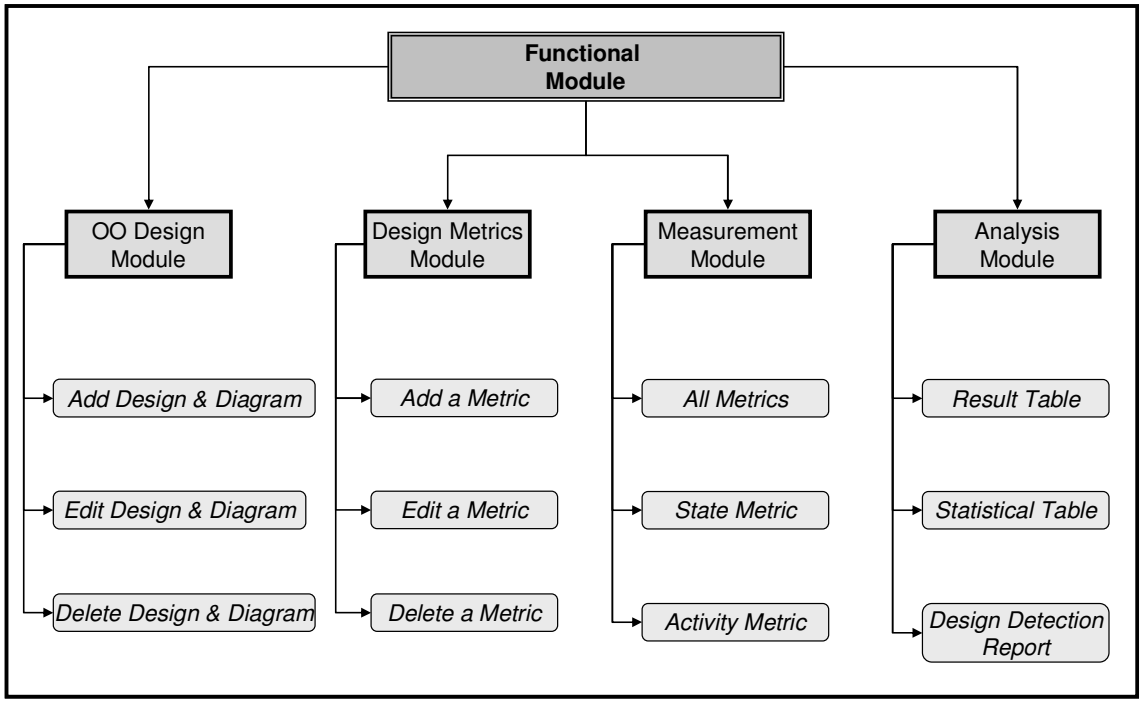

Figure 3: DEMOS Functional Module

DEMOS Data Module

The DEMOS data model is the database representation or schema. DEMOS data model consist of the following tables:

1. “Design” Table

This table stores all design project created by the user. It stores the design project name and description.

2. “CLASS_DIAGRAM” Table

This table stores all design related class diagrams uploaded by the user. It stores the class diagram name, description and file path.

3. “CLASSES” Table

This table stores all extracted classes from selected class diagram. It stores the class id and name.

4. “STATE_DIAGRAM” Table

This table stores all design related state diagrams uploaded by the user. It stores the state diagram name, description and file path.

5. “STATES” Table

This table stores all extracted states from selected state diagram. It stores the state id.

6. “ACTIVITY_DIAGRAM” Table

This table stores all design related activity diagrams uploaded by the user. It stores the activity diagram name, description and file path.

7. “ACTIVITIES” Table

This table stores all extracted activities from selected activity diagram. It stores the activity id.

8. “METRICS” Table

This table stores all design related metrics added by the user. It stores the metric's name, description, expression and its scope.

9. "CLASS_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into class elements. It stores the class diagram id, class reference, metric acronym and the result.

10. "CLASS_DIAGRAM_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into selected class diagrams. It stores the class diagram id, metric acronym and the result.

11. "STATE_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into state elements. It stores the state diagram id, state reference, metric acronym and the result.

12. "STATE_ DIAGRAM_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into selected state diagrams. It stores the state diagram id, metric acronym and the result.

13. "ACTIVITY_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into activity elements. It stores the activity diagram id, activity reference, metric acronym and the result.

14. "ACTIVITY_ DIAGRAM_METRICS_VALUES" Table

This table stores all extraction results from applying related metrics into selected activity diagrams. It stores the activity diagram id, metric acronym and the result.

15. “OO_PROPERTY” Table

This table stores all Object Oriented design related features, such as, Inheritance and complexity.

Here is the ER diagram of the previous table:

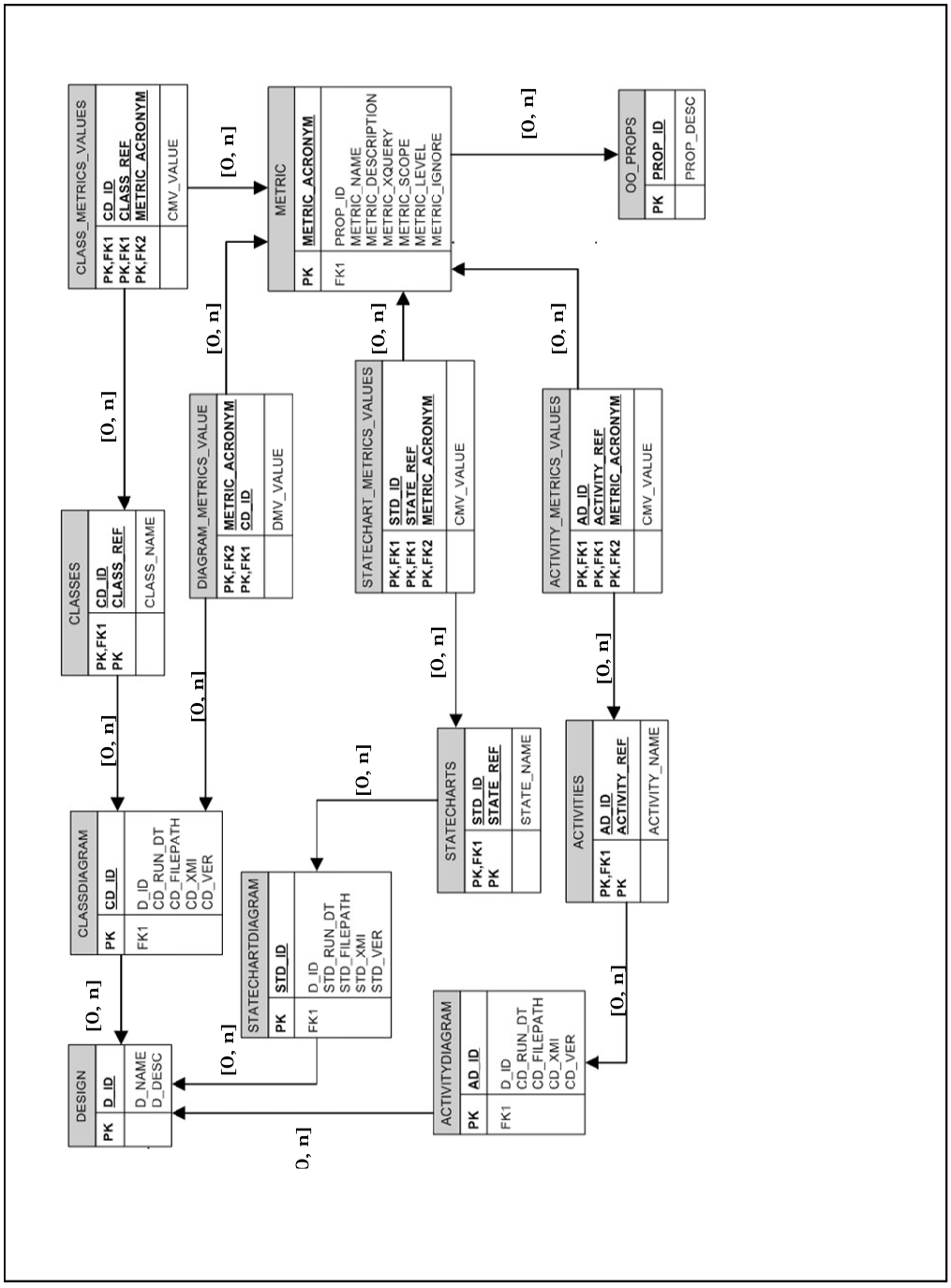

Figure 4: DEMOS Data Model

DEMOS Workflow

User can switch between various modules. Either manipulating metrics, design documents or play the main role for this tool which is measuring and analysis.

Here is the main workflow of DEMOS tool, we assume there is already a design project with its associated added diagrams and metrics:

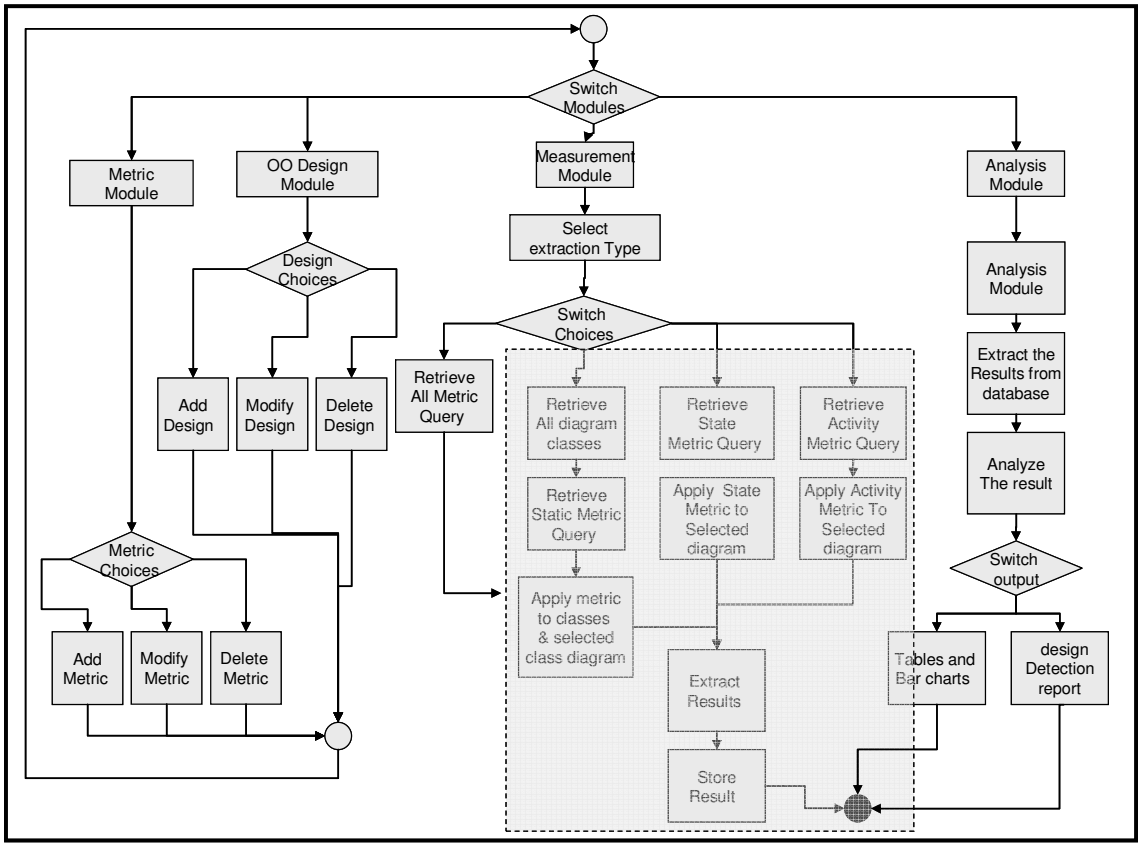

Figure 5: DEMOS Flowchart

- User selects a design project.
- After selection a design project, user can add design diagrams or select diagram(s) from available lists. There are class diagrams lists, state diagram lists and activity diagram lists.

- User can select only one design diagram from each available diagrams lists. Also, user can select one type diagram or all three types of diagrams.

- After user selections, he can go to extraction process. This tool allow user to extract all metrics or extract a specific diagram related metrics. Here I will list the work flow of each extraction type:

    - **Extract Class Diagrams**

    There are two extraction phases:

        - ***Classes Metrics Extraction***
            - Get all classes elements ids from xml file
            - Store the extracted ids into CLASSES table
            - For all metric that related to Class Scope do
            - For all classes ids stored in a CLASSES table do
            - Extract the metric on current class and stored the result in CLASS_METRIC_VALUES
        - ***Class Diagram Extraction***
            - For all metric that related to Class Diagram Scope do
            - Extract the metric on current class diagram and stored the result in CLASSDIGARM_METRIC_VALUES

    - **Extract State Diagrams**

    There are two extraction phases:

        - ***State Metrics Extraction***
            - Get all state elements ids from xml file
            - Store the extracted ids into STATES table
            - For all metric that related to State Scope do
            - For all states ids stored in a STATES table do

- Extract the metric on current state and stored the result in STATE_METRIC_VALUES

▪ ***State Diagram Extraction***

- For all metric that related to STATE Diagram Scope do
- Extract the metric on current state diagram and stored the result in STATE_DIGARM_METRIC_VALUES

○ **Extract Activity Diagrams**

There are two extraction phases:

▪ ***Activity Metrics Extraction***

- Get all activities elements ids from xml file
- Store the extracted ids into ACTIVITIES table
- For all metric that related to Activity Scope do
- For all activity ids stored in a ACTIVITIES table do
- Extract the metric on current activity and stored the result in ACTIVITIES _METRIC_VALUES

▪ ***Activity Diagram Extraction***

- For all metric that related to Activity Diagram Scope do
- Extract the metric on current activity diagram and stored the result in ACTIVITY_DIGARM_METRIC_VALUES

- Finally, user can switch between different view modes of results. Statistical computed results or designs generated reports and bar chart are the available view modes.

DEMOS Analysis process

Now, after extraction done, a post extraction phase takes place. Post extraction phase is an analysis and preview process.

DEMOS interface provide user with different views that let him to surfs among the extracted result and also provides user with different design quality assessment reports.

The following are the views and reports presented by DEMOS interface:

- **Design Metrics Extraction Results Table**

This table will provide all extracted result that stored in a table from previous extraction process. It will present two-dimensional table.

This table will show the diagram corresponding metrics with their associated elements and results.

User can select in this view to show the diagram metrics result or this diagram elements corresponding metrics results.

- **Design Metrics Extraction Statistical Table**

This table will provide statistical computation based on extracted result from the previous extraction process. It will present two-dimensional table.

This table will show the diagram corresponding metrics with their statistical results as in this list: (Count, Maximum, Minimum, Mean, Variance, and Standard Deviation)

User can presents this computational statistical result by bar chart.

- **Design Detection and Assessment Reports**

Based on extracted results, useful information, corresponds to the object oriented design, can be examine.

These are the information provided to be examined [3]:

1. **Detecting Structural Flaws**

   It means the errors that may be accidental or intended.

   - The number of association per class metric, (NAPC), is used to identify classes with no association
   - The total number of methods Metric, (TNM), is used to identify classes with no method

- Total Number of attributes metric, (TNA), is used to identify classes with no attribute
- Total Number of attributes and Total number methods Metric used together to identify empty classes.
- Detecting Isolated states
- Detecting no input states
- Detecting no output states
- Identify states that does not have names
- Detecting Isolated Activities
- Detecting no input States
- Detecting no input states
- Identify activities that does not have names

2. Detecting Design Outliers

This will gives the user the ability to show results according to a user defined thresholds. Ex: showing the highest class Inheritance depth.

3. Detecting Design Flaws

It means the deviation from the high quality design characteristics.

- Data Classes

  Data classes means a class that hold data only and which another classes may be relay on them.

  Since the data attribute separated from the methods, may be this indicate a meaning less data abstraction.

  In DEMOS tool, I will use the model presented by J. Rail [24]:

$$\begin{aligned}\text{Data Classes} = {} & (\text{WOC, Bottom Values (33\%)}) + \\ & (\text{WOC, Lower Than (0.33)}) + \\ & (\text{NPA, Higher Than (5)})\end{aligned}$$

Where:

a. WOC is a weight of a class which is calculated as following:
   - Number of non – inherited method in the class divided by the total number of non inherited methods and the total number of non inherited attributes
b. NPA is a Number of Public Attributes in a class

- God Classes

It means the centralized class which participates almost in all work flows of a design. This type of class can affect the reusability and understandability of the design due to the complexity.

In DEMOS tool, I will use the model presented by J. Rail [24]:

$$\text{God Classes} = (\text{DCC, Top Values (20\%)}) + (\text{DCC, Higher Than (4)}) + (\text{WMC, Higher Than (20)})$$

r

a. DCC is a Direct Class Coupling, which counts the classes directly related to a class through Inheritance or relationships.
   - WMC is a weighted Method Counts Metrics, which indicates the total complexity of all methods in a class.

4. Discover Fault-Proneness of classes

This mean the probability that a class will has a fault. Here I will use the same model used in DM Crawler which based on an equation proposed by K. EL Emmam.

His equation defined as the following [26], [30]:

$$\pi = 1/ (1+e)^{-3.97+ (1.06*DIT) + (0464*TNA) + (1.47*OCMEC)}$$

Where:

DIT → "Depth of Inheritance tree" Metric

TNA→ "Total Number of Attribute" Metric

OCMEC → "The Number of Other Classes that have Methods with parameter Types of this Class" Metric

DEMOS Post Extraction Reports

- Class Diagram Measurement Report
- Classes Measurement Report
- State Diagram Measurement Report
- States Measurement Report
- Activity Diagram Measurement Report
- Activities Measurement Report
- Class Design Structural Flaws Detection Report
- State Design Structural Flaws Detection Report
- Activity Design Structural Flaws Detection Report
- Design Flaws Detection Report
- Design Diagram Outliers
- Class Diagram Fault Proneness Assessment

Used Software for Development

Used Software for Development

In this tool I used the stylus studio editor to edit and test all xquery expressions that used to express the design metrics. This editor simplify the quick testing to ensure that the metrics working properly.

For expressing the UML diagrams there are many alternatives editors options, however there is a main feature should be available in a these editor which is the capability of transforming the UML diagrams in XMI format. This will allow applying designed metrics that expressed in Xquery format to these kinds of documents. So I use the “Poseidon for UML “editor.

In implementation I use c#.net to implement the GUI interface. I use SQL Server 2000 to handle my database tables, and then I connect this

database to my interface to allow easy retrieving and storing transactions.

I use “Altova Xml Engine” for extracting the XQuery expressions, which represents the design metrics, from the XMI files that represent UML diagrams.

These are the software used to develop the DEMOS tool.

Software Tool	Purpose
Microsoft SQL Server 2000	DBMS
Microsoft Visual Studio 2005 – C#.Net	Interface Development
Poseidon For UML PE 5.0.1	UML Diagrams Editor
Stylus Studio 2007 XML Enterprise Suite	Xquery Editor
Altova XML 2007	Xquery Engine

Table 4: Used Software for Development

Development Critical Phases

The main critical phase I face in this project is how to do the extraction automatically without bothering the user with extract the result manually.

So, I found many engines, for example: “Data Direct” and “Qizx/Open” but since there are a lack of helpful supporting, I decide to search for another, so I found “Altova XML”, which is new and very helpful engine. I add this engine to my code as DLL library. Then I use available commands for retrieving and extractions.

Another critical phase I face in this project is crashing of the database server. This came as a result of change the authentication of accessing the database accidentally when I try to define a dataset in c# for developing reports. So I remove the server and reinstall it again.

DEMOS Features

This tool will has these features:

- DEMOS based on precise and will formal approach which will provide us with consistent and will provable model.
- It is automated process which will ensure accurate and fast results.
- This tool has GUI which will make usage of it easier than manual approach.
- It has the ability to support adding, modifying or removing metrics and designs easily.
- It allows assessing different designs and compare between them to decide appropriate one as a solution.
- This tool will be valuable for studying different Object oriented features, such as Coupling, Inheritance and cohesion.
- This tool will provide users with reports that detect design features which will help them before implementation phase.

Comparison to the Other Works

These are the comparison features between SD Metrics, DM Crawler and DEMOS.

Input Source:

All of these tools accept the UML diagram as XMI file format but how to process and deal with these files is difference.

Measure Extraction:

SD Metrics use Java and C++ to express the design metrics, which will be effort consuming and hard to apply changes.

DM Crawler and DEMOS use the Xquery expression to express the metrics. This approach will give the developer variety of good features as I explain previously in chapter3, under Xquery advantages section.

Ability to Add New UML Diagrams and Design Metrics:

In DM Crawler and DEMOS, it's very easy to expand the tool to adopt another UML diagrams and new design Metrics, because it depends on adding them as standard input, i.e. XMI file and Xquery expression.

However, this is so hard in SD Metrics because it depends on re-programming the interface to accept new changes.

Supported UML Diagrams and metrics:

SD Metrics supports both static and dynamic UML diagrams, while in DM Crawler only static diagrams are supported. In DEMOS, both diagrams types are supported.

SD Metrics support only 45 static metrics and 18 dynamic metrics while DM Crawler supports 65 static metrics and no dynamic metrics available. In DEMOS, I support 65 static metrics and 33 dynamic metrics.

Programming Language:

SD Metrics use Java and C++ to extract the result from the XMI files by applying the Xquery expressions that represent the design metrics. In another side, the DM Crawler use Visual Basic6 and the DEMOS use C#.Net.

I use the c# for these two advantages which is not available in VB6:

- The “using” statement, which makes unmanaged resource disposal simple.
- Explicit interface implementation, where an interface which is already implemented in a base class can be re-implemented separately in a derived class.

I use the c# for this advantage which is not available in Java:

- Allowing the drag and drop components features to the “Form”

Speed of the Tool:

Since SD Metrics express the metrics programmatically and no need to retrieve data from database, the execution time is so fast.

However, in DM Crawler and DEMOS the execution time is slower because it depends on two factors:

- First, it depends on the Xquery Engine that been used. DM Crawler use Qizx/Open and execute the extraction as a batch file. Hence the Qizx/Open developer not concern about the speed, so the DM Crawler extraction is slow.

  In DEMOS, the execution time is better than DM Crawler, because we use “Altova XML” [25], which is a highly performance tool more than Qizx/Open. It needs approximately 0.5 second for each metrics execution.

- Second, retrieving metrics and storing the result from and to database play a main role in latency of execution time.

In the following table, I summarize the comparison feature between available these tools.

Comparison criteria	Tools		
	SD Metrics[22]	DM Crawler	DEMOS
Input source	XMI	XMI	XMI
Measure extraction by	JAVA/ C++	XQuery	XQuery
Output destination	HTML File	Database	Database
Number of supported class metrics	48	65	65
Number of supported State metrics	7	-	11
Number of supported Activity metrics	10	-	22
Total supported metrics	65	65	88
Ability to support new diagrams	Very hard	Easy	Easy
Ability to support new metrics	Very hard	Easy	Easy
Programming language	JAVA/ C++	Visual Basic	C#.Net
Speed of the tool	Fast	Slow	Medium
Post extraction function	Tables and Charting	Tables ,Charting Bad smells detection, Structural flow detection, Spotting outliers	Tables, Charting, Bad smells detection, Structural flow detection, Spotting outliers

Table 5: Comparison Table