بسم الله الرحمن الرحيم

Sudan University of Science and Technology

College of Graduate Studies

Development of a Decision Support System for Hydraulic and Structural Design of Water & Wastewater Treatment Units

تطوير نظام لدعم قرار التصميم الهيدروليكي والإنشائي لوحدات معالجة المياه والمياه العادمة

A Thesis Submitted in Complete Fulfilment of the Requirements for the Degree of Doctor of Philosophy in Environmental Engineering

By:

Hisham Isam Mohamed Abdel-Magid

Supervisors:

Dr. Eng. Yousif Ali Yousif

Dr. Eng. Alsadig Alhadi Alhasan

January 2015

Dedication

The researcher dedicates this effort to his:

beloved late mother, may Allah rest her soul in the eternity of heaven ...

dear friend and mentor, his cherished father, may Allah bless him with a long and healthy life ...

kind and warm brothers and sisters…

compassionate and dear wife…

precious and beautiful daughters and sons …

the souls of his dearly loved grandmothers and grandfathers ...

May Allah bless them all.

Acknowledgements

All thanks and gratefulness is for Allah, the most gracious, the most merciful.

The author expresses his in depth thanks to all those who aided him in accomplishing this work towards achieving its goals and objectives. Special thanks would go to his supervisors Dr. Eng. Yousif Ali Yousif and Dr. Eng. Alsadig Alhadi Alhasan.

Very special thanks are directed to: his friend, brother, mentor, and great father Prof. Dr. CEng. Isam Mohammed Abdel-Magid for his infinite support, his dear adored and treasured mother Eng. Layla Saleh Mahmoud, his lovely brother and sisters (Mr. Dr. Mohammed, Journalist and media specialist Mrs. Lubna, Eng. Mrs. Taghried, Eng. Tasneem, and Mrs. Ayah) for their patience and encouragements, his kind wife Mrs. Nuha and his beloved little ones Miss Weam, Miss Riham, Master Isam (junior) and Master Bassam.

Sincere thanks also are expressed to those great people who didn't hesitate a second in helping him in every possible way and form: Dr. Abdel-Ghani Abdel-Galil (Manager of Environmental Audit, Inspection and Monitoring Project in Qatar) of Environmental Solutions Inc. KSA; Dr. Baraka Mohamed Kabir (Head of Environmental Engineering Department), Dr. Suhair Abdel-Qayoum, Dr. Mohamed Abugaib, Eng. Abusabah Elfatih, Eng. Alsadig Almaleeh, Eng Khattab Ahmed, Mrs. Sabah Sid Ahmed and Ms. Buloud Abdel-Marouf at the College of Water and Environmental Engineering, Sudan University of Science and Technology (SUST); Dr. Abdallah M. T. Shigidi and Dr. Sharaf-Eldin Bannaga at College of Engineering, SUST; Dr. Adil Al-Khidir (Head of Civil Engineering Department), Dr. Gamal Murtada (Director of Water Research Centre), Dr. Elfadil Azrag, Eng. Mohamed Omer, Eng. Lamees Abdel-Basit, and Eng. Asim Elsanosi of Faculty of Engineering, University of Khartoum. Dr. Osman Naggar (General Manager of General Water Corporation) and Eng. Hisham Mirghani of General Water Corporation; Dr. Khalid Abdel-Fattah at College of Engineering, Karari University; Dr. Zienab Abdel-Rahim (Director, Institute of Technological Research) of National Center for Research; Eng. Amal Rabbah (Deputy General Manager) of Industrial Research and Consultancy Center; Eng. Nazar A. Saad & Eng. Atif A. A. Obeid of Kenana Engineering & Technical Services (KETS); Dr. Abdul-Rahman ben

Salih Hariri (Dean College of Engineering), Prof. Isam Mohamed, Dr. Sami Abullah Osman of Environmental Engineering Department - University of Dammam, Kingdom of Saudi Arabia; Dr. AbdelWahid Hago (ex-Head of Department of Civil Engineering), Sultan Qaboos University, Sultanate of Oman; Dr. Hatim Sharif of University of Texas at San Antonio, USA; Prof. Robert Kalin and Dr. Tara Beattie of University of Strathclyde; Eng. Zelalem Mekonnen of Addis Ababa Institute of Technology, Addis Ababa University, Ethiopia; Dr. Yousif Ibrahim (ENTRO Officer in Charge), Dr. Ahmed Khalid, Dr. Mamdouh Antar, Eng. Robel Tilaye, and Eng. Azeb Mersha of Eastern Nile Technical Regional Office (ENTRO), Nile Basin Initiative (NBI), Ethiopia; Dr. Elnasir Abdelwahab of Water Resources Planning Model, NBI, Ethiopia; Eng. Samar Kamaledin of Parsons Engineering Consultants, Sultanate of Oman; Eng. Mohamed Rimawi (Chief Resident Engineer), Eng. Amir Elginaid Awad (Chief Senior Structural Engineer) and Eng. Ahmed Tiema of Dar Al-Handasa Consultants, Qatar; Eng. Zuhair Hamid of Powerline Engineering PLC, Qatar.

The researcher

Table of Contents

Subject ***Page No.***

Dedication i
Acknowledgements ii
Table of Contents iv
List of Tables vi
List of Figures vii
Acronyms x
Abstract xi
Abstract (Arabic) xii

Chapter One: Introduction 1

1.1 Background 2
1.2 Research Objectives 3
1.2.1. General Objective 3
1.2.2. Specific Objectives 3
1.3 Research Hypothesis and Research Questions 4
1.4 Research Methodology 4
1.5 Expected Outcomes 5
1.6 From Vision to Action (Beneficiaries and Stakeholders) 5
1.7 Project Plan and Time Frame 5

Chapter Two: Literature Review 9

2.1 Preamble 10
2.2 Decision Support Systems in Water and Wastewater Treatment 17
2.3 Wastewater Treatment Plant (WWTP) Design Limitations 21
2.4 Mathematical Modelling of WT/WWTU 22
2.4a Previous Software for WWTU Design and Limitations Incurred 22
2.4b Mathematical Models and Computer Simulation 25
2.5 DSS Objectives and Framework 28
2.5a DSS Objectives 28
2.5b DSS Framework 30

Chapter Three: Materials and Methods ***31***

3.1 Research Methodology 32

3.2 Computer Hardware 32

3.3 Software Functionality 33

3.4 Software Development and Implementation 35

3.5 Model Validation and Software Troubleshooting 40

3.6 Software Dissemination 40

Chapter Four: Results and Discussion ***41***

4.1 Results 42

4.1.1 Model Conceptualisation 42

4.1.1a Abstraction of Design Modules 42

4.1.1b DSS Software Layout 42

4.1.1c WISAM's Software Structure 48

4.1.2 DSS Software Formulation, Validation and Verification 68

4.1.2a Software Development Under VB.Net 68

4.1.2b Development of MS Excel Calculation Sheets 76

4.1.2c GIS Integration 78

4.2 Discussion of Results 86

Chapter Five: Conclusions and Recommendations ***96***

5.1 Conclusions 97

5.2 Recommendations 98

References ***99***

Appendix ***108***

The Formulated Software (WISAM©) VB Source Code

List of Tables

Table No.	Title	Page
1.1	SWOT Analysis for The Research Project	3
2.1	Major Wastewater Treatment Units	18
2.2	Summary of Some Water Treatment Decision Support Systems as reviewed by Hamouda (2009)	24
3.1	Tested Computer Hardware Configuration Ranges	32
3.2	Typical flowchart shapes	33
3.3	Typical UML Class Diagram shapes	34
3.4	Visual Basic .NET Commonly Used Data Types	37
4.1	Summary Description of WISAM's Core Parameters	71
4.2	Sample of Moment Coefficient Factors at Tank Walls for Case#3: Free Top & Fixed Base	84
4.3	Rearranged Moment Coefficient for GIS Import	85

List of Figures

Figure No.	Title	Page
1.1	Research Project Schedule	8
2.1	UML Level 0 package diagram	15
2.2	UML Level M package diagram	16
2.3	Wastewater Treatment Units	19
2.4	Water Treatment Methods	20
3.1	Microsoft Visual Basic.NET IDE	36
3.2	Vector Data Representation	43
3.3	Raster data grid	44
3.4	Triangulated Irregular Network (TIN)	44
4.1	Conceptual Design Structure of The DSS Software Platform	47
4.2	Conceptual Identification Process of a U-Plug	49
4.3	Conceptual Layout Design of WISAM’s Main Window GUI	51
4.4	Simple U-Plug Linkage With Configuration Accessibility	52
4.5	Different Layout Configurations of U-Plug Linkages	53
4.6	Main Software Flowchart	55
4.7	Flowchart of Defining New U-Plug Type Subroutine	58
4.8	Flowchart of Defining U-Plug Linkages	61
4.9	Flowchart of Processing Hydraulic Calculations	62
4.10	Flowchart of Processing Structural Calculations	63
4.11	Flowchart of Importing U-Plug Subroutine	64
4.12	Flowchart of Invoking New U-Plug Instance Subroutine	66
4.13	UML Class Diagram for WISAM’s Layout Structure	67
4.14	UML Class Diagram for WISAM’s Core Structure	72
4.15	Main Screen for Creating a New VB Project	73
4.16	VB.NET Main IDE	73
4.17	VB.NET Toolbox Subsets	74
4.18	Addition of Programming Code Modules	75
4.19	Addition of Database Components	75
4.20	Addition of General File Modules	76

4.21	Addition of Various Windows Forms	76
4.22	Addition of a Reporting Agent	76
4.23	Creation of WISAM's Main MDI	77
4.24	Creation of WISAM's Modelling Window	77
4.25	Creation of WISAM's Properties Window	78
4.26	Creation of WISAM's ToolBox Window	78
4.27	Creation of WISAM's U-Plug Builder Interface (the Inputs page)	79
4.28	Creation of WISAM's U-Plug Builder Interface (the Outputs page)	79
4.29	Creation of WISAM's U-Plug Builder Interface (the Equations page)	80
4.30	Creation of WISAM's U-Plug Builder Interface (the Code Compilation page)	80
4.31	Formulation of WISAM's Programming Code	81
4.32	Calculation Sheet of Rectangular Concrete Tanks	82
4.33	Embedded Design Coefficients in The Calculation Sheet of Rectangular Concrete Tanks	83
4.34	ArcGIS-ArcMap Software Interface	87
4.35	Adding XY GIS Data	87
4.36	Identification of Geographic Coordinates	88
4.37	The Imported Coefficient Points Table	88
4.38	Exporting Point Data Event – Menu Item	89
4.39	Exporting Point Data Event – Properties Window	89
4.40	Display of Imported Coefficients via the Attribute Table	90
4.41	Definition of Screening U-Plug – Assignment of Inputs	92
4.42	Definition of Screening U-Plug – Assignment of Outputs	92
4.43	Definition of Screening U-Plug – Equation Setup	93
4.44	Definition of Screening U-Plug – Statement of Equation	93
4.45	Definition of Screening U-Plug – Equation Listing	94
4.46	Definition of Screening U-Plug – VB.NET Code Generation	94
4.47	Final Model Window for a Flocculation & Coagulation U-Plug	95
4.48	Final Model Window for a Sedimentation U-Plug	95

4.49	Output Calculation Sheet	96
4.50	GIS Representation of a Calculated Moment	96
4.51	Calculated Moment Values in GIS Environment	97
4.52	Using the Spline Tool for Spatial Interpolation of Moments	97
4.53	The Spatial Distribution of Calculated Moments	98
4.54	Contoured Representation of Calculated Moments	98
4.55	3D Representation of Calculated Moments	99
4.56	WISAM's Online Websites	100

Acronyms

DSS	Decision Support System
IDE	Integrated Design Environment
GIS	Geographical Information System
GUI	Graphical User Interface
MDI	Multiple-Document Interface
OMG	Object Management Group
OOP	Object-oriented programming
OOS	Object Oriented Software
RS	Remote Sensing
SMART	Specific, Manageable, Achievable, Relevant, Time-related
SWOT	Strengths, Weaknesses, Opportunities, Threats
UML	Unified Modelling Language™
URL	Universal Resource Locator
VB	Visual Basic©
WISAM	Wastewater's Interactive & Simplified Analysis Model
WTP	Water Treatment Plant
WTU	Water Treatment Unit
WWT	Wastewater Treatment
WWTP	Wastewater Treatment Plant
WWTU	Wastewater Treatment Unit

Abstract

This research work concentrated on developing a computer-aided Decision Support System (DSS) to facilitate design tasks and aid decision support of water and wastewater treatment units and processes. The developed system is entitled *Wastewater's Interactive & Simplified Analysis Model (WISAM©).*

The formulated DSS has been modelled and developed using Microsoft Visual Basic.NET programming language, with thorough implementation of Object Oriented Programming (OOP) guides, as well as the conceptual aid of Unified Modelling Language (UML) paradigms. The developed programme structure has been formed in such a dynamic package that eases future inclusion, addition and update of individual Water Treatment Units and Wastewater Treatment Units (WTU/WWTU). Ultimately, it allows designing, simulating and optimizing WTU/WWTU performance for all defined processes and units.

The results obtained from running the software on some selected WTU/WWTU types can be exported to a database file, which can be opened in a GIS/Spreadsheet/database management software (such as ESRI ArcGIS, MS Excel, and MS Access respectively). An online website has also been launched as an integration with WISAM software for interactivity and communication with interested environmental engineers and research scientists in the field. This website platform can be accessed 24 hours a day, 7 days a week for the days of the year via URLs: *http://www.sourceforge.net/projects/WISAM* or *http://wisam.sourceforge.net.*

Validation criteria have been met in compliance with Software Engineering and Development methods. Validation of such software via verifying its logic has been the governing concept of confidence.

The research concluded, among many other points, to emphasise the fact that WISAM have a strong potential in leading the modern concept of WTU/WWTU analysis and design; with its unique structure as a DSS platform. Finally, a strong recommendation has emerged from the research findings for adopting WISAM's capabilities by engineers and practitioners in the field. Additionally, further recommendations have included several enhancement points that could add to WISAM's future expansion.

المستخلص

ركز هذا البحث على تطوير نظام لدعم القرار بمساعدة الحاسوب (DSS) وذلك لتسهيل مهام التصميم وللمساعدة في اتخاذ القرار المتعلق بعمليات ووحدات معالجة المياه والمياه العادمة. سُمي النظام المطور "*الأنموذج التفاعلي المبسط لتحليل وحدات معالجة المياه (WISAM)* ".

استُخدِمت لغة البرمجة Microsoft Visual Basic. NET لصياغة وتطوير الأنموذج الحاسوبي لنظام دعم القرار، وذلك بالتنفيذ الدقيق لأدلة البرمجة غرضية التوجه (OOP)، فضلاً عن المساعدات المفاهيمية لأطر لغة النمذجة الموحدة (UML). وقد شُكِّل هيكل البرنامج المطور ووُضِع بالصورة المرنة التي تعين أي إدراج مستقبل أو أي إضافة وتحديث للوحدات المستغلة لمعالجة المياه والمياه العادمة. وبنهاية المطاف يتيح البرنامج المطور تصميم وحدات التنقية وعمليات المعالجة ومحاكاتها وتحسين الأداء لجميع العمليات والوحدات المحددة.

إن النتائج التي حُصل عليها من تشغيل البرنامج على بعض الأنواع المختارة من وحدات معالجة المياه والمياه العادمة يمكن تصديرها إلى ملف لقاعدة بيانات. ومن ثم يمكن فتح قاعدة البيانات هذه في نظم المعلومات الجغرافية أو برامج الجدولة أو إدارة قواعد البيانات مثل برنامج ESRI ArcGIS، أو MS Excel أو MS Access على الترتيب. إضافة لذلك، فقد أُطلق موقع على شبكة الانترنت بصفة تكاملية مع برنامج WISAM للتفاعل والتواصل مع المهندسين البيئيين والعلماء الباحثين المهتمين بهذا المجال. إذ يمكن الوصول إلى منصة هذا الموقع خلال ساعات اليوم الأربع والعشرين طيلة أيام الأسبوع السبعة بعدد أيام السنة عبر الرابط التالي:

http://wisam.sourceforge.net/ أو *http://www.sourceforge.net/ projects/WISAM* .

لقد استوفيت معايير التحقق من صحة الأداء امتثالاً لطرق هندسة وتطوير البرمجيات. كما جرى التحقق من صحة مثل هذه البرامج عن طريق التأكد من أن المنطق فيها هو الحاكم لمفهوم الثقة.

بالاضافة لعدة نقاط جوهرية أخرى، فقد خلُصَ البحث للتأكيد على حقيقة أن برنامج WISAM له قدرة فائقة لقيادة المفهوم الحديث لتحليل وتصميم وحدات تنقية المياه وعمليات معالجة المياه العادمة، وذلك بهيكله المتفرد كمنصة لنظام دعم القرار بمساعدة الحاسوب. ختاماً، أوصت الدراسة بتبني إمكانيات برنامج WISAM بواسطة المهندسين والمختصين في المجال. كما اشتملت التوصيات على العديد من أوجه التحسين الذي يمكن إدخاله خلال التوسعة المستقبلية للبرنامج.

CHAPTER ONE

INTRODUCTION

1.1 Background

Several procedures exist for designing treatment facilities and sequencing treatment processes. Mainly four stages could be depicted for developing a treatment decision support system (DSS). These stages cover the analysis and interpretation of the given problem, developing the reasoning models, actual decision support and usability; further to validating and verifying the DSS logic. The first stage can be problem specific, such as the concern about a specific contaminant, treatment process, or by having generic analysis such as batch processing of different contaminants removal. The second stage includes the numerical representation of gathered knowledge from the first stage and developing the reasoning models of the same. The third stage incorporates the optimisation of all factors for the generation and evaluation of best possible alternatives in the form of an actual decision support. The final stage “ensures usability by validating and verifying the DSS logic, as well as enhancing user interactivity with the developed DSS” (Hamouda, 2009).

Wastewater treatment (WWT) and reuse is a key factor in water sustainability. On one hand, appropriate structural and hydraulic design of retaining structures, holding tanks and treatment facilities; is a major stage towards attaining an efficient treatment plant. On the other hand, finance and cost impact is an important guide while assessing the adequacy of such designs (Rowe and Abdel-Magid, 1995).

Mathematical analysis and computational modelling proved to be of great value in terms of design optimization for a better cost saving. This aspect serves as a powerful tool in strategic infrastructural development and decision making. Therefore, a need for well-studied engineering solution is deemed necessary.

There are many factors dominating hydraulic and structural design of aqueous retaining structures. Some of which are the dynamic factors of design (such as damping factor, thermal effect, volume change, WWT unit’s shape, etc.). Combining various dynamic factors causes a very complicated course of analysis and design. Hence, the need emerges for developing an engineering concept; capable of delivering optimum results with respect to the effect of each studied factor.

1.2 Research Objectives

1.2.1 General Objective

The main objective of this research work is to develop sound decision support systems (DSS) for the appropriate selection of water and wastewater treatment units and trains; in an effort to address an integrated approach towards analysis of various factors affecting the design of water/wastewater treatment units.

The endeavours to offer an answer for the question *"What is the optimum implementation of software engineering paradigms for the hydraulic and structural design of wastewater treatment units"*, with special focus on developing a relevant Decision Support System (DSS).

1.2.2 Specific Objectives

In order to outline research objectives and aims the SWOT[1]/SMART[2] matrix analysis procedure was adopted. Strength, weaknesses, opportunities and threats are as presented in table 1.1.

Table 1.1: SWOT Analysis for The Research Project

Strength	**Weaknesses**	**Opportunities**	**Threats**
Availability of qualified expertise for supervision	Lack of holistic computer programs addressing research problem	New research idea	Publishing similar work worldwide
Availability of appropriate, updated & valid computer programs	Absence of an integrated technical cadre	Support of caring companies	Launching a similar design platform by software enterprises during research handling stage
Global research field	Lack of suitable funds for the work	Possibility of selling generated program	

1 Strengths: any internal asset, resources and capabilities that can be used as a basis for developing competitive advantage, value proposition and fight off threats.
Weaknesses: the absence of certain strengths, resources or capabilities which would be necessary to be competitive and distinguish from the competitors. They address internal deficits hindering the organization in meeting demands.
Opportunities: new opportunities that exist in the external environment. They shoulder any external circumstance or trend that favors the demand for an organization's specific competence.
Threats: changes in the external environment which represent threats to the company. Threats are for any external circumstance or trend which will unfavorably influence demand for an organization's competence.

2 Specific (Significant, stretching, simple, sustainable): identify your target clearly and how you would recognize if/when it has been achieved.
Manageable (Motivational, manageable, meaningful): within the situation in which you are working.
Achievable (Appropriate, agreed, assignable, attainable, actionable, action-oriented, adjustable, ambitious, aligned with corporate goals, aspirational, acceptable, aggressive): that is within your reach.
Relevant (Result-based, results-oriented, resourced, resonant, realistic, reasonable): to your situation and professional development needs.
Time related (Time-oriented, time-framed, timed, time-based, time boxed, time-specific, timetabled, time limited, time/cost limited, trackable, tangible, timely, time-sensitive, timeframe): so that there is a commitment to review progress and avoid slippage.

The specific objectives of the research work undertaken are to:

1) Delineate a systematic procedure for hydraulic and further structural design of selected types of WWT units; according to approved standards.
2) Use computational and integrated approaches to derive optimization scenarios of analysis and design for selected WWT units.
3) Formulate a mathematical model in form of a dynamic and upgradeable computer programme to serve as a scalable platform with capabilities to analyse, design, and optimize the design of WWT units.
4) Attempt to produce a competent computer package through use of state of the art programming approaches.
5) Disseminate research findings to the public domain arena for collaborative knowledge sharing and software evaluation and upgrading through a dynamic internet domain.

1.3 Research Hypotheses and Research Questions

The research hypotheses evolve around the following phrases:

1) If an efficient and user-friendly model can produce reliable results, then it would provide authorities and decision makers with solid bases for better policy making, decision taking and master planning.
2) If the hydraulic and structural design of WWT units could be gathered in one computer model; then the current efforts of inter-portability between different software packages would be eliminated.
3) If mathematical and statistical design approaches proved to be effective then a computer model would ease the evaluation of designed WWT units.
4) If a WWT unit design is properly optimised, then construction cost would be decreased.

1.4 Research Methodology

The methodology to be adopted in this research work will follow an analytical approach, supported by sound engineering theories, appropriate modelling and software development mechanisms. Developed computer model will use standard Decision Support System development procedures and sophisticated computer languages. The software will emphasise the application of International design standards in absence of any Sudanese Engineering code of practice for water and wastewater treatment and disposal.

Aiding and helping software for formulating the model will include many non-proprietary software (such as LibreOffice, XMind, AgroUML).

1.5 Expected Outcomes

Main expected outcomes include the following:

1) Comprehensive, robust, and friendly-to-use DSS software that will host various design modules for the hydraulic and structural engineering design of waste-water treatment plants.
2) Design sheet reports for individual treatment units including calculation logs and related design drawings.
3) Validation procedure for rechecking design quality.

1.6 From Vision to Action (Beneficiaries and Stakeholders)

The expected beneficiaries and stakeholders to utilize effectively the outcomes of this research work would include the following:

1) Design firms, organizations, institutions, municipalities and related ministries.
2) Specialized engineering societies.
3) Design and practising engineers.
4) Engineering staff, researchers and students.
5) Treatment plant maintenance engineers.

1.7 Project Plan and Time Frame

An action plan is a detailed plan outlining actions needed to reach one or more goals. It is an organizational strategy to identify necessary steps towards a goal. Start action planning with SWOT analysis which is a tool for auditing as a part of the strategic planning process and helps to focus on key issues. Once key issues have been identified, objectives can be formulated. Strengths and weaknesses are internal while opportunities and threats are external factors.

SWOT helps looking at the balance between strengths and weaknesses in a given situation. Therefore, it helps recognizing developmental needs. Then, plan of action must be expressed to meet those developmental needs. This can be achieved by setting targets considering SMART (Abdel-Magid and Abdel-Magid, 2014).

The proposed research plan consists of literature review, information data collection & analysis, computational model formulation & operation, design optimization & development, report write-up, and panel evaluation.

The research period was though over a three-year project time line was divided to multiple tasks as delineated in the following action sequencing work steps:

1. **Literature review** of water and wastewater treatment units, related design concepts, decision support systems, computer software and related programs. WWT unit types, along relative hydraulic and structural design concepts in conformity with international design codes and standards, as well as computational modelling and programming aspects. Proposed task duration is 359 days.
2. **Information & data collection** concerning a selected case study. Proposed task duration is 240 days.
3. **Data analysis and model formulation** using Visual Basic programming language and others. Proposed task duration is 261 days.
4. **Model operation** under a commonly used operating system platform (namely MS Windows), **and design troubleshooting and validation** of developed software. Proposed task duration is 223 days.
5. Continuous **thesis writing** through all research stages and up to **final report compilation**, with an overall duration of about 550 days.
6. Examiners **panel evaluation** would take place upon project completion.

Figure 1.1 illustrates the proposed research project schedule for the period between January 2010 and October 2012.

However, major restructuring of the intended DSS core and concept had taken place due to vital feedbacks received while discussing the project philosophy and approach in the presented papers among several workshops (national and international) [3], as well as the conducted lengthy dedicated seminars to this project[4]. The result of such modifications and incorporation of comments had led the research project to span for a longer period (between January 2010 and 2015).

[3] Workshop of Planning, Information & Knowledge Development for Eastern Nile Capacities, organized by Eastern Nile Technical Regional Office (ENTRO) of the Nile Basin Initiative (NBI), Addis Ababa, Ethiopia, 10th to 20th Oct. 2011.
ENPM First National Workshop in Sudan, organized by ENTRO-NBI and University of Khartoum, Khartoum, Sudan, 22nd to 24th May 2012.
ENPM First National Workshop in Egypt, organized by ENTRO-NBI and Cairo University, Alexandria, Egypt, 09th to 12th Jul. 2012.
ENPM Third Regional Workshop, ENTRO-NBI, Mekelle, Ethiopia, 23rd to 27th Sep. 2012.
ENPM Fourth Regional Workshop, ENTRO-NBI, Khartoum, Sudan, 18th to 22nd Nov. 2012.

[4] PhD Seminar on Conceptual Optimization of Dynamic Factors Affecting the Structural Design of Wastewater Treatment Units, held at Badi lecture hall, College of Water and Environmental Engineering, Sudan University of Science and Technology (SUST), Khartoum, Sudan, 15/05/2012.

WBS	Name	Start	Finish	Work	Duration
1	**Literature review**	**Jan 19**	**Jun 5**	**508d**	**359d**
1.1	Types of WWT units	Jan 19	Jun 19	108d	108d
1.2	WWTU Hydraulic design concepts	Apr 20	Sep 6	100d	100d
1.3	WWTU Structural design concepts	Jul 19	Dec 5	100d	100d
1.4	International standards & codes of practice	Oct 18	Mar 6	100d	100d
1.5	Computational modelling & programming	Jan 17	Jun 5	100d	100d
2	**Information & data collection**	**May 1**	**Mar 30**	**240d**	**240d**
2.1	Information & data collection	May 1	Mar 30	240d	240d
3	**Data analysis & model formulation**	**Sep 29**	**Sep 28**	**380d**	**261d**
3.1	Data breakdown analysis	Sep 29	Apr 26	150d	150d
3.2	Computational model formulation	Nov 13	Mar 2	80d	80d
3.3	Software programming	Mar 5	Sep 28	150d	150d
4	**Model operation & design optimization**	**Oct 1**	**Aug 6**	**223d**	**223d**
4.1	Model operation and verification	Oct 1	Jan 23	83d	83d
4.2	Optimization of selected design factors	Jan 24	Aug 6	140d	140d
5	**Thesis writing**	**Sep 7**	**Aug 6**	**500d**	**500d**
5.1	Thesis writing	Sep 7	Aug 6	500d	500d
6	**Final report compilation & project presentation**	**Aug 7**	**Oct 8**	**55d**	**45d**
6.1	Final report compilation & editing	Aug 7	Oct 8	45d	45d
6.2	Presentation & discussion	Sep 25	Oct 8	10d	10d
7	**Panel Evaluation**	**Oct 9**	**Oct 20**	**9d**	**8d**
7.1	Panel discussion	Oct 9	Oct 9	1d	1d
7.2	Implementation of examinars' final comments	Oct 9	Oct 20	8d	8d
8	Project completion (2012 Oct 01)	Oct 20	Oct 20	N/A	N/A

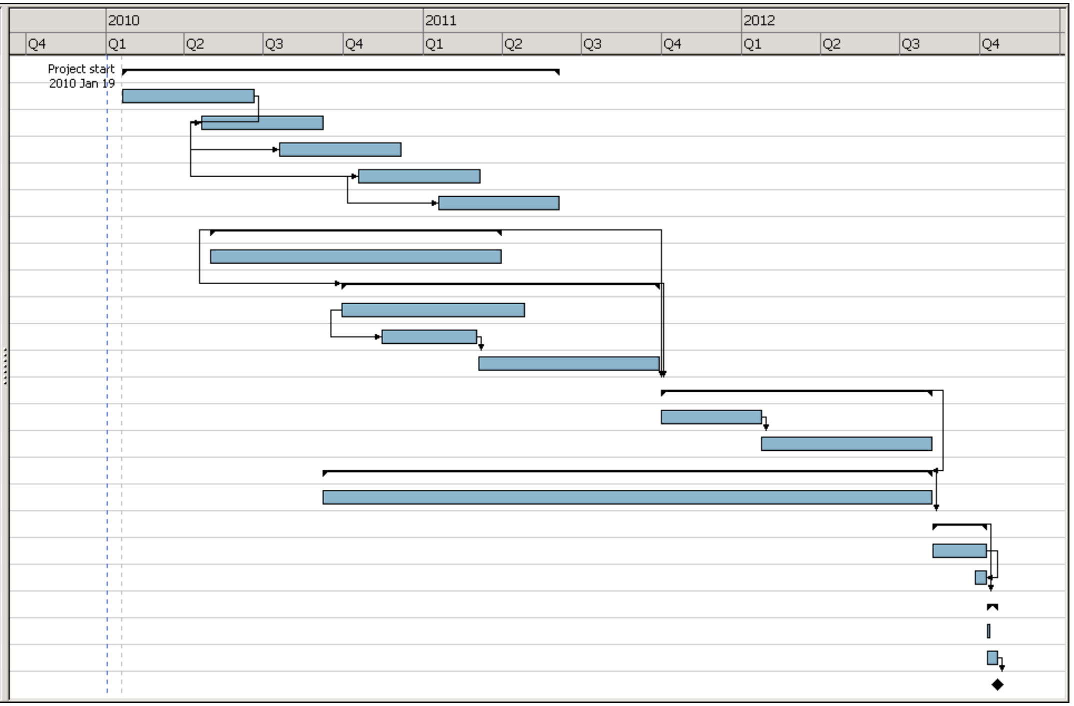


Figure 1.1: Research Project Schedule

CHAPTER TWO

LITERATURE REVIEW

2.1 Preamble

A decision support system, DSS, is an interactive software-based system used to help decision-makers compile useful information from a combination of raw data, documents, and personal knowledge; to identify and solve problems; and to make an optimized decision. The DSS architecture consists of the database (or knowledge base), the model (i.e., the decision context and user criteria), and the user interface. The main advantages of using a DSS include examination of multiple alternatives, better understanding of the processes, identification of unpredicted situations, enhanced communication, cost effectiveness, and better use of data and resources (Rinaldi and He1, 2014).

The application DSS and its development as a decision support tool for hydraulic and structural design of water & wastewater treatment units would allow holistic design approaches, support rapid assessment of treatment systems and decision-making at multi levels, better water management at various sectors and upgrade quality. This research work attempts to present the application of DSS in water and wastewater practices, and to future improvements and fostering of design standards, protocols and codes of practices

The development and application DSS and other metaheuristics for the optimisation of water and wastewater systems synthesizes shared problem traits, common engineering design challenges, and needed advances across major water and wastewater applications.

Conceptual design of water and wastewater treatment flow sheets entails choice and arrangement of relevant technologies to formulate sets of reasonable treatment options and scenaris that focus on appointed purposes. Accordingly, a design methodology is advocated to assist engineers and designers in the improvement, effectiveness and creativeness of conceptual designs of water and wastewater treatment units and processes. As recommended by Freitasa et al (2000) this hierarchical process design package can be described as an expert system coupled to a relational database and external programs integrated as a knowledge-based management system.

This research work clarifies the directions for better solving water and wastewater treatment design problems using this suggested platform. DSS applications to real-world problems require understanding fitness properties and their effects on DSS performance, focus on problem formulations and decompositions, understanding DSS theoretic frameworks and computational efficiency, and aiding real decision-making in complex, uncertain application contexts. The major pillars of this research work employed Object

Oriented Programming (OOP), Unified Modelling Language (UML), and geographical information systems (GIS). Visual basic integrated development environment (IDE) framed the backbone of programing hub along with some advanced Microsoft Excel spreadsheet creation. The chapter explored main objectives of DSS together with related framework as targeted to decision support systems in water treatment plants through modelling of units and processes.

Object-oriented programming (OOP) is a programming concept that denotes concepts as "objects" that have data fields (attributes that define the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs (Black, 2013). Not all of the essential characteristics and concepts appear in all object-oriented programming languages. For instance, OOP that uses classes is sometimes referred to as *class-based programming,* while *prototype-based programming* does not typically use classes. Therefore, a considerably different yet analogous terminology is exploited to outline the concepts of object and instance. Pierce (2004) view as futile any attempt to distil OOP to a minimal set of features. Nonetheless, he identifies fundamental features that support the OOP programming style in most object-oriented languages as:

- Dynamic dispatch – when a method is invoked on an object, the object itself determines what code gets executed by looking up the method at run time in a table associated with the object. This feature distinguishes an object from an abstract data type (or module), which has a fixed (static) implementation of the operations for all instances.
- Encapsulation (or multi-methods, in which case the state is kept separate)
- Subtype polymorphism.
- Object inheritance (or delegation).
- Open recursion – a special variable (syntactically it may be a keyword), usually called this or self, that allows a method body to invoke another method body of the same object. This variable is late-bound; it allows a method defined in one class to invoke another method that is defined later, in some subclass thereof.

Similarly, Mitchell (2003) identifies four main descriptions: dynamic dispatch, abstraction, subtype polymorphism, and inheritance. Scott (2006) considers only encapsulation, inheritance and dynamic dispatch. Additional concepts used in object-oriented programming include: classes of objects, instances of classes, methods which act on the attached objects, message passing and abstraction.

Unified Modelling Language (UML) is a diagramming language or notation to specify, visualize and document models of Object Oriented Software systems, OOS. UML as it is controlled by the Object Management Group (OMG) and is the industry standard for describing a software “graphically”. UML is designed for OOS design and has limited use for other programming paradigms. The UML model elements are used to create diagrams, which represent a certain part, or a point of view of the system. (Hensgen, 2003). The following types of diagrams are supported by Umbrello UML Modeller (Hensgen, 2003):

- *Use Case Diagrams* show users of the system use cases (the scenarios when they use the system), and their relationships.
- *Class Diagrams* show classes and relationships between them.
- *Sequence Diagrams* show objects and a sequence of method calls they make to other objects.
- *Collaboration Diagrams* show objects and their relationship, putting emphasis on objects that participate in the message exchange.
- *State Diagrams* show states, state changes and events in an object or a part of the system.
- *Activity Diagrams* show activities and the changes from one activity to another with the events occurring in some part of the system.
- *Component Diagrams* show the high level programming components (such as KParts or Java Beans)..
- *Deployment Diagrams* show instances of components and their relationships.
- *Entity Relationship Diagrams* show data and relationships and constraints between data.

System developers have used Unified Modelling Language (UML) to specify, visualize, construct, and document systems. Essentially, it enables one to communicate solutions in a consistent, tool-supported language. Today, UML has become the standard method for modelling software systems. UML is a visual language for capturing software designs and patterns capturing and expressing relationships, behaviours, and high-level ideas in a notation that's easy to learn and efficient to write. UML is visual; just about everything in it has a graphical representation. Various UML elements as well as their representations are summarized herein (Pilone, 2005):

Diagrams: UML 2.0 divides diagrams into two categories: *structural diagrams* and *behavioural diagrams*. Structural diagrams are used to capture the physical organization of elements in the system relating objects. Structural diagrams include:_class, component, composite structure, deployment diagrams package, behavioural, activity, communication, interaction overview, sequence, state machine, timing and use case diagrams.

- Class diagrams use classes and interfaces to capture details about entities that make up the system and the static relationships between them. Class diagrams are one of the most commonly used UML diagrams.
- Component diagrams show the organization and dependencies involved in implementation of a system. They can group smaller elements into larger deployable pieces with varying details.
- Composite structure diagrams: As systems become more complex, the relationships between elements grow in complexity as well. Conceptually, these structure diagrams link class diagrams and component diagrams; they don't emphasize the design detail that class diagrams do or the implementation detail that component structures do. Instead, composite structures show how elements in the system combine to realize complex patterns.
- Deployment diagrams show how the system is actually executed and assigned to various pieces of hardware. They are typically used to show how components are configured at runtime.
- Package diagrams are really special types of class diagrams. They use the same notation but their focus is on how classes and interfaces are grouped together.
- Behavioural diagrams focus on the behaviour of elements in a system. They may be used to capture requirements, operations, and internal state changes for elements.
- Activity diagrams capture flow from one behaviour or activity, to the next. They are similar in concept to a classic flowchart, but are much more expressive.
- Communication diagrams are a type of interaction diagram that focuses on the elements involved in a particular behaviour and what messages they pass back and forth. Communication diagrams emphasize the objects involved more than the order and nature of the messages exchanged.

- Interaction overview diagrams are simplified versions of activity diagrams. Instead of emphasizing the activity at each step, they emphasize which element or elements are involved in performing that activity. The UML specification describes interaction diagrams as emphasizing who has the focus of control throughout the execution of a system.
- Sequence diagrams are a type of interaction diagrams that emphasize the type and order of messages passed between elements during execution. Sequence diagrams are the most common type of interaction diagrams and are very intuitive to new users of UML.
- State machine diagrams capture the internal state transitions of an element. The element could be as small as a single class or as large as the entire system. They are commonly used to model embedded systems and protocol specifications or implementations.
- Timing diagrams are a type of interaction diagrams that emphasize detailed timing specifications for messages. They are often used to model real-time systems. They have specific notation to indicate how long a system has to process or respond to messages, and how external interruptions are factored into execution.
- Use case diagrams capture functional requirements for a system. They provide an implementation-independent view of what a system is supposed to do and allow the modeller to focus on user needs rather than realization details.

The UML is a visual language for specifying, constructing, and documenting the artefacts of systems. It is a general-purpose modelling language that can be used with all major object and component methods, and that can be applied to all application domains (e.g., health, finance, telecom, aerospace) and implementation platforms (e.g., J2EE, .NET). Under the stewardship of the OMG, the UML has emerged as the software industry's dominant modelling language. (OMG, 2011).

The groupings provided by language units and their increments do serve to simplify the definition of UML compliance. The stratification of language units is used as the foundation for defining compliance in UML. Namely, the set of modelling concepts of UML is partitioned into horizontal layers of increasing capability called compliance levels. Compliance levels cut across the various language units, although some language units are only present in the upper levels. As their name suggests, each compliance level is a distinct compliance point. For ease of model interchange, there are just two compliance levels defined for UML Infrastructure (OMG, 2011):

- *Level 0 (L0)* - This contains a single language unit that provides for modelling the kinds of class-based structures encountered in most popular object-oriented programming languages. As such, it provides an entry-level modelling capability. More importantly, it represents a low-cost common denominator that can serve as a basis for interoperability between different categories of modelling tools.
- *Metamodel Constructs (LM)* - This adds an extra language unit for more advanced class-based structures used for building metamodels such as UML itself.

As noted, compliance levels build on supporting compliance levels. The principal mechanism used in this specification for achieving this is package merge. Package merge allows modelling concepts defined at one level to be extended with new features. Most importantly, this is achieved in the context of the same namespace, which enables interchange of models at different levels of compliance. For this reason, all compliance levels are defined as extensions to a single core "UML" package that defines the common namespace shared by all the compliance levels. Level 0 is defined by the top-level metamodel shown in figure 2.1.

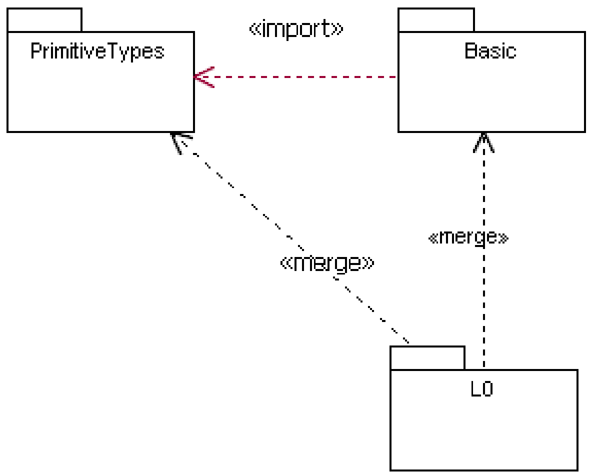


Figure 2.1: UML Level 0 package diagram (OMG, 2011)

In the L0 model, "UML" is originally an empty package that simply merges in the contents of the Basic package from the UML Infrastructure. This package contains elementary concepts such as Class, Package, DataType, Operation, etc. At the next level (Level LM),

the contents of the “UML” package, now including the packages merged into Level 0 and their contents, are extended with the Constructs package (see figure 2.2).
Note that LM does not explicitly merge Basic, since the elements in Basic are already incorporated into the corresponding elements in Constructs.

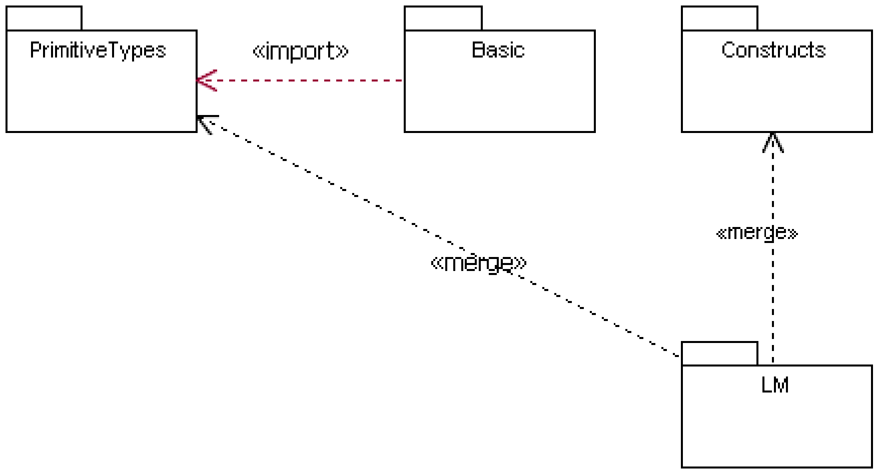


Figure 2.2: UML Level M package diagram (OMG, 2011)

A GIS is a computer system capable of capturing, storing, analysing, and displaying geographically referenced information; i.e. data identified according to location. Practitioners also define a GIS as including procedures, operating personnel, and spatial data that enter the system. The power of a GIS comes from its ability to relate different information in a spatial context and to reach a conclusion about this relationship. Most of the available information worldwide contains a location reference placing that information at some point on the globe. Different kinds of data in map form can be entered into a GIS. A GIS can also convert existing digital information, which may not yet be in a map form that can be recognized and used. A GIS can be used to emphasize the spatial relationships among the objects being mapped. Data capture - putting the information into the system - involves identifying the objects on the map, their absolute location on the Earth's surface, and their spatial relationships. Software tools that automatically extract features from satellite images or aerial photographs are gradually replacing what has traditionally been a time-consuming capture process. Objects are identified in a series of attribute tables - the "information" part of a GIS. Spatial relationships, such as whether features intersect or whether they are adjacent, are the key to all GIS-based analysis (Briggs, 2006).

2.2 Decision Support Systems in Water and Wastewater Treatment

Wastewater signifies a combination of the liquid or water-carried wastes removed from residences, institutions, commercial, and industrial establishments, together with such groundwater, surface water, and storm water as may be present (Metcalf, 2013). Wastewater contains impurities or pollutants in the form of solids, liquids or gases or their combinations in such a concentration that is harmful if disposed into the environment (Fredrick 1976, Lee and Lin 2000, Mara 2004, Davis and Cornwell, 2006, Karia and Christian, 2006, Be´line 2007, Guyer 2011, Hammer 2011, Benedetti 2012). Problems that are associated with unsuitable wastewater discharges include (Gerardi 2002, Iacopozzi *et al* 2007, Gallego *et al* 2008, L´opez *et al* 2008, Sala-Garridoa *et al* 2008):

- Introduction of diseases (via disease-causing agents), and other public health long-term physiological effects (by newly created organic substances).
- Accumulation of highly persistent detergents, pesticides and other toxic substances and compounds.
- Generation of taste, odour (e.g. carbon dioxide, hydrogen sulphide, methane gas, ammonia and other trace gases such as hydrogen, and nitrogen).
- Pollution by grease and oils which may render bathing sites unusable, or present extra problems for treatment works, or produce unsightly conditions, and interfere with the processes of biodegradation.
- Establishment of eutrophic conditions (enrichment of water by plant nutrients, etc.).
- Production of objectionable and dangerous levels of solids on bottom areas of water courses or along their banks. A condition may lead to degradation of water quality.

The increasing concern regarding environmental destruction and pollution has produced a growing awareness of the need for more effective wastewater Treatment Plants (WWTPs). The main goal of a WWTP is to reduce the pollution level of urban and/or industrial wastewaters, prior to discharge to the environment, stabilize organic pollutants, reduce number of disease-causing agents found in sewage, prevent pollutants from entering water sources, reduce odours and other nuisances resulting from sewage, water reclamation and reuse and by-product recovery and use (Rowe and Abdel-Magid, 1995). Wastewaters, containing basically solids, organic matter, nutrients and oils, are treated in successive stages within a WWTP. This is achieved through stages that incorporate pre-treatment (where influent wastewater is prepared for further treatment by removing debris, sand, rocks, gravel, etc.), primary treatment (separates the ready settleable and floatable solids from the wastewater stream) and secondary treatment stage (involves biological treatment to reduce soluble biodegradable organic matter from wastewater). Table 2.1 outlines major wastewater treatment units. Figures 2.3 and 2.4 illustrate a generalized flow diagram of treatment units and options (McCabe *et al*, 2004, Nathanson, 2007, Nemerow *et al* 2009, Abdel-Magid 2014).

Table 2.1: Major Wastewater Treatment Units

Preliminary treatment	
Screening	Bars, mesh or strainer to remove large solids
Grit removal	Removing grit and inorganic matter (e.g. sand) but not organic matter.
Storm overflow	Diverting sewage in excess of treatment plant capacity to storm water holding tanks.
Primary treatment	
Primary sedimentation	Settling of suspended solids (only 40 to 60 percent are removed), no chemicals are added.
Secondary treatment	
Aerobic oxidation of organic matter	Biodegrading organic matter through the action of microorganisms in a biological treatment unit such as: activated sludge or trickling filter plant, etc.
Secondary sedimentation	Settling out of sludge containing microorganisms to produce a treated effluent.
Tertiary treatment (Effluent polishing) (Advanced Treatment)	
Finalizing treatment	Polishing of effluent by operations such as sand filters, micro-strainers, etc.
Sludge treatment	
Anaerobic digestion	Decomposing thickened sludge in absence of oxygen.
Gravity thickening	Thickening of primary and secondary sludge.
Mechanical dewatering	Removing water from sewage sludges by methods such as centrifuges, pressure or vacuum filters, etc.
Drying beds	Drying sewage sludge in open atmosphere.
Sludge disposal	
Composted to be used as a soil conditioner (Digested only). Dumped at sea (Undigested). Incinerated (Normally undigested but thickened). Landfilled (Preferably digested and dewatered or dried).	

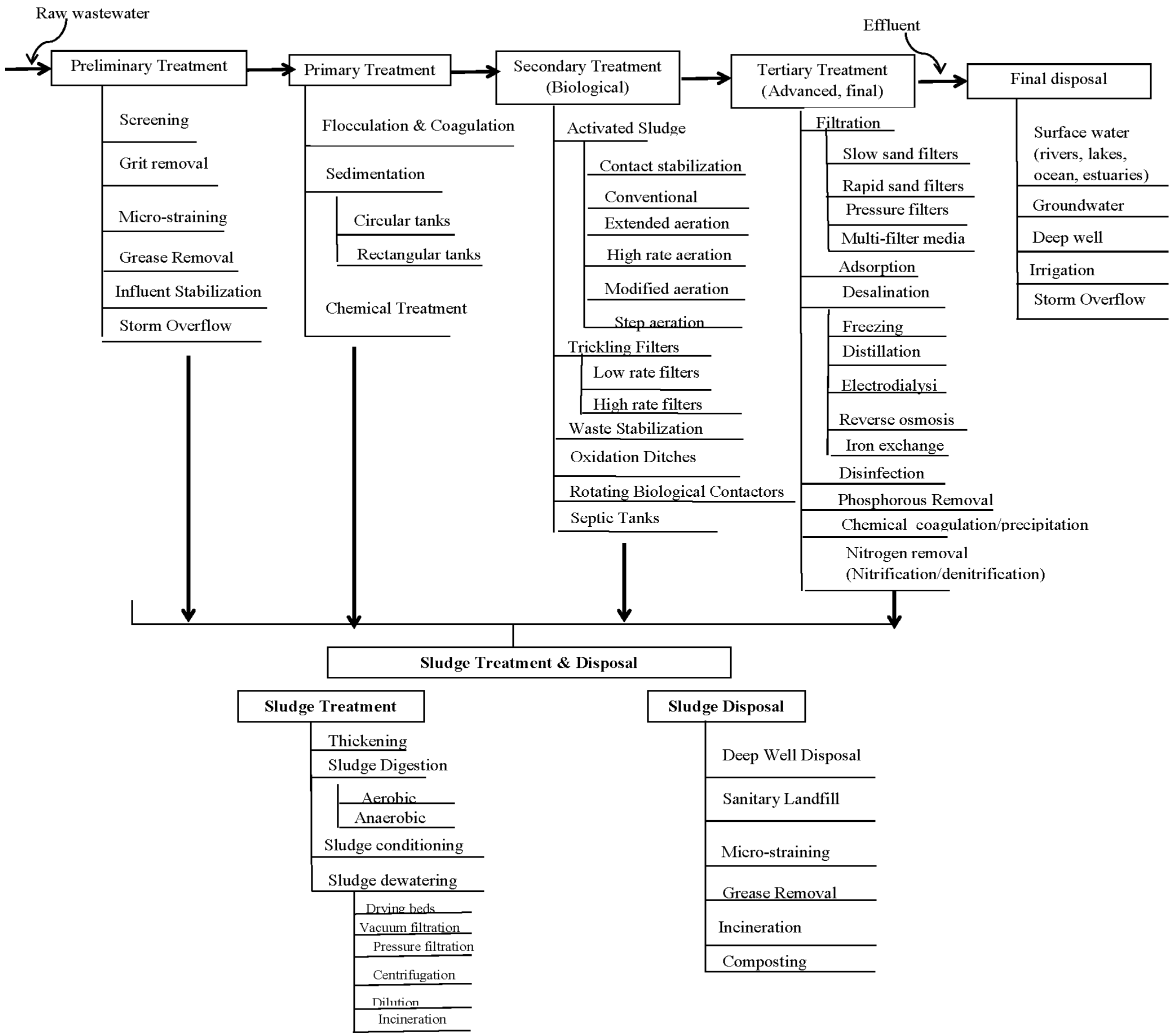


Figure 2.3: Wastewater Treatment Units

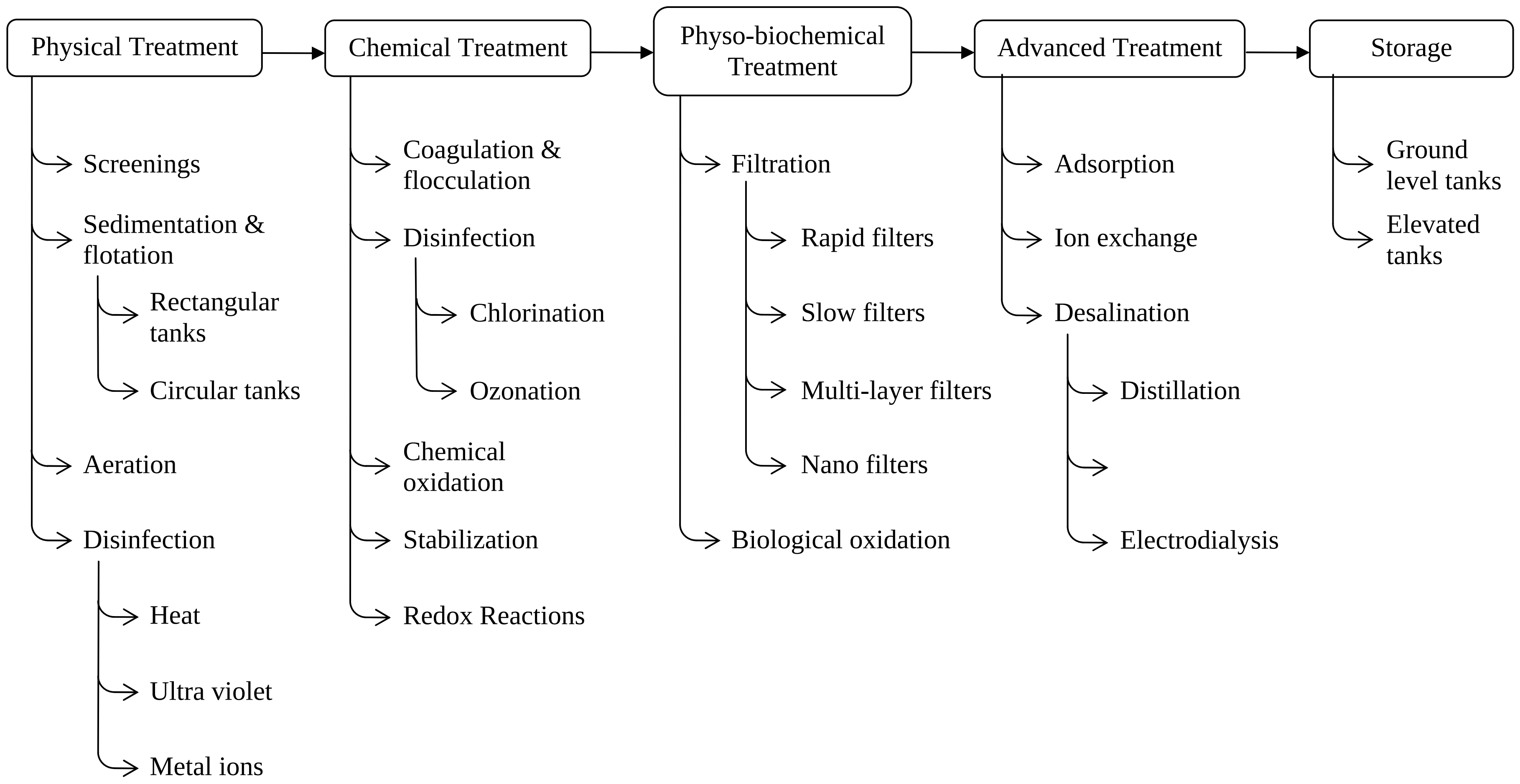


Figure 2.4: Water Treatment Methods

2.3 Wastewater Treatment Plant (WWTP) Design Limitations

Designing new wastewater treatment plants, WWTP, and upgrading existing ones are common environmental, chemical and civil engineering tasks. Complete WWTP design is a complex problem for several reasons that incorporate (Puig *et al* 2008, Viessman *et al* 2008, Burger *et al* 2011, Sin *et al* 2011, Shahriari *et al* 2012, Zarghami and Akbariyeh, 2012, Stefanakis and Tsihrintzis, 2012):

- Influence of a large number of multiple factors and parameters affecting complete WWTP design.
- Reliable information on characteristics of wastewater to be treated is usually lacking.
- Design must ensure that regulation effluent requirements will be met under possible variable inflow, hydraulic and organic loadings and climatic conditions.
- Wastewater treatment involves many interrelated physical, chemical and biological processes. This augmented difficulties of obtaining generalized mathematical models describing such systems.
- Unavailability or scantiness of good estimates for some parameters required for biological models.
- Need to perform a simultaneous design for every treatment unit process before finalizing design of whole plant.
- Necessity to include influence of recycling pollutants from sludge processing stream to wastewater processing influent to get the complete WWTP sizing.
- Need to design each treatment unit to involve selection of associated equipment (such as aeration, mixing, heating, etc.). This is because the performance characteristics of commercial equipment can modify unit sizing.
- Need to use a trial-and-error design procedure until adjusting design of each treatment unit to required effluent and sludge standards.
- Large temporal variations which occur in wastewater composition, concentrations, and flow rates.
- Poor operation of most municipal wastewater treatment processes. Gross failures are all too frequent and there are significant variations in treatment plant efficiency, not only from one plant to another, but also on daily and hourly basis in same plant. Daily variations from 60 to 95 % efficiency in BOD removal are not uncommon and these variations can have a significant effect on water quality of a receiving stream (Andrews, 1974, Chen *et al* 2010).

2.4 Mathematical Modelling of WTU/WWTU

2.4.a Previous Software for WWTU Design and Limitations Incurred

Several relevant contributions in WWTP design computer programs have been developed. Examples of such software may include, but not limited to, the following:

- First approaches to EPA executive (Smith and Eilers, 1968), ESTHER-SPCHEN (Chen et al., 1972), and SEPSIM (Environment Canada, 1974).
- Improved computer tools focusing on development of user interface facilities and inclusion of more flexible treatment units sequences or cost estimations represented by: CAST (Chang and Liaw, 1985), CAP- DET (Getty et al., 1987), and the softwares of Spinos and Marinos-Kouris (1992) and Kao et al. (1993).
- DATAR software package developed for automated design of wastewater treatment plants. A user-friendly environment has been implemented to facilitate design tasks, allowing rapid evaluation of different alternatives as well as performing sensitivity analysis. Flexible treatment plant configurations can be established with preliminary, primary, biological and tertiary wastewater treatments, and sludge treatment units. The design process includes treatment units sizing, plant layout, hydraulic profile calculation and equipment assignment. Mathematical models describing treatment processes have been formulated taking into account the variation in waste quality parameters (Gabaldo´ n et al 1998).
- Ferrer et al (2008) presented DESASS (DEsign and Simulation of Activated Sludge Systems) to design, simulate and optimize wastewater treatment plants. The mathematical model implemented is the Biological Nutrient Removal Model which allows simulating the most important physical, chemical and biological processes taking place in treatment plants. DESASS calculates performance under steady or transient state of whole treatment schemes including primary settlers, volatile fatty acid generation systems by primary sludge fermentation, activated sludge systems for biological organic matter and nutrient removal, chemical phosphorus precipitation, secondary settlers, gravity thickeners and sludge digesters (aerobic and anaerobic). Biological conversions occurring in settlers and thickeners (primary sludge fermentation, de-nitrification) are also taken into account considered as reactive elements (Castro 2007, Hakanen 2011).
- In 2009, Hamouda has conducted a comparison between sixteen different models for either domestic water, wastewater, or industrial wastewater treatment units. The

comparison highlighted the used approach in each model (being a mere technical or incorporating an economic component, or a full system analysis); as well as the employed mathematical techniques and strengths of each model (refer to table 2.2).

- Fang et al (2010, 2011) integrated dynamic model developed through combining a mechanistic model, a neural network (NN) model and a genetic algorithm approach, in order to simulate performance of a full-scale municipal wastewater treatment plant (WWTP) with substantial influent fluctuations. As the base of the integrated model, the mechanistic model was initially established based on an activated sludge model and the EAWAG bio-P module, and was used to generate residuals for the NN model. The NN model was employed to build a relationship between input and output variables. The network weights of the NN model were optimized with a genetic algorithm approach. The model is demonstrated to be an effective and useful tool to simulate performance of WWTPs (Bumble 2000).
- Recently, it could be noticed from the emerging market that many companies has worked in the field of developing and producing software packages for various design tasks. The researcher (2015) is hereby naming a few of such packages for the illustration of potential competitors in the market; such as Aqua Designer & Aqua Aero (by BITControl GmbH, Schleid, Germany), CapdetWorks & GPS-X Pro (by Hydromantis Environmental Software Solutions Inc., Ontario, Canada), Backflow Pro (by Alpha-Omega Computers Inc., Florida, USA), ARTS hydraulic design software (by Aquavarra Research Limited, Dublin, Ireland), SASSPro (by HTI Systems LLC., Texas, USA), BioWin (by EnviroSim Associates Ltd., Ontario, Canada), WEST (by MOSTforWATER N.V., Belgium), and STOAT / Plan-It STOAT (by WRc Group, Wiltshire, UK).

Limitations of available software packages include the following:

- Limitations of all models to hydraulic design aspects with no structural components addressed or added.
- Omission of factors such as equipment design or climatic conditions.
- Absence of inclusion of advances attained in biological wastewater treatment modeling even though their applicability to design purposes is limited.
- They were not developed for complete design of a full-scale WWTP, with all the interrelations established.
- Difficulties in availability of model parameters, including wastewater characterization.

- Incurred cost of programs and software due to use of non-open sources in programming models.

Therefore, a missing gap of holistic modelling approach has been identified by this research work pertaining to the design of water/wastewater treatment units.

Table 2.2 Summary of Some Water Treatment Decision Support Systems as reviewed by Hamouda (2009)

Model name	Scope	Approach	Employed techniques	Strengths
-	WWT	Technical & economic	Rule-based, heuristic search, neural networks	Certainty factors for the developed rules
–	WWT	Technical & economic	Process modeling, mathematical programming	Solves mass balance on a treatment train Graphical display of designs
-	WWT	Technical & economic	Case-based reasoning, heuristic search	Define cost per unit removal of contaminant
–	IWWT	Technical design	Knowledge-based expert system	Allows user intervention during selection
SOWAT	WWT	Technical & economic	Rule-based, heuristic search, fuzzy logic	Fuzzy functions for technology performance Ability to check a user defined train
–	WWT	Technical & economic	Expert system, fuzzy logic	Certainty factor for technology treatability User defined fuzzy preference of technologies
MEMFES	IWWT	System analysis	Expert system, simulation, analytical hierarchy process	A tutor provides justification for outcome Surveyed the system's user-friendliness
–	WWT	Technical & economic	Simulation, issue-based information systems	Reports describe the deliberation over a decision Searching design records using keywords
SANEX	WWT	System analysis	Conjunctive elimination, multi-attribute utility technique	Multi-disciplinary set of sustainability indicators Multi-level amalgamation used for rating
–	IWWT	Technical & economic	Knowledge-based system, heuristic search	Easy update of process database Possible communication with other programs
WAWTTAR	DWT WWT	System analysis	Modelling and simulation, screening, multi-criteria decision analysis	Output: least cost alternative, assesses risk, and more Community specific data considered in the decision
WASDA	WWT	Technical design	Rule-based, design equations	Friendly user interface Process design calculation module
WADO	IWWT	Technical & economic	Rule-based, mixed integer non-linear programming	Investigates regeneration opportunities from water used in industrial processes
WTRNet	WWT	Technical & economic	Modelling & simulation, linear & NL programming, genetic algorithm	Provides user guidance for treatment train selection through either an expert or a stepwise approach
–	WWT	System analysis	Analytical hierarchy process, grey relational analysis	Allows comparison between alternatives considering the entire criteria
Zhu &	DWT	System analysis	Bayesian probability networks	Considers performance uncertainty Variables measuring impact on public health

2.4.b Mathematical Models and Computer Simulation

Mathematical models are commonly used for more quantitative description of process performance and consist of one or more equations relating the important inputs, outputs and characteristics of the process. Mathematical models may be classified in many different ways. One of the most important for wastewater treatment processes is the distinction between dynamic and steady state models. Most models currently in use are based on the assumption of steady state. Steady state models have proven their value on a qualitative basis by indicating needed changes in process design and also have the advantage of experimental and computational simplicity. However, in most instances they are not adequate to describe process operation since the inputs to the processes are far from constant and there is considerable variation in effluent quality with respect to time. Wastewater treatment processes should be modeled as dynamic systems since the reactor is stirred and contents are homogeneous, the concentration of tracer in the reactor and in the reactor effluent are identical. The reaction term is zero since the tracer is inert and does not participate in any reactions. When the flow rate and reactor volume are constant, the process can be classified as a first order, linear system with constant coefficients. The order is determined by the highest order derivative of the output and the system is linear since all derivatives and variables are raised only to the first power and there are no products of derivatives and/or variables. Most used mathematical models included the following:

1. **Empirical design criteria**: Design is performed to ensure that plant effluents (water and sludge) will meet regulation quality requirements. For this purpose, variation in water quality characteristics are evaluated at each treatment unit.
2. **Neural network (NN) approach:** The NN approach is a powerful and effective tool to deal with problems to extract information out of complex, non-linear data without requiring prior knowledge of the relationships of the process parameters. NN approach may be introduced into the mechanistic models to improve their simulating capacity. Because of its interpolative capability to capture effects of some external disturbances, NN approach has been successfully applied in multivariate non-linear bio-processes as a useful tool to construct model. However, the NN is typically used as a "black-box" approach, hiding the physics of the model process, and lacks for extrapolative capacity. In addition, the gradient algorithm usually used in the back-propagation NN is a local search algorithm and may tend to fall into a local minimum and result in inconsistent and unpredictable performance. Genetic algorithm (GA), based on the principles of survival of the

fittest strategy, has been proven to be a powerful search and optimization method to solve problems with objective functions that are not continuous or differentiable (Fang et al, 2010).

3. **Artificial Intelligence,** AI, methodology, is the use of artificial neural networks (ANN). ANNs are normally very effective to capture the non-linear relationships that exist between variables in complex systems, and can also be applied in situations where insufficient process knowledge is available to construct a white-box model of the system. AI is a research area that involves use of ANN, genetic algorithms (GA), fuzzy logic, rule-based systems, knowledge-based systems, ontologies, case-based systems, agents, etc. (Gernaey, et al, 2004, Hamed et al, 2004, Dellana et al, 2009, Fernandez et al, 2009, Chen et al 2007, Roda 2000).
4. **Dynamic mathematical models** are usually necessary for the description of time variant phenomena, as is commonly encountered in wastewater treatment processes. Models for different types of reactors can be developed by applying material and energy balances using the fundamental transport, stoichiometric, thermochemical and kinetic relationships. The models usually consist of sets of non-linear differential equations for which analytical solutions are not available. However, solutions to the equations or prediction of process performance with respect to time can be obtained by computer simulation. Dynamic modeling and computer simulation are useful tools in developing better procedures for process start-up, prediction and prevention of process failures, and improvement of process performance by consideration of dynamic behavior during both the design of a process and its associated control system. (Andrew, 1974).
5. **White-box models or deterministic models**, are based on first engineering principles with model equations developed from general balance equations applied to mass and other conserved quantities, resulting in a set of differential equations (Gernaey etal, 2004).
6. **Black-box models,** i.e. models entirely identified based on input–output data without reflecting physical, biological or chemical process knowledge in the model structure can be applied to trigger appropriate control actions in good time. Typical black-box model examples applied for time series modeling are autoregressive (AR) models, autoregressive moving average (ARMA) models, AR with external input models (ARX), ARMA models with external input (ARMAX) and Box–Jenkins (transfer function) models . The advantages of white-box and black-box

modeling can be combined in a hybrid modeling scheme. Hybrid model is a term that is used to designate models based on first engineering principles, where specific functionalities, e.g. reaction kinetics, have to be estimated from process data (Gernaey et al, 2004, Pons 2008).

After a dynamic mathematical model has been developed for a process, the equations which comprise the model must be solved in order to predict the behavior of the process with respect to time. This procedure is known as simulation and can be defined as the use of a model to explore the effects of changing conditions on the real system. Obviously, the model must be a reasonable representation of the real system in order for the results to be meaningful since the simulation results can be no better than the mathematical model and data on which they are based.

In developing mathematical models, it is desirable to iterate between model development, computer simulation, physical experimentation and field observations since these complement one another. Knowledge gained in simulation is useful for modifying the model, guiding physical experimentation, and establishing the type and frequency of field observations needed. This iterative technique also points out another important aspect of modeling and simulation, this being the need for model verification. Computer simulation can lead to the generation of large quantities of worthless results if the model is not a reasonable representation of the real process.

Mathematical models for designing physical and chemical treatment units adapted from the basic literature (Metcalf and Eddy, 2013; WEF, 2008a, 2008b, ASCE 2012, Gabaldo´ n 1998) are based on empirical design criteria, such as retention time, organic loading, hydraulic loading, etc. Special attention has been paid to biological processes modeling, both for wastewater treatment and sludge stabilization processes (WEF 1992, 2008, Ferrer et al. 1998, Uggetti et al 2011).

2.5 DSS Objectives and Framework

2.5a DSS Objectives

Water and wastewater treatment systems are complex and dynamic in nature. The challenge of treating water to a required quality level is influenced by the various interactions of factors impacting the effectiveness of a water treatment system. The design of a water treatment train will depend on water quality, regulatory requirements, consumer/environmental concerns, construction challenges, operational constraints, available treatment technologies, and economic feasibility. Although the purpose of the treatment system being developed may befor dr inking, domestic wastewater, or industrial wastewater treatment, the problem of designing an appropriate treatment system is similar. Basically a treatment train is composed of a series of processes and the number of such processes has been steadily growing, making the selection of an optimum sequence an important challenge faced by a designer (Hamouda et.al, 2009 and Joksimovic et al. 2006, Prat 2012, Rivas 2008).

The overall objective of a DSS project is to enhance the ability of core parties and stakeholders to quantify problems related to wastewater treatment, reuse and final disposal and to identify measures to be taken to improve the exiting situation. An added objective of the project is to facilitate a common water resources monitoring framework that will allow transfer and exchange of data and that will facilitate to identify, plan and analyze different reuse options. The key issue of the project is to improve the availability of wastewater treatment and reuse related information for water managers, planners, and operators in order to promote cooperation between the core parties. A main component of the project therefore is the development of a DSS that can match a wide range of wastewater quality resources to an ever increasing number of reuse options, mainly agriculture. In connection to these objectives the DSS project is contributing to increased capacity for field monitoring (water quality and water quantity) and development of the database (de Schutter, 2007).

Multi-criteria decision analysis methods provide a consistent framework in order to extend the Boolean overlays that are supported by software packages to the consideration of decision criteria as well. This integration provides to DMs a valuable tool that allows effective decision-making especially when different groups of interests participate in the process (Anagnostopoulos, et. al., 2010).

The core of the Knowledge-Based Decision Support System (KB-DSS) embraces two objectives. The first one is to assist in the selection of the treatment level adequate to fulfill the target quality standards for the receiving environment. The second one is to select the specific type of treatment (Comas et.al., 2003).

Hakanen et.al 2011 combined a process simulator to simulate wastewater treatment and an interactive multiobjective optimization software to aid the designer during the design process. They obtained a practically useful tool for decision support.

Information technology has played an increasing role in the planning, design, and operation of water treatment systems. A decision support system (DSS) is an information system that supports a user in choosing a consistent, near-optimum solution for a particular problem in a reduced timeframe. Environmental Decision Support Systems (EDSSs) have been presented as interactive, flexible and adaptable computer-based systems able to tackle these complex and illstructured domains. An EDSS can link numerical models/algorithms with knowledge-based techniques, geographical information systems and on-linedata, among other technologies. They have been developed to help environmental decision makers choose between alternatives (Poch et al., 2004).

Models can be used to test scenarios and evaluate failures (e.g. SS collapse), or to assess certain measures intended to improve the performance of the system against perturbations (e.g. increased hydraulic load). These models can also be used to evaluate realtime control. The results of the simulated scenarios provide the EDSS with relevant and useful knowledge about the management of the wastewater infrastructures (Muschalla, 2008).

Hamouda et.al, 2009 points out that the continuously changing drivers of the water treatment industry, embodied by rigorous environmental and health regulations and the challenge of emerging contaminants, necessitates the development of decision support systems for the selection of appropriate treatment trains. They determined that there is a need to develop integrated decision support systems that are generic, usable and consider a system analysis approach.

2.5b DSS Framework

The outline of the DSS has started from a combination of a systems analysis (model), the available database and model and the decision framework according to the global structure. A decision support system is helpful in situations where a decision depends on, or is influenced by, a large number of factors, rendering the decision procedure complex.

A properly designed DSS should provide an easy-to-use, usually graphics enhanced, working environment for the development, processing, and analysis of decision alternatives on the basis of a policy analysis framework (de Schutter, 2007).

Prat et.al., (2012) applied integrated modeling of an urban wastewater system (UWS) to simulate and analyze their behavior, and to optimize performance against different types of perturbations.

Hidalgo, et. al., (2007) stated that decision-making in environmental projects can be complex principally due to the inherent existence of trade-offs between socio-political, environmental and economic factors. They focused the aims of their project on the development of a software tool able to apply a scoring system for existing wastewater facilities based on the potential safe reuse of the final effluent. The scope of their work is to present the multi-criteria analysis user friendly software that has been developed. The tool is able to guide the responsible authorities to the most efficient solutions in terms of can be sustainable. The input data for the specific model are quite simple and can be easily collected (e.g. data concerning the population served by the facility, the possibilities for agricultural reuse of the water in the area, specific requirements or preferences on cultural, economical, technological or social issues), while the outcome is the ranking of the alternative scenarios and the suggestion of specific processes and treatment systems.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Research Methodology

In this research work, analytical and mathematical approaches shall be adopted. A software model is to be formulated by means of a combination between VisualBasic.NET programming language, advanced MS-Excel modelling, and GIS integration. The anticipated model algorithm creation is to follow the state of the art techniques in software engineering (namely the Object Oriented Programming and Unified Modelling Language paradigms). Thorough validation is to be conducted on the model against logical flaws and arithmetic errors.

3.2 Computer Hardware

In order to facilitate the design and operation of the planned Decision Support System, a certain computer setup has to be acquired. The researcher obtained the state-of-the-art IT technology for subject research work. Notwithstanding the same, the developed software has been programmed with the mind-set of compatibility with lower computer configurations without jeopardising the DSS functionality.

The following table (3.1) summarises various configuration ranges of computer hardware as involved in this research.

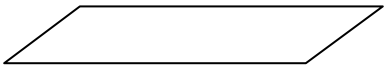
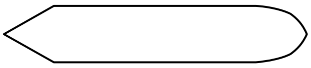
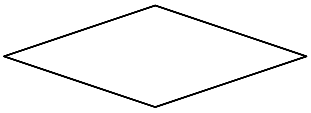
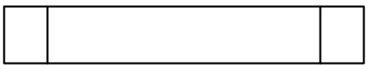
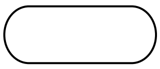
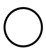
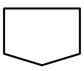
Table 3.1: Tested Computer Hardware Configuration Ranges

Configuration range	Processor Type	Processor clock speed	Memory	MS Windows version	.Net Framework version
Minimum	Intel© Core2Duo™ (Dual core)	2.0 GHz	2.00 GB DDR2 (677 MHz FSB)	Windows XP 32-bit (service pack 2)	2.0
Maximum	Intel© Core™ i7 (Quad core)	2.7 GHz	32.00 GB DDR3 (1666 MHz FSB)	Windows 8 64-bit	4.0

3.3 Software Functionality

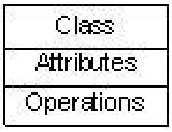
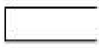
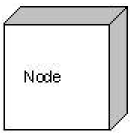
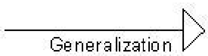
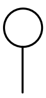
Graphical representation of software's functionality and flow sequence had always been the favourable to software developers. Such presentation of the software's workflow is governed by *Flow Chart* diagrams. The aim of using Flow Charts is to illustrate the software's sequential flow via understandable and standardised shapes rather than referring to a certain programming language among others. Another supporting point of using Flow Charts is the ease of understanding and interpretation of the application's core functionality. Additionally, logical errors would be spotted and adequately corrected in simpler fashion rather than digging in a bunch of code syntax. Table 3.2 lists some of the typical flow chart shapes as used in this research work.

Table 3.2: Typical flowchart shapes (NIIT, 2001)

Shape	Usage
	Data entry (inputs)
	Processes
	Results display (outputs)
	Decision structure for conditional evaluation (decisions)
	Subroutine, function, or sub-programme (call for external subroutine)
	Flow lines, to link the chart components in its logical flow order
	Start/End flow chart indicator
	On-page link
	Across-pages link

Since the flow chart diagrams cater for representing subject software's functionality, a need has emerged for another type of diagrams that targets the visual symbolization for specifying, constructing, and documenting the artefacts of computer systems. Hence, the developed Unified Modelling Language (UML) has been utilized in this research work. Emphasis was stressed to use the Class Diagrams form among various UML diagrams. A Class Diagram shows the used classes as being programmed in the software as well as the relationships between them. Table 3.3 lists some of typical UML Class Diagram shapes as used in this research work.

Table 3.3: Typical UML Class Diagram shapes (OMG, 2011)

Shape	Usage
	Class *(Class represents set of objects having similar responsibilities)*
	Interface *(Interface defines a set of operations which specify the responsibility of a class)*
	Collaboration *(Collaboration defines interaction between elements)*
	Node *(A node can be defined as a physical element that exists at run time)*
	Generalization *(Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects)*
	Dependency *(Dependency is a relationship between two things in which change in one element also affects the other one)*
	Implementation *(Implementation is a relationship which refer to the association of an Interface)*

3.4 Software Development and Implementation

In this research work, design efforts had their primary focus to build a genuine, robust and intact computer software platform. The following points were identified as the pillars of building the intended Decision Support System platform (DSS):

1. The primary use of the DSS is to perform both hydraulic & structural analysis and design of water / wastewater treatment units (WTU/WWTU).
2. The DSS should have the ability to define a WTU/WWTU design engine *on-the-fly*, i.e. without the need for prior knowledge of computer programming language by the engineering user.
3. The DSS must be able to link between its built-in design modules and external design engines, i.e. vide passing any required inputs from the DSS to external engine and read back the output results from such external module.
4. Explore the possibility of utilizing the potential use of Geographic Information Systems (GIS) in the field of civil engineering designs (as in structural analysis for instance).

Having the aforementioned pillars in mind, the researcher opted to take several design paths in order to manifest such theory into a working reality. Therefore, the software design and development efforts were split into the following paths:

1. **Software Development Path 'A': Visual Basic .Net**

   This path would be considered as the primary and most laborious task. In this path, the core DSS functionality would be developed using Visual Basic .Net 2010 (VB.Net) programming language. The outcome of this task would be a well-formulated and user-friendly computer software. As an illustration of its functionality, the DSS shall be used to generate a design engine for the hydraulic design of a selected WTU/WWTU. Furthermore, the DSS shall be equipped with a Report-generator engine that enables the documentation of the modelled WT/WWT unit(s).

   VB.NET is an object-oriented computer programming language that can be viewed as an evolution of the classic Visual Basic (VB), implemented on the .NET Framework. Microsoft currently supplies two main editions of IDEs for developing in Visual Basic: Microsoft Visual Studio, which is commercial software and Visual Basic Express Edition, which is free of charge (see figure 3.1). The command-line compiler, VBC.EXE, is installed as part of the freeware .NET Framework SDK. The utilised version in this research work is VB.NET 2010. The major data types that are being used in VB.NET programming environment is shown in table 3.4 (Halvorson, 2010).

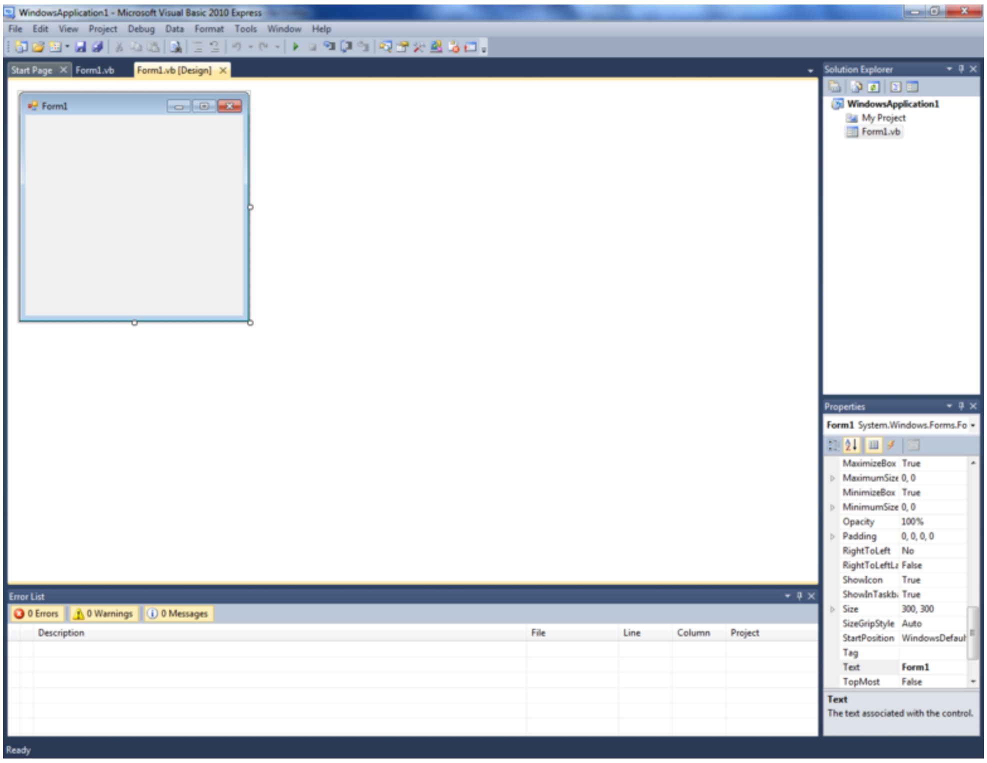


Figure 3.1: Microsoft Visual Basic.NET IDE

1. **<u>Software Development Path 'B': Advanced MS Excel</u>**

   As a by-product of this research effort, fully-functional and independent spreadsheets shall be produced using the advanced functions of Microsoft Excel 2010. The intent of the spreadsheets is to cater for the structural analysis of rectangular tanks according to the guidelines provided at the publication of the Portland Cement Association (Munshi, 1998) as well as its structural design as concrete tanks according to Part 3 of the Eurocode2, EN 1992 Eurocode2: Design of Concrete Structures, EN1992-3 Liquid Retaining and Containment Structures, with specific consideration of the UK National Annex to Eurocode (Reynolds *et al.* 2008 and Threlfall 2013). The aimed output of this task is to illustrate the DSS's ability of linking to external design engines outside the DSS modelling environment (as in the case of linking to an external spreadsheet file). The form of the link shall be in the ability to open, read, write-into, and retrieve information from the spreadsheet files.

Table 3.4: Visual Basic .NET Commonly Used Data Types

Data Type	Range	VB Sample Usage
`Short`	-32,768 through 32,767	`Dim Birds As Short` `Birds = 12500`
`Integer`	-2,147,483,648 through 2,147,483,647	`Dim Insects As Integer` `Insects = 37500000`
`Long`	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	`Dim WorldPop As Long` `WorldPop = 4800000004`
`Single`	-3.4028235E38 through 3.4028235E38	`Dim Price As Single` `Price = 899.99`
`Double`	-1.79769313486231E308 through 1.79769313486231E308	`Dim Pi As Double` `Pi = 3.1415926535`
`Decimal`	values up to +/-79,228 x 1024	`Dim Debt As Decimal` `Debt = 7600300.50`
`Char`	Any Unicode symbol in the range 0–65,535	`Dim UnicodeChar As Char` `UnicodeChar = "Ä"`
`String`	0 to approximately 2 billion 16-bit Unicode characters	`Dim Dog As String` `Dog = "pointer"`
`Boolean`	True or False (during conversions, 0 is converted to False, other values to True)	`Dim Flag as Boolean` `Flag = True`
`Date`	January 1, 0001, through December 31, 9999	`Dim Birthday as Date` `Birthday = #3/1/1963#`

2. **<u>Software Development Path 'C': GIS Integration</u>**

   Similar to Path 'B' of the software development in this research, another by-product shall be introduced in the form of an orphan module. Such endeavour would serve two purposes: one of which is the illustration of the DSS's ability of linkage to external executable codes even if written using a different language than VB.Net; and the other purpose is to introduce the potential of GIS application on the smaller level as in the design of concrete walls.

   ESRI ArcGIS Desktop for Windows is one of the powerful GIS tools in the arena. ArcGIS stores and manages geographic data in a number of formats. The three basic data models that ArcGIS uses are vector, raster, and TIN. Tabular data also can be imported into ArcGIS (ESRI, 2012).

a) Vector

Vector data models represent geographic phenomena with points, lines, and polygons.

Points are pairs of x,y coordinates, lines are sets of coordinate pairs that define a shape, and polygons are sets of coordinate pairs defining boundaries that enclose areas, see figure 3.2.

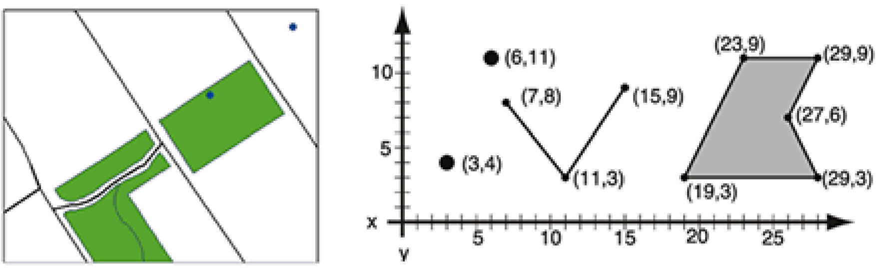


Figure 3.2: Vector Data Representation (ESRI, 2012)

Coordinates are most often pairs (x,y) or triplets (x,y,z, where z represents a value such as elevation). The coordinate values depend on the geographic coordinate system in which the data is stored.

ArcGIS stores vector data in feature classes and collections of topologically related feature classes. The attributes associated with the features are stored in data tables.

ArcGIS uses three different implementations of the vector model to represent feature data: coverages, shapefiles, and geodatabases.

Vector data models are useful for representing and storing discrete features such as buildings, pipes or parcel boundaries.

b) Raster

A raster model (otherwise known as a raster dataset image), in its simplest form is a matrix (grid) of cells (see figure 3.3).

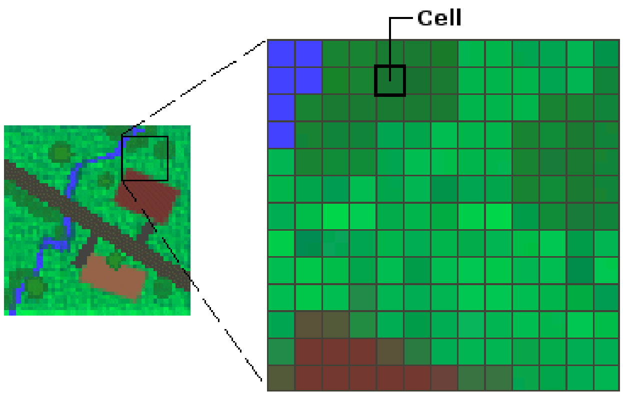


Figure 3.3: Raster data grid (ESRI, 2012)

Each cell has a width and height and is a portion of the entire area represented by the raster. The dimension of the cells can be as large or as small as needed to represent the area and the features within the area, such as a square kilometre, square meter, or even square centimetre. The cell size determines how coarse or fine the patterns or features will appear. The smaller the cell size, the more detail the area will have. However, the greater the number of cells, the longer it will take to process and it will require more storage space. If a cell size is too large, information may be lost or subtle patterns may be obscured.

c) Triangulated Irregular Network (TIN)

In a triangulated irregular network (TIN) model (figure 3.4), the world is represented as a network of linked triangles drawn between irregularly spaced points with x, y, and z values. TINs are an efficient way to store and analyze surfaces.

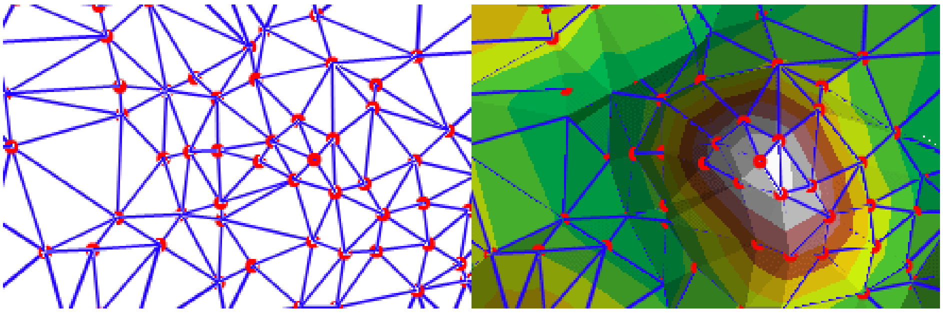


Figure 3.4: Triangulated Irregular Network (TIN) (ESRI, 2012)

Heterogeneous surfaces that vary sharply in some areas and less in others can be modelled more accurately, in a given volume of data, with a triangulated surface than with a raster. That is because many points can be placed where the surface is highly variable, and fewer points can be placed where the surface is less variable. In using only the necessary points, TIN's also provide a more efficient method to store data. ArcGIS stores triangulated surfaces as TIN datasets. As with rasters, TIN datasets can be added to a map in ArcMap and managed with ArcCatalog.

d) Tabular

GIS could be referred as a database that understands geometry. Like other databases, ArcGIS provides the ability of linking tables of data together. Just about any table of data can be joined to an existing feature class or raster dataset if they share an attribute.

Geocoding is another means of getting tabular data on a map. Perhaps the simplest example of geocoding is plotting points based on tables of geographic coordinates.

3.5 Model Validation and Software Troubleshooting

Validation of the software integrity shall be examined via tracing all sources of logical and arithmetic errors. Additionally, a comparison with manual calculations in the illustrated examples would suffice at this stage of software development, scope and time frame availed for this research work.

3.6 Software Dissemination

At the end of the software development phase, the produced DSS shall be availed for public use and evaluation. An online repository would be linked for downloading the software from a hosting website in the internet. Feedback from users of the DSS shall be collected periodically and analysed for future enhancement of the software

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Results

4.1.1 Model Conceptualisation

4.1.1a Abstraction of Design Modules

In this research work, a computer software was designed in such a way that it would facilitate expandability and future enhancement. The concept of *Unit Plug-in (U-Plug)* was therefore introduced. A U-Plug is the calculation engine for a specific type of water/wastewater treatment unit. The software model has to serve as a base platform to gear each engaged U-Plug, i.e. design, development, run, export or import, integration with other U-Plug items ...etc.

Figure (4.1) shows conceptual design structure of the intended Decision Support System (DSS) software platform.

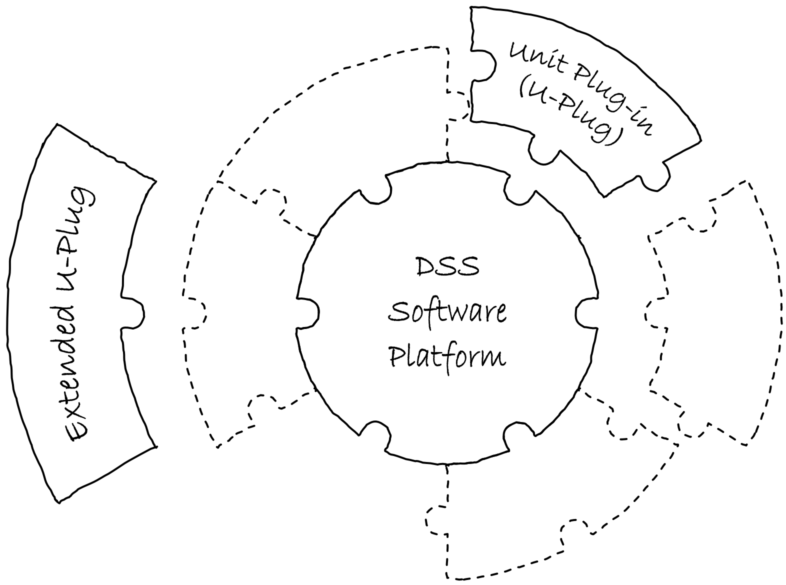


Figure 4.1: Conceptual Design Structure of The DSS Software Platform

4.1.1b DSS Software Layout

An ultimate goal of this research work is to deliver a decision support system (DSS) for designated water/wastewater treatment engineering design. Hence, the DSS's expandability is a genuine part of its core structure. Being driven by such concept, the researcher sought to versatile areas of modelling application by developing competitive software; a software that would serve as an incubation platform for designing various types of water treatment units (WTU) / wastewater treatment units (WWTU).

In order to envisage the aforementioned concept an abstract design framework have been formulated and implemented in the developed DSS platform; named ***Wastewater's Interactive & Simplified Analysis Model (WISAM)***. All WTU/WWTU U-Plug items in WISAM had to be trailed with an abstract level of similarity, i.e. each U-Plug has to have a list of inputs, a group of outputs, and governing mathematical and engineering equations. Hence, a special set of forms / windows would aid in the formulation process of identifying any U-Plug for its associated WTU/WWTU calculations. As illustrated in figure (4.2), the conceptual identification process of a U-Plug engine would consist of the following step levels:

1. **General Information window:** wherein generic information are to be entered, pertaining to the WWTU category, identification title, ...etc.
2. **Definition of inputs list:** where abbreviation, description, initial value and measurement unit are to be stated for each input parameter.
3. **Definition of outputs list:** where abbreviation, description, and measurement unit are to be stated for every output parameter.
4. **Declaration of governing equations:** which could split into two paths:
   a. List definition of simple equations; for structured / sequential step-by-step equations.
   b. List definition of complex equations; for procedural / multifarious equations, along with multiple argument setup (such as conditional cases, iterated loops, ...etc). This path would also cater for identify any links to external design engines (such as stand-alone spreadsheets, dynamic link libraries (DLLs), other executable programmes, .. etc.).
5. **Review window for confirmation of all definitions:** where it would show a confirmation dialogue for the previously gathered information before final creation of the U-Plug definition module.
6. **U-Plug generation and compilation window:** wherein the software would generate a Visual Basic .NET (VB.NET) syntax code and ask the user for a confirmation pertaining to starting code compilation. Such window would also give an advanced user the ability to modify the automatically generated VB code if desired.

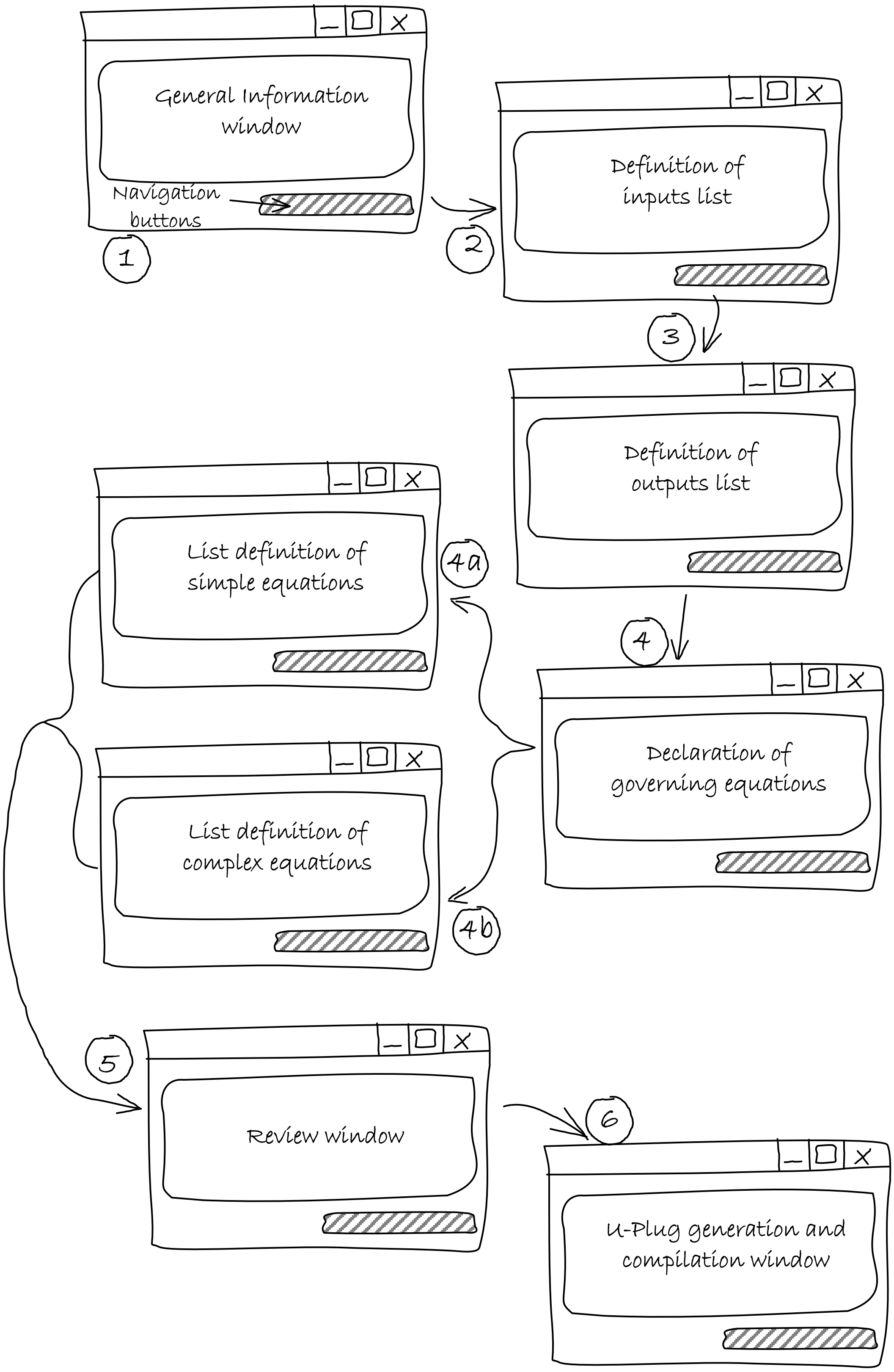


Figure 4.2: Conceptual Identification Process of a U-Plug

The formulated model, WISAM, sought to have a user-friendly as well as dynamically adjustable graphical user interface (GUI). Such criteria have been followed, while keeping in mind the current user's experience with MS Windows™ operating system environment and/or familiarity with known modelling environments in the market.

Figure (4.3) illustrates the conceptual layout design of WISAM's main window. The main window GUI has been divided into several working areas. Each area, presented in figure (4.3), would serve a specific purpose as outlined below:

1. **Title bar:** This bar is the primary one used by any operating system to reflect the name of the running software. In this software, Title bar is intended to show the software's name as well as the title of active modelling session / document.
2. **Menu bar:** The menu bar would contain various application menus. Each menu categorises a specific group of related commands. Most of the software operations could be managed through this bar.
3. **Primary Toolbar:** Typically, a toolbar would contain icons that are shortcuts for selected menu commands. In this research, a primary toolbar is intended to be used for displaying basic tool headers / categories. The U-Plugs Toolbox shall follow such header types and show its subsequent commands.
4. **U-Plugs Toolbox:** The set of commands carried by this area is set to follow selected tool sets from the Primary Toolbar.
5. **Modelling window:** This window would serve as the main document area. It was developed as a drawing canvas in order to ease schematisation of desired treatment units' layout ordering.
6. **Properties window:** This window would show the main input, output, and descriptive properties of selected unit element.
7. **Results window:** This window would present result outputs relevant to chosen operation. It would detail encountered warnings and errors, either from the geometrical / schematic design integrity; or at analysis operations at runtime.
8. **Status bar:** The Status bar would be located at the bottom of the window to report information relative to the selected operation, drawing instructions, analysis stage, completion messages, error indicators ... etc.

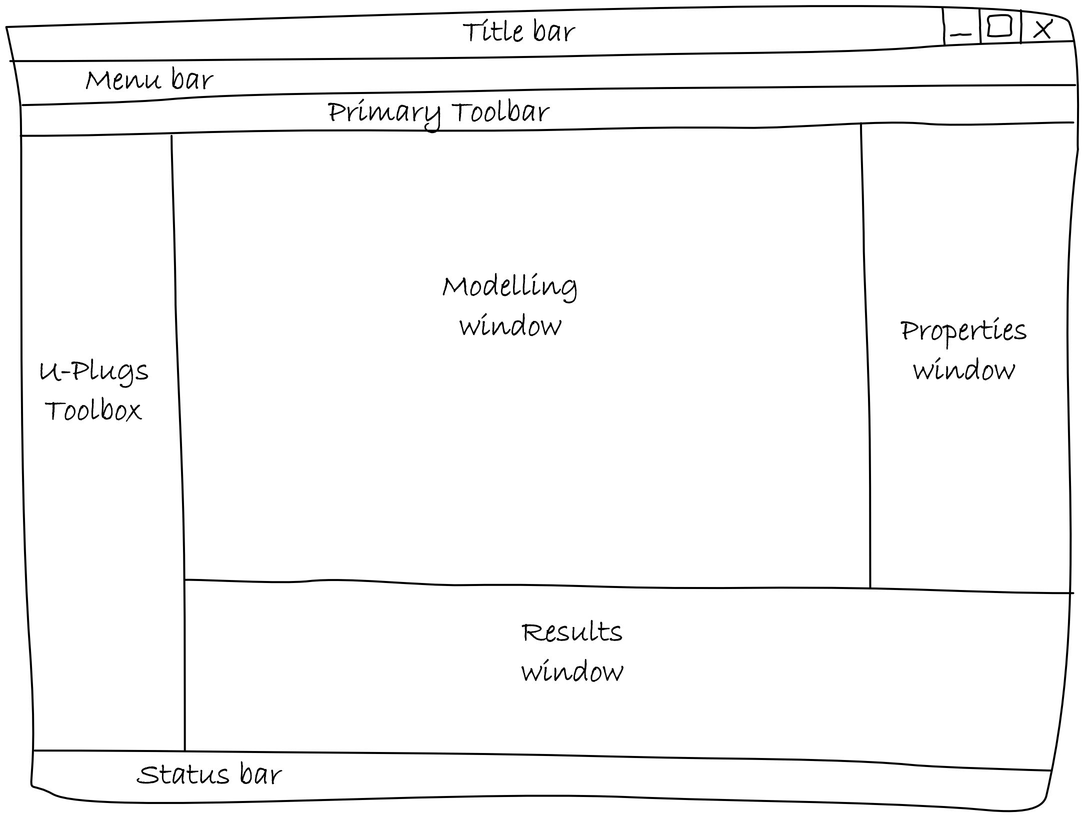


Figure 4.3: Conceptual Layout Design of WISAM's Main Window GUI

Since the *Modelling window* is set to be the main schematisation palette, a graphical illustration would be the best method for easing the user's experience in using WISAM. Therefore, each WTU/WWTU is to be graphically represented by a unique icon. The icon image would be associated with a U-Plug code instance. Furthermore, the graphical line link between different icons would grant an access to associate various output-to-input interchange options; i.e. *variables' remapping*. Figure (4.4) shows a simple U-Plug linkage with configuration accessibility. Furthermore, a U-Plug would have the ability to cater for complex possibilities of user's layout configurations, as shown on Figure (4.5).

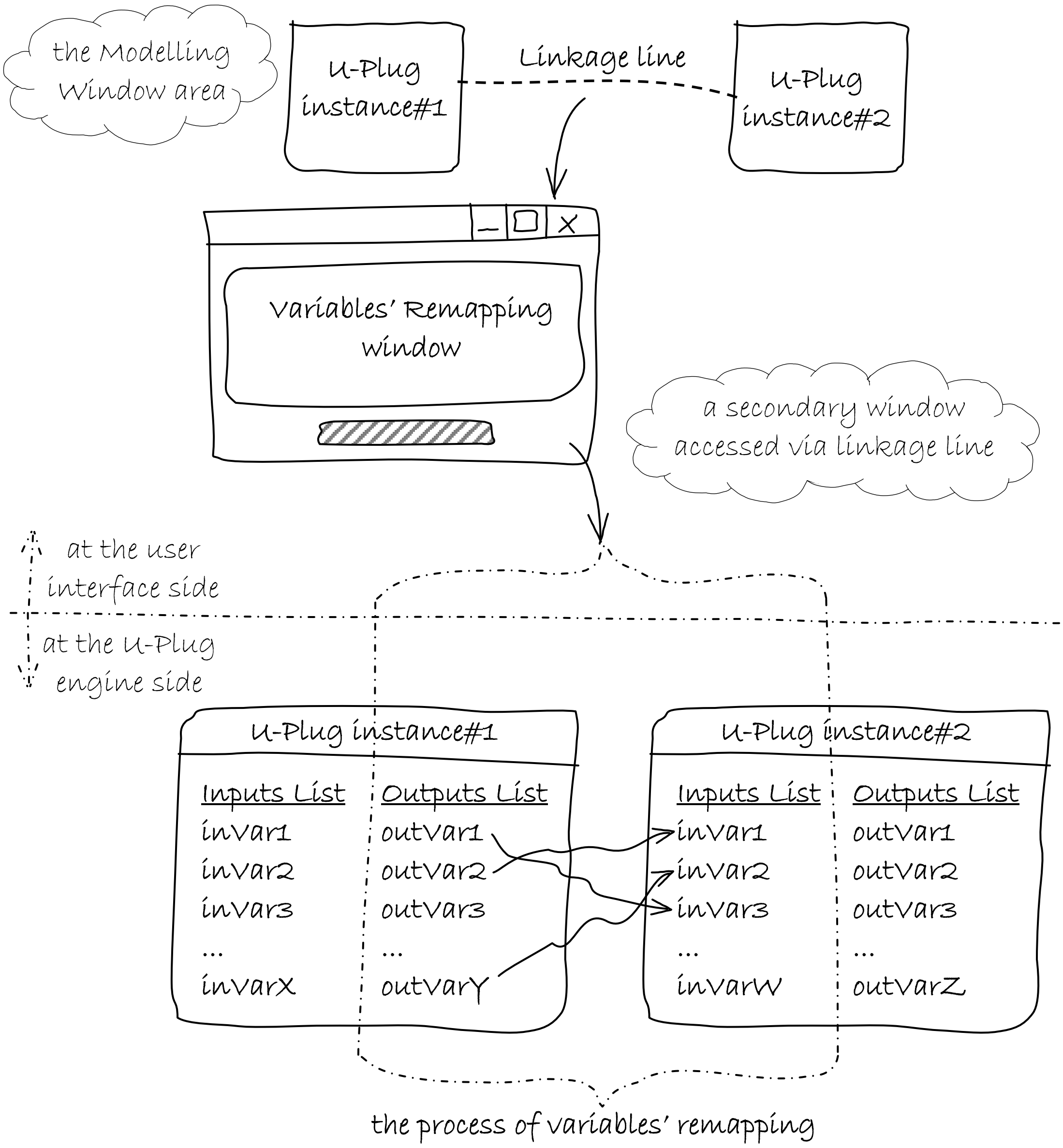


Figure 4.4: Simple U-Plug Linkage With Configuration Accessibility

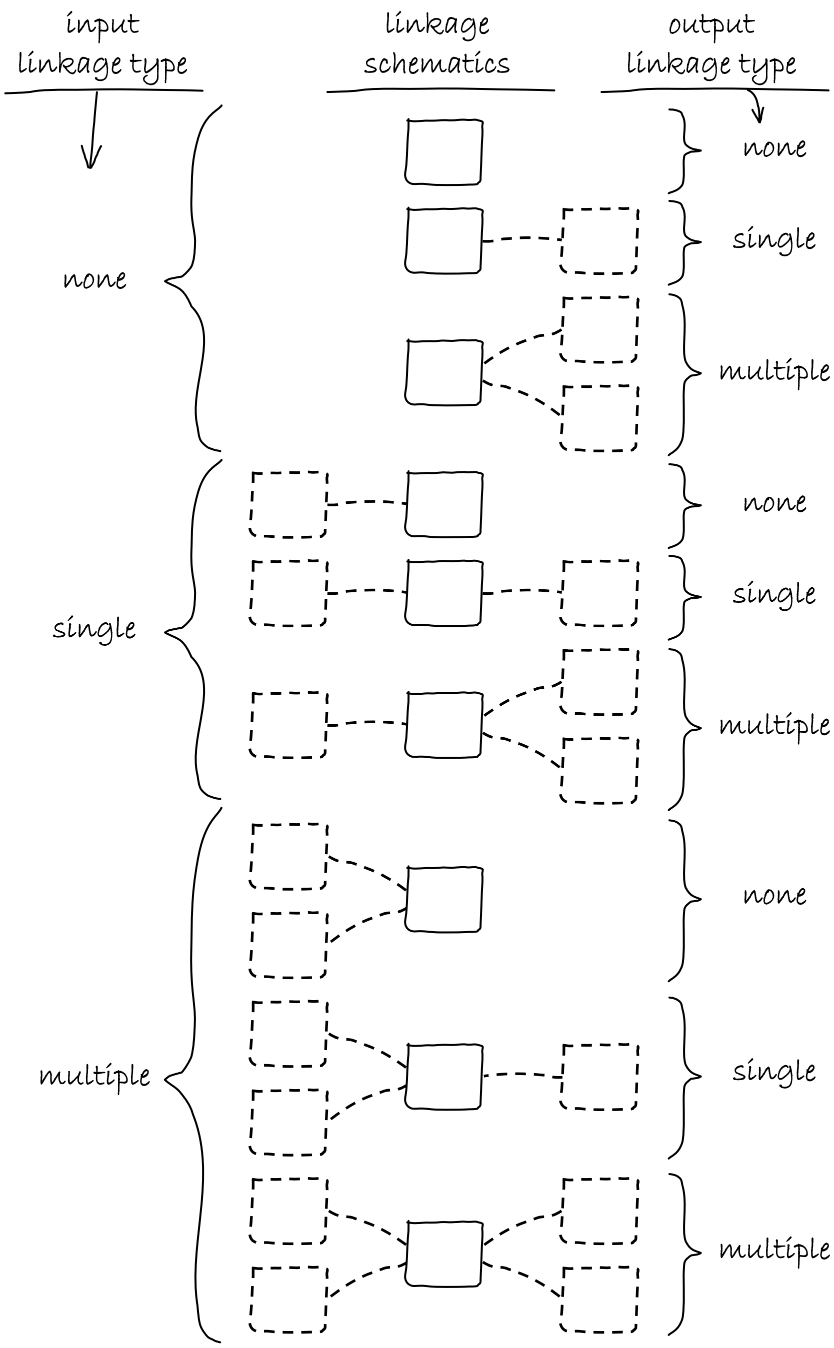


Figure 4.5: Different Layout Configurations of U-Plug Linkages

4.1.1c WISAM's Software Structure

In order to realize the subject software structure, several illustrative flowchart diagrams have been comprehended. Figures 4.6 to 4.12 represent the major flowcharts of the software's core functionality; which shall be explained in the following paragraphs.

Since the whole DSS concept revolves around WTU/WWTU units, it thought to be somehow appropriate that each treatment unit (U-Plug) would belong to a certain defined group of items together with other similar U-Plugs. Example of categorisation might be *{Preliminary, Primary, Secondary, Tertiary, Sludge Disposal, ...etc.}*, or perhaps a more generalized categorisation such as *{Physical, Biological, Chemical}*. Either ways, a U-Plug definition should belong to a certain group of U-Plugs named *Category*. Consequently, each category would have a specific identification code in the software called *CategoryID*.

The Main Software Flowchart shown on figures 4.6a to 4.6c reflects the primary functionality of loading the software at each stage of run. Initially, the list of defined U-Plugs has to be read and displayed according to their order in the stored Category listing. Also, a definition of new U-Plug could also be reached and added to the software repository. Likewise, other processes could similarly be made to an existing U-Plug definition, such as modify, import, or export. Additionally, the chart investigates the possibilities of running a modelling process. During the runtime, the user would set the schematic of desired U-Plugs and its linkages. Each defined instance of a U-Plug would be tagged with a specific identification code called *UnitID*. Accordingly, the modelling process may either be hydraulic, structural, or both for the defined Units in the subject modelling session.

In order to simplify and concise the main workflow of the software, it had to be split into logical sections. Each process section was defined in a separate set of programmed instructions called *subroutines*. Figures 4.7 to 4.12 illustrates various implemented subroutines.

Defining a new U-Plug has to be invoked as illustrated on figures 4.7a & 4.7b. Wherein, the parent Category has to be identified for the desired U-Plug. Subsequent check of existence would be conducted in order to either integrate the new U-Plug definition within an existing Category; or create a new Category from the input. A secondary check would be carried out in order to avoid duplication of a U-Plug definition. A third level of conditional checks would finally be exercised for proper definition of guiding equations (hydraulic and/or structural).

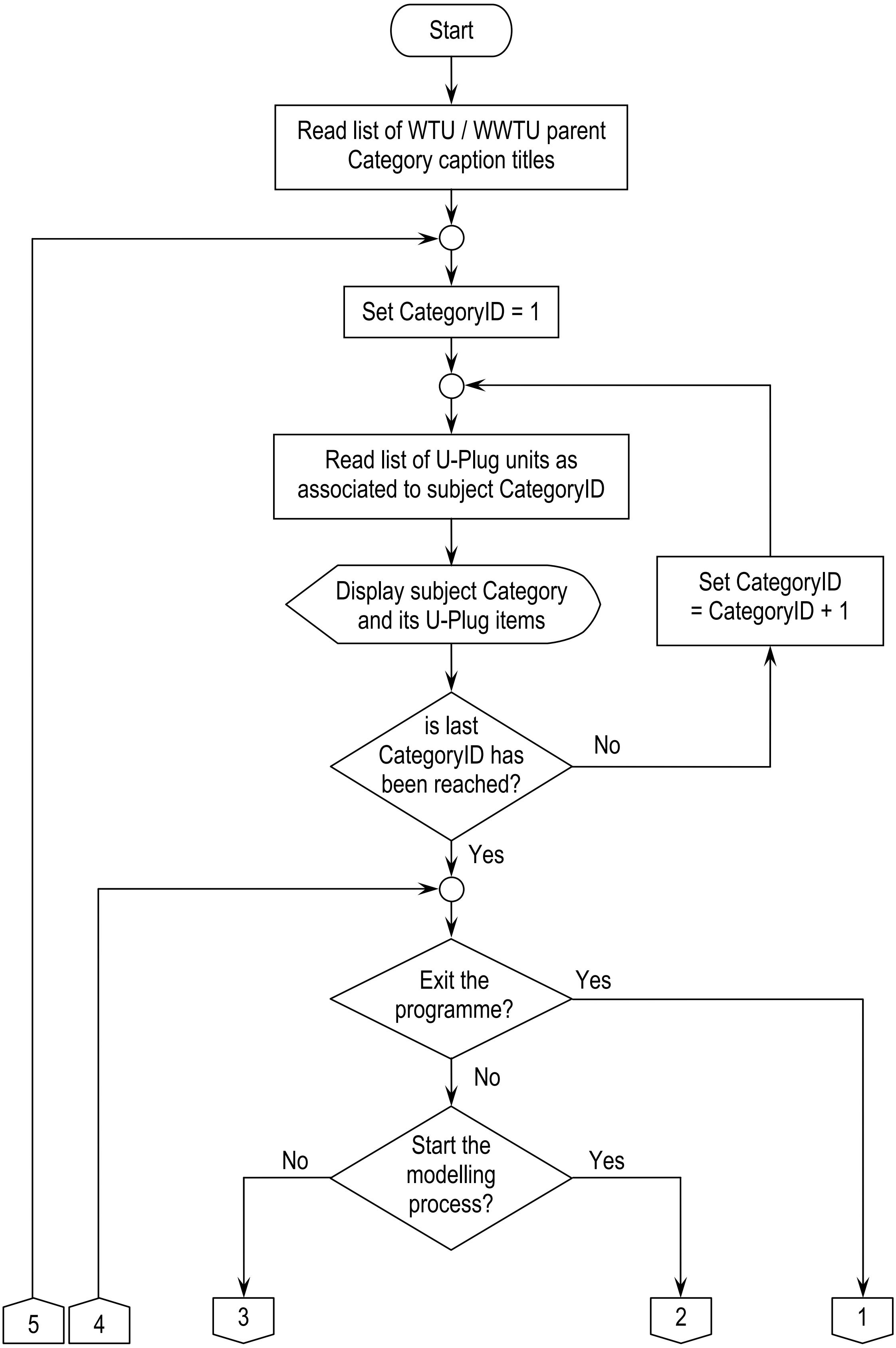


Figure 4.6a: Main Software Flowchart

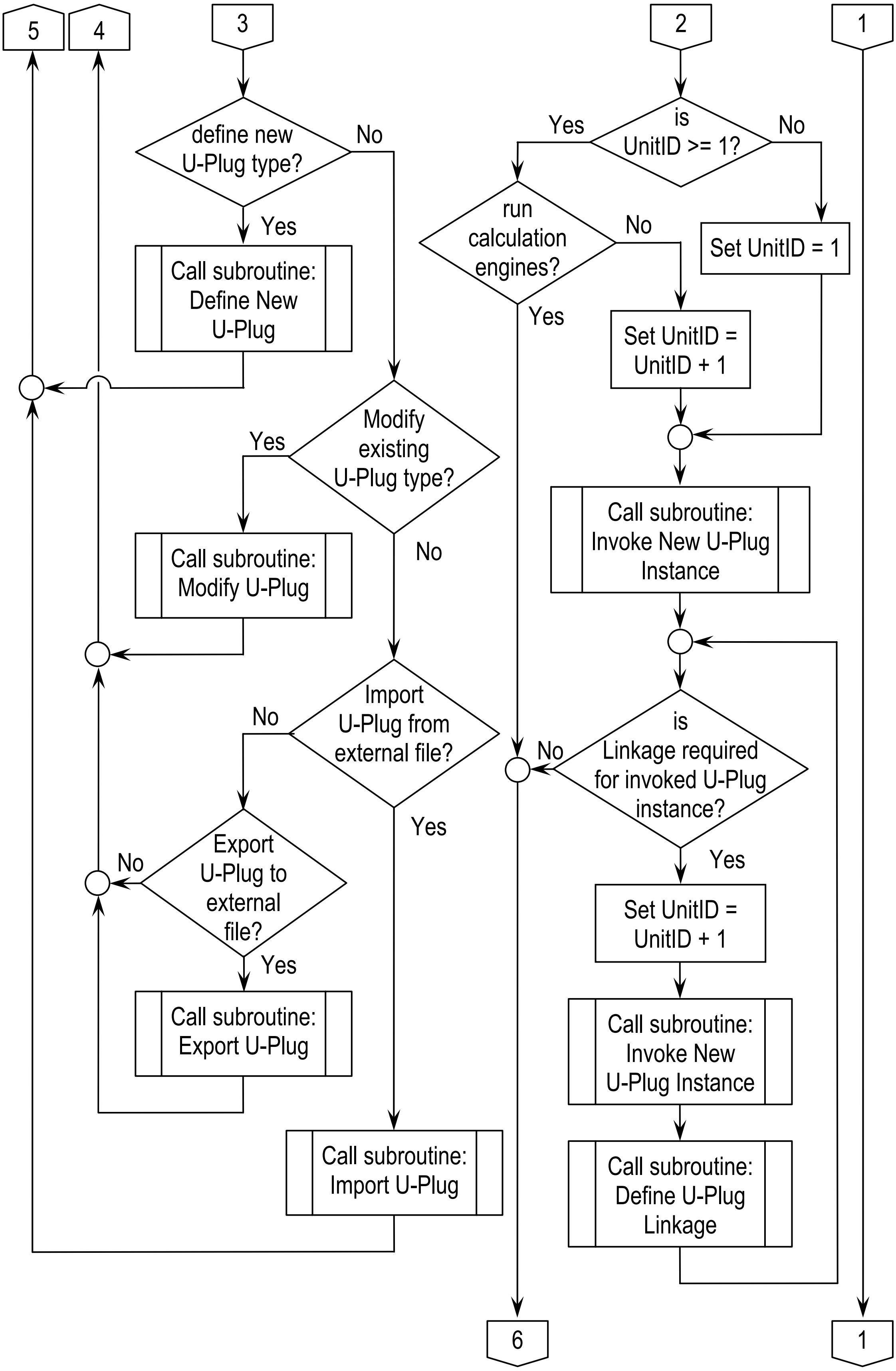


Figure 4.6b: Main Software Flowchart (cont'd)

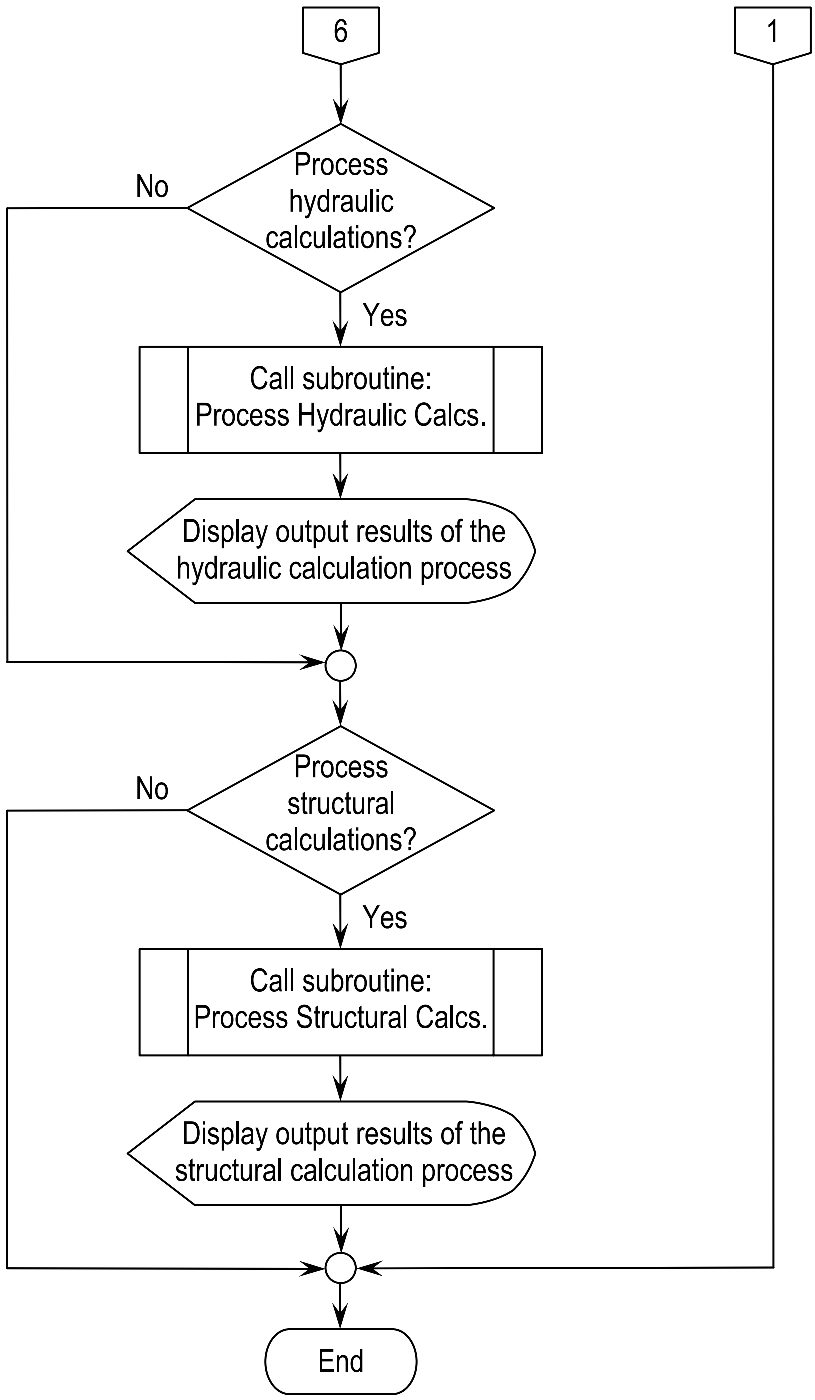


Figure 4.6c: Main Software Flowchart (cont'd)

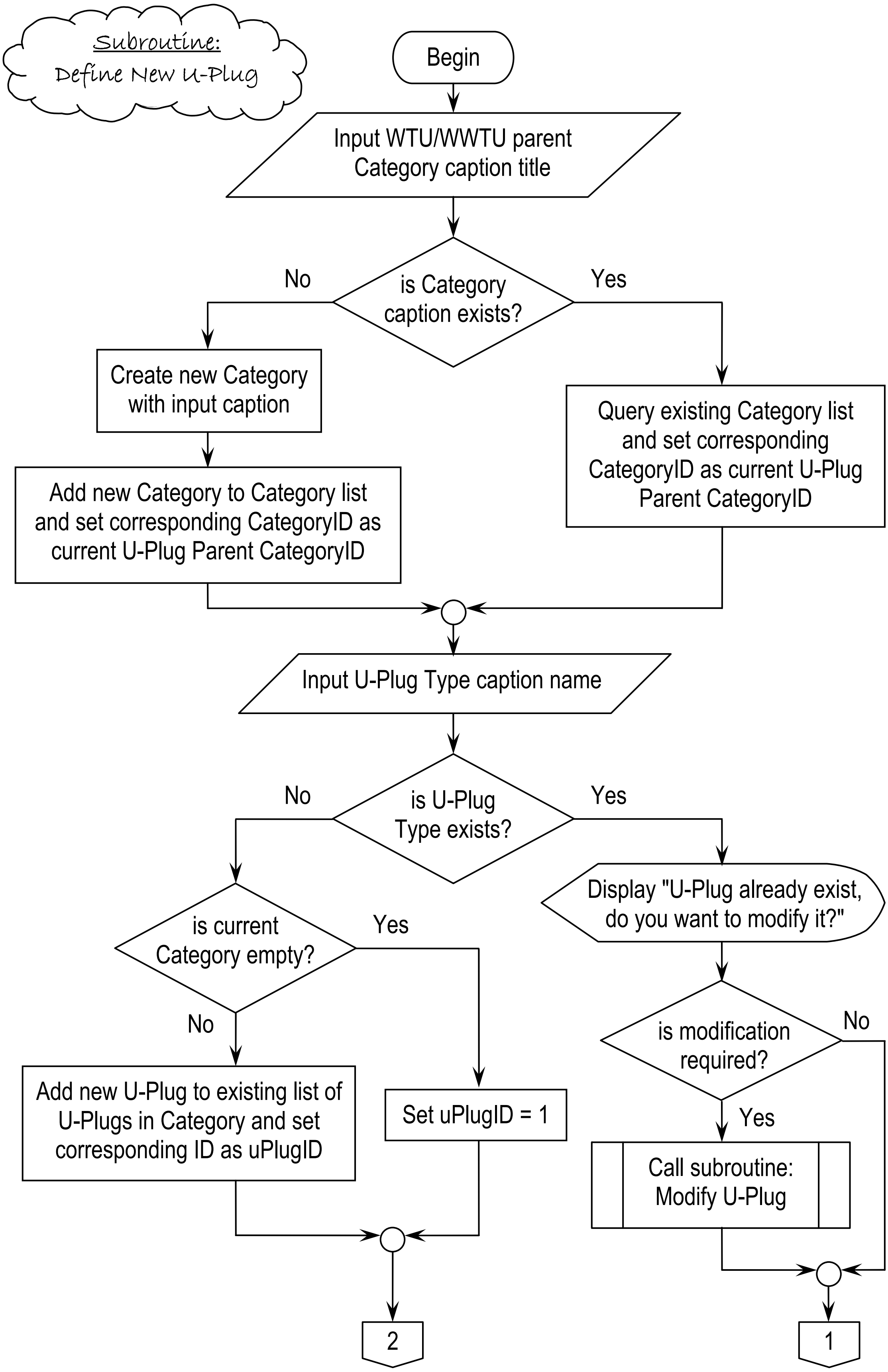


Figure 4.7a: Flowchart of Defining New U-Plug Type Subroutine

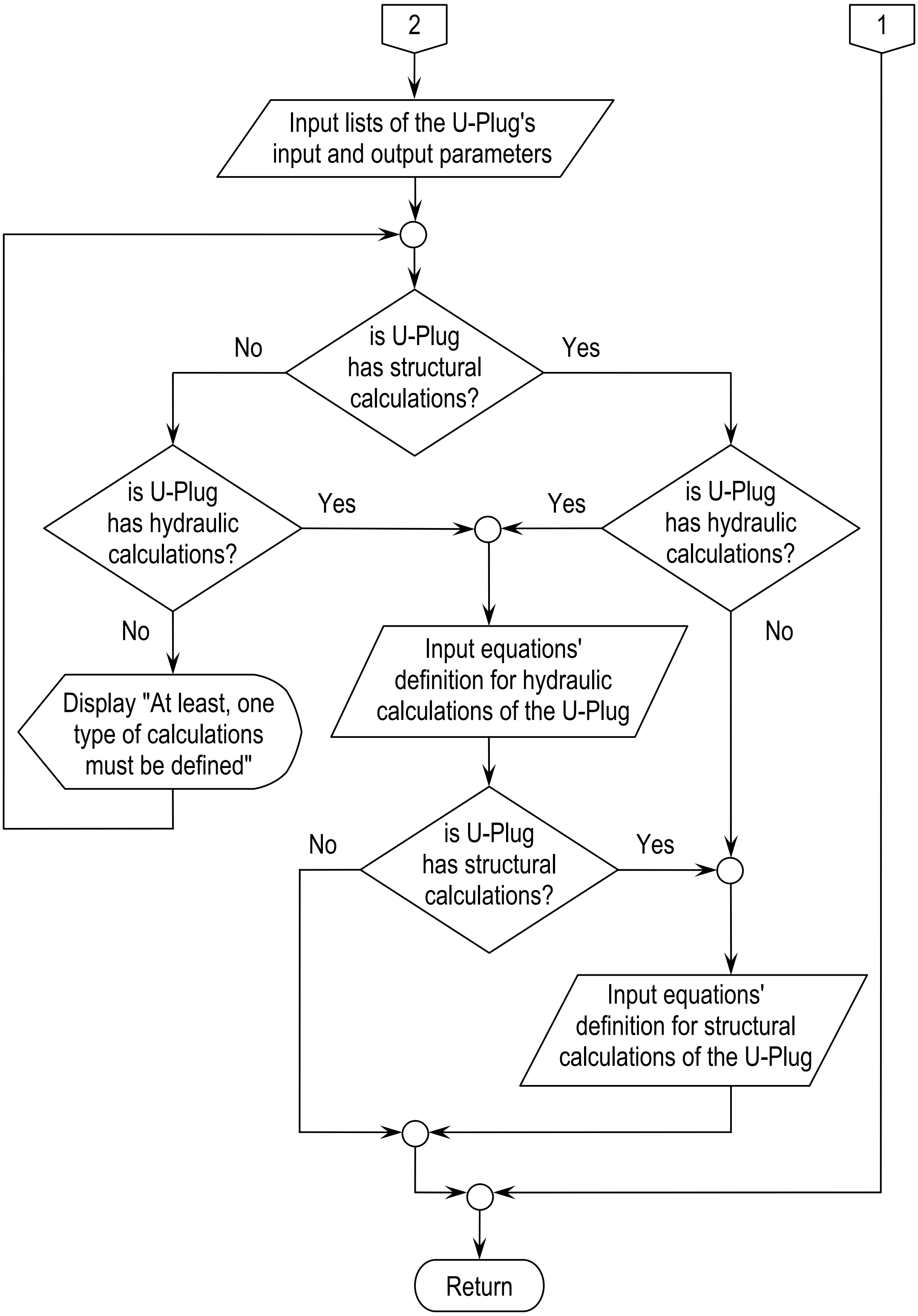


Figure 4.7b: Flowchart of Defining New U-Plug Type Subroutine (cont'd)

Figure (4.8) exemplifies the process of defining U-Plug linkages. Basically, a remapping index list would be created so as to set the connection between an output parameter in an initial U-Plug (*U-PlugIn*) to its corresponding input parameter in the targeted U-Pulg (*U-PlugOut*). If the targeted U-Plug has had an extra set of inputs (a part from the remapped items); an input of missing values would then be required.

The calling subroutines for commencing hydraulic or structural processes are shown on figures 4.9 & 4.10 respectively. Either process would examine the list of U-Plug instances in the running modelling session. Then, by returning to each instance's definition, the appropriate calculation engine would be invoked. It worth noting in this regard that the calculation engines would not necessarily be part of the software itself, on the contrary, a design engine could be an external file or executable script (as set by the initial definition of relevant U-Plug). The aforementioned check-and-invoke procedure would be applied for each and every one of the used U-Plug instances.

The software platform design of WISAM was engineered in such a way that it maintains the flexibility for future expansion of its functionality. Keeping this aspect in mind, figures 4.11a & 4.11b were hence produced to describe the functionality of importing an existing U-Plug definition. As shown on the flowchart, the U-Plug type and Category are to be identified from the external source of import. A reasonable check would be conducted to assure consistency of captioning between the imported *Caption* against the existing ones. Afterwards, possibilities of import would be examined, which include either a replacement of existing U-Plug definition, amendment to the software's list of U-Plugs, or a direct exit from the subroutine without making any modifications.

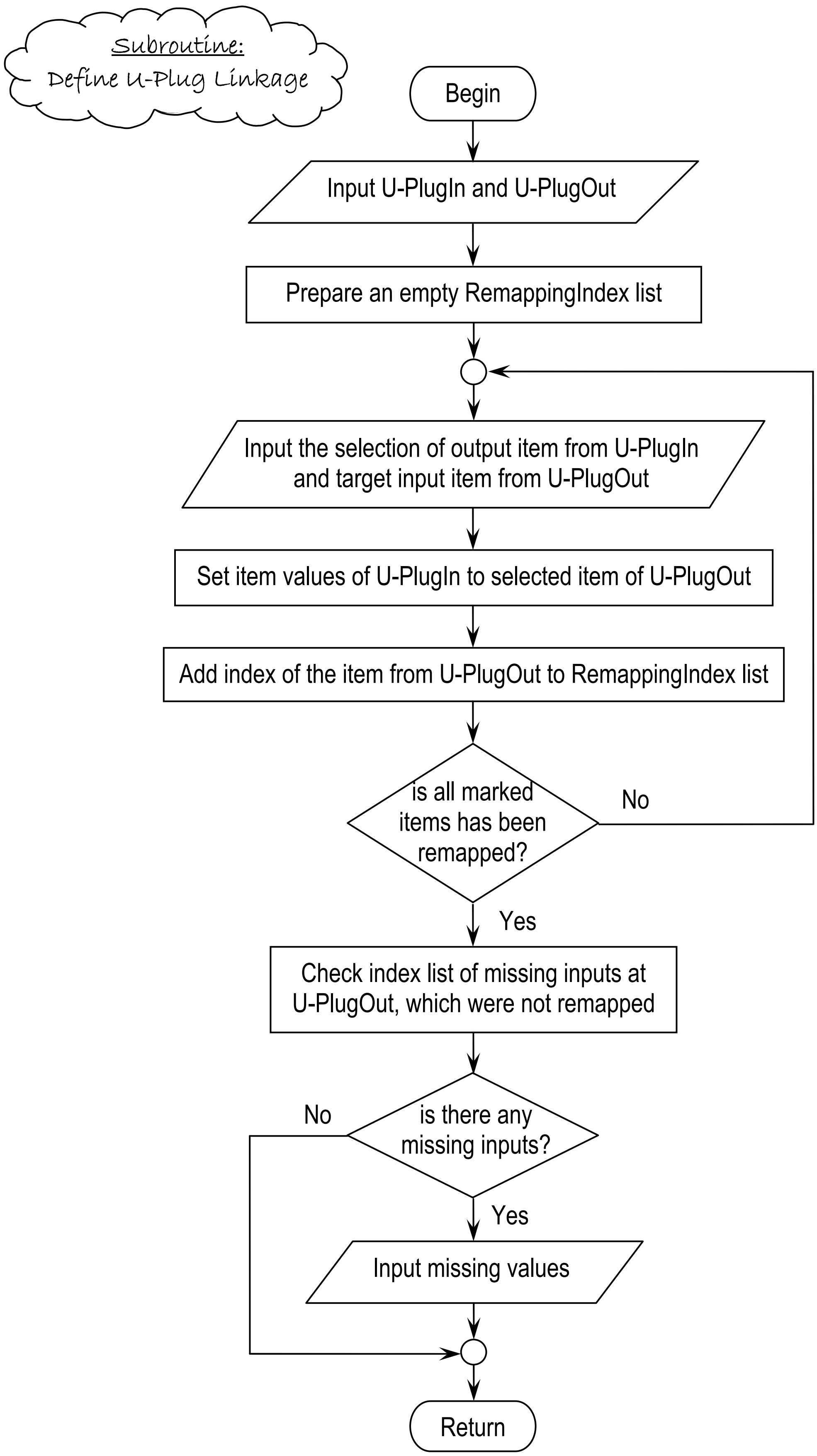


Figure 4.8: Flowchart of Defining U-Plug Linkages

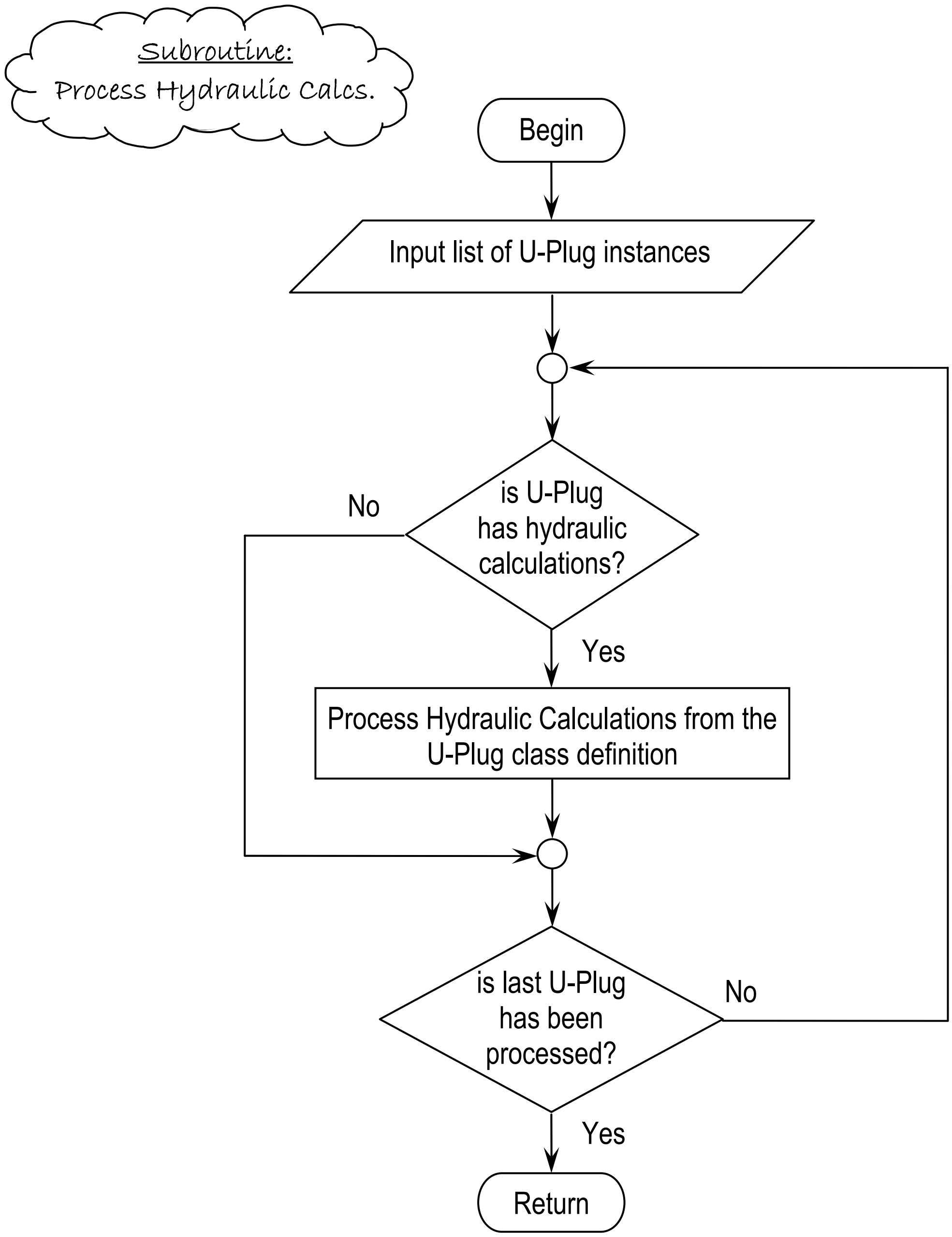


Figure 4.9: Flowchart of Processing Hydraulic Calculations

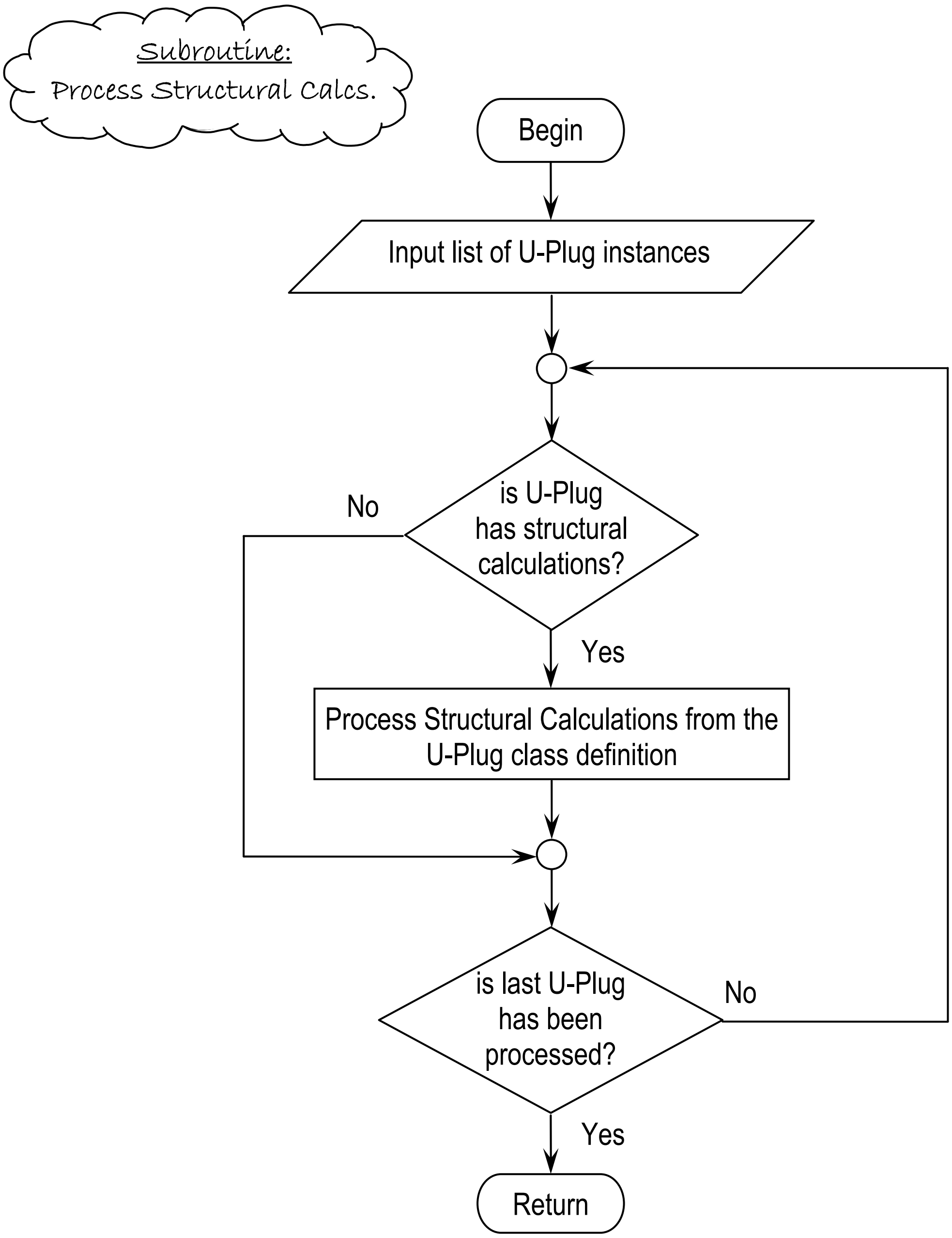


Figure 4.10: Flowchart of Processing Structural Calculations

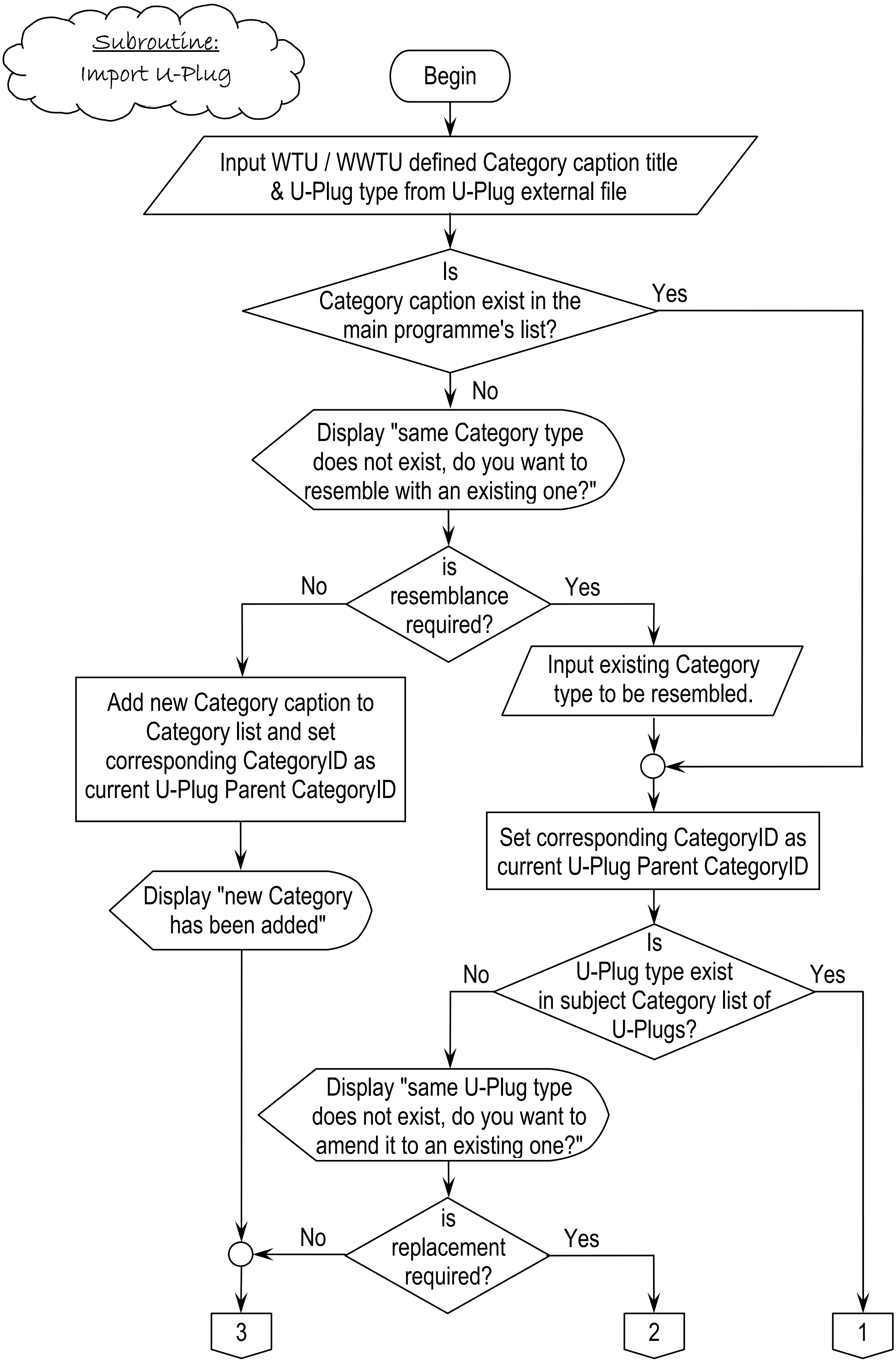


Figure 4.11a: Flowchart of Importing U-Plug Subroutine

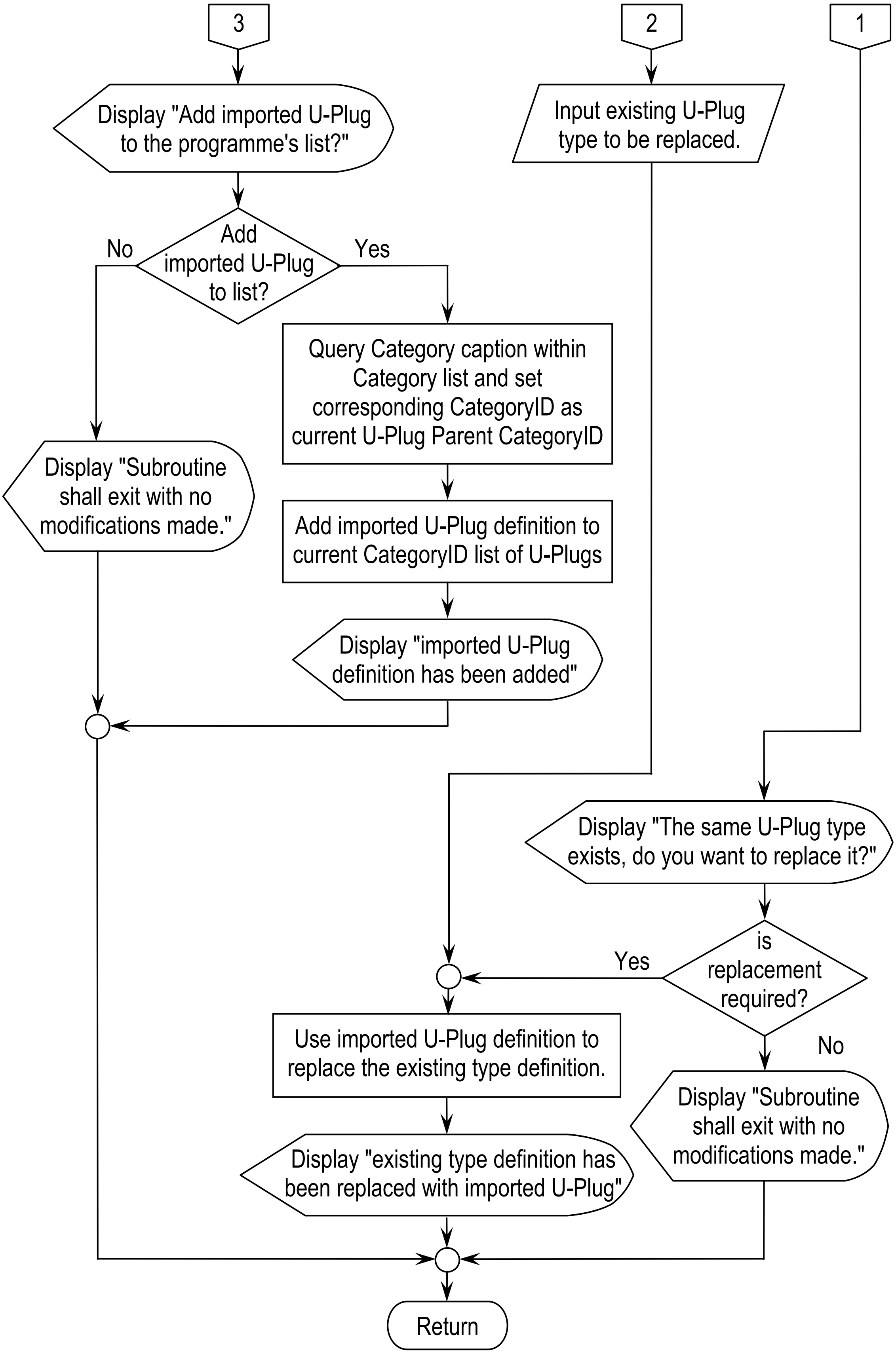


Figure 4.11b: Flowchart of Importing U-Plug Subroutine (cont'd)

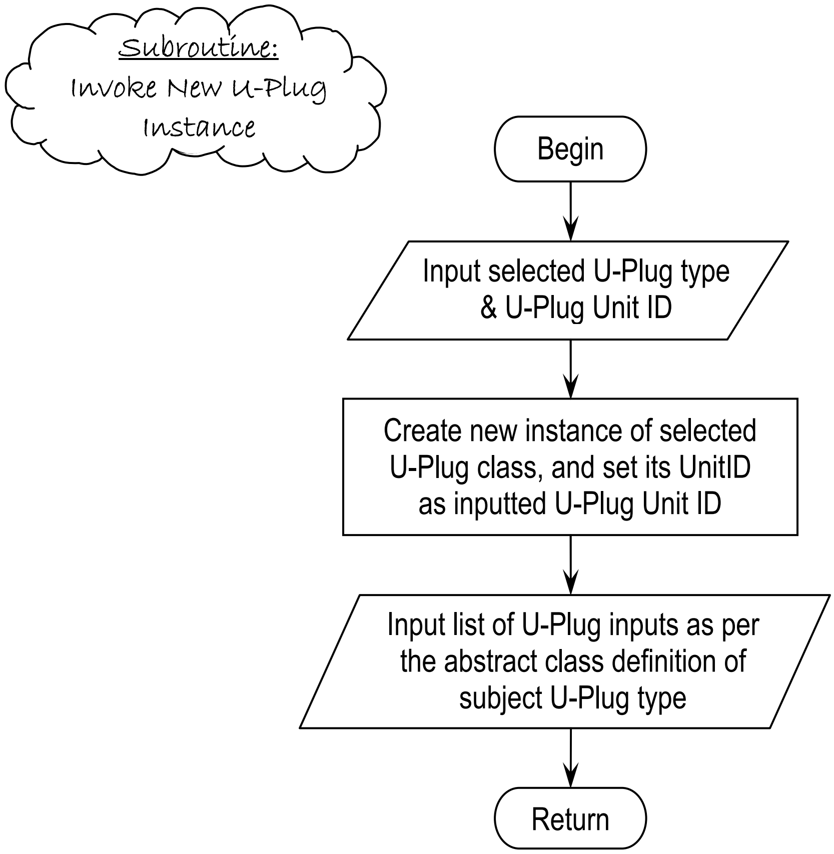


Figure 4.12: Flowchart of Invoking New U-Plug Instance Subroutine

The previous sections have explained the logical sequences of WISAM’s core functionality in the graphical form of Flow Charts. In the following paragraphs, the DSS calculation engine of WISAM will be explained in more detailed manner.

WISAM's DSS engine has been developed following the Unified Modelling Language (UML) paradigm. Figure 4.13 represents the class diagram for WISAM's layout structure. The major components of the presented class diagram falls in one of four categories: *Structure*, *Interface*, *Class*, and *Form*.

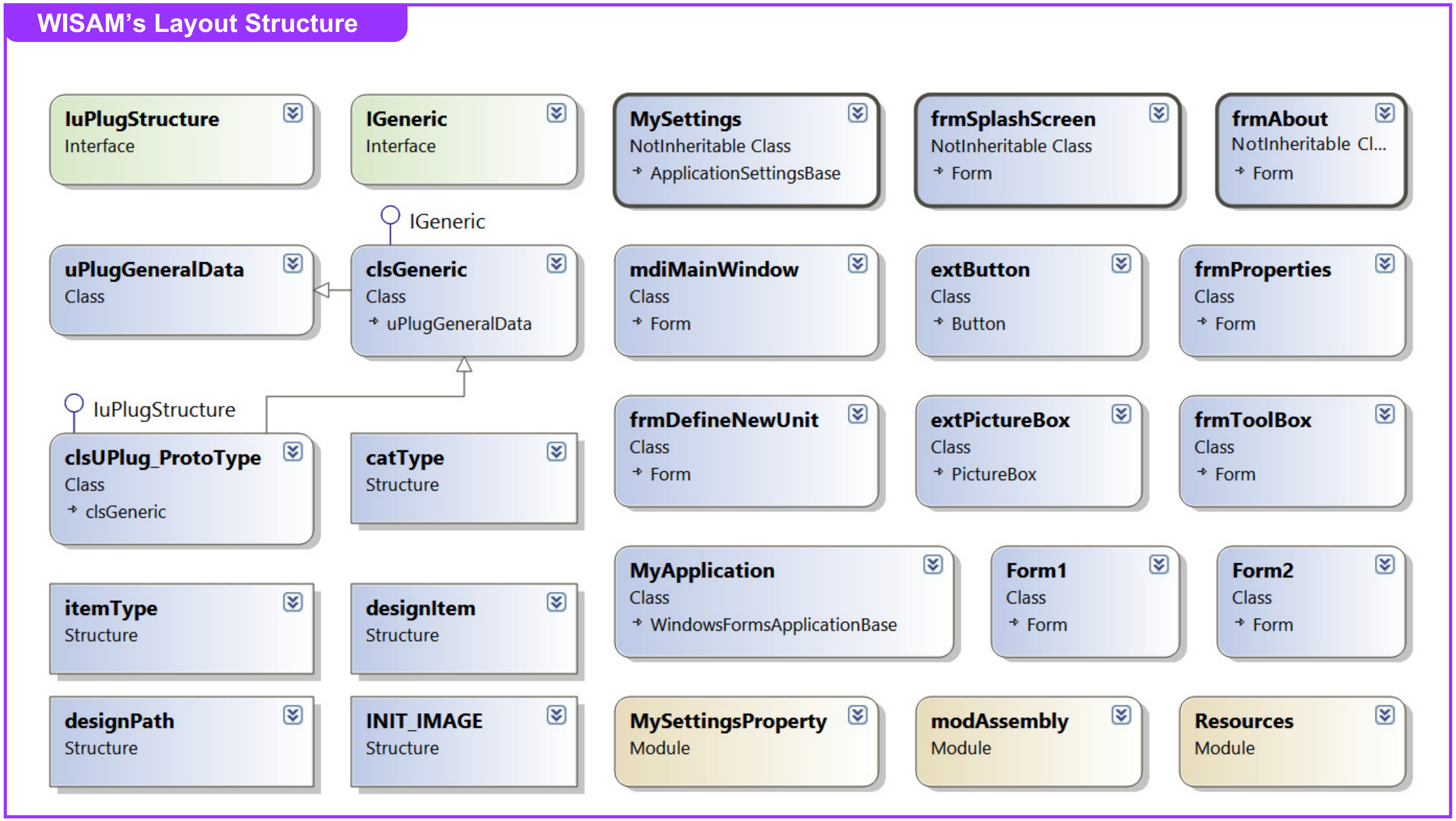


Figure 4.13: UML Class Diagram for WISAM's Layout Structure

Each *Structure* component is intended to define a new data type apart from the pre-defined types in VB.Net. Following the philosophy of this research work, two major types were defined as follows:

1. **catType** structure is to be used for representing each CategoryID, and it contains the following data fields:
    a. *tuParentCatCaption* stores the text value of a U-Plug's parent category name. This field have the VB.NET text type of *String*.
    b. *tuParentCatID* stores the un-signed numeric value of a U-Plug's parent category ID number from the loaded list of U-Plug categories. This field have the VB.NET numeric type of *UInteger*.
2. **itemType** structure is the primary type that will be used for defining any variable in the calculation set of a U-Plug, be it an input or an output variable. This structure contains the following data fields:
    a. *abbreviation* stores the text value of the defined variable's abbreviation, which will be used in displaying results. This field have the VB.NET text type of *String*.
    b. *description* stores the text value of an elaborated description of what the defined variable is. This field have the VB.NET text type of *String*.
    c. *unit* stores the text value of the measurement unit for the variable. This field have the VB.NET text type of *String*.
    d. *value* stores the numeric value of the entered or calculated variable (depends on the variable case of being an input or an output to the U-Plug). This field have the VB.NET numeric type of *Double*.

As explained by the Object Oriented Programming (OOP) approach, the *Interface* component is used to identify the programming definition that could be implemented by a Class component. In this regard, the following two Interfaces were defined:

1. **IGeneric** interface is to be used in the abstract level of a WISAM's class definition, and it have the following functions (*Methods*):
    a. *SetItemType* defines the initiation of a variable item via its prescribed properties. The result from this method is returned in the form of *itemType* data type. The local parameters for this method follow the same components of an itemType. These parameters are:
        - *a*, which passes the *abbreviation* value of an item.

- *d*, which passes the textual *description* of an item.
- *v*, which passes the numeric *value* of an item.
- *u*, which passes the measurement *unit* of an item.

b. *SetParentCatType* defines the initiation of a U-Plug's parent category. The result from this method is returned in the form of *catType* data type. The local parameters for this method follow the same components of a catType. These parameters are:
   - *i*, which passes the CategoryID of a U-Plug's category.
   - *c*, which passes the caption text of the U-Plug's category.

2. **IuPlugStructure** interface is to be used in the abstract level of a WISAM's U-Plug definition, and it have the following Methods:
   a. *GetInputs* defines the call for retrieving the data inputs from a U-Plug instance. The result from this method is returned in the form of a VB.NET text type of *String*. This method have a single local parameter named *type_of_WWTUnit* of a text *String* data type.
   b. *LoadDefaults* defines the call for retrieving the default data inputs as stored in a U-Plug type definition. The result from this method is returned in the form of a VB.NET content type of *Object*. This method have a single local parameter named *i* of a numeric *Integer* data type.
   c. *LoadFromFile* defines the call for retrieving the data inputs from an external file. The result from this method is returned in the form of a VB.NET text type of *String*. This method have a single local parameter named *name_of_file* of a text *String* data type.
   d. *SaveToFile* defines the call for exporting the data variables of a U-Plug to an external file. The result from this method is returned in the form of a VB.NET text type of *String*. This method have a single local parameter named *name_of_file* of a text *String* data type.
   e. *ProcessHydroCalc* defines the call for invoking a U-Plug's calculation engine of Hydraulic analysis/design based on the list of inputs. The result from this method is the list of calculated items; which are returned in the form of a VB.NET series type of *ArrayList*. This method have a single local parameter named *inputsList* of an *ArrayList* data type.

f. *ProcessStrucCalc* defines the call for invoking a U-Plug's calculation engine of Structural analysis/design based on the list of inputs. The result from this method is the list of calculated items; which are returned in the form of a VB.NET series type of *ArrayList*. This method have a single local parameter named *inputsList* of an *ArrayList* data type.

Each *class* in the layout structure encompasses a specified set of programmed instructions. In this research, the major defined classes for WISAM's core structure may be listed as follows:

1. **uPlugGeneralData** class defines the primary parameters which has to be availed in every U-Plug definition. This class have eighteen fields, described as shown in Table 4.1.
2. **clsGeneric** class defines the primary parameters (fields and methods) which has to be availed in every U-Plug definition. The fields of this class are completely inherited from *uPlugGeneralData*, whilst its methods are imported from the implementation of the *IGeneric* interface.
3. **clsUPlug_ProtoType** class is a dummy representation of any U-Plug definition, i.e. every U-Plug definition will have the same structure of this prototype concept. Basically, this class inherits all the properties of *clsGeneric* class along with implementing the methods of *IuPlugStructure* interface. Subsequently, WISAM is capable of incorporating any number of U-Plug definitions as far as they follow similar definition to *clsUPlug_ProtoType*.

Table 4.1 reflects the summary description of WISAM's core parameters, whereas figure 4.14 illustrates the UML Class Diagram for WISAM's core structure.

Table 4.1: Summary Description of WISAM's Core Parameters

Item Name	Item Type	Parameters	
		Name	Data Type
catType	Structure	*Fields:*	
		tuParentCatCaption	String
		tuParentCatID	UInteger
		Methods:	
		-	-
itemType	Structure	*Fields:*	
		abbreviation	String
		Description	String
		Unit	String
		Value	Double
		Methods:	
		-	-
IGeneric	Interface	*Fields:*	
		-	-
		Methods:	
		SetItemType( )	itemType
		SetParentCatType( )	catType
IuPlugStructure	Interface	*Fields:*	
		-	-
		Methods:	
		GetInputs( )	String
		LoadDefaults( )	Object
		LoadFromFile( )	String
		SaveToFile( )	String
		ProcessHydroCalc( )	ArrayList
		ProcessStrucCalc( )	ArrayList
uPlugGeneralData	Class	*Fields:*	
		uPlugParentCat	catType
		uPlugType	String
		uPlugTypeCaption	String
		uPlugTypeImg	Image
		uPlugUnitCaption	String
		uPlugUnitID	UInteger
		hasHydroCalcs	Boolean
		uPlugHydroDispImg	Image
		uPlugHydroSetOfImg	String
		hasStrucCalcs	Boolean
		uPlugStrucDispImg	Image
		uPlugStrucSetOfImg	String
		inputList	ArrayList
		outputList	ArrayList
		isLinkedIn	Boolean
		lstLinkedIn	List(Of UInteger)
		isLinkedOut	Boolean
		lstLinkedOut	List(Of UInteger)
		Methods:	
		-	-
clsGeneric	Class - Inherits: uPlugGeneralData - Implements: IGeneric	*Fields:*	
		(only as inherited/implemented)	
		Methods:	
		(only as inherited/implemented)	
clsUPlug_ProtoType *(or any defined name for a specific U-Plug)*	Class - Inherits: clsGeneric - Implements: IuPlugStructure	*Fields:*	
		(only as inherited/implemented)	
		Methods:	
		(only as inherited/implemented)	

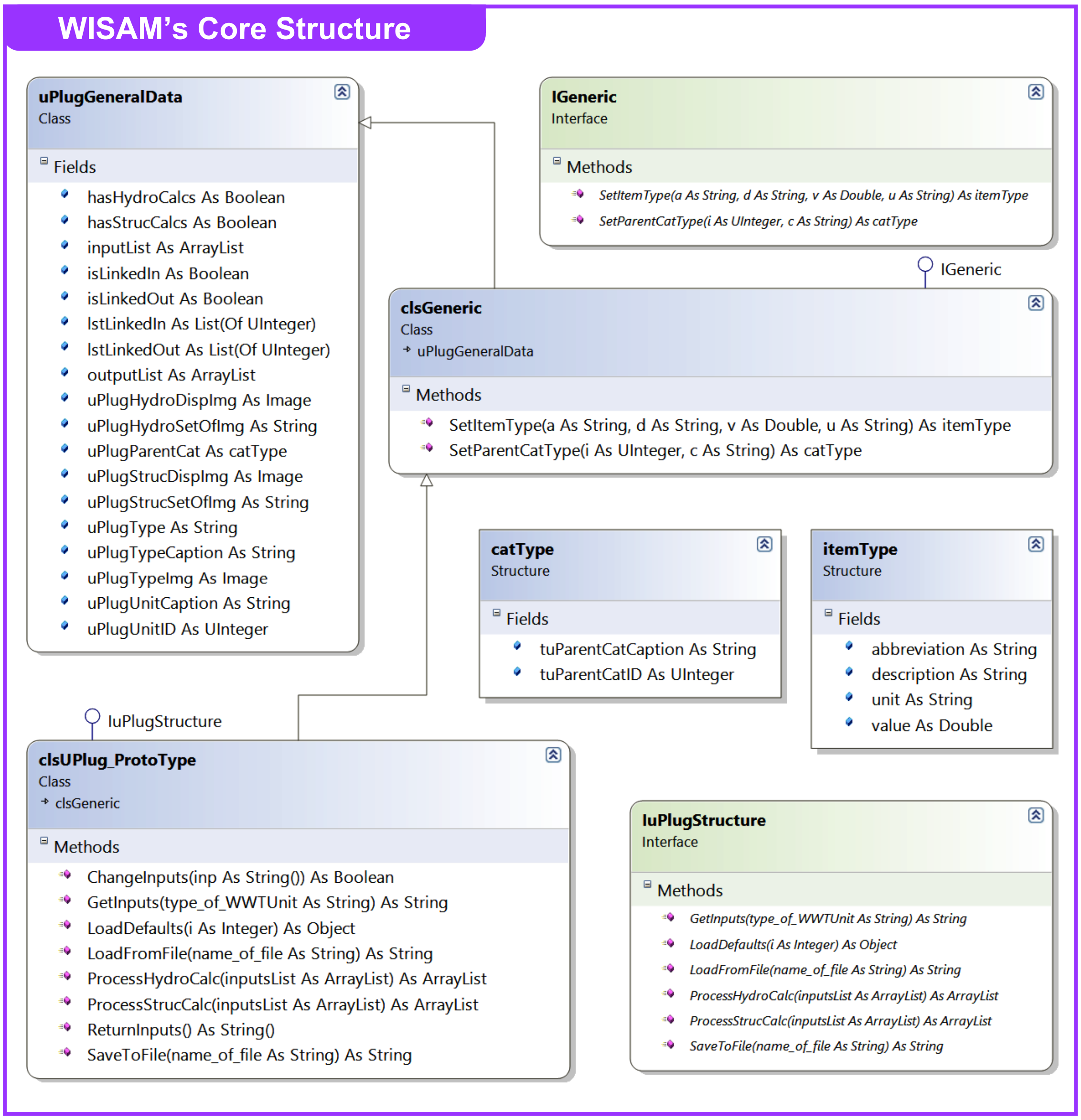


Figure 4.14: UML Class Diagram for WISAM's Core Structure

4.1.2 DSS Software Formulation, Validation and Verification

4.1.2a Software Development Under VB.Net

The Integrated Design Environment (IDE) of Visual Basic .NET has been utilised to build WISAM's graphical user interfaces (GUIs) and implementation code.

The main screen window for creating a new VB.NET Solution is the starting point for developing a Windows Application programme (see figure 4.15). Selecting to create a *Windows Forms Application* leads to creating a default GUI form as shown on figure 4.16.

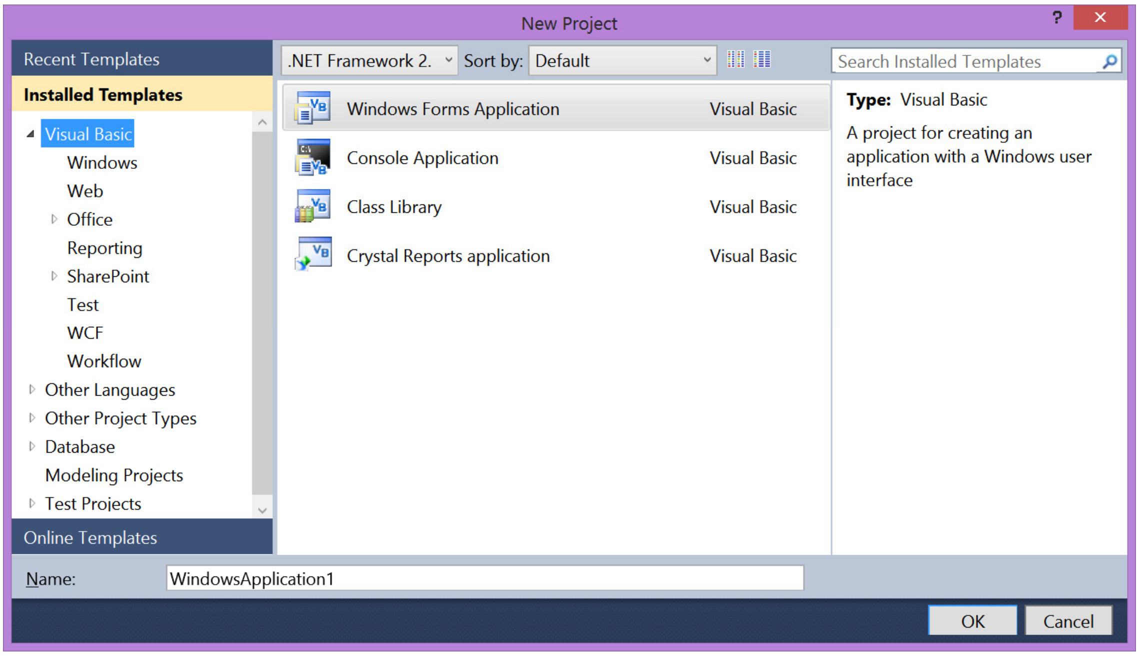


Figure 4.15: Main Screen for Creating a New VB Project

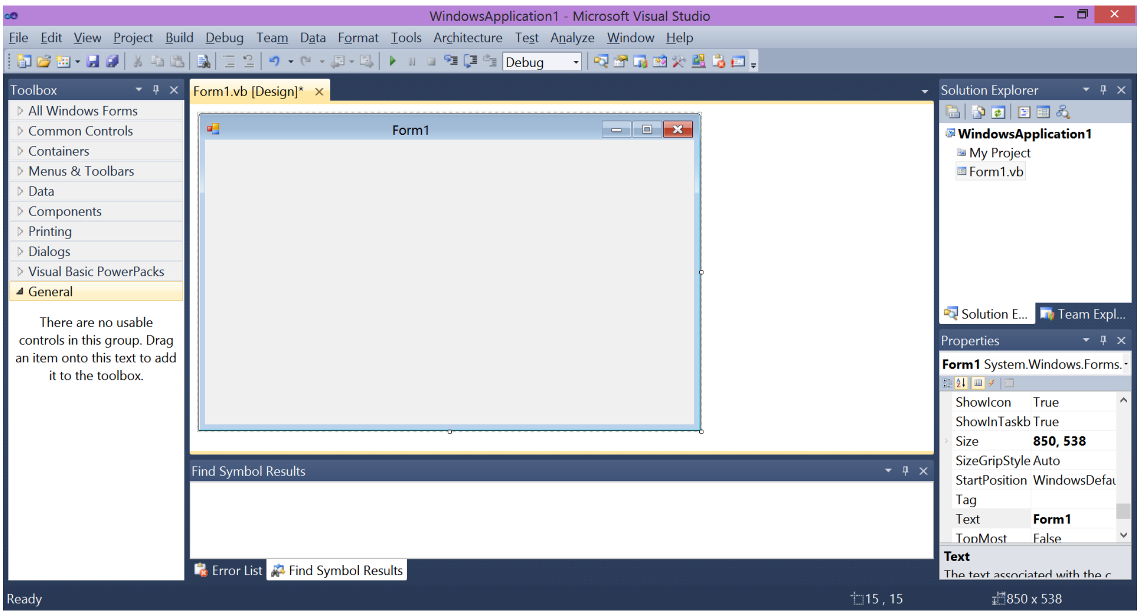


Figure 4.16: VB.NET Main IDE

The toolbox subset at the left side of VB.NET IDE contains almost every component that is required to develop a GUI under MS-Windows operating system. As it could be seen on figure 4.17, the Toolbox items are grouped in several categories; which include: Common Controls, Containers, Menus and Toolbars, Data, Components, Printing, Dialogs, and many other categories that could be loaded whenever desired.

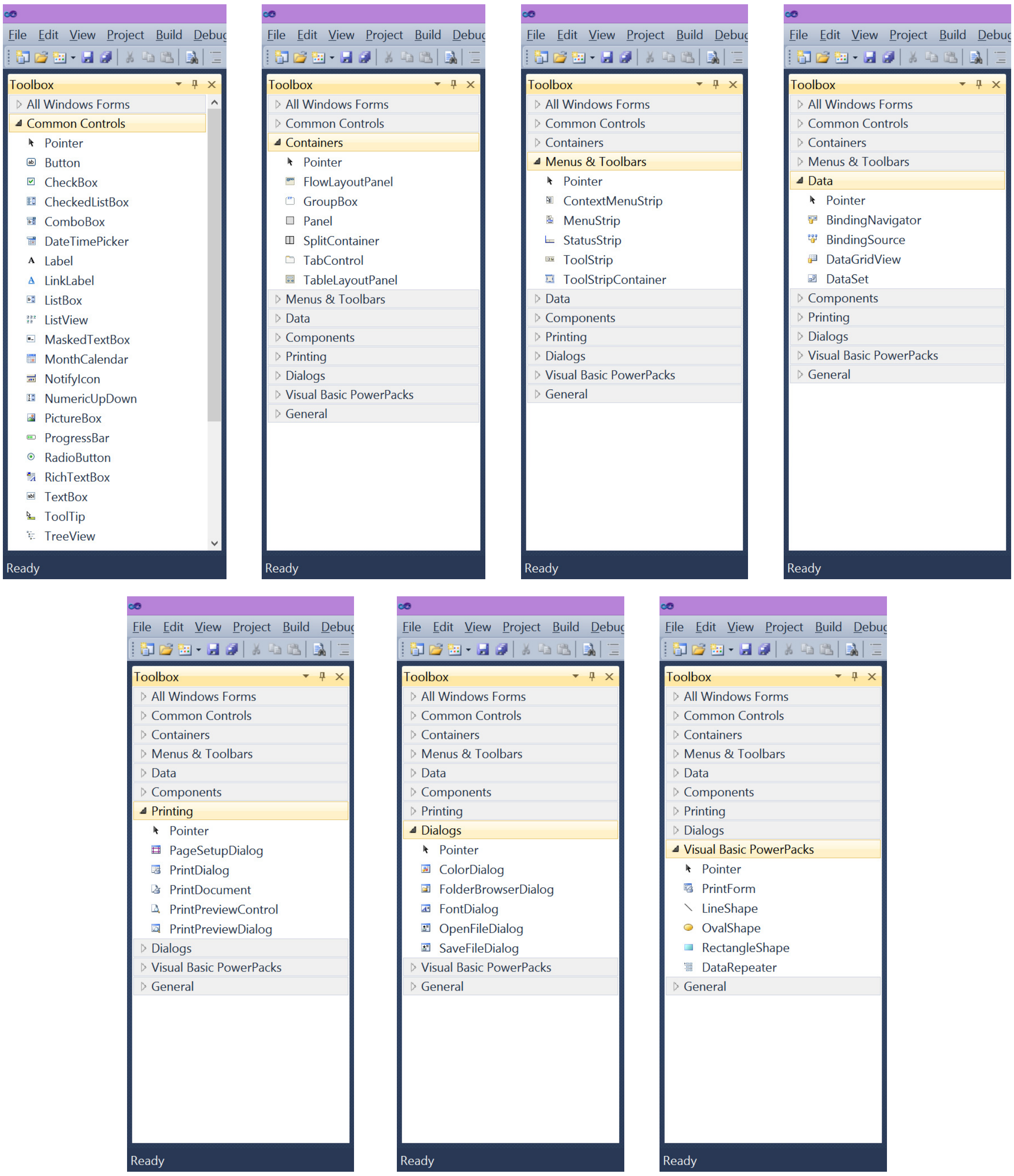


Figure 4.17: VB.NET Toolbox Subsets

In addition to the standard Windows Forms (i.e. graphical user interfaces), various programming components could be added to the Solution project at any stage of software development. Such components may include but not limited to: Code files and modules, Database connectors, General diagrams and external file resources, Windows Forms, and Reporting agents (refer to figures 4.18 – 4.22).

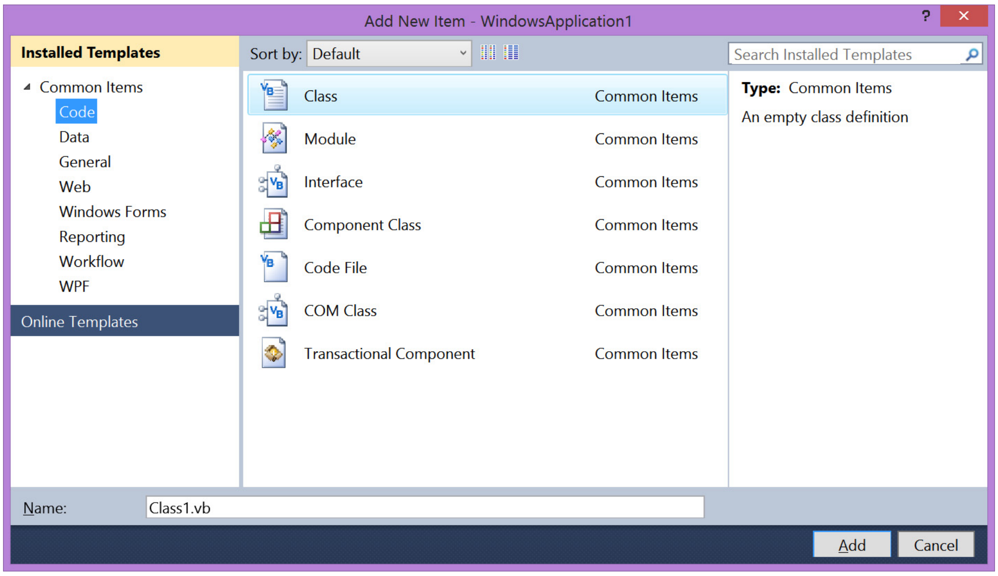


Figure 4.18: Addition of Programming Code Modules

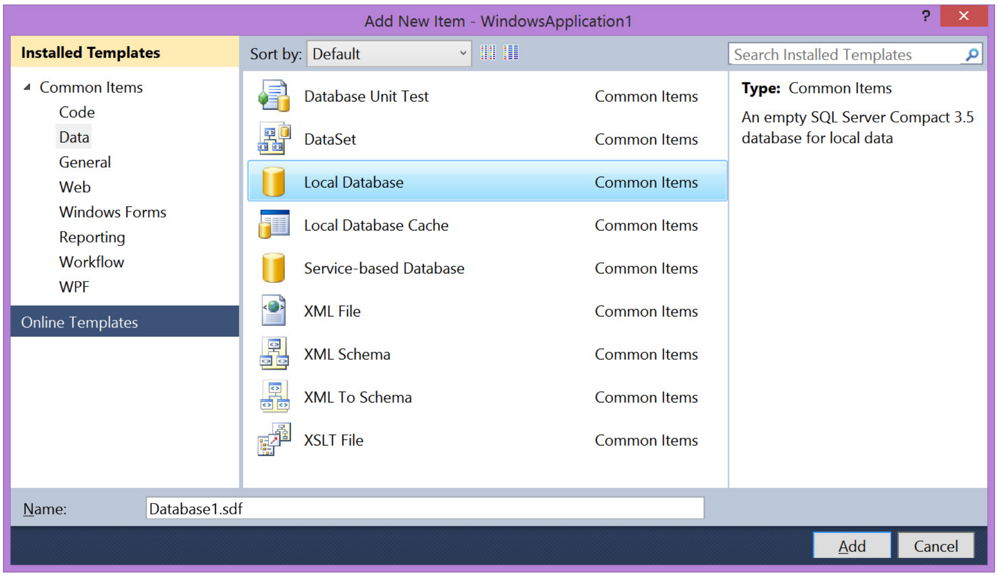


Figure 4.19: Addition of Database Components

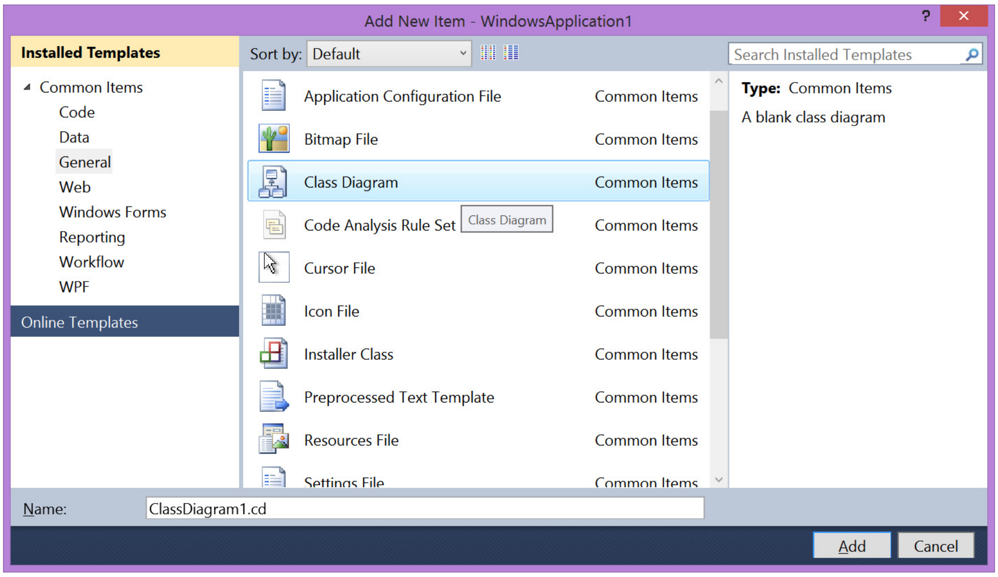


Figure 4.20: Addition of General File Modules

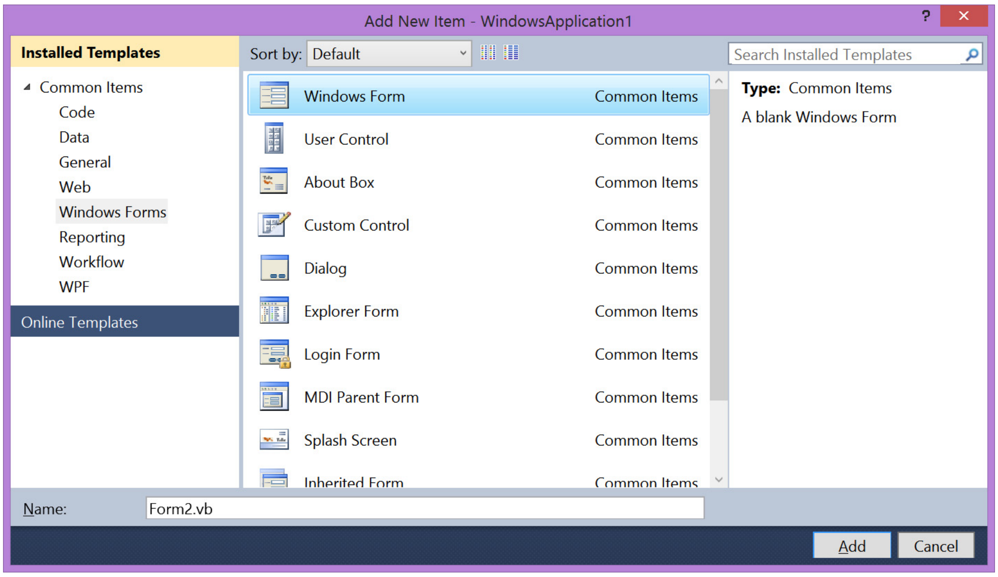


Figure 4.21: Addition of Various Windows Forms

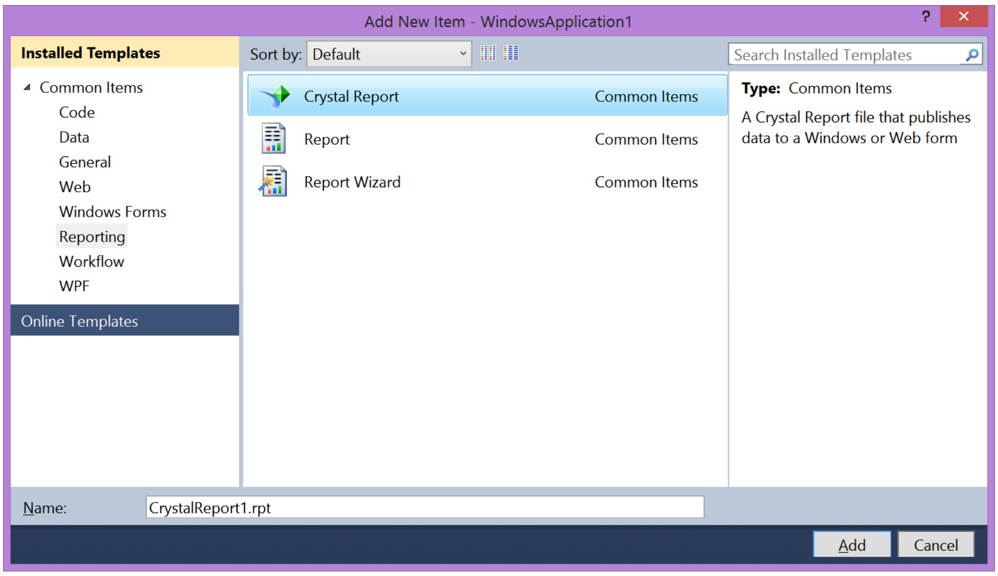


Figure 4.22: Addition of a Reporting Agent

As per VB.NET GUI design concepts, WISAM has to have a Multiple-Document Interface (MDI) form; which would act as the primary container for other modelling forms in the software (see figure 4.23). Afterwards, the Toolbox components at VB.NET IDE were used to design and create several windows forms for WISAM. Figures 4.24 – 4.30 show the major GUI forms that respectively illustrate WISAM's Modelling Window, WISAM's Properties Window, ToolBox Window, as well as different pages of WISAM's U-Plug Builder Interface.

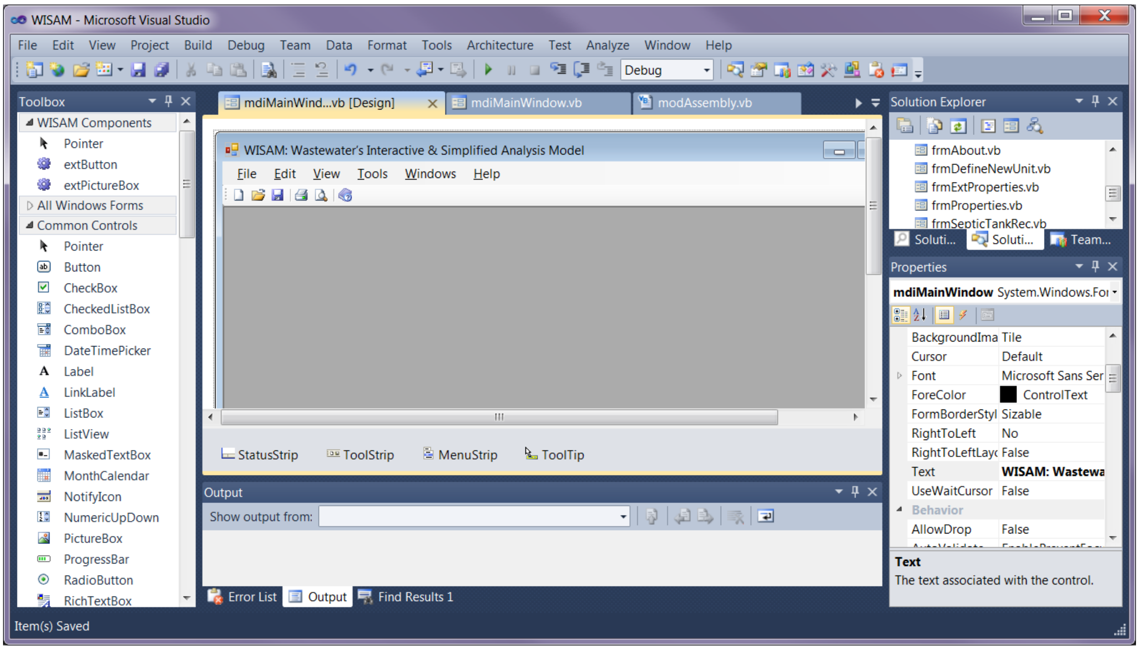


Figure 4.23: Creation of WISAM's Main MDI

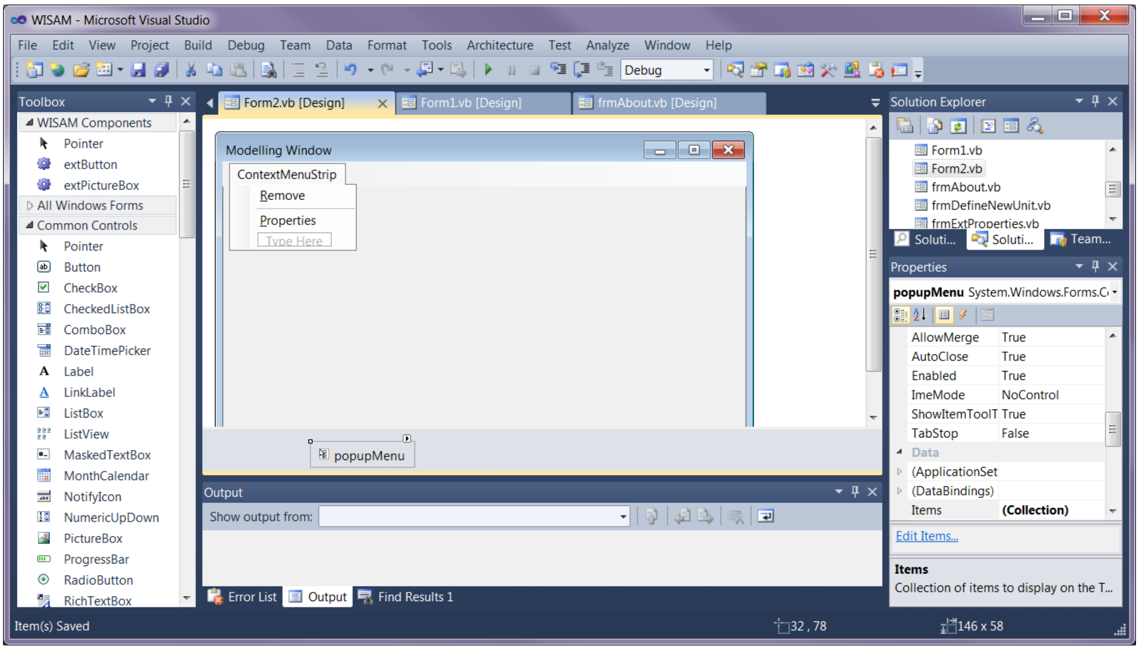


Figure 4.24: Creation of WISAM's Modelling Window

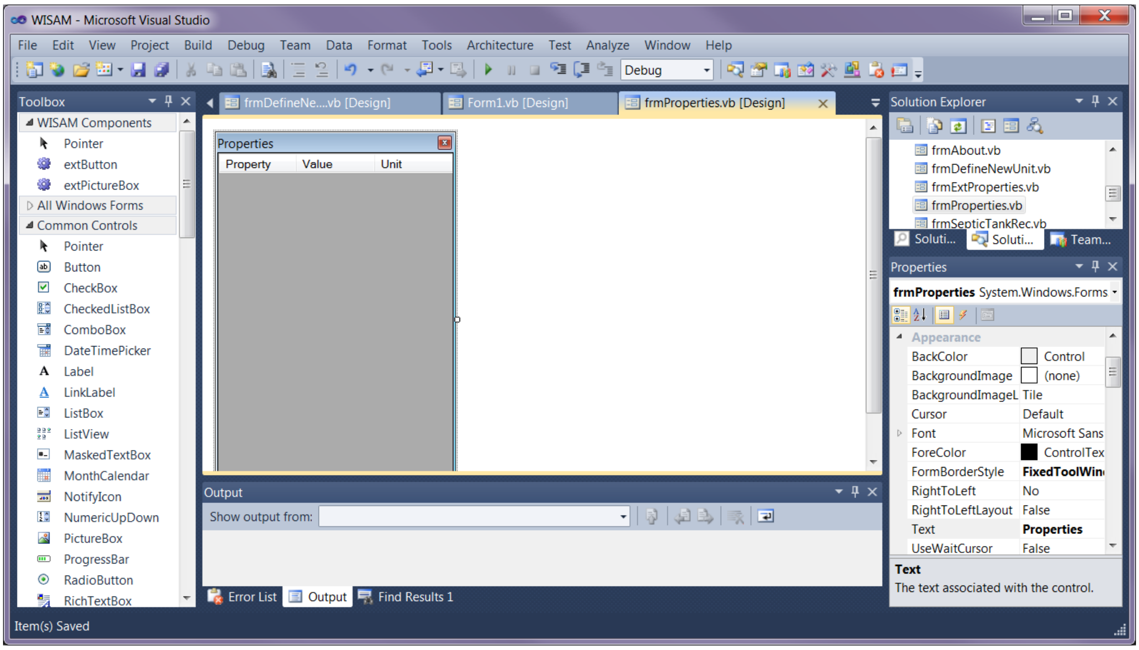


Figure 4.25: Creation of WISAM's Properties Window

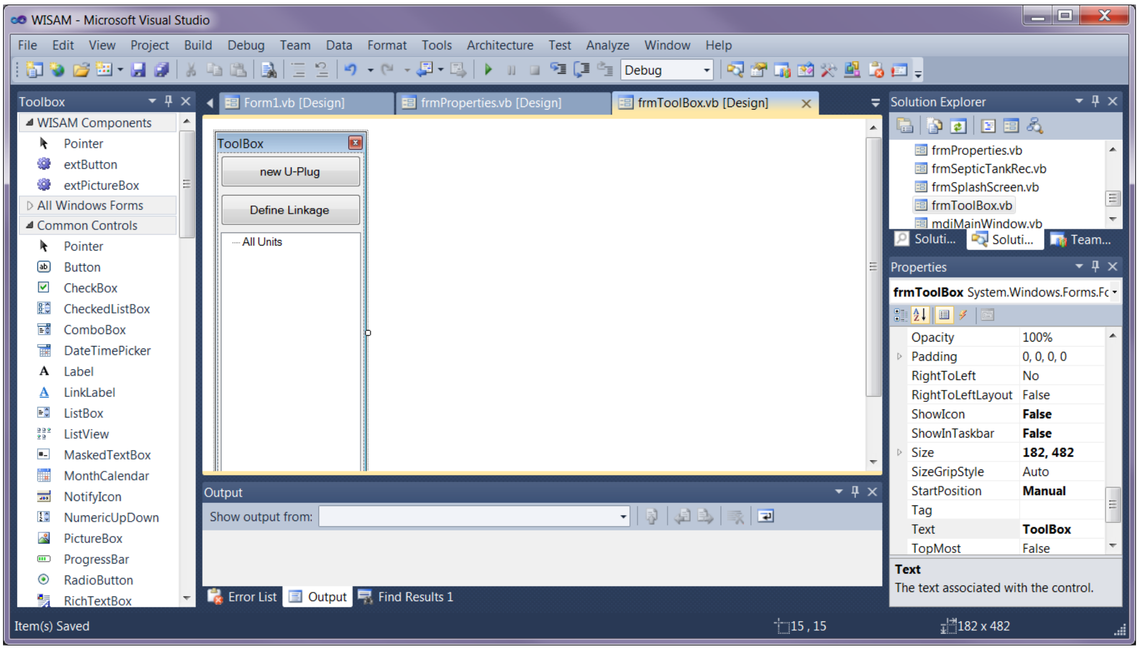


Figure 4.26: Creation of WISAM's ToolBox Window

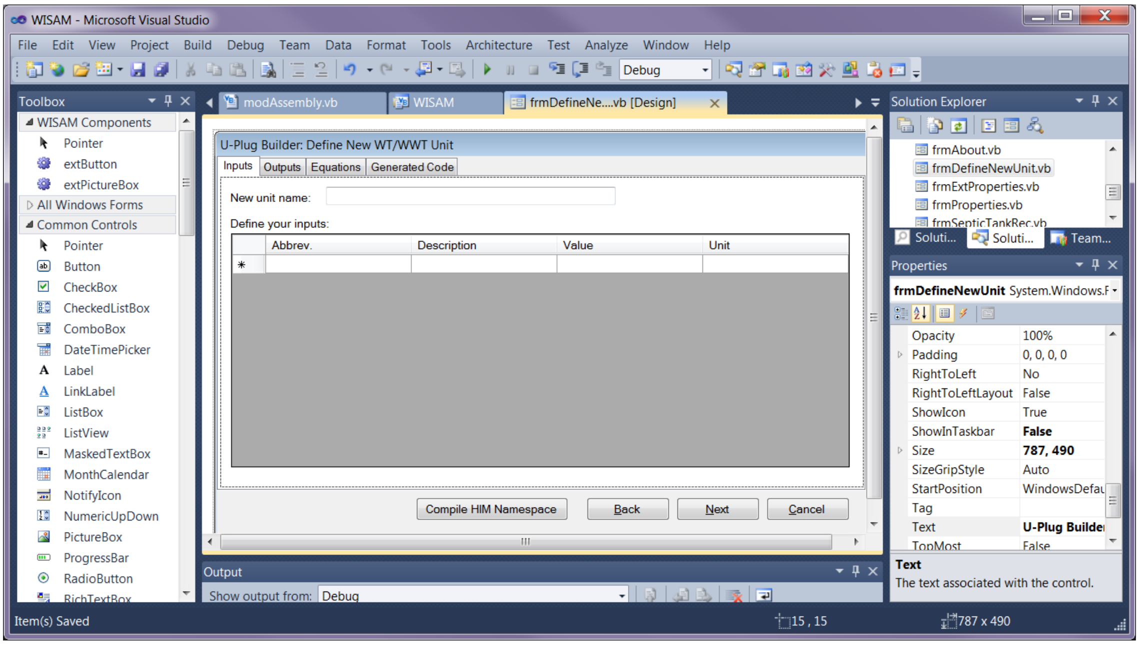


Figure 4.27: Creation of WISAM's U-Plug Builder Interface (the Inputs page)

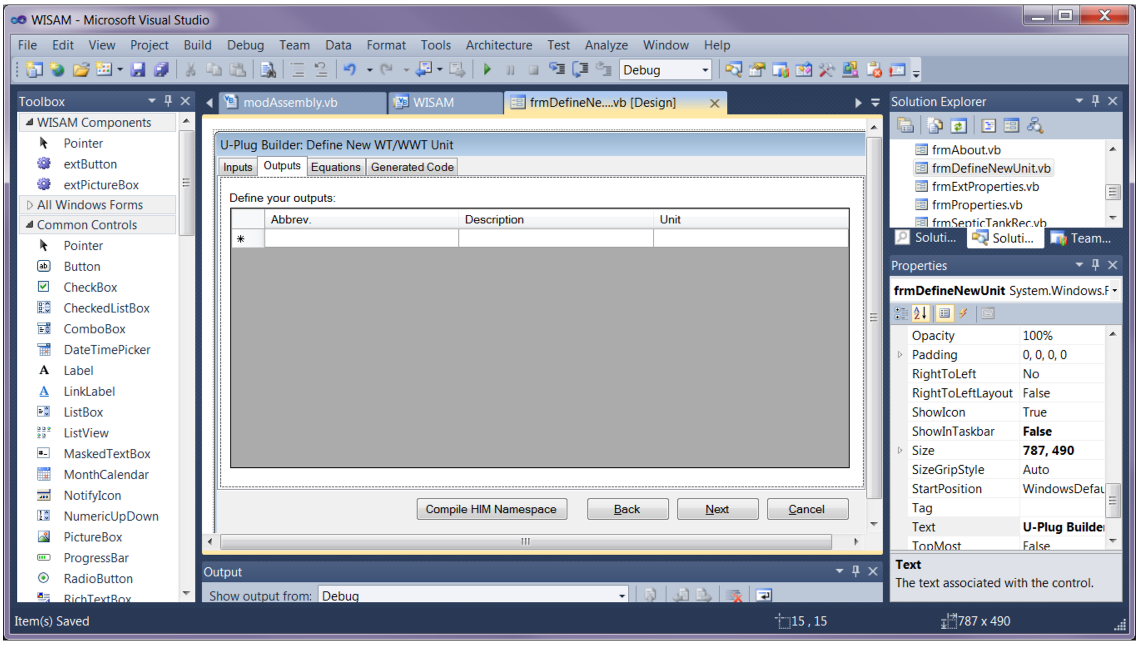


Figure 4.28: Creation of WISAM's U-Plug Builder Interface (the Outputs page)

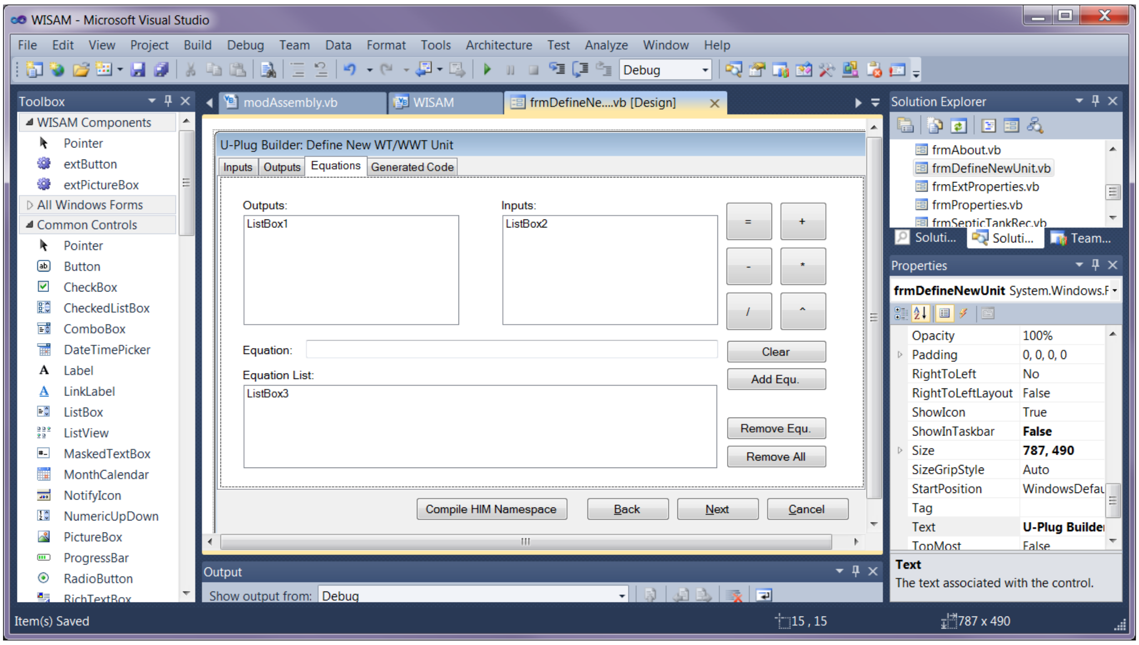


Figure 4.29: Creation of WISAM's U-Plug Builder Interface (the Equations page)

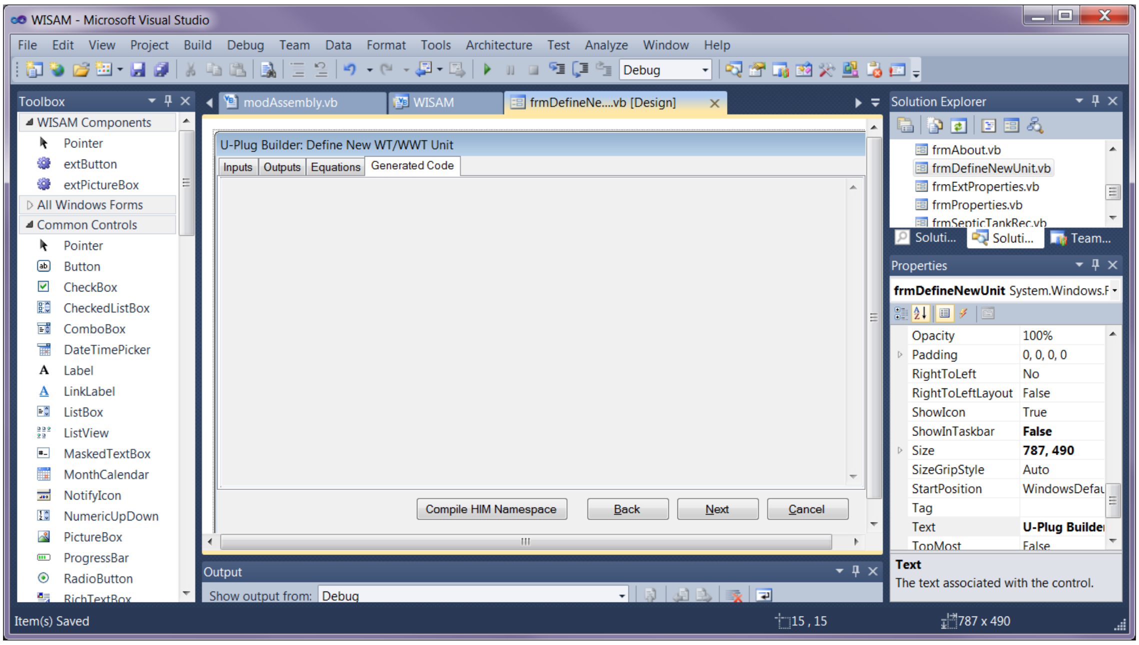


Figure 4.30: Creation of WISAM's U-Plug Builder Interface

(the Code Compilation page)

Finally, VB.NET Code Editor has been thoroughly used to construct WISAM's design engine and operational commands. Figure 4.31 represents VB.NET Code Editor whilst the Appendix to this thesis contains WISAM's core code in VB.NET language. Additionally, manual validation and verification has been carried out for the used equations in each modelled treatment unit according to relevant international standard and/or code of practice.

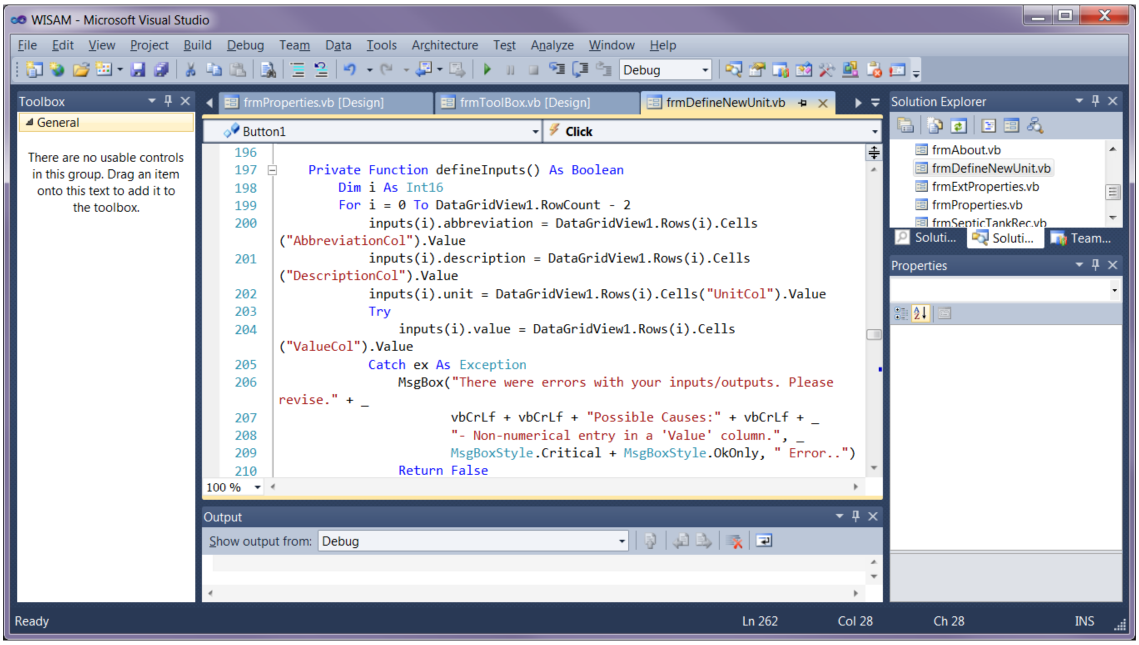


Figure 4.31: Formulation of WISAM's Programming Code

4.1.2b Development of MS Excel Calculation Sheets

Microsoft Excel was used to develop a calculation sheet for the structural analysis of rectangular water retaining structures. The adopted technical procedure followed the Eurocode2 recommendations as explained by Reynolds *et.al* (2008) and Threlfall (2013).

The developed calculation sheet (shown on figure 4.32) is intended to serve both as a stand-alone design model in addition to its primary purpose as an example for WISAM's integration with external design engines. The former approach is facilitated by embedding various tables of design coefficients (namely: moments, shear forces, and stiffness) within the calculation sheet (see figure 4.33); as well as automating the proper selection of the same.

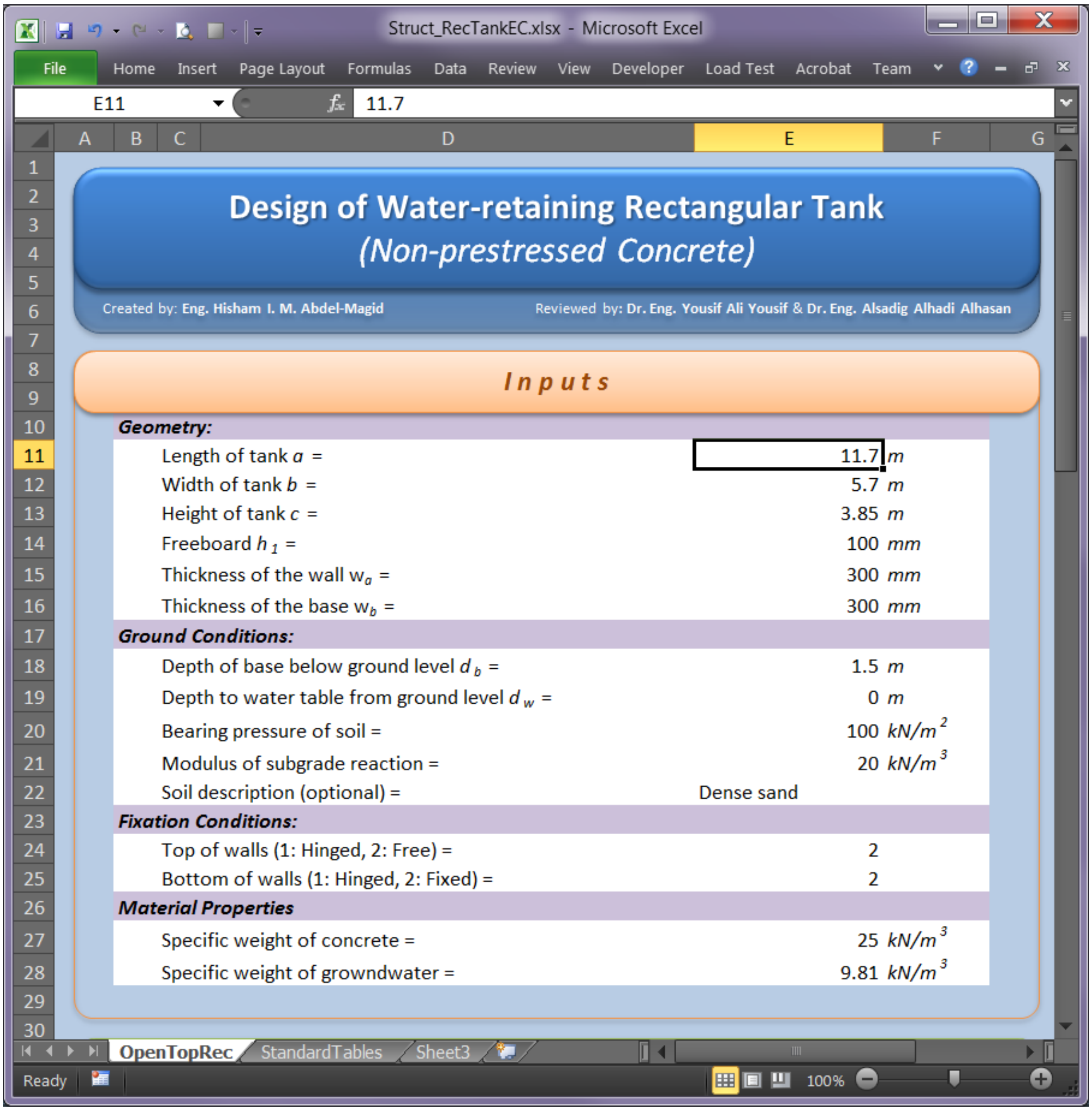
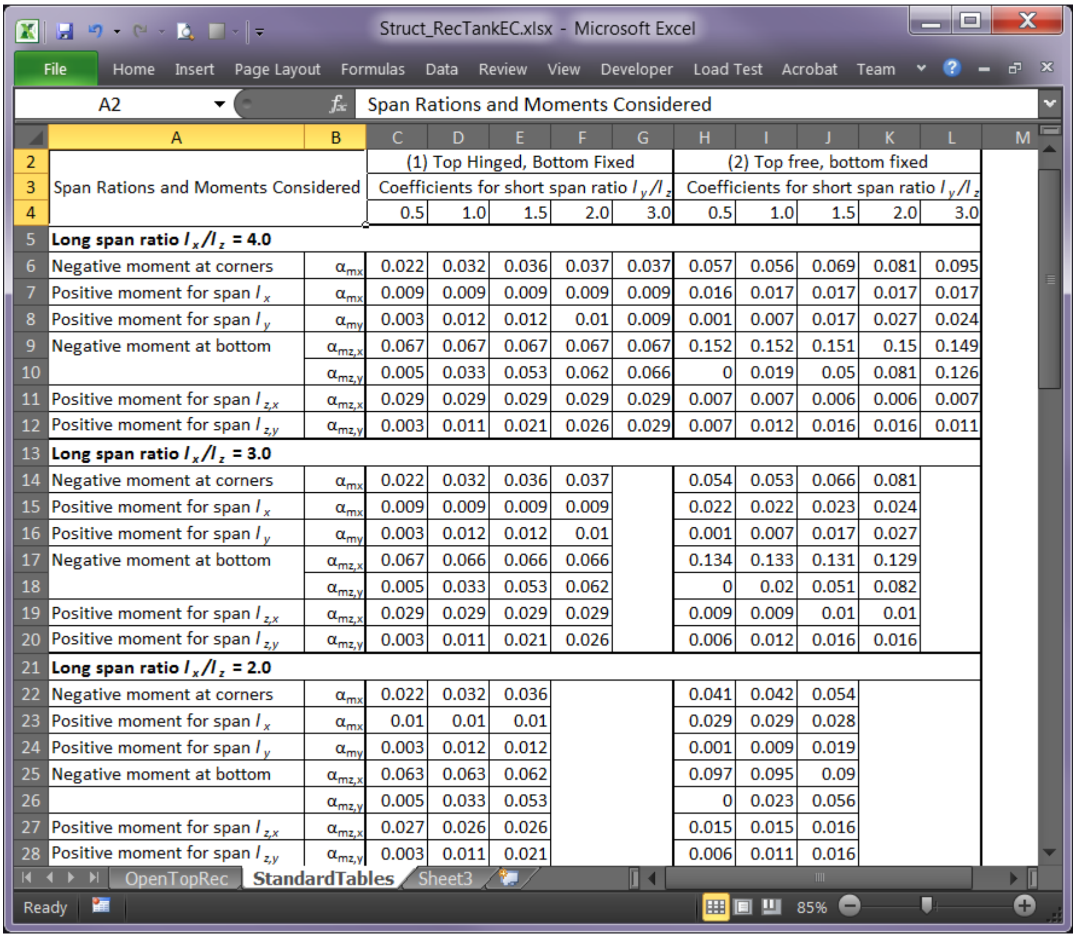


Figure 4.32: Calculation Sheet of Rectangular Concrete Tanks



	A	B	C	D	E	F	G	H	I	J	K	L
2	Span Rations and Moments Considered		(1) Top Hinged, Bottom Fixed					(2) Top free, bottom fixed				
3			Coefficients for short span ratio l_y/l_z					Coefficients for short span ratio l_y/l_z				
4			0.5	1.0	1.5	2.0	3.0	0.5	1.0	1.5	2.0	3.0
5	**Long span ratio l_x/l_z = 4.0**											
6	Negative moment at corners	α_{mx}	0.022	0.032	0.036	0.037	0.037	0.057	0.056	0.069	0.081	0.095
7	Positive moment for span l_x	α_{mx}	0.009	0.009	0.009	0.009	0.009	0.016	0.017	0.017	0.017	0.017
8	Positive moment for span l_y	α_{my}	0.003	0.012	0.012	0.01	0.009	0.001	0.007	0.017	0.027	0.024
9	Negative moment at bottom	$\alpha_{mz,x}$	0.067	0.067	0.067	0.067	0.067	0.152	0.152	0.151	0.15	0.149
10		$\alpha_{mz,y}$	0.005	0.033	0.053	0.062	0.066	0	0.019	0.05	0.081	0.126
11	Positive moment for span $l_{z,x}$	$\alpha_{mz,x}$	0.029	0.029	0.029	0.029	0.029	0.007	0.007	0.006	0.006	0.007
12	Positive moment for span $l_{z,y}$	$\alpha_{mz,y}$	0.003	0.011	0.021	0.026	0.029	0.007	0.012	0.016	0.016	0.011
13	**Long span ratio l_x/l_z = 3.0**											
14	Negative moment at corners	α_{mx}	0.022	0.032	0.036	0.037		0.054	0.053	0.066	0.081	
15	Positive moment for span l_x	α_{mx}	0.009	0.009	0.009	0.009		0.022	0.022	0.023	0.024	
16	Positive moment for span l_y	α_{my}	0.003	0.012	0.012	0.01		0.001	0.007	0.017	0.027	
17	Negative moment at bottom	$\alpha_{mz,x}$	0.067	0.066	0.066	0.066		0.134	0.133	0.131	0.129	
18		$\alpha_{mz,y}$	0.005	0.033	0.053	0.062		0	0.02	0.051	0.082	
19	Positive moment for span $l_{z,x}$	$\alpha_{mz,x}$	0.029	0.029	0.029	0.029		0.009	0.009	0.01	0.01	
20	Positive moment for span $l_{z,y}$	$\alpha_{mz,y}$	0.003	0.011	0.021	0.026		0.006	0.012	0.016	0.016	
21	**Long span ratio l_x/l_z = 2.0**											
22	Negative moment at corners	α_{mx}	0.022	0.032	0.036			0.041	0.042	0.054		
23	Positive moment for span l_x	α_{mx}	0.01	0.01	0.01			0.029	0.029	0.028		
24	Positive moment for span l_y	α_{my}	0.003	0.012	0.012			0.001	0.009	0.019		
25	Negative moment at bottom	$\alpha_{mz,x}$	0.063	0.063	0.062			0.097	0.095	0.09		
26		$\alpha_{mz,y}$	0.005	0.033	0.053			0	0.023	0.056		
27	Positive moment for span $l_{z,x}$	$\alpha_{mz,x}$	0.027	0.026	0.026			0.015	0.015	0.016		
28	Positive moment for span $l_{z,y}$	$\alpha_{mz,y}$	0.003	0.011	0.021			0.006	0.011	0.016		

Figure 4.33: Embedded Design Coefficients in The Calculation Sheet of Rectangular Concrete Tanks

4.1.2c GIS Integration

In order to explore the Geographic Information System (GIS) potential in this research work, ESRI's computer package ArcGIS v10.1 has been utilised. It has been noticed from the detailed design factors, provided by the Portland Cement Association (Munshi, 1998) in more than 250 tables, that a systematic grid layer could be mimicked using a GIS *shapefile* concept. Microsoft Excel was used to rearrange the tables in a specific manner that allows later import into GIS environment.

Table 4.2 reflects an example of the original tables as provided in wall fixation Case#3 by Munshi (1998) for the moment coefficient of a Free Top & Fixed Base tank walls (considering a length/height ratio of b/c=3.0 and width/height ration of c/a=1.5). The location of each moment is presented in a 0.1 distance interval at each side of the wall. Table 4.3 shows the suitably rearranged table of the fixation Case#3 for further import into GIS environment, where columns *b_fac_X* & *a_fac_Y* represents the X & Y coordinates of each point respectively.

Table 4.2: Sample of Moment Coefficient Factors at Tank Walls for Case#3: Free Top & Fixed Base (Munshi, 1998)

b/a = 3.0, c/a=1.5		Mx Coefficient						My Coefficient						Mxy Coefficient					
		Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b	Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b	Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b
Long Side	**TOP**	-9	0	0	0	0	0	-44	-15	11	20	23	23	6	17	17	13	7	0
	0.9a	-13	-2	2	4	4	5	-66	-13	11	19	21	21	8	15	17	13	7	0
	0.8a	-12	-2	4	7	8	8	-61	-11	11	18	19	19	8	15	17	13	7	0
	0.7a	-11	0	6	9	10	10	-57	-9	10	16	17	16	8	16	17	13	7	0
	0.6a	-11	1	7	9	9	8	-53	-6	10	13	14	13	8	17	18	14	7	0
	0.5a	-10	1	6	5	4	3	-48	-4	8	10	10	9	7	18	18	13	7	0
	0.4a	-8	0	1	-3	-7	-8	-41	-2	6	6	5	4	7	19	18	13	6	0
	0.3a	-6	-3	-9	-17	-23	-26	-31	-1	2	1	-1	-1	6	18	17	11	5	0
	0.2a	-4	-9	-25	-39	-48	-51	-18	-2	-3	-6	-8	-8	4	16	13	9	4	0
	0.1a	-1	-21	-49	-70	-82	-86	-6	-4	-9	-13	-16	-17	3	11	8	5	2	0
	BOT.	0	-41	-84	-112	-126	-131	0	-8	-17	-22	-25	-26	0	0	0	0	0	0

b/a = 3.0, c/a=1.5		Mx Coefficient						My Coefficient						Mxy Coefficient					
		Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b	Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b	Corner	0.1b / 0.9b	0.2b / 0.8b	0.3b / 0.7b	0.4b / 0.6b	0.5b
Short Side	**TOP**	-9	0	0	0	0	0	-44	-35	-11	5	14	17	6	3	0	1	1	0
	0.9a	-13	-5	-1	1	2	2	-66	-32	-10	6	14	17	8	4	1	1	1	0
	0.8a	-12	-6	-1	3	5	6	-61	-30	-8	7	15	17	8	4	1	1	1	0
	0.7a	-11	-4	2	6	9	10	-57	-26	-5	8	15	17	8	4	0	1	1	0
	0.6a	-11	-3	4	10	13	14	-53	-22	-3	9	15	17	8	2	1	2	2	0
	0.5a	-10	-1	7	12	15	16	-48	-18	-1	9	14	15	7	1	3	4	3	0
	0.4a	-8	1	8	13	15	16	-41	-13	1	8	12	13	7	2	5	5	3	0
	0.3a	-6	2	7	10	11	11	-31	-8	2	6	8	9	6	4	7	7	4	0
	0.2a	-4	1	3	2	1	0	-18	-4	1	3	4	4	4	6	8	7	4	0
	0.1a	-1	-2	-7	-14	-18	-20	-6	-2	-1	-2	-3	-3	3	5	6	5	3	0
	BOT.	0	-8	-25	-39	-48	-51	0	-2	-5	-8	-10	-10	0	0	0	0	0	0

Note: *Moment = Coef.* $\times qa^2/1000$

Table 4.3: Rearranged Moment Coefficient for GIS Import

b_fac_X	a_fac_Y	C3Mx_L	C3My_L	C3Mxy_L	C3Mx_S	C3My_S	C3Mxy_S
0	10	-9	-44	6	-9	-44	6
0	9	-13	-66	8	-13	-66	8
0	8	-12	-61	8	-12	-61	8
0	7	-11	-57	8	-11	-57	8
0	6	-11	-53	8	-11	-53	8
0	5	-10	-48	7	-10	-48	7
0	4	-8	-41	7	-8	-41	7
0	3	-6	-31	6	-6	-31	6
0	2	-4	-18	4	-4	-18	4
0	1	-1	-6	3	-1	-6	3
0	0	0	0	0	0	0	0
1	10	0	-15	17	0	-35	3
1	9	-2	-13	15	-5	-32	4
1	8	-2	-11	15	-6	-30	4
1	7	0	-9	16	-4	-26	4
1	6	1	-6	17	-3	-22	2
1	5	1	-4	18	-1	-18	1
1	4	0	-2	19	1	-13	2
1	3	-3	-1	18	2	-8	4
1	2	-9	-2	16	1	-4	6
1	1	-21	-4	11	-2	-2	5
1	0	-41	-8	0	-8	-2	0
2	10	0	11	17	0	-11	0
2	9	2	11	17	-1	-10	1
2	8	4	11	17	-1	-8	1
2	7	6	10	17	2	-5	0
2	6	7	10	18	4	-3	1
2	5	6	8	18	7	-1	3
2	4	1	6	18	8	1	5
2	3	-9	2	17	7	2	7
2	2	-25	-3	13	3	1	8
2	1	-49	-9	8	-7	-1	6
2	0	-84	-17	0	-25	-5	0
3	10	0	20	13	0	5	1
3	9	4	19	13	1	6	1
3	8	7	18	13	3	7	1
3	7	9	16	13	6	8	1
3	6	9	13	14	10	9	2
3	5	5	10	13	12	9	4
3	4	-3	6	13	13	8	5
3	3	-17	1	11	10	6	7
3	2	-39	-6	9	2	3	7
3	1	-70	-13	5	-14	-2	5
3	0	-112	-22	0	-39	-8	0
4	10	0	23	7	0	14	1
4	9	4	21	7	2	14	1
4	8	8	19	7	5	15	1
4	7	10	17	7	9	15	1
4	6	9	14	7	13	15	2
4	5	4	10	7	15	14	3
4	4	-7	5	6	15	12	3
4	3	-23	-1	5	11	8	4
4	2	-48	-8	4	1	4	4
4	1	-82	-16	2	-18	-3	3
4	0	-126	-25	0	-48	-10	0
5	10	0	23	0	0	17	0
5	9	5	21	0	2	17	0
5	8	8	19	0	6	17	0
5	7	10	16	0	10	17	0
5	6	8	13	0	14	17	0
5	5	3	9	0	16	15	0

5	4	-8	4	0	16	13	0
5	3	-26	-1	0	11	9	0
5	2	-51	-8	0	0	4	0
5	1	-86	-17	0	-20	-3	0
5	0	-131	-26	0	-51	-10	0
6	10	0	23	7	0	14	1
6	9	4	21	7	2	14	1
6	8	8	19	7	5	15	1
6	7	10	17	7	9	15	1
6	6	9	14	7	13	15	2
6	5	4	10	7	15	14	3
6	4	-7	5	6	15	12	3
6	3	-23	-1	5	11	8	4
6	2	-48	-8	4	1	4	4
6	1	-82	-16	2	-18	-3	3
6	0	-126	-25	0	-48	-10	0
7	10	0	20	13	0	5	1
7	9	4	19	13	1	6	1
7	8	7	18	13	3	7	1
7	7	9	16	13	6	8	1
7	6	9	13	14	10	9	2
7	5	5	10	13	12	9	4
7	4	-3	6	13	13	8	5
7	3	-17	1	11	10	6	7
7	2	-39	-6	9	2	3	7
7	1	-70	-13	5	-14	-2	5
7	0	-112	-22	0	-39	-8	0
8	10	0	11	17	0	-11	0
8	9	2	11	17	-1	-10	1
8	8	4	11	17	-1	-8	1
8	7	6	10	17	2	-5	0
8	6	7	10	18	4	-3	1
8	5	6	8	18	7	-1	3
8	4	1	6	18	8	1	5
8	3	-9	2	17	7	2	7
8	2	-25	-3	13	3	1	8
8	1	-49	-9	8	-7	-1	6
8	0	-84	-17	0	-25	-5	0
9	10	0	-15	17	0	-35	3
9	9	-2	-13	15	-5	-32	4
9	8	-2	-11	15	-6	-30	4
9	7	0	-9	16	-4	-26	4
9	6	1	-6	17	-3	-22	2
9	5	1	-4	18	-1	-18	1
9	4	0	-2	19	1	-13	2
9	3	-3	-1	18	2	-8	4
9	2	-9	-2	16	1	-4	6
9	1	-21	-4	11	-2	-2	5
9	0	-41	-8	0	-8	-2	0
10	10	-9	-44	6	-9	-44	6
10	9	-13	-66	8	-13	-66	8
10	8	-12	-61	8	-12	-61	8
10	7	-11	-57	8	-11	-57	8
10	6	-11	-53	8	-11	-53	8
10	5	-10	-48	7	-10	-48	7
10	4	-8	-41	7	-8	-41	7
10	3	-6	-31	6	-6	-31	6
10	2	-4	-18	4	-4	-18	4
10	1	-1	-6	3	-1	-6	3
10	0	0	0	0	0	0	0

In order to visualise the generated points, ArcGIS Desktop package was considered (see figure 4.34). As it could be seen in figure 4.35, the addition of a geographically referenced data could be accessed via the tool menu item *Add XY Data* in ArcMap environment.

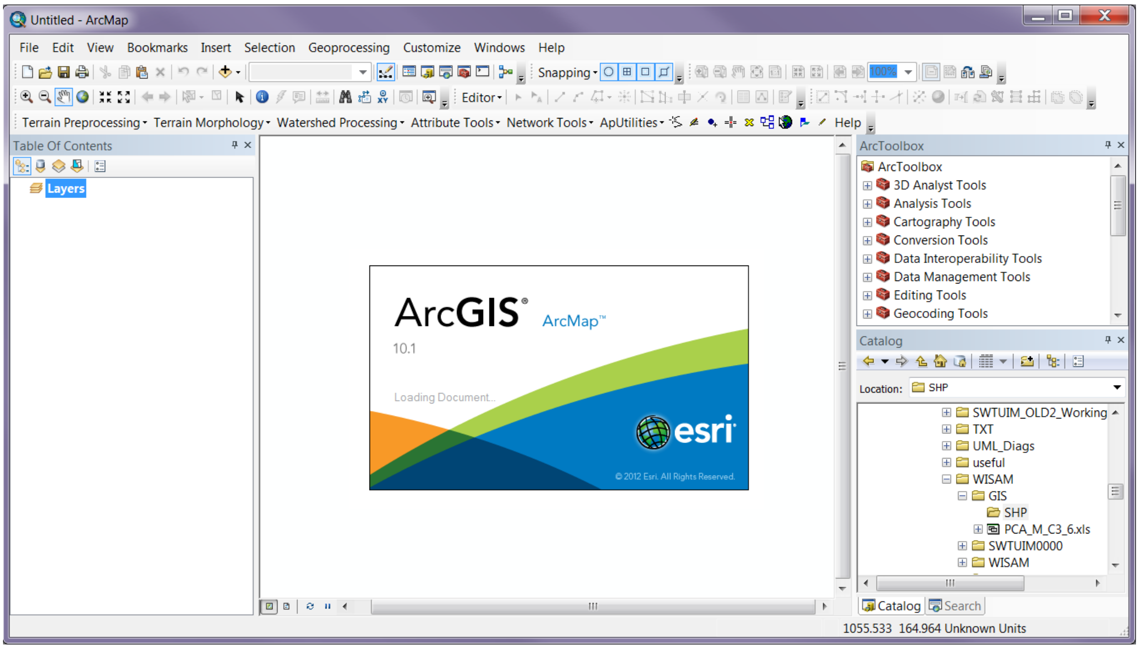


Figure 4.34: ArcGIS-ArcMap Software Interface

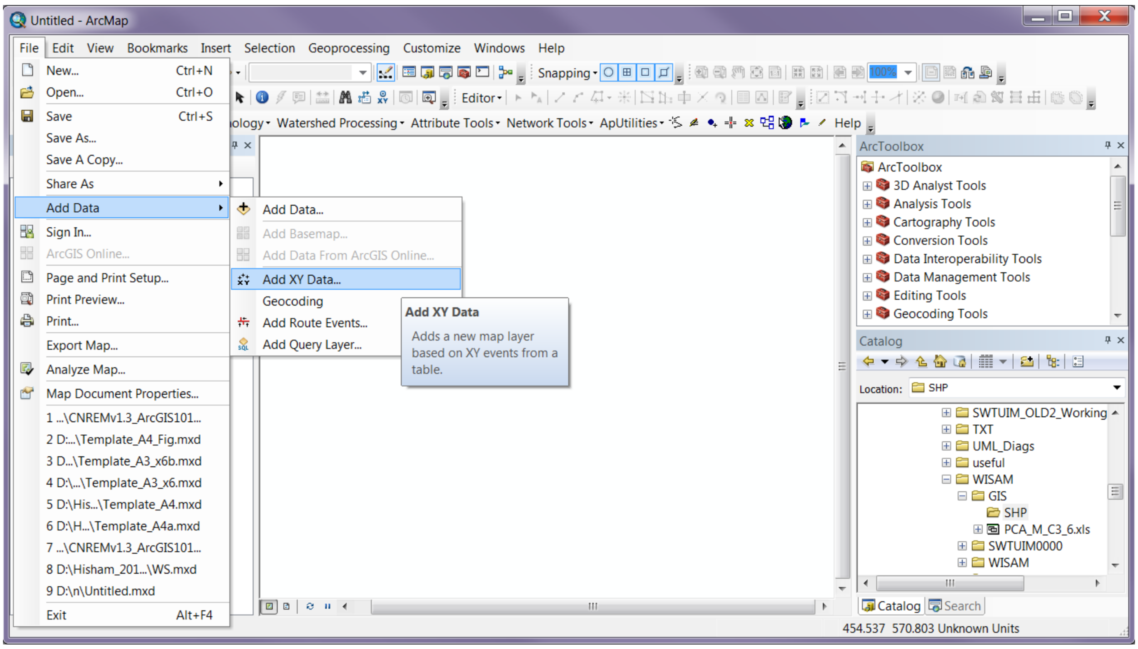


Figure 4.35: Adding XY GIS Data

The prepared GIS table is thereafter been selected and the corresponding coordinates fields have been identified (see figure 4.36). At this stage, no geographic system was selected due to the fact that this representation is merely for the purpose of calculations and not an actual representation of a geographic location in reality. The imported coefficient points table is shown in the following figure 4.37.

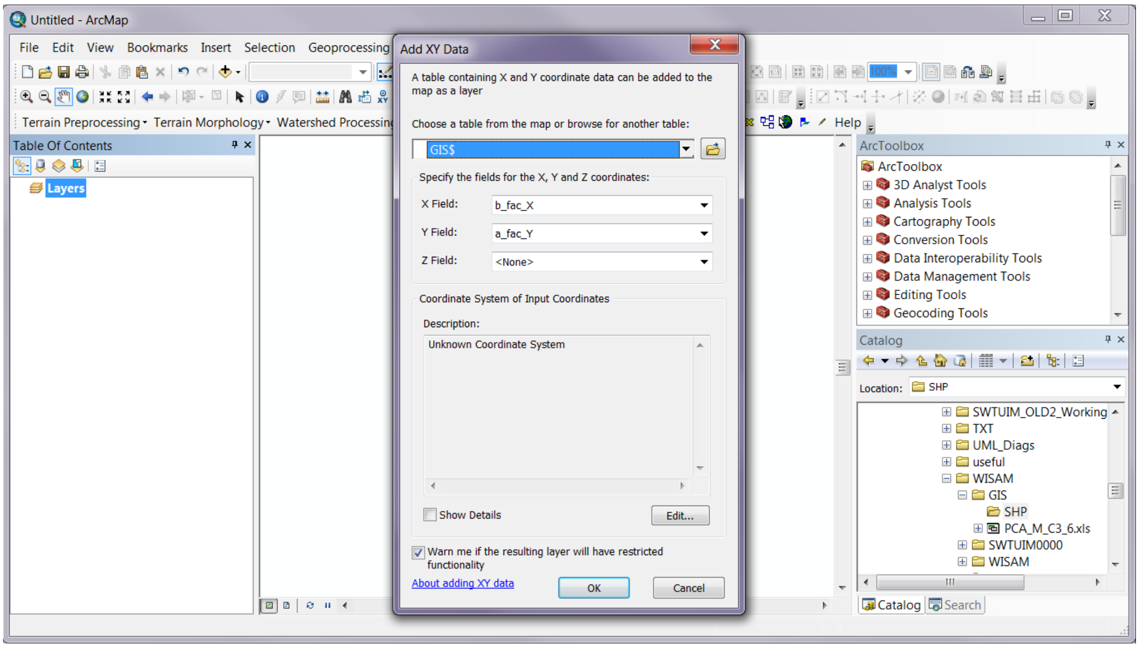


Figure 4.36: Identification of Geographic Coordinates

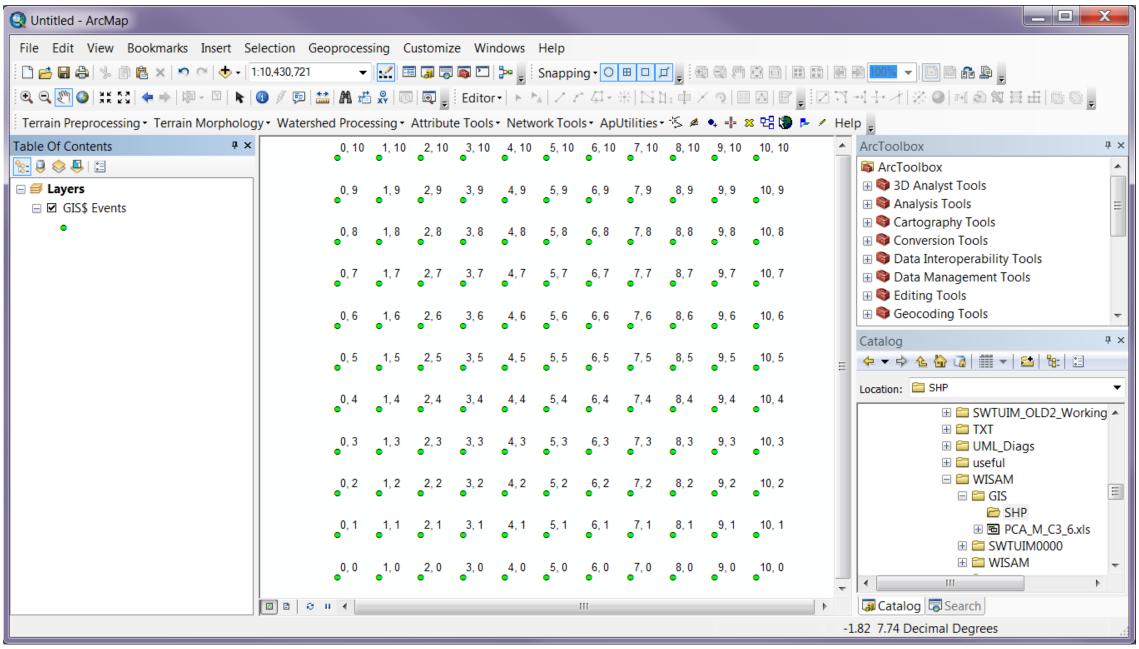


Figure 4.37: The Imported Coefficient Points Table

Since the imported coefficient points are displayed as a temporary event in GIS, a permanent copy has to be saved following the shown steps at figures 4.38 and 4.39. The imported coefficient values could be displayed via showing the *Attribute Table* of the resulting GIS shapefile as represented on figure 4.40.

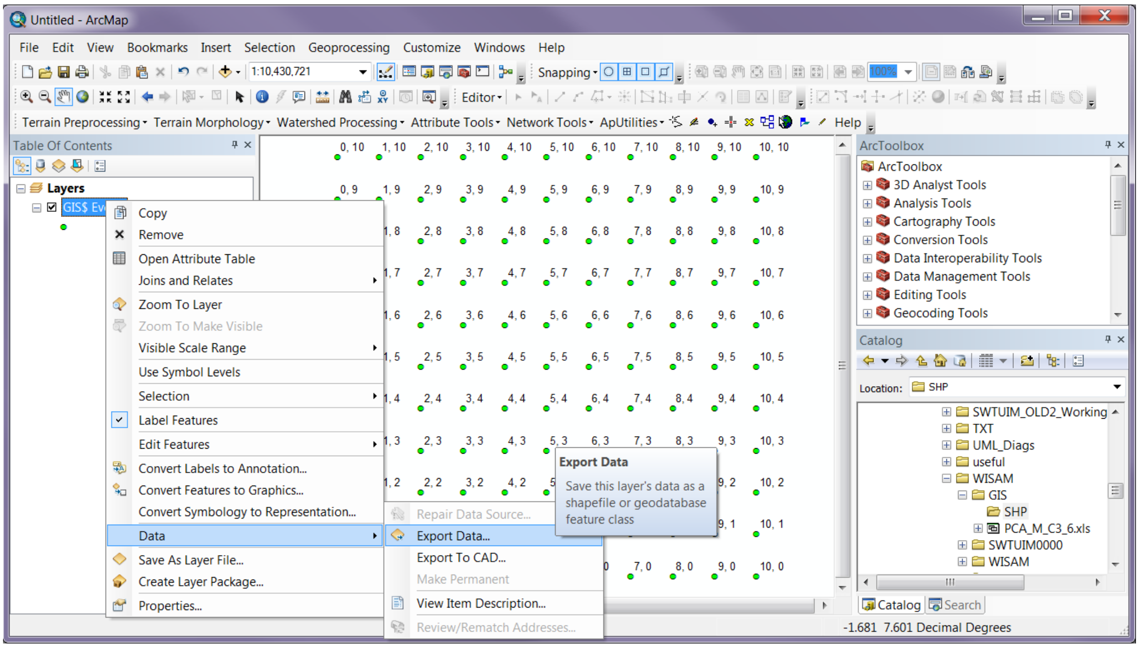


Figure 4.38: Exporting Point Data Event – Menu Item

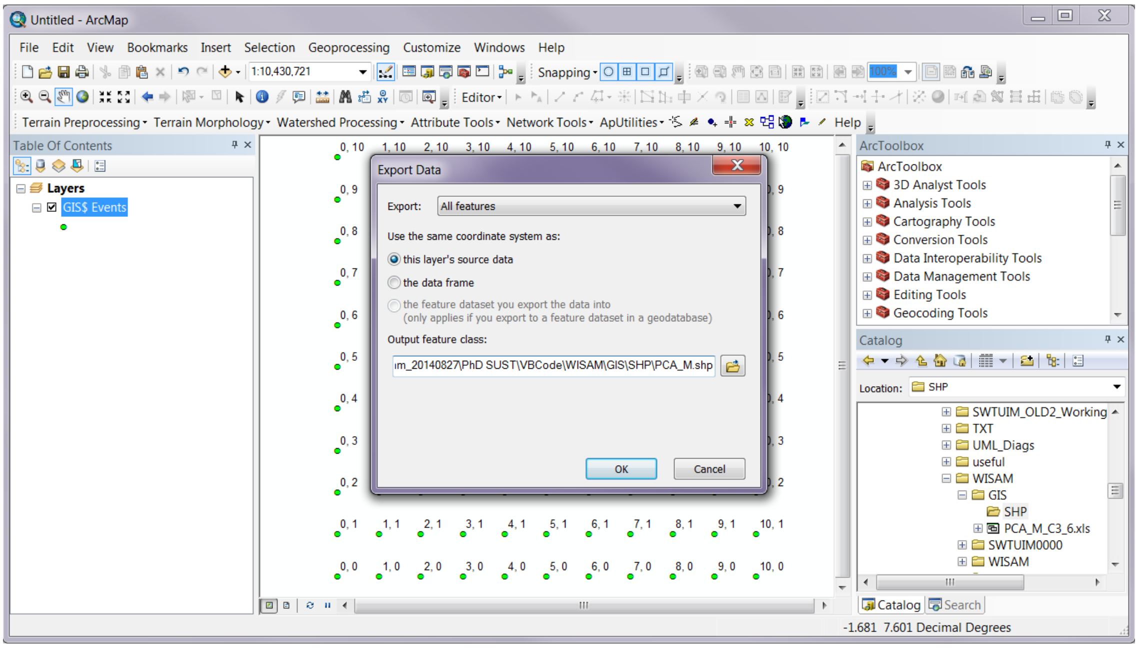


Figure 4.39: Exporting Point Data Event – Properties Window

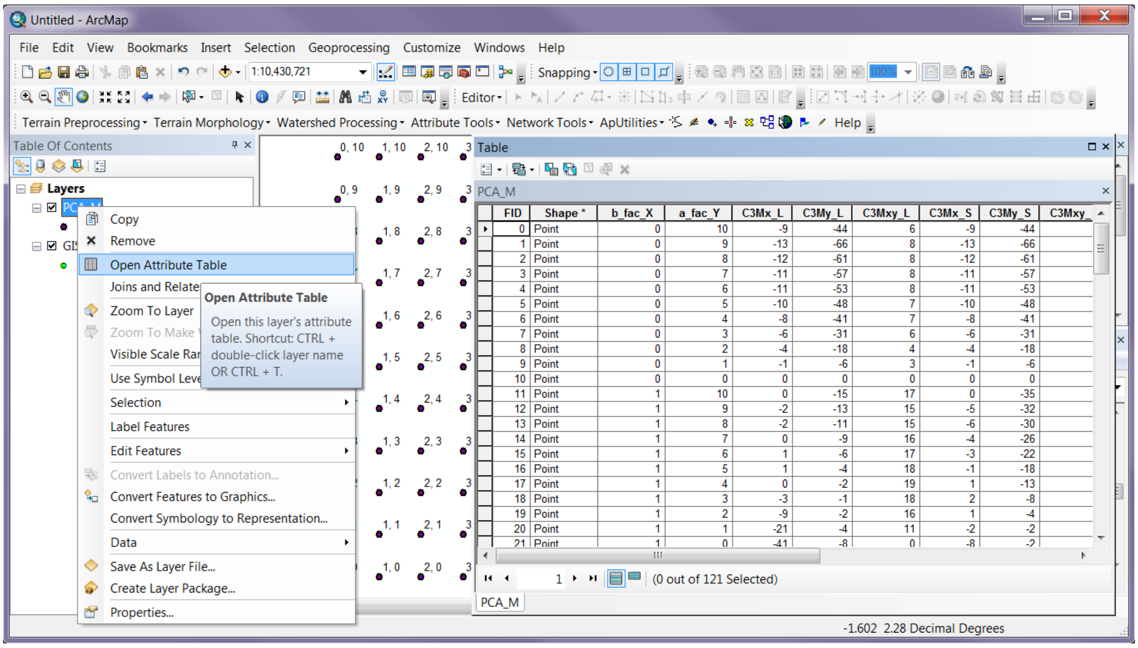


Figure 4.40: Display of Imported Coefficients via the Attribute Table

4.2 Discussion of Results

The successful development of WISAM as a stand-alone DSS, as well as defining its integration with external sources such as MS-Excel and GIS platforms, has been explored as discussed hereafter. Basically, three design approaches were identified for running the model:

1. The hydraulic design of three WT/WWT units (namely Screening, Flocculation & Coagulation, and Sedimentation tanks), which sets key examples of how WISAM's core engine could be used for defining U-Plugs on-the-fly. The design equations for all units followed the recommendations of Abdel-Magid *et al* (1997).
2. The structural design of concrete rectangular water-retaining tank via external call to the MS-Excel design calculation sheet, the thing that highlights WISAM's capability to externally link with other design engines. The calculated results were verified via manual calculations and the produced results report is embedded within the dynamic calculation sheet; as in line with the Eurocode2 recommendations on Reynolds *et.al* (2008) and Threlfall (2013).
3. Extended structural analysis by means of utilising GIS shapefile dataset and the potential of future integration with overall analysis and design of WTU/WWTU.

As it could be seen from figures 4.41 to 4.46, WISAM's Unit Builder was used to identify the list of inputs, outputs, governing equations, and VB.NET code generation for the Screening U-Plug. Similarly, flocculation & coagulation and sedimentation tank were compiled as shown on figures 4.47 and 4.48 respectively, whereas, the resulting calculation sheet output could be presented as shown on figure 4.49.

As for the GIS representation, figure 4.50 reflects the point distribution of calculated moments and figure 4.51 shows the included dataset of the same. With the GIS spatial interpolation capabilities, it is easy to generate an interpolated surface from the specified data points (see figure 4.52). Further enhancement of appearance could be achieved via colour coding, contouring, or even 3D representation (as shown on figure 4.53, 4.54 and 4.55 respectively).

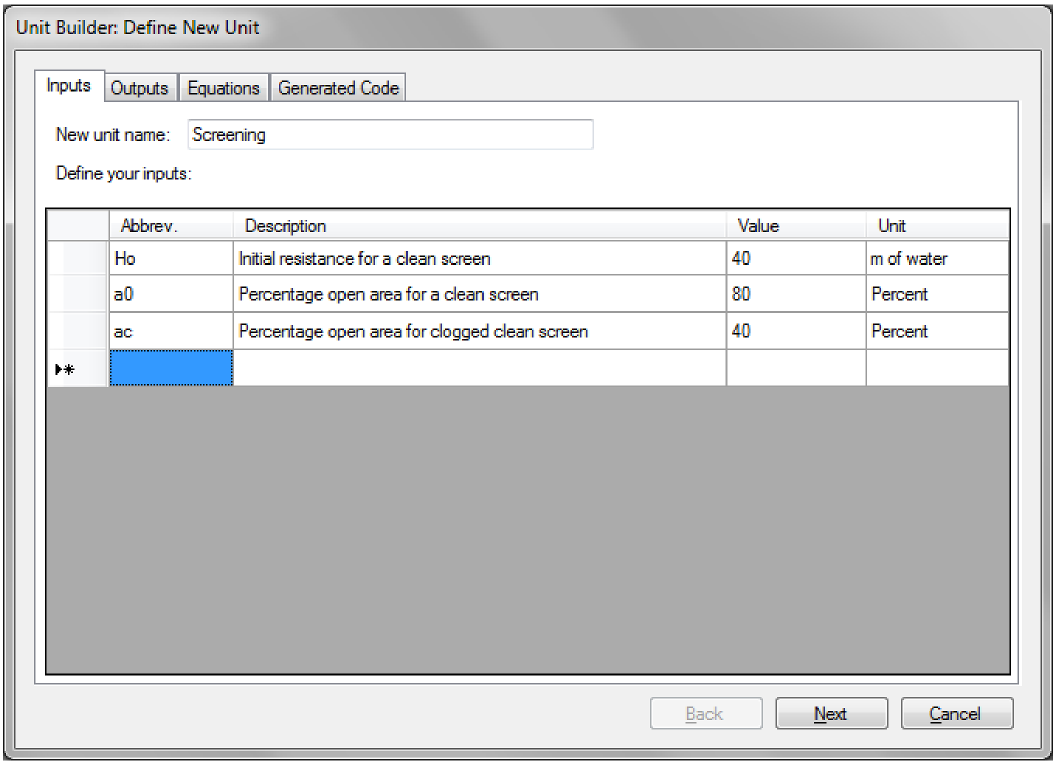


Figure 4.41: Definition of Screening U-Plug – Assignment of Inputs

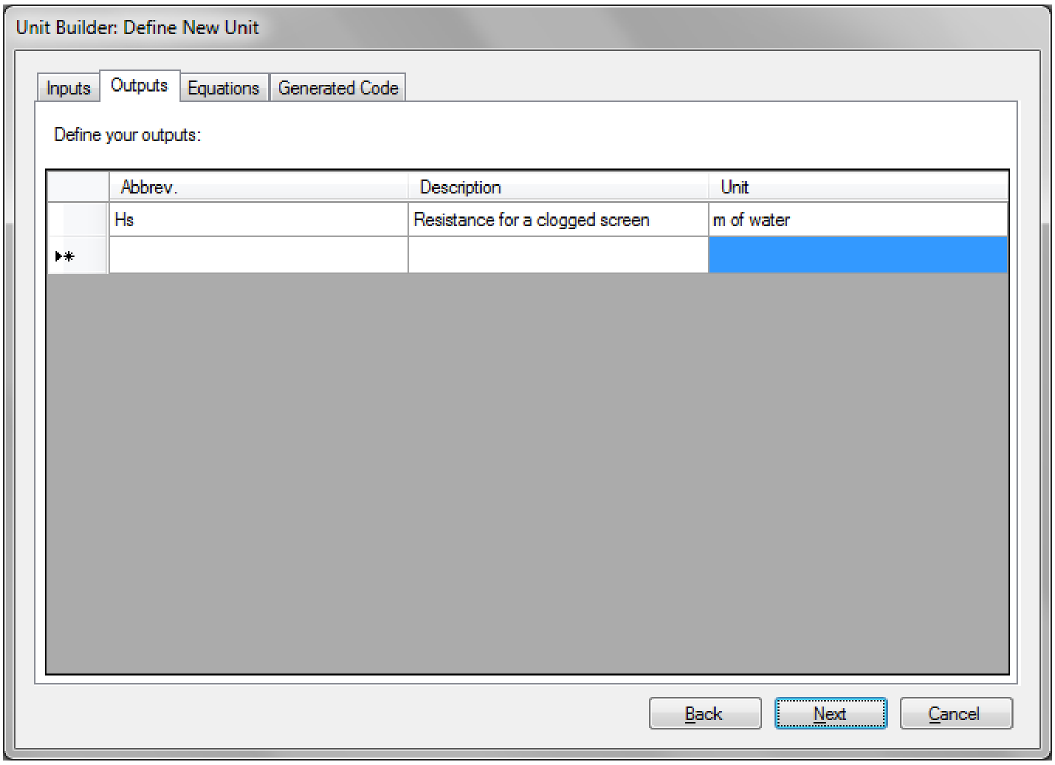


Figure 4.42: Definition of Screening U-Plug – Assignment of Outputs

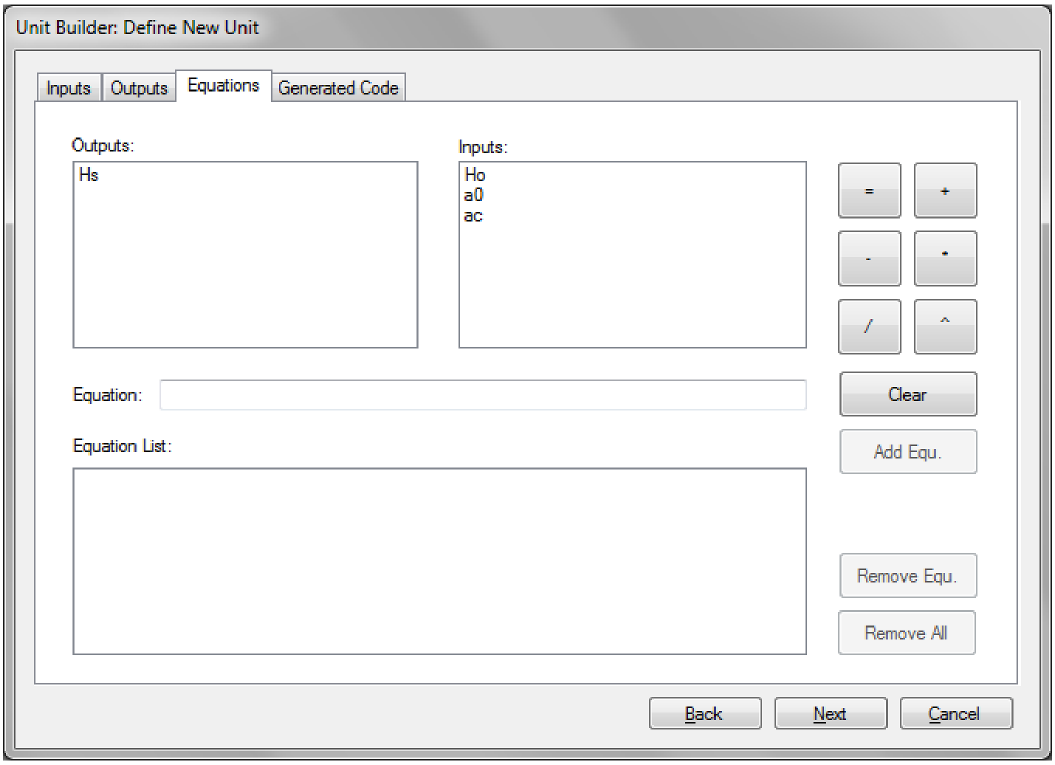


Figure 4.43: Definition of Screening U-Plug – Equation Setup

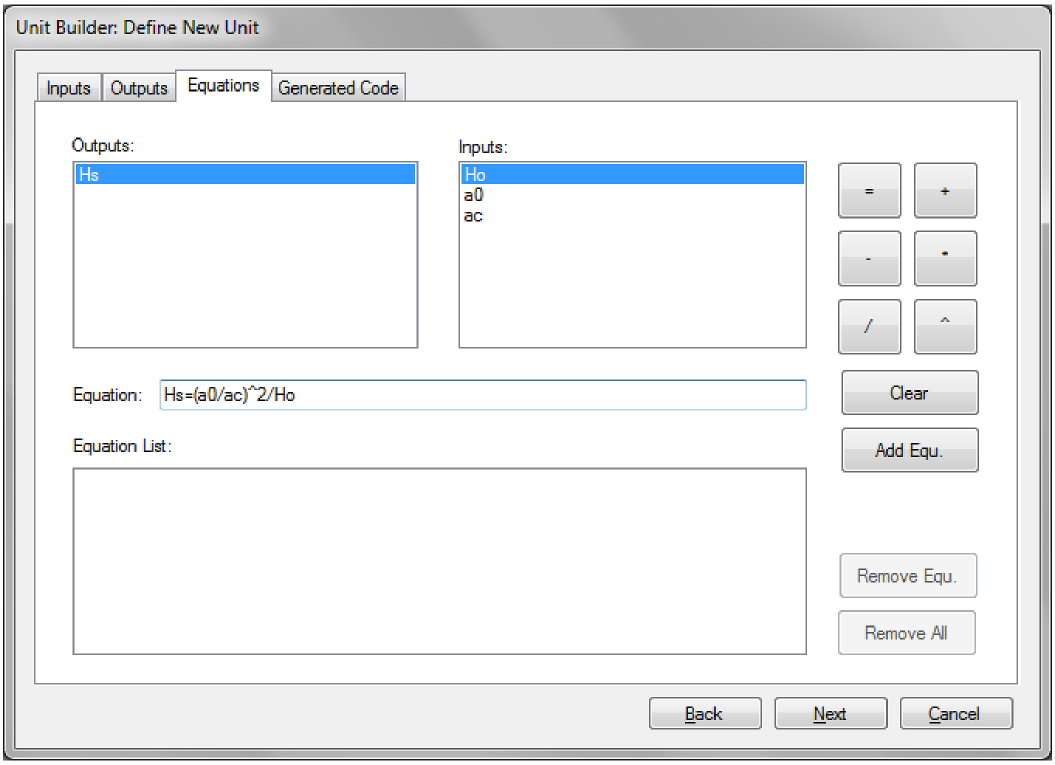


Figure 4.44: Definition of Screening U-Plug – Statement of Equation

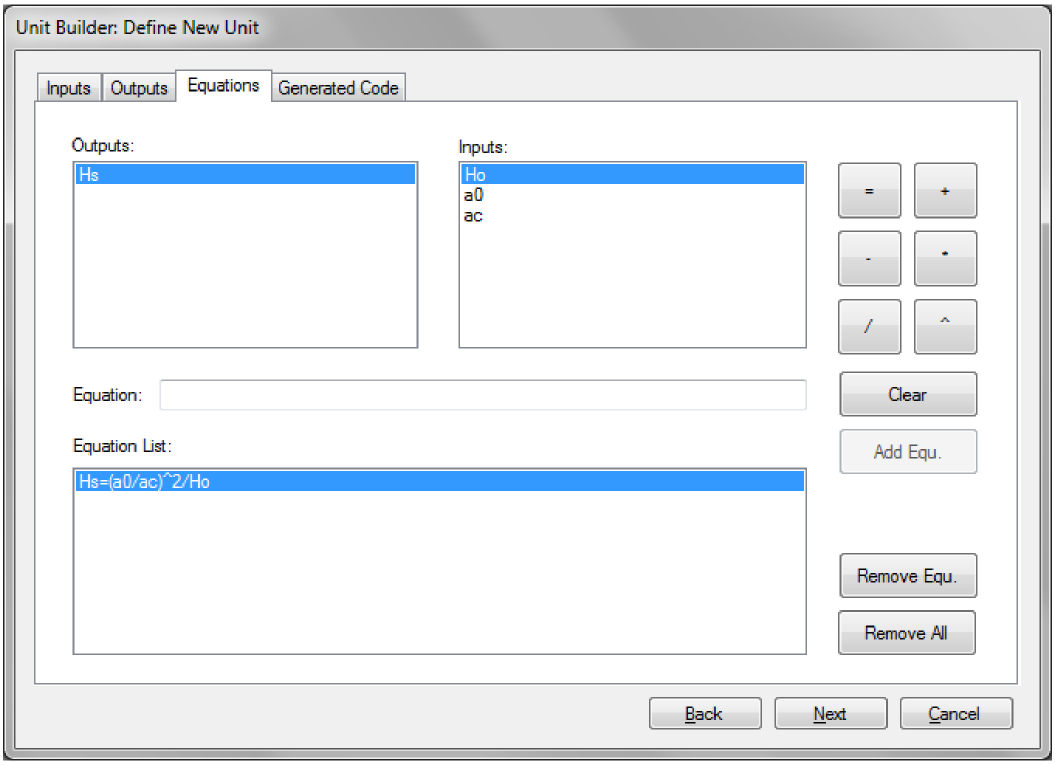


Figure 4.45: Definition of Screening U-Plug – Equation Listing

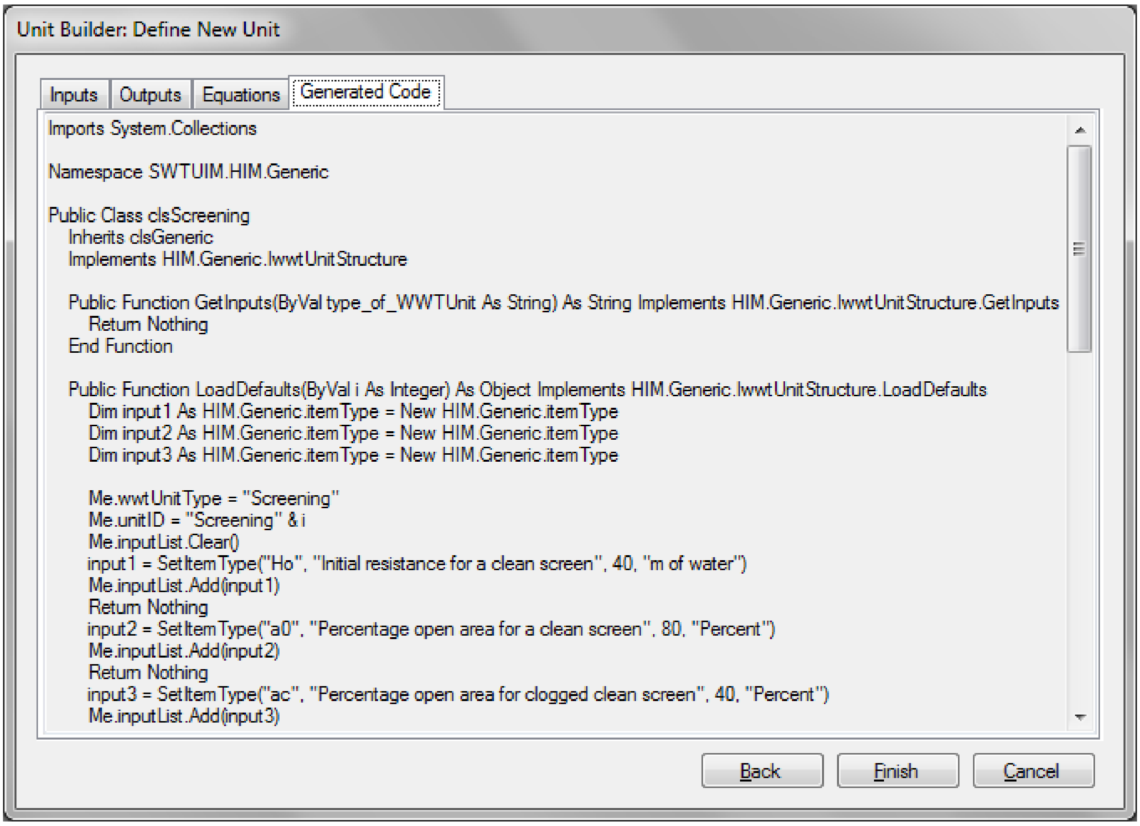


Figure 4.46: Definition of Screening U-Plug – VB.NET Code Generation

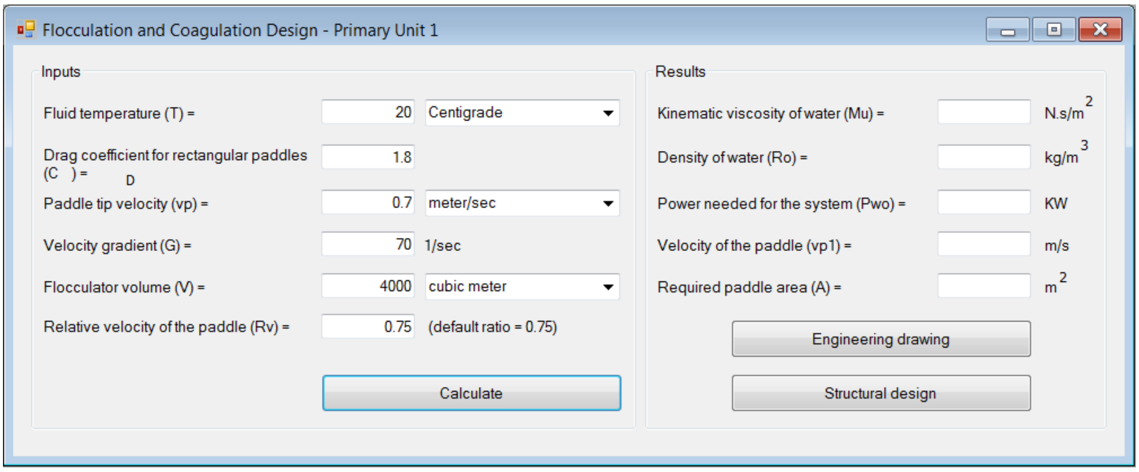


Figure 4.47: Final Model Window for a Flocculation & Coagulation U-Plug

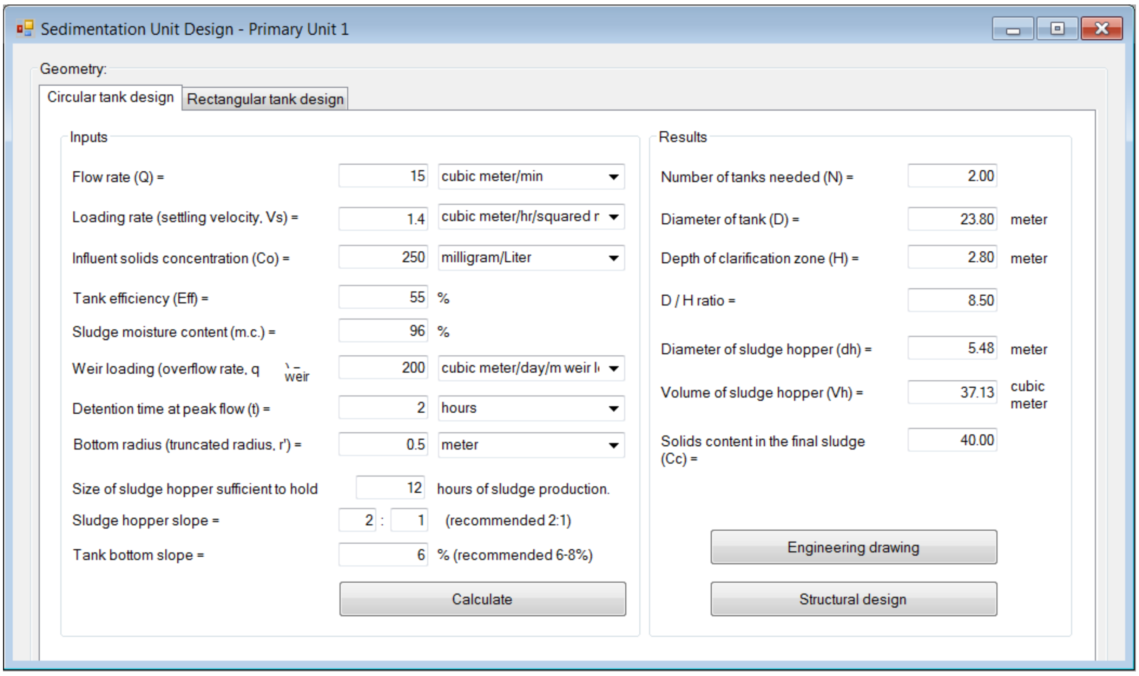


Figure 4.48: Final Model Window for a Sedimentation U-Plug

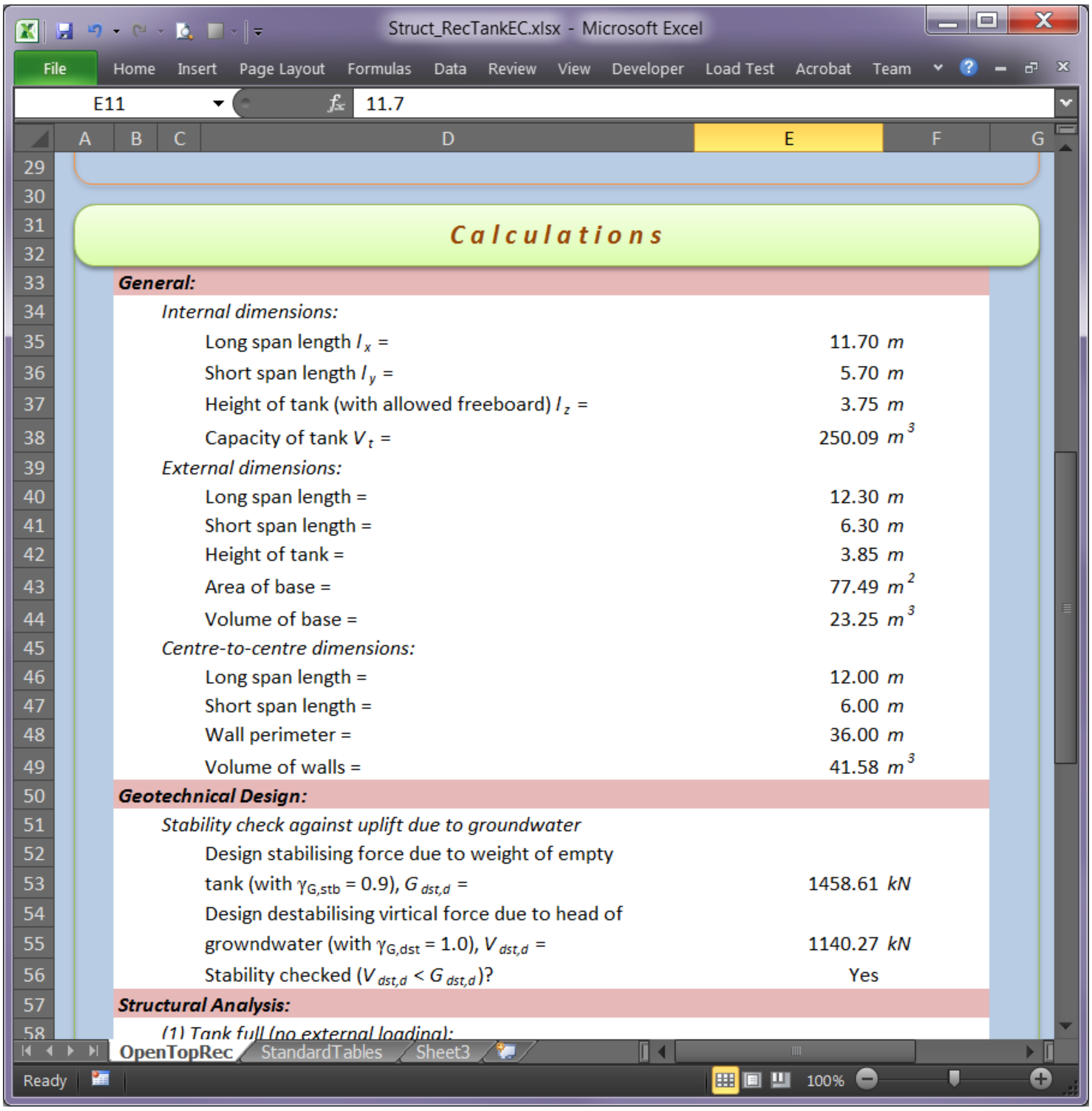


Figure 4.49: Output Calculation Sheet

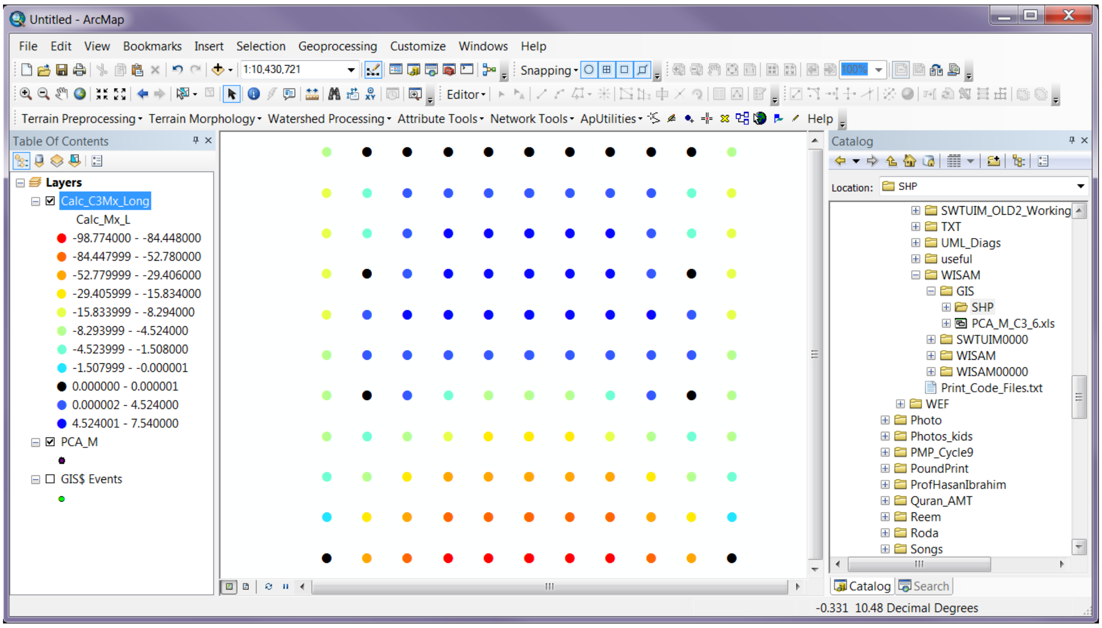


Figure 4.50: GIS Representation of a Calculated Moment

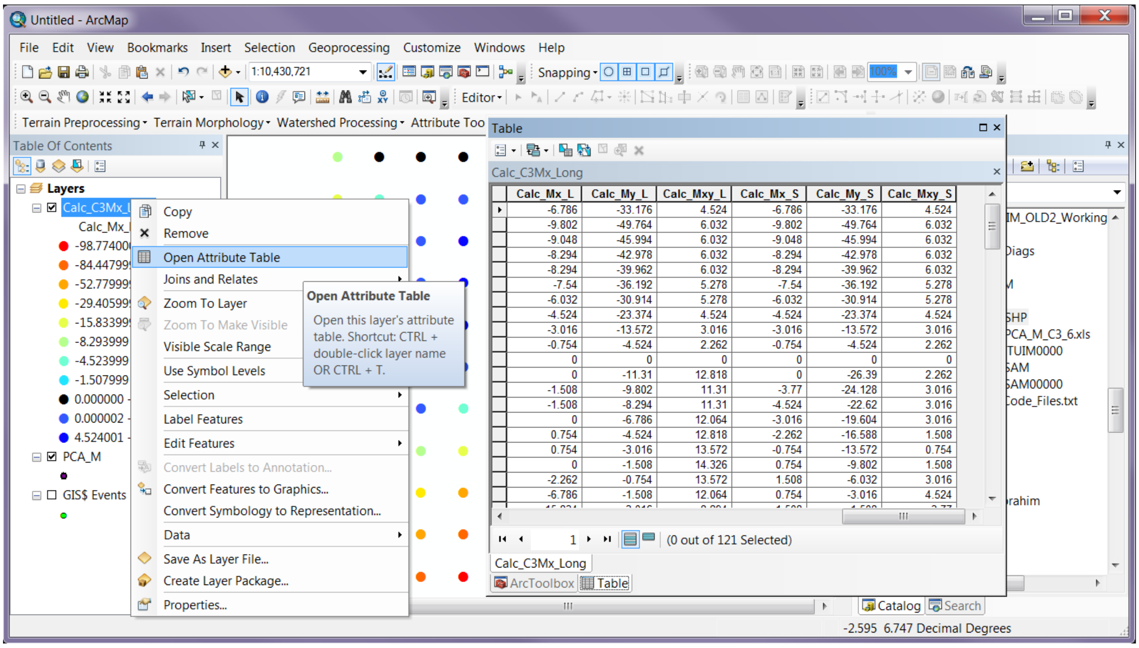


Figure 4.51: Calculated Moment Values in GIS Environment

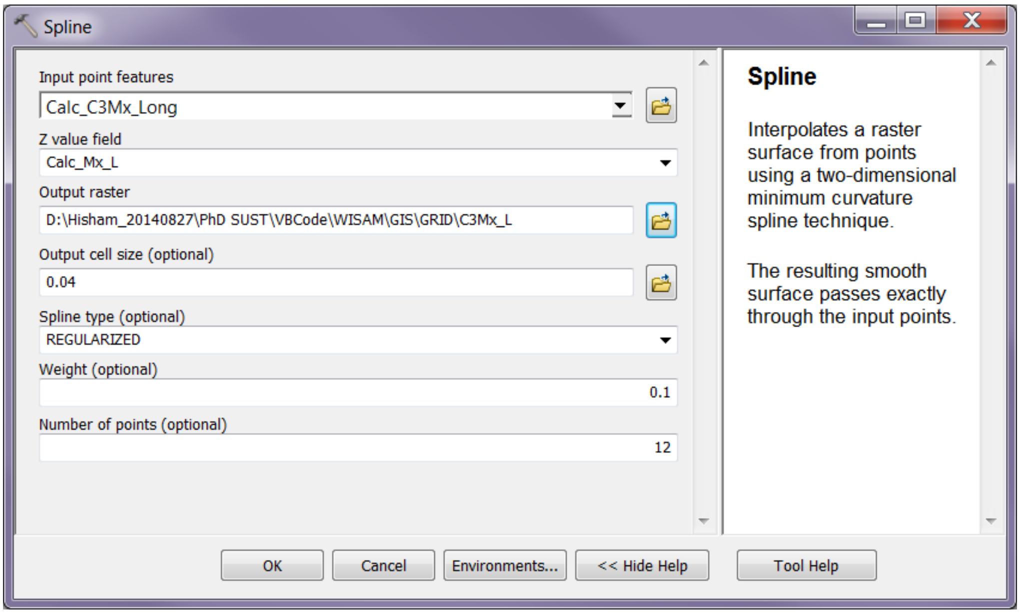


Figure 4.52: Using the Spline Tool for Spatial Interpolation of Moments

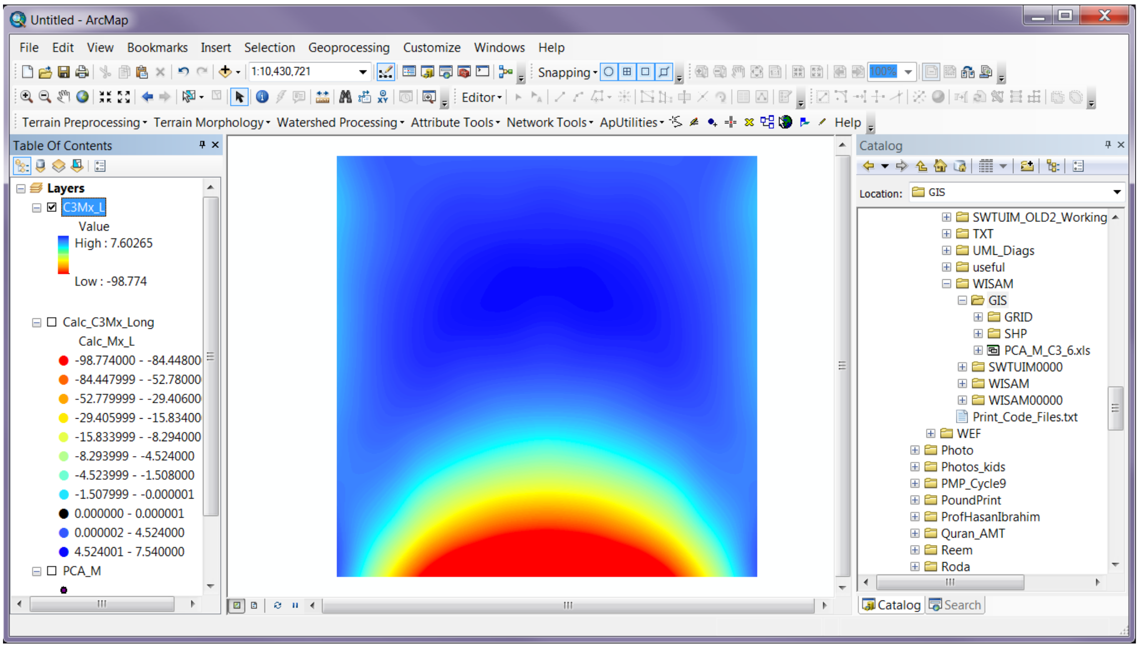


Figure 4.53: The Spatial Distribution of Calculated Moments

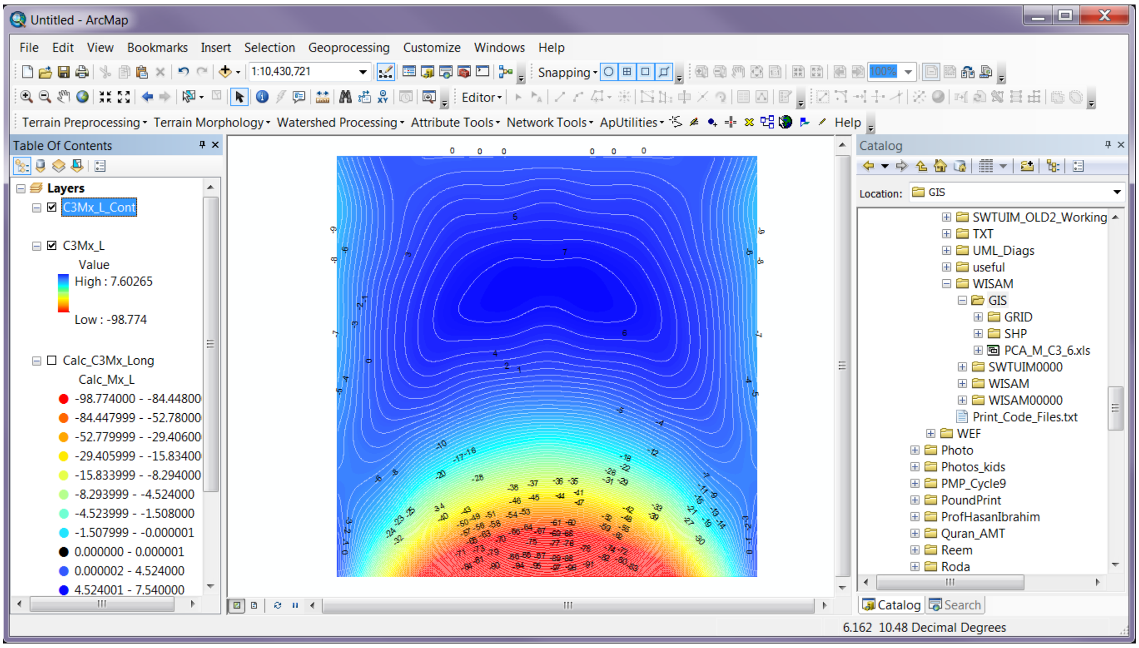


Figure 4.54: Contoured Representation of Calculated Moments

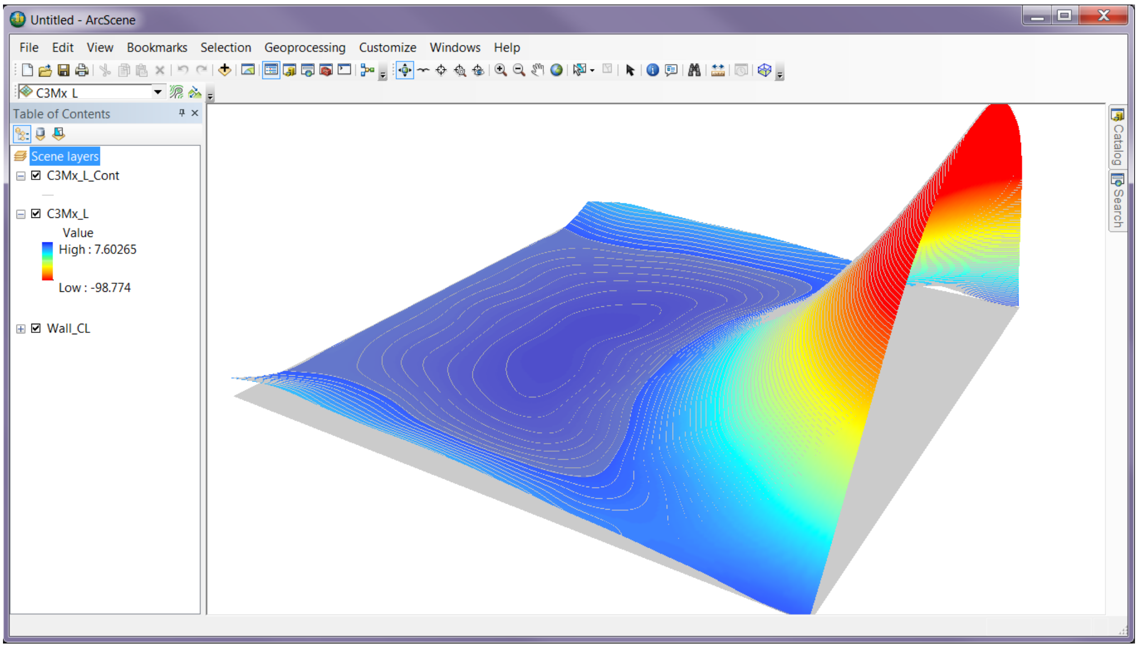


Figure 4.55: 3D Representation of Calculated Moments

Finally, WISAM's dissemination has been addressed by creating online webpages via hosted internet websites (accessed via: *http://wisam.sourceforge.net/* or *http://sourceforge.net/projects/wisam/*). Figure 4.56 show the project's main page, where the following could be achieved:

- Download installation package and software manual.
- Accessibility to updated and online knowledge content.
- Reviews feedback and discussion from relevant experts in the field.
- Receipt and response to comments from users.
- Upgrade and enhancement of the software in global collaborative context.

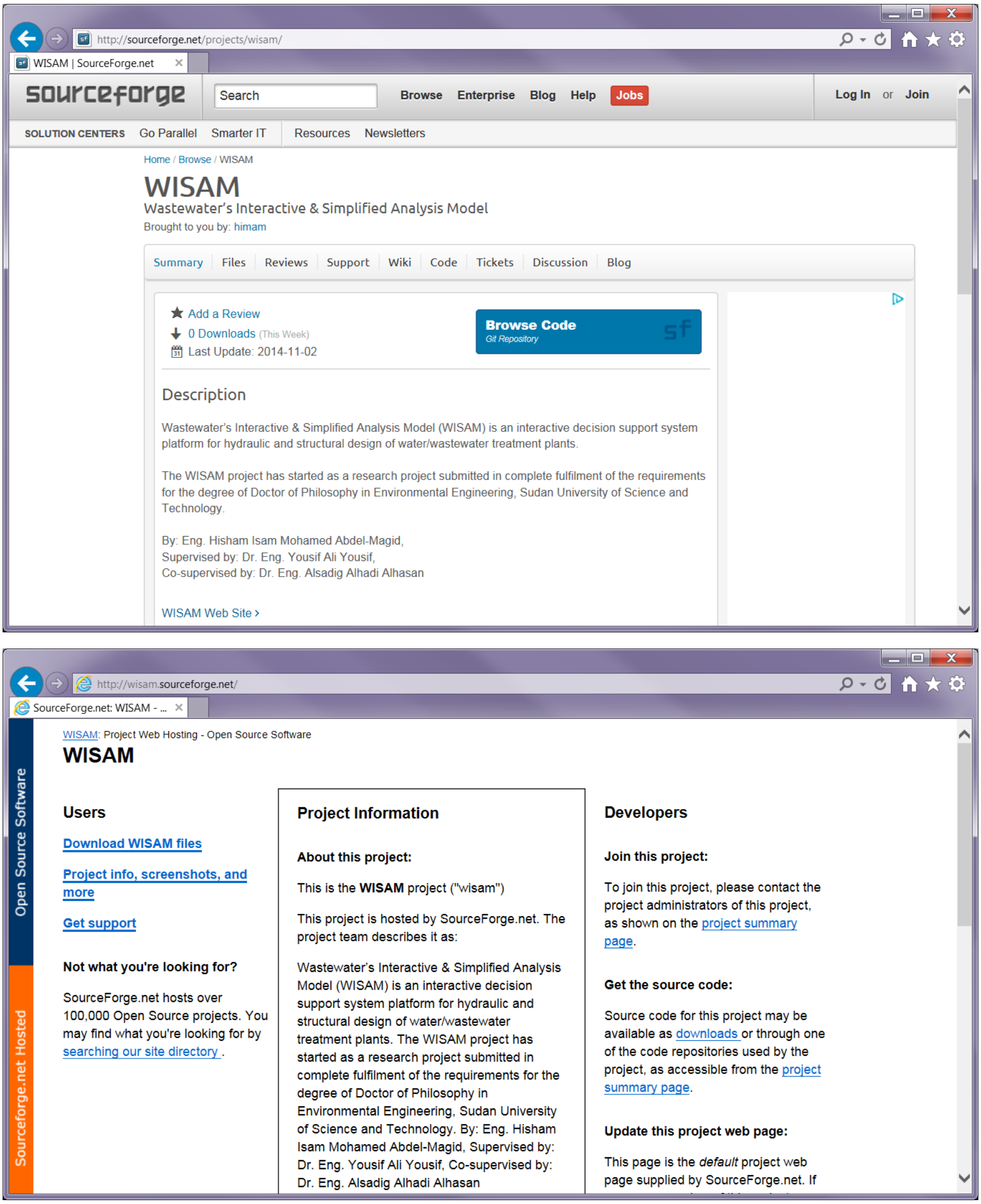


Figure 4.56: WISAM's Online Websites

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The major conclusions that emerged from this research work can be summarised as follows:

5.1.1. A computer-aided Decision Support System entitled *WISAM* has been modelled and developed using Microsoft Visual Basic .NET programming language, with thorough implementation of Object Oriented Programming (OOP) guides, as well as the conceptual aid of Unified Modelling Language (UML) paradigms.

5.1.2. A user-friendly interface has been implemented and developed to facilitate design tasks and decision support of water and wastewater treatment units.

5.1.3. Introduced interface provides interactive access to input, output and action screens. This facility enabled modifications to be done easily and results to be immediately updated during the design process.

5.1.4. To obtain complete WTP/WWTP design, the DSS platform has been engineered to enable swift addition of any desired mathematical models; such as models for the design of many preliminary, primary, secondary and tertiary water and wastewater treatment and sludge final disposal unit operations and processes.

5.1.5. Both hydraulic and structural design functionalities have been incorporated from-within the abstract level of computational routines, so as to facilitate rapid development and implementation.

5.1.6. The formulated programme structure has been formed in such a dynamic package that eases future inclusion, addition and updating of individual WTU/WWTU.

5.1.7. The programme classes have been stored separately from the Graphical User Interface (GUI) as Dynamic Link Libraries (DLLs). This is to emphasise any desired engine-exchange with other models, or complete change of GUI while maintaining the calculation algorithms.

5.1.8. The calculation output results could be visualized from within WISAM's environment, as well as exported to database files for any desired integration with compatible software (such as ESRI ArcGIS, MS Excel, and MS Access).

5.1.9. An online website has been launched as an integration with WISAM software for interactivity and communication with interested environmental engineers and research scientists in the field. This website platform can be accessed 24 hours a day, 7 days a week for the days of the year via URLs: *http://wisam.sourceforge.net* or *http://www.sourceforge.net/ projects/wisam*.

5.2 Recommendations

The main recommendations for future enhancement and update to the subject research area may be listed as hereafter:

5.2.1. Identifying suitable national and regional design parameters for local use (standardization is a must).

5.2.2. Inclusion of more design codes and multi-criteria analysis algorithms.

5.2.3. Increasing the number of modelled units and adding batch-processing.

5.2.4. Enhancing data interoperability with GIS geo-datasets and relevant software.

5.2.5. Hosting and dynamically maintaining WISAM's online site for collaborative knowledge development and sharing with stakeholders.

5.2.6. Adoption of WISAM by a national engineering institute firm for sustainable usage, development and implementation.

5.2.7. Increasing the model capabilities by gathering a multidisciplinary task force and group of design engineers (chemical, structural, mechanical, and hydraulic), architects, fine art specialists, computer programmers and software engineers, internet media.

5.2.8. Attempting to produce a competent computer package through use of Free and Open Source programming approaches.

5.2.9. Launching a platform-independent software version (to run under Windows, MacOS, Linux, Solaris, FreeBSD,...etc).

References

Abdel-Magid, I.M. (2014), Problems solving in environmental engineering, Dammam University Press, Dammam, KSA.

Abdel-Magid, I. M., Mohammed, A. H., and Rowe, D. R., (1997), Modelling methods for environmental engineers, CRC Press\Lewis Publishers Inc, 2000 Corporate Blvd. N. W., Boca Raton, FL, USA.

Abdel-Magid, I.M. and Abdel-Magid, T.I.M., (2014), Guidelines for Thesis Preparation, Unpublished document, personal communication.

American Society of Civil Engineers, American Water Works Association (ASCE), (2012), Water treatment plant design, McGraw-Hill Professional; 5 edi., New York.

Anagnostopoulos, K. P., Vavatsikos, A. P., Spiropoulos, N. and Kraias, I., (2010), Land suitability analysis for natural wastewater treatment systems using a new GIS add-in for supporting criterion weight elicitation methods, Operational Research International Journal, pp. 10:91–108.

Andrews, J.F., (1974), Dynamic models and control strategies for Wastewater treatment processes, review paper, Water Research Vol. 8. pp. 261 - 289. Pergamon Press.

Be´line, F., Boursier, H. Daumer, M.L., Guiziou, F. and Paul, E., (2007), Modelling of biological processes during aerobic treatment of piggery wastewater aiming at process optimization, Bioresource Technology Journal 98, pp. 3298–3308.

Benedetti, L., De Baets, B., Nopens, I., Vanrolleghem, P.A., (2010), Multi-criteria analysis of wastewater treatment plant design and control scenarios under uncertainty, Environmental Modelling & Software Journal 25, pp. 616–621.

Black, A.P., (2013), Object-oriented programming: Some history, and challenges for the next fifty years, Information and Computation Journal, vol. 231, pp. 3–20, http://www.sciencedirect.com.ezp.uod.edu.sa/science/article/pii/S0890540113000795

Briggs, R, (2006), Introduction to GIS, UT-Dallas, Accessed online via URL: http://www.utdallas.edu/ ~briggs/poec5319/arcview.ppt.

Bumble, S., (2000), Computer Simulated Plant Design for Waste Minimization/ Pollution Prevention, CRC Press LLC, USA.

Burger, R., Diehl, S. and Nopens, I., (2011), A consistent modelling methodology for secondary settling tanks in wastewater treatment, Water Research Journal, vol. 45, pp. 2247-2260.

Castro, P. M. Matos, H. A. and Novais, A. Q., (2007), An efficient heuristic procedure for the optimal design of wastewater treatment systems, Resources, Conservation and Recycling Journal, vol. 50, pp. 158–185.

Chang, S.Y., Liaw, S.L., (1985), Generating design for wastewater system. Journal of Environmental Engineering ASCE 111 (5), pp. 665– 679.

Chen, G.K., Fan, L.T., Erickson, L.E., (1972), Computer software for wastewater treatment plant design. Journal of the Water Pollution Control Federation, 44 (5), 746–762.

Chen, H., Yu, R., Ning, S. and Huang, H., (2010), Forecasting effluent quality of an industry wastewater treatment plant by evolutionary grey dynamic model, Resources, Conservation and Recycling Journal, 54, pp. 235–241.

Chen, J. and Chang, N., (2007), Mining the fuzzy control rules of aeration in a Submerged Biofilm Wastewater Treatment Process, Engineering Applications of Artificial Intelligence Journal, 20, pp. 959–969.

Comas, J; Alemany, J; Poch, M; Torrens, A; Salgot, M; Bou, J., (2003), Development of a knowledge-based decision support system for identifying adequate wastewater treatment for small communities, Water Science And Technology Journal, Vol. 48, Issue 11-12, pp. 393 – 400, http://ud.summon. serialssolutions.com

Davis, M. L. and Cornwell, D. A., (2006), Introduction to Environmental Engineering, McGraw-Hill Publishers, 4th edition, New York.

de Schutter, J. , L.,G., (2007), Development of A Decision Support System For Wastewater Reuse In The Middle East, Wastewater Reuse –Risk Assessment, Decision-Making and Environmental Security Journal, pp. 11–21.

Dellana S. A. and West D., (2009), Predictive modelling for wastewater applications: Linear and nonlinear approaches, Environmental Modelling & Software Journal, vol. 24, pp. 96–106.

Environment Canada, (1974), Workshop on Computer Aided Design and Simulation of Waste Treatment Systems. Report no. EPS 6-WP-74-1.

ESRI, (2012), ESRI ArcGIS 10.1 desktop software help, 380 New York Street, Redlands, CA 92373-8100, USA.

Evenson, E. J. & Baetz, B. W. (1994), Selection and sequencing of hazardous waste treatment processes: a knowledge-based systems approach. Waste Management Journal, vol. 14 (2), 161–165.

Fang, F., Ni, B., Li, W., Sheng, G. and Yu, H., (2011), A simulation-based integrated approach to optimize the biological nutrient removal process in a full-scale wastewater treatment plant, Chemical Engineering Journal 174, pp. 635– 643.

Fang, F., Ni, B.J., Xie, W.M., Sheng, G.P., Liu, S.G., Tong, Z.H. and Yu, H.Q., (2010), An integrated dynamic model for simulating a full-scale municipal wastewater treatment plant under fluctuating conditions, Chemical Engineering Journal 160, pp. 522–529.

Fernandez F.J., Seco A., Ferrer J., Rodrigo M.A., (2009), Use of neurofuzzy networks to improve wastewater flow-rate forecasting, Environmental Modelling & Software Journal, vol. 24, pp. 686–693.

Ferrer, J., Seco, A., Gabaldo´ n, C., Marzal, P., (1998), A steady-state model for the design of biological wastewater treatments. Journal of Environmental Engineering ASCE (submitted).

Ferrer, J., Seco, A., Serralta, J., Ribes, J., Manga, J., Asensi, E., J.J. Morenilla d, F. Llavador, (2008), DESASS: A software tool for designing, simulating and optimizing WWTPs, Environmental Modelling & Software Journal, 23, pp. 19-26.

Finney, B. A. & Gerheart, R. A. (2004), A User's Manual for WAWTTAR. Environmental Resources Engineering, Humboldt State University, Arcata, CA.

Gabaldo´ n., C., Ferrer, J.,Seco., A., Marzal, P., (1998), A software for the integrated design of wastewater treatment plants, Elsevier Science, Environmental Modelling & Software Journal, vol. 13, pp. 31–44.

Gallego, A., Hospido, A., Moreira, M. T. and Feijoo, G., (2008), Environmental performance of wastewater treatment plants for small populations, Resources, Conservation and Recycling Journal, vol. 52, pp. 931–940.

Gerardi, M.H., (2002), Settleability Problems and Loss of Solids in the Activated Sludge Process, Wastewater Microbiology Journal.

Gernaey, K.V., van Loosdrecht, M. C.M., Henze, M., Lind M. and Jørgensen, S.B., (2004) Review Activated sludge wastewater treatment plant modelling and simulation: state of the art, Environmental Modelling & Software Journal, vol. 19, pp. 763–783.

Getty, D.B., Koussis, J.M., Parker, F.L., (1987), CAD comparisons for wastewater treatment facilities. Environmental Technology Letters 8, pp. 405–418.

Guyer, J.P., (2011), Introduction to Preliminary Wastewater Treatment, Continuing Education and Development Inc., USA.

Hakanen, J; Miettinen, K; Sahlstedt, K, (2011), Wastewater treatment: New insight provided, Interactive-multi-objective optimization by Elsevier Science, Amsterdam, V. 51, pp. 328 -337.

Halvorson, M., (2010), Microsoft Visual Basic 2010: Step by Step, Microsoft Press, Washington.

Hamed, M. M., Khalafallah, M. G., and Hassanien, E.A., (2004), Prediction of wastewater treatment plant performance using artificial neural networks, Environmental Modelling & Software Journal, vol. 19, pp. 919–928.

Hammer, M. J., (2011), Water and Wastewater Technology, Prentice Hall publishers; Englewood Cliffs, New Jersey.

Hamouda, M. A., Anderson, W. B. and Huck, P. M., (2009), Decision support systems in water and wastewater treatment process selection and design: a review, Water Science & Technology (WST) Journal, pp. 1757-1770.

Heller, M., Garlapati, S. & Aithala, K. 1998 Expert membrane system design and selection for metal finishing waste water treatment. Expert Systems with Applications Journal, vol. 14 (3), 341–353.

Hensgen, P. and UmbrelloUML Modeller Authors (2003), Umbrello UML Modeller Handbook, KDE Documentation, published online via: docs.kde.org/stable/en/ kdesdk/umbrello/index.html.

Hidalgo, D., Irusta, R.,Martinez, l., Fatta, D. and Papadopoulos, A. (2007), Development of a multi-function software decision support tool for the promotion of the safe reuse of treated urban wastewater, Desalination Journal, 215, pp. 90–103.

Huisman, L., (1977), Sedimentation and Flotation: Sedimentation and Flotation, - Mechanical Filtration, - Slow Sand Filtration, - Rapid Sand Filtration, Delft University of Technology, Herdruk.

Iacopozzi, I., Innocenti,V., Marsili-Libelli, S. and Giusti, E., (2007), A modified Activated Sludge Model No. 3 (ASM3) with two-step nitrification-denitrification, Environmental Modelling & Software 22, pp. 847-861.

Inês S.F. Freitasa, I.S.F., Costa, C. A.V., and Boaventurab, R. A.R., (2000), Conceptual design of industrial wastewater treatment processes: primary treatment, Computers & Chemical Engineering Journal, Vol. 24, Issues 2–7, pp. 1725–1730, http://www.sciencedirect.com.ezp.uod.edu.sa/science/article/pii/S0098135400004506

Joksimovic, D., Kubik, J., Hlavinek, P., Savic, D. & Walters, G. (2006), Development of an integrated simulation model for treatment and distribution of reclaimed water. Desalination Journal, 188 (1–3), 9–20.

Kao, J.J., Brill, E.D., Pfeffer, J.T., Geselbracht, J.J., (1993), Computer- based environment for wastewater treatment plant design. Journal of Environmental Engineering, ASCE 119 (5), 931–945.

Karia, G.L., and Christian, R.A., (2006), Wastewater Treatment: Concepts and Design Approach, Prentice Hall of India Private Limited, India.

Krovvidy, S. & Wee, W. G. (1993) Wastewater treatment systems from case-based reasoning. Machine Learning Journal, 10 (3), 341–363.

Krovvidy, S., Wee, W. G., Suidan, M., Summers, R. S., Coleman, J. J. & Rossman, L. (1994) Intelligent sequence planning for wastewater treatment systems. IEEE Expert: Intelligent Systems and Their Application Journal, vol. 9 (6), 15–20.

Krovvidy, S., Wee, W. G., Summers, R. S. & Coleman, J. J. (1991), An AI approach for wastewater treatment systems. Journal of Applied Intelligence, vol. 1 (3), 247–261.

L´opez, P. R. Lav´ın, A. G. L´opez, M. M. M. de las Heras, J. L. B., (2008), Flow models for rectangular sedimentation tanks, Chemical Engineering and Processing 47, pp. 1705–1716

Lee, C.C. Editor in chief, and Lin, S.D., (2000), Handbook of Environmental Engineering Calculations, McGraw-Hill Inc., USA.

Loetscher, T. & Keller, J. (2002), A decision support system for selecting sanitation systems in developing countries. Socio- Econ. Plann. Sci. 36 (4), 267–290.

Mara, D., (2004), Domestic Wastewater Treatment in Developing Countries, Routledge.

McBean, E. A. & Zhu, Z. J. Y. (2007). Selection of water treatment processes using Bayesian decision network analyses. J. Environ. Eng. Sci. 6(1), 95–102.

McCabe, W., Smith, J., and Harriott, P., (2004), Unit Operations of Chemical Engineering, McGraw Hill Chemical Engineering Series.

Metcalf and Eddy Inc., (2013), revised by George Tchobanoglous, G., Burton, F.L., and Stensel, H.D., Wastewater Engineering: Treatment, Disposal and Reuse, McGraw Hill Higher Education; 5th Revised edi., New York.

Mitchell, J. C. (2003), Concepts in programming languages, Cambridge University Press, 2003, p.278.

Munshi, J.A. (1998), Rectangular Concrete Tanks, 5th ed., Portland Cement Association, Illinois.

Muschalla, D. (2008), Optimization of integrated urban wastewater systems using multi-objective evolution strategies. Urban Water Journal 5 (1), 59–67.

Nathanson, J.A., (2007), Basic Environmental Technology: Water Supply, Waste Management & Pollution Control, Prentice Hall, Englewood Cliffs, New Jersy.

Nemerow, N. L., Agardy, F.J. and Salvato, J. A., (2009), Environmental Engineering, Wiley.

NIIT, (2001), Programming Logic and Techniques: Student Guide, NIIT press, India.

Object Management Group (OMG), (2011), OMG Unified Modelling Language™ (OMG UML) Superstructure version 2.4.1, Object Management Group, published online via: http://www.omg.org/spec/UML/2.4.1/Superstructure.

Pierce, B. C., (2004), Advanced Topics in Types and Programming Languages, The MIT Press; 1 edi.

Pilone, N. Pitman, (2005), UML 2.0 in a Nutshell, O'Reilly Publishers, ISBN: 0-596-00795-7.

Poch, M., Comas, J., Rodríguez-Roda, I., Sànchez-Marrè, M. &Cortès, U. (2004), Designing and building real environmental decision support systems. Environmental Modelling and Software 19 (9), 857–873.

Pons, M.N., Silva, M.C.L., Potier, O. and Arnos, E., Philippe Battaglia, (2008), Modelling of an hybrid wastewater treatment plant, 18th European Symposium on Computer Aided Process Engineering – ESCAPE 18 Bertrand Braunschweig and Xavier Joulia (Editors), Elsevier B.V.

Prat, P., Benedetti, L.,Corominas, L., Comas, J. and Poch, M., (2012), Model-based knowledge acquisition in environmental decision support system for wastewater integrated management, Water Science And Technology: A Journal Of The International Association On Water Pollution Research, Vol. 65 (6), pp. 1123-9.

Puig, S., van Loosdrecht, M.C.M. J. and Meijer, Colprimb, S.C.F., (2008), Data evaluation of full-scale wastewater treatment plants by mass balance, water research 42, pp. 4645 – 4655, journal homepage: www.elsevier.com/locate/ waters.

Reynolds, C.E, Steedman, J.C, and Threlfall, A.J., (2008), Reynolds's Reinforced Concrete Designer's Handbook, 11th Ed., Taylor and Francis, New York.

Rivas, A. Irizar, I. And Ayesa, E., (2008), Model-based optimization of Wastewater Treatment Plants design, Environmental Modelling & Software 23, pp. 435-450

Roda, I.R. Poch, M. and Banares-Alcantara, R., (2000), Application of a support system to the design of wastewater treatment plants, Artificial Intelligence in Engineering 14, pp. 45–61

Rodriguez-Roda, I., Poch, M. & Ban˜ ares-Alca´ ntara, R. (2000), Conceptual design of wastewater treatment plants using a design support system. J. Chem. Technol. Biotechnol. 75 (1), 73–81.

Rowe, D.R. and Abdel-Magid, I.M. (1995), Handbook of Wastewater Reclamation and Reuse, Lewis Pub., Chelsea.

Sairan, F. M., Ujang, Z., Salim, M. R. & Din, M. F. M. (2004), Architecture of decision support system for WASDA: the module for sequencing batch reactor. In ASIAWATER 2004 International Conference on Water and Wastewater: Technology and Management in Asia, The Mines, Kuala Lumpur.

Sala-Garridoa, R., Molinos-Senante, M. and Hernandez-Sancho, F., (2011), Comparing the efficiency of wastewater treatment technologies through a DEA metafrontier model, Chemical Engineering Journal 173, pp. 766– 772

Scott, M. L., (2006), Programming language pragmatics, Edition 2, Morgan Kaufmann, p. 470.

Shahriari, H. Warith, M., Hamoda, M., and Kennedy, K. J., (2012), Anaerobic digestion of organic fraction of municipal solid waste combining two pretreatment modalities, high temperature microwave and hydrogen peroxide, Waste Management 32, pp. 41–52.

Sin, G. Gernaey, K. V., Neumann, M. B., van Loosdrecht, M. C.M. and Gujer, W., (2011), Global sensitivity analysis in wastewater treatment plant model applications: Prioritizing sources of uncertainty, Water Research 45, pp. 639-651.

Smith, R., Eilers, R.G., (1968), Executive digital computer program for preliminary design of wastewater treatment systems. Water Pollution Series WP-20-14, NTIS-PB 222765.

Spinos, M., Marinos-Kouris, D., (1992), Integrated computer aided pro- cess design of wastewater treament plants on a PC system. Water Science Technology 25 (1), 107–112.

Stefanakis, A. I. And Tsihrintzis, V. A. (2012), Effect of various design and operation parameters on performance of pilot-scale Sludge Drying Reed Beds, Ecological Engineering 38, pp. 65– 78.

Threlfall, T., 2013, Worked Examples for the Design of Concrete Structures to Eurocode2, CRC Press, New York.

Uggetti, E., Ferrer, I., Molist, J., and Garcı´a, J., (2011) , Technical, economic and environmental assessment of sludge treatment wetlands, water research 45, pp. 573 – 582.

Ullmer, C., Kunde, N., Lassahn, A., Gruhn, G. & Schulz, K. (2005), WADOY water design optimization—methodology and software for the synthesis of process water systems. J. Cleaner Prod. 13 (5), 485–494.

Viessman, W., Hammer, M. J., Perez, E.M., and Chadik, P.A., (2008), Water Supply and Pollution Control, Prentice Hall.

Water Environment Federation, American Society of Civil Engineers, (1992), Design of Municipal Wastewater Treatment Plants, vol. I. & 2 Book Press, Inc., Vermont.

WEF (Water Environment Federation), (2008a), Industrial wastewater management, treatment, and disposal, manual of practice No.FD-3 3rd Ed., WEF Press, USA.

WEF (Water Environment Federation), (2008b), Operation of municipal wastewater ttreatment plants, manual of practice No.11 6th Ed., WEF Press, USA.

Wikipedia (2013), Unified Modelling Language, URL: en.wikipedia.org/wiki/ Unified_Modelling_Language. http://en.wikipedia.org/wiki/Object-oriented_ programming. http://en.wikipedia.org/wiki/Visual_Basic_.NET

Wukovits, W., Harasek, M. & Friedl, A. (2003), A knowledge based system to support the process selection during waste water treatment. Resour. Conserv. Recycling 37 (3), 205–215.

Yang, C.-T. & Kao, J.-J. (1996), An expert system for selecting and sequencing wastewater treatment processes. Water Sci. Technol. 34 (3), 347–353.

Zarghami, M. and Akbariyeh, S., (2012), System dynamics modelling for complex urban water systems: Application to the city of Tabriz, Iran, Resources, Conservation and Recycling 60, pp. 99– 106.

Zeng, G. M., Jiang, R., Huang, G. H., Xu, M. & Li, J. B. (2007), Optimization of wastewater treatment alternative selection by hierarchy grey relational analysis. J. Environ. Manage. 82 (2), 250–259.

APPENDIX

The Formulated Software (WISAM©) VB Source Code

(The following pages contain the VB.NET code listing of WISAM)

```vb
' ==============================================================================
' FILE NAME: clsGeneric.vb
' ==============================================================================


Public Class clsGeneric
    Inherits HIM.Generic.uPlugGeneralData
    Implements HIM.Generic.IGeneric

    Public Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As HIM.Generic.itemType Implements HIM.Generic.IGeneric.SetItemType
        Dim tmpItem As HIM.Generic.itemType = New HIM.Generic.itemType
        tmpItem.abbreviation = a
        tmpItem.description = d
        tmpItem.value = v
        tmpItem.unit = u
        Return tmpItem
    End Function

    Public Function SetParentCatType(ByVal i As UInteger, ByVal c As String) As HIM.Generic.catType Implements HIM.Generic.IGeneric.SetParentCatType
        Dim tmpCat As HIM.Generic.catType = New HIM.Generic.catType
        tmpCat.tuParentCatID = i
        tmpCat.tuParentCatCaption = c
        Return tmpCat
    End Function
End Class
```

```vb
' ============================================================================
' FILE NAME: clsSepticTankRec.vb
' ============================================================================

Public Class clsSepticTankRec
    Inherits clsGeneric
    Implements HIM.Generic.IuPlugStructure


    Public Function GetInputs(ByVal type_of_WWTUnit As String) As String Implements HIM.Generic.IuPlugStructure.GetInputs

        Return Nothing
    End Function

    Public Function LoadDefaults(ByVal i As Integer) As Object Implements HIM.Generic.IuPlugStructure.LoadDefaults
        Dim input1 As HIM.Generic.itemType = New HIM.Generic.itemType
        Dim input2 As HIM.Generic.itemType = New HIM.Generic.itemType

        Me.uPlugUnitType = "SepticTank"
        Me.uPlugUnitID = "SepticTank" & i
        Me.inputList.Clear()
        input1 = SetItemType("I1", "First Input", 100, "m/s")
        Me.inputList.Add(input1)
        input2 = SetItemType("I2", "Second Input", 2, "m/s")
        Me.inputList.Add(input2)
        Return Nothing
    End Function

    Public Function LoadFromFile(ByVal name_of_file As String) As String Implements HIM.Generic.IuPlugStructure.LoadFromFile
        ' Read data from existing file.
        Return Nothing
    End Function

    Public Function ProcessHydroCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessHydroCalc
        ' Perform WWTUnit's processes and calculations.
        Dim outputItem As HIM.Generic.itemType = New HIM.Generic.itemType
        Dim outputList As ArrayList = New ArrayList

        outputList.Clear()

        outputItem.abbreviation = "V"
        outputItem.description = "Tank volume"
        outputItem.unit = "m"
        outputItem.value = inputsList(0).value + inputsList(1).value
        outputList.Add(outputItem)


        Return outputList
    End Function

    Public Function SaveToFile(ByVal name_of_file As String) As String Implements
```

```
    HIM.Generic.IuPlugStructure.SaveToFile
        ' Print data to specific file.
        Return Nothing
    End Function

    'Public Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As HIM.Generic.itemType Implements HIM.Generic.IuPlugStructure.SetItemType
    '    Dim tmpItem As HIM.Generic.itemType = New HIM.Generic.itemType
    '    tmpItem.abbreviation = a
    '    tmpItem.description = d
    '    tmpItem.value = v
    '    tmpItem.unit = u
    '    Return tmpItem
    'End Function

    Public Function ProcessStrucCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessStrucCalc
        Return Nothing
    End Function
End Class
```

```
' ===============================================================================
' FILE NAME: clsWWTUnit_ProtoType.vb
' ===============================================================================

''Imports WISAM.HIM.Generic

Public Class clsWWTUnit_ProtoType
    Inherits clsGeneric
    Implements HIM.Generic.IuPlugStructure

    Public Function GetInputs(ByVal type_of_WWTUnit As String) As String Implements HIM.Generic.IuPlugStructure.GetInputs

        Return Nothing
    End Function

    Public Function ChangeInputs(ByVal inp As String()) As Boolean
        Dim i As Integer
        Dim tmp As HIM.Generic.itemType
        For i = 0 To inp.Length - 1 Step 4
            tmp.abbreviation = inp(i)
            tmp.description = inp(i + 1)
            tmp.value = inp(i + 2)
            tmp.unit = inp(i + 3)
            inputList(i / 4) = tmp
        Next
        Return True
    End Function

    Public Function ReturnInputs() As String()
        'Return inputList.ToArray(GetType(HIM.Generic.itemType))
        Dim i As Int16
        Dim j(inputList.Count * 4 - 1) As String
        Dim tmp As HIM.Generic.itemType
        For i = 0 To inputList.Count - 1
            tmp = inputList(i)
            j(i * 4) = tmp.abbreviation
            j((i * 4) + 1) = tmp.description
            j((i * 4) + 2) = tmp.value
            j((i * 4) + 3) = tmp.unit
        Next
        Return j
    End Function

    Public Function LoadDefaults(ByVal i As Integer) As Object Implements HIM.Generic.IuPlugStructure.LoadDefaults
        Dim input1 As HIM.Generic.itemType = New HIM.Generic.itemType
        Dim input2 As HIM.Generic.itemType = New HIM.Generic.itemType

        Me.uPlugUnitType = "uPlugUnitType1"
        Me.uPlugUnitID = "uPlugUnitType1" & i
        Me.inputList.Clear()
        input1 = SetItemType("I1", "First Input", 100, "m/s")
        Me.inputList.Add(input1)
```

```
        input2 = SetItemType("I2", "Second Input", 2, "m/s")
        Me.inputList.Add(input2)
        Return Nothing
    End Function

    Public Function LoadFromFile(ByVal name_of_file As String) As String Implements HIM.Generic.IuPlugStructure.LoadFromFile
        ' Read data from existing file.
        Return Nothing
    End Function

    Public Function ProcessHydroCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessHydroCalc
        ' Perform WWTUnit's processes and calculations.
        Dim output1 As HIM.Generic.itemType = New HIM.Generic.itemType
        Dim output2 As HIM.Generic.itemType = New HIM.Generic.itemType
        Dim outputList As ArrayList = New ArrayList

        outputList.Clear()

        output1.abbreviation = "F1"
        output1.description = "First Output"
        output1.unit = "m"
        output1.value = inputsList(0).value + inputsList(1).value
        outputList.Add(output1)

        output2.abbreviation = "F2"
        output2.description = "Second Output"
        output2.unit = "m"
        output2.value = inputsList(0).value - inputsList(1).value
        outputList.Add(output2)

        Return outputList
    End Function

    Public Function SaveToFile(ByVal name_of_file As String) As String Implements HIM.Generic.IuPlugStructure.SaveToFile
        ' Print data to specific file.
        Return Nothing
    End Function

    'Public Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As HIM.Generic.itemType Implements HIM.Generic.IuPlugStructure.SetItemType
    '    Dim tmpItem As HIM.Generic.itemType = New HIM.Generic.itemType
    '    tmpItem.abbreviation = a
    '    tmpItem.description = d
    '    tmpItem.value = v
    '    tmpItem.unit = u
    '    Return tmpItem
    'End Function

    Public Function ProcessStrucCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessStrucCalc
```

```
        Return Nothing
    End Function
End Class

```

```vb
Public Class extButton
    Inherits Windows.Forms.Button

    Public myTypeOfUnit As Integer = -1

    Public Sub New(ByVal typeOfUnit As Integer)
        myTypeOfUnit = typeOfUnit
    End Sub

    Private Sub extButton_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Click
        'My.Forms.mdiMainWindow.
        m_frmDesign.setNewItemType = myTypeOfUnit
        'My.Forms.mdiMainWindow.
        m_frmDesign.setIsAddingNewItem = True
        'My.Forms.mdiMainWindow.
        m_frmDesign.setIsAddingNewPath = False
        My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Select where to add the new '" + _
            unitNames(myTypeOfUnit) + "' unit.."

    End Sub

    Public Sub setParent()
        Me.Parent = m_frmToolBox.FlowLayoutPanel1 'My.Forms.mdiMainWindow.
    End Sub
End Class

```

```
'===============================================================================
' FILE NAME: defExtPictureBox.vb
'===============================================================================

'Module defExtPictureBox
Imports System.Resources

Namespace HIM.Generic
    'Property designItems As ArrayList
    Structure INIT_IMAGE
        Const W As Integer = 30
        Const H As Integer = 30
    End Structure

    Structure designPath
        Dim startItemTag As String
        Dim endItemTag As String
    End Structure

    Structure designItem
        Dim X, Y As Integer
        Dim Type As Char
        Dim itemNo As Integer
    End Structure

    Public Class extPictureBox
        Inherits Windows.Forms.PictureBox

        Private isMoving As Boolean = False
        Private lastMousePos As Point
        Public itemType As Integer
        Public isLinked As Boolean = False
        Public linkedPathNo As Int16
        Public linkedAs As String
        Public myNumber As Int16


        Public Sub New()
            Me.Cursor = Cursors.SizeAll
            Me.SizeMode = PictureBoxSizeMode.StretchImage
            Me.Width = INIT_IMAGE.W
            Me.Height = INIT_IMAGE.H

        End Sub

        Private Sub extPictureBox_MouseClick(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseClick
            m_frmDesign.setNewItemType = itemType

            If m_frmDesign.setIsAddingNewPath Then
                '*** Checks to see if the first point is empty.. If yes, it means this
                '*** pictureBox will be the start point of the path..
                If (m_frmDesign.pathStart.X < 0 Or
```

```
                    m_frmDesign.pathStart.Y < 0) Then
                    m_frmDesign.pathStart =
                        New Point(Me.Location.X + (Me.Width / 2), Me.Location.Y + (Me.Height / 2))
                    My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Select path end point.."
                    isLinked = True
                    linkedPathNo = m_frmDesign.lastPath
                    linkedAs = "Start"
                Else
                    m_frmDesign.pathEnd =
                        New Point(Me.Location.X + (Me.Width / 2), Me.Location.Y + (Me.Height / 2))
                    m_frmDesign.setIsAddingNewPath = False
                    m_frmDesign.pathStart = New Point(-1, -1)
                    My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Ready"
                    isLinked = True
                    linkedPathNo = m_frmDesign.lastPath
                    linkedAs = "End"
                    m_frmDesign.createNewPathLine()
                End If
            End If

            With m_frmProperties.DataGridView1
                'asmMembers = asmType.GetMember("inputList")
                asmMethod = asmType(itemType).GetMethod("ReturnInputs")
                '''''Dim inp As itemType() = Convert.ChangeType(asmMethod.Invoke(asmObj(myNumber), Nothing), GetType(itemType))
                '''''Dim ip As itemType() = asmMethod.Invoke(asmObj(myNumber), Nothing)
                Dim inp As String() = asmMethod.Invoke(asmObj(myNumber), Nothing)

                Dim j As Int16
                .Rows.Clear()
                For j = 0 To inp.Length - 2 Step 4
                    .Rows.Add(inp(j), inp(j + 2))
                Next

                Exit Sub

                If .Rows.Count < 3 Then
                    .Rows.Clear()
                    .Rows.Add("X", Me.Left)
                    .Rows.Add("Y", Me.Top)
                    .Rows.Add("Type", Me.itemType)
                Else
                    Dim i As Integer
                    For i = 0 To .Rows.Count - 1
                        Select Case .Rows(i).Cells("PropertyCol").Value
                            Case "X" : .Rows(i).Cells("ValueCol").Value = Me.Left
                            Case "Y" : .Rows(i).Cells("ValueCol").Value = Me.Top
                            Case "Type" : .Rows(i).Cells("ValueCol").Value = Me.itemType
                        End Select
```

```
                    Next
                End If
            End With
        End Sub

        Private Sub extPictureBox_MouseDoubleClick(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseDoubleClick
            m_frmDesign.PropertiesToolStripMenuItem1.PerformClick()
        End Sub

        Private Sub extPictureBox_MouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseDown
            If e.Button <> Windows.Forms.MouseButtons.Left Then Exit Sub
            isMoving = True
            lastMousePos.X = e.X
            lastMousePos.Y = e.Y
        End Sub

        Private Sub extPictureBox_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseMove
            If Not isMoving Then Exit Sub
            Dim i, j As Double
            i = Me.Location.X + (e.X - lastMousePos.X)
            j = Me.Location.Y + (e.Y - lastMousePos.Y)
            Me.Location = New Point(i, j)
            If isLinked Then
                If linkedAs = "Start" Then
                    m_frmDesign.designPaths(linkedPathNo).StartPoint = _
                        New Point(Me.Location.X + (Me.Width / 2), Me.Location.Y + (Me.Width / 2))
                Else
                    m_frmDesign.designPaths(linkedPathNo).EndPoint = _
                        New Point(Me.Location.X + (Me.Width / 2), Me.Location.Y + (Me.Width / 2))
                End If
            End If
        End Sub

        Private Sub extPictureBox_MouseUp(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Me.MouseUp
            isMoving = False

            With m_frmDesign
                If .lastDesignItem < 0 Then
                    .lastDesignItem = myNumber
                    .activeDesignItem = myNumber
                Else
                    .lastDesignItem = .activeDesignItem
                    .activeDesignItem = myNumber
                End If
            End With

            drawMyBorder()
        End Sub

        Private Sub drawMyBorder()
```

```
            Dim n As extPictureBox
            For Each p As Control In m_frmDesign.PictureBox2.Controls
                If p.GetType().Name = "extPictureBox" Then
                    n = Convert.ChangeType(p, GetType(extPictureBox))
                    n.BorderStyle = Windows.Forms.BorderStyle.None
                End If
            Next
            Me.BorderStyle = Windows.Forms.BorderStyle.FixedSingle

        End Sub
    End Class

End Namespace
'End Module

```

```
' ================================================================================
' FILE NAME: defGeneric.vb
' ================================================================================

Imports System.Collections          '*** for the ArrayList struct
Imports System.Collections.Generic  '*** for the List structure
'****** uncomment the following line if you will define any thing that is 'Image' type
'Imports System.Drawing.Image

'****** I changed the following 'Image' definitions into 'String' definitions
'Public uPlugTypeImg As Image
'Public uPlugHydroDispImg As Image     'Display image at the U-Plug inputs/outputs configuration form
'Public uPlugStrucDispImg As Image     'Display image at the U-Plug inputs/outputs configuration form

Namespace HIM.Generic
    Public Interface IuPlugStructure 'was IwwtUnitStructure
        '****** The compiler refused the following function definition without a return type,
        '****** so i defined the return as 'Object'..
        Function LoadDefaults(ByVal i As Integer) As Object
        Function LoadFromFile(ByVal name_of_file As String) As String
        Function SaveToFile(ByVal name_of_file As String) As String
        Function ProcessHydroCalc(ByVal inputsList As ArrayList) As ArrayList
        Function ProcessStrucCalc(ByVal inputsList As ArrayList) As ArrayList
        Function GetInputs(ByVal type_of_WWTUnit As String) As String
        'Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As itemType
    End Interface

    Public Class uPlugGeneralData 'was wwtGeneralData
        ''Modifications Comments:
        '
        'uPlugUnitType >>> uPlugUnitType
        'isLinked >>> isLinkedIn + isLinkedOut
        'inputLink >>> lstLinkedIn
        'outputLink >>> lstLinkedOut
        'uPlugUnitID >>> uPlugUnitID

        Public uPlugParentCat As catType

        Public uPlugTypeImg As String
        Public uPlugTypeCaption As String

        Public uPlugUnitID As UInteger = 0  'This would serve as a Unique Primary Key
        Public uPlugUnitType As String      'The name of uPlug Type in general
        Public uPlugUnitCaption As String   'Specific name for this unit
        Public isLinkedIn As Boolean = False
        Public isLinkedOut As Boolean = False
        Public lstLinkedIn As List(Of UInteger)     'List of linkedIn uPlugUnitID's
```

```
        Public lstLinkedOut As List(Of UInteger)    'List of linkedOut uPlugUnitID's

        ''' <summary>
        ''' List of all inputs (hydraulic + structural)
        ''' </summary>
        Public inputList As ArrayList = New ArrayList ' Type of members is itemType
        Public outputList As ArrayList = New ArrayList ' Type of members is itemType

        ''' <summary>
        ''' Indicator of Hydraulic Calculations Algorithm
        ''' </summary>
        Public hasHydroCalcs As Boolean = True
        Public uPlugHydroDispImg As String    'Display image at the U-Plug inputs/outputs configuration form
        Public uPlugHydroSetOfImg As String   'Path to a PDF or DWG file with standard design drawings

        ''' <summary>
        ''' Indicator of Structural Calculations Algorithm
        ''' </summary>
        Public hasStrucCalcs As Boolean = False
        Public uPlugStrucDispImg As String    'Display image at the U-Plug inputs/outputs configuration form
        Public uPlugStrucSetOfImg As String   'Path to a PDF or DWG file with standard design drawings
        ''' <summary>
        ''' The X Coordinate (Easting)
        ''' </summary>
        Public CoordX As Double
        ''' <summary>
        ''' The Y Coordinate (Northing)
        ''' </summary>
        Public CoordY As Double


    End Class

    Public Structure itemType
        Public abbreviation As String
        Public description As String
        Public value As Double
        Public unit As String
        ''' <summary>
        ''' Type of this item: h=hydraulic, s=structural, b=both
        ''' </summary>
        Public typeOfItem As Char
    End Structure

    Public Structure catType
        Public tuParentCatID As UInteger     'ID of major category type for a Treatment Unit
        Public tuParentCatCaption As String  'Caption/Title of the category
    End Structure
```

```
    Public Interface IGeneric
        Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As itemType
        Function SetParentCatType(ByVal i As UInteger, ByVal c As String) As catType
    End Interface

End Namespace

```

```vb
' ===============================================================================
' FILE NAME: Form1.vb
' ===============================================================================

'Imports WISAM.HIM.Generic
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim w As clsWWTUnit_ProtoType = New clsWWTUnit_ProtoType
        Dim i As HIM.Generic.itemType
        w.LoadDefaults(1)
        w.outputList = w.ProcessHydroCalc(w.inputList)
        For Each i In w.outputList
            MsgBox(i.abbreviation + " :  " & i.description + " : " & i.value & " " & i.unit)
        Next
    End Sub
End Class

```

```
Imports WISAM.HIM.Generic
Imports System.Drawing.Graphics
Imports Microsoft.VisualBasic.PowerPacks
Imports System.Reflection

Public Class Form2
    'Implements HIM.Generic.IdesignItems

    Private designItems As ArrayList = New ArrayList
    Public activeDesignItem As Int16 = -1
    Public lastDesignItem As Int16 = -1
    Public Const MAX_PATHS = 10
    Public designPaths(MAX_PATHS) As LineShape
    Public lastPath = 0
    Private isAddingNewItem As Boolean = False
    Private isAddingNewPath As Boolean = False
    'Private d As designItem
    Private newItemType As Integer = -1
    Private lastItemNo As Int16 = 1
    Private tmpDesignPath As designPath
    Public canvas As ShapeContainer
    Public tmpLine As LineShape

    Protected Friend lastMousePos As Point = New Point(0, 0)
    Protected Friend pathStart As Point = New Point(-1, -1)
    Protected Friend pathEnd As Point = New Point(-1, -1)

    Public Sub setStartPath(ByVal value As Object)
        tmpDesignPath.startItemTag = value
    End Sub

    Public Sub setEndPath(ByVal value As Object)
        tmpDesignPath.endItemTag = value
    End Sub

    Property setIsAddingNewPath As Boolean
        Get
            Return isAddingNewPath
        End Get
        Set(ByVal value As Boolean)
            isAddingNewPath = value
        End Set
    End Property

    Property setIsAddingNewItem As Boolean
        Get
            Return isAddingNewItem
        End Get
        Set(ByVal value As Boolean)
            isAddingNewItem = value
            If value Then PictureBox2.Cursor = Cursors.Cross
        End Set
    End Property

    Property setNewItemType As Integer
        Get
```

```
            Return newItemType
        End Get
        Set(ByVal value As Integer)
            newItemType = value
        End Set
    End Property

    'Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    '     isAddingNewItem = True
    '     newItemType = "W"
    '     PictureBox2.Cursor = Cursors.Cross
    'End Sub

    Private Sub PictureBox2_MouseClick(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox2.MouseClick
        If Not isAddingNewItem Then Exit Sub
        If e.Button <> Windows.Forms.MouseButtons.Left Then Exit Sub

        'd.X = e.X
        'd.Y = e.Y
        'd.Type = newItemType
        'designItems.Add(d)

        Dim img As New extPictureBox
        img.Image = PictureBox1.Image
        img.Parent = PictureBox2
        img.Location = New Point(e.X - (HIM.Generic.INIT_IMAGE.W / 2),
                                 e.Y - (HIM.Generic.INIT_IMAGE.H / 2))
        img.Visible = True
        img.itemType = newItemType
        img.ContextMenuStrip = Me.popupMenu
        img.Tag = CStr(lastItemNo)
        img.myNumber = lastItemNo
        isAddingNewItem = False

        'Dim asmMethod As MethodInfo
        Dim o(0) As Object
        o(0) = 1
        asmMethod = asmType(newItemType).GetMethod("LoadDefaults")
        asmObj(lastItemNo) = Activator.CreateInstance(asmType(newItemType))
        asmMethod.Invoke(asmObj(lastItemNo), o)
        'Dim cls As New clsGeneric
        'cls.uPlugUnitType = newItemType
        lastItemNo += 1

        PictureBox2.Cursor = Cursors.Default
        My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Ready"

    End Sub

    Private Sub PictureBox2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PictureBox2.Click

    End Sub

```

```
    Private Sub PictureBox2_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles PictureBox2.MouseMove
        My.Forms.mdiMainWindow.ToolStripStatusLabel1.Text = "Pos: " + CStr(e.X) + ", " + CStr(e.Y)

        If Not isAddingNewPath Then Exit Sub
        If pathStart.X < 0 Or pathStart.Y < 0 Then Exit Sub

        tmpLine.SendToBack()
        tmpLine.StartPoint = pathStart
        tmpLine.EndPoint = e.Location
        tmpLine.Visible = True
        Exit Sub


        lastMousePos = e.Location
        PictureBox2.Invalidate()
        Exit Sub

        Dim rc As New Rectangle

        If e.X > pathStart.X Then
            rc.X = pathStart.X
            rc.Width = e.X - pathStart.X
        Else
            rc.X = e.X
            rc.Width = pathStart.X - e.X
        End If
        If e.Y > pathStart.Y Then
            rc.Y = pathStart.Y
            rc.Height = e.Y - pathStart.Y
        Else
            rc.Y = e.Y
            rc.Height = pathStart.Y - e.Y
        End If
        lastMousePos = e.Location
        PictureBox2.Invalidate(rc)

    End Sub

    Private Sub PictureBox2_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles PictureBox2.Paint
        'e.Graphics.Clear(Me.BackColor)
        'e.Graphics.DrawLine(Pens.Black, pathStart, lastMousePos)
        For Each n As Control In PictureBox2.Controls
            If n.GetType().Name = "extPictureBox" Then
                n.Refresh()
            End If
        Next

    End Sub

    Public Sub New()

        ' This call is required by the designer.
        InitializeComponent()
```

```vb

        ' Add any initialization after the InitializeComponent() call.
        canvas = New ShapeContainer
        tmpLine = New LineShape
        canvas.Parent = PictureBox2
        tmpLine.Parent = canvas
        tmpLine.BorderWidth = 2
        tmpLine.BorderStyle = Drawing2D.DashStyle.Dash
        Dim i As Int16
        For i = 0 To MAX_PATHS - 1
            designPaths(i) = New LineShape
            designPaths(i).Parent = canvas
            designPaths(i).Visible = False
            designPaths(i).BorderWidth = 2
            designPaths(i).BorderStyle = Drawing2D.DashStyle.Dash
        Next
    End Sub

    Sub createNewPathLine()
        'Dim tmp As New LineShape(canvas)
        ''tmp = tmpLine
        ''tmp.Parent = canvas
        ''designPaths.Add(tmp)
        ''tmp.Visible = True
        designPaths(lastPath).X1 = tmpLine.X1
        designPaths(lastPath).X2 = tmpLine.X2
        designPaths(lastPath).Y1 = tmpLine.Y1
        designPaths(lastPath).Y2 = tmpLine.Y2
        designPaths(lastPath).Visible = True
        lastPath += 1
        tmpLine.Visible = False
    End Sub

    Private Sub RemoveToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles RemoveToolStripMenuItem1.Click
        Dim n As extPictureBox
        For Each p As Control In Me.PictureBox2.Controls
            If p.GetType().Name = "extPictureBox" Then
                n = Convert.ChangeType(p, GetType(extPictureBox))
                If n.BorderStyle = Windows.Forms.BorderStyle.FixedSingle Then
                    PictureBox2.Controls.Remove(p)
                End If
            End If
        Next

    End Sub

    Private Sub PropertiesToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PropertiesToolStripMenuItem1.Click
        Dim prop As New frmExtProperties
        prop.ShowDialog()
        prop.Dispose()

    End Sub
End Class
```

```vb
Public NotInheritable Class frmAbout

    Private Sub frmAbout_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        ' Set the title of the form.
        Dim ApplicationTitle As String
        If My.Application.Info.Title <> "" Then
            ApplicationTitle = My.Application.Info.Title
        Else
            ApplicationTitle = System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)
        End If
        Me.Text = String.Format("About {0}", ApplicationTitle)
        ' Initialize all of the text displayed on the About Box.
        ' TODO: Customize the application's assembly information in the "Application" pane of the project
        '    properties dialog (under the "Project" menu).
        Me.LabelProductName.Text = My.Application.Info.ProductName
        Me.LabelVersion.Text = String.Format("Version {0}", My.Application.Info.Version.ToString)
        Me.LabelCopyright.Text = My.Application.Info.Copyright
        Me.LabelCompanyName.Text = My.Application.Info.CompanyName
        Me.TextBoxDescription.Text = My.Application.Info.Description
    End Sub

    Private Sub OKButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OKButton.Click
        Me.Close()
    End Sub

End Class
```

```
' ===============================================================================
' FILE NAME: frmDefineNewUnit.vb
' ===============================================================================

Imports Microsoft.VisualBasic
Imports System.CodeDom
Imports System.CodeDom.Compiler

Public Class frmDefineNewUnit

    Public Const MAX_INPUTS = 10
    Public Const MAX_OUTPUTS = 10

    Private totalInputs, totalOutputs, totalEquations As Int16
    Private inputs(MAX_INPUTS) As HIM.Generic.itemType
    Private outputs(MAX_OUTPUTS) As HIM.Generic.itemType
    Private Equations() As String

    Private Sub compileClassCode()
        Dim cp As CompilerParameters = New CompilerParameters
        '****** To compile the base 'clsGeneric', uncomment the following line, and comment out
        '****** the line below it...
        '****** Make a copy of the 'defGeneric.vb' file and name it 'defGeneric.txt' in the app. dir

        'Only ONE of the following three lines would be active at a time
        '(The default is the normal case for running the final programme)
        'cp.OutputAssembly = Application.StartupPath + "\defGeneric.dll"
        cp.OutputAssembly = Application.StartupPath + "\clsGeneric.dll"
        'cp.OutputAssembly = Application.StartupPath + "\cls" + StrConv(TextBox3.Text, VbStrConv.ProperCase) + ".dll"


        cp.ReferencedAssemblies.Add("System.dll")
        cp.ReferencedAssemblies.Add("System.dll")
        cp.ReferencedAssemblies.Add("System.Data.dll")
        cp.ReferencedAssemblies.Add("System.Xml.dll")
        cp.ReferencedAssemblies.Add("mscorlib.dll")
        cp.ReferencedAssemblies.Add("System.Windows.Forms.dll")

        '************ comment out those two lines if you are compiling the 'defGeneric'
        '************ or 'clsGeneric' file
        cp.ReferencedAssemblies.Add(Application.StartupPath + "\defGeneric.dll")
        'cp.ReferencedAssemblies.Add(Application.StartupPath + "\clsGeneric.dll")

        cp.WarningLevel = 3
        cp.CompilerOptions = "/target:library /optimize"
        cp.GenerateExecutable = False
        cp.GenerateInMemory = False

        Dim tfc As New TempFileCollection(Application.StartupPath, False)
```

```
        Dim cr As New CompilerResults(tfc)

        '********** comment the following code line if you are compiling the 'clsGeneric',
        '********** and uncomment the following line ...

        'Only ONE of the following three lines would be active at a time
        '(The default is the normal case for running the final programme)
        'cr = CodeDomProvider.CreateProvider("VisualBasic").CompileAssemblyFromSource(cp, TextBox2.Text)
        cr = CodeDomProvider.CreateProvider("VisualBasic").CompileAssemblyFromFile(cp, Application.StartupPath + "\defGeneric.txt")
        'cr = CodeDomProvider.CreateProvider("VisualBasic").CompileAssemblyFromFile(cp, Application.StartupPath + "\clsGeneric.txt")

        Dim sc As System.Collections.Specialized.StringCollection = cr.Output

        If cr.Errors.Count > 0 Then
            For Each ce As CompilerError In cr.Errors
                MsgBox(ce.ErrorNumber + " : " + ce.ErrorText, MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, _
                       "Error in compilation..")
            Next
        Else
            MsgBox("Compilation succeeded. Output file:" + Application.StartupPath + "\cls" + _
                   StrConv(TextBox3.Text, VbStrConv.ProperCase) + ".dll", _
                   MsgBoxStyle.OkOnly + MsgBoxStyle.Information, "Compilation succeeded.")
            '******* if compiling defGeneric or clsGeneric, uncomment the following line
            'Exit Sub
            Dim File As IO.StreamReader
            Dim str As New System.Collections.Specialized.StringCollection
            File = My.Computer.FileSystem.OpenTextFileReader(Application.StartupPath() + "\uPlugs.dat")
            While Not File.EndOfStream
                str.Add(File.ReadLine)
            End While
            File.Close()
            Dim i As Int16
            For i = 0 To str.Count - 1
                str(i) = str(i).Trim
                If str(i).StartsWith("Units=") Then
                    str(i) = "Units=" + CStr(CInt(str(i).Substring(str(i).IndexOf("=") + 1)) + 1)
                    Exit For
                End If
            Next
            Dim File2 As IO.StreamWriter
            File2 = My.Computer.FileSystem.OpenTextFileWriter(Application.StartupPath() + "\uPlugs.dat", False)
            For i = 0 To str.Count - 1
                File2.WriteLine(str(i))
```

```
            Next
            File2.WriteLine()
            File2.WriteLine("[" + StrConv(TextBox3.Text, VbStrConv.ProperCase) + "]")
            File2.WriteLine("unitName=" + StrConv(TextBox3.Text, VbStrConv.ProperCase))
            File2.Close()
            Me.Close()
        End If

    End Sub

    Private Sub buildClassCode()
        Dim i As Int16
        TextBox2.Text = "Imports System.Collections"
        TextBox2.Text += vbCrLf + ""
        TextBox2.Text += vbCrLf + "Namespace HIM.Generic"
        TextBox2.Text += vbCrLf + ""
        TextBox2.Text += vbCrLf + "Public Class cls" + StrConv(TextBox3.Text, VbStrConv.ProperCase)
        TextBox2.Text += vbCrLf + "    Inherits clsGeneric"
        TextBox2.Text += vbCrLf + "    Implements Generic.IuPlugStructure"
        TextBox2.Text += vbCrLf
        TextBox2.Text += vbCrLf + "    Public Function GetInputs(ByVal type_of_WWTUnit As String) As String Implements HIM.Generic.IuPlugStructure.GetInputs"
        TextBox2.Text += vbCrLf + "        Return Nothing"
        TextBox2.Text += vbCrLf + "    End Function"
        TextBox2.Text += vbCrLf
        TextBox2.Text += vbCrLf + "    Public Function LoadDefaults(ByVal i As Integer) As Object Implements HIM.Generic.IuPlugStructure.LoadDefaults"
        For i = 0 To totalInputs - 1
            TextBox2.Text += vbCrLf + "        Dim input" + CStr(i + 1) + " As HIM.Generic.itemType = New HIM.Generic.itemType"
        Next
        TextBox2.Text += vbCrLf
        TextBox2.Text += vbCrLf + "        Me.uPlugUnitType = " + Chr(34) + TextBox3.Text + Chr(34)
        TextBox2.Text += vbCrLf + "        Me.uPlugUnitID = " + Chr(34) + TextBox3.Text + Chr(34) + " & i"
        TextBox2.Text += vbCrLf + "        Me.inputList.Clear()"
        For i = 0 To totalInputs - 1
            TextBox2.Text += vbCrLf + "        input" + CStr(i + 1) + " = SetItemType(" + Chr(34) + _
                inputs(i).abbreviation + Chr(34) + ", " + Chr(34) + inputs(i).description + _
                Chr(34) + ", " + CStr(inputs(i).value) + ", " + Chr(34) + inputs(i).unit + Chr(34) + ")"
            TextBox2.Text += vbCrLf + "        Me.inputList.Add(input" + CStr(i + 1) + ")"
            TextBox2.Text += vbCrLf + "        Return Nothing"
        Next
        TextBox2.Text += vbCrLf + "    End Function"
        TextBox2.Text += vbCrLf
        TextBox2.Text += vbCrLf + "    Public Function LoadFromFile(ByVal name_of_file As String) As String Implements
```

```
HIM.Generic.IuPlugStructure.LoadFromFile"
TextBox2.Text += vbCrLf + "        ' Read data from existing file."
TextBox2.Text += vbCrLf + "        Return Nothing"
TextBox2.Text += vbCrLf + "    End Function"
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf + "    Public Function ProcessHydroCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessHydroCalc"
TextBox2.Text += vbCrLf + "        ' Perform WWTUnit's processes and calculations."
TextBox2.Text += vbCrLf + "        Dim outputItem As HIM.Generic.itemType = New HIM.Generic.itemType"
TextBox2.Text += vbCrLf + "        Dim outputList As ArrayList = New ArrayList"
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf + "        outputList.Clear()"
TextBox2.Text += vbCrLf

For i = 0 To totalOutputs - 1
    TextBox2.Text += vbCrLf + "            outputItem.abbreviation = " + Chr(34) + outputs(i).abbreviation + Chr(34)
    TextBox2.Text += vbCrLf + "            outputItem.description = " + Chr(34) + outputs(i).description + Chr(34)
    TextBox2.Text += vbCrLf + "            outputItem.unit = " + Chr(34) + outputs(i).unit + Chr(34)
    TextBox2.Text += vbCrLf + "            outputItem.value = " + parseEquation(i)
    TextBox2.Text += vbCrLf + "            outputList.Add(outputItem)"
Next

TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf + "        Return outputList"
TextBox2.Text += vbCrLf + "    End Function"
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf + "    Public Function SaveToFile(ByVal name_of_file As String) As String Implements HIM.Generic.IuPlugStructure.SaveToFile"
TextBox2.Text += vbCrLf + "        ' Print data to specific file."
TextBox2.Text += vbCrLf + "        Return Nothing"
TextBox2.Text += vbCrLf + "    End Function"
TextBox2.Text += vbCrLf
TextBox2.Text += vbCrLf + "    'Public Function SetItemType(ByVal a As String, ByVal d As String, ByVal v As Double, ByVal u As String) As HIM.Generic.itemType Implements HIM.Generic.IuPlugStructure.SetItemType"
TextBox2.Text += vbCrLf + "        '    Dim tmpItem As HIM.Generic.itemType = New HIM.Generic.itemType"
TextBox2.Text += vbCrLf + "        '    tmpItem.abbreviation = a"
TextBox2.Text += vbCrLf + "        '    tmpItem.description = d"
TextBox2.Text += vbCrLf + "        '    tmpItem.value = v"
TextBox2.Text += vbCrLf + "        '    tmpItem.unit = u"
TextBox2.Text += vbCrLf + "        '    Return tmpItem"
TextBox2.Text += vbCrLf + "    'End Function"
TextBox2.Text += vbCrLf
```

```
        TextBox2.Text += vbCrLf + "    Public Function ProcessStrucCalc(ByVal inputsList As System.Collections.ArrayList) As System.Collections.ArrayList Implements HIM.Generic.IuPlugStructure.ProcessStrucCalc"
        TextBox2.Text += vbCrLf + "        Return Nothing"
        TextBox2.Text += vbCrLf + "    End Function"

        TextBox2.Text += vbCrLf + "    End Class"

        TextBox2.Text += vbCrLf + ""
        TextBox2.Text += vbCrLf + "End Namespace"

    End Sub

    Private Function parseEquation(ByVal i As Integer) As String
        Dim s As String = Equations(i).Substring(Equations(i).IndexOf("=") + 1)
        Dim d As Integer
        For d = 0 To totalInputs - 1
            s = s.Replace(inputs(d).abbreviation, "inputList(" + d.ToString + ")")
        Next
        Return s

    End Function

    Private Function defineInputs() As Boolean
        Dim i As Int16
        For i = 0 To DataGridView1.RowCount - 2
            inputs(i).abbreviation = DataGridView1.Rows(i).Cells("AbbreviationCol").Value
            inputs(i).description = DataGridView1.Rows(i).Cells("DescriptionCol").Value
            inputs(i).unit = DataGridView1.Rows(i).Cells("UnitCol").Value
            Try
                inputs(i).value = DataGridView1.Rows(i).Cells("ValueCol").Value
            Catch ex As Exception
                MsgBox("There were errors with your inputs/outputs. Please revise." + _
                       vbCrLf + vbCrLf + "Possible Causes:" + vbCrLf + _
                       "- Non-numerical entry in a 'Value' column.", _
                       MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, " Error..")
                Return False
            End Try
        Next
        totalInputs = i
        Return True
    End Function

    Private Function defineOutputs() As Boolean
        Dim i As Int16
        For i = 0 To DataGridView2.RowCount - 2
            outputs(i).abbreviation = DataGridView2.Rows(i).Cells("AbbreviationCol2").Value
            outputs(i).description = DataGridView2.Rows(i).Cells("DescriptionCol2").Value
            outputs(i).unit = DataGridView2.Rows(i).Cells("UnitCol2").Value
```

```
                ("ValueCol").Value
            Next
            totalOutputs = i
            Return True
        End Function

        Private Function fillInListBoxes() As Boolean
            ListBox1.Items.Clear()
            ListBox2.Items.Clear()
            Dim i As Int16
            Try
                If DataGridView2.RowCount > 1 Then
                    For i = 0 To DataGridView2.RowCount - 2
                        ListBox1.Items.Add(outputs(i).abbreviation)
                    Next
                End If
                If DataGridView1.RowCount > 1 Then
                    For i = 0 To DataGridView1.RowCount - 2
                        ListBox2.Items.Add(inputs(i).abbreviation)
                    Next
                End If
                Return True
            Catch e As ArgumentNullException
                MsgBox("There were errors with your inputs/outputs. Please revise."
                + _
                       vbCrLf + vbCrLf + "Possible Causes:" + vbCrLf + _
                       "- Empty entries in some of the fields.", _
                       MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, " Error..")
                If TabControl1.SelectedTab.Text = "TabPage1" Then
                    DataGridView1.Focus()
                Else
                    DataGridView2.Focus()
                End If
                Return False
            End Try

        End Function

        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles Button1.Click
            Select Case TabControl1.SelectedTab.Text
                Case "TabPage1"
                    If TextBox3.Text.Trim().Length = 0 Then
                        MsgBox("You must provide a name for the new unit type.",
                        MsgBoxStyle.Information + MsgBoxStyle.OkOnly, "Error..")
                        Exit Sub
                    End If
                    If DataGridView1.RowCount <= 1 Then
                        MsgBox("You must define at least one input.",
                        MsgBoxStyle.OkOnly + MsgBoxStyle.Exclamation, "Error..")
                        DataGridView1.Focus()
                        Exit Sub
                    End If
                    If defineInputs() Then
                        TabControl1.SelectTab("TabPage2")
                    End If
```

```
            Case "TabPage2"
                If DataGridView2.RowCount <= 1 Then
                    MsgBox("You must define at least one output.", MsgBoxStyle.OkOnly + MsgBoxStyle.Exclamation, "Error..")
                    DataGridView2.Focus()
                    Exit Sub
                End If
                If defineOutputs() Then
                    If fillInListBoxes() Then
                        TabControl1.SelectTab("TabPage3")
                    End If
                End If
            Case "TabPage3"
                If totalEquations <= 1 Then
                    MsgBox("You must define at least one output equation.", MsgBoxStyle.OkOnly + MsgBoxStyle.Exclamation, "Error..")
                    Exit Sub
                End If
                buildClassCode()
                TabControl1.SelectTab("TabPage4")
                Button1.Text = "&Finish"
            Case "TabPage4" 'TabControl1.SelectTab("TabPage4")
                compileClassCode()
        End Select
        Button3.Enabled = True
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
        Select Case TabControl1.SelectedTab.Text
            Case "TabPage1" 'Button3.Enabled = False
            Case "TabPage2"
                TabControl1.SelectTab("TabPage1")
                Button3.Enabled = False
            Case "TabPage3"
                TabControl1.SelectTab("TabPage2")
            Case "TabPage4"
                TabControl1.SelectTab("TabPage3")
                Button1.Text = "&Next"
        End Select

    End Sub

    Private Sub frmDefineNewUnit_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        totalEquations = 1
    End Sub

    Private Sub Button11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button11.Click
        totalEquations += 1
        ReDim Equations(totalEquations)
        ListBox3.Items.Add(TextBox1.Text)
        Equations(totalEquations - 2) = TextBox1.Text
        TextBox1.Text = ""
        Button11.Enabled = False
```

```
        Button13.Enabled = True
    End Sub

    Private Sub Button10_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button10.Click
        TextBox1.Text = ""
    End Sub

    Private Sub Button13_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button13.Click
        Dim i As Int16
        i = MsgBox("Remove all equations?. This can not be undone.", MsgBoxStyle.Exclamation + MsgBoxStyle.YesNo, "Confirm delete...")
        Select Case i
            Case MsgBoxResult.Yes
                totalEquations = 1
                ListBox3.Items.Clear()
                Button13.Enabled = False
            Case Else
                Exit Sub
        End Select
    End Sub

    Private Sub ListBox1_MouseDoubleClick(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles ListBox1.MouseDoubleClick
        If ListBox1.SelectedIndex < 0 Then Exit Sub
        TextBox1.Text += ListBox1.SelectedItem.ToString
    End Sub

    Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged

    End Sub

    Private Sub ListBox2_MouseDoubleClick(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles ListBox2.MouseDoubleClick
        If ListBox2.SelectedIndex < 0 Then Exit Sub
        TextBox1.Text += ListBox2.SelectedItem.ToString
    End Sub

    Private Sub ListBox2_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox2.SelectedIndexChanged

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
        TextBox1.Text += "="
    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
        TextBox1.Text += "+"
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
```

```
        System.EventArgs) Handles Button6.Click
            TextBox1.Text += "-"
        End Sub

        Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
            TextBox1.Text += "*"
        End Sub

        Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button8.Click
            TextBox1.Text += "/"
        End Sub

        Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button9.Click
            TextBox1.Text += "^"
        End Sub

        Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged
            If TextBox1.Text.Length = 0 Then
                Button11.Enabled = False
            Else
                Button11.Enabled = True
            End If
        End Sub

        Private Sub ListBox3_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox3.SelectedIndexChanged
            Button12.Enabled = True
        End Sub

        Private Sub Button14_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button14.Click
            compileClassCode()

        End Sub
    End Class
```

```vb
' ==============================================================================
' FILE NAME: frmProperties.vb
' ==============================================================================

Public Class frmProperties

    Private Sub DataGridView1_CellContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellContentClick

    End Sub

    Private Sub DataGridView1_CellEndEdit(ByVal sender As Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles DataGridView1.CellEndEdit
        Dim tmp(DataGridView1.RowCount * 4 - 1) As String
        Dim i As Int16
        For i = 0 To tmp.Length - 1 Step 4
            tmp(i) = DataGridView1.Rows(i / 4).Cells(0).Value
            tmp(i + 1) = DataGridView1.Rows(i / 4).Cells(0).Value
            tmp(i + 2) = DataGridView1.Rows(i / 4).Cells(1).Value
            tmp(i + 3) = DataGridView1.Rows(i / 4).Cells(1).Value
        Next
        asmMethod = asmType(m_frmDesign.setNewItemType).GetMethod("ChangeInputs")
        Dim tmp2() As Object = {tmp}
        Dim b As Boolean = asmMethod.Invoke(asmObj(m_frmDesign.activeDesignItem), tmp2)

    End Sub
End Class
```

```vb
' ==============================================================================
' FILE NAME: frmToolBox.vb
' ==============================================================================

Public Class frmToolBox

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        m_frmDesign.setNewItemType = "W"
        m_frmDesign.setIsAddingNewItem = True
        m_frmDesign.setIsAddingNewPath = False
        My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Select where to add the new unit.."

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        m_frmDesign.setIsAddingNewPath = True
        m_frmDesign.setIsAddingNewItem = False
        My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Select path start point.."

    End Sub

    REM Private Sub frmToolBox_Shown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shown
    '
    '
    '        Exit Sub
    '
    '
    REM Dim i As Integer
    '    Dim tmp As extButton
    REM     For i = 0 To buttons.Length - 2
    '    'buttons(i).Parent = FlowLayoutPanel1
    '    '            buttons(i).setParent()
    '            tmp = buttons(i)
    '            tmp.Dock = DockStyle.Fill
    '            tmp.Visible = True
    '            FlowLayoutPanel1.Controls.Add(tmp)
    '    'buttons(i).Dock = DockStyle.Fill

    'buttons(i).Visible = True
    REM         TreeView1.Nodes(0).Nodes.Add(buttons(i).Text)
    REM     Next
    REM End Sub

    Private Sub TreeView1_AfterSelect(ByVal sender As System.Object, ByVal e As System.Windows.Forms.TreeViewEventArgs) Handles TreeView1.AfterSelect
    End Sub

    Private Sub TreeView1_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles TreeView1.KeyDown
        If e.KeyCode = Keys.Escape Then
            m_frmDesign.setIsAddingNewItem = False
```

```
                m_frmDesign.setIsAddingNewPath = False
                My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Ready"
            End If

        End Sub

        Private Sub TreeView1_NodeMouseClick(ByVal sender As Object, ByVal e As System.Windows.Forms.TreeNodeMouseClickEventArgs) Handles TreeView1.NodeMouseClick
            '********** if a Root node is selected, reset the status bar to 'Ready', and exit sub
            If e.Node.Level = 0 Then
                My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Ready"
                Exit Sub
            End If
            '********** Otherwise, allow the user to add new unit of the type selected...
            m_frmDesign.setNewItemType = e.Node.Parent.Tag
            m_frmDesign.setIsAddingNewItem = True
            m_frmDesign.setIsAddingNewPath = False
            My.Forms.mdiMainWindow.ToolStripStatusLabel.Text = "Select where to add the new '" + _
                unitNames(m_frmDesign.setNewItemType) + "' unit.."

        End Sub
    End Class
```

```vb
'===============================================================================
' FILE NAME: mdiMainWindow.vb
'===============================================================================

Imports System.Windows.Forms
Imports System.Reflection

Public Class mdiMainWindow

    Private Sub ShowNewForm(ByVal sender As Object, ByVal e As EventArgs) Handles NewToolStripMenuItem.Click, NewToolStripButton.Click, NewWindowToolStripMenuItem.Click
        ' Create a new instance of the child form.
        Dim ChildForm As New System.Windows.Forms.Form
        ' Make it a child of this MDI form before showing it.
        ChildForm.MdiParent = Me

        m_ChildFormNumber += 1
        ChildForm.Text = "Window " & m_ChildFormNumber

        ChildForm.Show()
    End Sub

    Private Sub OpenFile(ByVal sender As Object, ByVal e As EventArgs) Handles OpenToolStripMenuItem.Click, OpenToolStripButton.Click
        Dim OpenFileDialog As New OpenFileDialog
        OpenFileDialog.InitialDirectory = My.Computer.FileSystem.SpecialDirectories.MyDocuments
        OpenFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"
        If (OpenFileDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK) Then
            Dim FileName As String = OpenFileDialog.FileName
            ' TODO: Add code here to open the file.
        End If
    End Sub

    Private Sub SaveAsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles SaveAsToolStripMenuItem.Click
        Dim SaveFileDialog As New SaveFileDialog
        SaveFileDialog.InitialDirectory = My.Computer.FileSystem.SpecialDirectories.MyDocuments
        SaveFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"

        If (SaveFileDialog.ShowDialog(Me) = System.Windows.Forms.DialogResult.OK) Then
            Dim FileName As String = SaveFileDialog.FileName
            ' TODO: Add code here to save the current contents of the form to a file.
        End If
    End Sub


    Private Sub ExitToolsStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles ExitToolStripMenuItem.Click
```

```vb
        Me.Close()
    End Sub

    Private Sub CutToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles CutToolStripMenuItem.Click
        ' Use My.Computer.Clipboard to insert the selected text or images into the clipboard
    End Sub

    Private Sub CopyToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles CopyToolStripMenuItem.Click
        ' Use My.Computer.Clipboard to insert the selected text or images into the clipboard
    End Sub

    Private Sub PasteToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles PasteToolStripMenuItem.Click
        'Use My.Computer.Clipboard.GetText() or My.Computer.Clipboard.GetData to retrieve information from the clipboard.
    End Sub

    Private Sub ToolBarToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles ToolBarToolStripMenuItem.Click
        Me.ToolStrip.Visible = Me.ToolBarToolStripMenuItem.Checked
    End Sub

    Private Sub StatusBarToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles StatusBarToolStripMenuItem.Click
        Me.StatusStrip.Visible = Me.StatusBarToolStripMenuItem.Checked
    End Sub

    Private Sub CascadeToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles CascadeToolStripMenuItem.Click
        Me.LayoutMdi(MdiLayout.Cascade)
    End Sub

    Private Sub TileVerticalToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles TileVerticalToolStripMenuItem.Click
        Me.LayoutMdi(MdiLayout.TileVertical)
    End Sub

    Private Sub TileHorizontalToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles TileHorizontalToolStripMenuItem.Click
        Me.LayoutMdi(MdiLayout.TileHorizontal)
    End Sub

    Private Sub ArrangeIconsToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles ArrangeIconsToolStripMenuItem.Click
        Me.LayoutMdi(MdiLayout.ArrangeIcons)
    End Sub

    Private Sub CloseAllToolStripMenuItem_Click(ByVal sender As Object, ByVal e As EventArgs) Handles CloseAllToolStripMenuItem.Click
        ' Close all child forms of the parent.
        For Each ChildForm As Form In Me.MdiChildren
            ChildForm.Close()
```

```vb
        Next
    End Sub

    Private m_ChildFormNumber As Integer
    '    Public m_frmDesign As Form2
    '    Public m_frmProperties As frmProperties
    '    Public m_frmToolBox As frmToolBox

    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click
        If frmAbout.Modal = False Then frmAbout.Show(Me)
    End Sub

    Private Sub mdiMainWindow_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Me.SetDesktopLocation(0, 0)

        loadChildren()
        parseSettingsFile()
        showChildren()

        m_frmToolBox.TreeView1.ExpandAll()
        ToolboxToolStripMenuItem.Checked = True
        PropertiesToolStripMenuItem.Checked = True
        DesignerToolStripMenuItem.Checked = True

        Me.ToolStripStatusLabel.Text = "Ready"
        Me.ToolStripStatusLabel1.Text = ""
    End Sub

    Public Sub parseSettingsFile()
        Dim File As IO.StreamReader
        File = My.Computer.FileSystem.OpenTextFileReader(Application.StartupPath() + "\uPlugs.dat")
        Dim str As String
        Dim i As Int16, totalUnits As Int16
        Dim prop, value As String
        Dim tmpAsmName As String
        Dim currentUnit As Int16 = -1
        str = File.ReadLine()
        While Not File.EndOfStream
            str = str.Trim()
            '**** Comments in the file start with #
            '**** any line that starts with a pound sign is a comment
            If str.StartsWith("#") Then
                str = File.ReadLine()
            Else
                '**** The line that starts with 'Units=' is the number of units defined
                If str.StartsWith("Units") Then
                    totalUnits = CInt(str.Substring(str.IndexOf("=") + 1))
                    If totalUnits <= 0 Then
                        MsgBox("Unit Defition file is corrupt. You might need to re-install the program.", _
                                  MsgBoxStyle.Critical + MsgBoxStyle.OkOnly, "Fatal Error..")
```

```vb
                        Application.Exit()
                    End If
                    ReDim unitNames(totalUnits)
                    ReDim asm(totalUnits)
                    ReDim asmType(totalUnits)
                    ReDim buttons(totalUnits)
                    str = File.ReadLine()
                    'End If
                    '**** Unit-specific definitions -- depends on the number totalUnits
                ElseIf str.StartsWith("[") Then
                    str = str.Remove(0, 1)
                    If str.EndsWith("]") Then str = str.Remove(str.Length - 1, 1)
                    currentUnit += 1
                    tmpAsmName = str

                    str = File.ReadLine()
                    While Not str Is Nothing
                        If str.StartsWith("[") Then Exit While
                        If str.StartsWith("#") Then
                            str = File.ReadLine()
                        Else
                            i = str.IndexOf("=")
                            If i > 0 Then
                                prop = str.Substring(0, i)
                                value = str.Substring(i + 1, str.Length - i - 1)
                                addNewUnitButton(prop, value, tmpAsmName, currentUnit)
                            End If
                            str = File.ReadLine()
                        End If
                    End While
                Else
                    str = File.ReadLine()
                End If
            End If
        End While
    End Sub

    Public Sub addNewUnitButton(ByVal prop As String, ByVal value As String, ByVal tmpAsmName As String, _
                                ByVal currentUnit As Int16)
        Exit Sub

        If prop = "unitName" Then
            asm(currentUnit) = Assembly.LoadFrom(Application.StartupPath + "\cls" + tmpAsmName + ".dll")
            unitNames(currentUnit) = value
            asmType(currentUnit) = asm(currentUnit).GetType("WISAM.HIM.Generic.cls" + unitNames(currentUnit))

            buttons(currentUnit) = New extButton(currentUnit)
            buttons(currentUnit).Text = unitNames(currentUnit)

            buttons(currentUnit).Parent = m_frmToolBox.FlowLayoutPanel1
```

```
            buttons(currentUnit).Visible = True

        ElseIf prop = "unitType" Then
            'With m_frmToolBox.TreeView1
            '************** Is there a Root node with the current Type?
            Dim tn() As TreeNode = m_frmToolBox.TreeView1.Nodes.Find(value, True)
            '************** If yes, add the current unit under that Root node
            If tn.Length > 0 Then
                tn(0).Nodes.Add(unitNames(currentUnit), unitNames(currentUnit))
                '************** Otherwise, create a new Root node with the type selected,
                '************** Then add the current unit under it
            Else
                m_frmToolBox.TreeView1.Nodes.Add(value, value)
                tn = m_frmToolBox.TreeView1.Nodes.Find(value, True)
                tn(0).Tag = currentUnit
                'tn = m_frmToolBox.TreeView1.Nodes.Find(value, True)
                tn(0).Nodes.Add(unitNames(currentUnit), unitNames(currentUnit))
            End If
            'End With
        End If

    End Sub


    Private Sub PropertiesToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PropertiesToolStripMenuItem.Click
        If PropertiesToolStripMenuItem.Checked Then
            PropertiesToolStripMenuItem.Checked = False
            m_frmProperties.Hide()
        Else
            Try
                m_frmProperties.Show()
            Catch ex As Exception
                m_frmProperties = New frmProperties
                m_frmProperties.MdiParent = Me
                m_frmProperties.Show()
            End Try
            PropertiesToolStripMenuItem.Checked = True
        End If
    End Sub

    Private Sub ToolboxToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ToolboxToolStripMenuItem.Click
        If ToolboxToolStripMenuItem.Checked Then
            ToolboxToolStripMenuItem.Checked = False
            m_frmToolBox.Hide()
        Else
            Try
                m_frmToolBox.Show()
            Catch ex As Exception
                m_frmToolBox = New frmToolBox
                m_frmToolBox.MdiParent = Me
                m_frmToolBox.Show()
            End Try
```

```
                ToolboxToolStripMenuItem.Checked = True
            End If
        End Sub

        Private Sub DesignerToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DesignerToolStripMenuItem.Click
            If DesignerToolStripMenuItem.Checked Then
                DesignerToolStripMenuItem.Checked = False
                m_frmDesign.Hide()
            Else
                Try
                    m_frmDesign.Show()
                Catch ex As Exception
                    m_frmDesign = New Form2
                    m_frmDesign.MdiParent = Me
                    m_frmDesign.Show()
                End Try
                DesignerToolStripMenuItem.Checked = True
            End If
        End Sub

        Private Sub DefineNewUnitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles DefineNewUnitToolStripMenuItem.Click
            Dim newUnit As New frmDefineNewUnit
            newUnit.ShowDialog()
            newUnit.Dispose()
        End Sub

    End Class

```

```
' ==============================================================================
' FILE NAME: modAssembly.vb
' ==============================================================================

Imports System.Reflection

Module modAssembly
    Public asm() As Assembly
    Public asmType() As Type
    Public asmMethod As MethodInfo
    Public asmObj(20) As Object
    Public asmMembers() As MemberInfo

    Public unitNames() As String

    Public buttons() As extButton

    '***** GENERAL FORM DEFINITIONS
    Public m_frmDesign As Form2
    Public m_frmProperties As frmProperties
    Public m_frmToolBox As frmToolBox


    Public Sub loadChildren()

        m_frmDesign = New Form2
        m_frmDesign.MdiParent = My.Forms.mdiMainWindow
        With My.Forms.mdiMainWindow
            m_frmDesign.SetDesktopLocation(.DesktopLocation.X + ((.ClientSize.Width - m_frmDesign.Width) / 2),
                                           .DesktopLocation.Y + ((.ClientSize.Height - m_frmDesign.Height) / 2))
        End With

        m_frmProperties = New frmProperties
        m_frmProperties.MdiParent = My.Forms.mdiMainWindow
        With My.Forms.mdiMainWindow
            m_frmProperties.SetDesktopLocation(.DesktopLocation.X + (.ClientSize.Width - m_frmProperties.Width), 0)
        End With
        REM Me.DesktopLocation.Y + (Me.ClientSize.Height - m_frmProperties.Height))

        m_frmToolBox = New frmToolBox
        m_frmToolBox.MdiParent = My.Forms.mdiMainWindow
        m_frmToolBox.SetDesktopLocation(0, 0)
        m_frmToolBox.TreeView1.Nodes.Clear()

        'm_frmDesign.Show()
        'm_frmProperties.Show()
        'm_frmToolBox.Show()

    End Sub

    Public Sub showChildren()
        m_frmDesign.Visible = True
```

```
        m_frmProperties.Visible = True
        m_frmToolBox.Visible = True

        m_frmDesign.Show()
        m_frmProperties.Show()
        m_frmToolBox.Show()

    End Sub

End Module

```