International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)

http://ijcjournal.org/

The Unified Methodology for Application and Database Development UMADD (Using "abbCode" Framework)

Ahmed Fahmy*

Address: 11511, Cairo, Egypt
Email: ahmedfahmyaee@yahoo.com

Abstract

The study describes a new methodology for architecting software using a new developed tool in a way that make development processes easier, increase productivity and reducing bugs and potential errors avoiding overheads.

Keywords: Software architecture; Software Development; Software Design Model; Code Reusing; Development Tool.

1. Introduction

While tools and third parties are developed by time to make a development process easier and done effectively in a sufficient way in order to increase productivity in all ways as possible. With existence of many development tools and code libraries which are developed software architecture methodologies stay kept untouched and no revolutionary changes is done and no new methodologies is made. Developer and database administrators still working with data using SQL language and another traditional technologies where data is exists away from an application level so a software performance and n-tier architecture and OOP principles become a critical and need a great of real consideration when design and develop software. As a result of what have been mentioned and the productivity issue is in concern some questions are asked to go beyond the limitations of the existed software architecture designing and implementing approaches and principles: How much sequences should be repeated, How much chunks of codes and functions should be written to be called for one time, How many functions should be written for each stack or sequence and how many lines of code should be written for each function.

---

* Corresponding author.

It is not about OOP, it is all about the real meaning of simplicity, readability, maintainability and the real meaning of dynamic software architecture and productivity as not seen before.

2. Main Philosophy

"*Coding is nothing but thinking*" The phrase which is written by the author of abbCode project which refers to the main philosophy behind UMADD software architecture since the world is extremely expanding and business is going more complicated in the same time the system must fulfills all it's auto growing requirements and satisfies it's needs there becomes no time to spend in development processes the issue which really effects the quality of the final software product.

So the idea if effort spent in writing code is totally saved the time will be only to think about creative ideas, new methodologies and great algorithms will be designed and executed in order to get the state of the art software product created by great and relaxed developer's mind.

3. UMADD Architecture

3.1. Overview

UMADD is an abbreviation of Unified Methodology for Application and Database Development, which is can lead to understand the idea which is behind, the main idea of the architecture is to composite the levels of software in order to achieve unified software architecture that consists of only less number of managed levels.

3.2. UMADD Architecture in Details

Where the software basically consists of three levels (front end (Presentation), application and backend (Database)), see the following. [1]

Figure 1: Traditional Software Architecture Levels [2]

In the traditional software architecture the three levels are separated with a dataflow in ongoing request and in upcoming responses which are expressed using the above arrows.

As a result the software three levels are separated physically and programmatically.

So if some principles are needed to differentiates from the physical and programmatic separations:

- **The physical type of separation:** Refers to a type of separation where the level of the software is

physically exists a way form other levels of the software the situation where the data need to be traveled between using application's requests and responses.

Types of configuration processes between levels like connection strings, web services, cloud or server configurations etc are considered types of physical connection logic.

- **The programmatic type of separation:** As the opposite of concept of physical type the programmatic type is type of separation where the levels is physically exists in the same level close to each others where the separation is done programmatically since each one of the physical related levels have its specific logic which is not corresponds to another physical related level so as example in this case the developer in need to develop a third logic layer between two programmatically separated levels without any logic to connect them physically since they are already physically related and connected.

Any types of logical coding to manipulate data and make various processes against it or to act with user inputs and respond to it are considered type of programmatic connection logic.

So if we can write a simple equation to summarize where:

Ly = Layer

Ph-CL = Physical Connection Logic

Pr-CL = Programmatic Connection Logic

The function is:

$$Ly(Ph\text{-}CL \;\&\; Pr\text{-}CL \mid Pr\text{-}CL) \qquad (1)$$

Figure 2: UMADD Software Architecture Levels [2]

As we have noticed the traditional software architecture has a physical type separation between the three software levels, where the situation is not the same in UMADD architecture which as it clearly appears from the above figure has only the physical type of separation between the two levels only the presentation level and the composite level which is a composite of two programmatically separated levels: application level and back end level. So we need only one physical connecting logic layer in UMADD architecture.

So in the following comparison figures notice the physical and programmatic connection logics in depth where the red arrow refers to physical connection logic and the blue arrow refers to programmatic connection logic:

Figure 3: Traditional Software Architecture Levels with Connection Logic [2]

Pair types of physical and programmatic connection logic are must to use when each level works with others in traditional software architecture.

Figure 4: UMADD Software Architecture Levels with Connection Logic [2]

Pair types of physical and programmatic connection logic are used only when working between front end and composite levels and only a programmatic type is used when working with application and back end levels since they are natively composed as one physical level.

4. Tool Behind the Methodology

abbCode is a fully object oriented library built based on and on top of .Net and ASP.Net frameworks using c# programming language so it is compatible and built to use with ASP.Net, C# and SQL Server database. "*abbCode is an ASP.Net library which encapsulates a chunk of .net code to achieve specific tasks which are used in server side. Each chunk of code is abbreviated in a function to be called by the end user.*" The description from Egyptian Information Technology Industry development Agency's software registration documents. abbDB is a part of abbCode which is an initialization of new relational database management systems generation which supports all relation types between data entities: one to one, one to many and many to many.

Where the data storage is locally exists on the server and the database engine is a native part of abbCode library so no any installation or configuration is required to interact or working with abbDB the something that make working with a data is part of application coding process.

abbDB has a powerful database engine which make the abbDB database is a self managed, data manipulator and auto process detector using unique seamless smart abbCode APDA, ICGA and IDLMA algorithms which give the abbDB its smart and powerful behavior.

Native physical nature of abbCode, abbCode internal smart algorithms, No-SQL methodology and Inline query expressions are a collection that distinguish abbDB as new generation of database management system which

reach a peak with a performance and productivity and clearly reflects the quality factors.

5. Main Algorithms

5.1. Algorithms Overview

5.1.1. Code Driving via Parameters Algorithm CDPA

CDPA is an abbCode based algorithm built to give the ultimate controls of the entire code by allowing developer to configure, redirect even ultimately change the operation and final result by only configuring the same function's parameters in a predefined restricted format specific for the desired process and result.

5.1.2. Auto Process Detecting and Redirecting Algorithm APDA

Once the developer formats the function's parameter to meet a desired task and result formats, APDA get the parameters and internally choose suitable processes which correspond with the supplied parameter's format in order to achieve the developer's desired processes and result.

5.1.3. Internal Code Generation Algorithm ICGA

It is a third level of abbCode internal algorithms serialization which depends on the two previous algorithms CDPA to get the developer's ideas and criteria and APDA to detect and redirect to suitable process then it is a time of generating and executing a collection of abbCode functions behind the scene to achieve the previously specified task using ICGA algorithm.

5.1.4. Internal Database Logic Manipulation Algorithm IDLM

abbDB uses all of abbCode internal algorithms to manipulate data and make all the required processes inside the database without writing any application logical code to achieve so calculation, cryptography, etc are auto generated using the smart internal database engine.

5.2. Developer's Algorithms Interaction

All of algorithms which stand behind the UMADD architecture using "abbCode" project exchange data and work in symphony with each other’s without any kind intervention from the developer who only simply works with CDPA so others are kept untouched and completely work behind the seen based on the data given by the initializing CDPA algorithm only which controls and configure the whole internal processes.

Developer(CDPA(APDA, ICGA ,IDLMA)) *(1)*

6. Coding Principles and Implementation

6.1. Ease of Use

Effortless steps to understand and familiarize since no specific mentality and technical language are needed "*Development becomes a task for a non developers!*" a quote which may nearly widely said.

Since the functions definition in abbCode is almost the same as name of the task which the function is created for and that the way that all abbCode functions definitions written.

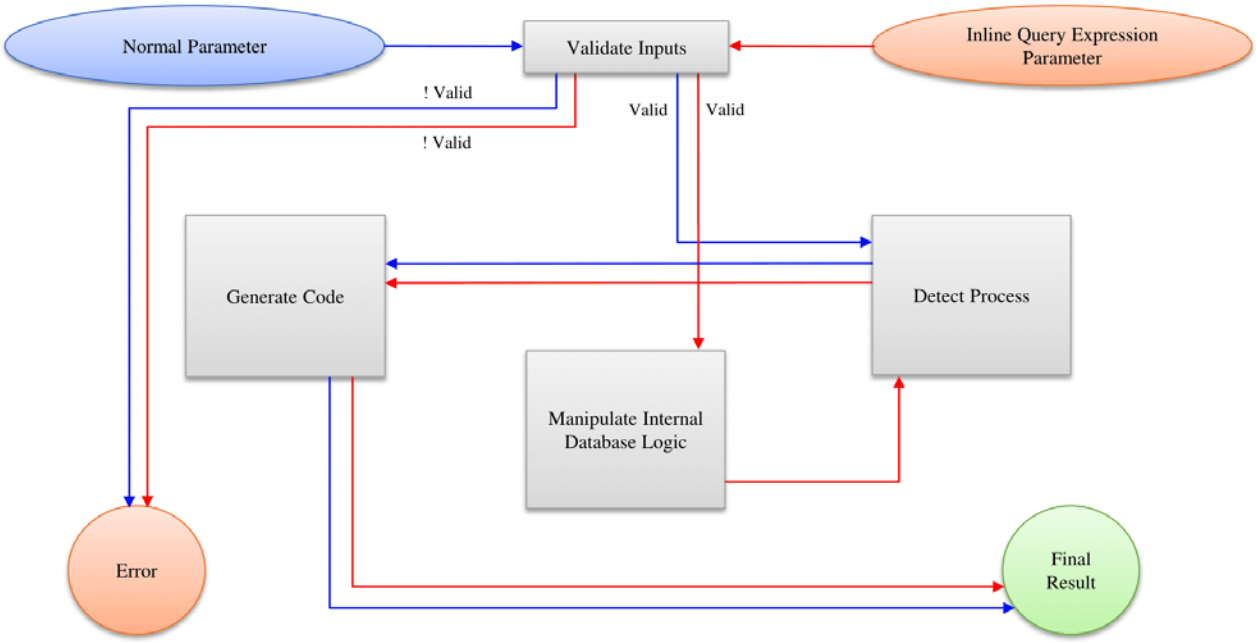

Figure3: Algorithms in Action [2]

So nothing but calling the function is needed to fulfill a requirements and achieving specific task in all abbCode functions so see the following two chunks of code as a comparison between traditional ASP.NET code and abbCode to implement uploading function.

Traditional Code [3]

```
if (Uploader.PostedFile.ContentLength != 0)
{
   if (Uploader.PostedFile.ContentLength > 1048576)
   {
      lblStatus.Text = "Too large. This file is not allowed";
   }
   else
   {
      string destDir = Server.MapPath("./Upload");
      string fileName = Path.GetFileName(Uploader.PostedFile.FileName);
      string destPath = Path.Combine(destDir, fileName);
      Uploader.PostedFile.SaveAs(destPath);
      lblStatus.Text = "Thanks for submitting your file.";
```

```
    }
}
```

abbCode

```
UploadFile(List<string> ControlId, List<string> FileDestination, List<string> FileName,
            string ContentPlaceHolderId = null);
```

Notice the difference between the traditional and abbCode methodologies where how many lines of codes to create a functionality to upload only one file to the server whenever the simple one line to call an abbCode function *UploadFile()* can upload multiple files at the time effectively each file to the different specific destination simply by configuring the first three first parameters as follows:

- List of FileUpload controls IDs.
- List of directories on the server to upload each file to.
- List of desired file names to end uploaded with.

* Three lists must be the same items count since each item index corresponds with equivalent item index in the next list.

When compare traditional methodologies to abbCode to do tasks in ASP.NET you then compare a coding with an almost no coding methodology.

6.2. Readability

All function definitions, parameters and returned static values are written in simple clear English language, since abbCode is originally built to be self explanatory and have no period of time to learn, so clear function definition and parameter definition syntax which can refer clearly to the function's desired inputs and task.

```
FindButtonControl(string ControlId, string ContentPlaceHolderId = null);
RenameFile(string CurrentFileName, string NewFileName);
```

FindButtonControl() is one of a collections of functions to find, access and interacts with web controls from outside its native environment the functionality which has a great effect of interaction between presentation and composite level since the developer can work with all web controls inside the composite level which strongly supports the unified architecture.

6.3. Code Form Unifying

1. All function's definitions and parameters are written in the same way using the same syntax and technical language.

2. The complex data is always populated and send to abbCode using two types of array which are two dimensional and jagged arrays.

6.4. Code Driving

Great and total control over the code and data flow is allowed by only changing, configuring, ignoring even editing the parameters of the function in a simple words in abbCode tasks is executed by calling function and results is executed by configuring parameters.

```
InsertXmlData(string File, string RootNode, string ParentNode, string ParentNodeAttributes,
        string[][] ChildNodes, bool CheckDataKey = false, string 1ToManyFilePath = null,
        string 1ToManyXPathExpression = null);
```

The behavior of *InsertXmlData()* function could be changed in sometimes as follows for example:

1. If the first optional parameter *CheckDataKey* is supplied the data will not be inserted to the XML file if there is a node with the same data value.

The parameter gives the function a behavior of primary key violation of a normal database.

2. The second two optional parameters *1ToManyFilePath* and *1ToManyXPathExpression* which specify the XML file that works as a parent that has a primary key of the current file the situation which gives working with XML files the powerful and flexibility of normal relational database.

* Notice that XML file data manipulation process ultimately changes to behave as a relational database tables. The code driving feature of abbCode library is achieved by working correctly with parameters which are passed to function by configuring or ignoring some optional parameters in some cases in a way that can ultimately changes the way the functions work and gives another results as we shown.

6.5. Extendibility

It is a control in other words where the same function could gives a totally new task and results by configuring its parameter's set.

```
SelectSqlData(string Conn, string Sproc, string[] Params);
SelectSqlData(string Conn, string Sproc, string[,] Params, string Order = null);
SelectSqlData(string Conn, string Sproc, string[,] Params, string Order = null, string PageSize = null,
        string CurrentPage = null, string HdnCurrentPage = null, string HdnTotalPages = null,
        string HdnTotalRows = null, string BtnFirst = null, string BtnPrevious = null,
        string TxtPageNumber = null, string BtnPageNumber = null, string BtnNext = null,
        string BtnLast = null, string LblCurrentPage = null, string LblTotalPages = null,
        string LblTotalSelectedRows = null, string LblTotalRows = null,
        string NavigationLevel = null, string ContentPlaceHolderId = null);
```

SelectSqlData() and not only shows an extremely behavior which ultimately extend the task from selecting one

single record to select a huge set of data efficiently with ultimate control of a paging process and pager controls with no further code to write but calling this function even for UI.

6.6. No-SQL Language Methodology

abbDB uses normal abbCode function with an inline query expression to interact with a database without any middle languages or technologies such SQL languages but using Inline query expression, Inline query expression has some specific characteristics:

- It is a type of smart expression used as a normal string parameter to pass to abbCode *ExecuteDatabaseQuery(string Expression)* function to executes.
- User clear syntax using commas, quotations, braces and curly braces, to organize the syntax and improve readability.
- Allows multiple data entities creations and writing complex composite CRUD queries using one single expression to be executed as a single one query.
- Internal code logic is executed if desired by only calling the function with a data field as aparameter within the expression EncryptMD5 function.

```
ExecuteDatabaseQuery ("database_name{tables{tbl_name1{col_name1[data_type(size), allow_null,
key_role, auto_increment,
func(EncryptMD5)],
col_name2[data_type(size), allow_null,
key_role, auto_increment]},
"tbl_name2{col_name1[data_type(size), allow_null,
key_role, auto_increment,
col_name2[data_type(size), allow_null,
key_role, auto_increment]}},
"relations{relation_name{tbl_name1.col_name1 >
tbl_name2.col_name1}}}");
```

6.7. Productivity

As a result of the easiness, readability, code form unified and driving, extendibility and No-SQL language methodology the UMADD design model reaches a peak with a productivity factor.

6.8. Maintainability

When no code is written no maintenance is required it is also a reconfiguring if necessary to say.

1. Instead of traditional methods of writing code with a special attention to every characters and each line of code to execute a specific task in the desired manner the coding becomes just a calling of functions, the methodology which decreases the potential bugs and coding errors.

2. abbCode is a smart auto code generator which detects the process and generates and manages all the code internally behind the developer's area so usability and maintainability is a process of just reconfiguring

of existed logic whatever the changes of the system requirements is extremely big.

3. The unified software architecture avoid the developers a lot of potential bothering bugs which mostly thrown due to a lot of required configuration in a physical connection level which connects application and the back end level since they are already one physical level.

7. Conclusions

As a result at the end UMADD Architecture is an idea implemented and executed using abbCode framework which are also an idea of productivity and creativity that go beyond the revolutionary effect the software architecture in a way which make the architecture is also auto generated process done by it.

So the "abbCode" project idea is not limited to specific development platform or programming language the idea is a methodology and a way of thinking which should be implemented using all used programming languages and technologies.

8. Recommendations

The time of a new form of software engineering researches must initialize where the papers should propose a new methodologies and technologies which can ultimately change the overall architecture and the way the code is written in order to achieve the real meaning of productivity based on main factors of quality and time of implementation the issue which will lead to get the developer completely focused in only creating a logic for a great and smart state of the art software products by overlapping the software processes and development overheads and headache of coding for a complicated systems.

References

[1] Tony Marston. What is the 3-Tier Architecture. Internet: www.tonymarston.net/php-mysql/3-tier-architecture.html, Oct. 14, 2012 [May. 21, 2016].

[2] Ahmed Fahmy. “The Unified Methodology for Application and Database Development UMADD (Using "Abb" Project: "abbCode" ASP.NET Code Library and "abbDB" RDBMS),” poster paper presented at the 2nd Int. Conf. Africa and Middle East Conference in Software Engineering AMECSE, Egypt, 2016.

[3] Matthew MacDonald, Adam Freeman, and Mario Szpuszta. *Pro ASP.NET 4 in C#2010*. 4th ed. United States of America: Apress. ISBN-13(pbk): 978-1-4302-2529-4, 2010, pp. 550–551.

AUTHOR BIOGRAPHY

Ahmed Fahmy is a software engineer and independent researcher interested in developing taking a responsibility of creating new software algorithms, methodologies and tools believing in changing the whole software industry making it simpler, easier achieving state of the art smart software to satisfy the rapidly on growing life needs.

Ahmed is the author and original programmer of abbCode project the tool which used in this proposal.

He can be reached at ahmedfahmyaee@yahoo.com.