CHAPTER 3

CRITICAL ANALYSIS

3.0 BACKGROUND

The use of CBD, Multi Tiered Design and HCI has become popular, especially in the development of desktop, client server application or web. Because of some limitation these technique is not popular in the domain of mobile application especially PPC; limited use of resources such as memory and processor. Therefore this project is directed towards defining component by using those techniques and mobile application, typically focusing on relatively small and statically mobile application which is PPC.

In this project we focus on CBD technique to develop the mobile application for PPC incorporate with the Multi Tiered Design and HCI concept for mobile application; in particular COM and our target platform is PPC.

This project’s goal was to create a MFAS which could be used for mobile application; PPC with CBD, Multi Tiered Design and HCI techniques, which for instance time specifications and resource usage. We implement this project by eMbedded Visual Basic 3.0 and eMbedded Visual C++ as the core programming language. MFAS created from the tool is fully compatible with PPC Systems.
Mobile system; PPC, have a lot of problems nowadays, the main reason is due to cost of hardware such as memory and processor. The costs here refer to limited hardware resources and flexibility of small devices PPC compare PC. To solve these problems it is necessary to develop an efficient mobile application system to overcome these problems. Now there are such mobile applications which are platform dependent, such systems are difficult to upgrade and customize. There is another drawback of these systems that they are not reusable. CBD technology provides the facilities to develop such a natural units for reuse.

Framework for CBD is a combination of different concept related to components and also among their relations to each other. Before identifying development platform for developing CBD for mobile application system; PPC, it is necessary to identify the whole transportation for different components. It is just because of a reason that component is used as the basic unit in CBD for PPC.

Now in the market different languages have been developed which use different techniques like Visual Basic (VB) from Microsoft uses Visual Basic Control which are known as VBX and Java beans are used in Java. To make the development of application easier there are different tools to share and distribute the application. But there are special underlying services which are used by all of those approaches by providing communication and coordination which is necessary in the development process of the application. The communication among the components is possible through plumbing which is the infrastructure of components. It is mentioned that to apply CBD, Multi Tiered design, HCI into PPC.
This chapter identifies and analyses the problem when CBD, Multi Tiered Design and HCI techniques and concept into PPC, proposed techniques, solution for those problems and the framework combination of CBD, Multi Tiered Design and HCI for MFAS.

3.1 INTRODUCTION

Computer are popular from centuries, but nowadays in 21st century they are getting much popular in every area of life especially in automobiles, robots, health care systems and other industries too. Due to this, the use of computers has been increased and also creating difficulties in system due to hardware and software. To overcome with software difficulties, new approach towards software has been introducing CBD and software developers are adopting this CBD approach.

The reason or the advantage to adopt this CBD approach is, by doing it will increase the efficiency and complete it in minimum time. It is also used to create the real-time applications for real-time systems. For last one decade, it has increased the usage of real time systems. In mobile application system both hardware and as well as software are merge in a way that give us better performance.

Before, we start the critical analysis and empirical evaluation of using software components in PPC. We have to do a lot of preparation. First of all we have to test a simple com component in embedded Visual C++ (eVC) for Windows CE or PPC and embedded Visual C++(eVC).
Secondly, we also have to make simple com client for testing the software in the PPC emulator by using the com library; API. So, in order to build this simple com component, first we have to create an ATL project and then we select server type as Dynamic Link Library (DLL). After this we include com class in this ATL project and later on, we include methods to this com class for implementing it with eVB as interface layer using.

In the last, after implementing the com class we will build and download the component. Now here we leave the emulator in running and return to visual studios for creating a simple com client. After creating a com client, we have to download and test the client.

CBD gives the fast technique for the assembly of reusable, consistent and programs. It is very useful and doing well in the development of PPC.

The practical part of the project is to investigate commercial or open source implementations of the models experimentally, by implementing component-based application with real time characteristics. Analysis and measurements should be made to determine the component models effect on timing predictability, time and memory overheads. This will require implementation of not or component based reference applications to obtain comparison measurements one possible application to implement the CBD, Multi Tiered Design, HCI methodologies and architecture to develop an efficient MFAS.

In this thesis we did the critical analysis and empirical evaluation on CBD, Multi Tiered Design and HCI for MFAS by using PPC.
3.2 COMPONENT BASED DEVELOPMENT (CBD)

In CBD, component is having many characteristic properties like it is unit of independent deployment, secondly it is a unit of third party composition and there is no observable state. All these properties having many implications like to deploy the component independently then in this regard it should be separated from its own environment and as well as other components. So it is known as unit of deployment. The thing worthwhile mentioning is that the most important and basic unit for the CBD is a component and on the other hand object is a basic entity for the Object Oriented Development (OOD). We can easily differentiate between objects and components due to deployment and as well as composition. It may contain one or more classes along with objects. Hence, it is not possible to deploy the classes in parts but on the other hand with respect to inheritance then components becomes dependent. (Clemens Szyperski, 2002)

A component is well defined by Clemens Szyperski which gives proper understanding about it. A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties. (Clemens Szyperski, 2002)
3.2.1 Component and Interfaces

First we have to understand that what is interface. The interface is a thing which is needed to connect the components. But basically interfaces are based on the set of operations which is defined by the clients. It is also known as the specification of its access point. (Clemens Szyperski, 2002)

So, there are mainly two different types of interfaces, one is imported and second one is exported interfaces. (Ivica Crnkovic, Magnus Larsson, 2002). As it is cleared through the name imported something which we import from outside so imported interfaces are those interfaces which is imported to utilize the services for the components, while on the other hand exported interfaces are those interfaces which gives the services or functionality to the situation taking it from component. (Clemens Szyperski, 2002)

3.2.2 Direct and Indirect Interfaces

It is possible to unite the concept about both direct and as well as indirect interfaces into one thing. But in the starting the major and basic concept was to use the static objects. We can model the procedural interfaces as an object interface with respect to component. The thing worthwhile mentioning is that many component implementations is only made because to have own interface which should look like object interfaces. It is possible with the pure Object-Oriented (OO) languages. By modeling the component interfaces in order to solve and take help for semantics as object interfaces. Hence, the most important thing in this regards is to remember the consequences of object interface. On the other hand, object interface defined an indirection which is called a method dispatch.
3.2.3 Component Development

In a component development all the process is in many aspects same with respect to system development. As far as system development is concerned it is based on the following things like gathering requirements, analyzing it. Hence, on the other hand component in way similar to system development like designing it, implementing, verification and finally delivered it. (Magnus Larsson, 2002). If the developers wants to build a new component then the only way is to reuse the other components and as well as they always used the procedure which is needed for the component evaluation with respect to system development in spite of having some difference. The thing worthwhile mentioning is that it is built to solve some problems and merge a part of that thing. It is possible to reuse them in many kinds of different products. There are following hurdles for these facts like we required a detailed component specification and to manage these specifications are not easy and we also need a lot of effort to develop reusable components. There is a process figure below which shows or describe about component evaluation and as well as system development. (Magnus Larsson, 2002).

![Component Development Process](image)

**Figure 3.1 CBD Process: (a) Component Evaluation and (b) System Development**

*(Magnus Larsson, 2002)*
3.3 INTRODUCTION TO COMPONENT OBJECT MODEL

It is a software architecture which allows the programmers and developer to build an application which is based on or can built from binary software components so this architecture is called as Component Object Model (COM).

COM is the core architecture that builds the base for higher level software services, as it is provided by Object Linking and Embedding called OLE. Object Linking and Embedding (OLE) service period has a range of different aspects like system functionality; inter application scripting, complex documents and software interactions. (Dr. Dobbs Journal, December 1994)

User can get different functionalities definitely from these services. On the other hand they distribute a basic prerequisite for a system that allows binary software components, derived from mixture of pre-existing customer’s components and components from diverse software vendors, to attach to and communicate with everyone in a sharp manner.

This system is completed by Component Object Model (COM), a software architecture that performs the following: (Dr. Dobbs Journal, December 1994)

1. Binary standard for component interoperability.
2. Language independent
3. Worked on different platforms like Microsoft Windows, Macintosh, UNIX and Apple systems.
4. Robustness evolution of component-based software’s and systems.
5. Extensible for programmers in a steady way.
6. Single programming model for components to communicate inside the similar process, and also transversely process and network limits.
7. Shared memory management between the components
8. Gives error and status reporting
9. Also allows dynamic loading and as well as unloading of components.

The thing worthwhile mentioning is that COM is a universal architecture for the component software. However Microsoft is using COM to deal with definite areas like controls, automation, storage and naming, compound documents and data transfer through which any programmer can take benefit of the structure and basic provided by COM. (Dr. Dobbs Journal, December 1994)

3.4 COMPONENT- BASED DEVELOPMENT PROBLEM

The most significant problems which CBD are like to designed a system in such a way so that binary executables can be able to interoperate, from different vendors and written by them on different times and different locations of the world.

Basic prerequisite of software component architecture is that it should be high quality performance. Before solving problem, we should give answers to the following things below: (Dr. Dobbs Journal, December 1994)

1. *Basic interoperability:*

How a programmer can build their own matchless binary components and also assured that how it will interoperate with other binary components which are built by other programmer. (Dr. Dobbs Journal, December 1994)
2. **Versioning:**

   Without getting all the system components which are already upgraded, so that one system component can also be upgraded. (Dr. Dobbs Journal, December 1994)

3. **Language Independence:**

   To assured that how the components which are written in different languages can communicate with each other. (Dr. Dobbs Journal, December 1994)

4. **Transparent cross-process interoperability:**

   By providing one programming model to the programmers, how can we assured that it will provide flexibility to them to write such a components that can be run in-process and as well as cross process and more over with cross network. (Dr. Dobbs Journal, December 1994)

### 3.5 COMPONENT BASED DEVELOPMENT FOR PPC

First, we have to understand that what is component, it use to design as a basic structure with respect to engineering and you will find a lot of definition of components according to component based software engineering but the best definition regarding component is defined by Szyperski (Clemens Szyperski, 2002, Ivica Crnkovic, 2005).

After reading this definition we can surely say that can do composition and as well as deployment at runtime. This definition is very popular with respect to embedded system and it is especially useful to separate component interface and implementation. The source code of a component is written in a high-level language only. The properties of the component is sum up by its own interface and visible. On the other hand, we can add the requirements and functional properties in the component interface provided by
embedded systems. The interfaces are also used as object interfaces for late binding with the help of polymorphism in general-purpose component technologies. There are a lot of reasons like limited resources and restriction for real-time due to which we can do component deployment and as well as composition at design time in spite of run time. It also allows the composition tools to create a monolithic firmware to get the better performance.

### 3.6 MFAS COMPONENT FOR PPC

During the development of MFAS for PPC, there were fewer challenges. But from the last few years there are three major problems has been occurred.

1. The major problem is the size and complexity of software.
2. Secondly, the diversity of products and its software is also increasing.
3. Development time should be reduced.

To design MFAS; a component-oriented approach is an ideal way to handle the diversity of software in PPC. The MFAS component model, used for PPC, allows late binding of reusable components with no additional overhead.

COM is the one of the best solution to develop PPC product families due to the combination of component model with an architectural description language. There are some requirements related to domain. (Rob van Ommering, Frank van der Linden, Jeff Kramer, Jeff Magee, 2000)
1. There a lot of connection between one or more components which are constant and only used at configuration time and to reduce the runtime, there is only one solution; the use of static binding.

2. We can upgrade the component with help of high-end products.

3. There is a special notation used for the required interfaces.

![Diagram](image)

**Figure 3.2 Taking the Binding Knowledge Out of the Components**

(Rob van Ommering, Frank van der Linden, Jeff Kramer, Jeff Magee, 2000)

From the figure will see that it shows the requirements for the required interfaces from the figure 3.3 (a) there is a two components; A and B1 and if A wants to access the component B1 then A needs to import and get the required information into A and however it is not possible to merge A with B2 as you can see in figure 3.3 (b). there is one possible solution for it and that is allow A to import component B and then configuration management system has to select among B1 and B2, but by doing this still we cannot have the permission to do so and create product 3 as it is shown in figure 3.3 (c). Hence, A is restricted to B1 and B2 according to the condition which is found during runtime. As a result, it takes binding information from the components. Hence we can say that A is set to be a required interface while B1 and B2 are responsible to provide an interface. The binding is available at the product level.
MFAS interface is consisting of tiny set of functions. We can get the functionalities via interface which is provided by the component and to complete this task, it needs functionality with respect to surroundings via interface.

3.7 PPC DEVELOPMENT TOOLS

According to Microsoft, it is divided into two different categories; the application developers and the platform developers. In windows, the integrated development environment is used by platform developer and it is known as platform builder but on the other hand real time applications runs with PPC and other than that it also provides the Embedded Visual tools.

We can create software development kit which is based on the operating system of the PPC by using platform builder which also allows us to write an application, which we can run on the target platform.

A software development kit is based on header, help files and library that are used by developers to write applications.
3.8 MFAS WITH CBD, MULTI TIER DESIGN AND HCI

In MFAS we create software components mainly with an intention of being reused in various software systems and reduce development time. Components are designed to interact with its environment through its well-defined interfaces but to encapsulate their implementation.

Development of MFAS with CBD, Multi Tier Design and HCI brings the potential for significant reduction in the development cost and time-to-market of enterprise software systems because we can assemble such systems from a set of reusable components rather than building them from scratch.

MFAS also increase the reliability of software systems - each reusable component undergoes several review and inspection stages in the course of its original development and previous uses, and CBD relies on explicitly defined architectures and interfaces.

MFAS improve the maintainability of enterprise software systems by allowing new, higher-quality components to replace old ones and enhancing the quality of enterprise software systems - application-domain experts develop components, and then we assemble those components into software systems.
The CBD life cycle differs from traditional way of developing software systems in many ways. The CBD phase includes new activities such as selection and creation of software architectures, as well as selection and customization of a set of software components. In addition, the implementation phase deals with the integration of a set of software components within appropriate software architectures. This requires developing wrappers that glue reusable components together to build the software system, rather than extensive coding to build a software system from scratch. Late integration of components developed by others eliminates the confidence usually drawn from integration testing in a traditional software engineering model. We architect and design extensibility into a system and all its parts otherwise the components in the resulting system will not be independently producible and deployable. Table 3.1 shows the comparison of conventional approach to CBD approach.

![Table 3.1: Comparison of Conventional Approach to CBD](image)

MFAS also adopted the Multi Tier Design and HCI concept for PPC applications. When functionality is divided into separate components, implementation becomes less important than published interfaces. As a user of a component, we need only focus on the desired functionality and how to access it through the interface.
We can focus on using the functionality in components rather than implementing the functionality themselves. Multi Tier Design concept clearly separates the responsibility of different layers and different component. We working on different components of the same project need only worry about the interfaces published because the components underneath those interfaces function independently.

With CBD and Multi Tier Design, bugs are easier to trace, and because other parts of the application would not be affected by bugs contained in one component, the developer maintaining the app can focus on keeping the interfaces consistent.

While single-file installations are fairly simple from a developer's perspective, most installations require multiple file installations anyway, possibly the inclusion of third-party controls. Also, installing the same functionality with many applications ultimately proves problematic.

3.9 PROBLEM STATEMENT

Most of the current system, end user maintains his or her record manually by writes down customer information and sales records on the book for every transaction and record. At the end of each day, end user return to their respective office and record in the customer and sale information into paper documentation and filling using traditional method at headquarters.

All this information is recorded by hand in their paper documentation, an exhausting and inefficient method. As increase of customer volume, there will be more
filling and documentation work need to be done. Information retrieving, sharing and decision making is very difficult.

This over-reliance on manual records has led to some significant issues during operation, including discrepancies between expected revenue and actual cash received at the end of each month, due to handwriting errors. Data accuracy is not there due to manual process. The head office could not maintain data accurate, real-time sales data, and the system was vulnerable to fraud. We found that the manual system was just not efficient.

3.10 MFAS CRITICAL ANALYSIS

Previous section we discussed about the existing manual system process flow, CBD, Multi Tier Design and HCI and existing system problem statement. Here; MFAS critical analysis we discuss about the comparison of existing manual system with MFAS.

Mainly MFAS migrate completely away from the traditional, manual manner of existing system. PPC application systems developed to support and replace the existing process flow. With the MFAS in place, user can synchronize their customer information and transaction data at the end of each day with the PC via the USB cradle, and have the information automatically transmitted back to the PC or notebook.

The greatest benefit of migrating to a Pocket PC solution has been the automation of the process flow. User now has an accurate view of the customer and transaction information at any point in time, and since data is entered electronically
using the PPC, errors are eliminated almost entirely, and have a body of business-critical data that is reliable, accurate, and easily analyzed to spot trends and help determine market strategy for the entire division.

The MFAS solution has practically eliminated human errors due to unrecognizable handwriting or typos. By simplifying data entry, MFAS solution has also significantly reduced the time taken for user to manually enter their customer and transaction information at the end of the working day, leaving more time for them to do what they were hired to do - generate sales and meet more customer.

MFAS helps user create new value for their business through the strategic use of their data available. MFAS also able to generate some report and result that able to help in decision making. Data retrieve and search is much easier compare to manual process.

PPC development without applying CBD, Multi Tier Design and HCI design concept, utilized 100% of the PPC device resources. This make the system slow in respond, easy hang and not allow multitasking.

Due to the processor, RAM and storage resource limitation in mobile device; PPC, we utilized the CBD, Multi Tier design and HCI concept which is famous in PC application development.

CBD allow of component reuse, by having component reuse feature, MFAS do not need to load many components to support different feature and system function. This will also reduce MFAS system development time due to component reusable.
Multi Tier Design separates the system into different type of layer which able to improve the security and data retrieves process. HCI design provides a standard of MFAS design. MFAS user friendly design, increase the system usability.

Overall MFAS with CBD, Multi Tier Design and HCI overcome the entire manual process problem and utilized PPC resources intelligently to speed up the operation process and system performance.

3.11 OVERALL SUMMARY OF THE CHAPTER

This chapter has outlined the issues and shortcomings regarding the current CBS in mobile application; Windows CE or PPC. The chapter includes the current problem faced by developer particularly for PPC, CBD in mobile programming, interaction and relationship between the components

The critical analysis of the CBS and the analysis will assist in the requirements elicitation process in Chapter 5, as well as the design and implementation of the CBS, in Chapter 6 and Chapter 7 respectively.