INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Coordination Based Design For Tailorable Business Software: A Computerized Maintenance Management System Example

© By Charlie Cox

Table of Contents

Acknowledgments

I would like to take this opportunity to express my appreciation to those who made it possible for me to complete this thesis. First and foremost, Donna, deserves my heartfelt love and thanks for supporting this effort, never complaining once about the tremendous amount of time I spent secluded in the University of Ottawa Library, or in the bedroom, or basement, etc. "plugging away" at this work - time that could have been, otherwise, devoted to my family. Secondly, Moshe Krieger, my research advisor, and friend, has been instrumental in guiding and coaching me to completion of this thesis. He has demonstrated to me, over the years that I have been pursuing my post graduate degree, a remarkable talent and passion for helping his students, like me, to complete their thesis work, and to strive for excellence in doing so. Thank you, Moshe, most sincerely. And, all the best to you in your retirement (I hope you have forgiven your daughter for the wood carving set). I would also like to thank Muddesir Siddiqui for his assistance in teaching me how to perform the rudimentary tasks that I needed to know to code my example Computerized Maintenance Management System. He selflessly helped me on several occasions as well, to overcome errors and bugs in my code. With his help (and friendship) I was able to acquire the basic programming skills required to conclude this important part of the thesis. Lastly, my employer, Lockheed Martin Canada, has provided me with funding assistance to cover (although only partially) the tuition costs, and I would be remiss not to express my thanks for this support.

Abstract

This thesis investigates the applicability of "Coordination Based Design" (CBD) as a software design methodology for the development of large commercial tailorable software systems, with the use of components. CBD provides an intuitive engineering approach to developing software and effectively facilitates its modification, growth and tailoring. More specifically, for the design of large systems, such as a Computerized Maintenance Management System (CMMS), it is argued that CBD is a practicable means of accomplishing this. The CMMS was chosen as an example of a commercial application to discuss in relation to CBD, as it is a "good fit", due to the many challenges in the design, tailoring and upkeep of this class of applications. The fundamentals of CBD, which include the following key principles, are discussed in further detail in the body of the thesis.

- CBD provides a means of performing requirements analysis by identifying the main system inputs, termed "Events" and associated with each event, the system determines, initiates and completes the appropriate "Response(s)";

- Through the CBD methodology, the system is designed and implemented in terms of:
    a. "Coordinators" which properly sequence and initiate units of work to be performed, and;
    b. "Activity Executors" (modules) which implement the units of work, and which respond exclusively to a Coordinator for direction;

- CBD is particularly well suited to support system "tailoring", both in terms of modifying the Coordinator to initiate different or additional responses and the activity executors which allow easy alteration or addition of units of work. The principle idea of tailoring is that a "core" system (in a particular application domain) is developed which contains the basic necessary functionality. This system can then be tailored, by adding to or modifying the core functionality to suit a specific customer need.

- CBD can be used advantageously with components of different specialized or Commercial Off The Shelf (COTS) packages to implement a "Component Based Design" system.

To define a meaningful CMMS example, a survey was done, by selecting a set of various CMMS packages and then determining the more typical types of functionality available. This information was then used as a basis for establishing a reasonable set of functional requirements on which to base the design. This indicates that, in the case of complex systems, regardless of the business environment, a similar survey can be advantageously used to quickly and confidently identify the major domain specific system events and responses. Also examined, in this thesis, are some of the technical details of how CMMSs are implemented, e.g. utilization, hardware platforms, programming languages, and the integration of CMMSs with other types of COTS software.

Acronyms and Abbreviations

API	Application Programming Interface
CBD	Coordination Based Design
CMMS	Computerized Maintenance Management System
COTS	Commercial Off the Shelf
DAO	Data Access Object
DB	Database
Eqpt.	Equipment
Inv.	Inventory
LAN	Local Area Network
Maint.	Maintenance
MSL	Minimum Stock Level
OOT	Object Oriented Technology
PAL	Process Activity Language
PdM	Predictive Maintenance
PDR	Preliminary Design Review
PM	Preventive Maintenance
PO	Purchase Order
PR	Purchase Request
Qty.	Quantity
RODB	Restricted Object Based Design
Sys	System
UI	User Interface
WAN	Wide Area Network
WO	Work Order

1. Introduction

1.1 Motivation

To-date, the CBD methodology has existed primarily as an academic vehicle for post graduate research at the University of Ottawa and, has been the subject of several papers and thesis reports; in this regard, the merits of CBD have been comprehensively discussed in terms of several different types of software systems and application domains, such as real time systems, shrink wrapped software and simulators, etc. In this thesis, the CBD methodology, its characteristics and advantages are considered for the design of various large domain specific systems, however, particular attention is paid to its applicability as a means of achieving software tailorability and component-based design.

In the early years of software development, tailorability often took the form of version control, i.e. providing or releasing different versions of software to different classes of users. Now, software reuse and tailorability are often considered as being closely related, but while software reuse is a general practice to reduce the software development effort, we believe that tailoring is essentially a different concept from reuse, and pertains more directly to the development of basic core functionality (domain specific units of work) which is then tailored or "tuned" to suit particular user requirements, through the selection and integration of additional functionality.

Software tailoring is not addressed in journals and literature to the same extent as reuse, nor is it widely used in industry for the creation and deployment of operational software systems / tools in the field. Because there are commonly many types of industry and business which continue to use large software systems with similar capabilities, but differing individual characteristics, the ability to tailor software will be a prominent way to customize these systems in the future. And, for software houses which develop and market domain specific software, tailoring offers significant advantages, by allowing early customer input and ultimate satisfaction through products which closely fit their needs. To this end, the theme of using CBD as a practical means of tailoring software is discussed in the various chapters of this document.

The CMMS was chosen as a real world example of the type of system to discuss in relation to tailoring and the CBD methodology, for the following two principle reasons:

a. Newly introduced CMMSs commonly require customization (tailoring) to fit the individual or unique requirements of the sites (e.g. plants or factories, etc.) in which they are installed, and;

b. the author's several years of experience in the equipment maintenance environment.

Today, there are many commercially available CMMSs on the market, which fit a wide range of needs and work environments, e.g. GP MaTe [GP14, TM15] and MP2 Professional 5.0 [MP 11]. Basically, for any environment (e.g. industrial or military) in which maintenance is performed (by people), the employment of a CMMS will normally offer the benefit of helping to ensure that the work is initiated and controlled systematically and that it is performed in accordance with maintenance standards and at the required intervals. Usually, for larger maintenance environments, greater benefits and savings can be realized by using CMMSs. And, as the market for these applications is growing rapidly, CMMSs are continuously

evolving and being upgraded by the system developers, to meet changing needs, and in order to remain competitive.

Providing CMMSs that can be easily "made to fit" is a constant challenge to companies that develop these products, and there is often much attention paid, in trade magazines, maintenance journals and Web-based literature to this need. There are numerous examples of CMMS systems purchased by customers which have quickly fallen into disuse because requirements were not defined adequately beforehand or the purchased system could not be made to fit the specific needs of the customer. Supporting this is the fact that attributes of CMMSs, which are commercially successful, include modularity, scaleability configurability, and notably, the ease with which it can be tailored to the specific needs of a customer.

The remainder of this chapter introduces the organization and contents of the report, and the main subject areas of the thesis.

1.1.1 The Evolution of this Thesis - A Brief Description

While pursuing the main objective of arguing the merits of the CBD software development methodology, a number of side issues needed to be investigated, of which some were anticipated and planned, and some not. The main challenge, in this regard, was how best to accomplish this aim, considering the authors experience and knowledge and some of the CBD-related work done to-date, some of which is discussed in the thesis.

Applying CBD to the design of large, complex systems was the first scenario looked at, but this theme seemed too broad at the time and wasn't an area in which the author had any particular experience. Military and civilian maintenance environments, however is an area in which the author had spent several years - unfortunately, this did not include any previous experience with CMMSs. However, due to the importance of ensuring that the discussion is not just academic, the generalized CMMS was chosen, as a real world example, to investigate. At first, it was thought that CBD could be used to design and implement a "simple" CMMS from scratch, but, after an initial investigation, it was quickly realized that CMMSs are not inherently simple, and to create one without benefit of using or reusing software components in some way, would not be a practical way to proceed.

The task, then, was to find a workable way to develop a CMMS example using some sort of previously created work and to build upon it. Happily, another post-graduate student (Robert Au Yang) had coincidentally completed a project which successfully demonstrated, through an intuitive yet practical example, that CBD could be applied as a means for creating a "component-based" design using COTS software. It was then decided to use Au Yang's project [AY 28] as this starting point, to modify and build upon, (by using his initial design and code) to develop an example CMMS, for demonstration purposes. In so doing, this supported the position that CBD provides a viable means of designing and then tailoring a large software system, in this case, built using COTS components.

After starting out to build the example CMMS in this way, using Au Yang's code as an initial building block, it was discovered that the system design could, indeed, be easily modified to change or add functionality, create new or combined responses and implement new modules. For this reason, it was decided to investigate and promote CBD in this thesis, from the perspective of its software tailorability potential.

1.2 Basic Concepts

To provide proper context for the thesis, we will first present, in a little more detail, some of the basic concepts referred to previously. These topics include good software design, CBD, CMMSs and tailorability.

1.2.1 Challenges/Requirements of Good Software Design

1.2.1.1 What Constitutes Good Software

It is important to review some of the attributes of well designed software before pursuing CBD further, as in this thesis, it is argued that CBD is a methodology which provides the framework to design good quality software.

There are several generally accepted characteristics of what constitute good software, which are discussed below. Arguably, the most important quality to consider is functionality - does the system, in fact, perform the required work? Additionally, it is highly important that software be simple, flexible (easily modifiable to accommodate changing requirements) and implementable using current technology. It is also important to remember that the different measures of quality may vary in importance depending on the type of application being built. To address specific application needs, there is the important requirement for affordable, well engineered products targeted to these specific situations, and available in the shortest possible time. Custom designing software is not the only way to develop target products, of course, and very often today, designs are re-used where possible.

There are, generally, two major types of software in use today - customized software which is typically one-of-a-kind and expensive and Commercial Off the Shelf (COTS) software which are more ubiquitous, less expensive, however, may not provide the best fit to user requirements.

Software products also have static characteristics (e.g. the structure of the program) and, dynamic characteristics (e.g. the run-time), so the assessment of software quality should address both. For basic data processing systems, generally the more important characteristics are the static ones, and one assesses the quality of structural complexity (e.g. numerous branches, "Ifs" and "Whiles"..). For event-driven and embedded systems, efficiency and the real-time characteristics are more important. And for an aircraft system, where the consequences of error are severe, reliability and safety are the prominent factors.

What are considered to be the most important aspects of software quality also varies depending on the perspective taken, e.g. to a software user, supplier or developer, the various attributes of software quality may take on greater or less significance. To a user the salient criteria generally include [Gen2]:

- Functionality;
- Coexistence and interoperability;
- Ease of use;
- Lack of surprises;
- Adequate and usable documentation, and;
- Ease of installation and update / cut over, including data.

From the perspective of the software developer, the following considerations could be considered more important [Gen2]:

- Ease of learning for maintainers;
- Adaptability
- Structure for timeliness and cost-effective implementation;
- Exception handling;
- Testability and measurability;
- Analyzability and predictability;
- Adequacy of scaffolding and support tools;
- Professionalism of programming;
- Efficiency and performance;
- Ability to convince third parties of correctness, conformance, etc.

1.2.1.2 The "ILITIES" [DB3]

Another characterization of software quality is contained in a grouping sometimes casually referred to as the "ilities". Obviously, many of these attributes of quality software are interdependent. These "ilities" are usually defined to consist of:

Reliability

Reliability is a dynamic measure and is generally described in terms of:

- Completeness, i.e., accommodating different types of events and system states;
- Consistent behavior;
- Robustness;
- How likely is complete system failure, in the face of component failure or conflict?
- Whether a system can easily achieve graceful shutdown and recovery.

A reliable software application carries out the expected task without variation and when required. Note, that a system can be correct, but yet unreliable. A system can also work well statically, however, when run on-line, poorer results could occur. One way of implementing reliability is through redundancy, where a second identical component of the system can take over, in the case of failure of the first. The other usual way of ensuring reliability is through simplicity, which is not always possible.

Efficiency

Efficiency involves such considerations as whether the system makes efficient use of its resources, such as memory and processor time. In the early days of software development, systems were created at the very lowest levels but while systems implemented in machine code or assembler are relatively efficient, they are not particularly maintainable. Furthermore, for many systems developed today, with high performance processors and cheap memory, efficiency is sometimes of secondary importance. It is interesting to note that while many software designers argue that efficiency is realized through good algorithms, it is not easy to create good algorithms using machine language.

Maintainability

The factor of maintainability is more consequential with large systems, where longevity is important, as initial procurement costs are typically quite high. Separation of concerns in program design generally allows software to be more easily maintained, and whether the architecture of the code allows the maintainer to understand pieces of the program without

requiring an understanding of the entire package, is a major consideration. Also, understandability, i.e., readability of code is highly important to a product's maintainability, and accurate and thorough traceability from the design to the code is an indispensable requirement for maintainability. To this end, CBD, which is discussed in further detail in Chapter 2, provides a modular and traceable architecture, very conducive to software maintenance.

Usability

The "user-friendliness" of the software product is a measure of its quality, as well - and a very important one. If a program is too difficult or frustrating to use, it will likely fall quickly into disuse. Software, therefore, should be easy to learn, intuitive, concise and be compatible with user habits (e.g. does not impose additional work on the user). If a system is not user friendly, the other "ilities" will then make little difference.

Simplicity

Factors affecting a program's simplicity include control flow complexity (number of execution paths), and the complexity of the information itself.

Modularity

By virtue of the often used term "software module", its importance is significant. Modularity relates to the separation of concerns in a design. A designer should group functions within modules to minimize interaction between them. If modules each concern one feature, this facilitates ease of comprehension and maintenance. Modules should be easy to replace and be reusable as well. Code structure should readily support upgrades, evolution and the implementation of functional changes with minimal operational impact.

Most common measures of modularity can be described in terms of "coupling" and "cohesion". Coupling represents inter module connectivity. In his book "Software Design", David Budgen describes the forms of module coupling as shown in Table 1-1, below.

Table 1-1 - Forms of Module Coupling[DB3]

Form	Features	Desirability
Data Coupling	Modules communicate by parameters or data items that have no control element	High
Stamp Coupling	Modules make use of a common data type	Moderate
Control Coupling (Activating)	Modules transfer control in a structured manner (e.g. procedure call)	Necessary
Control Coupling (Coordinating)	Module A passes a parameter to Module B that is used by B to determine the actions of B (a flag)	Undesirable
Common-environment Coupling	Modules contain references to some common data area - any change to the format of the block requires that all of the modules using it must also be modified.	Undesirable

Cohesion is the measure of whether the components of a module can be considered to be functionally related. (Ideally, they should all have a common purpose). Table 1-2 contains a list of Budgen's description of the principle forms of cohesion.

Table 1-2 - Forms of Module Cohesion[DB3]

Form	Features	Desirability
Functional	All elements contribute to the execution of a single problem task.	High
Sequential	Outputs from one element of a module are the inputs to another module element.	Quite High
Communicational	All elements concerned with operations that use the same input and output data	Fairly
Procedural	Module elements are related by the order in which their operations must occur	Not very
Temporal	module element operations are related by the time at which they occur, usually linked by some event such as system initialization.	Not very
Logical	Elements perform operations that are logically similar, but not which involve difficult actions internally.	Definitely not
Coincidental	The elements are not linked by any conceptual link.	Definitely not.

Information-Hiding
Information hiding is related to modularity, i.e., information regarding structures and interfaces should not be "visible" outside the module. In this way, the integrity of the module is maintained, as other modules cannot directly access internal module parts.

1.2.1.3 The "ilities" as Applied to Software Tools

A large number of software applications can be classified as "tools" rather than "products". The CMMS is a prime example of a software tool, as they provide maintenance managers, engineers, and technicians, etc. with the capability to carry out their duties in an efficient and cost effective manner. At the very least, a software tool is similar to any other kind of application, in that, whether or not it is judged satisfactory or acceptable in the eyes of the users, depends largely on it having the required functionality, being dependable, delivered on-time and within budget

Even though most discussion concerning software quality is generally applicable to software products of all kinds, we emphasize that these above-mentioned "ilities" can take on even more importance in the case of software tools, as in many cases, they have been integrated deeply into the fabric of different work environments such as industry, the military and business.

1.2.2 An Overview of Coordination Based Design

High quality in software can only be realized if the system design is simple, well organized and easily understood. To accomplish this, a systematic design process is necessary, which requires the designer to carry out the following general steps:

- Formulate a requirements specification;
- Develop the design;
- Implement the design;
- Test, and;
- Deploy.

In CBD, requirements analysis and specification is accomplished by defining events and event responses. These events can be occurrences external to the system which the system senses (indirectly) or direct inputs to the system (e.g. messages from another

system or keyboard input). Following receipt or detection of the event, the CBD system then determines the appropriate response or output and implements it.

One of the main measures of software quality is simplicity in system design and complete ease of use, and methodologies that minimize complexity will usually meet with greater success and be more widely used. The CBD methodology espouses an architecture and design technique which is inherently simple to understand and use. In this method, one really focuses on the work that has to be done, and its execution. To this end, one of the principle pillars of CBD is that its architecture and functional organization force the separation of the two main concerns of system response, that is the ***execution*** of software code from the management (or ***coordination***) of the execution. By focusing on a design strategy where these two essential ingredients of working software are distinct from each other, a certain measure of simplicity and effectiveness is achieved. Figure 1-2, below, illustrates the CBD architecture.

In CBD, the primary building blocks of functionality are called "activities", which can be otherwise thought of as discrete units of work that the system has to perform. The ease with which these activities can be used, reused and combined in different ways to create and "tailor" the design is the fundamental aspect of CBD tailorability. A basic or core set of responses is the starting point, from which one tailors the final system by adding/deleting responses, if necessary, and by integrating existing simple responses into more complex "combined" responses, to fit specific user needs. Further discussion, related to activities and their usefulness in designing and building functions of varying complexity to suit specific customer requirements, is contained in Chapter 2.

Another advantage of CBD is that it is generic, to the extent that implementation of the design is not tied to any particular programming language(s). As well, activity executors or modules can be created using whatever different COTS packages the designer requires or prefers. And, because Coordinator(s) and execution modules are kept distinct, a CBD design can be easily tested. This is further supported by Sylvain Lemire who states, in his thesis entitled "Multiactivity as a Restricted Object Based Design Approach" [SL35], - "Each execution module can be tested separately by exercising each of the service calls for the various input conditions" and "For the modifications, one only needs to test the new or modified execution modules or response managers".

System deployment is also possible incrementally, which is especially important for effective customer buy-in and use. A multi-step approach, like this, provides the customer/user with the cost savings and freedom to acquire only the desired features, and then to grow or upgrade the tool as the needs arise. See Chapter 2 for a more in-depth description of CBD.

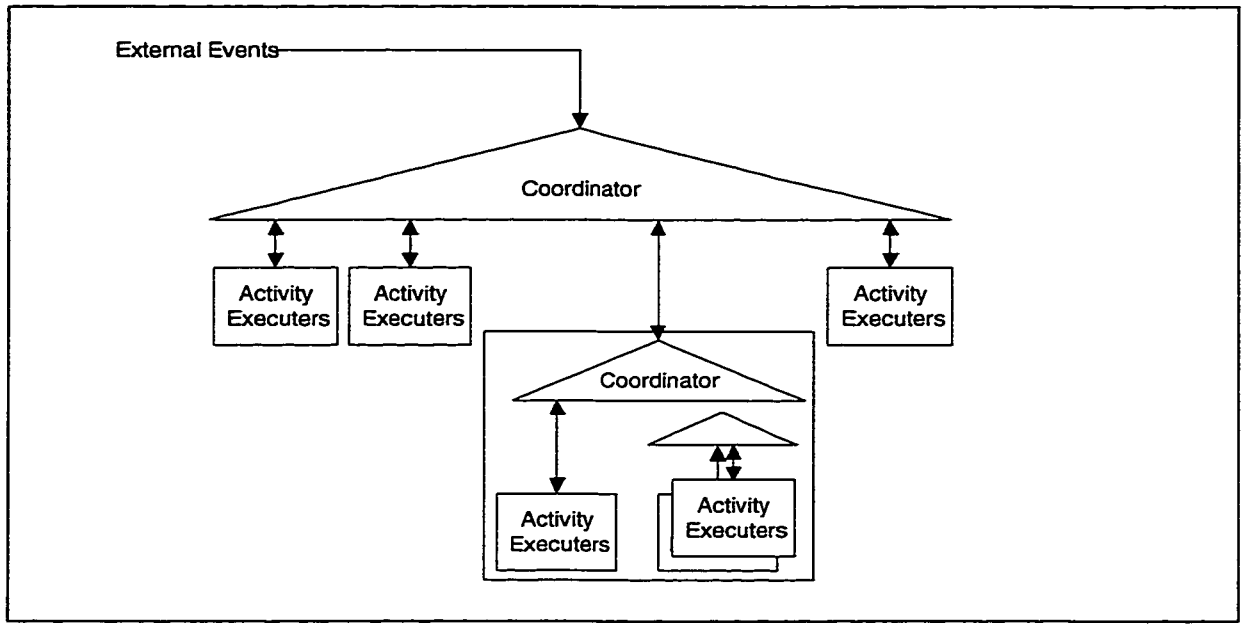


Figure 1-2 - Architecture of a System Based on Coordination Based Design [MK1]

1.2.3 What is a Computerized Maintenance Management System

A CMMS is an integrated software tool, designed for workload management in factories, plants and other facilities. The capabilities often provided by CMMSs include:

- Preventive and corrective maintenance;
- Work order generation;
- Purchasing;
- Inventory, and;
- Many other related functional capabilities.

Maintenance managers control and coordinate maintenance activities using information captured from CMMS, and can automatically monitor the state or health of equipment, as well. Although the equipment, and information related to the equipment being maintained is of principle importance to CMMSs, there are many other sources and types of information concerning people and finances, etc. which are needed by maintenance managers to carry out their responsibilities. An important feature of CMMSs is to provide the capability to experiment with potential solutions to problems using different combinations of resources (for example, equipment / personnel / money).

CMMSs allow managers to control maintenance activities, to facilitate decreased equipment downtime, increased worker utilization and increased utilization of other resources. There are numerous other touted advantages of CMMSs, including improved maintenance quality, improved safety, improved maintenance costs and labor records.

The complexity of CMMSs can vary considerably. Low end products only include basic functionality such as work order automation and equipment history databases. High end packages can offer many different complex modules which implement asset care management, predictive maintenance and many other capabilities. Costs can range from hundreds to hundreds of thousands of dollars.

Chapter 3 contains a more detailed discussion of CMMS's, including what factors are understood to describe CMMS quality, a survey of some commercial CMMSs and the types of functionality commonly available in these systems today.

1.2.4 Custom Versus Commercial Off the Shelf Software

The vast majority of big businesses and industries depend on complex software systems for their proper operation (e.g., support, engineering, management, finance, etc.). Generally, there are two choices available in selecting and implementing the required system, custom designed software and Commercial Off the Shelf (COTS). Typically, the earlier software systems were "one of a kind", and specialized to serve individual purposes. Although custom designed systems will likely provide a better match in terms of satisfying the business and industry requirements, they are ordinarily only affordable to organizations with "deep pockets", and require a lengthy period of development, integration and test before being ready for operational use. Additionally, custom developed systems might require specialized expertise to maintain and operate, and are often plagued by development cost overrun .

COTS applications, on the other hand, have certainly come to be widely used today. They are more readily available, and less expensive than custom designed systems, nevertheless, a COTS system has the disadvantage that all required functionality will not be provided and / or that undesired (and costly) functionality will not be optional. Moreover, COTS applications are now marketed together in "suites" (e.g. Microsoft Office), potentially making this problem even more significant. Also, businesses often have to modify their "modus operendi" to suit the COTS products available. When employing diverse COTS components, other considerations include: "Maintenance and release schedules are under the control of the COTS software vendor" [V36] and "You are one of the many customers and your voice may be lost in the crowd when requesting changes to the product." [V36]

1.2.5 The Need for Tailorable Software

The term "tailoring" is often used interchangeably in technical literature to describe the act of adapting or modifying software. However, in this thesis, we will use a more restrictive interpretation of the term, (i.e. task or interaction based tailorability), where one starts with a core of basic or common functional components which are then built upon to provide the user with a system that meets (or is tailored to) their specific requirements. In this section, we will first briefly discuss what modification or customization of software means, in general. This is followed by a discussion of software tailorability, as the term is applied in the thesis.

1.2.5.1 Adapting Software - A General Discussion

Although it is often desirable to freeze system requirements during the design process, in many complex operational in-service systems, these requirements are typically difficult to capture perfectly on the first pass and may significantly evolve to adapt to changing circumstances. In these cases, the process of requirements definition will require iteration. As a result, methodologies for designing reusable, adaptable, customizable software, and associated technologies are in high demand. Today, a level of computer hardware maturity or "computing power" has been reached, where it is not usually considered to be the major cost driver. Indeed, the ability to design large, user specific software systems is typically, a far more arduous challenge. And, to design them in such a way that they can evolve easily with changing requirements is of paramount importance.

There are several reasons for the modification of software; as an example, it may be necessary to change or further develop a software application during use, to adapt it to

needs that were not accounted for, or understood originally. Kinds of software modification commonly practiced generally include customization, integration, and extension, each of which address different system considerations.

There can be two potential conflicting goals of software adaptability, i.e., the need for maintaining "architectural discipline" vs. the tendency to introduce quick fixes, i.e. change the software, at lower levels. However, software which is changed hurriedly and on an ad hoc basis, without much concern for the integrity of its architecture, will soon become unmanageable. Successful software modification techniques, therefore, must provide a sufficiently robust architecture that is not put at risk by frequent alterations to this low level functionality.

Complex systems, additionally, require well-defined components, i.e. software architectures, and component interactions. New techniques which involve "Software Architecting" are starting to provide higher levels of abstraction for adaptability. However, constantly changing requirements necessitate easily changeable components and alterable ways of interaction. Some programming languages include useful abstractions for adaptation. However, most only support component adaptation, e.g. a form of controlled polymorphism (i.e. universal, ad hoc) through subtyping, genericity, and overloading.

In his paper Tailorable Groupware: Issues, Methods, and Architectures, [TG6] Volker Wolf argues that "...it is important to allow ordinary users to tailor a system's functionality because the demands for changing a system are frequent and it is costly to develop new systems from scratch." Other issues associated with software modifiability include how to design the functionality in a way that anticipates future needs and how to design a user interface that allows users to alter a system's functionality. (Note that Mr. Wolf's use of the term "tailor" does not correspond exactly to how the word is defined in this thesis.)

Several companies specialize in the adaptation of software, including CableData, Power 2000 Inc. and there are numerous international workshops addressing the issues of tailorability, including:

- Adaptable and Adaptive Software Workshop at OOPSLA '95 [AA5], and;
- GROUP'97 - Tailorable Groupware: Issues, Methods and Architectures [TG6].

At Oklahoma State University, a major research program [PL7] is underway, which addresses adaptable software. The principle objective of this research project is to use partial evaluation techniques to automatically customize adaptable software written in modern imperative programming languages. Partial evaluation is an automatic tool for program specialization, customization, and optimization. It, therefore, provides the foundation for a paradigm of adaptable software construction and instantiation. Partial evaluation technology will be modified to handle modern language features, and through experiments with large real-world software systems, methodologies can be developed for constructing and customizing adaptable software.

Following is a list of some other current methodologies or areas of work concerning the design / development of modifiable software:

1. The Demeter Method is a software design and development method for deriving object-oriented software from informal specifications, such as use cases and scenarios. The software is not described directly at the object-oriented level, rather at the adaptive object-oriented level with the double benefit that the software gets both shorter and

considerably more flexible. This software is called adaptive because it adjusts automatically to a large number of context changes. [AOOS30].

2. The objective of the Twente Research and Education and Software Engineering (TRESE) project is to perform research on compositional object technology. This research includes the development of object-oriented frameworks, models, methodologies and supporting tools for creating and maintaining adaptable software [AOOS30].

3. The Software Technology for Adaptable, Reliable Systems program (STARS) is sponsored by the Advanced Research Projects Agency (ARPA). The goal of STARS is to increase software productivity, reliability, and quality by integrating support for modern software development processes and reuse concepts within software engineering environment technology [AOOS30].

In certain cases, it can be practical, and efficient to build up a system using COTS components. Perhaps, one obvious advantage is that COTS provides ready to use capability that only needs to be integrated into the system in a practical way. Integrating the desired features of different COTS applications into a specific customer solution has great potential merit. In his article "Evolving Toward Systems Integration", Tse L. Wang states that to address the limitations of two basic categories of software products available (i.e. custom and shrink wrapped), systems integration is a way to improve product development - "A systems integrator creates a product by integrating existing products from diverse vendors in a coherent whole, tying the components together with one or more integration technologies.....The software that forms the glue should be built in high-level environments that support rapid, incremental development and deployment" [TLW4].

1.2.5.2 Tailoring - A Building Block Approach to System Development

Tailored software offers a "compromise" between COTS and custom software solutions, and provides the potential to take advantage of the benefits of both. To achieve a tailorable software system, (regardless of the application environment) one needs to start with a certain "baseline" of system domain specific functionality which can be expanded (tailored) to satisfy a specific customer's needs. Essentially a set of core or common "building block" components (i.e. activities and atomic responses) would be specified, to which other optional components could be added, as needed. Once this high level design is completed, these core and optional components could also be easily "tuned" or modified to achieve the best customer fit. The ease with which CBD allows this to be accomplished is especially noteworthy, and is primarily due to the strict independence of the activities from each other (i.e., they typically only respond to and communicate with the Coordinator).

Another important advantage that the tailoring strategy offers, includes the opportunity for software development houses to maintain libraries of "ready-to-use" activities and atomic responses which can be quickly and easily brought together to create more complex system responses. And additionally, beyond rapid development, a significant benefit of a tailorable system is its ability to support early prototyping and provide continuous user validation. In this way, applications are built up quickly, which facilitates early customer exposure and feedback.

1.3 Thesis Organization and Layout

This thesis consists of two parts - the report and the software application developed as an example of the type of tailorable system that can be designed and implemented using CBD. The report itself, is organized in the following manner:

- Chapter 1 - Introduction The purpose of this Introduction was firstly, to define the objective of the thesis, provide some background information and history concerning the evolution of the work carried out in pursuit of this objective and to preview the principle themes presented (i.e. CBD, CMMSs, the principles of good software design, Custom vs. COTS and software tailorability);

- Chapter 2 - Coordination Based Design For Large Tailorable Component Based Systems - In this chapter, a general review of the CBD methodology and architecture is provided. A description of the main principles of CBD is furnished, including, the concept of separating out in the design, the coordination of work from its actual execution. This chapter emphasizes how software tailorability is achieved through the CBD technique, together with the use of modules developed from different COTS applications.

- Chapter 3 - Computerized Maintenance Management Systems - An Overview and Market Survey - As the CMMS was the type of domain specific software tool chosen to demonstrate the merits and applicability of CBD, it was deemed necessary to introduce CMMSs at this point in the thesis. A survey of selected commercially available CMMSs is presented in Chapter 3, which also contains a relatively detailed examination of the different capabilities that these systems offer. In part, the purpose of this chapter was also to develop an information base upon which a reasonable requirement specification could be developed for the CBD design example contained in Chapter 4.

- Chapter 4 - Coordination Based Design of a Sample CMMS - Chapter 4 contains an example design of a CMMS (with limited functionality) using the CBD process. This includes the definition of a set of system requirements and follows through to include the detailed design.

- Chapter 5 - System Implementation - This chapter describes how the CMMS design, contained in Chapter 4, was implemented. It provides a description of the development platform used, how the user interface, Coordinator and execution modules were created and what functionality was included. Lastly, the integration and test activities performed are briefly covered.

- Chapter 6 - Concluding Material - Lastly, as one of the most important and meaningful measures of success is quality, this final chapter re-examines the CBD methodology from the perspective of its significant potential to produce high quality software. Included here, as well, is a discussion of the challenges and difficulties experienced while completing this thesis, together with some of the skills and knowledge gained in the process. The chapter closes with some thoughts regarding examples of work or research that would be advantageous to pursue as follow-on work.

- Bibliography

- Annex A - CMMS Functionality Tables - Detailed tabular information describing the functionality of the different CMMS's reviewed.

- Annex B - The Process Activity Language - A description of the Process Activity Language (PAL) which can be used in CBD in a formal manner, to express system requirements.

- Annex C - Coordinator Code - A printout of the Visual C++ code used to develop the CMMS example Coordinator.

- Annex D - CMMS Database Entity Relationship Diagram - showing the configuration of the CMMS example database, developed in MS Access.

- Annex E - Activity Executor Code - a printout of the Visual Basic code used to create the example CMMS activity executors (execution modules).

- Annex F - Local User Interface Code - A printout of the Visual C++ code used to develop the user interface (main screen and dialog boxes).

1.4 Summary

In this chapter we have introduced the basic subject matter to be addressed in this thesis and the manner in which it is presented. Brief overviews of CBD and CMMSs were included as part of this introductory chapter, to provide the reader with a preview of these technologies in advance of the more detailed sections which follow. It was also important to present the brief discussion of software quality in order to show that the CBD methodology does provide a design environment conducive to producing high quality software. In addition, tailoring, in concert with the CBD development environment was introduced because it offers an economical and expedient way of producing software and it was through tailoring a previously developed software system, (using CBD) that the CMMS example was developed for this thesis.

2. Coordination Based Design For Large, Tailorable, Software Systems

2.1 The Coordination Based Design Methodology

As introduced in Section 1.2.1, and as it will be further discussed in this chapter, CBD is a complete software development methodology, and offers many advantages including the following principle capabilities, which will be further expanded in the following sections:

- A comprehensive means of accomplishing requirements analysis and specification, through the definition of events and responses;
- A unique and effective system design architecture, in which the coordination and execution of the work is separated, with design components being directly traceable back to the requirements model;
- The requirements analysis and subsequent system design are able to be carried out using the same paradigm (i.e. CBD), thereby fostering effective traceability and compliance, and;
- Freedom to implement the design in a chosen programming environment.

The CBD process (as illustrated in Figure 2-1) is soundly based on accepted, well tested system engineering principles involving:

1. Requirements Analysis and Definition
2. System Specification and Design, and;
3. Implementation.

Sections 2.1.1 to 2.1.4 provide a detailed explanation of these design steps.

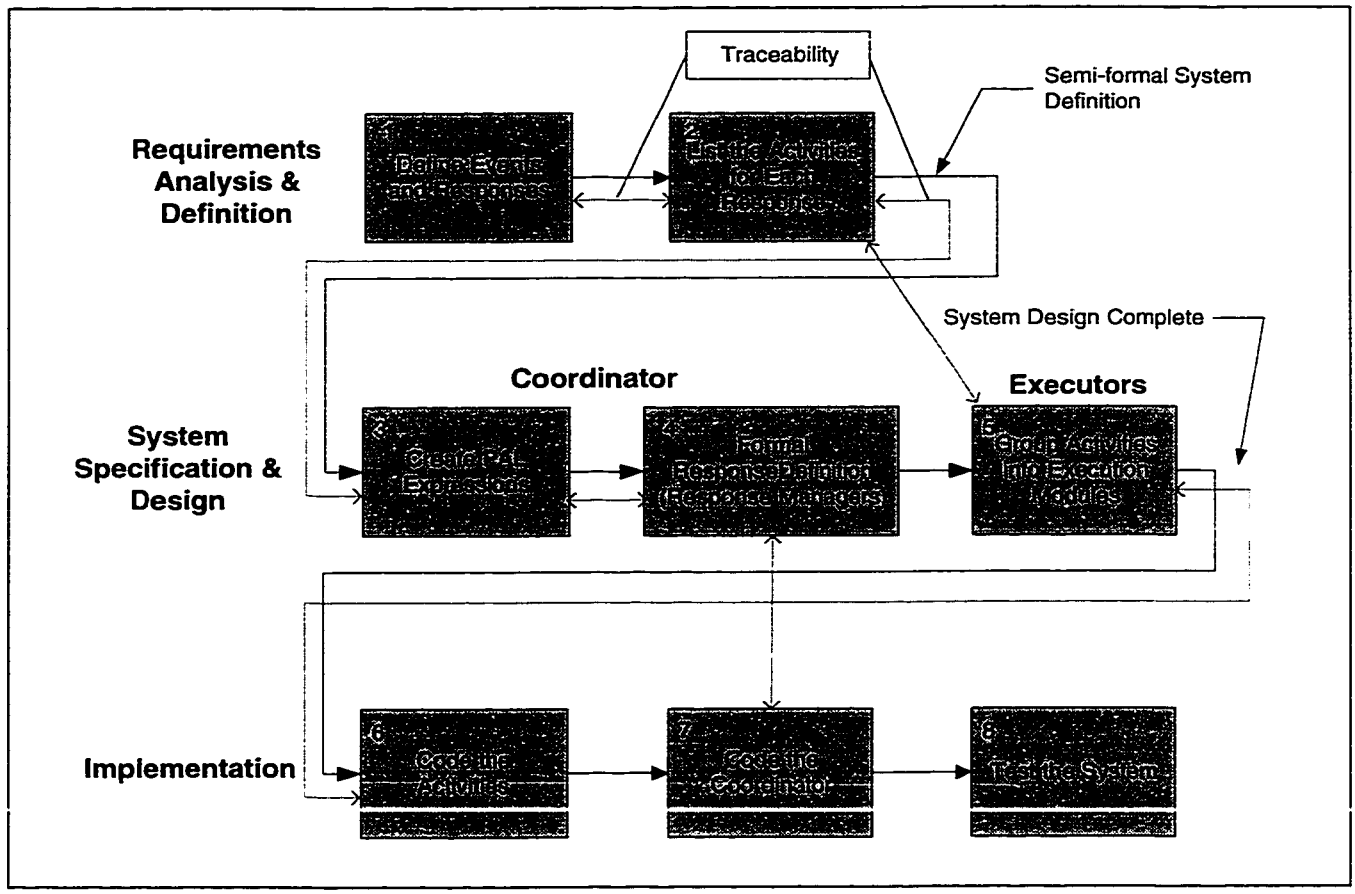


Figure 2-1 - The CBD Process

2.1.1 Requirements Analysis & Definition (Steps 1 and 2)

The CBD process is based on the "Multiactivity Paradigm" [MK1], originally developed for the design of embedded systems. This paradigm is predicated on there being two basic components of work - the various "Activities" of work that have to be executed and the coordination of these activities. Activities can be considered to be "meaningful sequences of operations or actions that can be executed, once started, from the beginning to end without having to wait for anything (e.g. information from other activities, resources, etc.)" [MK1]. In this regard, if the activity execution is separated from the associated coordination or management, then the resulting system is more flexible and easier to design. The emphasis made by Multiactivity, therefore, is that activities or tasks should focus expressly on the performance of the work rather than on cooperating and working with other tasks.

In order to satisfactorily complete the system requirements analysis and definition, a sound understanding of the application and operational domains is required by the requirements specialist(s). In CBD, the requirements which describe the behavior, functionality and performance of the system to be designed, are fully articulated by first defining events, (inputs to the system) followed by specifying how the system is to respond to these event(s). The event responses are formulated as a (comprehensive) set of reactions to the inputs, which can be accepted and /or sensed depending on the system.

Define Events and Responses

Generally, events can include commands or instructions from a User Interface or be received from other systems through communications interfaces. An example of an event (relating to the CMMS) is the user input request to display inventory item information on the

screen. Alternately, a message could be received from a payroll system, for example, querying the number of hours worked over a period of time, by a specific employee.

For each of these specified events, or stimuli, the system is required to react in a particular way. Once the events (functions/ processes), accepted by the system are known, the responses are then set in motion. Formulating the required events and the respective system responses is the essence of determining the system requirements. An example of a response, pertaining to the CMMS example would simply be "display the inventory item information following the input of a request for this information". These events and responses typically match closely and therefore inherently support the need for traceability in the design.

List the Activities of Each Response

Responses consist of a set of activities which are given the required precedence of execution, then scheduled and subsequently executed. Activities (tasks) are considered the smallest units of work of interest to the requirements analyst specifying the system. Activities are independent, i.e., they do not communicate or cooperate with each other, but are not necessarily specific or unique to a particular event response. In many cases, event responses use different combinations of the activities and therefore activities may appear in more than one event response. Examples of activities which could comprise the above-mentioned event response might be:

- Open database;
- Retrieve and display inventory item, and;
- Close database

Once the events have been identified, the respective system responses are defined as a list of activities that have to be executed in a given precedence relationship. With these activities identified, a "semi-formal" system definition now exists. This preliminary description of the system operation is now in a form where it could be disclosed to the user or customer in order to secure buy-in / approval, before proceeding with the detailed design and may be used as part of a package of review material for a Preliminary Design Review (PDR).

2.1.2 System Design (Steps 3, 4 and 5)

Systems created using CBD include, at least, one Coordinator and a number of executors. The following sections describe these entities and outline the process for their design.

2.1.2.1 The Coordinator

Managing the work of the system processes / activities is the responsibility of the Coordinator. It accepts and interprets events or inputs to the system and initiates the necessary work to be performed, or responses required and sends the appropriate instructions to the activity executors. The Coordinator is aware of the current timing needs of each process (event response), and maintains a current list of ready-to-run activities. It also ensures sequencing and scheduling of the activities. There can also be time, interaction and error handling capabilities performed by the Coordinator, which allows the supervision of time outs, interaction between processes and fault management. Notice, in Figure 1-2, that the vertical Coordinator - Executor relationship can exist in all the different layers of the design. Figure 2-2, below, illustrates an example Coordinator structure.

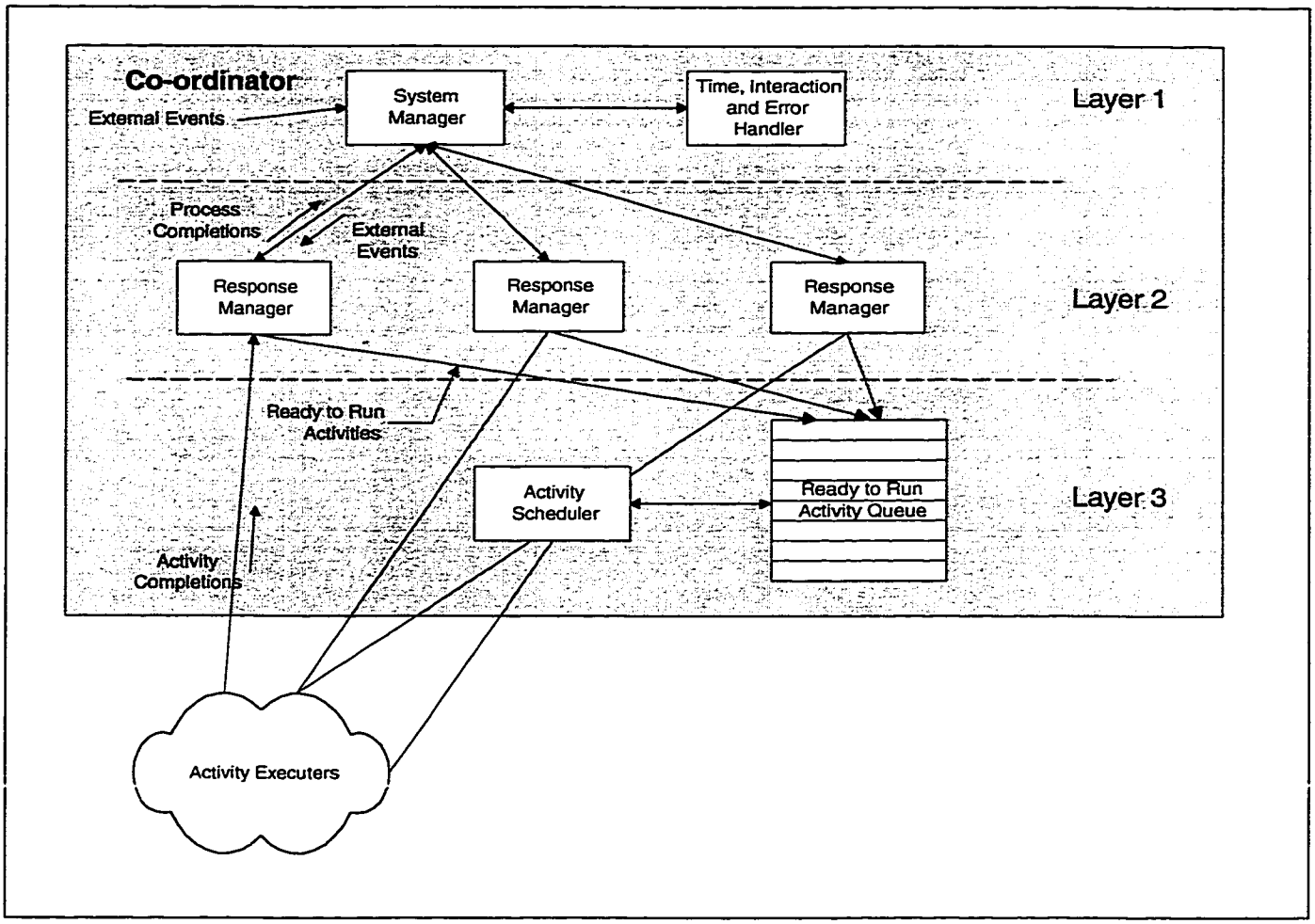


Figure 2-2 - The Coordinator Structure

The Coordinator is derived from the formal definition of the event responses. One method for accomplishing this is through the use of the Process Activity Language (PAL), which was developed as part of the CBD methodology, at the University of Ottawa [MK1]. The PAL is used to define the sequence or precedence in activity execution of the responses and any temporal relationships between them. Please refer to Annex B for a more in depth description of the PAL language, its processes, constructs and conditions.

Response Managers are entities contained in the Coordinator which control all of the responses the system provides (one response manager per response). These managers contain the activity related information (i.e. which activities to invoke and in what order or sequence). The response managers essentially "add the intelligence" by allowing choices of responses to be invoked depending on the type or value of the event received. For instance, in the above example of the event requesting that inventory item information be displayed, the response manager could either cause the requested information be displayed or display a message indicating that no such item existed in the database (if that were the case). Again, "clean" traceability is maintained in the design of the Coordinator because it is based directly on the responses defined in accordance with the requirements analysis and the direct mapping of activities to responses.

2.1.2.2 The Execution Modules

Execution modules are simply aggregations of activities that are normally grouped together to maintain separation of concern. For example, activities could be grouped according to similarity in functionality, type of information or in some other logical way. Note, that these execution modules could be compared to "Objects" with the particular activities being invoked by a call from the Response Manager.

Coordination based systems are flexible and adaptable because one can either add extra activities to the executors or extra activity executors can be easily integrated into the system, as required. In this way, functionality can be added or modified to suit changing or evolving requirements. As the execution modules consist solely of a grouping of specific activities, their structure is simple and traceabililty is easily maintained.

Actually, the job of grouping the (units of work) activities into execution modules, as with any software design undertaking, requires application domain knowledge, and experience. This grouping must be accomplished in a way that too much overhead is not imposed on the Coordinator, and access to and control of critical resources needs to be carefully considered. Each module should relate narrowly, to whatever extent possible, to a specific application concern and access to common types of data. For instance, in the CMMS example, it may make best sense to create one execution module concerning the equipment inventory and another for any activities related to maintenance work orders, as the data relating to these two groups is contained in different tables of the CMMS database. The requirement for concurrent execution is another factor which could affect the choice of how activities are grouped.

2.1.3 System Implementation (Steps 6, 7 and 8)

System implementation is, essentially, the encoding of the various managers and executors. The main point concerning the Implementation Phase is that with CBD, there are no limitations regarding the programming language to be used. In one example of a system designed using CBD - a simple personal agenda system, PAL expressions were used to capture the design, and the language used to implement the design was C. In another example, (a Sales Support System) the design was implemented using a combination of Visual C++ and MS Visual Basic, (these development programs were used for coding the example CMMS as well). This programming language independence is an important aspect, adding considerable advantage to the use of CBD - making it a generically applicable methodology.

Again, it is important to emphasize here, that activity executors can be implemented using COTS components. Which COTS components are used largely depends on the particular role or type of functionality that the executor undertakes, but the range of COTS application types available to choose from is virtually boundless.

It is, of course, necessary to do a good job of testing the system design, which should first involve ensuring that the Coordinator and its user or environmental interfaces are working properly. Then, the interface to an execution module(s) and the ability to properly invoke the appropriate activities needs to be verified. Once this rudimentary end-to-end event response capability works as required, the basic system can then deemed to be working. With this done, the system can then be easily expanded and tested incrementally. As in any software development methodology, regression testing is always required; however, due to the hierarchy and modularity of the CBD architecture, the impact of changes or additions to system functionality will be minimal.

2.1.4 CBD Traceability

CBD intrinsically supports traceability throughout the design process. (The dashed lines in Figure 2-1 illustrate the traceability from the requirements to the design and the code.) As the requirements are articulated strictly as events and directly corresponding responses, no extra effort, on the part of systems analysts or requirements specialists is needed to relate what the system is required to do, as a result of inputs at its external interface(s). This feature of "built-in" traceability exists within the design of the Coordinator, because the responses are cleanly specified as one or more activities to be carried out in a particular sequence or precedence. Furthermore, because the Execution Modules consist of a collection of activities, the correlation, between the functionality that these executors perform in response to a given event, is simple.

In a system designed using CBD, the structure of the code will closely reflect the design structure, as well. The code making up the Coordinator and Executors is written as implementations of responses and activities respectively, thereby maintaining a close and easily understood link to the design and requirements. Figure 2-3 illustrates the traceability features offered by CBD.

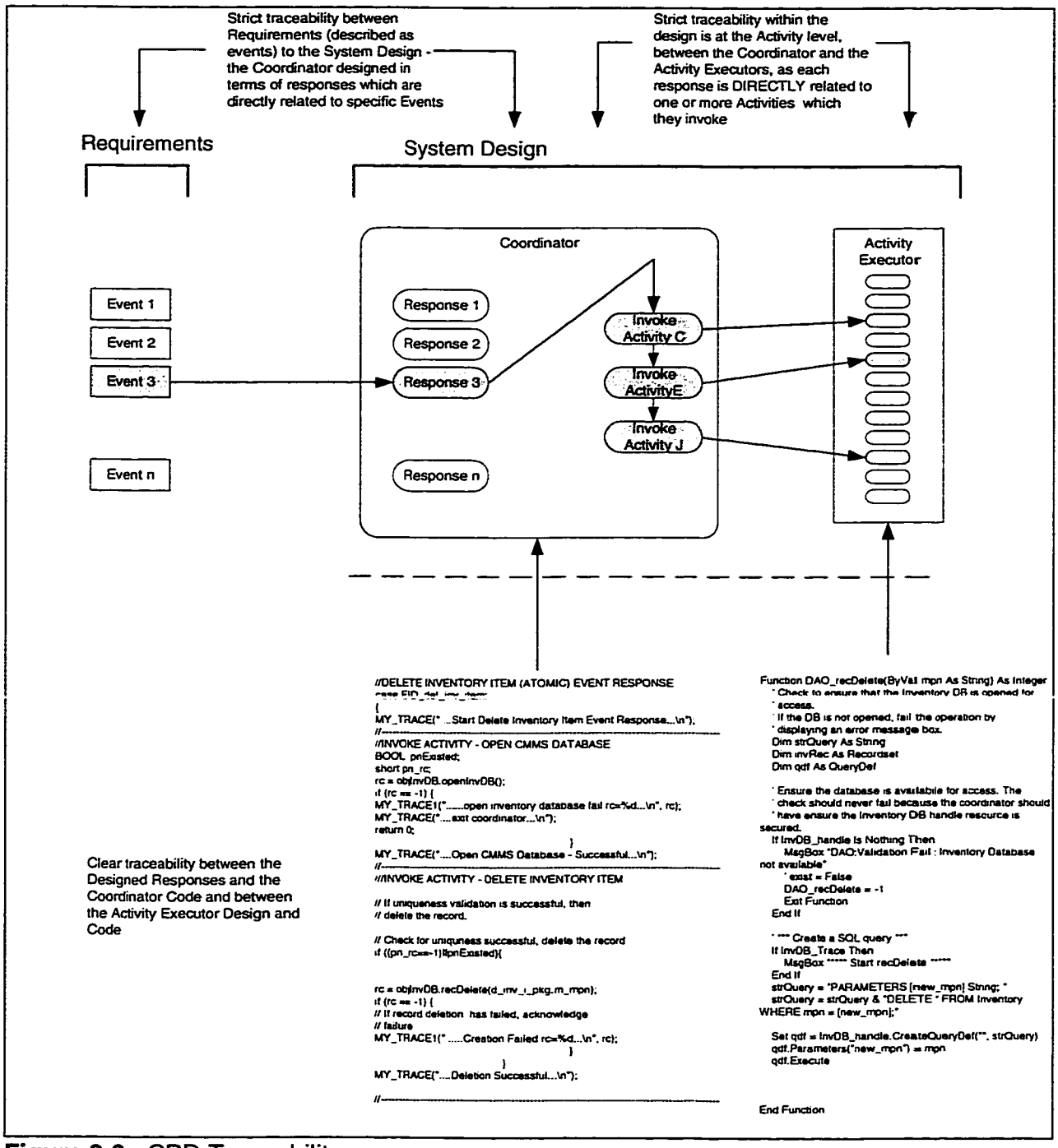


Figure 2-3 - CBD Traceability

2.2 An Overview of Selected CBD Related Work To-Date

2.2.1 Design Tools for CBD

In order to show that one can create a workable design environment, a series of tools were developed by various students during the course of their studies at the University of Ottawa. A brief description of these tools is provided below.

Extended PAL Editor [CHU31]
The Extended PAL Editor allows the entry, modification and integration of partial responses which have been defined using PAL expressions or pseudocode. This tool performs lexical and syntactical checks on the expressions and translates them into indented pseudocode. Still under development, is a graphical module to generate process activity nets.

PAL Verifier [HAR32]
The PAL Verifier checks for any temporal problems (e.g. deadlock or starvation) of the interacting PAL processes. It also determines whether correctness properties are satisfied on a global state graph representation of the system's temporal behavior.

PAL Simulator [SAN33]
The PAL Simulator interprets and executes PAL processes and it is used in two different modes. Firstly, the tool can be used to check whether PAL processes are correctly specified (to do this the simulator is run assuming infinite resources, i.e. a very large number of general purpose activity executors). Secondly, the PAL Simulator can used to verify the system can meet its design requirements (constraints) by specifying:

- Number and type of executors;
- Activity to executor allocation, and;
- A number of activity scheduling schemes.

The PAL Simulator also includes an event generator to simulate various types of external events.

2.2.2 Embedded /Event Driven Systems [MK1]

The coordination-based technique for software design was initially developed for embedded (event-driven) systems, which typically have embedded computers for ongoing monitoring and control purposes. Early simple embedded systems assumed sequential response, where an operation did not begin until the preceding one had been completed. However today, embedded or real-time systems require multiple processors and concurrent execution - which require more complex coordination between the activities.

Restricted Object Based Design (RODB) [MK26], for event driven systems, is based on the Multiactivity Paradigm, and applicable specifically to "Shrink-Wrapped" or "multiversion interactive" software. The execution modules (Restricted Objects) reply only to the response manager which provides simple sequencing and decision making functions. To design a system using RODB, the PAL language is normally used.

Examples of applications developed using RODB include educational simulators, including the Program Execution Simulation Tool (PEST) [ES34] and a Generalized Assembly Language Executor (GALE) [ES34]. PEST shows the information flow during program execution, at the microstep level. GALE consists of five execution modules and allows the selection of the machine type, addressing modes and word size.

There was also shrink-wrapped software developed - a Personal Agenda System [ES34], which has monthly and daily windows. Additional windows were also provided in this application, with simple editing capabilities for Add, Delete and Modify commands.

2.2.3 CBD With COTS Modules

Robert Au Yang's Project (COORDINATION-BASED DESIGN; TOWARD COMPONENT-BASED SOFTWARE DESIGN) [AY28] demonstrates the utility of the CBD process for integrating off-the-shelf software. As part of this project, two example systems were designed and implemented. The software packages employed were MS Access and MS Excel. The first application developed was a Sales Support System. Based on this system, Au-Yang demonstrated the ease of modification to the design, by developing a second example (House Mortgage Support System) from the first system. Figure 2-4, below shows the architecture of the Sales Support System developed.

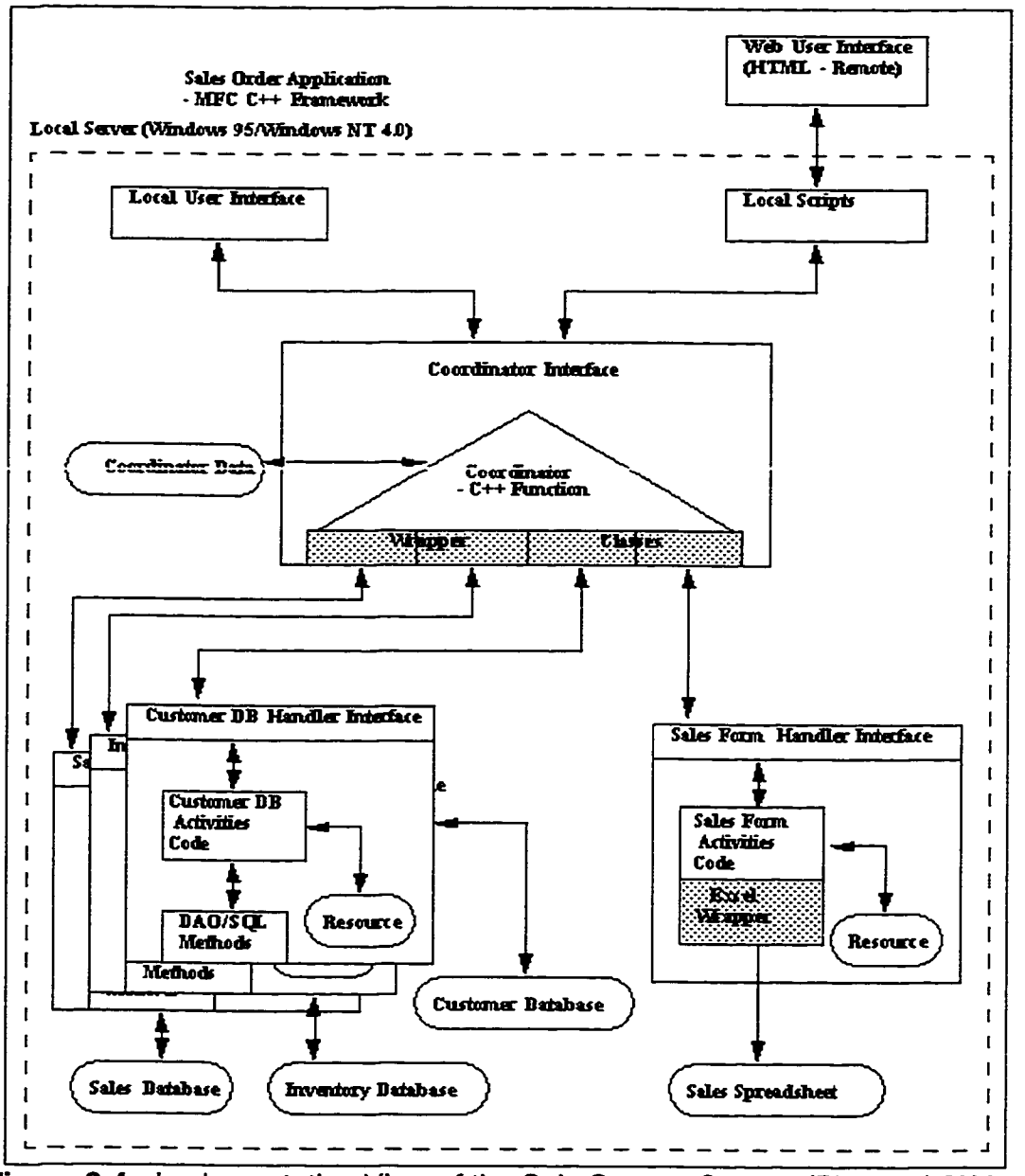


Figure 2-4 - Implementation View of the Sale Support System (Phase 2) [AY28]

For the Sales Support System, customer requirements were first specified, as indicated in Table 2-1, below, which contains a sample of the event responses designated for this system. The bolded activities are the ones implemented in code.

Table 2-1 - Summary of Event Responses of the Sales Support System

Event	Activities (in Order of Execution)	Resources
Login : Selection of user type	Display screen Set UserType Set Menu states	Login menu Login dialog box User Type variable Menu enable/disable state
Retrieve Whole Inventory	Display screen **Open Inv DB** Allocate record buf. **Get record from Inv DB** **Close Inv DB** Print record Unallocate record buf	Retrieve Inv. Menu Retrieve Result dialog box InvDb handle Number of records variable Record buf. Variable
Add Item	**Display screen** **Open Inv DB** **Create record to Inv DB** **Close Inv DB**	**Add Inv. Menu** **Add Inv. dialog box** **Add Inv.Ok message** **Add Inv.Fail message** **InvDB handle** **record buf variable**
Edit Item	Display screen **Open Inv DB** **Get record from Inv DB** **Put record to Inv DB** **Close Inv DB**	Edit Inv. Menu Edit Inv. dialog box Edit Inv.Ok message Edit Inv.Fail message **InvDB handle** **record buf variable**
Enter Sales Order	**Display screen** **Open Inv DB** **Get record from Inv DB** **Put record in Inv DB** **Close Inv DB** **Open Cust.DB** **Create record in Cust.DB** **Close Cust.DB** **Open Sales DB** **Create record in Sales DB** **Close Sales DB** **Open Sales Form** **Calculate IS** **Accumulate SOS** **Calculate PST** **Calculate FST** **Calculate ST** **Return ST** **Create printable SOF** **Print SOF** **Close Sales Form**	**Sale Order menu** **Sale Order dialog box** **Sale Order print OK message** **InvDB handle** **CustDB handle** **Inv record buf variable** **Cust record buf variable** **Printable SOF handle**

Next, the execution modules were defined. Related activities were grouped into six modules (the bolded ones were implemented):

1. **Inventory Database Management;**
2. **Sales Database Management;**
3. **Computation Management;**
4. **Sales Form Printing;**
5. Report Generation, and;
6. **Customer Database Management.**

One observation made by Au yang was that because the entire COTS packages were required to be loaded, as opposed to only the necessary subset of functionality, there is a resultant high demand on system resources. However, he concluded that CBD is a viable methodology for COTS integration, which demonstrates the strength of component-based

software design. He also showed, through the second example, that CBD easily allows reuse and tailorability.

2.3 CBD For Modifiable Software

CBD offers the system designer / user the capability to design software in an intuitive, organized fashion and through its simplicity, designers can easily modify, and expand their software programs.

2.3.1 System Development Using Existing Software

Systems designed using the CBD approach, lend themselves easily to being modified, or customized, to create completely new systems in different application domains. Au Yang shows this capability in his project by creating the House Shopping Support System from the Sales Support System. He writes "...CBD provides the framework for the partitioning of the work and coordination, thus, simplifying the modification process and making the adaptation feasible" [AY28]. Creating this second design would be admittedly more difficult if the domains were not similar. In this case, however, the activities, events and responses are the same or similar and the database structure is the same. The systems differ the most with respect to the data.

Similarity in architecture is one of the most important criteria that affects the ease with which large scale re-use and customizing can be accomplished. According to AT&T, one of the principle elements required to achieve maximum benefit from software re-use is explained in the following way: "Architecture frameworks for reuse must be developed. Reuse at the architectural level, rather than at the subroutine level, is required to achieve large-scale gains in productivity" [ALSR29]. Systems developed through CBD fulfill this requirement of architecture similarity.

2.3.2 Addition of New Software Modules

The capability to add new software modules to existing CMMS packages is a considerable advantage which some vendors actively advertise. This is particularly attractive to first time CMMS buyers and in industries characterized by significant growth and change. CBD is a methodology which clearly provides for the ability to add on to current packages, due to strict module independence. To accomplish this, modules can be designed and implemented separately using existing as well as newly developed activities. Of course, the Coordinator(s) also must be first updated to contain the required response managers. In the example CMMS developed in this thesis, the task of adding the different modules was found not to be complicated or difficult, despite being considerably time intensive, due to the Author's inexperience with the intricacies of performing the required programming.

2.3.3 Tailorability in CBD - Atomic and Combined Responses

The ability to modify software is desirable at all levels of code. At every level, CBD ensures that the program is well structured and organized, allowing the code requiring modification to be easily located and changed without affecting neighboring code. Individual activities can be easily added to, changed or omitted from a module, and in the Coordinator, it is easy to add or modify the responses.

As discussed above, the principle concepts of CBD require the definition of responses that consist of a pre-determined set of activities. Individual activities can also be part of the make-up of more than one event response. Moreover, it is also possible to combine elemental or "atomic" responses into "combined" responses, depending on the nature of the system being designed. Where it is possible to do so, it makes good sense,

organizationally, to group responses hierarchically; this is consistent with the basic CBD response / activity relationship.

For simple systems, involving relatively few responses and activities, it may not be necessary or helpful to define combined responses, however, for large and or complex systems, particular advantage may exist in doing so. The modules need only to be populated with the desired activities and by re-using and combining responses, the Coordinators can be re-designed and coded expeditiously. Figure 2-5, below, illustrates the concept of atomic and combined responses.

In addition to the ability to effectively build up a system design by selecting or creating the required atomic and combined responses, these responses can then be easily swapped in or out, changed, and made more or less sophisticated as the need arises - this is the heart of the concept of tailorability through CBD.

In addition to the organizational benefits associated with the grouping of responses in this fashion, this technique could also facilitate experimentation into combining large numbers of different responses (say from a Response Library) to achieve an efficient large scale software solution or product. These combined responses provide the capability to design and implement powerful and sophisticated functionality, which is only limited by the contents of the response library and the skill of the developer. We believe that, with the appropriate amount of experience, a developer will be able to quickly and efficiently design and build a system by drawing upon a collection of atomic and combined responses contained in a response library. Furthermore, through this straight forward CBD methodology, the learning curve is gentle, for a designer to gain a reasonable level of competence.

CBD, therefore, allows tailoring at all levels of granularity through, either changing the individual activities or event responses or through the ability to easily add, or remove complete groups of responses and the associated activity executors (modules) without impacting the others. The use of atomic and combined responses further enhances tailorability through the use of CBD.

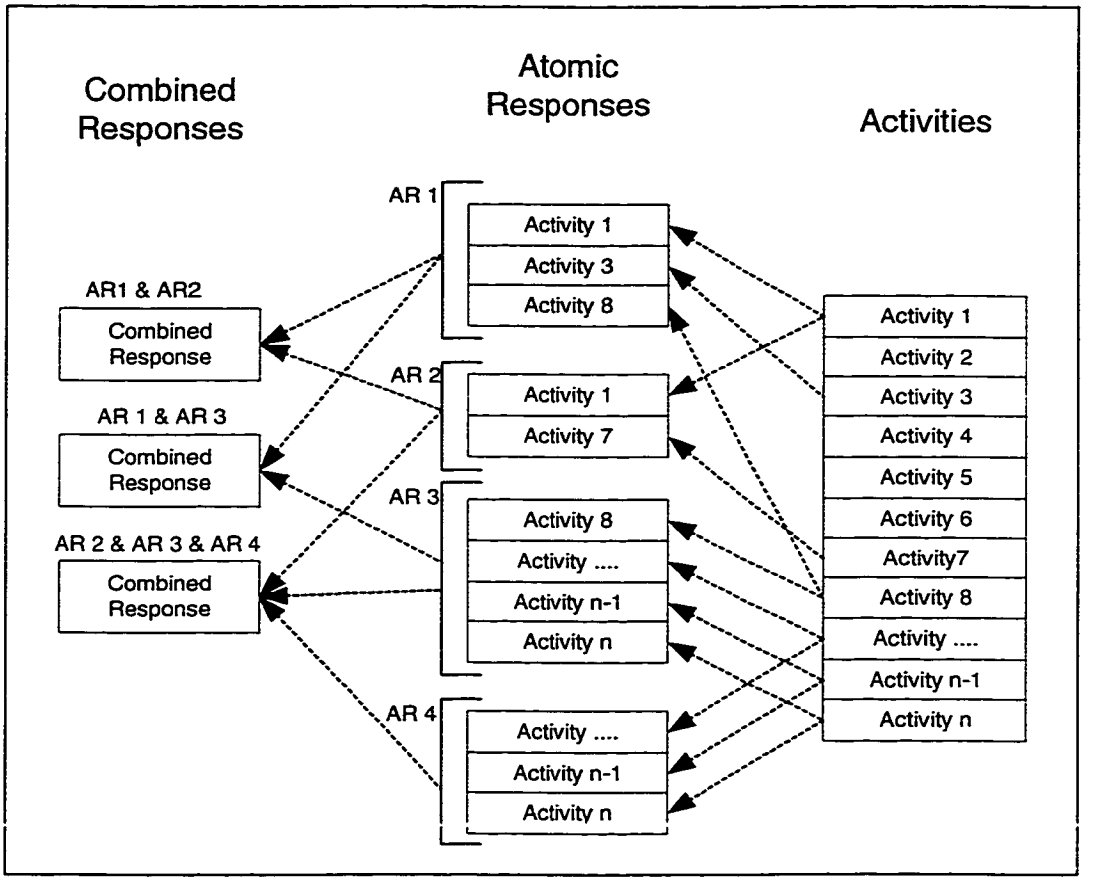


Figure 2-5 - Atomic and Combined Responses

2.4 Conclusions

The requirements specification technique of using events and responses, the separation of coordination from execution in CBD, and the associated hierarchical architecture constitute a sound practical environment for software design. Fundamentally, the overall theme involved is, simplicity and tailorability. Today's popular design methodologies, e.g. Structured Design, Object Oriented Design, Patterns, etc., have certainly met with commercial success, however, it is suggested here, that the principles of CBD are sound enough that this methodology has the essential ingredients to achieve commercial success, as well.

CBD systems are easy to organize, manage and design functionally. Advantages of CBD include many benefits which coincide with the qualities and capabilities of the best CMMS software packages.

Many articles about CMMS's herald the merits of the type of language or programming technique used in their development. Often, it is suggested that the better CMMSs are built through object-oriented programming. In the article "Key Functions of Maintenance Management Systems" [KF10] the author recommends that some of the CMMS selection criteria should be:

- Primary function of the system;
- Features and functionality of the system;
- Modularity of the software;

- Client-server implementation;
- Object-oriented programming;
- Operating system environment, and;
- ODBC compliant;

Noticeably absent from current literature, articles or papers, however, is any mention of the importance of using the proper design methodology. The only exception to this, typically being the often discussed need to ensure software modularity. The importance of a proper design environment (such as CBD), nevertheless, cannot be understated.

It is suggested that CBD offers a technique and structure for the development of software packages that is well suited for CMMS development, or any other type of business software. CBD simplifies modification (through tailorability) allowing software to remain in use for longer periods of time. The modularity of CBD also lends itself perfectly to development of modern business systems, most of which must be modular to be competitive in today's marketplace. And, add-on modules can be implemented without impact to the other parts of the software package. Additional functionality can be easily supplemented as requirements change or can be tailored to suit different client environments. Furthermore, activity executors can also be appended to overcome bottlenecks and provide extra performance, such as response time - an important quality of good business systems. At the lowest level, resources are allocated at once, ahead of time which reduces contention handling.

As concluded by M. Krieger and S. Lemire [MK26], CBD or "RODB is also, to a great extent, self documenting." Identifying and recording the events and responses constitutes documentation of the requirements specification. Defining the PAL expressions, response managers, execution modules is effectively documenting the design, and as can be seen in Annexes C, E and F, the structure of the code easily follows the design, documenting the implementation.

Because of its centralized nature, it is simple to implement and various scheduling schemes. The Coordinator is aware of the current timing needs of each Multiactivity process. Ready-to-run activities and the status of each processor is also tracked to meet deadlines and to achieve load balancing. Once libraries of atomic and combined responses and commonly used core execution modules have been established, the principle remaining tasks are only integration and test. Additionally, because of the simplicity of the CBD technique, developers can more easily specialize in a particular application domain, and alternately easily gain competence in different application areas. A potential disadvantage of CBD, however, is that the strict hierarchical "master/slave relationship between the Coordinators and Execution Modules, will likely impose costs in terms of system performance.

While CBD was and is particularly well suited for embedded, real-time, event driven systems, it has been argued that its applicability is much broader. Potentially, any system, for which an application's functionality is invoked by an event (whether that is a keyboard generated command, input message from some other system or a sensor), can be designed using CBD. Au-Yang's Project also effectively showed that CBD is widely applicable enough that it can also be used to design a system that integrates functionality from COTS products. Given the requisite amount of time, money, and commercial interest, CBD has the potential to achieve renown, similar to commercial software design methodologies.

3. Computerized Maintenance Management Systems - An Overview and Market Survey

3.1 Basic CMMS Functionality

Several assessments, of the most common features of CMMSs, are available in many articles written to-date. These features are typically grouped or contained in specific software modules, some of which are part of the core system functionality and some of which are optional. Many of the modules are common among the different packages, however, there is no strict standard suite of capabilities which the CMMS modules adhere to and there is a degree of functionality overlap among them, as some of the capabilities are captured or contained in different modules, depending on the particular product.

Two trade magazine articles "Finding the 'Right' CMMS" [FRC9] and "Key Functions of Maintenance Management Systems (CMMS)" [KF10] contain assessments of what the most common CMMS features are. From Table 3-1, which compares these two lists, it is obvious that many of these assessments can vary significantly.

Table 3-1 - Common Features of CMMSs

Finding the 'Right' CMMS	Key Functions of Maintenance Management Systems
Work order	Work order generation and tracking
PM	Preventive maintenance
Scheduling	Planning and scheduling
Equipment	Equipment resource planning
	Job cost calculation
	Failure analysis
Material inventory	Inventory control
	Purchase requisition
	Calculation and reporting of costs
	Production of management reports
	Multi-level security
Record	
History	
Material and tool control	
Engineering standards	
Management controls and reports	

Personnel management is, also, an indispensable part of overall maintenance management. Storage and retrieval of information on an individual's skill level, labor rates, availability and training certification is often required. Report generation, Standard Query Language (SQL) query capability, graphs and charts are also prevalent capabilities provided by many commercial CMMSs.

Predictive maintenance (PdM) is a capability furnished by some of the more sophisticated packages. There are also separate PdM packages which can be integrated with CMMS software. This capability has existed for a number of years, but often is not fully implemented or utilized.

An example of a system providing PdM functionality, is MP2 [MP11] which offers the capability to monitor variables such as temperature, vibration and amperage. It flags equipment to be checked based on sensing statistical exceptions. Tolerances can also be set by manufacturer's min./max. readings, statistical limits, or trends, and condition monitoring (e.g. for qualities such as color temperature to touch, clean/dirty) which allows equipment condition to be tracked and identified for work order input.

Another example, where human intervention is not needed, is where a critical piece of equipment can be monitored continuously to ensure the temperature is within acceptable limits. If the temperature exceeds the constraints, a control loop can activate a fan to blow cool air onto the affected area or equipment.

3.2 What Makes a Quality CMMS

Consideration of the functionality provided by the CMMS package is the usual initial criterion for determination of its suitability. However, the ease of use, i.e. user friendliness, is also an essential factor in a CMMS's success. And, there are several other factors which must be considered, as well. The following sub-sections describe some of the principle characteristics that a "quality" CMMS should exhibit. It is interesting to note that several of these measures of quality are "ilites", as well, as discussed in Chapter 1.

3.2.1 Adaptability / Scaleability

As there are many "one of a kind" CMMSs installed in various maintenance locations and facilities, they must be inherently adaptable in order to be commercially competitive. Not only are there very many different types of CMMS environments, but there is also a variety of sizes of systems as well, from very small to very large multi-location systems. The market for CMMSs is large, resulting in a very competitive business. Vendors need to maintain a competitive edge with respect to the ability to tailor and maintain their products in a world which is affected significantly by changing technology. The ability to change or modify a product to suit new customers or different environments is therefore critically important.

Modularity is one quality of architecture which CMMSs must have, to accommodate this adaptability/scaleability criteria. Modules typically provide specific functionality such as work order generation, inventory management and bar coding capability, etc. Modular CMMS's can therefore be designed to specific user needs, both in terms of type and size of business, addressing a wide range of demands and over a period of time, as requirements change. A possible problem which may occur, however, if too many modules are built-in, is the number of concurrent responses being implemented may become too many, slowing down response time and degrading performance. Open systems standards / architecture is critical to being able to integrate other modules easily, expand features or support future enhancements.

3.2.2 Ease of Integration With Other Software Packages

Figure 3-1 shows an example of other systems integrated with a CMMS. Types of applications which CMMSs are often required to integrate with include accounting, human resources, and manufacturing systems. One of the most important requirements of integration is for the CMMS to support the same type of database management system (e.g. Oracle, Informix, Sybase, SQL/Server, etc.) used by the other systems.

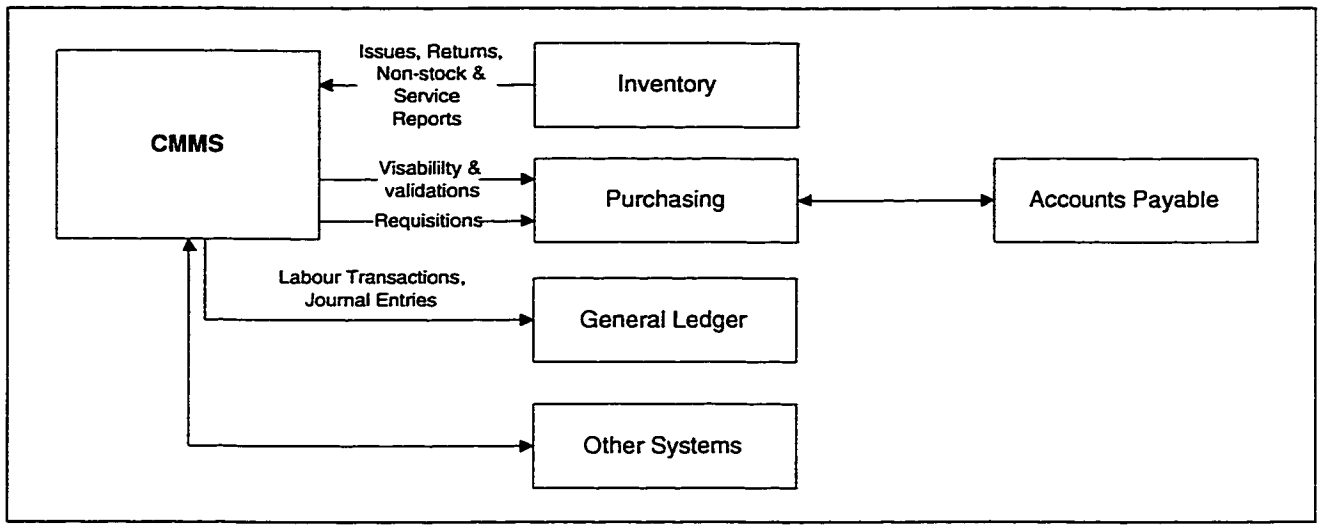


Figure 3-1 - CMMS Integration With Other Software Packages [PS12]

The Huntsville Hospital [FS16] is an example of an organization that has successfully embraced the capabilities of a CMMS and has integrated its CMMS with three other hospital systems as shown in Figure 3-2, below.

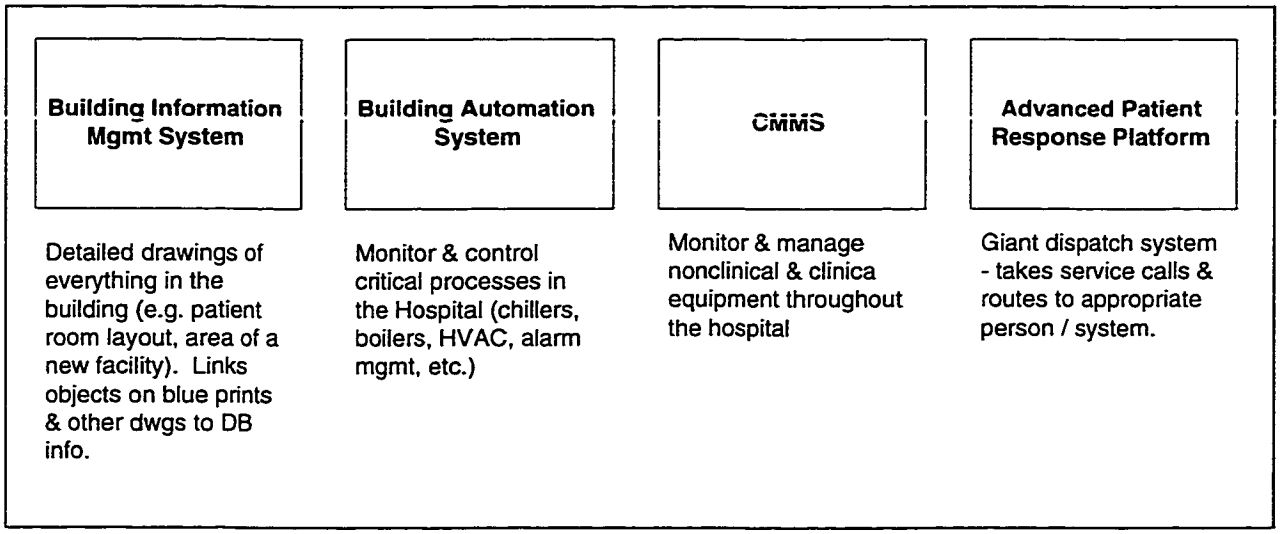


Figure 3-2 - Huntsville Hospital System

Figure 3-3 shows the structure of the Enterprise MPAC Client-Server architecture with third party application interfaces. One of the advantages provided by this architecture is that the APIs provide a measure of isolation between the external application and the internal procedures, which allows the stored procedures to be modified and reused without affecting the API(s).

Many preferred CMMS vendors adopt a "best of the breed" approach, the objective of which is, to integrate the most popular or capable specialty packages (e.g. best accounting package, best manufacturing package, etc.) from different vendors. However, integrating multi-vendor hardware and software packages is not without its difficulties, such as database compatibility. As discussed in Chapter 2, however, CBD is a methodology which is well suited to component based design.

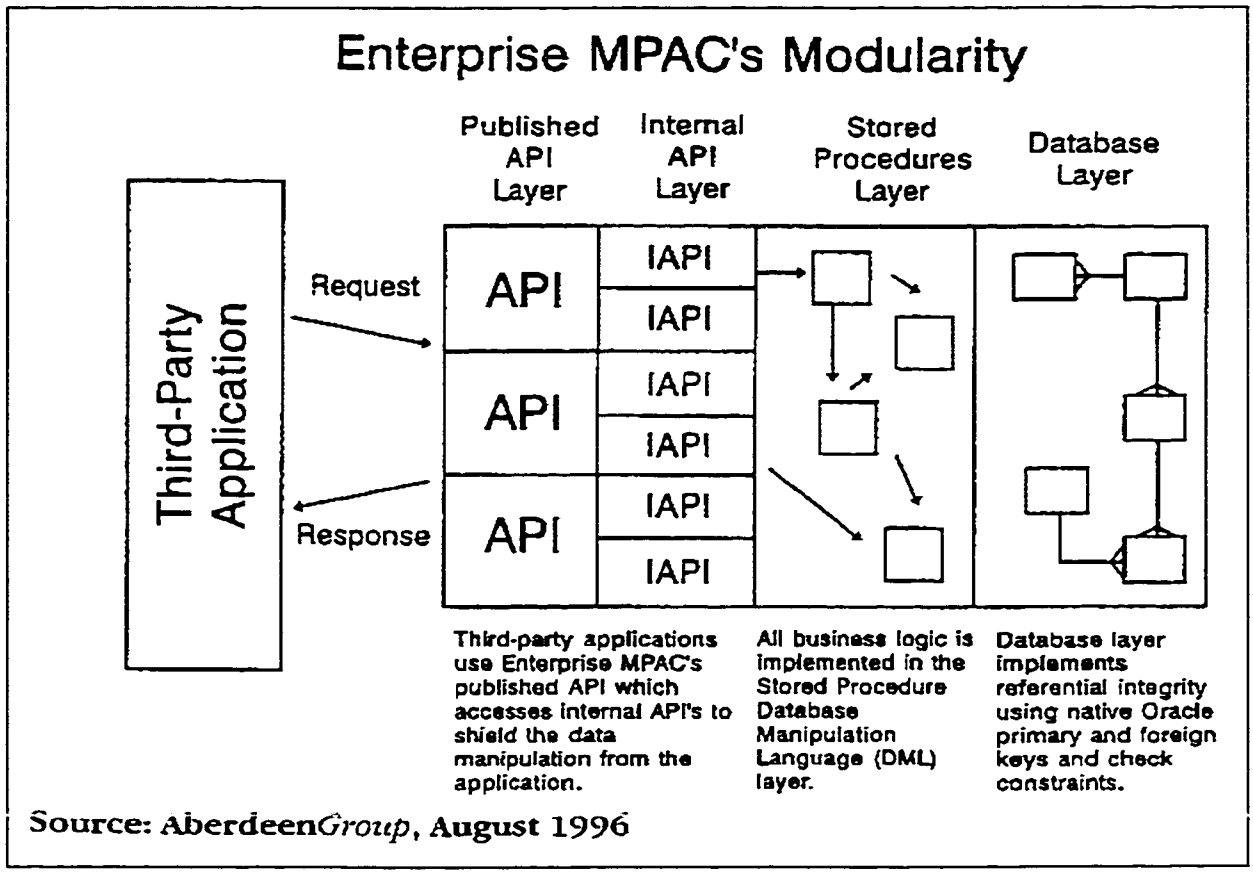


Figure 3-3 - Encapsulation of MPAC Business Logic [TSW13]

3.2.3 Maintainability/Supportability

From the customer's or end-user's point of view, the system must be supportable by the vendor. This implies, not only certain commitment on the part of the vendor but also highlights the importance of the need for robustness and ease of maintenance as being paramount.

3.2.4 Response Time and Accessibility

Typical CMMS users can include many different groups, including managers, technicians and data entry personnel. An important quality of a good CMMS is, therefore, its ability to provide access to the different users easily and simultaneously, requiring multi-user data files and shared data management.

Response time is therefore a factor which affects whether or not a CMMS system gets used well. Problems can occur, if several users are accessing the same module at the same time, resulting in degradation if the system cannot be tuned to provide the requisite response.

3.2.5 Configurability

Today's systems often allow the end users to configure the CMMS and change its look and feel. This is also an important capability for any successful CMMS to provide.

3.3 CMMS Architectures and Development

Many CMMSs are constantly being evolved to ensure advantage of current technology is taken and to suit new customer requirements. In the following paragraphs, some of the common CMMS architectures and development tools are discussed.

CMMSs and the capabilities they offer are often underutilized. Data is easy to collect, however, what to do with the data and how to convert it to usable information is often a more difficult challenge. Few plants or maintenance organizations, though, have the resources to devote to this need, and implementation of CMMSs is not always successful, as a result. Another significant cause of disuse is poorly defined and misunderstood requirements. As there are many sizes and types of commercially available CMMSs, this sometimes results in not providing the "right fit". CBD, however, (through its definition of events and responses) offers an intuitive means of formulating system requirements (as described in Section 2.1.1).

3.3.1 CMMS System Architecture

Most CMMS's include a relational database. Some can be configured to operate with different databases, for example DynaStar 2000 will operate with Oracle, Sybase and MS SQL Server as well as Watcom SQL database. They operate over different media such as LANs and WANs which facilitate network access to multi-user data files and shared databases. The client server architecture is a typical configuration with network and single user versions also available for some CMMS's. The Enterprise MPAC product, using a three tier client-server architecture, is illustrated in Figure 3-4.

3.3.2 Platforms

CMMSs are available for most common operating environments (e.g., UNIX, Windows 95, and Windows NT. UNIX-based servers are the most common, however, migration to Windows-NT is likely in the near future. Other platforms used include DOS microcomputers, OS/2, Macintosh, IBM AS400, IBM Mainframe and DEC.

3.3.3 Programming Languages Used

Some CMMS tools are created "from the ground up" and others are based on existing packages, for example MS Access. Languages used to develop CMMSs include:

- MS Visual C++;
- C++, Delphi;
- Open Insight/Basic+
- MS Visual Basic;
- Powerbuilder;
- RPG400, and;
- C, C++, COBOL, Object Pascal

A particular common trend with some suppliers of CMMS's, is the employment of Object Oriented Technology (OOT). Using industry standard object-oriented methodologies and technologies is the underlying architecture and technology upon which many build their client / server applications. CMMS products are defined as a set of objects, business rules, and data models in an object repository, which are independent of hardware, operating system, database, or execution environment. These CMMSs take advantage of specific database and operating system capabilities, like the database vendor's native APIs, as they communicate directly to the database with no intermediate layers, thus leading to full

utilization of native database capabilities such as distributed data management, triggers, and stored procedures. This allows database independence.

3.4 A Survey Of Computerized Maintenance Management Systems

There is a plethora of commercially available CMMSs on the market in active use in many maintenance environments, today. This section introduces several commercially available CMMSs, discusses their general features and capabilities and identifies any significant advantages or limitations. It was found, from this survey of current product literature, that most of the applications offer comprehensive sets of functionality, which are typically available as core and optional modules. The selection of CMMSs presented here was based exclusively on literature and other information obtained from the vendors directly, through the Internet or from industry periodicals. Further details on each of these applications, discussed below, is contained in Annex A, in tabular form. The intention, here, was to gather information on a sufficiently sized sampling of CMMSs to understand the general functionality offered and the range of types of commercial CMMSs available today. In this regard, it is not suggested that the packages reviewed below, represent an in depth or rigorous cross section of the CMMS market.

For the example CMMS designed and implemented as part of this thesis, (Chapters 4 and 5) functionality to be designed, was determined based on the review of the capabilities noted in the product literature reviewed.

3.4.1 GP MaTe [GP14], [TM15]

GP MaTe is a comprehensive package, used as a corporate standard by several companies including Ashland Chemical, Reliance Electric, Seagrams and Air Products and Chemicals. It uses a server database manager called Unidata, a Pick descendant that runs on UNIX, DEC VMS, and NT. The application development language is UniBasic, an extended BASIC that is a descendent of Pick Basic (part of the original Pick operating system). The original application code for GP MaTe was written in R-BASIC. All the data files that make up GP MaTe were directly ported from Advanced Revelations to Unidata.

The GP MaTe 4 client was written in Visual Basic 5.0, using a set of third party controls for presentation of data to the user. Communication between client and server is carried out using an in-house developed client/server TCP/IP communications system. The communications system server was written in C and uses Berkeley sockets or was written in Microsoft Visual C/C++ and uses Winsock.dll, depending on server platform. Server application code is completely platform independent (i.e., one set of source code runs within Unidata on all server platforms). The client half of the communications system was written in Microsoft Visual C/C++ and encapsulated as a single dynamic link library (DLL).

Other DLLs, all developed in-house, are part of the GP MaTe client. All of the Windows based components of GP maTe 4 were written either in Microsoft Visual Basic 5.0 or Microsoft Visual C/C++ 5.0, except one DLL that was compiled with Microsoft Visual C/C++ 4.1. Reports for GP MaTe 4 were produced using Crystal Reports and an in-house developed ODBC driver. The ODBC driver uses the same TCP/IP communications system that is used by the rest of GP MaTe. According to Tom Murphy, Director of Product Development for GP Solutions, "If there is a major lesson learned from our experience with Version 3 of GP mate, it is:

1. Using mature standard development tools like Unidata, Visual Basic, and Visual C/C++, and;

2. Developing our own specialized tools to extend the functionality of the standard development tools, gives us much more control of our own destiny and allows us to produce a superior product."

3.4.2 DynaStar 2000 [DS17]

This product meets a range of user requirements, from single users to multi-user network environments. DynaStar 2000 provides the usual suite of CMMS capabilities available on the market, with the exception of predictive maintenance. The DynaStar 2000 Maintenance Management System (MMS), used with Windows 95, comes with a standard Watcom SQL database, but is capable of interfacing with major SQL databases, as well.

3.4.3 MP2 Professional 5.0 [MP11]

MP2 Professional is a high end CMMS package which includes all the requisite capabilities, including predictive maintenance. Variables such as temperature, vibration or amperage can be monitored to identify statistical trends. When limits are reached, MP2 alerts the maintainer to make the necessary repairs, before damage or equipment failure occurs. Tolerances can be specified, including manufacturers min/max. readings, statistical limits or trends. For observable qualities, (e.g. colour, temperature to the touch, clean or dirty oil) conditions can be tracked when performing maintenance.

MP2 Professional 5.0 has been completely written from ground up to take advantage of Delphi and MS SQL Server client server database. It has a compiled front-end resulting in the on-screen commands not needing to go through an interpreter before being processed. It was designed with a thin-client architecture, (there are over 800 stored procedures). MP2 also advertises a Component Tree Feature - which allows matching of sub-components (children) to major equipment types (parent).

3.4.4 PMC for Windows [PMC18]

PMC for Windows is general maintenance management system which provides most of the usual CMMS functionality. PMC For Windows also includes several "extras" such as OLE embedding, bar code interface, Internet access, CDROM interface and the documentation of procedures to meet standards and quality assurance requirements. PMC For Windows was also written in Microsoft Access which facilitates easy user customization.

3.4.5 Work Order Tracking [Sho19]

Work Order Tracking (WOT) was designed for managing equipment in a facility (e.g. factory, building, hospital, etc.). It was developed using Paradox for Windows as a database application, running on a network allowing multiple access to users. It was designed and tested on-site for two years, evolving to its current form. This offers some advantage, as one of the most significant problems, experienced in having CMMSs accepted readily in the work place, is the ease of use and relevance to the employees and managers using it. WOT does not include PdM or purchasing functionality, although bar code interface capability is.

3.4.6 Maintenance Management Systems (Bass Associates Inc.) [MM20]

This product includes the capability to support scheduling, records, spare parts, training, outside contracting and a database. Maintenance Management Systems (MMS) has both

scheduling and analytical functions. It is designed to be accessible from a LAN or stand alone. From the documentation reviewed, the Bass Associates system does not include a PdM module.

3.4.7 Atlas Equipment Manager (2000) [CM21]

Atlas 2000 was designed to adapt to any industry or type of organization. It provides statistical predictive maintenance functionality and includes the capability of monitoring the condition of each piece of equipment and automatically generates PdM work orders from user-defined variables. Otherwise, this product supplies the normal and full compliment of CMMS functionality.

3.4.8 PlantLIFE Maintenance [PL22]

PlantLIFE Maintenance is tailored to meet the requirements of today's modern process, chemical and power plants. It is a CMMS with typical capabilities and can be fully integrated with other PlantLIFE software modules, e.g. Safety Management, Document Management, CAD and Plant data.

3.4.9 Enterprise MPAC [TSW13]

The MPAC system is one of the few that offer many extra capabilities, not ordinarily available from other CMMS vendors. This system has a 3-tier client-server architecture which consists of an Application Logic Layer, a Database Layer and a Presentation Layer. The Presentation Layer is responsible for generating the graphical user interface. The Application Logic Layer providing a query interface between the Database and Presentation Layers is divided between the client and server platforms. The Database Layer consists of the Oracle RDBMS.

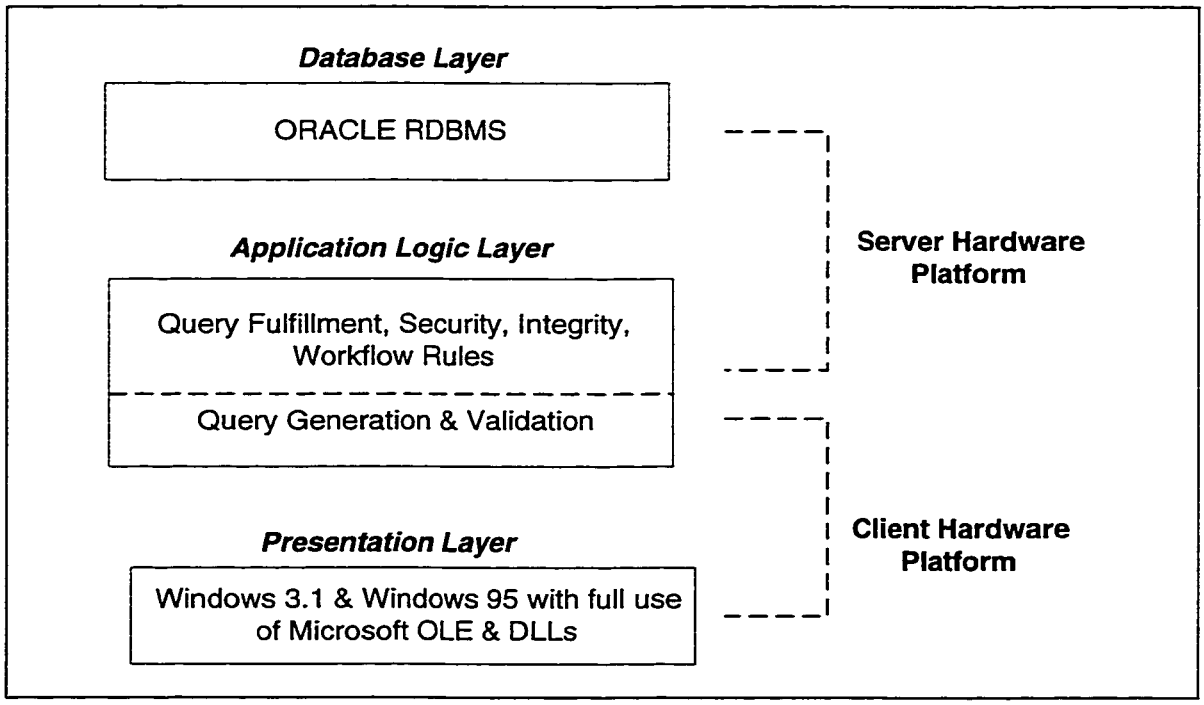


Figure 3-4 - MPAC 3-Tier Architecture

3.4.10 MPulse [MPu23], [DG24]

MPulse was designed to support Preventive maintenance programs for equipment, facilities, grounds and vehicles. The MPulse CMMS software graphical user interface was developed using Multimedia Toolbook by Asymetrix. Custom DLL's were developed in Pascal, to link to a Paradox database engine. To-date, the DLL's have mostly been rewritten in C++ and other database products are being considered. Although MPulse may move to a more open architecture, the current product has a closed architecture and does not support integration with MIS or other user systems. As far as development methodology, the customers genuine needs drive the technology used to meet them. The development process involves setting aggressive milestones and meeting them, using standard project management practices.

3.4.11 SOMAX Information Management System [SO25]

SOMAX has the normal assortment of functional features, as well. The types of establishments employing this system include pharmaceuticals, mining, healthcare, continuous and discrete manufacturing facilities and power plants. It consists of 12 integrated modules and integrates with CAD programs, graphics, word processors or spreadsheets from any of these modules. This product is written using Object Oriented Design Methodology in CA-Visual Objects (by Computer Associates). Data storage is in the industry standard xBase format, or optionally in Oracle.

3.4.12 Summary of CMMS Features Determined

Regardless of different levels of detail with which the CMMSs are described, and the somewhat differing module compositions, it is possible to roughly compare the features offered by some of the different vendors and present them in a tabular format, in Table 3-1, below. This table was based on the survey information contained in Annex A.

These findings are also shown graphically in Figure 3-5, which follows. Despite the relatively high level of granularity used to describe the functionality, it can be seen that the most prevalent or common features offered by CMMSs are shown below. Notably, all of these features appear (at least once) in Table 3-1, which contains two separate but varied assessments of typical CMMS functionality.

1. Personnel / Labour;
2. Work Orders;
3. Work Scheduling;
4. Inventory;
5. History / Analysis / Reporting;
6. Purchasing, and;
7. Security.

However, many CMMSs offer a wide compliment of various other features (some more obscure than others) to suit the requirements of specific customers or areas of the marketplace. Interestingly, none of the literature reviewed regarding the CMMS packages discussed in this paper, indicate that any of these products provide the capability to actively respond to exceeded limits or unsafe conditions detected. The capability to automatically implement a specific action (or response), other than merely to alert personnel to the problem could be of significant benefit. For example, if an overheated piece of equipment was automatically detected by a condition monitoring sensor, an appropriate response initiated by the CMMS might be to turn on a cooling fan in addition to implementing the necessary warnings or alarms.

Table 3-2 - Comparison of CMMS Features

	Equipment Management	Personnel / Labour	Work Orders	Purchasing	Inventory	PdM	History / Analysis / Reporting	CAD	Work Scheduling	Budgeting	Keys / Locks	Invoicing
Dynastar 2000	X	X	X	X	X	X	X	X	X	X		
MP2 Professional	X	X	X	X	X	X	X	X	X	X		
PMC for Windows		X	X	X	X		X	X	X			
WOT	X	X	X		X		X		X			
Bass Associates MMS	X	X	X	X	X		X		X			
Atlas Equipment Manager 2000	X	X	X	X	X	X	X		X			X
PlantLIFE Maintenance	X	X	X	X	X		X		X			
Enterprise MPAC	X	X	X	X	X	X	X		X			
MPulse		X	X	X	X		X	X			X	
SOMAX MMS	X	X	X	X	X		X	X	X			

	Project Tracking	Warranty Tracking	Electronic Commerce	EDMS	Security	Tools / TE / Calibration
Dynastar 2000					X	
MP2 Professional					X	
PMC for Windows		X			X	
WOT					X	
Bass Associates MMS					X	X
Atlas Equipment Manager 2000					X	
PlantLIFE Maintenance						
Enterprise MPAC	X	X	X	X	X	
MPulse						X
SOMAX MMS					X	

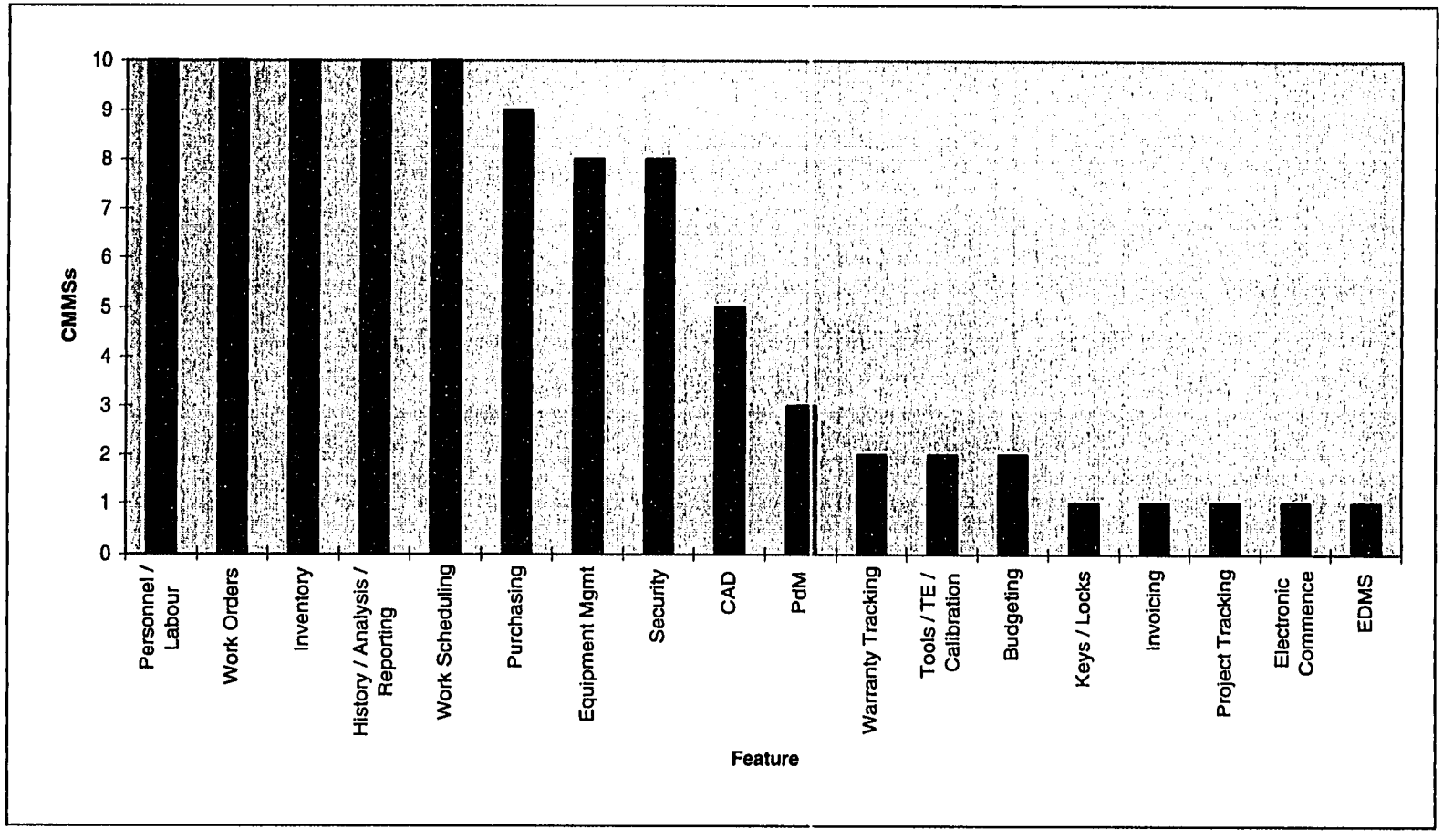


Figure 3-5 - CMMS Feature Prevalence

3.5 Summary

The CMMS is a popular type of commercial software tool; which is prevalent in many kinds of industries and facilitate higher productivity to the organizations that employ them. When properly implemented, the range of benefits to managers and technicians alone is extremely broad and the results in cost and time savings can be substantial. This chapter has examined the prevalence of features or functionality available from a wide range of commercial CMMSs on the market today.

It was difficult to readily compare these features from one package to another, as capabilities were often grouped or contained in different modules. Additionally, the titles of features sometimes varied from vendor to vendor and the products were described in different levels of detail. Nevertheless, it was possible to describe the most common features found in current CMMSs and identify many useful optional or specialized capabilities. This information, as presented in Table 3-2, may prove useful to the CMMS industry for the purposes of market research or product planning.

Vendors sometimes address different niches or specific customers, however, many of the characteristics described above (e.g. modularity, integration with other applications, capability to grow and adapt) are commonly heralded in much of the product specifications and other literature.

Although no specific cost data was gathered with respect to those surveyed, CMMSs vary widely in cost and complexity. However, an important associated factor, for which no information was readily available, was the cost of modifying or adapting these commercial products to suit specific customer needs. This, of course, is an extremely important consideration and one that would be very interesting to investigate further.

For any particular business environment or industry, the task of performing a survey, as was done in this case, can be an efficient and practical means of performing an initial "capture" of core functionality. Any software house preparing to enter a new market area can easily carry out this kind of exercise, from which can be based an initial set of events and system responses, in accordance with the CBD methodology.

4. Design of a Sample CMMS

4.1 Introduction

In Chapter 2, the CBD process was described in some detail, and Chapter 3 offered a description of CMMS products on the market today. In this chapter the CBD requirements analysis and design process, previously described, is "exercised" by developing an example CMMS design. The first step was to establish a set of requirements in the form of event responses and then to identify and list the associated activities. Following this, the design of the Coordinator and the set of activity executors were completed. The actual details of implementation are addressed in Chapter 5.

4.2 Requirements Analysis and Definition

4.2.1 Requirement Statements

It is a common practice in industry that the process of requirements analysis and definition is started by system engineer(s) and/or application specialist(s) interviewing the customer, and by reviewing extensive written descriptions of operational requirements and concepts. Requirements statements are normally then parsed from the narrative format, into a table, database or requirements management tool for easy maintenance and better traceability. To this end, Column 2 of Table 4-1 lists a set of CMMS requirement statements which were taken from available CMMS product literature (and also from Chapter 3). This list is far from exhaustive, and is meant to serve, only, as a sample of modern CMMS functionality. These requirements statements now serve as the basis for this sample CMMS design.

It is important to emphasize, here, that the ability to easily translate the requirements statements directly into events, as was done in Table 4-1, is a powerful feature of CBD. This firm requirements "link" back to any narrative description of the system requirements provides easily understood proof that the design will match the user needs.

4.2.2 Events and Responses

(Step 1 of the CBD Process - Figure 2-1). In most CMMS's, the primary source of input commands and data is from the keyboard. In this example CMMS as well, most events, or inputs to this system consist of the keyboard entries (Column 3 of Table 4-1). Other events, e.g. line 15 Time Trigger or Meter Trigger are automatically generated. Responses (Column 4 of Table 4-1), involve interaction with either the CMMS database or an Excel spreadsheet, with system outputs occurring as displayed information in the dialog boxes, or as printouts.

For ease of grouping, and subsequent activity executor design, these statements were arranged into categories of capability, which are common in the CMMS industry. These consist of:

- Equipment Management Requirements;
- Inventory Requirements;
- Work Order Requirements;
- Purchase Order Requirements;
- Personnel Information Requirements, and;

Note that not all requirements statements in Table 4-1 translate into event responses, some only impact the database design. Furthermore, only a subset of the event responses (the shaded ones) identified as requirements are actually included in the design and subsequent implementation. It was a subjective decision, that this was sufficient functionality to demonstrate that the CBD technology is a practical means of designing a working system which could be easily tailored.

In Section 2.3.3, the technique of using Atomic and Combined responses was introduced and discussed, as a way to tailor and integrate event responses. Lines 10 and 28 in Table 4-1 are examples of tailoring with atomic and combined responses. The combined response in Line 10, "Create WO" is comprised of three atomic responses: "Get Equipment Information", "Get Employee Information" and "Save Work Order". The Combined response in line 28, "Create PO" consists of atomic responses "Purchase Order Submit" and "Purchase Order Confirm/Print". The complete set of system requirements captured for the CMMS example is contained in Table 4-1, which spans the next four pages.

Table 4-1 - Sample Requirements Analysis

No.	Narrative (System Level) Requirement Statements	Events	Responses
1.	**EQUIPMENT MANAGEMENT REQUIREMENTS**		
2.	Track maintenance activities by type of repairs and reason for breakdown	• *[Impacts database design]* • UI: Generate Fault Type Report or • UI: Generate Repair Type Report	• *[Impacts database design]* • Sys: Generate Fault Type Report or • Sys: Generate Repair Type Report
3.	CMMS shall allow different user types	• Login : Selection of User Type	
4.	**INVENTORY REQUIREMENTS**		
5.	*Add and delete inventory items from the user interface*	• *UI: Add Inventory Item or* • *UI: Delete Inventory Item*	• *Sys: Add Inventory Item or* • *Sys: Delete Inventory Item*
6.	Produce inventory report	• UI: Generate Inventory Report	• Sys: Generate Inventory Report
7.	*Main inventory information includes:* • *Serial no,* • *part no,* • *manufacturers part no,* • *vendors part no,* • *cost,* • *locations and* • *quantity on hand.*	• *[Impacts database design and Inventory Report]*	*N/A*
8.	Identify separate reorder points for each inventory item based on usage history.	• UI: Edit Inventory Item	• Sys: Edit Inventory Item
9.	**WORK ORDER REQUIREMENTS**		
10.	*Create a work order from the user interface*	• *UI: Create WO*	• *Sys: Create WO* *Implemented as a Combined Event Responses (Get Equipment Information, Get Employee Information, Save Work Order)*
11.	*Display a work order at the user interface*	• *UI: Display WO*	• *Sys: Display WO*
12.	Close a Work Order	• UI: Close WO	• Sys: Close WO
13.	Automatically deleting items from inventory, when work orders closed (the WOs must indicate the parts used in performing the work).	• UI: Close WO	• Amend Inventory Quantity on WO Closure
14.	User interface allows (Equipment Profile Screen) users to list and view all WOs for a specific piece of equipment.	• UI: Display WOs (Eqpt Specific)	• Sys: Display WOs
15.	Automatic generation of work orders resulting from the following triggers: • time (daily, weekly, monthly, semi-annually, annually) • meter,	• Sys: Time Trigger or • Sys: Meter Trigger or • Sys: Variable	• Sys: Create WO

No.	Narrative (System Level) Requirement Statements	Events	Responses
	• time or meter, • variables,		
16.	PM procedures are contained in the CMMS database and can be entered into a WO. Procedures can be specific or generic to a piece of equipment. They can then be tied to one or more pieces of equipment as optional or mandatory. If mandatory, they are printed out as part of the WO package, any time a WO is generated for that equipment.	• UI: Add PM Procedure or • UI: Edit PM Procedure • *[Impacts database design]*	• Sys: Add PM Procedure or • Sys: Edit PM Procedure • *[Impacts database design]*
17.	Following is a list of information initially entered into the WO: • *Detailed task description* • Tools, parts and part nos. whether parts are in stock • *Equipment and equipment location* • *Employee performing tasks* • *Date and time work was assigned and completed* • Shift hours • Name of supervisor to contact if necessary, safety procedures and personal protective equipment. Required • Assign and track labour costs (by craft and task code)	• *[Input performed when WO created]* • *[Impacts WO screen layout and database design.]*	
18.	When work completed, the following detail is recorded in the WO: • Employee(s) who performed the work, • parts used, • nature of deficiencies encountered • CM and PM measures taken (recorded automatically in the Equipment History), • Changes to original WOs • Completion date	• UI: Edit WO	Sys: Edit WO
19.	• The CMMS shall provide the capability to view all WOs assigned to a person	• UI: Generate WO Assignment Report	• Sys: Generate WO Assignment Report
20.	Update WO automatically on receipt of material / parts	• UI: Add Inventory Item	• [As a function of the database query which generates the WO, the WO when opened every time will automatically show the new parts available in he Inventory after it has been added.]
21.	Reserve materials for WOs	UI: Reserve Inventory Item	Sys: Reserve Inventory Item
22.	CMMS user interface shall allow viewing of the following from a WO screen: • standard parts list for equipment to be worked on, • parts on-hand, • parts reserved/committed, • parts on-order, • *repair history of equipment to be worked on,* • repair history of components to be worked on.	• *[Impacts database and WO screen design]*	*N/A*

No.	Narrative (System Level) Requirement Statements	UI Events	System Responses
	• parts in transit.		
23.	*Select new, reconditioned or used parts on WO*	*UI: Create WO or* UI: Edit WO	*Sys: Create WO or* Sys: Edit WO
24.	*Save a WO*	*UI: Save WO*	*Sys: Save WO*
25.	*Update an Existing WO*	*UI: Update a WO*	*Sys: Update WO*
26.	*Get the status of a WO*	*UI: Get WO Status*	*Sys: Get WO Status*
27.	**PURCHASE ORDER REQUIREMENTS**		
28.	Display, *create, modify* or delete a PO from the User Interface	• UI: Display PO or • *UI: Create PO or* • *UI: Edit PO or* • UI: Delete PO	• Sys: Display PO or • *Sys: Create PO (implemented as a set of Combined Event Responses (Purchase Order Submit, Purchase Order Confirm/Print)* or • Sys: Delete PO
29.	Automatically generate and print POs as parts quantities fall below/reach user defined minimum to maintain desired stock level.	• UI: Delete Inventory Item (At Min. Stock Level)	Sys: - Delete Inventory Item and - Create PO and - Print PO (Combined Response)
30.	CMMS tracks current POs and purchase history.	• UI: Generate Purchase History Report	• Sys: Generate Purchase History Report
31.	Tracking of equipment by part no. & serial no. System then tracks movement of serial numbered components to and from inventory.	• UI: Generate Equipment Location Report • *[Impacts database design]*	• Sys: Generate Equipment Location Report
32.	Equipment usage data is maintained by the CMMS	• *[Impacts database design]* • UI: Generate Eqpt Usage Report	
33.	**PERSONNEL INFORMATION REQUIREMENTS**	•	
34.	Maintain complete employee records including wage and craft information	• UI: Create New Employee Record or • UI: Edit Employee Record or • UI: Delete Employee Record or • UI: Generate Employee History Report	• Sys: Create New Employee Record or • Sys: Edit Employee Record or • Sys: Delete Employee Record or • Sys: Generate Employee History Report
35.	Have multiple wage codes for employees performing different jobs or who work different shifts, providing the flexibility to charge appropriate wage for a given task	• *[Impacts database design and WO format]*	
36.	Track hours per employee or per wage code, sick time and overtime.	• UI: Generate Employee Time Report • *[Impacts database design and Employee History Report]*	• Sys: Generate Employee Time Report

No.	Narrative (System Level) Requirement Statements	Events	Responses
37.	Generate histories of training requirements, training completed, and employee certifications.	• UI: Generate Employee Training Report	• Sys: Generate Employee Training Report
38.	Maintain records of employee performance	• *[Impacts database design]* • UI: Employee Performance Report	• Sys: Employee Performance Report
39.	CMMS shall record, track and report:: • Equipment downtime • Failures	• UI: Generate Equipment Downtime Report or • UI: Generate Equipment Failures Report"	• Sys: Generate Equipment Downtime Report or • Sys: Generate Equipment Failures Report
40.	CMMS shall provide printing capability for: • WOs • *POs* • Reports	• UI: Print WO • *UI: Print PO* • UI: Print Inventory Report • UI: Print Purchase History Report • UI: Print Eqpt Location Report • UI: Print Eqpt Usage Report • UI: Print Employee Time Report • UI: Print Employee History Report • UI: Print Employee Training Report • UI: Print Employee Performance Report • UI: Print Eqpt Downtime Report • UI: Print Eqpt Failures Report	• Sys: Print WO • *Sys: Print PO* • Sys: Print Inventory Report • Sys: Print Purchase History Report • Sys: Print Eqpt Location Report • Sys: Print Eqpt Usage Report • Sys: Print Employee Time Report • Sys: Print Employee History Report • Sys: Print Employee Training Report • Sys: Print Employee Performance Report • Sys: Print Eqpt Downtime Report • Sys: Print Eqpt Failures Report
41.	The CMMS automatically captures the dates for the next weekly and monthly maintenance routines.	• Sys: Compare today's date to next (weekly or monthly) maintenance dates	• Next maintenance dates internally updated by 1 week or 1 month (if today's date > maintenance date).
42.	On input of location and equipment type, the CMMS shall display the serial numbers of the equipment items, plus dates for the next weekly and monthly routines.	• UI: Enter location and equipment type	• Sys: Display serial numbers and dates for next weekly routine; • Sys: Display serial numbers and dates for next monthly routine

4.2.3 List the Activities for Each Response

(Step 2 of the CBD Process - Figure 2-1). The detail of the responses articulated above is really at the very highest level, and it is now necessary to disclose, more precisely, the low level make-up of these responses by identifying the activities which must be carried out to implement them. A first step that could have been followed here might have been to merely list all the system activities and then correlate them with responses; instead the author found it more practical to identify activities on a "response by response" basis, as shown in Table 4-2 (pages 49 - 54).

In addition to identifying the activities, the order in which the activities are to be executed is also defined in Table 4-2. As in most software development programs, there was in this case as well, a need to revisit the requirements periodically and "iterate" or modify them during the design phase. This rigor helped to ensure that traceability between the requirements and design was maintained - there is a chance otherwise that the requirements could get "left behind" the design, as the experience level of a design team grows. If the requirements and design are not kept consistently "in tune" the risks of the product not being accepted by the customer dramatically increases.

In Table 4-2 below, each response listed in Table 4-1 is restated along with the constituent activities. Notice that many activities can and do occur in more than one response ("Open CMMS DB and Close CMMS DB, for example). Note, also, that here the activities are listed sequentially. This was essentially due to the simplicity of the responses and because the platform used uses a single processor. For multi-processor systems, it would have been possible to design concurrent execution of suitable activities. Again, the shaded responses are the ones which have been implemented in the example described in Chapter 5.

Table 4-2 - Identification of the Activities

No.	Response	Activities
1.	*Add Inventory Item*	• *Open CMMS DB* • *Check - Part Number not in "Inventory" Table* • *Create Record in "Inventory" Table* • *Close CMMS DB*
2.	Edit Inventory Item	• Open CMMS DB • Check - Part Number in "Inventory" Table • Get Record From "Inventory" Table • Put Record Into "Inventory" Table • Close CMMS DB
3.	*Delete Inventory Item*	• *Open CMMS DB* • *Check - Part Number in "Inventory" Table* • *Delete Record in "Inventory" Table* • *Close CMMS DB*
4.	Generate Inventory Item Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Inventory" Table • Compose Inventory Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
5.	*Get Equipment Information*	• *Open CMMS DB* • *Check if Serial No. Exists in "Operational_Equipment" Table*

No.	Response	Activities
		• *Get Equipment Information* • *Close CMMS DB*
6.	*Get Employee Information*	• *Open CMMS DB* • *Check if Trade Name Exists in "Crafts_Trades" Table* • *Check if Trade Level Exists in "Crafts_Trades" Table* • *Get Names of Qualified Employees* • *Close CMMS DB*
7.	*Save WO*	• *Open CMMS DB* • *Save WO Data* • *Close CMMS DB*
8.	*Display WO*	• *Open CMMS DB* • *Get Record From "Work_Orders" Table* • *Close CMMS DB*
9.	Edit WO	• Open CMMS DB • Get Record From "Work_Orders" Table • Put Record Into "Work_Orders" Table • Close CMMS DB
10.	Close WO	• Open CMMS DB • Get Record From "Work_Orders" Table • Put Record Into "Work_Orders" Table • Close CMMS DB
11.	*Update WO*	• *Open CMMS DB* • *Update WO Information in CMMS DB* • *Close CMMS DB*
12.	*Update and Get WO Status*	• *Open CMMS DB* • *Update Status - If WO Started* • *Update Status - If WO Started Late* • *Update Status - If WO Completed* • *Update Status - If WO Completed Late* • *Update Status - Started* • *Update Status - Completed* • *Get WO Status* • *Close CMMS DB*
13.	Add PM Procedure	• Open CMMS DB • Create Record in "PM_Type" Table • Close CMMS DB
14.	Edit PM Procedure	• Open CMMS DB • Get Record From "PM_Type" Table • Put Record Into "PM_Type" Table • Close CMMS DB
15.	Generate WO Assignment Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Work_Orders" Table • Get Record from "Employee" Table • Compose WO Assignment Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
16.	Amend Inventory Quantity on WO Closure	• Open CMMS DB • Get Record From "Work_Orders" Table • Put Record Into "Work_Orders" Table • Get Record from "Inventory" Table • Open EXCEL Inventory Worksheet • Calculate new IQty • Close EXCEL Inventory Worksheet

No.	Response	Activities
		• Update Record in Inventory Table • Close CMMS DB
17.	Reserve Inventory Item	• Open CMMS DB • Get Record From "Inventory" Table • Put Record Into "Inventory" Table • Close CMMS DB
18.	Display PO	• Open CMMS DB • Get Record From "Purchase_Orders" Table
19.	*Create Purchase Order* *Purchase Order Submit* *Purchase Order Confirm/Print*	*(Purchase Order Submit)* *(Get Vendor Information)* • *Open CMMS DB* • *Check If Vendor Name In "Vendors" Table* • *Get Record From "Vendors" Table* • *Close CMMS DB* *(Get Cost Data)* • *Open CMMS DB* • *Get Part Cost (up to 12 times)* • *Close CMMS DB* *(Calculate Purchase Order Totals)* • *Open Purchase Order Form* • *Calculate Purchase Order Totals* • *Close Purchase Order Form* *(Purchase Order Confirm/Print)* *(Update Vendor Data)* • *Open CMMS DB* • *Check if Vendor Name Exists in "Vendors" Table* • *Create Record in "Vendors" Table (or)* • *Update Record in "Vendors" Table* • *Close CMMS DB* *(Save Purchase Order)* • *Open CMMS DB* • *Save Purchase Order Record in "Purchase_Orders" Table (up to 12 times)* • *Close CMMS DB* *(Print Purchase Order)* • *Open Purchase Order Form* • *Calculate Purchase Order Totals* • *Print Purchase Order Form* • *Close Purchase Order Form*
20.	Edit PO	• Open CMMS DB • Get Record From" Purchase_Orders" Table • Put Record Into "Purchase_Orders" Table • Close CMMS DB
21.	Delete PO	• Open CMMS DB • Delete Record in "Purchase_Orders" Table • Close CMMS DB
22.	Create Purchase Order at MSL Generate Purchase Order Get Vendor Information Get Cost Data Calculate Purchase Order Totals Purchase Order Confirm/Print Update Vendor Data Save Purchase Order Print Purchase Order	(Generate Purchase Order) (Get Vendor Information) • Open CMMS DB • Get Record From "Vendors" Table • Close CMMS DB (Get Cost Data) • Open CMMS DB • Get Part Cost • Close CMMS DB (Calculate Purchase Order Totals) • Open Purchase Order Form • Calculate Purchase Order Totals • Close Purchase Order Form

No.	Response	Activities
		(Purchase Order Confirm/Print) (Update Vendor Data) • Update Vendor Data • Check if Vendor Name Exists in "Vendors" Table • Create Record in "Vendors" Table • Update Record in "Vendors" Table • Close CMMS DB (Save Purchase Order) • Open CMMS DB • Save Purchase Order Record in "Purchase_Orders" Table • Close CMMS DB (Print Purchase Order) • Open Purchase Order Form • Calculate Purchase Order Totals • Print Purchase Order Form • Close Purchase Order Form
23.	Generate Purchase History Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from in "Purchase_Orders" Table • Get Record from "Vendors" Table • Compose Purchase History Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
24.	Generate Equipment Location Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Operational_Equipment" Table • Compose Equipment Location Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
25.	Create New Employee Record	• Open CMMS DB • Create Record In "Employee" Table • Close CMMS DB
26.	Edit Employee Record	• Open CMMS DB • Get Record From "Employee" Table • Put Record Into "Employee" Table • Close CMMS DB
27.	Delete Employee Record	• Open CMMS DB • Delete Record in "Employee" Table • Close CMMS DB
28.	Generate Employee History Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Employee" Table • Compose Employee History Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
29.	Generate Employee Time Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Employee" Table • Get Record from "Work_Orders" Table • Compose Employee Time Report Record

No.	Response	Activities
		• Put record in ReportFile • Close CMMS DB • Close ReportFile
30.	Generate Employee Training Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Employee" Table • Compose Employee Training Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
31.	Employee Performance Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Employee" Table • Compose Employee Performance Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
32.	Generate Equipment Downtime Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Operational_Equipment" Table • Compose Equipment Downtime Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
33.	Generate Equipment Failures Report	• Open CMMS DB • Create ReportFile • Initialize ReportFile • Get Record from "Operational_Equipment" Table • Compose Equipment Location Report Record • Put record in ReportFile • Close CMMS DB • Close ReportFile
34.	Print WO	• Open CMMS DB • Create Printable WO Report Form • Print Inventory Report Form • Close CMMS DB
35.	Print Inventory Report	• Open CMMS DB • Create Printable Inventory Report Form • Print Inventory Report Form • Close CMMS DB
36.	Print Purchase History Report	• Open CMMS DB • Create Printable Purchase History Report Form • Close CMMS DB • Print Purchase History Report Form
37.	Print Eqpt Location Report	• Open CMMS DB • Create Printable Operational Equipment Report Form • Close CMMS DB • Print Operational Equipment Report Form
38.	Print WO Assignment Report	• Open CMMS DB • Create Printable WO Assignment Report Form • Close CMMS DB • Print Operational Equipment Report
39.	Print Employee Time Report	• Open CMMS DB

No.	Response	Activities
		• Create Printable Employee Time Report Form • Close CMMS DB • Print Employees Report Form
40.	Print Employee History Report	• Open CMMS DB • Create Printable Employees Report Form • Close CMMS DB • Print Employees Report Form
41.	Print Employee Training Report	• Open CMMS DB • Create Printable Employees Report Form • Close CMMS DB • Print Employees Report Form
42.	Print Employee Performance Report	• Open CMMS DB • Create Printable Employees Report Form • Close CMMS DB • Print Employees Report Form
43.	Print Eqpt Downtime Report	• Open CMMS DB • Create Printable Operational Equipment Report Form • Close CMMS DB • Print Operational Equipment Report Form
44.	Print Eqpt Failures Report	• Open CMMS DB • Create Printable Operational Equipment Report Form • Close CMMS DB • Print Operational Equipment Report Form
45.	Next maintenance dates internally updated by 1 week or 1 month (if today's date > maintenance date).	Internal database macro
46.	Display serial numbers and dates for next weekly routine	• Open CMMS DB • Display Next Weekly Maintenance Date from Work Orders Table • Close CMMS DB
47.	Display serial numbers and dates for next monthly routine	• Open CMMS DB • Display Next Monthly Maintenance Date from Work Orders Table • Close CMMS DB

With the activities identified, the requirements are now in a sufficiently organized and detailed form that they give a comprehensive description system functionality.

4.3 Design Phase

4.3.1 Coordinator Design

(Steps 3 and 4 of the CBD Process - Figure 2-1). The first part of the design, was to define the nature of the high level architecture. It was decided that only one Coordinator would be necessary (as shown in Figure 1-2, different configurations, with more than one Coordinator are possible). Only one Coordinator was used because the size of the example CMMS system and number of response managers was not overwhelmingly large. Additionally, the complexity of the functionality in the design was not considered significant enough to warrant more than one level of coordination. For example, if it would have been necessary to have large functional capabilities working concurrently, more than one level of coordination could be have been warranted.

A database to store and retrieve data would also be required. Figure 4-1 shows the initial high level architecture for the CMMS.

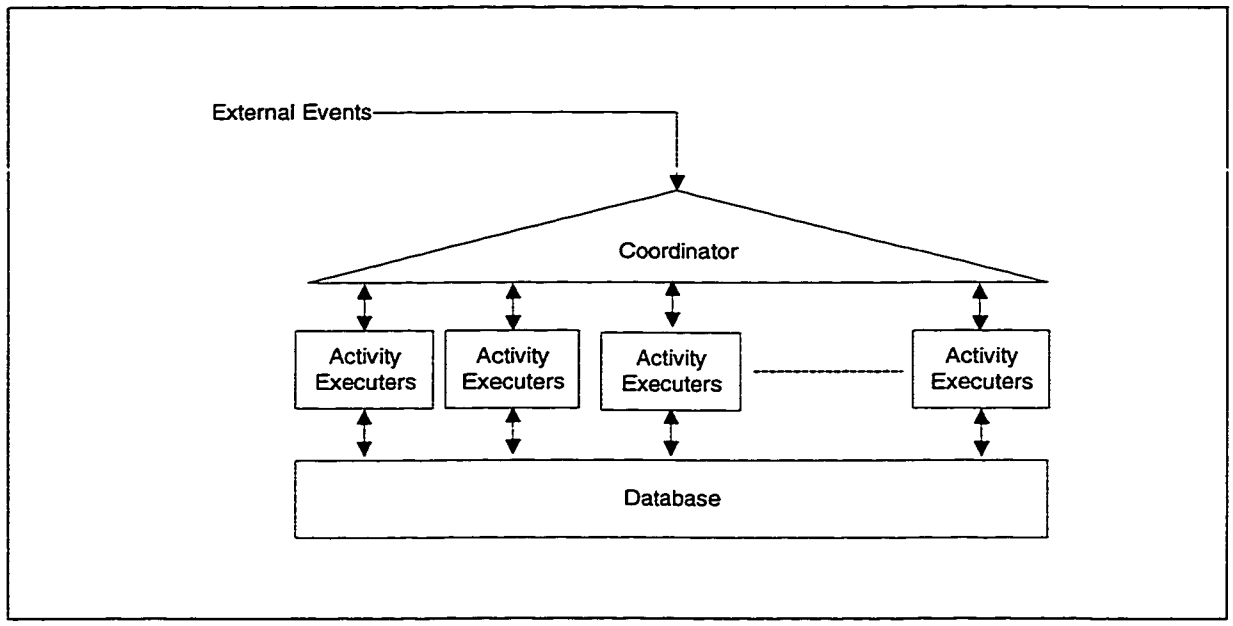


Figure 4-1 - Initial Example CMMS Architecture

4.3.1.1 Response Manager Design

The Process Activity Language (PAL) method for formalizing the response design in the Coordinator could have been used here, but as the complexity of the responses was relatively low, (and considering that a single processor system is being used) the activities were designed to execute sequentially. For this reason, the decision was made to use pseudocode instead. This pseudocode used to design the response managers is provided on this and the next three pages - only the implemented responses shown in Table 4-2 are included.

Add Inventory Item

```
Process Add_Inventory_Item
        Sequence
                %Assume that the "CMMS Screens" Menu has been enabled
                display Inventory Screen/Dialogue Box
                enter details of inventory item to be added
                wait_for_add_inventory_item_OK_clicked
                open CMMS database
                check if MPN exists in database
                if MPN exists
                        close CMMS Database
                Else
                        assign MPN, MPTD, ISL, MSL, ETNO, ATNO, SATNO, SN, IQTY, QR, N/U/R as new
                        entry in the Inventory Table
                check create new entry return code
                if create OK
                        display create OK message
                Else
                        display create fail message
                End If
                close CMMS DB
        End sequence
End
```

Delete Inventory Item

```
Process Delete_Inventory_Item
        Sequence
                %Assume that the "CMMS Screens" Menu has been enabled
                display Inventory Screen/Dialogue Box
                enter MPN of item to be deleted
                wait_for_delete_inventory_item_OK_clicked
                Open CMMS Database
                check if MPN exists in database
                if MPN not exist
                        close CMMS Database
                Else
                        delete record
                        display delete success message
                Else
                        display delete fail message
                End If
                close CMMS Database
        End sequence
End
```

Create Purchase Order

```
Sequence
        %Assume the "CMMS Screens" Menu has been enable
        display Purchase Order Screen/Dialogue Box With POID
        %Here, the user enters in the POIB, VNa, ETNo, PQty
        Case
                (Selection==Submit_Purchase_Order)
                        Open CMMS Database
                        Enter Vendor Name, Part No., quantity required
                        Check if Vendor Name in CMMS DB
                        If Vendor Name Not exist
                                Close CMMS DB
                        Else
                                Get Vendor Address and Phone No. from CMMS DB
                                Close CMMS DB
                        Open CMMS DB
                        Get Part cost data
                        Close CMMS DB
                        Open Purchase Order Form
                        Send data to Purchase Order Form and calculate totals
```

```
                    Display totals
        Case
            (Selection==Confirm/Print Purchase Order)
                    Open CMMS DB
                    Check if Vendor Name in CMMS DB
                    If Vendor Name Not exist
                            Save entered Vendor Data in CMMS DB
                    Else
                            Update Vendor Data in CMMS DB
                            Close CMMS DB
                    Open CMMS DB
                    Save WO data in CMMS DB
                    Close CMMS DB
                    Open Purchase Order Form
                    Send Purchase Order Data to Purchase Order Form and calculate totals
                    Print Purchase Order Form
                    Close Purchase Order Form
            End Sequence
        End
```

Work Order

```
    Sequence
        %Assume the "CMMS Screens" Menu has been enable
        display Work Order Screen/Dialogue Box
        %Here, the user enters in the equipment Serial No.
        Case
            (Selection==Get Equipment Information)
                    Open CMMS Database
                    Check if serial No. exists in the Operational Equipment Database
                    If serial No. Not exist
                    Close CMMS DB
                    Else
                    Get Equipment Type, location, description of maintenance performed
                    Close CMMS DB
            (Selection==Get Employee Information)
                    Open CMMS DB
                    Enter Trade Name and Trade Level
                    Check if Trade Name Exists
                    if Trade Name exists
                    Check if Trade Level exists
                    Else Close CMMS DB
                    If Trade Level exists
                    Get names of employees matching Trade Name, Trade Level and Location of
                    equipment
                    Close CMMS DB
            (Selection==Save WO)
                    Save WO Information
                    Close CMMS DB
            (Selection==Update WO)
                    Open CMMS DB
                    Enter WO No.
                    Get WO
                    Update displayed information
                    Save WO Information
                    Close CMMS DB
            (Selection==Get WO)
                    Open CMMS DB
                    Enter WO No.
                    Get WO
                    Save WO Information

            End Sequence
        End
```

Get Work Order Status

```
    Sequence
        %Assume the "CMMS Screens" Menu has been enable
        display Work Order Status Screen/Dialogue Box
```

```
%Here, the user enters in the equipment Serial No.
        (Selection==Get WO Status)
                Open CMMS Database
                Update Status - If WO Started
                Update Status - If WO Started Late
                Update Status - If WO Completed
                Update Status - If WO Completed Late
                Update Status - Started
                Update Status - Completed
                Get WO Status
                Close CMMS DB
        End
```

Get Next Weekly Maintenance Dates

```
Sequence
        %Assume the "CMMS Screens" Menu has been enable
        display Next Maintenance Screen/Dialogue Box
        %Here, the user enters in Location and Equipment Type.
                (Selection==Get Next Weekly Maintenance Dates)
                        Open CMMS Database
                        Display Serial Number(s) & Date(s)
                        Close CMMS DB
                End
```

Get Next Monthly Maintenance Dates

```
Sequence
        %Assume the "CMMS Screens" Menu has been enable
        display Next Maintenance Screen/Dialogue Box
        %Here, the user enters in Location and Equipment Type.
                (Selection==Get Next Monthly Maintenance Dates)
                        Open CMMS Database
                        Display Serial Number(s) & Date(s)
                        Close CMMS DB
                End
```

4.3.2 Activity Executor Design - Group Activities in Execution Modules

(Step 5 of the CBD Process - Figure 2-1). In deciding how to group the activities into activity executors, it first made sense to do this in terms of CMMS functional areas (specific application areas). The execution modules implemented correspond to some of the more common CMMS functional areas as shown in Figure 3-5.

At this point, a review of all the activities was performed to determine what functional areas to use. The most appropriate ones were determined to be those listed in the left-hand column of Table 4-3, below. In Section 4.2.2, the initial requirements statements were grouped into categories of capability similar to some of the Activity Executor functional areas, shown below. Although there is no deliberate relationship established between these two groups, the categories of capability identified earlier offer a preliminary glimpse at what the potential execution modules will be.

It was also decided to re-visit Au Yang's design and determine if any of it could be re-used for the CMMS design. There was some commonality between the two systems as indicated in Table 4-3. Those executors which were actually implemented in the CMMS example are shown in shaded text.

Table 4-3 - Similarity in Activity Executors

CMMS Activity Executors	Activity's Sales Support System Activity Executors
Inventory Data Management	*Inventory Database Management*
Vendor Data Management	*Customer Database Management*
Purchase Order Data Management	*Sales Database Management*
Purchase Order Management	*Computational Management, Sales Form Printing*
Work Order Management	
Employee Management	
Equipment Data Management	
PM Data Management	
Report Generation	Report Generation
Printing Management	

In Table 4-4, below, the activities described as part of the responses in Table 4-2, are assigned to specific activity executors. Although not always the case, generally, activities which accessed the same CMMS Database table(s) were assigned to the same executor. One advantage sought in the design of execution modules is that of database consistency control. This is achieved if the data is accessed one response at a time. As different executors can be functioning simultaneously, it makes sense to keep the chances to a minimum, that the same database table entry could be accessed by more than one response at the same point in time and provide inconsistent values.

The shaded activity executors and managed activities have been implemented in the CMMS example.

Table 4-4 - Assignment of Activities to Executors

Activity Executors	Managed Activities
Inventory Data Management	• *Open CMMS DB* • *Check - Part Number not in "Inventory" Table* • *Close CMMS DB* • *Check - Part Number in "Inventory" Table* • *Delete Record in "Inventory" Table* • *Put Record Into "Inventory" Table* • Get Record From "Inventory" Table • Update Record in Inventory Table • Open EXCEL Inventory Worksheet • Close EXCEL Inventory Worksheet • Calculate new Inventory Quantity
Vendor Data Management	• *Open CMMS DB* • *Close CMMS DB* • *Check if Vendor Name in "Vendors" Table* • *Create Record in "Vendors" Table* • *Get Record From "Vendors" Table* • *Update Record in "Vendors" Table*
Purchase Order Data Management	• *Open CMMS DB* • *Close CMMS DB* • *Get Record From "Purchase_Orders" Table* • *Get Part Cost* • *Save Purchase Order Record Into "Purchase_Orders" Table* • Delete Record From "Purchase_Orders" Table • Put Record Into "Purchase_Orders" Table

Activity Executors	Managed Activities
Purchase Order Management	• *Open Purchase Order Form* • *Populate Form and Calculate Purchase Order Totals* • *Close Purchase Order Form* • *Print Purchase Order Form*
Work Order Management	• *Open CMMS DB* • *Check if Serial No. in "Operational_Equipment" Table* • *Get Equipment Information* • *Check if Trade Name Exists in "Crafts_Trades" Table* • *Check if Trade Level Exists in "Crafts_Trades" Table* • *Get Names of Qualified Employees* • *Save WO Data* • *Update WO* • *Update Status - If WO Started* • *Update Status - If WO Started Late* • *Update Status - If WO Completed* • *Update Status - If WO Completed Late* • *Update Status - Started* • *Update Status - Completed* • *Get WO Status* • *Close CMMS DB* • Get WO From "Work_Orders" Table • Get Record From "Work_Orders" Table Put Record Into "Work_Orders" Table • *Display Serial Numbers and Dates for Next Weekly Maintenance Routines* • *Display Serial Numbers and Dates for Next Monthly Maintenance Routines*
Employee Management	• Open CMMS DB • Get Record From "Employee" Table • Put Record Into "Employee" Table • Delete Record in "Employee" Table • Create Record In "Employee" Table • Close CMMS DB
Equipment Data Management	• Open CMMS DB • Close CMMS DB • Get Record from "Operational_Equipment" Table
PM Data Management	• Open CMMS DB • Create Record in "PM_Type" Table • Get Record From "PM_Type" Table • Put Record Into "PM_Type" Table • Close CMMS DB
Report Generation	• Create Report File • Initialize Report File • Compose Employee Time Report • Compose Employee History Report • Compose Employee Training Report • Compose Employee Performance Report • Compose Inventory Report • Compose WO Assignment Report • Compose Purchase History Report • Compose Equipment Location Report • Compose Equipment Failures Report • Compose Equipment Downtime Report • Put report in Report File • Close Report File
Printing Management	• Create Printable Employee Time Report Form • Create Printable Employee History Report Form • Create Printable Employee Training Report Form • Create Printable Employee Performance Form • Create Printable Inventory Report Form

Activity Executors	Managed Activities
	• Create Printable WO Assignment Report Form • Create Printable Purchase History Report Form • Create Printable Equipment Location Report Form • Create Printable Equipment Failures Report Form • Create Printable Equipment Downtime Report Form • Print Employee Time Report Form • Print Employee History Report Form • Print Employee Training Report Form • Print Employee Performance Form • Print Inventory Report Form • Print WO Assignment Report Form • Print Purchase History Report Form • Print Equipment Location Report Form • Print Equipment Failures Report Form • Print Equipment Downtime Report Form

4.4 Design of the Example CMMS Through Reuse and Tailoring

The example CMMS was actually tailored in different ways. Originally, the design was modified from Au Yang's Sales Support System. Au Yang's system possessed a simple CBD structure (i.e. a Coordinator that could be readily modified and embellished, activity executors that interacted with a database in much the same way that a CMMS would) which was easily modeled to create the high level CMMS design. The ability of a CBD based system to be transformed in this way, into a completely new application is a powerful capability.

The detailed CMMS example system design grew and developed over a period of time to include additional functionality, and it was therefore necessary to modify existing responses, and add entirely new responses and execution modules. The task of tailoring the response design in the Coordinator involved altering or creating new responses resulted in the reuse of existing activities (some of which were re-used extensively, e.g. "Open CMMS DB" and "Close CMMS DB"). When it was required to add newly defined activities to the activity executors, the task involved, merely to determine which module to add them to, and in all cases, this choice was obvious because of the clear functionally defined boundaries of the executors. An additional aspect of the design which allowed efficient implementation was that there were no occasions when the rest of the system design had to be altered due to regression testing as the responses and activity executors acted independently.

Lastly, an important aspect of tailorability, introduced in this thesis, was the idea of undertaking a lower level "building block" approach to developing more elaborate responses form existing ones. This was done through the concept of "Atomic" and "Combined" responses as was discussed earlier in Sections 2.3.3 and 4.2.2. Although only limited examples of this technique were actually developed in the CMMS example, the potential to easily create substantial new designs with this simple technique is significant and should be further investigated in future work.

The ways in which responses can be combined is only limited by the need, the availability of the responses and the capability of the designer. Another example of a combined response which could have been implemented in the CMMS example is shown in Line 22 of Table 4-2. Here a Purchase Order (PO) is generated when minimum stock level (MSL) is reached. The steps, which could have been readily automated are:

- Get the Vendor Information (for the part which has reached MSL) from the database;
- Get the Cost data from the database;
- Calculate the PO Totals;
- Update Vendor Data (if necessary from the keyboard;
- Save the PO, and
- Print the PO.

One of the important advantages of being able to tailor a system such as the Example CMMS is that it is easier to accommodate the needs of different users in terms of how sophisticated to make the responses. On one hand, a user may prefer to interact closely with the system, however, a different user may require the system to operate in a more automated way. With CBD, the complexity of the combined responses can be tailored easily to accommodate these different preferences.

5. System Implementation

5.1 Implementation Methodology

System implementation corresponds to Steps 6, 7 and 8 of the CBD Process as shown in Figure 2-1. The code developed by Au Yang was used as a starting point to create the example CMMS. However, it was extensively modified and added to, to develop example CMMS system. MS Visual C++ was used for the development of the Local UI and the Coordinator while the activity executors were created using MS Visual Basic.

The parts of Au Yang's system (Figure 2-4) which were reused to develop the CMMS were the User Interface (UI), the Coordinator Wrapper, the Coordinator and the Execution Modules. Au Yang's UI contained dialog boxes for "Add Item" and "Sales Order Entry". These two screens were easily modified to the CMMS "Add Inventory Item" and "Purchase Order" screens, respectively. Four new dialog boxes were developed, subsequently - Delete Inventory Item (Figure 5-6), Work Order (Figure 5-9), Work Order Status (Figure 5-10) and Next Maintenance Dates (Figure 5-11).

Only the basic structure of Au Yang's Coordinator was maintained; the main effort here was to modify the code of the original responses and implement the required new ones. As the CMMS Coordinator grew, it was found that responses could be "reused" and combined to form other responses, in accordance with the design, and with relatively little difficulty. Section 5.4 provides further details regarding the Coordinator implementation.

The CMMS Visual Basic execution modules were also adapted from Au Yang's system. Section 5.5 discusses the details of how the executors were coded. Compared to Au Yang's database, the CMMS database is significantly more complex in design and implementation. The Excel-based Purchase Order Form, however, maintains a close resemblance to Au Yang's Sales Order Form.

Figure 5-1 provides a comparative illustration of the functionality contained in Au Yang's Sale Support System and the example CMMS.

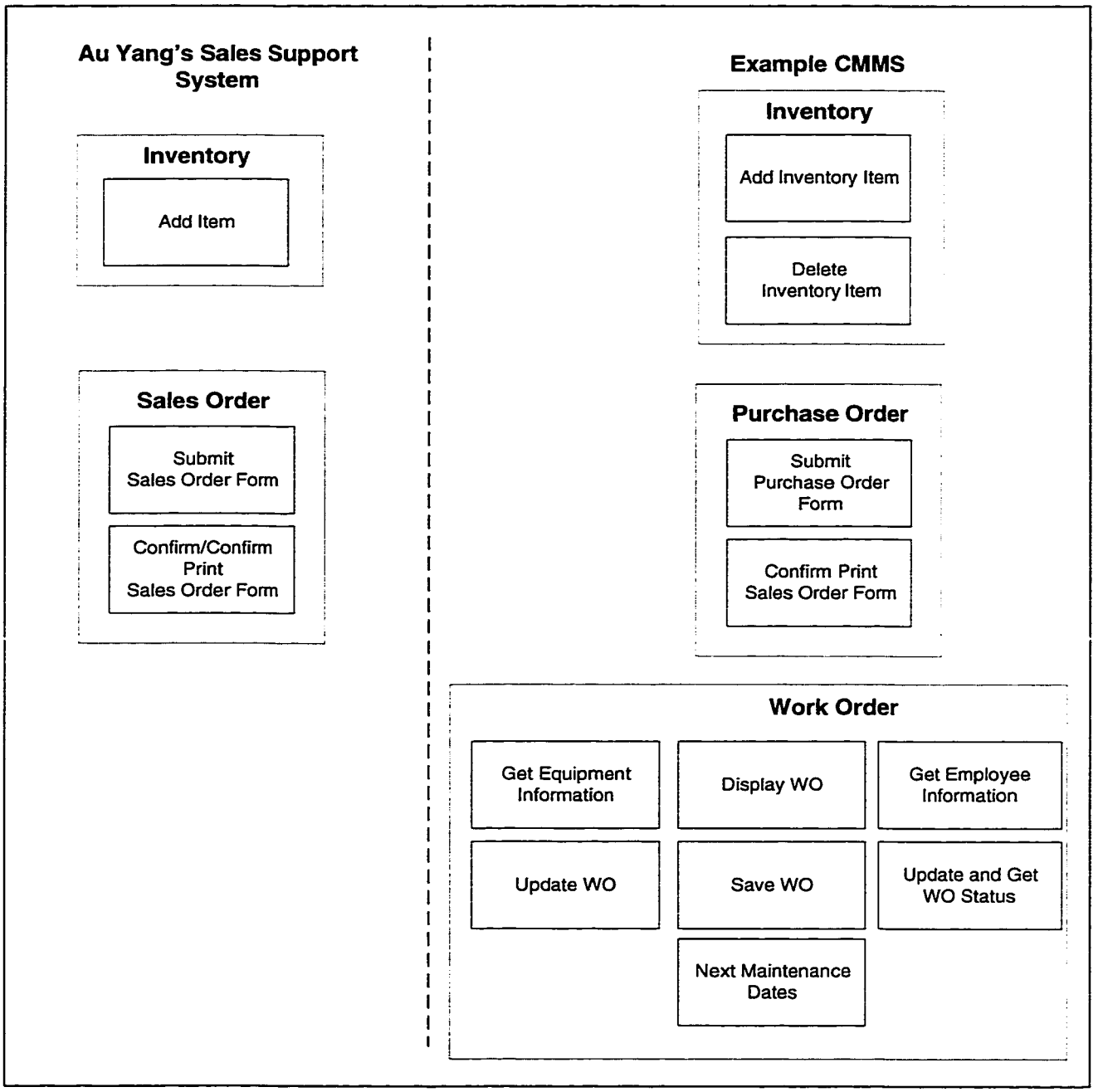


Figure 5-1 - Implemented Functionality (Sales Support System Vs. Example CMMS)

5.2 Implemented Architecture

The architecture implemented is illustrated in Figures 5-2 and 5-3. Figure 5-2 highlights the information / data flow within the CMMS. As can be seen, the data flow is entirely vertical, with no interaction between the modules. Figure 5-3 shows the same general structure, however, this is mainly from the perspective of the interfaces with the COTS components (the MS Access Database and the MS Excel spreadsheet). In Figure 5-3, the shaded boxes represent tables in the CMMS database which were actually accessed by the activity executors. The database tables shown in Figure 5-3 are likely representative of the number and type of tables that would exist in many commercial products.

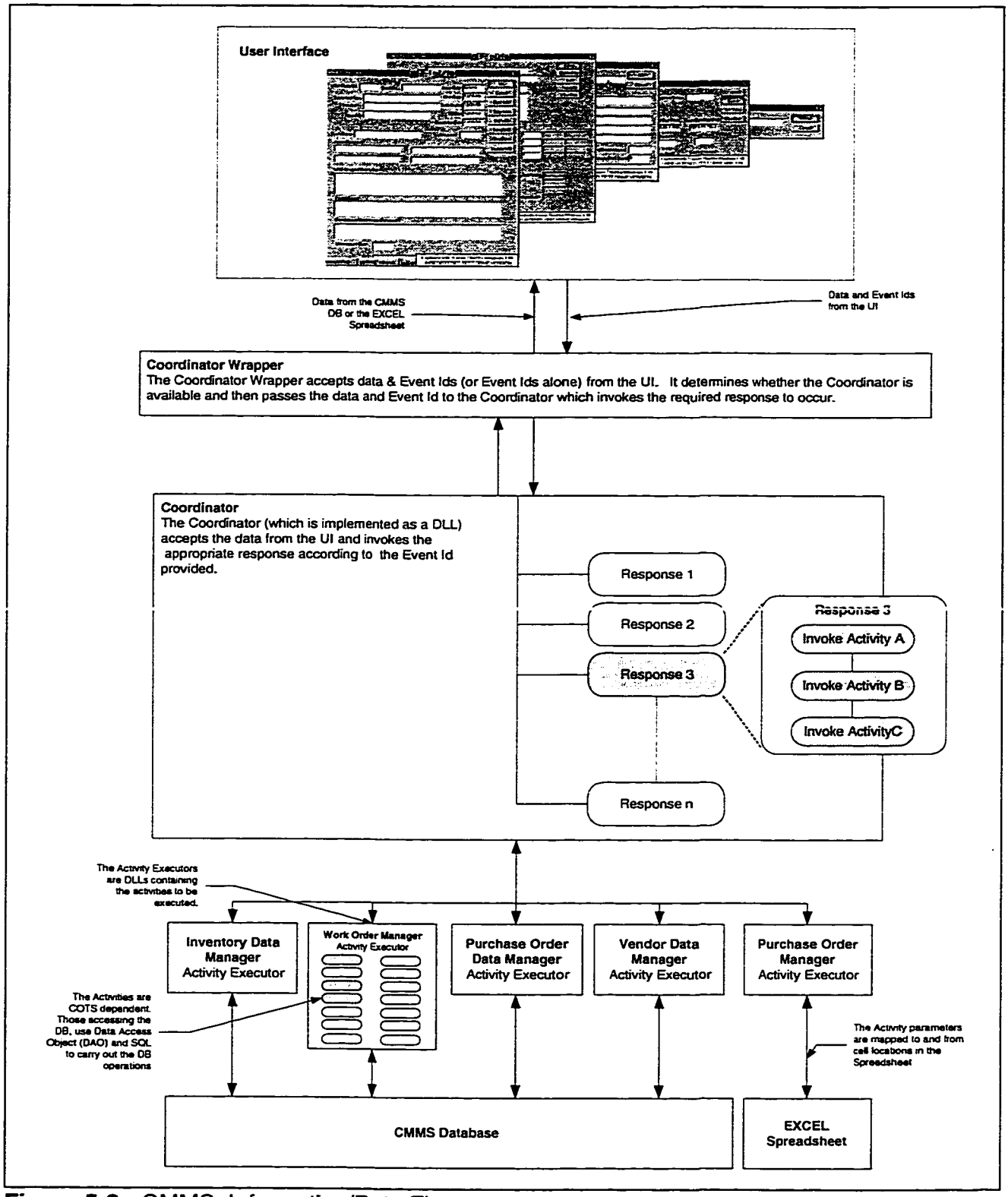


Figure 5-2 - CMMS Information/Data Flow

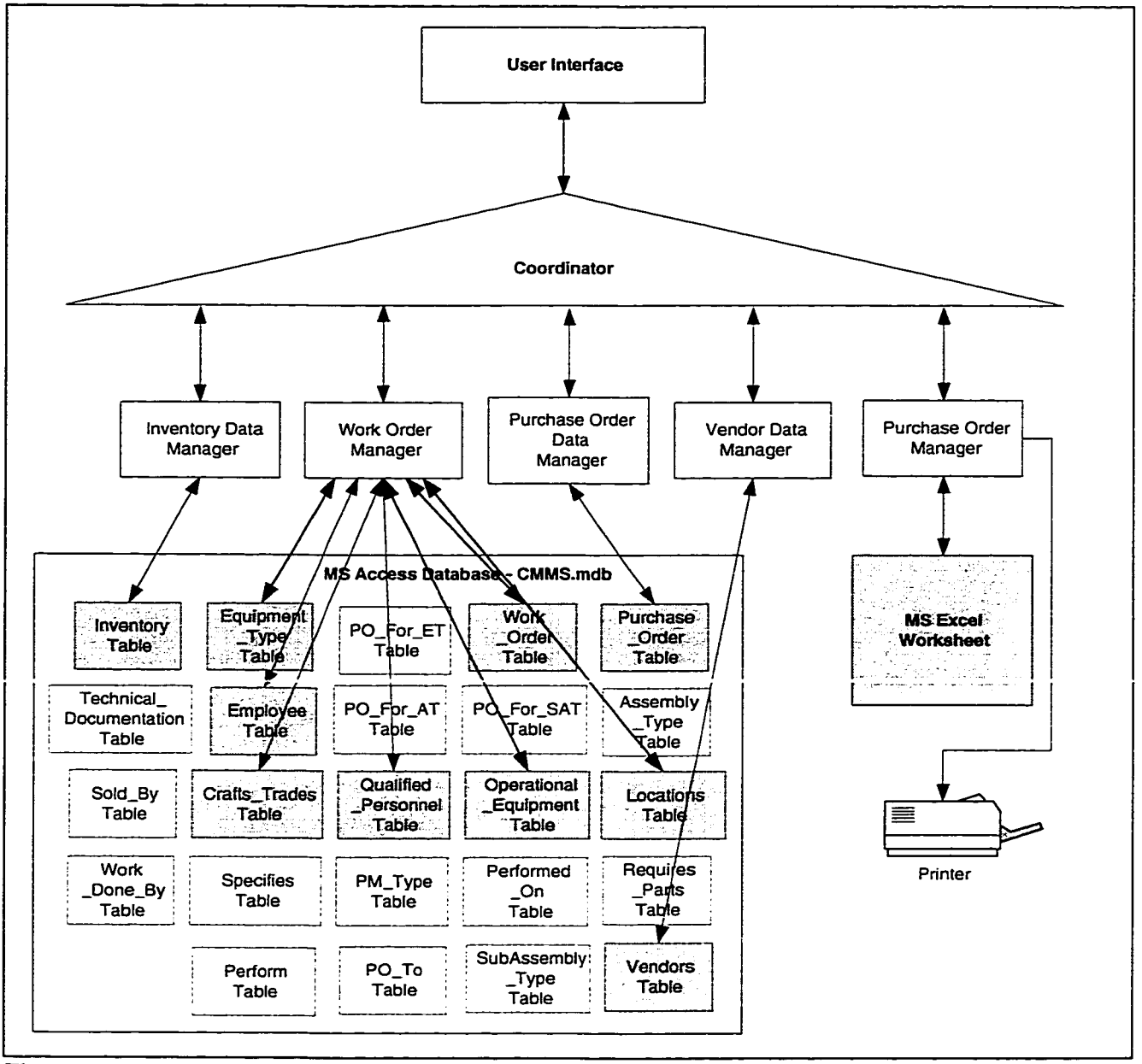


Figure 5-3 - Interface To COTS Packages

5.3 The User Interface and System Operation

The User Interface essentially consists of one main screen and a series of dialog boxes, which allow the user to display and store information in the CMMS database and create and print out a purchase order using an Excel spreadsheet. Section 5.3.1 lists the files which comprise the Local UI and subsequent sections provide descriptions of the CMMS example main screen and dialog boxes.

5.3.1 The Local UI Files

The Local UI files consist of the 22 files listed below. (Annex F contains a print out of the Local UI Code.)

1. AddInventoryItem.cpp
2. AddInventoryItem.h
3. CreatePurchaseOrder.cpp
4. CreatePurchaseOrder.h
5. CreateWO.cpp
6. CreateWO.h
7. DeleteInventoryItem.cpp
8. DeleteInventoryItem.h
9. WO_Statu.cpp
10. WO_Statu.h
11. NMaint.cpp
12. NMaint.h
13. LocalUI.cpp
14. LocalUI.h
15. LocalUI.rc
16. LocalUI.odl
17. LocalUIDoc.cpp
18. LocalUIDoc.h
19. LocalUIView.cpp
20. LocalUIView.h
21. Mainfrm.cpp
22. Mainfrm.h
23. StdAfx.cpp
24. StdAfx.h

Next, the different Local UI screens are described in detail.

5.3.2 Main CMMS Screen

Figure 5-4 illustrates the main CMMS screen. Although not visible in this figure, under the "CMMS Screens" bar, selections are provided to display dialog boxes for:

- Inventory;
- Purchase Orders;
- Work Orders;
- Employees, and;
- PM Routines.

Dialog boxes were created for:

- Add Inventory Item (Figure 5-5);
- Delete Inventory Item (Figure 5-6);
- Purchase Order (Figure 5-7);
- Printable Purchase Order Form (Figure 5-8);
- Work Order (Figure 5-9);

- Work Order Status (Figure 5-10), and;
- Next Maintenance Dates (Figure 5-11).

In addition to this capability, a macro was created which causes "traces" of the event responses, (including the activities executed) to be displayed on the Main CMMS Screen as shown in Figure 5-4. The response pictured is "Add Inventory Item". The traces shown in the figure are highlighted in the code below.

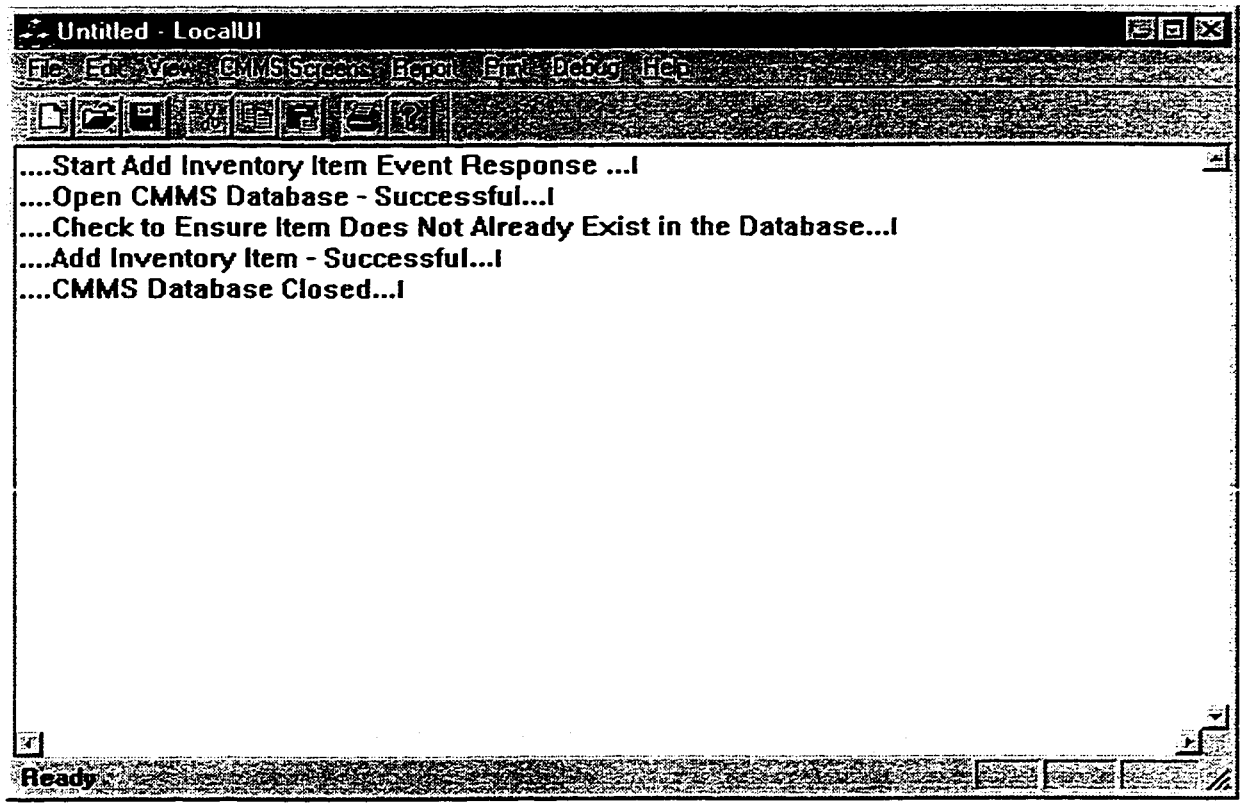


Figure 5-4 - Main CMMS Screen

The Coordinator code containing the traces for Add Inventory Item is shown below. These traces can be (and were) advantageously used for trouble shooting purposes, as they can be placed strategically throughout the code to announce different results, events or branches taken while executing the code.

```
//ADD INVENTORY ITEM (ATOMIC) EVENT RESPONSE
        case EID_add_inv_item:
        {
                MY_TRACE("....Start Add Inventory Item Event Response ...\n");
                        BOOL  pnExisted;
                        short n_rc;
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
                        rc = objInvDB.openInvDB();
                        if (rc == -1) {
                                MY_TRACE1("....Open CMMS Database - Failed rc=%d...\n", rc);
                                MY_TRACE("....Exit Coordinator...\n");
                                return 0;
                        }
                        MY_TRACE("....Open CMMS Database - Successful...\n");
                        // Obtain the IPN, EPN and ITD
```

```
                        // They are the member variables of the dialog.
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CHECK PART NUMBER NOT IN INVENTORY TABLE
                        MY_TRACE("....Check to Ensure Item Does Not Already Exist in the Database...\n");
                        n_rc = objInvDB.recExistPN(a_inv_i_pkg.m_mpn,
                                                                        &pnExisted);
                        // If uniqueness validation is successful, then
                        // create the record.
                        if ((n_rc == -1) || pnExisted) {
                                MY_TRACE("....MPN/EPN not unique...\n");
                                MY_TRACE2("....n_rc=%d  pnExisted=%s...\n",
                                                                n_rc, (pnExisted ? "True" : "False"));

                                rc = objInvDB.closeInvDB();
                                MY_TRACE1("....Close CMMS Database rc = %d...\n", rc);
                                MY_TRACE("....Exit Coordinator...\n");
                                return 0;
                        }
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CREATE RECORD IN INVENTORY TABLE OF CMMS DATABASE
                                rc = objInvDB.recCreate(a_inv_i_pkg.m_mptd, a_inv_i_pkg.m_mpn,
                                a_inv_i_pkg.m_sn, a_inv_i_pkg.m_atno, a_inv_i_pkg.m_etno,
                                a_inv_i_pkg.m_satno, a_inv_i_pkg.m_qr, a_inv_i_pkg.m_nur,
                                a_inv_i_pkg.m_iqty, a_inv_i_pkg.m_isl, a_inv_i_pkg.m_msl);

                        if (rc == -1) {
                                // If record creation has failed, acknowledge
                                // failure
                                MY_TRACE1("....Creation Failed rc=%d...\n", rc);
                                rc = objInvDB.closeInvDB();

                                MY_TRACE1("....Close CMMS Database rc = %d...\n", rc);
                                MY_TRACE("....Exit Coordinator...\n");
                                return 0;
                        }
                        // If record creation is successful, acknowledge
                        // success
                        MY_TRACE("....Add Inventory Item - Successful...\n");
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CLOSE CMMS DATABASE
                        rc = objInvDB.closeInvDB();
                        MY_TRACE("....CMMS Database Closed...\n");
}
break;
//END OF ADD INVENTORY ITEM (ATOMIC) EVENT RESPONSE
```

The traces can also be saved to be viewed later as a detailed record or transcript of the activities invoked, events handled, and system responses or used to reconstruct particular scenarios. These traces were very helpful for troubleshooting as it was possible to determine at which point the program collapsed by the last trace that appeared on the screen.

5.3.3 Add Inventory Item Screen

Figure 5-5 shows the Add Inventory Item Dialog Box. The "Add Inventory Item" function accepts information about a new inventory item, entered in the UI Add Inventory Item Dialog Box and stores it in the CMMS Database "Inventory" Table. This database table is first checked to ensure that the manufacturer's part number of item doesn't already exist.

The information stored consists of:

1. Manufacturer's Part Number;
2. Text Description;
3. Part Number;

4. Serial Number;
5. Equipment/Assembly/Sub-Assembly Number;
6. New/Used/Reconditioned;
7. Minimum Stock Level, and;
8. Item Stock Location.

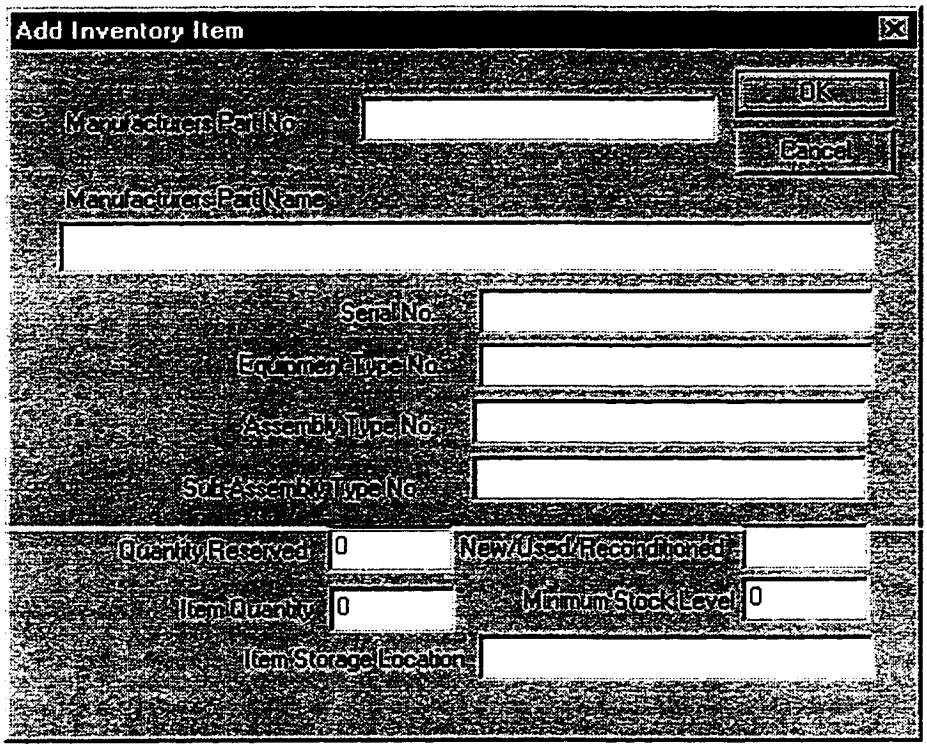


Figure 5-5 - Add Inventory Item Dialog Box

5.3.4 Delete Inventory Item Screen

Figure 5-6 shows the Delete Inventory Item Dialog Box. The "Delete Inventory Item" function deletes an entire record (Inventory Item) from the Inventory Table in the CMMS Database. It is initiated by inputting the manufacturer's part number followed by clicking the "Delete" button.

Figure 5-6 - Delete Inventory Item Dialog Box

5.3.5 Purchase Order Screen and Printout

Figure 5-7 shows the Purchase Order Dialog Box. This capability allows the user to generate a purchase order for required equipment or parts. The steps involved in creating a purchase order from this screen are as follows:

1. Enter the desired vendor name, the part numbers for an equipment type, assembly type and/or sub-assembly type, together with the quantities required;

2. Click the "Submit" button. Activities are invoked which open the DB, check to see if the Vendor Name exists in the database, and then causes the vendor information (address and phone number) and cost of the parts to be retrieved from the CMMS Database and displayed.

3. Upon clicking the "Confirm Print" button, the system sends the data to an Excel spreadsheet (Purchase_Order.xls), which calculates the total cost for each part number, the PST and FST, and the total cost for all parts ordered. These values are then returned to the screen. The purchase order form is also printed out with the relevant information contained (see Figure 5-8).

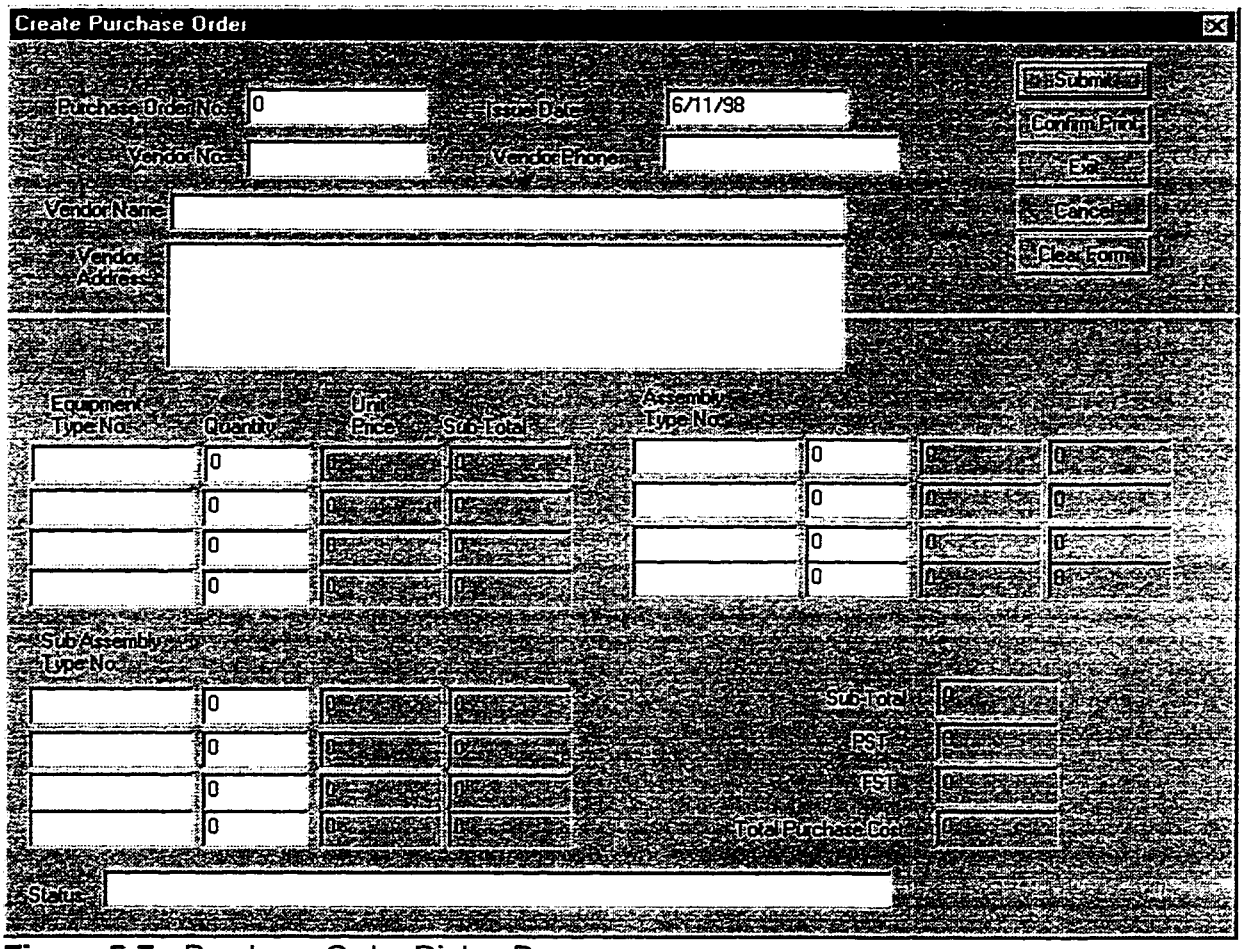
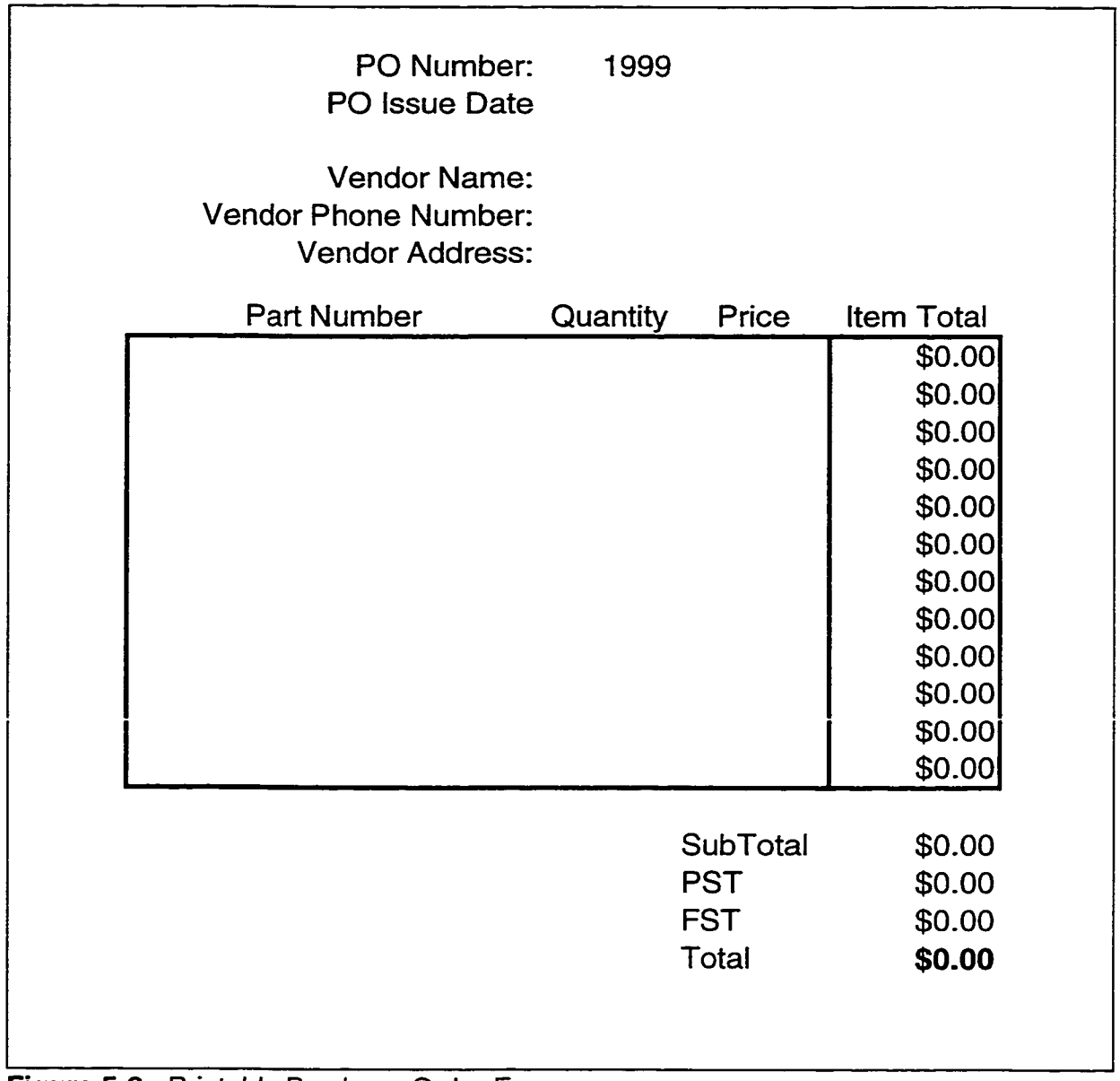


Figure 5-7 - Purchase Order Dialog Box



PO Number: 1999
PO Issue Date

Vendor Name:
Vendor Phone Number:
Vendor Address:

Part Number	Quantity	Price	Item Total
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00
			$0.00

SubTotal	$0.00
PST	$0.00
FST	$0.00
Total	**$0.00**

Figure 5-8 - Printable Purchase Order Form

5.3.6 Work Order Screen

The Work Order Dialog Box is shown in Figure 5-9. The actions involved in this scenario first provide for the initial creation of the work order (by the Manager, for example). The steps followed in the work order scenario are identified below:

1. The manager (initiator) of the work order enters the serial number of the piece of equipment requiring maintenance.

2. Clicking the "Eqpt. Info" button then causes the CMMS to find and display, in the work order dialog box, the Equipment Type Name, where it's geographically located and narrative descriptions of the previous maintenance performed on that piece of equipment.

3. The initiator then enters the:

   - Start Date for the work;
   - Target Completion Date for the work;
   - Work Assigned Description;
   - Trade Name of the technician(s) to perform the work, and;
   - Technicians Trade Level.

4. The CMMS queries the database and responds with names of technicians at the same location of the subject piece of equipment and with the required qualifications in terms of trade type and level. The initiator then has the choice of selecting/de-selecting any of the names displayed.

5. The next step is to then save the necessary information (in the Work Order Dialog Box) in the "Work_Orders" Table of the CMMS Database. At this point the work order has been opened - any subsequent entries in the work order would either be made by the initiator to modify it or by the technician(s) it is assigned to.

6. The assigned technician can display the WO (Edit WO button), after entering the WO Number and then update the WO by entering Date Started, Date Completed, Faults Found, Description of Maintenance Performed and Maintenance Time details in the edit boxes provided. Lastly, by clicking the "Update WO" button, the entire WO is updated/saved in the CMMS DB.

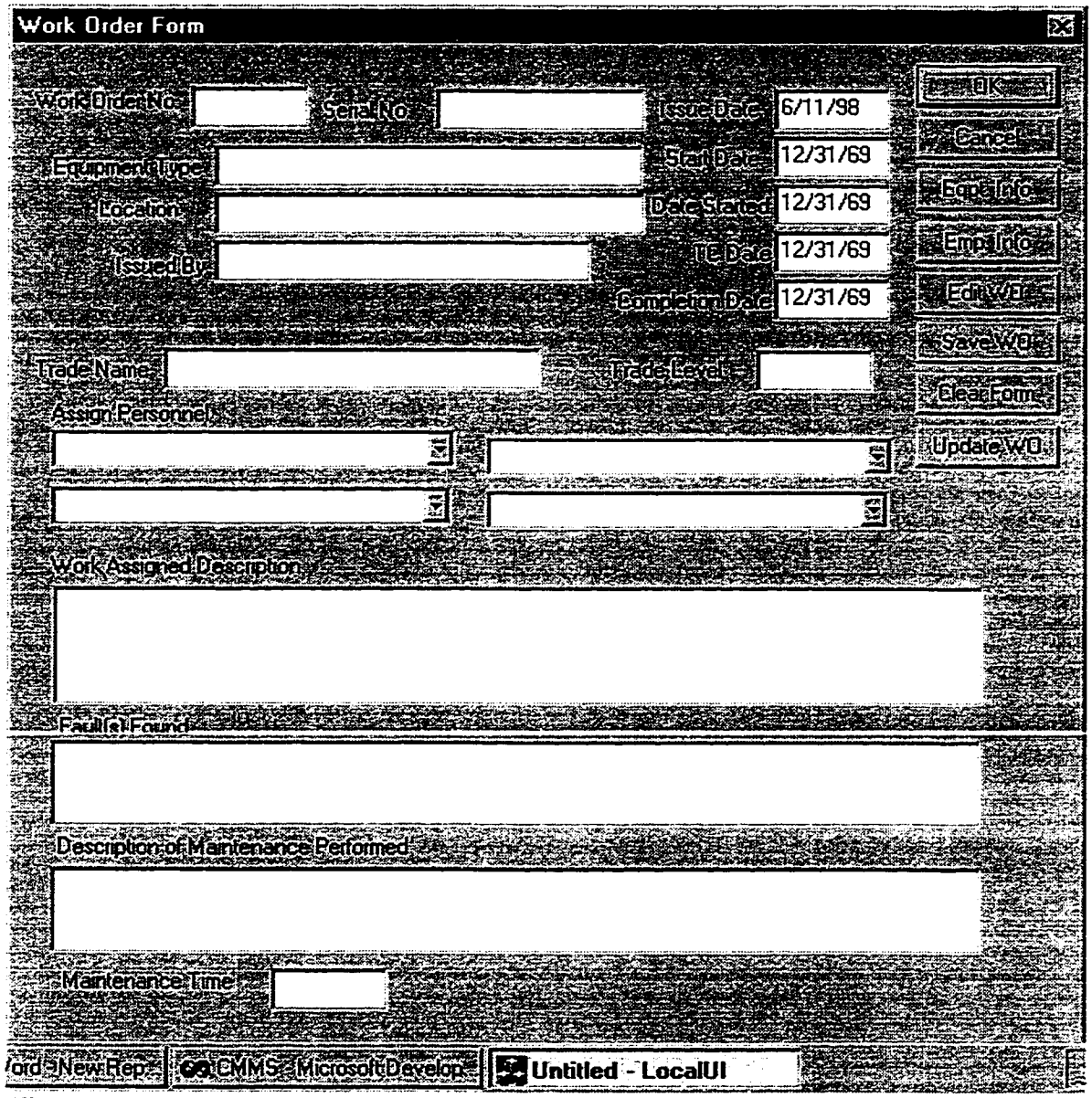


Figure 5-9 - Work Order Dialog Box

5.3.7 Get WO Status

This Dialog Box allows the status of any work order to be displayed, upon request. The user enters the WO number about which the status is required. The System then updates the status in the database. Specifically, the following is updated (yes or no) in the database:

- Whether the WO has been started;
- If it has been started, whether it was started after the Target Start Date;
- If it has been completed, and;
- If it has been completed, whether the completion date is after the Target Completion Date.

The updated status information in the database is then displayed in the dialog box, as shown in Figure 5-10.

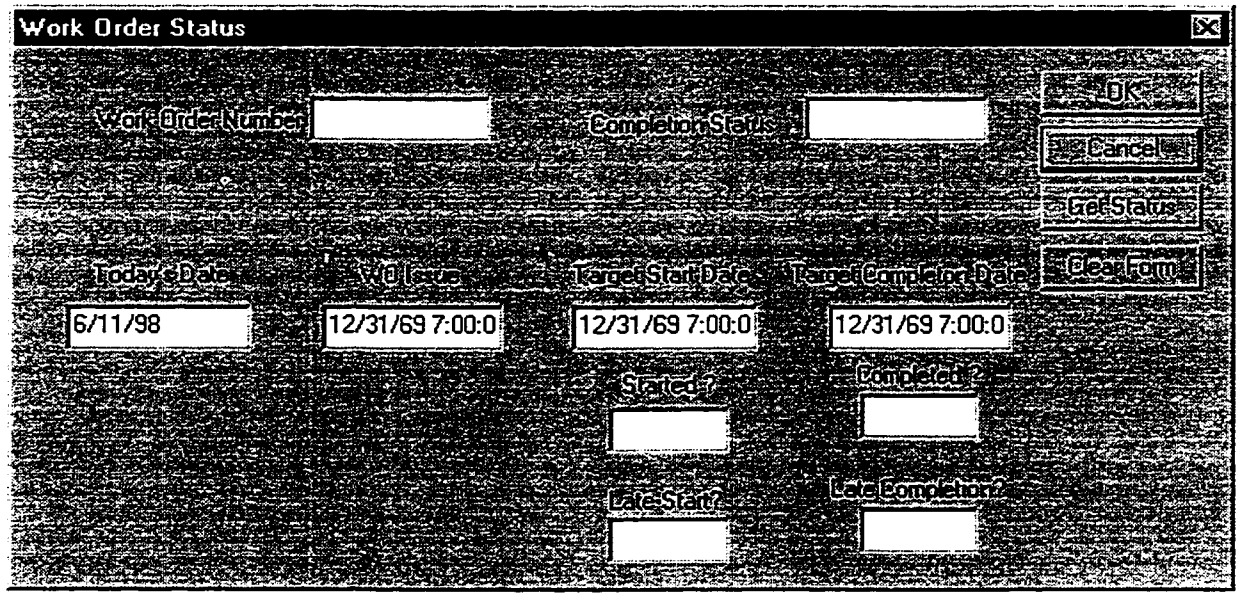


Figure 5-10 - Work Order Status Dialog Box

5.3.8 Next Maintenance Dates

This screen allows the user to extract, from the CMMS Database, dates of the next (weekly and monthly) maintenance routines and the serial numbers for a particular equipment type, and at a specified location. It was the intention to include with this scenario, the capability to automatically update the maintenance dates in the database. However, the time was not available to accomplish this. Instead, for the sake of demonstration, dates were manually stored in the "Next Weekly Maintenance" and Next Monthly Maintenance" fields. Accordingly, upon depressing the "Get Weekly" and "Get Monthly" buttons, the stored dates are displayed in the dialog box. Another limitation is that when multiple serial numbers and maintenance dates are required to be appear in dialog boxes, only one serial number and date are displayed. This restriction is programming related and could be rectified with limited further attention. Figure 5-11, below, shows the dialog box developed to accommodate the "Next Maintenance" functionality.

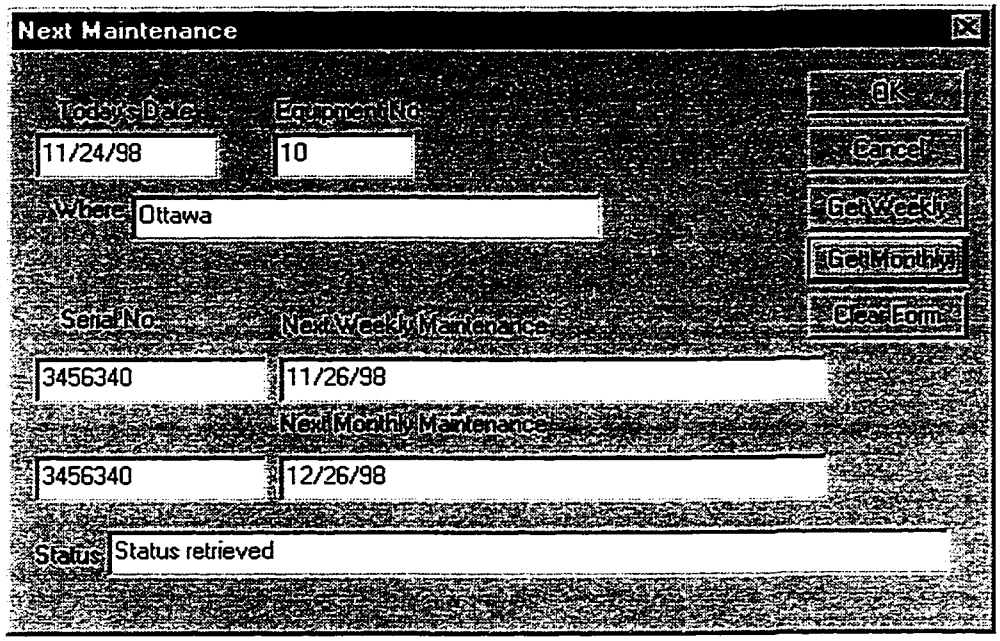


Figure 5-11 - Next Maintenance Dates

5.4 The Coordinator

The Coordinator was implemented as a Dynamic Link Library (DLL) in the Coord.cpp and Coord.h files. The CoordWrap.cpp and CoordWrap.h files serve as an interface between the Local UI files and the Coordinator. Annex C contains a print out of the Coordinator code. The file names for the Coordinator are listed below.

1. coord.h
2. coord.cpp
3. coordwrap.h
4. coordwrap.cpp

5.5 Activity Executors

The Activity Executors were created using Visual Basic. For comparison purposes, Table 4-3 shows the Au Yang modules which were the starting point for creating the CMMS executors.

Each of the activity executors is implemented as a DLL. These DLL files and the Visual Basic files used to create them are identified in Table 5-1. A print out of the Activity Executor files in contained in Annex E.

Table 5-1 - Activity Executors

Activity Executors	DLL File Names	Visual Basic File Names
Inventory Data Manager (InvDB_Handler.dll)	inv_dbhandler.cpp, inv_dbhandler.h	InvDBHandler.cls InvDB_Jet_DAO.bas
Work Order Manager (DB_Handler.dll)	dbhandler.cpp, dbhandler.h	DBHandler.cls DB_Jet_DAO.bas
Purchase Order Data Manager (PODBHandler.dll)	podb_dbhandler.cpp, podb_dbhandler.h	PODBHandler.cls PODB_Jet_DAO.bas

Activity Executors	DLL File Names	Visual Basic File Names
Vendor Data Manager (VenDBHandler.dll)	ven_dbhandler.cpp, ven_dbhandler.h	VenDBHandler.cls VenDB_Jet_DAO.bas
Purchase Order Manager (POFormPrint.dll)	poformprint.cpp, poformprint.h	POFormClass.cls XL_POForm.bas

5.6 CMMS COTS Software

The CMMS example employs MS Access and MS Excel as integral parts of the system. MS Access is used for the database (see Annex D), and MS Excel performs the mathematical calculations required to determine Item Subtotal costs, Purchase Order Subtotal, PST, FST and the total purchase price as part of the "Create Purchase Order" capability. The COTS file names are CMMS.mdb and Purchase_Order.xls and are provided in soft copy with this thesis.

5.7 Integration and Test

Integrating the software components was considerably time consuming. It involved first ensuring that Au Yang's software would work on the computer which was used in developing the CMMS example. No small task - as it first required transferring the code to the author's computer, then altering the file location information (directory structure) in the activity executors to point to new locations for the CMMS DB and the Purchase Order Form and finally registration of the DLL's.

The next step was to build the designed CMMS DB (Annex D). Once this was accomplished, the main effort was to create the CMMS "Add Inventory Item" dialog box, and modify the Coordinator and activity executor code accordingly. Once it was possible to successfully transfer inventory item information from the "Add Inventory Item" dialog box to the Inventory Table in the CMMS DB, the CMMS software essentially "worked" and a functioning baseline existed upon which the rest of the CMMS could be developed. It took approximately 4 months (part time) to achieve this working baseline.

Most responses involved displaying information from the database in an edit box or saving information as entries in a database table. Testing this functionality simply required verifying that the intended information was properly displayed on the screen or by opening the database in MS Access and checking that the appropriate table(s) were populated accordingly. The various database related functions required the creation of new Standard Query Language (SQL) scripts to access or store data. Before testing the end to end response (which required the SQL capability) verification that the SQL scripts worked properly was first performed in MS Access, directly. For the Purchase Order Form functionality, testing involved printing out the form populated with the information entered from the screen and the calculated totals.

5.8 Conclusions

The difficulty with which the CMMS example was implemented was largely a consequence of the effort required to understand and use the development platform. For an experienced programmer, this part of the work would have offered far less of a challenge than it did for the Author. Had it not been for the simplicity of CBD, it would have taken far longer to complete the system build. Nevertheless, Coordination Based Design lends itself to easy understanding and interpretation, and from this point of view can be confidently given to programmers utilizing different development tools with the expectation that it could be satisfactorily transformed into a working system. To verify this, an interesting follow-on activity would be to have a CBD designed system implemented in different programming languages.

6. Concluding Remarks

6.1 Quality Software Through CBD

As discussed in Chapter 1, the "ilities" is a group of attributes which describe various desirable aspects of software quality. After having been through the experience of designing the example CMMS using the CBD methodology, and then implementing it, it is the Author's opinion that software developed using this design environment will clearly exhibit most of these qualities. Below, a brief assessment is offered of CBD and the CMMS Example developed in relation to the "ilities".

Because of the hierarchical nature of the CBD architecture; the separation of coordination from execution and the "vertical" flow (from UI to Coordinator to activity executors) through the system, simplicity in design and implementation is achieved. The Coordinator and each of the execution modules can be separately implemented and tested, and traceability is easily and effectively established. In the CMMS example, all actions are executed in a straight forward manner, from UI, to Coordinator and down to the execution modules. And, data for display or storage follows this same "up and down" direction, as well.

Modularity is clearly an intrinsic characteristic of CBD. Each of the response managers in the Coordinator were easily distinguishable and have intuitive well defined boundaries, which could be separately implemented and tested. And, each of the activity executors were created as distinctly separate modules tested completely independently. In the effort to develop the example CMMS, it was found that it was relatively easy to design and implement additional functionality, either by adding to or modifying the Coordinator and activity executors. Minor changes or complete new suites of functionality could be added as required, with little consideration necessary in terms of its effect on existing functionality. For instance, the design and implementation of the entire Work Order Execution Module had no impact, whatsoever, on any of the existing Coordinator functionality or the other execution modules. Also, it was possible to include additional functionality (Work Order Status and Next Maintenance) in a short time, again without effect on the baseline system.

There are two principle ways to increase reliability in a system design: redundancy and simplicity. Although no redundancy was built into this CMMS example, an important advantage of CBD architecture is its inherent simplicity, consequently, there were no reliability problems experienced, at all. As each new capability was added, regression tests were performed to ensure functions previously developed remained error free - no problems were experienced during this testing or at any other time, once the functions were first working.

The attribute of "user-friendliness" is basically subjective in nature, and there is no design methodology (including CBD) that will guarantee, through its use, that the system or application developed will be user friendly. A simple and easily used and understood methodology, such as CBD is certainly "developer-friendly"; and, with the CBD technique, the designer can disclose the essence of the design, to the user, early in the process, to acquire user "buy-in". CBD also lends itself, well, to prototype based design as the developer and/or user can easily ensure, through the user interface, the presence of expected functionality. Otherwise, the major factor affecting a system's user-friendliness is the experience (in the application and operational environment) and skill of the designer(s).

The ease with which a software system can be maintained is related to the factors mentioned above, as well as other considerations, including separation of concerns, documentation, configuration control and management, and readability of the code, etc. Software designed through CBD certainly possesses characteristics consistent with maintainability. The CBD methodology stipulates separation of concerns / modularity and imposes simplicity in the design. CBD is essentially self documenting, in that the specification of events and responses constitutes the documentation of the requirements specification, the definition of the response managers and execution modules essentially documents the design [MK26]. Other factors such as configuration management and control require discipline and must be implemented regardless of the design methodology, however, through the uncomplicated CBD methodology these tasks are potentially made less onerous.

To summarize, CBD is a sound development methodology which is easily capable of producing quality software. No guarantees are offered, in this regard, as the skills and abilities of designers will always play an important role in the quality of the final product, however, this technique does offer great potential to produce high quality software and any analysis and design tool or technique which espouses the tenets of CBD will do likewise.

6.2 Tailorability Through CBD

A major objective of this thesis was to demonstrate that CBD is an excellent vehicle for tailoring software. The initial step was to start with a core set of domain specific functional requirements which could be generated, as shown in this thesis, by performing a survey of various products in existence. Due to the clean hierarchical structure of systems designed using CBD, it was shown that this survey information could then be effectively translated into traceable system level events and responses. This core system level functionality is then transformed into a set of discrete atomic responses. The importance, herein, is that this set of activities can then be tailored (reused and combined) as appropriate to develop different or more sophisticated responses to augment the core functionality and ultimately suit individual customer requirements.

It can be a powerful advantage, to any software house in the business of marketing a particular business software product, to be able to provide a tailored product to a customer quickly and effectively. This ability to tailor system functionality was demonstrated on different occasions during the development of the example CMMS, and it is important to note that to tailor the system in this was not difficult to accomplish. The ease of building combined responses using the different existing atomic responses proved to be an expedient way to develop more sophisticated responses and new sets of responses were easily added to the design and implemented (e.g. "WO Status" and "Next Maintenance") with the use of existing activities, and with short notice.

6.3 Reuse and Component-Based Design

Through the CBD methodology it was possible to effectively reuse the design and code of Au Yang's Sales Support System [AY28] for the CMMS example. Again, it was the simplicity and understandability of the Coordination Based Design that allowed this transformation to take place. The component-based aspect of Au Yang's system was reused successfully, as well, in the CMMS example, with the only effort needed to modify the MS Access database and EXCEL spreadsheet files to suit the needs of the CMMS design.

6.4 Traceability

This thesis demonstrated that the use of CBD supports traceability by way of its architectural simplicity, as the events and responses are explicitly defined as part of the specification of the requirements, which then cleanly correlate to response managers in the design of the Coordinator. Traceability is, then, easily established from the Coordinator to the execution modules, through the activities called up and to the implementation of the code. This complete traceability provided through CBD simplifies both the "understandability" of the design and provides the confidence that systems designed using CBD can be easily shown to be compliant.

6.5 My Learning Experience

6.5.1 CMMS's and Software Design

Although my background includes having worked in several maintenance environments, both as a technician and as a manager in the Canadian Forces and in industry, this did not include any knowledge of or familiarity with CMMS's prior to developing this thesis. A significant portion of the research and study was, therefore, focused on understanding the environments in which CMMSs are used and the main types of functionality that are typically offered. Through the research carried out, I was able to learn enough about these applications to demonstrate the applicability of CBD as a viable methodology for designing a system, such as a CMMS. This research included formulating a representative list of several CMMS vendors, reviewing their product literature and requesting information via the telephone or by E-mail. Trade magazines and internet articles were also extensively reviewed for CMMS related articles.

Importantly, this thesis has afforded me the opportunity to learn, in some detail, about software design, in general and CBD in particular. When initially starting the thesis, a general review of other software development methodologies was done, with the intention of doing an in-depth comparison with CBD. However, considering the time and effort required, it was decided instead to use CBD to develop a working business system example.

6.5.2 Programming / Implementation of the CMMS Example

Despite having started with previously developed code "in hand", (Au Yang's Sales Support System) the challenge and time required to learn how to modify and augment (tailor) this software, and make it work, was not trivial. However, because the design of the Sales Support System was easily understood, the only difficulty was related to learning the programming languages - the implementation environment (MS C++ and MS Visual Basic) had to be learned from scratch. In all, it was because of the simplicity and power of the CBD methodology that it was possible to effectively take up these challenges and complete the example CMMS, successfully. The magnitude and scope of the MS C++ and MS Visual Basic experience gained, was significant, having progressed from basically no programming knowledge, to having learned, created, practiced and used a broad range of programming concepts, including the following:

- Work spaces and projects;
- Classes, instances, Class Wizards;
- Member variables;
- Message boxes and traces;
- Header files;

- Creating and registering DLL's;
- Compiling and building;
- Debugging code (many, many hours spent);
- Break points;
- Creating and modifying the user interface (dialog boxes, edit boxes, etc.), and;
- Commenting;

It took approximately 6 months of intense "after hours" work to complete the CMMS example. Periodically, I required assistance with the programming from a fellow post graduate student (Muddesir Siddiqui). This was necessary, especially at the outset, so that I could learn the basic MS Visual C++ and MS Visual Basic concepts and skills. Enough was learned initially, to cut, paste and modify code, and then to write new code. As might be expected, I soon found that it was difficult to progress further without causing many errors and warnings to occur. As a consequence, I was able to develop reasonable troubleshooting skills and the exercise of remedying faults served extremely well as a means of re-enforcing my understanding of the development platforms and application programs employed. Throughout this exercise, there were also times when I required more assistance to resolve some faults, and when assisted (by Muddessir) in solving "difficult" errors, I was usually able to go on and solve many more; and, I found that as my experience with the languages grew, the number of errors I required assistance with, became profoundly fewer, and the calls for assistance much less frequent.

The CMMS database is significantly more complex than Au Yang's, although, the same database application was used - MS Access. Comparatively, my knowledge of relational database design, and SQL was considerably greater than my programming experience - I had acquired these skills as a result of "on-the-job" experience and the "Integrated Database Design" course taken as part of my Post Graduate training.

Overall, I consider the experience of developing the CMMS example an extremely worthwhile endeavor, not only as a viable example of software developed using CBD, but as a personal learning vehicle. It was invaluable in terms of solidifying an understanding of software design as well as the design implementation.

6.6 Possible Extensions

The CBD methodology has existed to-date exclusively within an academic environment, and there are many potential areas of further study appropriate to pursue, both in terms of continued post graduate research and in the area of its potential commercial viability. A few of these ideas are presented below.

In Object Oriented Design, objects are typically not as well bounded as they are in CBD, in that the CBD executors interact only with the Coordinator, but OO objects interface, as required, in a less structured manner with other objects. This would tend to support the argument that a system designed according to the CBD architecture would be comparatively efficient. However, in this thesis, there was no time or effort afforded to the consideration of the efficiency of the design and implementation. Efficiency and performance of software designed through CBD are important considerations, nevertheless, and therefore should be an area of study or follow-on work in the future through engineering papers and thesis work.

As stated in Section 5.8, an important assertion of CBD is that it is universally applicable to different methods of implementation. The merits of being able to implement a CBD design in different programming languages could have far reaching implications and benefits. It is recommended that an effort be undertaken to validate this argument and determine more precisely what practical advantages could be gained.

The CMMS was chosen as the type of application to explore in concert with CBD, for reasons particular to the Author, and is an example of a class of industrial or business application of which there are countless others. Any process, such as CBD, which promises simplicity and broad applicability, has the potential to offer significant savings in time, effort and expense and therefore should be attractive to both business, and industry. Software houses which specialize in business systems or other types of applications could very practically benefit from the use of this development methodology. They can develop domain specific components as execution modules which can be easily tailored to suit particular user requirements. Quite often, one can also use the same design and tailor it for different business applications. Examples might include large wholesale houses, mail order systems, large accounting and maintenance systems for different businesses and hospital systems. A more thorough investigation into the range of different types of these business applications which could be designed effectively using CBD, would offer further insight into how broadly applicable this technique is.

As discussed in Chapter 3, one of the more sophisticated and significant types of functionality available in some of the larger CMMS packages, is predictive maintenance (PdM). Considering that much of the original work relating to CBD concerned real-time, embedded systems, which are characterized by the effort to monitor (sense) and react to events occurring in the surrounding physical environment, investigating the applicability of CBD to design and implement a PdM system may constitute some interesting future work.

Lastly, the study of the CBD methodology in the academic environment continues to serve students and the discipline of software engineering well. An investigation into developing a transition strategy to move CBD from its current environment to commercial viability would be another interesting area to investigate.

7. Bibliography

Item	Reference
MK1.	Moshe Krieger, "The Multiactivity Paradigm: An Approach for the Design of Embedded Systems by Application Specialists" An unpublished paper, included in the course notes from the "Embedded Systems" Course, University of Ottawa, Winter 1995.
Gen2.	W. M. Gentleman, "If software quality is a perception, how do we measure it?", Conference Proceedings, The Sixth International Conference on Software Quality (6ICSQ), 28-30 October 1996, Ottawa, Ontario, Canada.
DB3.	David Bugden, Software Design, Addison-Wesley Publishing Company, 1994, pp. 64-74.
TLW4.	Tse L. Wang, "EVOLVING TOWARD SYSTEMS INTEGRATION", AT&T Bell Laboratories
AA5.	Adaptable and Adaptive Software Workshop at OOPSLA'95 (http://www.ccs.neu.edu/research/demeter/adaptable systems)
TG6.	Volker Wolf, Tailorable Groupware: Issues, Methods, and Architectures, volker@uran.informati.uni-bonn.de
PL7.	Programming Language Research at Oklahoma State University (http://www.cs.okstate.edu/~hatclif/pe-osu.html
KLJPL8.	Kenneth C. Laudon, Jane Price Laudon, "Management Information Systems A Contemporary Perspective", Macmillan Publishing Company, 1988, pp. 39-40
FRC9.	Finding the 'Right' CMMS - (http://facilitiesnet.com/NS/NS3mg6c.html)
KF10.	Key Functions of Maintenance Management Systems (CMMS) (reprinted from Automation Strategies, May, 1996), http://www.arcweb.com/Public/world/docs/cmms_ftn.htm
MP11.	MP2 Professional 5.0 http://ww.dstm.com/frames/products/mp2pro50.html
PS12.	Pat Gehl, J.B. Systems, Woodland Hills, Calif., "The bottom line for CMMS financials", Plant Services magazine, October 1996, pp. 69-70
TSW13.	"TSW's Enterprise MPAC – Turning Client-Server Technology into a Business Advantage", August 1996, pp. 7-9
GP14.	GP MaTe Maintenance Management Software CMMS http://www.gpsonline.com/gpmate.html
TM15.	E-mail from Thomas J. Murphy Ph.D., Director Product Development - GP Solutions, 30 June 1997
FS16.	CMMS Integrated with Facilities Systems http//:www.mt-online.com/current/2-97mis.html
DS17.	DynaStar 2000 (http://www.scai.com/~scaidy/scaprod1.htm)
PMC18.	PMC for Windows (http://www.dpsi-cmms.com/products/pmc/pmc.htm)

Item	Reference
Sho19.	Shoham Consulting & Associates Home Page http://tribeca.ios.com/~shoham/WOT.HTML
MM20.	Maintenance Management http://ws.bassengineering.com/bass/main_man.htm
CM21.	Computerized maintenance management software CMMS http://www.mcs.net/~lenghoff/maintenance.html
PL22.	PlantLIFE MAINTENANCE (http://www.rebus/com/products/plant/maintenance.html)
MPu23.	MPulse CMMS (http://www.spectech.com/mpulse/mp_prod.html)
DG24.	E-mail from Dennis Gonyier, Vice President of Plus Delta Performance, 9 July 1997
SO25.	SOMAX/Windows System Overview (http://www.somax.com/sy_ovrvw.html)
MK26.	Moshe Krieger, S. Lemire "Restricted Object Based Design of Event Driven Commercial Software"
MK27.	Moshe Krieger, "Multiactivity Paradigm for the Coordination of Flexible Manufacturing Systems (FMSs)" CASCON '94 Integrated Solutions (CD ROM), Toronto, Ontario, 1994.
AY28.	Robert Au-Yang B Eng., COORDINATION-BASED SOFTWARE DESIGN; TOWARD COMPONENT-BASED SOFTWARE DESIGN, M. Eng. Report, Ottawa-Carleton Institute for Electrical Engineering, 1998.
ALSR29.	"Architecture for Large-Scale Reuse", AT&T Technical Journal, November / December 1992.
AOOS30	Adaptable Object-Oriented Software Systems (http://progwww.vub.ac.be/pools/Adaptable/ontents.html)
CHU31	Chubukjian, A, "Extended PAL Editor: An Example of an Interactive Real-Time System Specification Tool", M.Eng. Report, University of Ottawa, Canada, April 1990.
HAR32	Harvey, R A, "Verification of Concurrent System Specifications Using Temporal Logic", MASc thesis, University of Ottawa (July 1989)
SAN33	San Martin, R E, "Prototype based design of embedded systems", MASc thesis, University of Ottawa, Canada (July 1990)
ES34	Moshe Krieger, "Embedded Systems Lecture Slides and Article Reprints", 1995, p ES's MA systems - 8
SL35	S. Lemire, "Multiactivity as a Restricted Object Based Design Approach", M.Sc. thesis, Carleton University, June 1993.
V36	Dr. M.R. Vigder, J.C. Dean, "Managing Long-Lived COTS Based Systems", National Research Council of Canada, (http://wwwsel.iit.nrc.ca/seldocs/cotsdocs/NRC41587.pdf)

8. Annex A - CMMS Functionality Tables (From Selected Commercial CMMSs)

Table A-1 - DynaStar 2000 Features and Capabilities

DynaStar 2000 Feature	DynaStar 2000 Capabilities
Equipment Manager	• User-definable fields • Warranty information • levels of equipment hierarchy • Equipment/component drawings • Users have the ability to customize their equipment hierarchy structure using up to six levels of detail. Instantly access equipment and component drawings, warranty information and a complete machine parts cross-reference.
Personnel Manager	• Personnel Photo • Identification • Skill Levels • Labor Rates • Crew Information • Stores and retrieves information on an individual's skill level, labor rates, availability, and course certification.
Work Order Manager	• Imports work order Requests • Lists Lockout/Tagout Procedures • Reserves Parts for work orders • Cost Accounting available by work orders • Provides the ability to quickly create and issue work orders with only a few keystrokes. During work order generation, a simple "click" of a button allows users to view all pending work requests and PM's for a given machine.
PM Manager	• User-defined PM Rules • Calendar Overview of PMs • Common Task Library • Parts Required • Users set up their own rules for PM scheduling and next due date calculations. The "PM rules" function eliminated duplication of PM paperwork commonly associated with monthly, quarterly, semi-annual, and annual inspections.
Scheduling Manager	• Multiple Level of Scheduling • User-defined • Auto-scheduling • Flexible On-screen Scheduling • Prints Schedule at Several Stages • Maintains jobs and generates work schedules based on personnel availability, skill level, equipment, priority, and available parts.
Inventory Manager	• On-screen Parts History • In-house Tool Usage Tracking • Up-to-date Vendor Information • Flags Parts for Reorder • Maintains parts information such as inventory location(s), quantity at each location, quantity reserved and quantity on order. Historical usage • and ordering information can also be quickly accessed and viewed by a simple "click" of the mouse.
Reports Manager	• Costing by Job or Equipment • Complete History Reports • Inventory Control Reports • Detail and Summary Reports • Produce detailed or summary reports used for scheduling summarization, job and equipment costing, maintenance histories, inventory • transactions and purchase order generation, and ordering information history.

Table A-2 - MP2 Professional 5.0 Features and Capabilities

MP2 Feature	MP2 Capabilities
Equipment Module	• Nameplate feature • Meters • Bill of Materials • Component Tree Feature • Safety Notes Field • Graphics Viewer • Sort and Filter • On-screen history
Work Order Module	• Preventive maintenance work orders • Corrective maintenance work orders • Labour Costs and labour hours • Call-In requests • Automatic Warranty tracking • Work flow management • Equipment or location based work orders • E-mail work order requests
Task Module	• Resource scheduling • Unlimited task instructions • Parts availability • Craft sequencing • Labour assignments • Task assignment
Purchasing Module	• Automatic POs and requisitions • Multiple approval levels • Purchasing Process • Control and tracking Features • Electronic Data Interchange • Granger Electronic Catalogue Interface • Multi-location receiving
Inventory Module	• Multi-level inventory management • Main inventory management • Substitute parts information • Usage data • LIFI, FIFO • ABC or EOQ analyses • Bar-code option • Global inventory management
Labour Module	• Multiple wage codes • Time keeping • Training histories
Security Module	• Read-only access • Menu-level access • Field-level access • Global security • Graphic Tree view • System wide changes
Statistical Predictive Maintenance	• Track process variables • specify tolerances • Condition monitoring
Analysis and Reporting Module	• Work order history • Equipment history • Inventory items • Documentation • Audit trail • Crystal reporting

MP2 Feature	MP2 Capabilities
	• Customizable reporting • E-mail output

Table A-3 - PMC For Windows Features and Capabilities

PMC For Windows Feature	PMC For Windows Capabilities
Work Order Scheduling With Prioritization	Work Order scheduling with weekly, 2-week, and monthly calendars; a user-defined calendar for valid days; and maintenance scheduling by day, meter/miles, and count/fuel specify parts, labor, and priority.
Complete purchasing system	automatically generates and prints purchase orders, processes receipts, lists approved suppliers, and monitors pricing.
Labour	forecasting and usage with comprehensive address book profiles Time-card entry system
Inventory	parts management including parts forecasting and usage, physical inventory system, multiple locations per item, non-stocked parts capabilities, part-to-asset cross references, and online warranty tracking. Automatic adjustments when parts are reserved, used, received, and purchased plus reorder alerts. Analysis of parts inventory, purchases, projects, budgets, downtime, and contractor services and warranties.
Maintenance History	Comprehensive maintenance histories
Reporting	More than 350 reports and graphs--that can be customized--on every aspect of your operation
OLE	Object Linking and Embedding (OLE) features so you can instantly access CAD drawings, schematics, spreadsheets, and other graphics
Interfaces	Interfaces for add-on products such as CD-ROM, Bar Code, telephony, and more.
Security	Flexible, multi-level security assignments

Table A-4 - WOT Features and Capabilities

WOT Feature	WOT Capabilities
Equipment Maintenance	Equipment lists, employee list, parts, vendors, PM and CM work orders, gathering accurate labour time (with use of small hand-held bar-code programmable readers). Full track of materials used for work orders (cost, vendor, contractor markup ext.).
Work Order Scheduling	Scheduled for the year using a resource leveling algorithm
Security Access	3 levels (supervisor, contract manager and clerk access).
Bar-Code Reader	Downloading all readers in 1 min.
Reports	Analyzing resource, cost of maintenance, equipment record card and more.

Table A-5 - Bass Associates Maintenance Management System Features and Capabilities

Bass Associates Maintenance Management System Feature	Bass Associates Maintenance Management System Capabilities
Database	ODBC (can be accessed by other databases). - Different databases can be used. Databases include Equipment ID List, Maintenance work orders, Equipment failure history, tool and test eqpt calibration record, equipment calibration & test record, utility cost & consumption, employee time cards.
Reports	Several
Work orders	Generate PM work orders
Requests for maint	Can be entered from any computer on the network.
Security	Personnel with access can generate reports from their desk top.
Work orders	Automatically generated
Purchasing	Effective purchasing management system

Table A-6 - Atlas Equipment Manager (2000) Features and Capabilities

Atlas Equipment Manager (2000) Feature(s)	Atlas Equipment Manager (2000) Capabilities
Information management	• Equipment database • Vendor database • Contractors database • Manufacturer database • Parts database • Customer database • Buyer database • Department database • Employee database • Projects database
Equipment / Asset Management	• Checkout • Location Tracking • Equipment Charges/Rental • Projects • Subassembly Tracking • Inventory • Equipment Value • Service History • Equipment Sales History • Cost History • Warranty Tracking
Work Order System	• Automatically generate work orders with these triggers: • Time • Meter • Time or Meter • Variables • On Receiving • On Check In • Generate repair orders and service requests • Generate project work orders • Generate employee training work orders

Atlas Equipment Manager (2000) Feature	Atlas Equipment Manager (2000) Capabilities
Inventory and Purchasing	• Automatically generate PO's or purchase requisitions • Track current PO's and purchase history • Track on-hand quantities, reorder points and parts history • Track parts with serial numbers • Stockrooms and multiple location inventory tracking
Invoicing and Charge Centres	• Track costs and generate invoices for service maintenance • Automatically track maintenance costs for departments or charge centers • Use billing codes for customers
Statistical Predictive Maintenance	• Monitor conditions for each piece of equipment • Automatically generate Predictive Maintenance work orders from user defined variables
Additional Features	• Automatically generate professional predefined reports • Crystal Reports report writer • User defined fields and formatting • Multiple exporting options • DDE (Dynamic Data Exchange) • OLE (Object Linking and Embedding) • Security (multilevel user defined)
Networking	• Multi-user data file • Shared databases

Table A-7 - PlantLIFE Maintenance Features and Capabilities

PlantLIFE Feature	PlantLIFE Capabilities
Work orders	PlantLIFE MAINTENANCE generates Work Orders for CM and PM activities. Equipment, parts, and personnel are assigned to the Work Order. Safety measures are added to ensure that the work is performed in a safe manner. When work is complete, all personnel data, parts, nature of deficiencies, corrective or preventive measures taken, changes to the original Work Order, and completion date are entered into the PlantLIFE MAINTENANCE program. The Work Order may now be posted and removed from the daily workflow. Posted Work Orders may be viewed at anytime using the Equipment profile or Work Order system options.
Work Scheduling	Scheduled work by discipline may be viewed and modified based on available manpower. The frequency of Preventive Maintenance work can be altered, and low priority work can be delayed.
Equipment Profile and history	PlantLIFE MAINTENANCE enables the user to manage all equipment data. The program's flexible and configurable data input capabilities allow for data import from external sources. The Equipment profile is included in the PlantLIFE MAINTENANCE product, or may also be directly accessed in PlantLIFE ACCESS, taking advantage of the graphical navigation feature by a simple mouse click on the appropriate equipment symbol. Links from the Equipment profile screens to PlantLIFE ACCESS provide instantaneous display of all drawings and documents that pertain to the equipment. PlantLIFE MAINTENANCE enables the assignment of Preventive Maintenance tasks to the equipment. The frequency and frequency intervals of such tasks can be set or modified. The Equipment profile screen enables the user to view all posted and active Work Orders for a specified piece of equipment. The Equipment history option will display immediately all Preventive and Corrective Maintenance that was performed on the named equipment item. The user is then able to identify repeatable trouble spots or problem types.
Part Maintenance	Parts information is integrated into many components of the PlantLIFE MAINTENANCE

PlantLIFE Feature	PlantLIFE Capabilities
	program. The information can be imported via PlantLIFE's configurable data input capability. This enables data from systems such as SAP/R3 or other part management systems to be available within PlantLIFE MAINTENANCE. Part information can easily be edited in PlantLIFE, and the optimum use of part manufacturers or vendors can readily be handled. As parts get used on the scheduled Work Order, the quantity of the parts is automatically adjusted. Purchase Order generation is triggered as stock levels fall below a minimum level.
Close Integration With PlantLIFE Modules	PlantLIFE MAINTENANCE is closely integrated with all other PlantLIFE software modules. PlantLIFE ACCESS gives the user access to all up-to-date documentation, and PlantLIFE SAFETY links in all the safety procedures.

Table A-8 - Enterprise MPAC Features and Capabilities

Enterprise MPAC Feature	Enterprise MPAC Capabilities
Asset Management	Record comprehensive information about each asset (description, drawing, technical information, inspection readings/limits, meter readings, cost, transaction history. Hierarchical asset structure.
Work Force Management	hierarchical organization of work
PM and PdM	PM and PdM separated into sub-tasks and steps. Attach parts lists to work orders. When a task is triggered, the work order is generated automatically
Work Requests	alerts to a required action. Records of maintenance requests kept.
Work Order Planning	Create, plan, schedule, revise work packages. Create work orders, order materials,
WO Scheduling	manage work order backlog and future PM/PdM tasks (manual or automatic).
Component Tracking (Optional)	oversees removable assets and parts. Components are tracked as they rotate between assets. Historical data tracked
Project Tracking (Optional	manages large projects
Warranty Tracking (Optional)	controls warranty information including terms, conditions, & work restrictions
Advanced Security	access control
Performance Indicators	Tracks and reports indicators affecting asset and work management cycles
Electronic Document Management System	Manage critical Document and information
Stores	Manages all warehouse and inventory functions (requisitioning, issuing, returns, receiving, physical inventory, inventory accounting)
Procurement	Buyers WorkBench, quotation processing, purchase orders, purchase agreements, company register, buyers
Electronic Commence	Leverage FAX, EDI, Internet/electronic network resources
Project Management	Integrates with 3rd party PM software
Enterprise Mail	E-Mail

Table A-9 - MPulse Features and Capabilities

MPulse Feature	MPulse Capabilities
Work Orders	Automatic generation, based on date. elapsed time, meter usage
Maintenance History	Based on work orders
Report	Dozens available
Database of task & safety instructions	Link them to work orders
Personnel	Skills qualifications
Parts and inventory	Maintains and tracks tools, parts and supplies. Automatic deducted from inventory If included on the work order. Lists of inventory suppliers and manufacturers maintained.
Purchase Requisitions	Generates and maintains history
Keys and Locks	Tracking possession
CAD Viewer	Links DFX and DWG files to maintenance tasks - can print out

Table A-10 - SOMAX Information Management System Features and Capabilities

Feature	Capabilities
Equipment	Equipment name, location, make, model, serial no., purchase vendor, maintenance vendor, warranty
Parts/Inventory, Vendors	Part Id, location, quantities for issue and purchase, supports multi-store rooms, supports parts and issue returns, supports quantity controlled parts counts, vendor specs and ABC analysis, Parts/vendor cross reference, Parts History Module records all changes to Parts Module with a date/time/user stamp, records parts usage by equipment, etc.
Personnel / Time Cards / Mill Call	Time cards system reports hours and costs to work orders and account codes, on-line retrieval of employee data, event history database to record performance and training, maintain personnel records, view date starting, current pay rate, view or change assigned shift, class, craft, etc., work history, contractor information, assign employee time to work orders
PM	Supports PdM/Proactive/Preventive activities; PM scheduling may be bypassed and work orders produced on demand; forecast system for PM work orders, balance workload, order parts, and plan equipment downtime; Meter readings initiate PM work orders; schedule work orders by Due (days), Done (since last PM), meter (miles, hours, cycles)Meter or Due, high or low warning values
Purchasing and PO History	Paperless and electronic purchasing, automatic purchasing at user-defined levels; Purchase orders and purchase requests created, routed, issued and tracked; models PRs from existing or historical POs; consolidate same-vendor PR to one PO; bar-code capability, purchasing history
Work Orders, Work Requests, Equipment History	Electronic work order routing, tracks CM and PM work orders form initiation, planning, scheduling, performance and completion. Records estimated and actual costs, material and parts reservation, schedule crafts, equipment downtime statistics

9. Annex B - The Process Activity Language

The PAL expressions describe the responses which are implemented by the response (or process) managers.(including activity precedence relations) and interaction with other responses. Specific primitives and constructs have been defined. Figure B-1 details an example of the PAL process and Figures B-2, B-3 and B-4 illustrate the different PAL constructs and conditions. Below is a summary of the principle PAL elements:

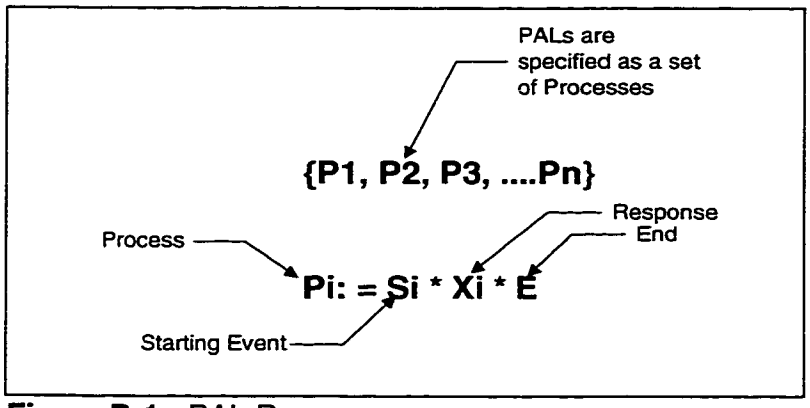


Figure B-1 - PAL Processes

Three primitives are required to represent processes. They are:

1. Activity - a schedulable unit of work with a bounded execution time;

2. Set Process Condition - establishes decision points. The syntax for Process Condition is pc_1. **STP(pc_1, expression)** is the expression used to represent Set Process Condition.

3. Delay - used for timing purposes. **D(n)** is used to express a delay of n time units.

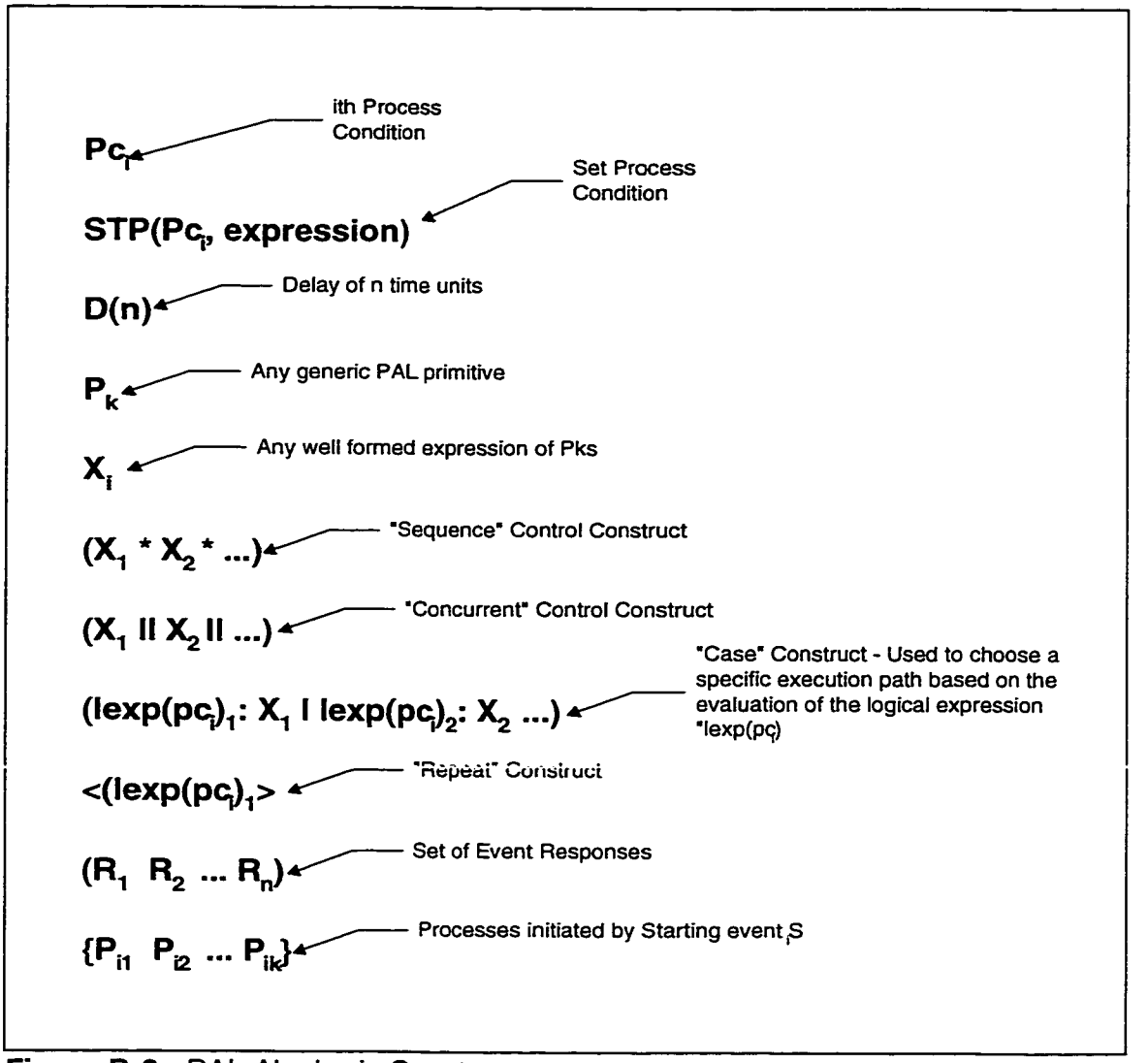


Figure B-2 - PAL Algebraic Constructs

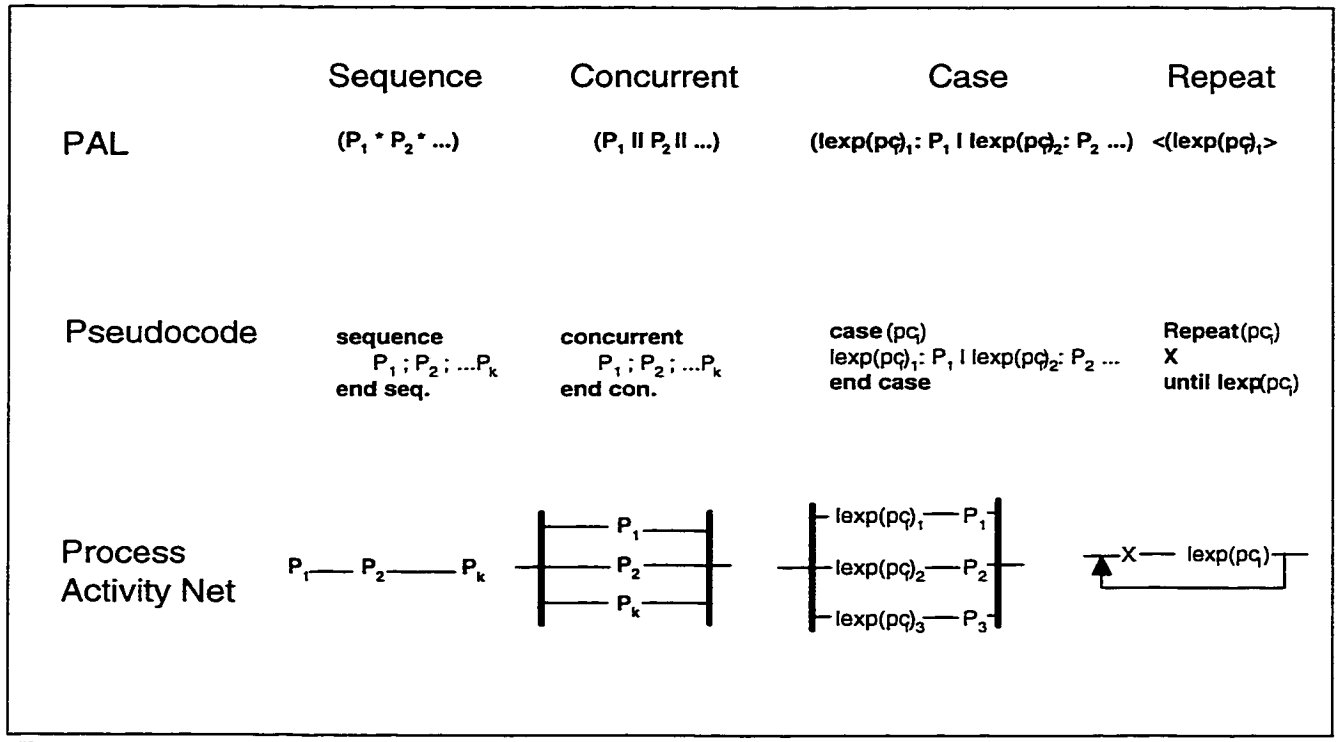


Figure B-3 - PAL Control Constructs

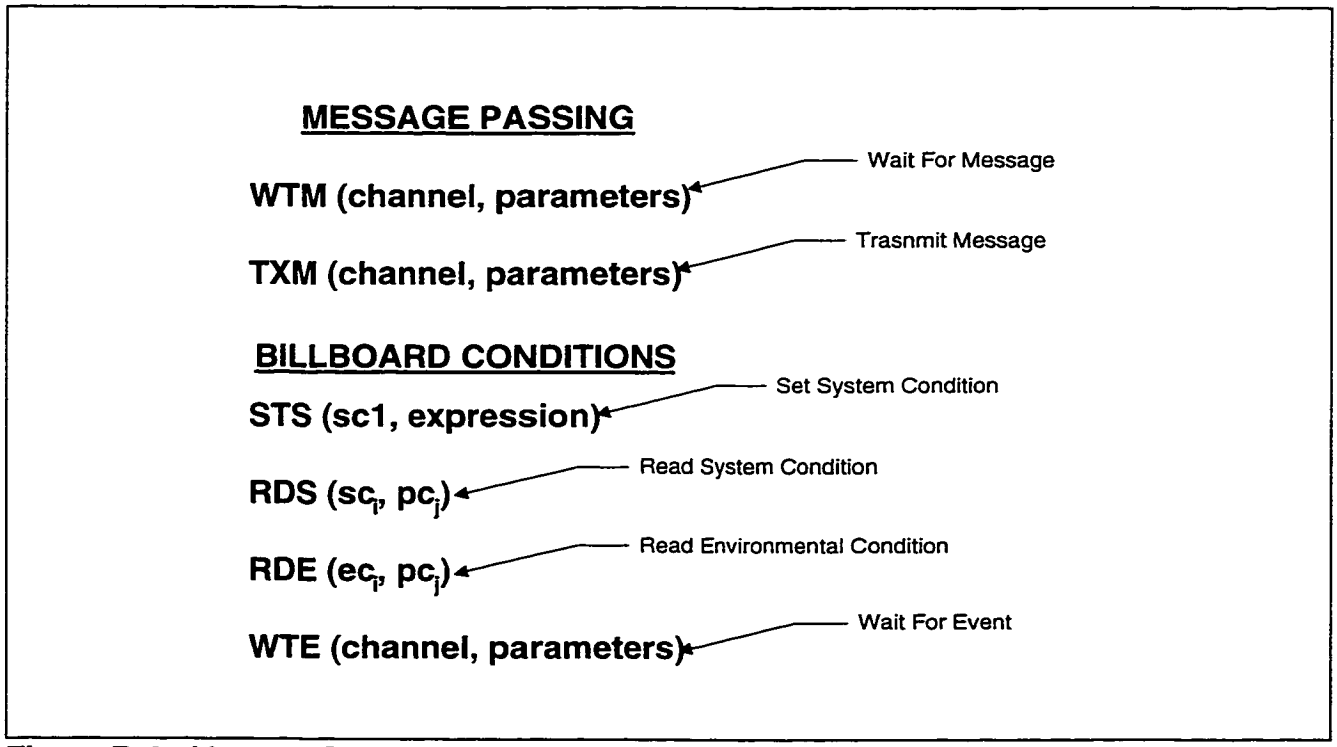


Figure B-4 - Message Passing and Billboard Conditions

10. Annex C - Coordinator Code

Coord.cpp

```
// Coord.cpp : Defines the initialization routines for the DLL.
//

#include "stdafx.h"
#include <afxdllx.h>
#include "Coord.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern  HWND g_hwndTrace;
extern  UINT g_uiTraceMessage ;



//-----------------------------------------------------
// Shared control information of the coordinator.
//
#pragma data_seg(".CoordData")

BOOL        DebugMsg = FALSE;
BOOL        coordBusy = FALSE;
int                     curEID = 0;
int                     coordInstance = 0;

#pragma data_seg()

// Instruct the linker to make the Shared section
// readable, writable, and shared.
//#pragma comment(lib, "MSVCRTD " "-section:.CoordData,rws")



// April '98 Charlie Cox
//
// Allocate the component object variables used by the coordinator
//
_InvDbHandler           objInvDB;
_PODbHandler            objPODB;
_VenDbHandler           objVenDB;
_POFormClass            objPOForm;
_DbHandler              objDB;

// Allocate the system variables used by the coordinator.

CAddInvItem a_inv_i_pkg;
CDelInvItem d_inv_i_pkg;
CCreatePO   create_PO_pkg;
CCreateWO   create_WO_pkg;
CWO_Statu   WO_Status_pkg;
NM          NM_pkg;
BOOL init_fail = FALSE;
int instance_image = 0; // Should be the same as coordInstance.

//-----------------------------------------------------


static AFX_EXTENSION_MODULE CoordDLL = { NULL, NULL };

extern "C" int APIENTRY
DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID lpReserved)
{
UNREFERENCED_PARAMETER(lpReserved);

if (dwReason == DLL_PROCESS_ATTACH)
{
MY_TRACE("COORD.DLL Initializing!...\n");

if (DebugMsg)
AfxMessageBox("COORD.DLL Initializing!");

// Extension DLL one-time initialization
if (!AfxInitExtensionModule(CoordDLL, hInstance))
return 0;

// Insert this DLL into the resource chain
// NOTE: If this Extension DLL is being implicitly linked to by
//  an MFC Regular DLL (such as an ActiveX Control)
//  instead of an MFC application, then you will want to
//  remove this line from DllMain and put it in a separate
//  function exported from this Extension DLL.  The Regular DLL
//  that uses this Extension DLL should then explicitly call that
//  function to initialize this Extension DLL.  Otherwise,
//  the CDynLinkLibrary object will not be attached to the
//  Regular DLL's resource chain, and serious problems will
// result.

new CDynLinkLibrary(CoordDLL);

// Also needs to initialize the coordinator
// when attached.
MY_TRACE("Init Coordinator...\n");
if (DebugMsg)
AfxMessageBox("Init Coordinator");
init_coord();

}
else if (dwReason == DLL_PROCESS_DETACH)
{
MY_TRACE("COORD.DLL Terminating!...\n");
if (DebugMsg)
AfxMessageBox("COORD.DLL Terminating!");
// Also need to deallocate resources used by the
// coordinator.
MY_TRACE("Exit Coordinator...\n");
if (DebugMsg)
AfxMessageBox("Exit Coordinator");
exit_coord();
// Terminate the library before destructors are called
AfxTermExtensionModule(CoordDLL);
}
return 1;   // ok
}


//-----------------------------------------------------
//
void init_coord()
{
// Declare the necessary local variables to setup connections to the
// component DLLs.
//
LPDISPATCH              pDisp;
LPUNKNOWN               pUnk;
CLSID                   clsid;
int rc = 0;

// Compose structure needed by the interface of the components.
// Note that there are multiple components required to support a given
// response.
// We'll be using the wrapper class for each of these components.

MY_TRACE("..Coordinator initialize resources...\n");
MY_TRACE("..Initialize InvDB_Handler...\n");
if (DebugMsg)
AfxMessageBox("..Initialize InvDB_Handler");

CLSIDFromProgID(L"InvDB_Handler.InvDBHandler", &clsid);
CString sguid;
sguid.Format("CLSID = %x-%x-%x...\n",clsid.Data1,clsid.Data2,clsid.Data3);
MY_TRACE(sguid);
if (GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
VERIFY(pUnk->QueryInterface(IID_IDispatch,
(void**)&pDisp) == S_OK);
objInvDB.AttachDispatch(pDisp);
pUnk->Release();
MY_TRACE("....IDispatch reference for InvDB_Handler.InvDBHandler
obtained and attached...\n");
if (DebugMsg)
AfxMessageBox("....IDispatch reference for InvDB_Handler.InvDBHandler
obtained and attached");

}
else {
MY_TRACE("....GetActiveObject fail and needs to create one...\n");
if (!objInvDB.CreateDispatch("InvDB_Handler.InvDBHandler")) {
MY_TRACE("....InvDB_Handler.InvDBHandler not found...\n");
MY_TRACE("....Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("....InvDB_Handler.InvDBHandler not found. Exit coord init");
init_fail = TRUE;
return;
            }
}


MY_TRACE("..Initialize PODB_Handler...\n");
if (DebugMsg)
AfxMessageBox("..Initialize PODB_Handler");

CLSIDFromProgID(L"PODB_Handler.PODBHandler", &clsid);
sguid.Format("CLSID = %x-%x-%x...\n",clsid.Data1,clsid.Data2,clsid.Data3);
MY_TRACE(sguid);

if (GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
VERIFY(pUnk->QueryInterface(IID_IDispatch,
```

```
(void**)&pDisp) == S_OK);
objPODB.AttachDispatch(pDisp);
pUnk->Release();
MY_TRACE("....IDispatch reference for PODB_Handler.PODBHandler
obtained and attached...\n");
if (DebugMsg)
AfxMessageBox("....IDispatch reference for PODB_Handler.PODB_Handler
obtained and attached");

}
else {
MY_TRACE("....GetActiveObject fail and needs to create one...\n");
if (!objPODB.CreateDispatch("PODB_Handler.PODBHandler")) {
MY_TRACE("....PODB_Handler.PODBHandler not found...\n");
MY_TRACE("....Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("....PODB_Handler.PODBHandler not found. Exit coord
init");
init_fail = TRUE;
return;
        }
}

MY_TRACE("..Initialize VenDb_Handler...\n");
if (DebugMsg)
AfxMessageBox("..Initialize VenDb_Handler");

CLSIDFromProgID(L"VenDb_Handler.VenDbHandler", &clsid);
sguid.Format("CLSID = %x-%x-%x...\n",clsid.Data1,clsid.Data2,clsid.Data3);
MY_TRACE(sguid);
if (GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
VERIFY(pUnk->QueryInterface(IID_IDispatch,
(void**)&pDisp) == S_OK);
objVenDB.AttachDispatch(pDisp);
pUnk->Release();
MY_TRACE("....IDispatch reference for VenDB_Handler.VenDBHandler
obtained and attached...\n");
if (DebugMsg)
AfxMessageBox("....IDispatch reference for VenDB_Handler.VenDbHandler
obtained and attached");

;
else {
MY_TRACE("....GetActiveObject fail and needs to create one...\n");
if (!objVenDB.CreateDispatch("VenDb_Handler.VenDbHandler")) {
MY_TRACE("....VenDB_Handler.VenDBHandler not found...\n");
MY_TRACE("....Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("....VenDb_Handler.VenDbHandler not found. Exit coord
init");
init_fail = TRUE;
return;
        }
}

MY_TRACE("..Initialize DB_Handler...\n");
if (DebugMsg)
AfxMessageBox("..Initialize DB_Handler");

CLSIDFromProgID(L"DB_Handler.DBHandler", &clsid);
sguid.Format("CLSID = %x-%x-%x...\n",clsid.Data1,clsid.Data2,clsid.Data3);
MY_TRACE(sguid);
if (GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
VERIFY(pUnk->QueryInterface(IID_IDispatch,
(void**)&pDisp) == S_OK);
objDB.AttachDispatch(pDisp);
pUnk->Release();
MY_TRACE("....IDispatch reference for DB_Handler.DBHandler obtained
and attached...\n");
if (DebugMsg)
AfxMessageBox("....IDispatch reference for DB_Handler.DBHandler obtained
and attached");

}
else {
MY_TRACE("....GetActiveObject fail and needs to create one...\n");
if (!objDB.CreateDispatch("DB_Handler.DbHandler")) {
MY_TRACE("....DB_Handler.DBHandler not found...\n");
MY_TRACE("....Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("....DB_Handler.DBHandler not found. Exit coord init");
init_fail = TRUE;
return;
        }
}

MY_TRACE("..Initialize POForm...\n");
if (DebugMsg)
AfxMessageBox("..Initialize POForm_Handler");

CLSIDFromProgID(L"POForm.POFormClass", &clsid);
sguid.Format("CLSID = %x-%x-%x...\n",clsid.Data1,clsid.Data2,clsid.Data3);
MY_TRACE(sguid);
if (GetActiveObject(clsid, NULL, &pUnk) == S_OK) {
VERIFY(pUnk->QueryInterface(IID_IDispatch,
(void**)&pDisp) == S_OK);
objPOForm.AttachDispatch(pDisp);
pUnk->Release();
MY_TRACE("....IDispatch reference for POForm.POFormClass obtained and
attached...\n");
if (DebugMsg)
AfxMessageBox("....IDispatch reference for POFormClass.POFormClass
obtained and attached");

}
else {
MY_TRACE("....GetActiveObject fail and needs to create one...\n");
if (!objPOForm.CreateDispatch("POForm.POFormClass")) {
Y_TRACE("....POForm.POFormClass not found...\n");
MY_TRACE("....Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("....POFormClass.POFormClass not found. Exit coord init");
init_fail = TRUE;
return;
        }
}

// Also set the default database access type to use DAO.
// This attribute can be moved to a programmable setup
// command later from the user.
// The other setup parameter is the location of the
// database.
MY_TRACE("..Set InvDB access selection to DAO...\n");
if (DebugMsg)
AfxMessageBox("..Set InvDB access selection to DAO");
objInvDB.SetInvDB_AccSel(1);
objInvDB.setInvDBTrace(FALSE);

MY_TRACE("..Set PODB access selection to DAO...\n");
if (DebugMsg)
AfxMessageBox("..Set PODB access selection to DAO");
objPODB.SetPODB_AccSel(1);
objPODB.setPODBTrace(FALSE);

MY_TRACE("..Set VDB access selection to DAO...\n");
if (DebugMsg)
AfxMessageBox("..Set VDB access selection to DAO");
objVenDB.SetVenDB_AccSel(1);
objVenDB.setVenDBTrace(FALSE);

MY_TRACE("..Set DB access selection to DAO...\n");
if (DebugMsg)
AfxMessageBox("..Set DB access selection to DAO");
objDB.SetDB_AccSel(1);

objDB.setDBTrace(FALSE);

MY_TRACE("..Set POFormClass access selection to Excel...\n");
if (DebugMsg)
AfxMessageBox("..Set POFormClass access selection to DAO");
objPOForm.SetPOForm_AccSel(1);
objPOForm.setPOFormTrace(FALSE);


MY_TRACE("..Exit Coordinator Initialization...\n");
if (DebugMsg)
AfxMessageBox("..Exit Coordinator Initialization");

init_fail = FALSE;

// Increment the instance count
rc = invoke_coord(EID_info_incr_coord_instance);
MY_TRACE1("..increment instance rc=%d...\n", rc);
  return;
}

//
// Destruct the coordinator and its managed resources
//
void exit_coord()
{
int             rc = 0;

// Make sure to release the handles on the component object instances
MY_TRACE("..Coordinator clean-up on exit...\n");
objInvDB.ReleaseDispatch();
objPODB.ReleaseDispatch();
objVenDB.ReleaseDispatch();
objPOForm.ReleaseDispatch();
objDB.ReleaseDispatch();

// Increment the instance count
```

```
rc = invoke_coord(EID_info_decr_coord_instance);
MY_TRACE1("..decrement instance rc=%d...\n", rc);

}

//-----------------------------------------------------------------
// **** Coordinator Utilitiy Procedures ****

// These functions will initialize the member variables into the default
constructor values.

void init_create_PO_pkg ()
{
        create_PO_pkg.m_atc = 0.0;
        create_PO_pkg.m_atc2 = 0.0;
        create_PO_pkg.m_atc3 = 0.0;
        create_PO_pkg.m_atc4 = 0.0;
        create_PO_pkg.m_atno = _T("");
        create_PO_pkg.m_atno2 = _T("");
        create_PO_pkg.m_atno3 = _T("");
        create_PO_pkg.m_atno4 = _T("");
        create_PO_pkg.m_etc = 0.0;
        create_PO_pkg.m_etc2 = 0.0;
        create_PO_pkg.m_etc3 = 0.0;
        create_PO_pkg.m_etc4 = 0.0;
        create_PO_pkg.m_etno = _T("");
        create_PO_pkg.m_etno2 = _T("");
        create_PO_pkg.m_etno3 = _T("");
        create_PO_pkg.m_etno4 = _T("");
        create_PO_pkg.m_fst = 0.0;
        create_PO_pkg.m_ist = 0.0;
        create_PO_pkg.m_ist10 = 0.0;
        create_PO_pkg.m_ist11 = 0.0;
        create_PO_pkg.m_ist12 = 0.0;
        create_PO_pkg.m_ist2 = 0.0;
        create_PO_pkg.m_ist3 = 0.0;
        create_PO_pkg.m_ist4 = 0.0;
        create_PO_pkg.m_ist5 = 0.0;
        create_PO_pkg.m_ist6 = 0.0;
        create_PO_pkg.m_ist7 = 0.0;
        create_PO_pkg.m_ist8 = 0.0;
        create_PO_pkg.m_ist9 = 0.0;
        create_PO_pkg.m_poid = time_t(0);
        create_PO_pkg.m_pon = 0;
        create_PO_pkg.m_post = 0.0;
        create_PO_pkg.m_pqty = 0;
        create_PO_pkg.m_pqty10 = 0;
        create_PO_pkg.m_pqty11 = 0;
        create_PO_pkg.m_pqty12 = 0;
        create_PO_pkg.m_pqty2 = 0;
        create_PO_pkg.m_pqty3 = 0;
        create_PO_pkg.m_pqty4 = 0;
        create_PO_pkg.m_pqty5 = 0;
        create_PO_pkg.m_pqty6 = 0;
        create_PO_pkg.m_pqty7 = 0;
        create_PO_pkg.m_pqty8 = 0;
        create_PO_pkg.m_pqty9 = 0;
        create_PO_pkg.m_pst = 0.0;
        create_PO_pkg.m_satno = _T("");
        create_PO_pkg.m_satno2 = _T("");
        create_PO_pkg.m_satno3 = _T("");
        create_PO_pkg.m_satno4 = _T("");
        create_PO_pkg.m_satc = 0.0;
        create_PO_pkg.m_satc2 = 0.0;
        create_PO_pkg.m_satc3 = 0.0;
        create_PO_pkg.m_satc4 = 0.0;
        create_PO_pkg.m_tpc = 0.0;
        create_PO_pkg.m_va = _T("");
        create_PO_pkg.m_vna = _T("");
        create_PO_pkg.m_vno = _T("");
        create_PO_pkg.m_vpn = _T("");
}

void init_create_WO_pkg ()
{
        create_WO_pkg.m_won                     = _T("");
        create_WO_pkg.m_sn                      = _T("");
        create_WO_pkg.m_id          = time_t(0);
        create_WO_pkg.m_ib                      = _T("");
        create_WO_pkg.m_sd          = time_t(0);
        create_WO_pkg.m_tcd         = time_t(0);
        create_WO_pkg.m_tna                     = _T("");
        create_WO_pkg.m_combo_en                = _T("");
        create_WO_pkg.m_tl                      = _T("");
        create_WO_pkg.m_combo_en2               = _T("");
        create_WO_pkg.m_combo_en3               = _T("");
        create_WO_pkg.m_combo_en4               = _T("");
        create_WO_pkg.m_dmp                     = _T("");

        create_WO_pkg.m_ff                      = _T("");
        create_WO_pkg.m_location                = _T("");
        create_WO_pkg.m_mt                      = _T("");
        create_WO_pkg.m_etna                    = _T("");
        create_WO_pkg.m_wad                     = _T("");

}

void init_WO_Status_pkg ()
{
        WO_Status_pkg.m_wono =                  _T("");
        WO_Status_pkg.m_cs =                    _T("");
        WO_Status_pkg.m_sdate =                 time_t(0);
        WO_Status_pkg.m_tcdate =                time_t(0);
        WO_Status_pkg.m_tdate =                 time_t(0);
        WO_Status_pkg.m_comp_ques =             _T("");
        WO_Status_pkg.m_late_start_ques =       _T("");
        WO_Status_pkg.m_late_comp_ques =        _T("");
        WO_Status_pkg.m_started_ques =          _T("");

}

//-----------------------------------------------------------------

int handle_event(int eid)
{
        short   rc = 0;
        short   cur_iq = 0;
        short   ven_rc;
        short   sn_rc;
        short   tna_rc;
        short   tl_rc;
        short   save_rc;
        short   update2_rc;
        short   update_st;
        short   update_st2;
        short   update_co;
        short   update_co2;
        short   update_cs1;
        short   update_cs2;
        short   edit_rc;
        short   status_rc;
        short   update_rc;
        short   nm_rc;
        BOOL    venExisted;
        BOOL    snExisted;
        BOOL    tnaExisted;
        BOOL    tlExisted;

        CString strMsgText;

        BSTR    m_bstrVA = NULL;
        BSTR    m_bstrVPN = NULL;
        BSTR    m_r8ATC = NULL;
        BSTR    m_bstrETNA = NULL;
        BSTR    m_bstrDMP = NULL;
        BSTR    m_bstrFF = NULL;
        BSTR    m_bstrMT = NULL;
        BSTR    m_bstrLOCATION = NULL;
        BSTR    m_bstrEN = NULL;
        BSTR    m_bstrEN2 = NULL;
        BSTR    m_bstrEN3= NULL;
        BSTR    m_bstrEN4 = NULL;
        BSTR    m_bstrMPTD = NULL;
        BSTR    m_bstrETC  = NULL;
        BSTR    m_bstrETC2 = NULL;
        BSTR    m_bstrETC3 = NULL;
        BSTR    m_bstrETC4 = NULL;

        // Select the response for the event invoked.
        //
        switch (eid)
        {
//-----------------------------------------------------------------
//-----------------------------------------------------------------
//ADD INVENTORY ITEM (ATOMIC) EVENT RESPONSE

case EID_add_inv_item:
{
MY_TRACE("....Start Add Inventory Item Event Response ...\n");

                                        BOOL  pnExisted;
                                        short n_rc;
//-----------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE

rc = objInvDB.openInvDB();

if (rc == -1) {
MY_TRACE1("....Open CMMS Database - Failed rc=%d...\n", rc);
```

```
MY_TRACE("....Exit Coordinator...\n");
return 0;
                                        }
MY_TRACE("....Open CMMS Database - Successful...\n");

// Obtain the IPN, EPN and ITD
// They are the member variables of the dialog.
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CHECK PART NUMBER NOT IN INVENTORY TABLE

MY_TRACE("....Check to Ensure Item Does Not Already Exist in the
Database...\n");
n_rc = objInvDB.recExistPN(a_inv_i_pkg.m_mpn,&pnExisted);

// If uniqueness validation is successful, then
// create the record.


if ((n_rc == -1) || pnExisted) {
MY_TRACE("....MPN/EPN not unique...\n");
MY_TRACE2("....n_rc=%d  pnExisted=%s...\n",
n_rc, (pnExisted ? "True" : "False"));

rc = objInvDB.closeInvDB();

MY_TRACE1("....Close CMMS Database rc = %d...\n", rc);
MY_TRACE("....Exit Coordinator...\n");
return 0;
                                        }
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CREATE RECORD IN INVENTORY TABLE OF
CMMS DATABASE

rc = objInvDB.recCreate(a_inv_i_pkg.m_mptd, a_inv_i_pkg.m_mpn,
a_inv_i_pkg.m_sn, a_inv_i_pkg.m_atno, a_inv_i_pkg.m_etno,
a_inv_i_pkg.m_satno, a_inv_i_pkg.m_qr, a_inv_i_pkg.m_nur,
a_inv_i_pkg.m_iqty, a_inv_i_pkg.m_isl, a_inv_i_pkg.m_msl);


if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1("....Creation Failed rc=%d...\n", rc);
rc = objInvDB.closeInvDB();

MY_TRACE1("....Close CMMS Database rc = %d...\n", rc);
MY_TRACE("....Exit Coordinator...\n");
return 0;
                                        }

// If record creation is successful, acknowledge
// success
MY_TRACE("....Add Inventory Item - Successful...\n");
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objInvDB.closeInvDB();

MY_TRACE("....CMMS Database Closed...\n");

}
break;
//END OF ADD INVENTORY ITEM (ATOMIC) EVENT RESPONSE
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
//DELETE INVENTORY ITEM (ATOMIC) EVENT RESPONSE
case EID_del_inv_item:
{
MY_TRACE("....Start Delete Inventory Item Event Response...\n");
//-------------------------------------------------------------------------
//INVOKE ACTIVITY - OPEN CMMS DATABASE
BOOL  pnExisted;
short pn_rc;
rc = objInvDB.openInvDB();
if (rc == -1) {
MY_TRACE1(".....open inventory database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
                                        }
MY_TRACE("....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------------
//INVOKE ACTIVITY - CHECK PART NUMBER IN INVENTORY TABLE

MY_TRACE("....Check to Ensure Item Exists in Database...\n");
pn_rc = objInvDB.recExistPN(d_inv_i_pkg.m_mpn,&pnExisted);
if (pn_rc == -1) {
MY_TRACE(".....Item not present...\n");
MY_TRACE2(".....pn_rc=%d  pnExisted=%s...\n", pn_rc, (pnExisted ? "True"
: "False"));
rc = objInvDB.closeInvDB();
MY_TRACE1(".....clos inventory database rc = %d...\n", rc);
MY_TRACE("....exit coord...\n");
return 0;
                                        }
//-------------------------------------------------------------------------
//INVOKE ACTIVITY - DELETE INVENTORY ITEM

// If uniqueness validation is successful, then
// delete the record.

// Check for uniquness successful, delete the record
if ((pn_rc==-1)||pnExisted){


rc = objInvDB.recDelete(d_inv_i_pkg.m_mpn);
if (rc == -1) {
// If record deletion  has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
                                        }
                                }
MY_TRACE("....Deletion Successful...\n");

//-------------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objInvDB.closeInvDB();

MY_TRACE("....CMMS Database Closed...\n");
return 0;

// If record deletion is successful, acknowledge
// success
// Close the inventory database
rc = objInvDB.closeInvDB();
MY_TRACE1(".....close inventory database rc = %d...\n", rc);

break;
//END OF DELETE INVENTORY ITEM EVENT RESPONSE
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
//SUBMIT PURCHASE ORDER (COMBINED) EVENT RESPONSE
case EID_purchase_order_submit:
MY_TRACE("....Start SUBMIT PURCHASE ORDER (Combined) Event
Response...\n");

if (DebugMsg)
AfxMessageBox("....Start PO Creation event response");

// **** Autofill Vendor Information ****
// Retrieve the Vendor information from the vendor database.
// This activity will update the vendor information on the purchase order form
// if the vendor was found in the database.
// Open the Vendor database
//-------------------------------------------------------------------------
//GET VENDOR INFORMATION (ATOMIC) EVENT RESPONSE
//INVOKE ACTIVITY - OPEN CMMS DATABASE
MY_TRACE(".....Start GET VENDOR INFORMATION (Atomic) Event
Response...\n");
rc = objVenDB.openVenDB();
 if (rc == -1) {
MY_TRACE1(".....Open vendor database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
                }
MY_TRACE("....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------------
// INVOKE ACTIVITY - CHECK IF VENDOR NAME IN VENDORS TABLE
MY_TRACE(".....Check If Vendor Name In CMMS Database...\n");
ven_rc = objVenDB.recExistVN(create_PO_pkg.m_vna,&venExisted);

// If uniqueness validation is successful, then
// create the record.
if (ven_rc == -1) {
MY_TRACE(".....rec Exist check failed...\n");
MY_TRACE1(".....ven_rc=%d...\n", ven_rc);
rc = objVenDB.closeVenDB();
MY_TRACE1(".....close vendo database rc = %d...\n", rc);
MY_TRACE("....ext coordinator...\n");
return -1;
                }
//-------------------------------------------------------------------------
//INVOKE ACTIVITY - GET RECORD FROM VENDORS TABLE
// Try to fill in Customer information
if (venExisted) {
MY_TRACE(".....Get Record From VENDORS Table...\n");

// vendor file exist, retrieve the record.
// But first allocate the buffer for the info.
// Since the order is not confirmed, therefore,
// do not create new database record if vendor
// not in database.
```

```
SysFreeString(m_bstrVA);
m_bstrVA = create_PO_pkg.m_va.AllocSysString();
SysFreeString(m_bstrVPN);
m_bstrVPN = create_PO_pkg.m_vpn.AllocSysString();
rc = objVenDB.getRecInfo(create_PO_pkg.m_vna, &m_bstrVA,
&m_bstrVPN);

create_PO_pkg.m_va = m_bstrVA;
create_PO_pkg.m_vpn = m_bstrVPN;

if (DebugMsg) {
strMsgText.Format("VA=%s VPN=%s", create_PO_pkg.m_va,
create_PO_pkg.m_vpn);
AfxMessageBox(strMsgText);
                    }

if (rc == -1) {
// If record retrieval has failed, acknowledge
// failure
MY_TRACE1(".....vendor Database Get Failed rc=%d...\n", rc);
rc = objVenDB.closeVenDB();
MY_TRACE1(".....close ven database rc = %d...\n", rc);
MY_TRACE(".....ex coordinator...\n");
return -1;
}
// If record creation is successful, acknowledge
// success
MY_TRACE(".....Record Retrieval - Successful...\n");
          }
//-------------------------------------------------------------------
// INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objVenDB.closeVenDB();
MY_TRACE(".....CMMS Database Closed...\n");
MY_TRACE("..........\n");
//END OF GET VENDOR INFORMATION (ATOMIC) EVENT RESPONSE
//-------------------------------------------------------------------

//-------------------------------------------------------------------
// GET COST DATA (ATOMIC) EVENT RESPONSE
MY_TRACE(".....Start Get Cost Data (Atomic) Event Response...\n");
// INVOKE ACTIVITY - OPEN CMMS DATABASE

rc = objPODB.openPODB();
if (rc == -1) {
MY_TRACE1(".....open PO database fail rc=%d...\n", rc);
MY_TRACE(".....exit coordinator...\n");
return 0;
                              }
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getETC(create_PO_pkg.m_etno, &create_PO_pkg.m_etc);
if (rc==0) {
MY_TRACE(".....Get Equipment Type 1 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_etc);
                    }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getETC2(create_PO_pkg.m_etno2, &create_PO_pkg.m_etc2);
if (rc==0) {
MY_TRACE(".....Get Equipment Type 2 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_etc2);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getETC3(create_PO_pkg.m_etno3, &create_PO_pkg.m_etc3);
if (rc==0) {
MY_TRACE(".....Get Equipment Type 3 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_etc3);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getETC4(create_PO_pkg.m_etno4, &create_PO_pkg.m_etc4);
if (rc==0) {
MY_TRACE(".....Get Equipment Type 4 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_etc4);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getATC(create_PO_pkg.m_atno, &create_PO_pkg.m_atc);

if (rc==0) {
MY_TRACE(".....Get Assembly Type 1 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getATC2(create_PO_pkg.m_atno2, &create_PO_pkg.m_atc2);

if (rc==0) {
MY_TRACE(".....Get Assembly Type 2 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getATC3(create_PO_pkg.m_atno3, &create_PO_pkg.m_atc3);

if (rc==0) {
MY_TRACE(".....Get Assembly Type 3 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getATC4(create_PO_pkg.m_atno4, &create_PO_pkg.m_atc4);

if (rc==0) {
MY_TRACE(".....Get Assembly Type 4 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getSATC(create_PO_pkg.m_satno, &create_PO_pkg.m_satc);

if (rc==0) {
MY_TRACE(".....Get Sub-Assembly Type 1 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getSATC2(create_PO_pkg.m_satno2,
&create_PO_pkg.m_satc2);

if (rc==0) {
MY_TRACE(".....Get Sub-Assembly Type 2 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getSATC3(create_PO_pkg.m_satno3,
&create_PO_pkg.m_satc3);

if (rc==0) {
MY_TRACE(".....Get Sub-Assembly Type 3 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET PART COST
rc = objPODB.getSATC4(create_PO_pkg.m_satno4,
&create_PO_pkg.m_satc4);

if (rc==0) {
MY_TRACE(".....Get Sub-Assembly Type 4 Cost - Successful...\n");
MY_TRACE1 (".....Retrieved Cost=%f...\n",create_PO_pkg.m_atc);
          }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objPODB.closePODB();
MY_TRACE(".....CMMS Database Closed..\n");

//END OF GET COST DATA (ATOMIC) EVENT RESPONSE
MY_TRACE("..........\n");
//-------------------------------------------------------------------
//-------------------------------------------------------------------
//CALCULATE PURCHASE ORDER TOTALS (ATOMIC)EVENT RESPONSE
MY_TRACE(".....CALCULATE PURCHASE ORDER TOTALS (Atomic) Event
Response...\n");
//INVOKE ACTIVITY - OPEN PURCHASE ORDER FORM

rc = objPOForm.pofOpen();
if (rc !=0){
MY_TRACE(".....Open WKSheet fail...\n");
if (DebugMsg)
AfxMessageBox(".....Open worksht fail");
goto skip_pof;
//roll-back code here
          }
MY_TRACE(".....Open Purchase Order Form - Successful..\n");

//-------------------------------------------------------------------
//INVOKE ACTIVITY - CALCULATE PURCHASE ORDER TOTALS
MY_TRACE(".....Calculate Purchase Order Totals...\n");
rc = objPOForm.pofSubmit(create_PO_pkg.m_poid,
create_PO_pkg.m_vna, create_PO_pkg.m_va, create_PO_pkg.m_vpn,
create_PO_pkg.m_atno, create_PO_pkg.m_pqty, create_PO_pkg.m_atc,
&create_PO_pkg.m_ist, create_PO_pkg.m_atno2, create_PO_pkg.m_pqty2,
create_PO_pkg.m_atc2, &create_PO_pkg.m_ist2, create_PO_pkg.m_atno3,
create_PO_pkg.m_pqty3, create_PO_pkg.m_atc3, &create_PO_pkg.m_ist3,
create_PO_pkg.m_atno4, create_PO_pkg.m_pqty4, create_PO_pkg.m_atc4,
&create_PO_pkg.m_ist4, create_PO_pkg.m_etno, create_PO_pkg.m_pqty5,
create_PO_pkg.m_etc, &create_PO_pkg.m_ist5, create_PO_pkg.m_etno2,
create_PO_pkg.m_pqty6, create_PO_pkg.m_etc2, &create_PO_pkg.m_ist6,
create_PO_pkg.m_etno3, create_PO_pkg.m_pqty7, create_PO_pkg.m_etc3,
```

```
&create_PO_pkg.m_ist7, create_PO_pkg.m_etno4, reate_PO_pkg.m_pqty8,
create_PO_pkg.m_etc4, &create_PO_pkg.m_ist8, create_PO_pkg.m_atno,
create_PO_pkg.m_pqty9, create_PO_pkg.m_satc, &create_PO_pkg.m_ist9,
create_PO_pkg.m_satno2, create_PO_pkg.m_pqty10,
create_PO_pkg.m_satc2, &create_PO_pkg.m_ist10,
create_PO_pkg.m_satno3, create_PO_pkg.m_pqty11,
create_PO_pkg.m_satc3, &create_PO_pkg.m_ist11,
create_PO_pkg.m_satno4, create_PO_pkg.m_pqty12,
create_PO_pkg.m_satc4, &create_PO_pkg.m_ist12,
&create_PO_pkg.m_post, &create_PO_pkg.m_pst, &create_PO_pkg.m_fst,
&create_PO_pkg.m_tpc);

if (rc !=0) {
MY_TRACE(".....Submit worksheet fail...\n");
if (DebugMsg)
AfxMessageBox(".....Submit worksheet fail");
goto skip_pof;
// Implement roll-back code here later
}

                {
CString strtemp;
strtemp.Format(".....POST=%f   PST=%f   FST=%f   TPC=%f...\n",
create_PO_pkg.m_post, create_PO_pkg.m_pst, create_PO_pkg.m_fst,
create_PO_pkg.m_tpc);
MY_TRACE(strtemp);
                }

//-----------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE PURCHASE ORDER FORM
MY_TRACE(".....Close Purchase Order Form...\n");
rc = objPOForm.pofClose();
if (rc != 0) {
MY_TRACE(".....Close worksheet fail...\n");
if (DebugMsg)
AfxMessageBox(".....Close worksheet fail");
goto skip_pof;
// Implement roll-back code here later
                }

create_PO_pkg.m_status = "Purchase order submitted, click CONFIRM to
commit and print";

skip_pof:
break;
//END OF CALCULATE PURCHASE ORDER TOTALS (ATOMIC)EVENT
RESPONSE
//END OF SUBMIT PURCHASE ORDER (COMBINED) EVENT RESPONSE
MY_TRACE("..........\n");
//-----------------------------------------------------------------
//-----------------------------------------------------------------
//CONFIRM/PRINT (COMBINED) EVENT RESPONSE
case EID_purchase_order_confirmprint:

MY_TRACE("..Start PURCHASE ORDER CONFIRM/PRINT (Combined)
Event Response...\n");

create_PO_pkg.m_status = "Confirming and Printing order, please wait...";

// **** Phase 1 ****
// Update the vendor information into the vendor database.
// This activity will add the vendor into the database if the
// vendor was not found in the database.
// Open the Vendor database
//-----------------------------------------------------------
//UPDATE VENDOR DATA (ATOMIC) EVENT RESPONSE
MY_TRACE("....Start UPDATE VENDOR DATA (Atomic) Event
Response...\n");
//INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objVenDB.openVenDB();
if (rc == -1) {
MY_TRACE1(".....open vendor database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
            }
MY_TRACE("....Open CMMS Database - Successful...\n");
//-----------------------------------------------------------
//INVOKE ACTIVITY - CHECK IF VENDOR NAME IN VENDORS TABLE
// Is the vendor info already contained in the Vendor DB
MY_TRACE(".....Check if Vendor Name Exists in VENDORS Table...\n");
ven_rc = objVenDB.recExistVN(create_PO_pkg.m_vna, &venExisted);

// If uniqueness validation is successful, then
// create the record.
if (ven_rc == -1) {
MY_TRACE(".....rec Exist check failed...\n");
MY_TRACE1(".....ven_rc=%d...\n", ven_rc);
rc = objVenDB.closeVenDB();
MY_TRACE1(".....close vendor databasrc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
            }
//-----------------------------------------------------------
//INVOKE ACTIVITY - CREATE RECORD IN VENDORS TABLE
if (!venExisted) {
MY_TRACE(".....Create Record in VENDORS Table...\n");

// Vender does not exist, create the record.
rc = objVenDB.recCreate(create_PO_pkg.m_vna, create_PO_pkg.m_va,
 create_PO_pkg.m_vpn, create_PO_pkg.m_vno);

if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
rc = objVenDB.closeVenDB();
MY_TRACE1(".....close vendr database rc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
            }
// If record creation is successful, acknowledge
// success
MY_TRACE(".....Creation Succesful...\n");
            }
//-----------------------------------------------------------
//INVOKE ACTIVITY - UPDATE RECORD IN VENDORS TABLE
else {
MY_TRACE(".....Update Record in VENDORS Table...\n");
// Customer file exists, use the passed-in values
// to update the Customer record, as the address
// information might have changed.

rc = objVenDB.recEditVA(create_PO_pkg.m_vna, create_PO_pkg.m_va,
create_PO_pkg.m_vpn);
            }
//-----------------------------------------------------------
// INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objVenDB.closeVenDB();
MY_TRACE(".....Close CMMS Database...\n");
//END OF UPDATE VENDOR DATA (ATOMIC) EVENT RESPONSE
MY_TRACE("..........\n");
//-----------------------------------------------------------
//-----------------------------------------------------------
//SAVE PURCHASE ORDER DATA (ATOMIC) EVENT RESPONSE
MY_TRACE(".....Start SAVE PURCHASE ORDER DATA (Atomic) Event
Response...\n");
//INVOKE ACTIVITY - OPEN CMMS DATABASE
// Create the purchase entry transaction into the purchase order Database.
rc = objPODB.openPODB();
if (rc == -1) {
MY_TRACE1(".....open PO database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
            }
MY_TRACE(".....Open CMMS Database  - Successful...\n");
//-----------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the purchase entry record for ETNO
if (create_PO_pkg.m_pqty > 0) {
MY_TRACE(".....Save Record (For Equipment Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_etno, create_PO_pkg.m_pqty,
create_PO_pkg.m_ist);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
            }
}
//-----------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ETNO2
if (create_PO_pkg.m_pqty2 > 0) {
MY_TRACE(".....Save Record (For Equipment Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_etno2, create_PO_pkg.m_pqty2,
create_PO_pkg.m_ist2);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                    }
                            }
//-----------------------------------------------------------
```

```
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ETNO3
if (create_PO_pkg.m_pqty3 > 0) {
MY_TRACE(".....Save Record (For Equipment Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_etno3, create_PO_pkg.m_pqty3,
 create_PO_pkg.m_ist3);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ETNO4
if (create_PO_pkg.m_pqty4 > 0) {
MY_TRACE(".....Save Record (For Equipment Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_etno4, create_PO_pkg.m_pqty4,
create_PO_pkg.m_ist4);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the purchase entry record for ATNO
if (create_PO_pkg.m_pqty5 > 0) {
MY_TRACE(".....Save Record (For Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_atno, create_PO_pkg.m_pqty5,
create_PO_pkg.m_ist5);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ATNO2
if (create_PO_pkg.m_pqty6 > 0) {
MY_TRACE(".....Save Record (For Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_atno2, create_PO_pkg.m_pqty6,
create_PO_pkg.m_ist6);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ATNO3
if (create_PO_pkg.m_pqty7 > 0) {
MY_TRACE(".....Save Record (For Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_atno3, create_PO_pkg.m_pqty7,
create_PO_pkg.m_ist7);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for ATNO4
if (create_PO_pkg.m_pqty8 > 0) {
MY_TRACE(".....Save Record (For Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_etno4, create_PO_pkg.m_pqty8,
create_PO_pkg.m_ist8);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the purchase entry record for SATNO
if (create_PO_pkg.m_pqty9 > 0) {
MY_TRACE(".....Save Record (For Sub-Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_satno, create_PO_pkg.m_pqty9,
create_PO_pkg.m_ist9);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for SATNO2
if (create_PO_pkg.m_pqty10 > 0) {
MY_TRACE(".....Save Record (For Sub-Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_satno2,
create_PO_pkg.m_pqty10, create_PO_pkg.m_ist10);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for SATNO3
if (create_PO_pkg.m_pqty11 > 0) {
MY_TRACE(".....Save Record (For Sub-Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_satno3,
create_PO_pkg.m_pqty11, create_PO_pkg.m_ist11);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE PURCHASE ORDER RECORD IN
PURCHASE_ORDERS TABLE
// Create the sale entry record for SATNO4
if (create_PO_pkg.m_pqty12 > 0) {
MY_TRACE(".....Save Record (For Sub-Assembly Purchase) in
PURCHASE_ORDERS Table...\n");
rc = objPODB.recCreate(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_vpn, create_PO_pkg.m_satno4,
create_PO_pkg.m_pqty12, create_PO_pkg.m_ist12);
if (rc == -1) {
// If record creation has failed, acknowledge
// failure
MY_TRACE1(".....Creation Failed rc=%d...\n", rc);
// Implement roll-back activity here.
goto skip3;
                                        }
                                }

// If record creation is successful, acknowledge
```

```
// success
//MY_TRACE(".....Creation Successful...\n");
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
skip3:
rc = objPODB.closePODB();
MY_TRACE(".....Close CMMS Database...\n");
//END OF SAVE PURCHASE ORDER DATA (ATOMIC) EVENT RESPONSE
MY_TRACE(".....\n");
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
//PRINT PURCHASE ORDER FORM (ATOMIC) EVENT RESPONSE
MY_TRACE(".....Start PRINT PURCHASE ORDER (Atomic) Event
Response...\n");
//INVOKE ACTIVITY - OPEN PURCHASE ORDER FORM
// Print the PO form by invoking the PO Form
// component.
// To perform the print, need to transfer the information
// to the hard copy form. This is done by calling the
// DLL procedure and passing in the list of parameters.
// Open the PO Form Handler to calculate the
// IST for each item and to calculate the POST,
// and TPC

MY_TRACE(".....Open Purchase Order Form...\n");
rc = objPOForm.pofOpen();
if (rc != 0) {
MY_TRACE(".....Open worksheet fail...\n");
goto skip4;
// Implement roll-back code here later
                                }
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - CALCULATE PURCHASE ORDER TOTALS
MY_TRACE(".....Submit Data and Calculate Totals...\n");
rc = objPOForm.pofSubmit(create_PO_pkg.m_poid, create_PO_pkg.m_vna,
create_PO_pkg.m_va, create_PO_pkg.m_vpn, create_PO_pkg.m_atno,
create_PO_pkg.m_pqty, create_PO_pkg.m_atc, &create_PO_pkg.m_ist,
create_PO_pkg.m_atno2, create_PO_pkg.m_pqty2, create_PO_pkg.m_atc2,
&create_PO_pkg.m_ist2, create_PO_pkg.m_atno3, create_PO_pkg.m_pqty3,
create_PO_pkg.m_atc3, &create_PO_pkg.m_ist3, create_PO_pkg.m_atno4,
create_PO_pkg.m_pqty4, create_PO_pkg.m_atc4, &create_PO_pkg.m_ist4,
create_PO_pkg.m_atno, create_PO_pkg.m_pqty5,
create_PO_pkg.m_etc, &create_PO_pkg.m_ist5, create_PO_pkg.m_etno2,
create_PO_pkg.m_pqty6, create_PO_pkg.m_etc2, &create_PO_pkg.m_ist6,
create_PO_pkg.m_etno3, create_PO_pkg.m_pqty7,
create_PO_pkg.m_etc3, &create_PO_pkg.m_ist7, create_PO_pkg.m_etno4,
create_PO_pkg.m_pqty8, create_PO_pkg.m_etc4, &create_PO_pkg.m_ist8,
create_PO_pkg.m_atno, create_PO_pkg.m_pqty9, create_PO_pkg.m_satc,
&create_PO_pkg.m_ist9, create_PO_pkg.m_satno2,
create_PO_pkg.m_pqty10, create_PO_pkg.m_satc2,
&create_PO_pkg.m_ist10, create_PO_pkg.m_satno3,
create_PO_pkg.m_pqty11,create_PO_pkg.m_satc3,
&create_PO_pkg.m_ist11,
create_PO_pkg.m_satno4, create_PO_pkg.m_pqty12,
create_PO_pkg.m_satc4, &create_PO_pkg.m_ist12,
&create_PO_pkg.m_post, &create_PO_pkg.m_pst,
&create_PO_pkg.m_fst, &create_PO_pkg.m_tpc);
if (rc !=0) {
MY_TRACE(".....Confirm worksheet fail...\n");
goto skip_pof;
// Implement roll-back code here later
                                }
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - PRINT PURCHASE ORDER FORM
if (eid == EID_purchase_order_confirmprint) {
MY_TRACE(".....Print Purchase Order Form...\n");
rc = objPOForm.pofPrint();
if (rc != 0) {
MY_TRACE(".....Print worksheet fail...\n");
goto skip4;
// Implement roll-back code here later
                                        }
                                }
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE PURCHASE ORDER FORM
MY_TRACE(".....Close Purchase Order Form...\n");
rc = objPOForm.pofClose();
if (rc != 0) {
MY_TRACE(".....Close worksheet fail...\n");
goto skip4;
// Implement roll-back code here later
                                }

skip4:
break;
//END OF PRINT PURCHASE ORDER FORM (ATOMIC) EVENT
RESPONSE
//END OF CONFIRM/PRINT (COMBINED) EVENT RESPONSE
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
//GET EQUIPMENT INFORMATION (ATOMIC) EVENT RESPONSE
case EID_get_eqpt_info:
MY_TRACE("....Start GET EQUIPMENT INFORMATION (Atomic) Event
Response...\n");
if (DebugMsg)
AfxMessageBox("....Start GET EQUIPMENT INFORMATION (Atomic) Event
Response");

// Retrieve the Eqpt information from the database (Op Eqpt, Eqpt Type
//and Location Tables.
// Open the CMMS database
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - CHECK IF SERIAL NUMBER EXISTS IN
OPERATIONAL_EQUIPMENT TABLE
// Check for uniqueness of Eqpt Serial No. in the CMMS DB
MY_TRACE(".....Check if Serial Number Exists in
OPERATIONAL_EQUIPMENT Table...\n");
sn_rc = objDB.recExistSN(create_WO_pkg.m_sn, &snExisted);

// If serial No. not in DB, try again - made a mistake
if (sn_rc == -1) {
MY_TRACE(".....no such serial no. exists - try again...\n");
MY_TRACE1(".....sn_rc=%d...\n", sn_rc);
rc = objDB.closeDB();
MY_TRACE1(".....close CMMS database rc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return -1;
}
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - GET EQUIPMENT INFORMATION
if (snExisted) {
// serial no. exist, retrieve the eqpt info.
// But first allocate the buffer for the info.

SysFreeString(m_bstrETNA);
m_bstrETNA = create_WO_pkg.m_etna.AllocSysString();
SysFreeString(m_bstrDMP);
m_bstrDMP = create_WO_pkg.m_dmp.AllocSysString();
SysFreeString(m_bstrFF);
m_bstrFF = create_WO_pkg.m_ff.AllocSysString();
SysFreeString(m_bstrMT);
m_bstrMT = create_WO_pkg.m_mt.AllocSysString();
SysFreeString(m_bstrLOCATION);
m_bstrLOCATION = create_WO_pkg.m_location.AllocSysString();
rc = objDB.getEqptInfo(create_WO_pkg.m_sn, &m_bstrETNA,
&m_bstrDMP,
&m_bstrFF,&m_bstrMT, &m_bstrLOCATION);

create_WO_pkg.m_etna = m_bstrETNA;
create_WO_pkg.m_dmp = m_bstrDMP;
create_WO_pkg.m_ff = m_bstrFF;
create_WO_pkg.m_mt = m_bstrMT;
create_WO_pkg.m_location = m_bstrLOCATION;

if (DebugMsg) {
strMsgText.Format("DMP=%s ETNA=%s LOCATION=%s",
create_WO_pkg.m_dmp, create_WO_pkg.m_etna,
create_WO_pkg.m_location);
AfxMessageBox(strMsgText);
        }
if (rc == -1) {
// If no serial no.existed, acknowledge failure
MY_TRACE1(".....no serial no. exists rc=%d...\n", rc);
rc = objDB.closeDB();
MY_TRACE1(".....close CMMS database rc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return -1;
        }
// If record creation is successful, acknowledge
// success
MY_TRACE(".....Get Equipment Information - Successful...\n");
        }
//-----------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
MY_TRACE(".....\n");
break;

//END OF GET EQUIPMENT INFORMATION (ATOMIC) EVENT RESPONSE
//-----------------------------------------------------------------------
//-----------------------------------------------------------------------
//GET EMPLOYEE INFORMATION (ATOMIC) EVENT RESPONSE
```

```
case EID_get_emp_info:

//there has to be a SQL query in the VB db handler to get the right info

MY_TRACE("....Start GET EMPLOYEE INFORMATION (Atomic) Event
Response...\n");
if (DebugMsg)
AfxMessageBox("....Start Get Emp Info event response");

// **** Autofill Vendor Information ****
// Fill in SD, TCD, WAD, TNA, TL, IB
//Retrieve the employee information from the database (Employee Table)
// Open the CMMS database
//-------------------------------------------------------------------

//INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....open CMMS database successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CHECK IF TRADE NAME IN CRAFTS_TRADES
TABLE
// Check that TNA exist (no typos made) in the CMMS DB
MY_TRACE(".....Check If Trade Name Exists in CRAFTS_TRADES
Table...\n");
tna_rc = objDB.recExistTNA(create_WO_pkg.m_tna, &tnaExisted);

// If TNA not in DB, try again - made a mistake
if (tna_rc == -1) {
MY_TRACE(".....TNA don't exist - try again...\n");
MY_TRACE1(".....tna_rc=%d...\n", tna_rc);
rc = objDB.closeDB();
MY_TRACE(".....close CMMS Database...\n");
MY_TRACE("....exit coordinator...\n");
return -1;
}
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CHECK IF TRADE LEVEL IN CRAFTS  TRADES
TABLE
if (tnaExisted) {
MY_TRACE(".....Check If Trade Level Exists in CRAFTS_TRADES
Table...\n");
tl_rc = objDB.recExistTL(create_WO_pkg.m_tl, &tlExisted);

// If TL not in DB, try again - made a mistake
if (tl_rc == -1) {
MY_TRACE(".....TL don't exist - try again...\n");
MY_TRACE1(".....tl_rc=%d...\n", tl_rc);
rc = objDB.closeDB();
MY_TRACE1(".....close CMMS database rc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return -1;
}
}
//-------------------------------------------------------------------
//INVOKE ACTIVITY - GET EMPLOYEE NAMES
if (tlExisted) {
MY_TRACE(".....Get Employee Information...\n");

// TNA and TL exist, retrieve the emp info.
// But first allocate the buffer for the info.

SysFreeString(m_bstrEN);
m_bstrEN = create_WO_pkg.m_combo_en.AllocSysString();

rc = objDB.getEmpInfo(create_WO_pkg.m_tna,create_WO_pkg.m_tl,
create_WO_pkg.m_location, &m_bstrEN);

create_WO_pkg.m_combo_en = m_bstrEN;

if (DebugMsg) {
strMsgText.Format("EN=%s", create_WO_pkg.m_combo_en);
AfxMessageBox(strMsgText);
        }

if (rc == -1) {
// If no TNA and TL don't exist, acknowledge failure
MY_TRACE1(".....no TNA/TL exists rc=%d...\n", rc);
rc = objDB.closeDB();
MY_TRACE1(".....close CMMS database rc = %d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return -1;
        }
// If record creation is successful, acknowledge
// success
//
MY_TRACE(".....Get emp info Successful...\n");
        }

//-------------------------------------------------------------------

//INVOKE ACTIVITY - CLOSE CMMS DATABASE
// Close the database
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
MY_TRACE("-------------------------------------------------------------\n");
break;

//END OF GET EMPLOYEE INFORMATION (ATOMIC) EVENT RESPONSE
//-------------------------------------------------------------------
//-------------------------------------------------------------------
//SAVE WORK ORDER (ATOMIC) EVENT RESPONSE
case EID_save:
//Save the WO in the database
MY_TRACE("....Start SAVE WORK ORDER Event Response...\n");
//-------------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - SAVE WORK ORDER DATA
save_rc = objDB.SaveRec(create_WO_pkg.m_won, create_WO_pkg.m_sn,
create_WO_pkg.m_id, create_WO_pkg.m_ib, create_WO_pkg.m_sd,
create_WO_pkg.m_tcd, create_WO_pkg.m_combo_en,
create_WO_pkg.m_combo_en2, create_WO_pkg.m_combo_en3,
create_WO_pkg.m_combo_en4, create_WO_pkg.m_wad,
create_WO_pkg.m_tna, create_WO_pkg.m_tl);


MY_TRACE(".....SAVE WORK ORDER DATA - Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;
//-------------------------------------------------------------------
//-------------------------------------------------------------------
//UPDATE WORK ORDER (ATOMIC) EVENT RESPONSE
case EID_Update:
//Update the WO in the database
MY_TRACE("....Start UPDATE WORK ORDER Event Response...\n");
//-------------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
        if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
        }
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - UPDATE WORK ORDER DATA
update_rc = objDB.recUpdateWO(create_WO_pkg.m_won,
create_WO_pkg.m_sn, create_WO_pkg.m_id, create_WO_pkg.m_ib,
create_WO_pkg.m_sd, create_WO_pkg.m_tcd,
create_WO_pkg.m_combo_en, create_WO_pkg.m_combo_en2,
create_WO_pkg.m_combo_en3, create_WO_pkg.m_combo_en4,
create_WO_pkg.m_wad, create_WO_pkg.m_tna, create_WO_pkg.m_tl,
create_WO_pkg.m_cd, create_WO_pkg.m_ds);

update2_rc = objDB.recUpdateWO2(create_WO_pkg.m_sn,
create_WO_pkg.m_dmp, create_WO_pkg.m_ff, create_WO_pkg.m_mt);


MY_TRACE(".....UPDATE WORK ORDER DATA - Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;

//-------------------------------------------------------------------
//-------------------------------------------------------------------

//RETRIEVE WORK ORDER (ATOMIC) EVENT RESPONSE
case EID_edit:
//Retrieve the WO in the database
MY_TRACE("....Start Retrieve WORK ORDER Event Response...\n");
//-------------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
```

```
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-----------------------------------------------------------------
//INVOKE ACTIVITY - RETRIEVE WORK ORDER DATA

{
BSTR bsn = create_WO_pkg.m_sn.AllocSysString();
BSTR bib = create_WO_pkg.m_ib.AllocSysString();
 BSTR bcombo_en = create_WO_pkg.m_combo_en.AllocSysString();
 BSTR bcombo_en2 = create_WO_pkg.m_combo_en2.AllocSysString();
 BSTR bcombo_en3 = create_WO_pkg.m_combo_en3.AllocSysString();
BSTR bcombo_en4 = create_WO_pkg.m_combo_en4.AllocSysString();
BSTR bdmp  = create_WO_pkg.m_dmp.AllocSysString();
BSTR bff = create_WO_pkg.m_ff.AllocSysString();
BSTR blocation = create_WO_pkg.m_location.AllocSysString();
BSTR bmt = create_WO_pkg.m_mt.AllocSysString();
BSTR betna = create_WO_pkg.m_etna.AllocSysString();
BSTR bwad = create_WO_pkg.m_wad.AllocSysString();
BSTR btna = create_WO_pkg.m_tna.AllocSysString();
BSTR btl = create_WO_pkg.m_tl.AllocSysString();

edit_rc = objDB.getWOInfo(create_WO_pkg.m_won, &bsn,
&(create_WO_pkg.m_id.m_dt), &bib,&(create_WO_pkg.m_sd.m_dt) ,
&(create_WO_pkg.m_tcd.m_dt), &bcombo_en,
&bcombo_en2,&bcombo_en3, &bcombo_en4, &bdmp, &bff, &blocation,
&bmt, &betna, &bwad, &btna, &btl, &(create_WO_pkg.m_cd.m_dt),
&(create_WO_pkg.m_ds.m_dt)
                              );

create_WO_pkg.m_sn = bsn;
create_WO_pkg.m_ib = bib;
create_WO_pkg.m_combo_en = bcombo_en;
create_WO_pkg.m_combo_en2 = bcombo_en2;
create_WO_pkg.m_combo_en3 = bcombo_en3;
create_WO_pkg.m_combo_en4 = bcombo_en4;
create_WO_pkg.m_dmp = bdmp;
create_WO_pkg.m_ff  = bff;
create_WO_pkg.m_location = blocation;
create_WO_pkg.m_mt  = bmt;
create_WO_pkg.m_etna = betna;
create_WO_pkg.m_wad = bwad;
create_WO_pkg.m_tna = btna;
create_WO_pkg.m_tl = btl;

SysFreeString(bsn);
SysFreeString(bib);
SysFreeString(bcombo_en);
SysFreeString(bcombo_en2);
SysFreeString(bcombo_en3);
SysFreeString(bcombo_en4);
SysFreeString(bdmp);
SysFreeString(bff);
SysFreeString(blocation);
SysFreeString(betna);
SysFreeString(bwad);
SysFreeString(btna);
SysFreeString(btl);

          }
MY_TRACE(".....RETRIEVE WORK ORDER DATA - Successful...\n");
//-----------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;

//-----------------------------------------------------------------
//-----------------------------------------------------------------
//GET WORK ORDER STATUS(ATOMIC) EVENT RESPONSE
case EID_wo_status:
//Get the WO Status Info from the database
MY_TRACE("....Start Get WORK ORDER Status Event Response...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");

//-----------------------------------------------------------------
// INVOKE ACTIVITY - UPDATE WO STATUS - WO STARTED

update_st2 = objDB.Update_Started2(WO_Status_pkg.m_started_ques);
MY_TRACE(".....UPDATE WO STATUS - IF STARTED - Successful...\n");
//-----------------------------------------------------------------
//INVOKE ACTIVITY - UPDATE WO STATUS - STARTED LATE

update_st = objDB.Update_Started(WO_Status_pkg.m_late_start_ques);
MY_TRACE(".....UPDATE WO STATUS - IF STARTED LATE -
Successful...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - UPDATE WO STATUS - WO COMPLETED
update_co2 = objDB.Update_Started4(WO_Status_pkg.m_comp_ques);
MY_TRACE(".....UPDATE WO STATUS - IF COMPLETED -
Successful...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - UPDATE WO STATUS - WO COMPLETED LATE
update_co = objDB.Update_Started3(WO_Status_pkg.m_late_comp_ques);
MY_TRACE(".....UPDATE WO STATUS - IF COMPLETED LATE -
Successful...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - UPDATE COMPLETION STATUS - STARTED
update_cs1 = objDB.cs1(WO_Status_pkg.m_cs);
//-----------------------------------------------------------------
// INVOKE ACTIVITY - UPDATE COMPLETION STATUS - COMPLETED
update_cs2 = objDB.cs2(WO_Status_pkg.m_cs);
//-----------------------------------------------------------------
// INVOKE ACTIVITY - GET WO STATUS
{
BSTR blate_start_ques =
WO_Status_pkg.m_late_start_ques.AllocSysString();
BSTR bstarted_ques = WO_Status_pkg.m_started_ques.AllocSysString();
BSTR blate_comp_ques =
WO_Status_pkg.m_late_comp_ques.AllocSysString();
BSTR bcomp_ques = WO_Status_pkg.m_comp_ques.AllocSysString();
BSTR bcs = WO_Status_pkg.m_cs.AllocSysString();

status_rc = objDB.getWOStatus(WO_Status_pkg.m_wono,
&(WO_Status_pkg.m_sdate.m_dt), &(WO_Status_pkg.m_tcdate.m_dt),
&(WO_Status_pkg.m_idate.m_dt), &blate_start_ques, &bstarted_ques,
&blate_comp_ques,  &bcomp_ques, &bcs );

WO_Status_pkg.m_late_start_ques = blate_start_ques;
WO_Status_pkg.m_started_ques = bstarted_ques;
WO_Status_pkg.m_late_comp_ques =  blate_comp_ques;
WO_Status_pkg.m_comp_ques = bcomp_ques;
WO_Status_pkg.m_cs = bcs;

SysFreeString(blate_start_ques);
SysFreeString(bstarted_ques);
SysFreeString(blate_comp_ques);
SysFreeString(bcomp_ques);
SysFreeString(bcs);

          }
MY_TRACE(".....GET WORK ORDER STATUS- Successful...\n");
//-----------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;

//-----------------------------------------------------------------
//-----------------------------------------------------------------
//GET NEXT WEEKLY MAINTENANCE - SERIAL NO.S & DATES(ATOMIC)
EVENT RESPONSE
case EID_get_wk_nm:
//Get the WO Status Info from the database
MY_TRACE("....Start Next Weekly Maintenance Status Event
Response...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-----------------------------------------------------------------
// INVOKE ACTIVITY - GET NEXT WEEKLY MAINT INFO

{
BSTR bsnw  = NM_pkg.m_snw.AllocSysString();
nm_rc = objDB.getWKnm(NM_pkg.m_etno,
NM_pkg.m_location,
&(NM_pkg.m_nwm.m_dt),
&bsnw
  );
NM_pkg.m_snw =          bsnw;
SysFreeString(bsnw);
}
MY_TRACE(".....GET Next Weekly Maint STATUS- Successful...\n");
//-----------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;
```

```
//-------------------------------------------------------------------
//-------------------------------------------------------------------
//GET NEXT MONTHLY MAINTENANCE - SERIAL NO.S & DATES(ATOMIC)
EVENT RESPONSE
case EID_get_mon_nm:
//Get the WO Status Info from the database
MY_TRACE("....Start Next Monthly Maintenance Status Event
Response...\n");
//-------------------------------------------------------------------
// INVOKE ACTIVITY - OPEN CMMS DATABASE
rc = objDB.openDB();
if (rc == -1) {
MY_TRACE1(".....open CMMS database fail rc=%d...\n", rc);
MY_TRACE("....exit coordinator...\n");
return 0;
}
MY_TRACE(".....Open CMMS Database - Successful...\n");
//-------------------------------------------------------------------
// INVOKE ACTIVITY - GET NEXT MONTHLY MAINT INFO
{
BSTR bsnm  = NM_pkg.m_snm.AllocSysString();
nm_rc = objDB.getMONnm(NM_pkg.m_etno,
NM_pkg.m_location,
&(NM_pkg.m_nmm.m_dt),
&bsnm
 );
NM_pkg.m_snm =          bsnm;
SysFreeString(bsnm);
}
MY_TRACE(".....GET Next Monthly Maint STATUS- Successful...\n");
//-------------------------------------------------------------------
//INVOKE ACTIVITY - CLOSE CMMS DATABASE
rc = objDB.closeDB();
MY_TRACE(".....Close CMMS Database...\n");
break;

case EID_debug_invDB_trace_enable:
objInvDB.setInvDBTrace(TRUE);
break;

case EID_debug_invDB_trace_disable:
objInvDB.setInvDBTrace(FALSE);
break;

case EID_debug_PODB_trace_enable:
objPODB.setPODBTrace(TRUE);
break;

case EID_debug_PODB_trace_disable:
objPODB.setPODBTrace(FALSE);
break;

case EID_debug_VenDB_trace_enable:
objVenDB.setVenDBTrace(TRUE);
break;

case EID_debug_VenDB_trace_disable:
objVenDB.setVenDBTrace(FALSE);
break;

case EID_info_incr_coord_instance:
coordInstance++;
break;

case EID_info_decr_coord_instance:
coordInstance--;
break;

case EID_info_get_coord_instance:
instance_image = coordInstance;
break;


case EID_debug_coord_tracemsg_enable:
DebugMsg = TRUE;
break;

case EID_debug_coord_tracemsg_disable:
DebugMsg = FALSE;
break;

default:
TRACE("....Unknown event id : %d...\n", eid);
break;

        }
MY_TRACE("....Exit Coordinator...\n");
return 0;
}
return 0;
}
// **** Coordinator Main ****
// Implement the coordinator
//

int invoke_coord(int eid)
{
short rc = 0;
HANDLE hMutex;
DWORD dwWaitResult;
CString strMsgText;
if (DebugMsg) {
strMsgText.Format("Inside Coordinator with event id = %d...\n", eid);
AfxMessageBox(strMsgText);
        }

//
MY_TRACE1("....Inside Coordinator with event id = %d...\n", eid);

// If initialization failed, do not continue
if (init_fail) {
MY_TRACE("....Exit Coordinator due to init fail...\n");
if (DebugMsg)
AfxMessageBox("....Exit Coordinator due to init fail");
return -1;
        }

//
// Setup the Coordinator Control State data
// Create a mutex with no initial owner.
hMutex = CreateMutex(NULL,  // no security attributes
FALSE,
// initially not owned
 "CoordMutex");  // name of mutex

if (hMutex == NULL)
{
// Check for error.
MY_TRACE("....Cannot get a handle on the Coordinator Mutex...\n");
MY_TRACE("....Exit Coordinator...\n");
if (DebugMsg) {
strMsgText.Format("Cannot get a handle on the Coordinator Mutex. Exit
Coordinator");
AfxMessageBox(strMsgText);
        }

return -1;
}

// Request ownership of mutex.

dwWaitResult = WaitForSingleObject(
hMutex,  // handle of mutex
2000L);  // two-second time-out interval

switch (dwWaitResult)
{
// The thread got mutex ownership.
case WAIT_OBJECT_0:

// Set the coordinator to busy state and
// the existing EID
coordBusy = TRUE;
curEID = eid;

// Execute the response
rc = handle_event(eid);
// Reset the coordinator state
coordBusy = FALSE;
curEID = 0;

// Release ownership of the mutex object.
 if (! ReleaseMutex(hMutex)) {
// Deal with error.
MY_TRACE("....Release Mutex error!...\n");
if (DebugMsg)
AfxMessageBox("....Release Mutex error!");
rc = -1;
        }

break;


// Cannot get mutex ownership due to time-out.
case WAIT_TIMEOUT:
MY_TRACE("....Request Mutex Timeout...\n");
if (DebugMsg)
AfxMessageBox("....Request Mutex Timeout");
rc = -2;
break;
```

```
// Got ownership of the abandoned mutex object.
case WAIT_ABANDONED:
MY_TRACE("....Request Mutex Abandoned...\n");
if (DebugMsg)
AfxMessageBox("....Request Mutex Abandoned");
rc = -3;
break;
   }
return rc;
}
```

Coord.h

```
// Coord.h : main header file for the COORD DLL
//

// This is the header file coord.h
// The purpose of this file is to declare all the included files needed
// by the coordinator (implemented in coordinator.cpp).
//
// Ensure to include header files that declare the class for the
// dialog box. This file will make use of individual dialog box class
// header files and so we need to modify this one every time a new class
// is generated.

#ifndef _COORD_H_
#define _COORD_H_

#include "AddInvItem.h"
#include "DelInvItem.h"
#include "CreatePO.h"
#include "CreateWO.h"
#include "WO_Statu.h"
#include "invdb_handler.h"
#include "vendb_Handler.h"
#include "podb_handler.h"
#include "poformprint.h"
#include "DB_Handler.h"
//
// Declare the event ID
//
typedef int t_event_id;
#define EID_config_InvDB                        1
#define EID_get_item_info                       8
#define EID_NM_Clear                            9
#define EID_PO_Clear                            15
#define EID_WO_Clear                            16
#define EID_Update                              17
#define EID_wo_status                           18
#define EID_WO_Status_Clear                     19
#define EID_debug_invDB_trace_enable            20
#define EID_debug_invDB_trace_disable           21
#define EID_debug_PODB_trace_enable             22
#define EID_debug_PODB_trace_disable            23
#define EID_debug_VenDB_trace_enable            24
#define EID_debug_VenDB_trace_disable           25
#define EID_info_incr_coord_instance            28
#define EID_info_decr_coord_instance            29
#define EID_info_get_coord_instance             30
#define EID_debug_coord_tracemsg_enable         31
#define EID_debug_coord_tracemsg_disable        32
#define EID_add_inv_item                        33
#define EID_del_inv_item                        34
#define EID_purchase_order_submit               35
#define EID_purchase_order_confirm              36
#define EID_purchase_order_confirmprint         37
#define EID_get_eqpt_info                       38
#define EID_get_emp_info                        39
#define EID_save                                40
#define EID_get_emp2_info                       41
#define EID_get_emp3_info                       42
#define EID_get_emp4_info                       43
#define EID_edit                                44
#define EID_get_wk_nm                           45
#define EID_get_mon_nm                          46

//
// Allocate the system variables used by the coordinator.
//
extern CAddInvItem      a_inv_i_pkg;                    // Add
Inventory Item Dialog box instance and data
extern CDelInvItem d_inv_i_pkg;
extern CCreatePO create_PO_pkg;
extern CCreateWO create_WO_pkg;
extern CWO_Statu WO_Status_pkg;
extern NM NM_pkg;
extern int instance_image;
extern BOOL DebugMsg;

//
// Declare the function prototype for the coordinator
// Note that the option is to implement the coordinator as
// 1) an object and add a class for it, or
// s) as a simple procedure.
// Here, we choose the second approach : simple procedure.
// (However, it may make sense to declare it as an object such that there
// is a consistent way of invoking the construction and destruction of the
// coordinator object instance, as well as a uniform way of invoking its
// methods)
//
extern void init_coord();
extern void exit_coord();
extern int invoke_coord(int eid);


#endif // !defined(_COORD_H_INCLUDED_)
```

Coordwrap.cpp

```
#include "stdafx.h"

#include "CoordWrap.h"
#include "Coord.h"
HWND g_hwndTrace =  NULL;
UINT g_uiTraceMessage =
::RegisterWindowMessage("DebugTraceMessage156");
// CoordWrap.cpp : Wrapper for the Main coordinator code.
//
// This wrapper code will implement all the externally callable
// functions that interface with different implementations of
// the User Interface.
//
// The approach is to provide a DLL function for each possible
// event that can be triggered by the UI commands.
// Note that the UI implementation is responsible to prompt the
// user for all the necessary information.
//
// When the function of the wrapper is invoked, the work it does
// is to use the information passed in from the UI, initialize
// the internal data structures and eventually call the
// "invoke_coordinator" routine that manage all the responses.
//
// Note that since there can be simultaneously different kinds
// of UI initiating requests which are implemented as different
// applications (but invoking the same coordinator DLL), there
// will be multiple threads. This can be resolved by either
// serializing the incoming requests, or managing through
// data. The chosen method here is to use data. For simplicity,
// the different instances of the coordinator wrapper will check
// for the availability of the coordinator by checking the
// "Control Data" (One can view this as the state of the
// system.


// Initialize the DLL, register the classes etc

extern "C" __declspec(dllexport) int Req_add_inventory_item(
Cstring      m_mptd,
Cstring      m_mpn,
CString      m_sn,
CString      m_atno,
CString      m_etno,
CString      m_satno,
short        m_qr,
CString      m_nur,
short        m_iqty,
CString      m_isl,
short        m_msl
                                        )

{
int rc = 0;

a_inv_i_pkg.m_mptd                      =m_mptd     ;
a_inv_i_pkg.m_mpn                       =m_mpn      ;
a_inv_i_pkg.m_sn                        =m_sn       ;
a_inv_i_pkg.m_atno                      =m_atno     ;
a_inv_i_pkg.m_etno                      =m_etno     ;
a_inv_i_pkg.m_satno                     =m_satno;
a_inv_i_pkg.m_qr                        =m_qr       ;
a_inv_i_pkg.m_nur                       =m_nur      ;
a_inv_i_pkg.m_iqty                      =m_iqty     ;
a_inv_i_pkg.m_isl                       =m_isl      ;
a_inv_i_pkg.m_msl                       =m_msl      ;


rc = invoke_coord(EID_add_inv_item);
return rc;
}

extern "C" __declspec(dllexport) int Req_create_wo(int wo_select,
CString                         m_won,
```

```
CString             *m_sn,
COleDateTime        *m_id,
CString             *m_ib,
COleDateTime        *m_sd,
COleDateTime        *m_tcd,
CString             *m_combo_en,
CString             *m_combo_en2,
CString             *m_combo_en3,
CString             *m_combo_en4,
CString             *m_dmp,
CString             *m_ff,
CString             *m_location,
CString             *m_mt,
CString             *m_etna,
CString             *m_wad,
Cstring             &m_status,
CString             *m_tna,
CString             *m_tl,
COleDateTime        *m_cd,
COleDateTime        *m_ds
            )
{
int rc = 0;
CString strMsgText;

if (DebugMsg) {
strMsgText.Format("Running Submit DLL for SN=%s IB=%s", m_sn, m_ib);
AfxMessageBox(strMsgText);
}
create_WO_pkg.m_won             =m_won;
create_WO_pkg.m_sn              ="m_sn;
create_WO_pkg.m_id              ="m_id;
create_WO_pkg.m_ib              ="m_ib;
create_WO_pkg.m_sd              ="m_sd;
create_WO_pkg.m_tcd             ="m_tcd;
create_WO_pkg.m_combo_en        ="m_combo_en;
create_WO_pkg.m_combo_en2       ="m_combo_en2;
create_WO_pkg.m_combo_en3       ="m_combo_en3;
create_WO_pkg.m_combo_en4       ="m_combo_en4;
create_WO_pkg.m_dmp             ="m_dmp;
create_WO_pkg.m_ff              ="m_ff;
create_WO_pkg.m_location        ="m_location;
create_WO_pkg.m_mt              ="m_mt;
create_WO_pkg.m_etna            ="m_etna;
create_WO_pkg.m_wad             ="m_wad;
create_WO_pkg.m_tna             = *m_tna;
create_WO_pkg.m_tl              = *m_tl;
create_WO_pkg.m_cd              ="m_cd;
create_WO_pkg.m_ds              ="m_ds;


switch (wo_select) {

case C_EQ_INFO:
rc = invoke_coord(EID_get_eqpt_info);
break;

case C_EMP_INFO:
rc = invoke_coord(EID_get_emp_info);
break;

case C_WOEDIT:
rc = invoke_coord(EID_edit);
break;

case C_SAVE:
rc = invoke_coord(EID_save);
break;

case C_WOUPDATE:
rc = invoke_coord(EID_Update);
break;


default:
return -1;
}
if (DebugMsg) {
strMsgText.Format("..Coordinator result = %d", rc);
AfxMessageBox(strMsgText);
}

if (rc != 0) {
return rc;
}

// Update data with the retrieved/calculated value
*m_sn =             create_WO_pkg.m_sn;
*m_id =             create_WO_pkg.m_id;
*m_ib =             create_WO_pkg.m_ib;
*m_sd =             create_WO_pkg.m_sd;
*m_tcd =            create_WO_pkg.m_tcd;
*m_combo_en =       create_WO_pkg.m_combo_en;
*m_combo_en2 =      create_WO_pkg.m_combo_en2;
*m_combo_en3 =      create_WO_pkg.m_combo_en3;
*m_combo_en4 =      create_WO_pkg.m_combo_en4;
*m_dmp =            create_WO_pkg.m_dmp;
*m_ff =             create_WO_pkg.m_ff;
*m_location =       create_WO_pkg.m_location;
*m_mt =             create_WO_pkg.m_mt;
*m_etna =           create_WO_pkg.m_etna;
*m_wad =            create_WO_pkg.m_wad;
*m_tna =            create_WO_pkg.m_tna;
*m_tl =             create_WO_pkg.m_tl;
*m_cd =             create_WO_pkg.m_cd;
*m_ds =             create_WO_pkg.m_ds;


//if (DebugMsg) {
//strMsgText.Format("Completed Process DLL for VNA=%s VPN=%s
//ETNO=%s PQTY=%d, ETC=%f, IST=%f",
// m_vna, m_vpn, m_etno, m_pqty, *m_etc, *m_ist);
//AfxMessageBox(strMsgText);
//              }

 return rc;
}

extern "C" __declspec(dllexport) int get_WO_Status(
CString             m_wono,
CString             *m_cs,
COleDateTime        *m_sdate,
COleDateTime        *m_tcdate,
COleDateTime        *m_tdate,
CString             *m_comp_ques,
CString             *m_late_start_ques,
CString             *m_late_comp_ques,
CString             *m_started_ques,
COleDateTime        *m_idate

            )

;
int rc = 0;
/*          CString strMsgText;

if (DebugMsg) {
strMsgText.Format("Running Submit DLL for SN=%s IB=%s", m_sn, m_ib);
AfxMessageBox(strMsgText);
            }*/

WO_Status_pkg.m_wono =          m_wono;
WO_Status_pkg.m_cs =            *m_cs;
WO_Status_pkg.m_sdate =         *m_sdate;
WO_Status_pkg.m_tcdate =        *m_tcdate;
WO_Status_pkg.m_tdate =         *m_tdate;
WO_Status_pkg.m_comp_ques =     *m_comp_ques;
WO_Status_pkg.m_late_start_ques = *m_late_start_ques;
WO_Status_pkg.m_late_comp_ques = *m_late_comp_ques;
WO_Status_pkg.m_started_ques =  *m_started_ques;
WO_Status_pkg.m_idate =         *m_idate;

//          switch (wo_status) {

//          case C_WO_STATUS:
rc = invoke_coord(EID_wo_status);
//                              break;

//          default:
//                              return -1;
//}


// Update data with the retrieved/calculated value
//          *m_wono     =       WO_Status_pkg.m_wono;
            *m_cs       =       WO_Status_pkg.m_cs;
*m_sdate =          WO_Status_pkg.m_sdate;
*m_tcdate =         WO_Status_pkg.m_tcdate;
*m_tdate =          WO_Status_pkg.m_tdate;
*m_comp_ques =      WO_Status_pkg.m_comp_ques;
*m_late_start_ques = WO_Status_pkg.m_late_start_ques;
*m_late_comp_ques = WO_Status_pkg.m_late_comp_ques;
*m_started_ques =   WO_Status_pkg.m_started_ques;
*m_idate =          WO_Status_pkg.m_idate;

            return rc;
}

extern "C" __declspec(dllexport) int Req_del_inventory_item(Cstring m_mpn
                                                            )
{
```

```
int rc = 0;

//              MY_TRACE("..Req Del Inventory Item : copy parameters...\n");

d_inv_i_pkg.m_mpn =     m_mpn       ;

rc = invoke_coord(EID_del_inv_item);
return rc;

}
extern "C" __declspec(dllexport) int Req_create_po(    int po_select,
double                  *m_atc,
double                  *m_atc2,
double                  *m_atc3,
double                  *m_atc4,
CString                 m_atno,
CString                 m_atno2,
CString                 m_atno3,
CString                 m_atno4,
double                  *m_etc,
double                  *m_etc2,
double                  *m_etc3,
double                  *m_etc4,
CString                 m_etno,
CString                 m_etno2,
CString                 m_etno3,
CString                 m_etno4,
double                  *m_fst,
double                  *m_ist,
double                  *m_ist10,
double                  *m_ist11,
double                  *m_ist12,
double                  *m_ist2,
double                  *m_ist3,
double                  *m_ist4,
double                  *m_ist5,
double                  *m_ist6,
double                  *m_ist7,
double                  *m_ist8,
double                  *m_ist9,
COleDateTime            m_poid,
int                     m_pon,
double                  *m_post,
short                   m_pqty,
short                   m_pqty10,
short                   m_pqty11,
short                   m_pqty12,
short                   m_pqty2,
short                   m_pqty3,
short                   m_pqty4,
short                   m_pqty5,
short                   m_pqty6,
short                   m_pqty7,
short                   m_pqty8,
short                   m_pqty9,
double                  *m_pst,
CString                 m_satno,
CString                 m_satno2,
CString                 m_satno3,
CString                 m_satno4,
double                  *m_satc,
double                  *m_satc2,
double                  *m_satc3,
double                  *m_satc4,
double                  *m_tpc,
CString                 *m_va,
CString                 m_vna,
CString                 m_vno,
CString                 *m_vpn,
Cstring                 & m_status

                        )
{
int rc = 0;
CString strMsgText;

if (DebugMsg) {
strMsgText.Format("Running Submit DLL for VNA=%s VPN=%s ETNO=%s
PQTY=%d", m_vna, m_vpn, m_etno, m_pqty);
AfxMessageBox(strMsgText);
            }

create_PO_pkg.m_atc=                    *m_atc;
create_PO_pkg.m_atc2=                   *m_atc2;
create_PO_pkg.m_atc3=                   *m_atc3;
create_PO_pkg.m_atc4=                   *m_atc4;
create_PO_pkg.m_atno=                   m_atno;
create_PO_pkg.m_atno2=                  m_atno2;
create_PO_pkg.m_atno3=                  m_atno3;
create_PO_pkg.m_atno4=                  m_atno4;
create_PO_pkg.m_etc=                    *m_etc;
create_PO_pkg.m_etc2=                   *m_etc2;
create_PO_pkg.m_etc3=                   *m_etc3;
create_PO_pkg.m_etc4=                   *m_etc4;
create_PO_pkg.m_etno=                   m_etno;
create_PO_pkg.m_etno2=                  m_etno2;
create_PO_pkg.m_etno3=                  m_etno3;
create_PO_pkg.m_etno4=                  m_etno4;
create_PO_pkg.m_fst=                    *m_fst;
create_PO_pkg.m_ist=                    *m_ist;
create_PO_pkg.m_ist10=                  *m_ist10;
create_PO_pkg.m_ist11=                  *m_ist11;
create_PO_pkg.m_ist12=                  *m_ist12;
create_PO_pkg.m_ist2=                   *m_ist2;
create_PO_pkg.m_ist3=                   *m_ist3;
create_PO_pkg.m_ist4=                   *m_ist4;
create_PO_pkg.m_ist5=                   *m_ist5;
create_PO_pkg.m_ist6=                   *m_ist6;
create_PO_pkg.m_ist7=                   *m_ist7;
create_PO_pkg.m_ist8=                   *m_ist8;
create_PO_pkg.m_ist9=                   *m_ist9;
create_PO_pkg.m_va=                     *m_va;
create_PO_pkg.m_vpn=                    *m_vpn;

// Copy Address information. If there is no entry
// insert a blank
if (DebugMsg) {
strMsgText.Format("After Copy: VA=%s", create_PO_pkg.m_va);
AfxMessageBox(strMsgText);
            }

create_PO_pkg.m_poid =                  m_poid;
create_PO_pkg.m_pon =                   m_pon;
create_PO_pkg.m_pos t=                  *m_post;
create_PO_pkg.m_pqty =                  m_pqty;
create_PO_pkg.m_pqty10 =                m_pqty10;
create_PO_pkg.m_pqty11=                 m_pqty11;
create_PO_pkg.m_pqty12=                 m_pqty12;
create_PO_pkg.m_pqty2 =                 m_pqty2;
create_PO_pkg.m_pqty3 =                 m_pqty3;
create_PO_pkg.m_pqty4 =                 m_pqty4;
create_PO_pkg.m_pqty5 =                 m_pqty5;
create_PO_pkg.m_pqty6 =                 m_pqty6;
create_PO_pkg.m_pqty7 =                 m_pqty7;
create_PO_pkg.m_pqty8 =                 m_pqty8;
create_PO_pkg.m_pqty9 =                 m_pqty9;
create_PO_pkg.m_pst =                   *m_pst;
create_PO_pkg.m_satno =                 m_satno;
create_PO_pkg.m_satno2 =                m_satno2;
create_PO_pkg.m_satno3 =                m_satno3;
create_PO_pkg.m_satno4 =                m_satno4;
create_PO_pkg.m_satc =                  *m_satc;
create_PO_pkg.m_satc2 =                 *m_satc2;
create_PO_pkg.m_satc3 =                 *m_satc3;
create_PO_pkg.m_satc4 =                 *m_satc4;
create_PO_pkg.m_tpc =                   *m_tpc;
create_PO_pkg.m_vna =                   m_vna;
create_PO_pkg.m_vno =                   m_vno;

switch (po_select) {


case C_SUBMIT:
rc = invoke_coord(EID_purchase_order_submit);
break;


case C_CONFIRMPRINT:
rc = invoke_coord(EID_purchase_order_confirmprint);
break;


default:
return -1;
}

if (DebugMsg) {
strMsgText.Format("..Coordinator result = %d", rc);
AfxMessageBox(strMsgText);
}

if (rc != 0) {
            return rc;
            }

// Update data with the retrieved/calculated value
*m_fst = create_PO_pkg.m_fst;
*m_ist = create_PO_pkg.m_ist;
*m_ist10 = create_PO_pkg.m_ist10;
*m_ist11 = create_PO_pkg.m_ist11;
*m_ist12 = create_PO_pkg.m_ist12;
*m_ist2 = create_PO_pkg.m_ist2;
```

```
*m_ist3 = create_PO_pkg.m_ist3;
*m_ist4 = create_PO_pkg.m_ist4;
*m_ist5 = create_PO_pkg.m_ist5;
*m_ist6 = create_PO_pkg.m_ist6;
*m_ist7 = create_PO_pkg.m_ist7;
*m_ist8 = create_PO_pkg.m_ist8;
*m_ist9 = create_PO_pkg.m_ist9;
*m_va = create_PO_pkg.m_va;
*m_vpn = create_PO_pkg.m_vpn;
*m_etc = create_PO_pkg.m_etc;
*m_etc2 = create_PO_pkg.m_etc2;
*m_etc3 = create_PO_pkg.m_etc3;
*m_etc4 = create_PO_pkg.m_etc4;
*m_atc = create_PO_pkg.m_atc;
*m_atc2 = create_PO_pkg.m_atc2;
*m_atc3 = create_PO_pkg.m_atc3;
*m_atc4 = create_PO_pkg.m_atc4;
*m_satc = create_PO_pkg.m_satc;
*m_satc2 = create_PO_pkg.m_satc2;
*m_satc3 = create_PO_pkg.m_satc3;
*m_satc4 = create_PO_pkg.m_satc4;
*m_pst = create_PO_pkg.m_pst;
*m_post = create_PO_pkg.m_post;
*m_tpc = create_PO_pkg.m_tpc;


if (DebugMsg) {
strMsgText.Format("Completed Process DLL for VNA=%s VPN=%s
ETNO=%s PQTY=%d, ETC=%f, IST=%f", m_vna, m_vpn, m_etno, m_pqty,
*m_etc, *m_ist);
AfxMessageBox(strMsgText);
        }

return rc;
}

extern "C" __declspec(dllexport) int get_NM(int nm_select,

        COleDateTime        *m_today,
        CString             m_etno,
        CString             m_location,
        COleDateTime        *m_nwm,
        COleDateTime        *m_nmm,
        CString             *m_snm,
        CString             *m_snw,
        CString&            m_status
        )
{
        int rc = 0;
        CString strMsgText;

        if (DebugMsg) {
strMsgText.Format("Running Submit DLL for ETNO=%s LOCATION=%s",
  m_etno, m_location);
AfxMessageBox(strMsgText);
        }

        NM_pkg.m_today =        *m_today;
        NM_pkg.m_etno =         m_etno;
        NM_pkg.m_location =     m_location;
        NM_pkg.m_nwm =          *m_nwm;
        NM_pkg.m_nmm =          *m_nmm;
        NM_pkg.m_snm =          *m_snm;
        NM_pkg.m_snw =          *m_snw;

        switch (nm_select) {
        case C_WK:
        rc = invoke_coord(EID_get_wk_nm);
        break;
        case C_MON:
        rc = invoke_coord(EID_get_mon_nm);
        break;

        default:
        return -1;
        }

        if (DebugMsg) {
        strMsgText.Format("..Coordinator result = %d", rc);
        AfxMessageBox(strMsgText);
        }

        if (rc != 0) {
        return rc;
        }


        // Update data with the retrieved/calculated value

*m_today =  NM_pkg.m_today;
*m_nwm =    NM_pkg.m_nwm;
*m_nmm =    NM_pkg.m_nmm;
*m_snm =    NM_pkg.m_snm;
*m_snw =    NM_pkg.m_snw;


//          if (DebugMsg) {
//          strMsgText.Format("Completed Process DLL for VNA=%s
VPN=%s ETNO=%s PQTY=%d, ETC=%f, IST=%f",
// m_vna, m_vpn, m_etno, m_pqty, *m_etc, *m_ist);
//AfxMessageBox(strMsgText);
//          }

  return rc;
}
//
// Request: PO Clear
//

extern "C" __declspec(dllexport) int Req_PO_Order_Clear()
{
int rc = 0;

MY_TRACE("..Req Sales Order Clear...\n");

rc = invoke_coord(EID_PO_Clear);
MY_TRACE1("..Coordinator result = %d...\n", rc);

return rc;
}

//
// Request: WO Clear
//

extern "C" __declspec(dllexport) int Req_WO_Order_Clear()
{
int rc = 0;

MY_TRACE("..Req Work Order Clear...\n");

rc = invoke_coord(EID_WO_Clear);
MY_TRACE1("..Coordinator result = %d...\n", rc);

return rc;
}

extern "C" __declspec(dllexport) int Get_WO_Status_Clear()
{
int rc = 0;

MY_TRACE("..Work Order Status Clear...\n");

rc = invoke_coord(EID_WO_Clear);
MY_TRACE1("..Coordinator result = %d...\n", rc);

return rc;
}

//
// Request: Set Coord Debug Message
//

extern "C" __declspec(dllexport) void Req_Set_TraceWindowHandle(HWND
hwnd)
{
g_hwndTrace =  hwnd;
}

extern "C" __declspec(dllexport) int Req_Set_DebugMsg(BOOL enable)
{
int rc = 0;

MY_TRACE("..Req Set coord debug message...\n");

if (enable)
rc = invoke_coord(EID_debug_coord_tracemsg_enable);
else
rc = invoke_coord(EID_debug_coord_tracemsg_disable);

MY_TRACE1("..Coordinator result = %d...\n", rc);

return rc;
}


//
// Request: Get Coord Instance
//
extern "C" __declspec(dllexport) int Req_Get_CoordInst(int *coord_inst)
{
int rc = 0;
```

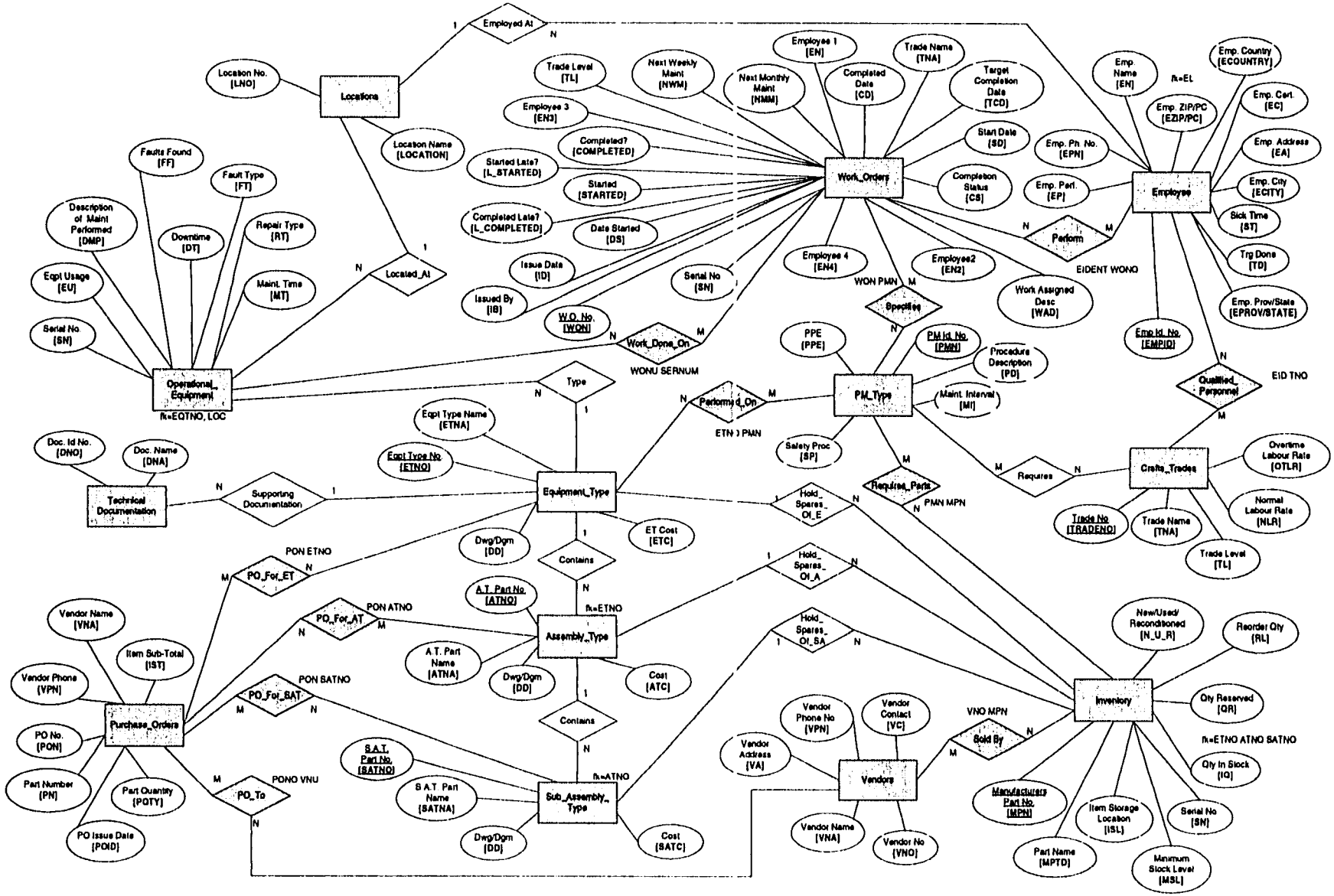
```
MY_TRACE("..Req Get Coord Instance...\n");
rc = invoke_coord(EID_info_get_coord_instance);
*coord_inst = instance_image;

MY_TRACE1("..Coordinator result = %d...\n", rc);

return rc;
}
```

Coordwrap.h

```
// CoordWrap.h : Wrapper for the Main coordinator code.
// Implementation in CoordWrap.cpp.
//
// This wrapper code will implement all the externally callable
// functions that interface with different implementations of
// the User Interfaces.
//
// The approach is to provide a DLL function for each possible
// event that can be triggered by the UI commands.
// Note that the UI implementation is responsible to prompt the
// user for all the necessary information.
//
// When the function of the wrapper is invoked, the work it does
// is to use the information passed in from the UI, initialize
// the internal data structures and eventually call the
// "invoke_coordinator" routine that manage all the responses.
//
// Note that since there can be simultaneously different kind
// of UI initiating requests which is implemented as different
// applications (but invoking the same coordinator DLL), there
// will be multiple threads. This can be resolved by either
// serializing the incoming requests, or manage through
// data. The chosen method here is to use data. For simplicity,
// the different instances of coordinator wrapper will check
// for the availability of the coordinator by checking the
// "Control Data" (One can view this as the state of the
// system.

// Initialize the DLL, register the classes etc
extern "C" __declspec(dllexport) int Req_add_inventory_item(
CString     m_mptd,
CString     m_mpn,
CString     m_sn,
CString     m_atno,
CString     m_etno,
CString     m_satno,
short       m_qr,
CString     m_nur,
short       m_iqty,
CString     m_isl,
short       m_msl
            );

#define     C_EQ_INFO           0
#define     C_EMP_INFO          1
#define     C_SAVE              2
#define     C_WOEDIT            3
#define     C_WOUPDATE          4

extern "C" __declspec(dllexport) int Req_create_wo(int wo_select,
CString                 m_won,
CString                 *m_sn,
COleDateTime            *m_id,
CString                 *m_ib,
COleDateTime            *m_sd,
COleDateTime            *m_tcd,
CString                 *m_combo_en,
CString                 *m_combo_en2,
CString                 *m_combo_en3,
CString                 *m_combo_en4,
CString                 *m_dmp,
CString                 *m_ff,
CString                 *m_location,
CString                 *m_mt,
CString                 *m_etna,
CString                 *m_wad,
Cstring                 &m_status,
CString                 *m_tna,
CString                 *m_tl,
COleDateTime            *m_cd,
COleDateTime            *m_ds

                                                    );
//#define     C_WO_STATUS
extern "C" __declspec(dllexport) int get_WO_Status(
CString                 m_won,
CString                 *m_cs,
COleDateTime            *m_sdate,
COleDateTime            *m_tcdate,
COleDateTime            *m_tdate,
CString                 *m_comp_ques,
CString                 *m_late_start_ques,
CString                 *m_late_comp_ques,
CString                 *m_started_ques,
COleDateTime            *m_idate
                        );

extern "C" __declspec(dllexport) int Req_del_inventory_item(
CString     m_mpn
            );

// Define accepted range for po_select
#define     C_SUBMIT            0
#define C_CONFIRMPRINT          1

extern "C" __declspec(dllexport) int Req_create_po(    int po_select,
double      *m_atc,
double      *m_atc2,
double      *m_atc3,
double      *m_atc4,
CString     m_atno,
CString     m_atno2,
CString     m_atno3,
CString     m_atno4,
double      *m_etc,
double      *m_etc2,
double      *m_etc3,
double      *m_etc4,
CString     m_etno,
CString     m_etno2,
CString     m_etno3,
CString     m_etno4,
double      *m_fst,
double      *m_ist,
double      *m_ist10,
double      *m_ist11,
double      *m_ist12,
double      *m_ist2,
double      *m_ist3,
double      *m_ist4,
double      *m_ist5
double      *m_ist6,
double      *m_ist7,
double      *m_ist8,
double      *m_ist9,
COleDateTime            m_poid,
int                     m_pon,
double      *m_post,
short       m_pqty,
short       m_pqty10,
short       m_pqty11,
short       m_pqty12,
short       m_pqty2,
short       m_pqty3,
short       m_pqty4,
short       m_pqty5,
short       m_pqty6,
short       m_pqty7,
short       m_pqty8,
short       m_pqty9,
double      *m_pst,
CString     m_satno,
CString     m_satno2,
CString     m_satno3,
CString     m_satno4,
double      *m_satc,
double      *m_satc2,
double      *m_satc3,
double      *m_satc4,
double      *m_tpc,
CString     *m_va,
CString     m_vna,
CString     m_vno,
CString     *m_vpn,
CString& m_status

            );

#define     C_WK        0
#define C_MON           1

extern "C" __declspec(dllexport) int get_NM( int nm_select,

            COleDateTime            *m_today,
            CString                         m_etno,
            CString                         m_location,
            COleDateTime            *m_nwm,
            COleDateTime            *m_nmm,
            CString                         *m_snm,
            CString                         *m_snw,
```

```
                CString&                m_status

                );

// Request: PO Clear

extern "C" __declspec(dllexport) int Req_PO_Order_Clear();

// Request: WO Clear

extern "C" __declspec(dllexport) int Req_WO_Order_Clear();

// Request: WO Status Clear

extern "C" __declspec(dllexport) int Get_WO_Status_Clear();

// Request: Set Coord Debug Message
//
extern "C" __declspec(dllexport) int Req_Set_DebugMsg(BOOL enable);

//
// Request: Get Coord Instance
//
extern "C" __declspec(dllexport) int Req_Get_CoordInst(int* instance);

extern "C" __declspec(dllexport) void Req_Set_TraceWindowHandle(HWND
hwnd);
```

11. Annex D - CMMS Database Entity Relationship Diagram


Employed At
Location No. (LNO)
Locations
Location Name (LOCATION)
Located_At
Faults Found (FF)
Fault Type (FT)
Description of Maint Performed (DMP)
Downtime (DT)
Repair Type (RT)
Eqpt Usage (EU)
Maint. Time (MT)
Serial No. (SN)
Operational_ Equipment
fk=EQTNO, LOC
Trade Level (TL)
Next Weekly Maint (NWM)
Next Monthly Maint (NMM)
Employee 1 (EN)
Trade Name (TNA)
Completed Date (CD)
Target Completion Date (TCD)
Employee 3 (EN3)
Completed? (COMPLETED)
Started Late? (L_STARTED)
Started (STARTED)
Completed Late? (L_COMPLETED)
Date Started (DS)
Start Date (SD)
Completion Status (CS)
Work_Orders
Issue Date (ID)
Issued By (IB)
W.O. No. (WON)
Serial No (SN)
Work_Done_On
WONU SERNUM
Employee 4 (EN4)
Employee2 (EN2)
WON PMN
Specifies
Work Assigned Desc (WAD)
Perform
EIDENT WONO
Emp. Name (EN)
Emp. Country (ECOUNTRY)
fk=EL
Emp. ZIP/PC (EZIP/PC)
Emp. Cert. (EC)
Emp. Ph. No. (EPN)
Emp. Address (EA)
Emp. Perf. (EP)
Employee
Emp. City (ECITY)
Sick Time (ST)
Trg Done (TD)
Emp. Prov/State (EPROV/STATE)
Emp Id. No. (EMPID)
Qualified_ Personnel
EID TNO
PPE (PPE)
PM Id. No. (PMN)
Procedure Description (PD)
Maint. Interval (MI)
Performed_On
ETN) PMN
PM_Type
Safety Proc (SP)
Requires_Parts
PMN MPN
Requires
Crafts_Trades
Overtime Labour Rate (OTLR)
Normal Labour Rate (NLR)
Trade No. (TRADENO)
Trade Name (TNA)
Trade Level (TL)
Type
Eqpt Type Name (ETNA)
Eqpt Type No. (ETNO)
Equipment_Type
Dwg/Dgm (DD)
ET Cost (ETC)
Contains
Doc. Id No. (DNO)
Doc. Name (DNA)
Technical Documentation
Supporting Documentation
Hold_ Spares_ Of_E
Hold_ Spares_ Of_A
Hold_ Spares_ Of_SA
PON ETNO
PO_For_ET
PON ATNO
PO_For_AT
PON SATNO
PO_For_SAT
A.T. Part No. (ATNO)
fk=ETNO
Assembly_Type
A.T. Part Name (ATNA)
Dwg/Dgm (DD)
Cost (ATC)
Contains
S.A.T. Part No. (SATNO)
S.A.T. Part Name (SATNA)
fk=ATNO
Sub_Assembly_ Type
Dwg/Dgm (DD)
Cost (SATC)
Vendor Name (VNA)
Item Sub-Total (IST)
Vendor Phone (VPN)
Purchase_Orders
PO No. (PON)
Part Number (PN)
Part Quantity (PQTY)
PO Issue Date (POID)
PONO VNU
PO_To
Vendor Phone No (VPN)
Vendor Contact (VC)
Vendor Address (VA)
Vendors
Vendor Name (VNA)
Vendor No (VNO)
VNO MPN
Sold By
New/Used/ Reconditioned (N_U_R)
Reorder Qty (RL)
Inventory
Qty Reserved (QR)
fk=ETNO ATNO SATNO
Qty In Stock (IQ)
Manufacturers Part No. (MPN)
Item Storage Location (ISL)
Serial No (SN)
Part Name (MPTD)
Minimum Stock Level (MSL)

12. Annex E - Activity Executor Code

InvDBHandler.cls

```
Option Explicit

Public InvDB_AccSel As Integer

Public Function openInvDB() As Integer
   ' This function opens the Inventory database.
   ' It assumes the name of the Inventory database is called "Inventory.mdb"
and
   ' is located in the path "e:\training\prototype\database". Change the drive
   ' and the path if necessary.
   ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
   ' calls.
   ' The opened database will be stored in the "InvDB" object handle declared
in
   ' the module "InvDBStor".
   Dim rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute openInvDB activity with Access Selection
" & InvDB_AccSel
   End If

   ' Open the database
   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_openInvDB
         If InvDB_Trace Then
            MsgBox "InvDB_DLL:DAO_openInvDB result = " & rc
         End If
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   openInvDB = rc
End Function


Public Function closeInvDB() As Integer
   ' This function closes the Inventory database.
   ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
   ' calls.
   ' The opened database object handle is stored in the variable declared in
   ' the corresponding module that handles the access methods.
   Dim rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute closeInvDB activity"
   End If

   ' Close the database
   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_closeInvDB
         If InvDB_Trace Then
            MsgBox "InvDB_DLL:DAO_closeInvDB result = " & rc
         End If
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   closeInvDB = rc
End Function

Public Function recExistPN(ByVal mpn As String, ByRef exist As Boolean) As
Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Inventory DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim pn_rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute MPN Uniqness activity "
   End If

   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         pn_rc = DAO_recExistPN(mpn, exist)
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         pn_rc = -1
   End Select
   recExistPN = pn_rc
End Function


Public Function recCreate(ByVal mptd As String, _
                  ByVal mpn As String, _
                  ByVal sn As String, _
                  ByVal atno As String, _
                  ByVal etno As String, _
                  ByVal satno As String, _
                  ByVal qr As Integer, _
                  ByVal nur As String, _
                  ByVal iqty As Integer, _
                  ByVal isl As String, _
                  ByVal msl As Integer) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Inventory DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute Create Inv Record activity"
   End If

   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recCreate(mptd, mpn, sn, atno, etno, satno, qr, nur, iqty,
isl, msl)
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recCreate = rc
End Function

Public Function recDelete(ByVal mpn As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Inventory DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute Create Inv Record activity"
   End If

   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recDelete(mpn)
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recDelete = rc
End Function

Public Function getMPTD(ByVal mpn As String, _
                  ByRef mptd As String) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the Inventory DB.
   Dim rc As Integer

   If InvDB_Trace Then
      MsgBox "InvDB_DLL:Execute Get Item Info activity"
   End If

   Select Case InvDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getMPTD(mpn, mptd)
      Case Else
         MsgBox "InvDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getMPTD = rc
End Function


Public Sub setInvDBTrace(ByVal ToTrace As Boolean)
```

```
  InvDB_Trace = ToTrace

End Sub
```

InvDB_Jet_DAO.bas

```
Option Explicit

' This module is to implement the database access using the VB built in Jet
engine
' and the DAO model.
Private InvWS_handle As Workspace
Private InvDB_handle As Database

Function DAO_openInvDB() As Integer
   ' Using the OpenDatabase Method in DAO
   ' It uses the OpenDatabase method to open the Microsoft Jet-based
Inventory database.

   ' Create Microsoft Jet Workspace object.
   ' Set wrkJet = CreateWorkspace("", "admin", "", dbUseJet)
   Set InvWS_handle = DBEngine.Workspaces(0)

   ' Open Database object from saved Microsoft Jet database
   ' for exclusive use (i.e. option = True).
   On Error GoTo openDbErr
      If InvDB_Trace Then
         MsgBox "DAO:Open database"
      End If
      Set InvDB_handle = InvWS_handle.OpenDatabase("C:\CMMS
Newest\SalesSystem.2\Project\Data\CMMS.mdb", True)
      DAO_openInvDB = 0
      Exit Function

openDbErr:
   MsgBox "DAO:Open Database error : " & Err.Number
   DAO_openInvDB = -1
   Exit Function

End Function


Function DAO_closeInvDB() As Integer
   ' Use the Close Method in DAO
   InvDB_handle.Close
   InvWS_handle.Close
   DAO_closeInvDB = 0
End Function


Function DAO_recExistPN(ByVal mpn As String, ByRef exist As Boolean) As
Integer
   ' Check to ensure that the Inventory DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
   Dim invRec As Recordset
   Dim strQuery As String
   Dim qdf As QueryDef

   ' Ensure the database is availabile for access. The
   ' check should never fail because the coordinator should
   ' have ensure the Inventory DB handle resource is secured.
   If InvDB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Inventory Database not available"
      exist = False
      DAO_recExistPN = -1
      Exit Function
   End If

   ' *** Create a SQL query ***
   If InvDB_Trace Then
      MsgBox "***** Start recExistPN *****
   End If
   strQuery = "PARAMETERS [new_mpn] String; "
   strQuery = strQuery & "SELECT * FROM Inventory WHERE mpn =
[new_mpn];"
   Set qdf = InvDB_handle.CreateQueryDef("", strQuery)
   qdf.Parameters("new_mpn") = mpn
   Set invRec = qdf.OpenRecordset(dbReadOnly)
   If invRec.EOF Then
      ' Record not found.
      If InvDB_Trace Then
         MsgBox "DAO:Record with mpn=" & mpn & " NOT found"
      End If
      exist = False
      DAO_recExistPN = 0
   Else
      ' Record found. No need to search further
      If InvDB_Trace Then
         MsgBox "DAO:Record with mpn=" & mpn & " found"
      End If
      exist = True
      DAO_recExistPN = 0
   End If
   invRec.Close
   qdf.Close

   If InvDB_Trace Then
      MsgBox "***** Exit recExistPN *****
   End If

End Function

Function DAO_recCreate(ByVal mptd As String, _
                       ByVal mpn As String, _
                       ByVal sn As String, _
                       ByVal atno As String, _
                       ByVal etno As String, _
                       ByVal satno As String, _
                       ByVal qr As Integer, _
                       ByVal nur As String, _
                       ByVal iqty As Integer, _
                       ByVal isl As String, _
                       ByVal msl As Integer) As Integer
   ' Check to ensure that the Inventory DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
   Dim invRec As Recordset

   ' Ensure the database is availabile for access. The
   ' check should never fail because the coordinator should
   ' have ensure the Inventory DB handle resource is secured.
   If InvDB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Inventory Database not available"
      DAO_recCreate = -1
      Exit Function
   End If

   Set invRec = InvDB_handle.OpenRecordset("Inventory", dbOpenTable)

   If InvDB_Trace Then
      MsgBox "DAO:Add new record"
   End If
   invRec.AddNew
   invRec!mpn = mpn
   invRec!mptd = mptd
   invRec!sn = sn
   invRec!atno = atno
   invRec!etno = etno
   invRec!satno = satno
   invRec!qr = qr
   invRec!nur = nur
   invRec!iqty = iqty
   invRec!isl = isl
   invRec!msl = msl
   invRec.Update
   invRec.Close

   DAO_recCreate = 0
End Function

Function DAO_recDelete(ByVal mpn As String) As Integer
   ' Check to ensure that the Inventory DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
   Dim strQuery As String
   Dim invRec As Recordset
   Dim qdf As QueryDef

   ' Ensure the database is availabile for access. The
   ' check should never fail because the coordinator should
   ' have ensure the Inventory DB handle resource is secured.
   If InvDB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Inventory Database not available"
      ' exist = False
      DAO_recDelete = -1
      Exit Function
   End If

   ' *** Create a SQL query ***
   If InvDB_Trace Then
      MsgBox "***** Start recDelete *****
   End If
   strQuery = "PARAMETERS [new_mpn] String; "
   strQuery = strQuery & "DELETE * FROM Inventory WHERE mpn =
[new_mpn];"
```

```
    Set qdf = InvDB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_mpn") = mpn
    qdf.Execute



End Function

Function DAO_getMPTD(ByVal mpn As String, _
                     ByRef mptd As String) As Integer
    ' Check to ensure that the Inventory DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim invRec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If InvDB_handle Is Nothing Then
       MsgBox "DAO:Validation Fail : Inventory Database not available"
       ' exist = False
       DAO_getMPTD = -1
       Exit Function
    End If

    ' *** Create a SQL query ***
    If InvDB_Trace Then
       MsgBox "***** Start getMPTD *****"
    End If
    strQuery = "PARAMETERS [new_mpn] String; "
    strQuery = strQuery & "SELECT MPTD FROM Inventory WHERE mpn =
[new_mpn];"
    Set qdf = InvDB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_mpn") = mpn
    Set invRec = qdf.OpenRecordset(dbOpenDynaset)
    If invRec.EOF Then
       ' Record not found.
       If InvDB_Trace Then
          MsgBox "DAO:Record with mpn=" & mpn & " NOT found"
       End If
       DAO_getMPTD = -1
    Else
       ' Record found. No need to search further
       If InvDB_Trace Then
          MsgBox "DAO:Record with mpn=" & mpn & " found"
       End If
       DAO_getMPTD = 0
    End If
    invRec.Close
    qdf.Close

    If InvDB_Trace Then
       MsgBox "***** Exit getMPTD  *****"
    End If

End Function
```

DBHandler.cls

```
Option Explicit

Public DB_AccSel As Integer

Public Function openDB() As Integer
    ' This function opens the Inventory database.
    ' It assumes the name of the Inventory database is called "Inventory.mdb"
and
    ' is located in the path "e:\training\prototype\database". Change the drive
    ' and the path if necessary.
    ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
    ' calls.
    ' The opened database will be stored in the "InvDB" object handle declared
in
    ' the module "InvDBStor".
    Dim rc As Integer

    If DB_Trace Then
       MsgBox "DB_DLL:Execute openDB activity with Access Selection " &
DB_AccSel
    End If

    ' Open the database
    Select Case DB_AccSel
       Case 1
          ' Return result code for the activity.
          rc = DAO_openDB
          If DB_Trace Then
             MsgBox "DB_DLL:DAO_openDB result = " & rc
          End If
       Case Else
          MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
          ' Return result code for the activity.
          rc = -1
    End Select
    openDB = rc
End Function


Public Function closeDB() As Integer
    ' This function closes the Inventory database.
    ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
    ' calls.
    ' The opened database object handle is stored in the variable declared in
    ' the corresponding module that handles the access methods.
    Dim rc As Integer

    If DB_Trace Then
       MsgBox "DB_DLL:Execute closeInvDB activity"
    End If

    ' Close the database
    Select Case DB_AccSel
       Case 1
          ' Return result code for the activity.
          rc = DAO_closeDB
          If DB_Trace Then
             MsgBox "DB_DLL:DAO_closeDB result = " & rc
          End If
       Case Else
          MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
          ' Return result code for the activity.
          rc = -1
    End Select
    closeDB = rc
End Function

Public Function recExistSN(ByVal sn As String, ByRef exist As Boolean) As
Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Inventory DB.
    ' Returns TRUE if such record exists, FALSE otherwise
    Dim rc As Integer

    If DB_Trace Then
       MsgBox "DB_DLL:Execute MPN Uniqness activity "
    End If

    Select Case DB_AccSel
       Case 1
          ' Return result code for the activity.
          rc = DAO_recExistSN(sn, exist)
       Case Else
          MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
          ' Return result code for the activity.
          rc = -1
    End Select
    recExistSN = rc
End Function

Public Function recExistTNA(ByVal tna As String, ByRef exist As Boolean)
As Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Inventory DB.
    ' Returns TRUE if such record exists, FALSE otherwise
    Dim rc As Integer

    If DB_Trace Then
       MsgBox "DB_DLL:Execute TNA Uniqness activity "
    End If

    Select Case DB_AccSel
       Case 1
          ' Return result code for the activity.
          rc = DAO_recExistTNA(tna, exist)
       Case Else
          MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
          ' Return result code for the activity.
          rc = -1
    End Select
    recExistTNA = rc
End Function
```

```
Public Function recExistTL(ByVal tl As String, ByRef exist As Boolean) As
Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Inventory DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute TL Uniqness activity "
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recExistTL(tl, exist)
      Case Else
         MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recExistTL = rc
End Function

Public Function SaveRec(ByVal won As String, _
                  ByVal sn As String, _
                  ByVal id As Date, _
                  ByVal ib As String, _
                  ByVal sd As Date, _
                  ByVal tcd As Date, _
                  ByVal combo_en As String, _
                  ByVal combo_en2 As String, _
                  ByVal combo_en3 As String, _
                  ByVal combo_en4 As String, _
                  ByVal wad As String, _
                  ByVal tna As String, _
                  ByVal tl As String) As Integer

   ' This function check if a record containing the fieldname
   ' is already existed in the Inventory DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Create Inv Record activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_SaveRec(won, sn, id, ib, _
         sd, tcd, combo_en, _
         combo_en2, combo_en3, combo_en4, wad, tna, tl)

      Case Else
         MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   SaveRec = rc
End Function

Public Function getEqptInfo(ByVal sn As String, _
                  ByRef etna As String, _
                  ByRef dmp As String, _
                  ByRef ff As String, _
                  ByRef mt As String, _
                  ByRef location As String) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the Inventory DB.
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Get Item Info activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getEqptInfo(sn, etna, dmp, ff, mt, location)
      Case Else
         MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getEqptInfo = rc
End Function

Public Function getWOInfo(ByVal won As String, _
                  ByRef sn As String, _
                  ByRef id As Date, _
                  ByRef ib As String, _
                  ByRef sd As Date, _
                  ByRef tcd As Date, _
                  ByRef combo_en As String, _
                  ByRef combo_en2 As String, _
                  ByRef combo_en3 As String, _
                  ByRef combo_en4 As String, _
                  ByRef dmp As String, _
                  ByRef ff As String, _
                  ByRef location As String, _
                  ByRef mt As String, _
                  ByRef etna As String, _
                  ByRef wad As String, _
                  ByRef tna As String, _
                  ByRef tl As String, _
                  ByRef cd As Date, _
                  ByRef ds As Date) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the Inventory DB.
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Get Item Info activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getWOInfo(won, sn, id, ib, sd, tcd, combo_en, combo_en2,
combo_en3, combo_en4, dmp, ff, location, mt, etna, wad, tna, tl, cd, ds)
      Case Else
         MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getWOInfo = rc
End Function

Public Function getEmpInfo(ByVal tna As String, _
                  ByVal tl As String, _
                  ByVal location As String, _
                  ByRef combo_en As String) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the Inventory DB.
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Get Emp Info activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getEmpInfo(tna, tl, location, combo_en)
      Case Else
         MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getEmpInfo = rc
End Function

Public Function recUpdateWO(ByVal won As String, _
                  ByVal sn As String, _
                  ByVal id As Date, _
                  ByVal ib As String, _
                  ByVal sd As Date, _
                  ByVal tcd As Date, _
                  ByVal combo_en As String, _
                  ByVal combo_en2 As String, _
                  ByVal combo_en3 As String, _
                  ByVal combo_en4 As String, _
                  ByVal wad As String, _
                  ByVal tna As String, _
                  ByVal tl As String, _
                  ByVal cd As Date, _
                  ByVal ds As Date) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Customer DB.
   ' If the part exists, assign the new item sale price value
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Edit WO activity"
   End If
```

```
    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_recUpdateWO(won, sn, id, ib, _
            sd, tcd, combo_en, _
            combo_en2, combo_en3, combo_en4, wad, tna, tl, cd, ds)
        Case Else
            MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    recUpdateWO = rc
End Function

Public Function recUpdateWO2(ByVal sn As String, _
                ByVal dmp As String, _
                ByVal ff As String, _
                ByVal mt As String) As Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Customer DB.
    ' If the part exists, assign the new item sale price value
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Edit WO activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_recUpdateWO2(sn, dmp, ff, mt)
        Case Else
            MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    recUpdateWO2 = rc
End Function

Public Function getWOStatus(ByVal wono As String,
                ByRef sdate As Date, _
                ByRef tcdate As Date, _
                ByRef idate As Date, _
                ByRef late_start_ques As String, _
                ByRef started_ques As String, _
                ByRef late_comp_ques As String, _
                ByRef comp_ques As String, _
                ByRef cs As String) As Integer
    ' This function retrieve the whole record and return
    ' the field values that are existed in the Inventory DB.
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Get Emp Info activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getWOStatus(wono, sdate, tcdate, idate, late_start_ques,
started_ques, late_comp_ques, comp_ques, cs)
        Case Else
            MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getWOStatus = rc
End Function

Public Function getWKnm(ByVal etno As String, _
                ByVal location As String, _
                ByRef nwm As Date, _
                ByRef snw As String) As Integer
    ' This function retrieve the whole record and return
    ' the field values that are existed in the Inventory DB.
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Get Emp Info activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getWKnm(etno, location, nwm, snw)
        Case Else
            MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getWKnm = rc
End Function

Public Function getMONnm(ByVal etno As String, _
                ByVal location As String, _
                ByRef nmm As Date, _
                ByRef snm As String) As Integer
    ' This function retrieve the whole record and return
    ' the field values that are existed in the Inventory DB.
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Get Emp Info activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getMONnm(etno, location, nmm, snm)
        Case Else
            MsgBox "DB_DLL:Other Access Selection chosen but not supported
yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getMONnm = rc
End Function

Public Function Update_Started(ByVal late_start_ques As String) As Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Customer DB.
    ' If the part exists, assign the new item sale price value
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Edit WO activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_Update_Started(late_start_ques)
        Case Else
            MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    Update_Started = rc
End Function

Public Function Update_Started2(ByVal started_ques As String) As Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Customer DB.
    ' If the part exists, assign the new item sale price value
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Edit WO activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_Update_Started2(started_ques)
        Case Else
            MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    Update_Started2 = rc
End Function
Public Function Update_Started3(ByVal late_comp_ques As String) As
Integer
    ' This function check if a record containing the fieldname
    ' is already existed in the Customer DB.
    ' If the part exists, assign the new item sale price value
    Dim rc As Integer

    If DB_Trace Then
        MsgBox "DB_DLL:Execute Edit WO activity"
    End If

    Select Case DB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_Update_Started3(late_comp_ques)
```

```
      Case Else
         MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   Update_Started3 = rc
End Function

Public Function Update_Started4(ByVal comp_ques As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Customer DB.
   ' If the part exists, assign the new item sale price value
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Edit WO activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_Update_Started4(comp_ques)
      Case Else
         MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   Update_Started4 = rc
End Function

Public Function cs1(ByVal cs As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Customer DB.
   ' If the part exists, assign the new item sale price value
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Edit WO activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_cs1(cs)
      Case Else
         MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   cs1 = rc
End Function
Public Function cs2(ByVal cs As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Customer DB.
   ' If the part exists, assign the new item sale price value
   Dim rc As Integer

   If DB_Trace Then
      MsgBox "DB_DLL:Execute Edit WO activity"
   End If

   Select Case DB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_cs2(cs)
      Case Else
         MsgBox "WODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   cs2 = rc
End Function



Public Sub setDBTrace(ByVal ToTrace As Boolean)

 DB_Trace = ToTrace

End Sub
```

DB_Jet_DAO.bas

```
Option Explicit

' This module is to implement the database access using the VB built in Jet
engine
' and the DAO model.
Private WS_handle As Workspace
Private DB_handle As Database


Function DAO_openDB() As Integer
   ' Using the OpenDatabase Method in DAO
   ' It uses the OpenDatabase method to open the Microsoft Jet-based
Inventory database.

   ' Create Microsoft Jet Workspace object.
   ' Set wrkJet = CreateWorkspace("", "admin", "", dbUseJet)
   Set WS_handle = DBEngine.Workspaces(0)

   ' Open Database object from saved Microsoft Jet database
   ' for exclusive use (i.e. option = True).
   On Error GoTo openDbErr
      If DB_Trace Then
         MsgBox "DAO:Open database"
      End If
      Set DB_handle = WS_handle.OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb", True)
      DAO_openDB = 0
      Exit Function

openDbErr:
   MsgBox "DAO:Open Database error : " & Err.Number
   DAO_openDB = -1
   Exit Function

End Function


Function DAO_closeDB() As Integer
   ' Use the Close Method in DAO
   DB_handle.Close
   WS_handle.Close
   DAO_closeDB = 0
End Function

Function DAO_recExistSN(ByVal sn As String, ByRef exist As Boolean) As
Integer
   ' Check to ensure that the DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
   Dim Rec As Recordset
   Dim strQuery As String
   Dim qdf As QueryDef

   ' Ensure the database is availabile for access. The
   ' check should never fail because the coordinator should
   ' have ensure the Inventory DB handle resource is secured.
   If DB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Inventory Database not available"
      exist = False
      DAO_recExistSN = -1
      Exit Function
   End If

   ' *** Create a SQL query ***
   If DB_Trace Then
      MsgBox "***** Start recExistSN *****"
   End If
   strQuery = "PARAMETERS [new_sn] String; "
   strQuery = strQuery & "SELECT * FROM Operational_Equipment WHERE
SN = [new_sn];"
   Set qdf = DB_handle.CreateQueryDef("", strQuery)
   qdf.Parameters("new_sn") = sn
   Set Rec = qdf.OpenRecordset(dbReadOnly)
   If Rec.EOF Then
      ' Record not found.
      If DB_Trace Then
         MsgBox "DAO:Record with SN=" & sn & " NOT found"
      End If
      exist = False
      DAO_recExistSN = 0
   Else
      ' Record found. No need to search further
      If DB_Trace Then
         MsgBox "DAO:Record with SN=" & sn & " found"
      End If
      exist = True
      DAO_recExistSN = 0
   End If
```

```
    Rec.Close
    qdf.Close

    If DB_Trace Then
        MsgBox "***** Exit recExistSN *****"
    End If

End Function

Function DAO_recExistTNA(ByVal tna As String, ByRef exist As Boolean) As
Integer
    ' Check to ensure that the DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        exist = False
        DAO_recExistTNA = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If DB_Trace Then
        MsgBox "***** Start recExistTNA *****"
    End If
    strQuery = "PARAMETERS [new_tna] String; "
    strQuery = strQuery & "SELECT * FROM Crafts_Trades WHERE TNA =
[new_tna];"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_tna") = tna
    Set Rec = qdf.OpenRecordset(dbReadOnly)
    If Rec.EOF Then
        ' Record not found.
        If DB_Trace Then
            MsgBox "DAO:Record with TNA=" & tna & " NOT found"
        End If
        exist = False
        DAO_recExistTNA = 0
    Else
        ' Record found. No need to search further
        If DB_Trace Then
            MsgBox "DAO:Record with TNA=" & tna & " found"
        End If
        exist = True
        DAO_recExistTNA = 0
    End If
    Rec.Close
    qdf.Close

    If DB_Trace Then
        MsgBox "***** Exit recExistTNA *****"
    End If

End Function

Function DAO_recExistTL(ByVal tl As String, ByRef exist As Boolean) As
Integer
    ' Check to ensure that the DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        exist = False
        DAO_recExistTL = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If DB_Trace Then
        MsgBox "***** Start recExistTL *****"
    End If
    strQuery = "PARAMETERS [new_tl] String; "
    strQuery = strQuery & "SELECT * FROM Crafts_Trades WHERE TL =
[new_tl];"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_tl") = tl
    Set Rec = qdf.OpenRecordset(dbReadOnly)
    If Rec.EOF Then
        ' Record not found.
        If DB_Trace Then
            MsgBox "DAO:Record with TL=" & tl & " NOT found"
        End If
        exist = False
        DAO_recExistTL = 0
    Else
        ' Record found. No need to search further
        If DB_Trace Then
            MsgBox "DAO:Record with TL=" & tl & " found"
        End If
        exist = True
        DAO_recExistTL = 0
    End If
    Rec.Close
    qdf.Close

    If DB_Trace Then
        MsgBox "***** Exit recExistTL *****"
    End If

End Function

Function DAO_SaveRec(ByVal won As String, _
                     ByVal sn As String, _
                     ByVal id As Date, _
                     ByVal ib As String, _
                     ByVal sd As Date, _
                     ByVal tcd As Date, _
                     ByVal combo_en As String, _
                     ByVal combo_en2 As String, _
                     ByVal combo_en3 As String, _
                     ByVal combo_en4 As String, _
                     ByVal wad As String, _
                     ByVal tna As String, _
                     ByVal tl As String) As Integer

    ' Check to ensure that the Inventory DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        DAO_SaveRec = -1
        Exit Function
    End If

    Set Rec = DB_handle.OpenRecordset("Work_Orders", dbOpenTable)

    If DB_Trace Then
        MsgBox "DAO:Add new record"
    End If
    Rec.AddNew
    Rec!won = won
    Rec!sn = sn
    Rec!id = id
    Rec!ib = ib
    Rec!sd = sd
    Rec!tcd = tcd
    Rec!en = combo_en
    Rec!en2 = combo_en2
    Rec!en3 = combo_en3
    Rec!en4 = combo_en4
    Rec!wad = wad
    Rec!tna = tna
    Rec!tl = tl
    Rec.Update
    Rec.Close

    DAO_SaveRec = 0
End Function
Function DAO_getEqptInfo(ByVal sn As String, _
                         ByRef etna As String, _
                         ByRef dmp As String, _
                         ByRef tf As String, _
                         ByRef mt As String, _
                         ByRef location As String) As Integer
    ' Check to ensure that the Inventory DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset
    Dim strQuery As String
```

```
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        ' exist = False
        DAO_getEqptInfo = -1
        Exit Function
    End If

    ' *** Create a SQL query ***NEED TO FIX SQL
    If DB_Trace Then
        MsgBox ***** Start getEqptInfo *****
    End If
    strQuery = "PARAMETERS [new_sn] String; "
    strQuery = strQuery & "SELECT ETNA, DMP, FF, MT, LOCATION FROM
Operational_Equipment, Equipment_Type, Locations WHERE SN=[new_sn]
AND EQTNO=ETNO AND LOC = LNO;"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_sn") = sn
    Set Rec = qdf.OpenRecordset(dbOpenDynaset)
    If Rec.EOF Then
        ' Record not found.
        If DB_Trace Then
            MsgBox "DAO:Record with SN=" & sn & " NOT found"
        End If
        DAO_getEqptInfo = -1
    Else
        ' Record found. No need to search further
        If DB_Trace Then
            MsgBox "DAO:Record with SN=" & sn & " found"
        End If
        ' Assign the new value to the IO field of the record
        etna = Rec!etna
        dmp = Rec!dmp
        ff = Rec!ff
        mt = Rec!mt
        location = Rec!location

        DAO_getEqptInfo = 0
    End If
    Rec.Close
    qdf.Close

    If DB_Trace Then
        MsgBox ***** Exit getEqptInfo *****
    End If

End Function

Function DAO_getWOInfo(ByVal won As String, _
                       ByRef sn As String, _
                       ByRef id As Date, _
                       ByRef ib As String, _
                       ByRef sd As Date, _
                       ByRef tcd As Date, _
                       ByRef combo_en As String, _
                       ByRef combo_en2 As String, _
                       ByRef combo_en3 As String, _
                       ByRef combo_en4 As String, _
                       ByRef dmp As String, _
                       ByRef ff As String, _
                       ByRef location As String, _
                       ByRef mt As String, _
                       ByRef etna As String, _
                       ByRef wad As String, _
                       ByRef tna As String, _
                       ByRef tl As String, _
                       ByRef cd As Date, _
                       ByRef ds As Date) As Integer
    ' Check to ensure that the Inventory DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        ' exist = False
        DAO_getWOInfo = -1
        Exit Function
    End If

    ' *** Create a SQL query ***NEED TO FIX SQL
    If DB_Trace Then
        MsgBox ***** Start getWOInfo *****
    End If
    strQuery = "PARAMETERS [new_won] String; "
    strQuery = strQuery & "SELECT Work_Orders.SN, ID, IB, SD, TCD, EN,
EN2, EN3, EN4, WAD, TNA, TL, CD, DS, DMP, FF, MT, ETNA, LOCATION
From Work_Orders, Operational_Equipment, Equipment_Type, Locations
WHERE WON=[new_won] AND Work_Orders.SN =
Operational_Equipment.SN AND EQTNO=ETNO AND LOC = LNO;"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_won") = won
    Set Rec = qdf.OpenRecordset(dbOpenDynaset)
    If Rec.EOF Then
        ' Record not found.
        If DB_Trace Then
            MsgBox "DAO:Record with WON=" & won & " NOT found"
        End If
        DAO_getWOInfo = -1
    Else
        ' Record found. No need to search further
        If DB_Trace Then
            MsgBox "DAO:Record with WON=" & won & " found"
        End If
        ' Assign the new value to the IO field of the record
        sn = Rec!sn
        id = Rec!id
        ib = Rec!ib
        sd = Rec!sd
        tcd = Rec!tcd
        combo_en = Rec!en
        combo_en2 = Rec!en2
        combo_en3 = Rec!en3
        combo_en4 = Rec!en4
        dmp = Rec!dmp
        ff = Rec!ff
        location = Rec!location
        mt = Rec!mt
        etna = Rec!etna
        wad = Rec!wad
        tna = Rec!tna
        tl = Rec!tl
        cd = Rec!cd
        ds = Rec!ds

        DAO_getWOInfo = 0
    End If
    Rec.Close
    qdf.Close

    If DB_Trace Then
        MsgBox ***** Exit getWOInfo *****
    End If

End Function


Function DAO_getEmpInfo(ByVal tna As String, _
                        ByVal tl As String, _
                        ByVal location As String, _
                        ByRef combo_en As String) As Integer

    ' Check to ensure that the DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim Rec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Inventory DB handle resource is secured.
    If DB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Inventory Database not available"
        ' exist = False
        DAO_getEmpInfo = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If DB_Trace Then
        MsgBox ***** Start getEmpInfo *****
    End If
    strQuery = "PARAMETERS [new_tna] String, [new_tl] String,
[new_location] String; "

    strQuery = strQuery & "SELECT EN FROM Employee, Crafts_Trades,
Qualified_Personnel WHERE EL = [new_location] AND EMPID = EID AND
TNO = TRADENO AND TNA = [new_tna] AND TL = [new_tl];"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
```

```
    qdf.Parameters("new_tna") = tna
    qdf.Parameters("new_tl") = tl
    qdf.Parameters("new_location") = location

      Set Rec = qdf.OpenRecordset(dbOpenDynaset)
    If Rec.EOF Then
      ' Record not found.
      If DB_Trace Then
        MsgBox "DAO:Record with TL=" & tl & " NOT found"
      End If
      DAO_getEmpInfo = -1
    Else
      ' Record found. No need to search further
      If DB_Trace Then
        MsgBox "DAO:Record with TL=" & tl & " found"
      End If
      ' Assign the new value to the IQ field of the record
      Rec.MoveFirst
      While Not Rec.EOF
       combo_en = combo_en + Rec!en
       combo_en = combo_en + ","
       Rec.MoveNext
      Wend

      DAO_getEmpInfo = 0
    End If
    Rec.Close
    qdf.Close


    If DB_Trace Then
      MsgBox "***** Exit getEmpInfo  *****
    End If

End Function

Function DAO_recUpdateWO(ByVal won As String, _
                ByVal sn As String, _
                ByVal id As Date, _
                ByVal ib As String, _
                ByVal sd As Date, _
                ByVal tcd As Date, _
                ByVal combo_en As String, _
                ByVal combo_en2 As String, _
                ByVal combo_en3 As String, _
                ByVal combo_en4 As String, _
                ByVal wad As String, _
                ByVal tna As String, _
                ByVal tl As String, _
                ByVal cd As Date, _
                ByVal ds As Date) As Integer

    ' Check to ensure that the Customer DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim WORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Customer DB handle resource is secured.
    If DB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Customer Database not available"
      ' exist = False
      DAO_recUpdateWO = -1
      Exit Function
    End If

    ' *** Create a SQL query ***
    If DB_Trace Then
      MsgBox "***** Start recUpdateWO *****
    End If
    strQuery = "PARAMETERS [new_won] String; "
    strQuery = strQuery & "SELECT * FROM Work_Orders WHERE WON =
[new_won];"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_won") = won
    Set WORec = qdf.OpenRecordset(dbOpenDynaset)
    If WORec.EOF Then
      ' Record not found.
      If DB_Trace Then
        MsgBox "DAO:Record with WON=" & won & " NOT found"
      End If
      DAO_recUpdateWO = 0
    Else
      ' Record found. No need to search further
      If DB_Trace Then
        MsgBox "DAO:Record with WON=" & won & " found"
      End If
      ' Assign the new value to the IQ field of the record
      WORec.Edit
      WORec!sn = sn
      WORec!id = id
      WORec!ib = ib
      WORec!sd = sd
      WORec!tcd = tcd
      WORec!en = combo_en
      WORec!en2 = combo_en2
      WORec!en3 = combo_en3
      WORec!en4 = combo_en4
      WORec!wad = wad
      WORec!tna = tna
      WORec!tl = tl
      WORec!cd = cd
      WORec!ds = ds
      WORec.Update
      DAO_recUpdateWO = 0
    End If

    WORec.Close
    qdf.Close

    If DB_Trace Then
      MsgBox "***** Exit recEditWO  *****
    End If

End Function

Function DAO_recUpdateWO2(ByVal sn As String, _
                ByVal dmp As String, _
                ByVal ff As String, _
                ByVal mt As String) As Integer

    ' Check to ensure that the Customer DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim WORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Customer DB handle resource is secured.
    If DB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Customer Database not available"
      ' exist = False
      DAO_recUpdateWO2 = -1
      Exit Function
    End If

    ' *** Create a SQL query ***
    If DB_Trace Then
      MsgBox "***** Start recUpdateWO2 *****
    End If
    strQuery = "PARAMETERS [new_sn] String; "
    strQuery = strQuery & "SELECT * FROM Operational_Equipment WHERE
SN = [new_sn];"
    Set qdf = DB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_sn") = sn
    Set WORec = qdf.OpenRecordset(dbOpenDynaset)
    If WORec.EOF Then
      ' Record not found.
      If DB_Trace Then
        MsgBox "DAO:Record with SN=" & sn & " NOT found"
      End If
      DAO_recUpdateWO2 = 0
    Else
      ' Record found. No need to search further
      If DB_Trace Then
        MsgBox "DAO:Record with SN=" & sn & " found"
      End If
      ' Assign the new value to the IQ field of the record
      WORec.Edit
      WORec!dmp = dmp
      WORec!ff = ff
      WORec!mt = mt
      WORec.Update
      DAO_recUpdateWO2 = 0
    End If

    WORec.Close
    qdf.Close

    If DB_Trace Then
      MsgBox "***** Exit recEditWO2  *****
    End If
```

```
End Function

Function DAO_getWOStatus(ByVal wono As String, _
                         ByRef sdate As Date, _
                         ByRef tcdate As Date, _
                         ByRef idate As Date, _
                         ByRef late_start_ques As String, _
                         ByRef started_ques, _
                         ByRef late_comp_ques As String, _
                         ByRef comp_ques As String, _
                         ByRef cs As String) As Integer
  ' Check to ensure that the Inventory DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim Rec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Inventory DB handle resource is secured.
  If DB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : Inventory Database not available"
    ' exist = False
    DAO_getWOStatus = -1
    Exit Function
  End If

  ' *** Create a SQL query ***NEED TO FIX SQL
  If DB_Trace Then
    MsgBox "***** Start getWOStatus *****"
  End If
  strQuery = "PARAMETERS [new_wono] String; "
  strQuery = strQuery & "SELECT SD, TCD, ID, L_STARTED, STARTED,
L_COMPLETED, COMPLETED, CS FROM Work_Orders WHERE
WON=[new_wono];"
  Set qdf = DB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_wono") = wono
  Set Rec = qdf.OpenRecordset(dbOpenDynaset)
  If Rec.EOF Then
    ' Record not found.
    If DB_Trace Then
      MsgBox "DAO:Record with WON=" & wono & " NOT found"
    End If
    DAO_getWOStatus = -1
  Else
    ' Record found. No need to search further
    If DB_Trace Then
      MsgBox "DAO:Record with WON=" & wono & " found"
    End If
    ' Assign the new value to the IO field of the record
    sdate = Rec!sd
    tcdate = Rec!tcd
    idate = Rec!id
    late_start_ques = Rec!l_started
    started_ques = Rec!started
    late_comp_ques = Rec!l_completed
    comp_ques = Rec!completed
    cs = Rec!cs
    DAO_getWOStatus = 0
  End If
  Rec.Close
  qdf.Close

  If DB_Trace Then
    MsgBox "***** Exit getWOStatus *****"
  End If

End Function

Function DAO_getWKnm(ByVal etno As String, _
                     ByVal location As String, _
                     ByRef nwm As Date, _
                     ByRef snw As String) As Integer
  ' Check to ensure that the Inventory DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim Rec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Inventory DB handle resource is secured.
  If DB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : Inventory Database not available"
    ' exist = False
    DAO_getWKnm = -1
    Exit Function
  End If

  ' *** Create a SQL query
  If DB_Trace Then
    MsgBox "***** Start get Next Weekly Maintenance *****"
  End If

  strQuery = "PARAMETERS [new_etno] String, [new_location] String; "
  strQuery = strQuery & "SELECT NWM, SN From
OPERATIONAL_EQUIPMENT, LOCATIONS WHERE
LOCATION=[new_location] AND LNO=LOC AND EQTNO=[new_etno];"
  Set qdf = DB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_etno") = etno
  qdf.Parameters("new_location") = location

  Set Rec = qdf.OpenRecordset(dbOpenDynaset)
  If Rec.EOF Then
    ' Record not found.
    If DB_Trace Then
      MsgBox "DAO:Record with ETNO=" & etno & " NOT found"
    End If
    DAO_getWKnm = -1
  Else
    ' Record found. No need to search further
    If DB_Trace Then
      MsgBox "DAO:Record with ETNO=" & etno & " found"
    End If
    ' Assign the new value to the IO field of the record
    ' etno = Rec!eqtno
    'location = Rec!location
    nwm = Rec!nwm
    snw = Rec!sn
    DAO_getWKnm = 0
  End If
  Rec.Close
  qdf.Close

  If DB_Trace Then
    MsgBox "***** Exit getWKnm *****"
  End If

End Function

Function DAO_getMONnm(ByVal etno As String, _
                      ByVal location As String, _
                      ByRef nmm As Date, _
                      ByRef snm As String) As Integer
  ' Check to ensure that the Inventory DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim Rec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Inventory DB handle resource is secured.
  If DB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : Inventory Database not available"
    ' exist = False
    DAO_getMONnm = -1
    Exit Function
  End If

  ' *** Create a SQL query
  If DB_Trace Then
    MsgBox "***** Start get Next Monthly Maintenance *****"
  End If

  strQuery = "PARAMETERS [new_etno] String, [new_location] String; "
  strQuery = strQuery & "SELECT NMM, SN From
OPERATIONAL_EQUIPMENT, LOCATIONS WHERE
LOCATION=[new_location] AND LNO=LOC AND EQTNO=[new_etno];"
  Set qdf = DB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_etno") = etno
  qdf.Parameters("new_location") = location

  Set Rec = qdf.OpenRecordset(dbOpenDynaset)
  If Rec.EOF Then
    ' Record not found.
    If DB_Trace Then
      MsgBox "DAO:Record with ETNO=" & etno & " NOT found"
    End If
    DAO_getMONnm = -1
  Else
    ' Record found. No need to search further
    If DB_Trace Then
      MsgBox "DAO:Record with ETNO=" & etno & " found"
    End If
    ' Assign the new value to the IO field of the record
```

```
    nmm = Rec!nmm
    snm = Rec!sn
    DAO_getMONnm = 0
  End If
  Rec.Close
  qdf.Close

  If DB_Trace Then
    MsgBox "***** Exit getMONnm  *****
  End If

End Function

Function DAO_Update_Started(ByVal late_start_ques As String) As Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET L_STARTED = 'Yes' " _
    & "WHERE DS>SD;"

  dbs.Close

End Function

Function DAO_Update_Started2(ByVal started_ques As String) As Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET STARTED = 'Yes' " _
    & "WHERE DS>ID;"

  dbs.Close

End Function
Function DAO_Update_Started3(ByVal late_comp_ques As String) As
Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET L_COMPLETED = 'Yes' " _
    & "WHERE CD>TCD;"

  dbs.Close

End Function

Function DAO_Update_Started4(ByVal comp_ques As String) As Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET COMPLETED = 'Yes' " _
    & "WHERE CD>DS;"

  dbs.Close

End Function

Function DAO_cs1(ByVal cs As String) As Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET CS = 'In Progress' " _
    & "WHERE DS>SD;" _

  dbs.Close

End Function

Function DAO_cs2(ByVal cs As String) As Integer

  Dim dbs As Database
  Dim qdf As QueryDef

  ' Modify this line to include the path to Northwind
  ' on your computer.
  Set dbs = OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb")

  ' Change values in the ReportsTo field to 5 for all
  ' employee records that currently have ReportsTo
  ' values of 2.
  dbs.Execute "UPDATE Work_Orders " _
    & "SET CS = 'Completed' "
    & "WHERE CD>DS;" _

  dbs.Close

End Function
```

PODBHandler.cls

```
Option Explicit

Public PODB_AccSel As Integer

Public Function openPODB() As Integer
  ' This function opens the PO database.
  ' It assumes the name of the PO database is called "CMMS.mdb" and
  ' is located in the path "e:\training\prototype\database". Change the drive
  ' and the path if necessary.
  ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
  ' calls.
  ' The opened database will be stored in the "PODB" object handle
declared in
  ' the module "PODBStor".
  Dim rc As Integer

  If PODB_Trace Then
    MsgBox "PODB_DLL:Execute openPODB activity with Access Selection
" & PODB_AccSel
  End If

  ' Open the database
  Select Case PODB_AccSel
    Case 1
      ' Return result code for the activity.
      rc = DAO_openPODB
      If PODB_Trace Then
        MsgBox "PODB_DLL:DAO_openPODB result = " & rc
      End If
    Case Else
      MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
      ' Return result code for the activity.
      rc = -1
  End Select
  openPODB = rc
End Function
```

```
Public Function closePODB() As Integer
   ' This function closes the PO database.
   ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
   ' calls.
   ' The opened database object handle is stored in the variable declared in
   ' the corresponding module that handles the access methods.
   Dim rc As Integer

   If PODB_Trace Then
      MsgBox "PODB_DLL:Execute closePODB activity"
   End If

   ' Close the database
   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_closePODB
         If PODB_Trace Then
            MsgBox "PODB_DLL:DAO_closePODB result = " & rc
         End If
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   closePODB = rc
End Function

Public Function recExistPN(ByVal poid As Date, ByVal pn As String, ByRef
exist As Boolean) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the PO DB. Keys are the POID and ATNO.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If PODB_Trace Then
      MsgBox "PODB_DLL:Execute PN Uniqness activity "
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recExistPN(poid, pn, exist)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recExistPN = rc
End Function

Public Function recCreate(ByVal poid As Date, _
                  ByVal vna As String, _
                  ByVal vpn As String, _
                  ByVal pn As String, _
                  ByVal pqty As Integer, _
                  ByVal ist As Double) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the PO DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Create PO Record activity"
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recCreate(poid, vna, vpn, pn, pqty, ist)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recCreate = rc
End Function

Public Function getRecInfoPOID(ByVal poid As Date, _
                     ByRef vna As String, _
                     ByRef vpn As String, _
                     ByRef pn As String, _
                     ByRef pqty As Integer, _
                     ByRef ist As Double) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the PO DB.
   Dim rc As Integer

   If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Get Item Info activity"
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getRecInfoPOID(poid, vna, vpn, pn, pqty, ist)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getRecInfoPOID = rc
End Function

Public Function getRecInfoPN(ByRef poid As Date, _
                  ByRef vna As String, _
                  ByRef vpn As String, _
                  ByVal pn As String, _
                  ByRef pqty As Integer, _
                  ByRef ist As Double) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the PO DB.
   Dim rc As Integer

   If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Get Item Info activity"
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getRecInfoPN(poid, vna, vpn, pn, pqty, ist)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getRecInfoPN = rc
End Function

Public Function getETC(ByVal etno As String, ByRef etc As Double) As
Integer
   Dim rc As Integer
      If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Get Item Info activity"
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getETC(etno, etc)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getETC = rc
End Function

Public Function getETC2(ByVal etno2 As String, ByRef etc2 As Double) As
Integer
   Dim rc As Integer
      If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Get Item Info activity"
   End If

   Select Case PODB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getETC2(etno2, etc2)
      Case Else
         MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getETC2 = rc
End Function

Public Function getETC3(ByVal etno3 As String, ByRef etc3 As Double) As
Integer
   Dim rc As Integer
      If PODB_Trace Then
      MsgBox "PODB_DLL:Execute Get Item Info activity"
   End If
```

```
    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getETC3(etno3, etc3)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getETC3 = rc
End Function

Public Function getETC4(ByVal etno4 As String, ByRef etc4 As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getETC4(etno4, etc4)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getETC4 = rc
End Function

Public Function getATC(ByVal atno As String, ByRef atc As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getATC(atno, atc)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getATC = rc
End Function

Public Function getATC2(ByVal atno2 As String, ByRef atc2 As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getATC2(atno2, atc2)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getATC2 = rc
End Function

Public Function getATC3(ByVal atno3 As String, ByRef atc3 As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getATC3(atno3, atc3)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getATC3 = rc
End Function

Public Function getATC4(ByVal atno4 As String, ByRef atc4 As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getATC4(atno4, atc4)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getATC4 = rc
End Function

Public Function getSATC(ByVal satno As String, ByRef satc As Double) As
Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getSATC(satno, satc)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getSATC = rc
End Function

Public Function getSATC2(ByVal satno2 As String, ByRef satc2 As Double)
As Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getSATC2(satno2, satc2)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getSATC2 = rc
End Function

Public Function getSATC3(ByVal satno3 As String, ByRef satc3 As Double)
As Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
        Case 1
            ' Return result code for the activity.
            rc = DAO_getATC3(satno3, satc3)
        Case Else
            MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
            ' Return result code for the activity.
            rc = -1
    End Select
    getSATC3 = rc
End Function

Public Function getSATC4(ByVal satno4 As String, ByRef satc4 As Double)
As Integer
    Dim rc As Integer
        If PODB_Trace Then
        MsgBox "PODB_DLL:Execute Get Item Info activity"
    End If

    Select Case PODB_AccSel
```

```
    Case 1
      ' Return result code for the activity.
      rc = DAO_getSATC4(satno4, satc4)
    Case Else
      MsgBox "PODB_DLL:Other Access Selection chosen but not
supported yet"
      ' Return result code for the activity.
      rc = -1
  End Select
  getSATC4 = rc
End Function
Public Sub setPODBTrace(ByVal ToTrace As Boolean)

 PODB_Trace = ToTrace

End Sub
```

PODB_Jet_DAO.bas

```
Option Explicit

' This module is to implement the database access using the VB built in Jet
engine
' and the DAO model.
Private POWS_handle As Workspace
Private PODB_handle As Database


Function DAO_openPODB() As Integer
  ' Using the OpenDatabase Method in DAO
  ' It uses the OpenDatabase method to open the Microsoft Jet-based Sales
database.

  ' Create Microsoft Jet Workspace object.
  ' Set wrkJet = CreateWorkspace("", "admin", "", dbUseJet)
  Set POWS_handle = DBEngine.Workspaces(0)

  ' Open Database object from saved Microsoft Jet database
  ' for exclusive use (i.e. option = True).
  On Error GoTo openDbErr
    If PODB_Trace Then
      MsgBox "DAO:Open database"
    End If
    Set PODB_handle = POWS_handle.OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb", True)
    DAO_openPODB = 0
    Exit Function

openDbErr:
  MsgBox "DAO:Open Database error : " & Err.Number
  DAO_openPODB = -1
  Exit Function

End Function

Function DAO_closePODB() As Integer
  ' Use the Close Method in DAO
  PODB_handle.Close
  POWS_handle.Close
  DAO_closePODB = 0
End Function


Function DAO_recExistPN(ByVal poid As Date, ByVal pn As String, ByRef
exist As Boolean) As Integer
  ' Check to ensure that the Sales DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Sales DB handle resource is secured.
  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    exist = False
    DAO_recExistPN = -1
    Exit Function
  End If

  ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_recExistPN *****"
  End If
  strQuery = "PARAMETERS [new_poid] String, [new_atno] String; "
  strQuery = strQuery & "SELECT * FROM Purchase_Orders WHERE POID
= [new_poid] AND PN = [new_pn];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_poid") = poid
  qdf.Parameters("new_pn") = pn
  Set PORec = qdf.OpenRecordset(dbReadOnly)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with POID=" & poid & " PN=" & pn & " NOT
found"
    End If
    exist = False
    DAO_recExistPN = 0
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with POID=" & poid & " PN=" & pn & " found"
    End If
    exist = True
    DAO_recExistPN = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_recExistPN *****"
  End If

End Function

Public Function DAO_recCreate(ByVal poid As Date, _
                  ByVal vna As String, ByVal vpn As String, _
                  ByVal pn As String, _
                  ByVal pqty As Integer, _
                  ByVal ist As Double) As Integer
  ' Check to ensure that the Sales DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim PORec As Recordset

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Sales DB handle resource is secured.
  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    DAO_recCreate = -1
    Exit Function
  End If

  Set PORec = PODB_handle.OpenRecordset("Purchase_Orders",
dbOpenTable)

  If PODB_Trace Then
    MsgBox "DAO:Add new record"
  End If
  PORec.AddNew
  PORec!poid = poid
  PORec!vna = vna
  PORec!pn = pn
  PORec!pqty = pqty
  PORec!ist = ist
  PORec.Update
  PORec.Close

  DAO_recCreate = 0

End Function

Public Function DAO_getRecInfoPOID(ByVal poid As Date, _
                  ByRef vna As String, _
                  ByRef vpn As String, _
                  ByRef pn As String, _
                  ByRef pqty As Integer, _
                  ByRef ist As Double) As Integer
  ' Check to ensure that the Sales DB is opened for
  ' access.
  ' If the DB is not opened, fail the operation by
  ' displaying an error message box.
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  ' Ensure the database is availabile for access. The
  ' check should never fail because the coordinator should
  ' have ensure the Sales DB handle resource is secured.
  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : Sales Database not available"
    ' exist = False
    DAO_getRecInfoPOID = -1
    Exit Function
  End If
```

```
    ' *** Create a SQL query ***
    If PODB_Trace Then
       MsgBox ***** Start DAO_getRecInfoPOID *****
    End If
    strQuery = "PARAMETERS [new_poid] String; "
    strQuery = strQuery & "SELECT * FROM Purchase_Orders WHERE POID
= [new_poid];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_poid") = poid
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
       ' Record not found.
       If PODB_Trace Then
          MsgBox "DAO:Record with POID=" & poid & " NOT found"
       End If
       DAO_getRecInfoPOID = -1
    Else
       ' Record found. No need to search further
       If PODB_Trace Then
          MsgBox "DAO:Record with POID=" & poid & " found"
       End If
       ' Assign the new value to the IO field of the record
       vna = PORec!vna
       vpn = PORec!vpn
       pn = PORec!pn
       pqty = PORec!pqty
       ist = PORec!ist
       DAO_getRecInfoPOID = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
       MsgBox ***** Exit DAO_getRecInfoPOID  *****
    End If

End Function

Public Function DAO_getRecInfoPN(ByRef poid As Date, _
                         ByRef vna As String, _
                         ByRef vpn As String, _
                         ByVal pn As String, _
                         ByRef pqty As Integer, _
                         ByRef ist As Double) As Integer
    ' Check to ensure that the Sales DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is available for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Sales DB handle resource is secured.
    If PODB_handle Is Nothing Then
       MsgBox "DAO:Validation Fail : PO Database not available"
       ' exist = False
       DAO_getRecInfoPN = -1
       Exit Function
    End If

    ' *** Create a SQL query ***
    If PODB_Trace Then
       MsgBox ***** Start DAO_getRecInfoATNO *****
    End If
    strQuery = "PARAMETERS [new_pn] String; "
    strQuery = strQuery & "SELECT * FROM Purchase_Orders WHERE PN =
[new_pn];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_pn") = pn
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
       ' Record not found.
       If PODB_Trace Then
          MsgBox "DAO:Record with PN=" & pn & " NOT found"
       End If
       DAO_getRecInfoPN = -1
    Else
       ' Record found. No need to search further
       If PODB_Trace Then
          MsgBox "DAO:Record with PN=" & pn & " found"
       End If
       ' Assign the new value to the IO field of the record
       poid = PORec!poid
       vna = PORec!vna
       vpn = PORec!vpn
       pn = PORec!pn
       pqty = PORec!pqty
       ist = PORec!ist
       DAO_getRecInfoPN = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
       MsgBox ***** Exit DAO_getRecInfoPN  *****
    End If

End Function

Public Function DAO_getETC(ByVal etno As String, ByRef etc As Double) As
Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
       MsgBox "DAO:Validation Fail : PO Database not available"
       ' exist = False
       DAO_getETC = -1
       Exit Function
    End If
       ' *** Create a SQL query ***
    If PODB_Trace Then
       MsgBox ***** Start DAO_getETC *****
    End If
    strQuery = "PARAMETERS [new_etno] String; "
    strQuery = strQuery & "SELECT ETC FROM Equipment_Type WHERE
ETNO=[new_etno];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_etno") = etno
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
       ' Record not found.
       If PODB_Trace Then
          MsgBox "DAO:Record with ETNO=" & etno & " NOT found"
       End If
       DAO_getETC = -1
    Else
       ' Record found. No need to search further
       If PODB_Trace Then
          MsgBox "DAO:Record with ETNO=" & etno & " found"
       End If
       ' Assign the new value to the IO field of the record
       etc = PORec!etc
       DAO_getETC = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
       MsgBox ***** Exit DAO_getETC  *****
    End If

End Function

Public Function DAO_getETC2(ByVal etno2 As String, ByRef etc2 As
Double) As Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
       MsgBox "DAO:Validation Fail : PO Database not available"
       ' exist = False
       DAO_getETC2 = -1
       Exit Function
    End If
       ' *** Create a SQL query ***
    If PODB_Trace Then
       MsgBox ***** Start DAO_getETC2 *****
    End If
    strQuery = "PARAMETERS [new_etno2] String; "
    strQuery = strQuery & "SELECT ETC FROM Equipment_Type WHERE
ETNO=[new_etno2];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_etno2") = etno2
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
       ' Record not found.
       If PODB_Trace Then
          MsgBox "DAO:Record with ETNO2=" & etno2 & " NOT found"
       End If
       DAO_getETC2 = -1
    Else
       ' Record found. No need to search further
       If PODB_Trace Then
          MsgBox "DAO:Record with ETNO2=" & etno2 & " found"
       End If
       ' Assign the new value to the IO field of the record
```

```
    etc2 = PORec!etc
    DAO_getETC2 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getETC2 *****
  End If

End Function


Public Function DAO_getETC3(ByVal etno3 As String, ByRef etc3 As
Double) As Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getETC3 = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getETC3 *****
  End If
  strQuery = "PARAMETERS [new_etno3] String; "
  strQuery = strQuery & "SELECT ETC FROM Equipment_Type WHERE
ETNO=[new_etno3];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_etno3") = etno3
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with ETNO3=" & etno3 & " NOT found"
    End If
    DAO_getETC3 = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with ETNO3=" & etno3 & " found"
    End If
    ' Assign the new value to the IQ field of the record
    etc3 = PORec!etc
    DAO_getETC3 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getETC3 *****
  End If

End Function

Public Function DAO_getETC4(ByVal etno4 As String, ByRef etc4 As
Double) As Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getETC4 = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getETC4 *****
  End If
  strQuery = "PARAMETERS [new_etno4] String; "
  strQuery = strQuery & "SELECT ETC FROM Equipment_Type WHERE
ETNO=[new_etno4];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_etno4") = etno4
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with ETNO4=" & etno4 & " NOT found"
    End If
    DAO_getETC4 = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with ETNO4=" & etno4 & " found"
    End If
    ' Assign the new value to the IQ field of the record
    etc4 = PORec!etc
    DAO_getETC4 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getRecInfoPN  *****
  End If

End Function

Public Function DAO_getATC(ByVal atno As String, ByRef atc As Double) As
Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getATC = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getATC *****
  End If
  strQuery = "PARAMETERS [new_atno] String; "
  strQuery = strQuery & "SELECT ATC FROM Assembly_Type WHERE
ATNO=[new_atno];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_atno") = atno
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with ATNO=" & atno & " NOT found"
    End If
    DAO_getATC = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with ATNO=" & atno & " found"
    End If
    ' Assign the new value to the IQ field of the record
    atc = PORec!atc
    DAO_getATC = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getATC  *****
  End If

End Function

Public Function DAO_getATC2(ByVal atno2 As String, ByRef atc2 As
Double) As Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getATC2 = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getATC2 *****
  End If
  strQuery = "PARAMETERS [new_atno2] String; "
  strQuery = strQuery & "SELECT ATC FROM Assembly_Type WHERE
ATNO=[new_atno2];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_atno2") = atno2
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with ATNO2=" & atno2 & " NOT found"
    End If
    DAO_getATC2 = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
```

```
        MsgBox "DAO:Record with ATNO2=" & atno2 & " found"
      End If
      ' Assign the new value to the IQ field of the record
      atc2 = PORec!atc
      DAO_getATC2 = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
      MsgBox "***** Exit DAO_getATC2 *****
    End If

End Function

Public Function DAO_getATC3(ByVal atno3 As String, ByRef atc3 As
Double) As Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : PO Database not available"
      ' exist = False
      DAO_getATC3 = -1
      Exit Function
    End If
      ' *** Create a SQL query ***
    If PODB_Trace Then
      MsgBox "***** Start DAO_getATC3 *****
    End If
    strQuery = "PARAMETERS [new_atno3] String; "
    strQuery = strQuery & "SELECT ATC FROM Assembly_Type WHERE
ATNO=[new_atno3];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_atno3") = atno3
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
      ' Record not found.
      If PODB_Trace Then
        MsgBox "DAO:Record with ATNO3=" & atno3 & " NOT found"
      End If
      DAO_getATC3 = -1
    Else
      ' Record found. No need to search further
      If PODB_Trace Then
        MsgBox "DAO:Record with ATNO3=" & atno3 & " found"
      End If
      ' Assign the new value to the IQ field of the record
      atc3 = PORec!atc
      DAO_getATC3 = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
      MsgBox "***** Exit DAO_getATC3 *****
    End If

End Function

Public Function DAO_getATC4(ByVal atno4 As String, ByRef atc4 As
Double) As Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : PO Database not available"
      ' exist = False
      DAO_getATC4 = -1
      Exit Function
    End If
      ' *** Create a SQL query ***
    If PODB_Trace Then
      MsgBox "***** Start DAO_getATC4 *****
    End If
    strQuery = "PARAMETERS [new_atno4] String; "
    strQuery = strQuery & "SELECT ATC FROM Assembly_Type WHERE
ATNO=[new_atno4];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_atno4") = atno4
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
      ' Record not found.
      If PODB_Trace Then
        MsgBox "DAO:Record with ATNO4=" & atno4 & " NOT found"
      End If
      DAO_getATC4 = -1
    Else
      ' Record found. No need to search further
      If PODB_Trace Then
        MsgBox "DAO:Record with ATNO4=" & atno4 & " found"
      End If
      ' Assign the new value to the IQ field of the record
      atc4 = PORec!atc
      DAO_getATC4 = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
      MsgBox "***** Exit DAO_getRecInfoPN *****
    End If

End Function

Public Function DAO_getSATC(ByVal satno As String, ByRef satc As
Double) As Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : PO Database not available"
      ' exist = False
      DAO_getSATC = -1
      Exit Function
    End If
      ' *** Create a SQL query ***
    If PODB_Trace Then
      MsgBox "***** Start DAO_getSATC *****
    End If
    strQuery = "PARAMETERS [new_satno] String; "
    strQuery = strQuery & "SELECT SATC FROM Sub_Assembly_Type
WHERE SATNO=[new_satno];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_satno") = satno
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
      ' Record not found.
      If PODB_Trace Then
        MsgBox "DAO:Record with SATNO=" & satno & " NOT found"
      End If
      DAO_getSATC = -1
    Else
      ' Record found. No need to search further
      If PODB_Trace Then
        MsgBox "DAO:Record with SATNO=" & satno & " found"
      End If
      ' Assign the new value to the IQ field of the record
      satc = PORec!satc
      DAO_getSATC = 0
    End If
    PORec.Close
    qdf.Close

    If PODB_Trace Then
      MsgBox "***** Exit DAO_getSATC *****
    End If

End Function

Public Function DAO_getSATC2(ByVal satno2 As String, ByRef satc2 As
Double) As Integer
    Dim PORec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    If PODB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : PO Database not available"
      ' exist = False
      DAO_getSATC2 = -1
      Exit Function
    End If
      ' *** Create a SQL query ***
    If PODB_Trace Then
      MsgBox "***** Start DAO_getSATC2 *****
    End If
    strQuery = "PARAMETERS [new_satno2] String; "
    strQuery = strQuery & "SELECT SATC FROM Sub_Assembly_Type
WHERE SATNO=[new_satno2];"
    Set qdf = PODB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_satno2") = satno2
    Set PORec = qdf.OpenRecordset(dbOpenDynaset)
    If PORec.EOF Then
      ' Record not found.
      If PODB_Trace Then
        MsgBox "DAO:Record with SATNO2=" & satno2 & " NOT found"
      End If
      DAO_getSATC2 = -1
    Else
```

```
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with SATNO2=" & satno2 & " found"
    End If
    ' Assign the new value to the IQ field of the record
    satc2 = PORec!satc
    DAO_getSATC2 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getSATC2 *****
  End If

End Function

Public Function DAO_getSATC3(ByVal satno3 As String, ByRef satc3 As
Double) As Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getSATC3 = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getSATC3 *****
  End If
  strQuery = "PARAMETERS [new_satno3] String; "
  strQuery = strQuery & "SELECT SATC FROM Sub_Assembly_Type
WHERE SATNO=[new_satno3];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_satno3") = satno3
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with SATNO3=" & satno3 & " NOT found"
    End If
    DAO_getSATC3 = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with SATNO3=" & satno3 & " found"
    End If
    ' Assign the new value to the IQ field of the record
    satc3 = PORec!satc
    DAO_getSATC3 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getSATC3 *****
  End If

End Function

Public Function DAO_getSATC4(ByVal satno4 As String, ByRef satc4 As
Double) As Integer
  Dim PORec As Recordset
  Dim strQuery As String
  Dim qdf As QueryDef

  If PODB_handle Is Nothing Then
    MsgBox "DAO:Validation Fail : PO Database not available"
    ' exist = False
    DAO_getSATC4 = -1
    Exit Function
  End If
    ' *** Create a SQL query ***
  If PODB_Trace Then
    MsgBox "***** Start DAO_getSATC4 *****
  End If
  strQuery = "PARAMETERS [new_satno4] String; "
  strQuery = strQuery & "SELECT SATC FROM Sub_Assembly_Type
WHERE SATNO=[new_satno4];"
  Set qdf = PODB_handle.CreateQueryDef("", strQuery)
  qdf.Parameters("new_satno4") = satno4
  Set PORec = qdf.OpenRecordset(dbOpenDynaset)
  If PORec.EOF Then
    ' Record not found.
    If PODB_Trace Then
      MsgBox "DAO:Record with SATNO4=" & satno4 & " NOT found"
    End If
    DAO_getSATC4 = -1
  Else
    ' Record found. No need to search further
    If PODB_Trace Then
      MsgBox "DAO:Record with SATNO4=" & satno4 & " found"
    End If
    ' Assign the new value to the IQ field of the record
    satc4 = PORec!satc
    DAO_getSATC4 = 0
  End If
  PORec.Close
  qdf.Close

  If PODB_Trace Then
    MsgBox "***** Exit DAO_getRecInfoPN *****
  End If

End Function
```

VenDBHandler.cls

```
Option Explicit

Public VenDB_AccSel As Integer

Public Function openVenDB() As Integer
  ' This function opens the Vendor database.
  ' It assumes the name of the Vendor database is called "CMMS.mdb" and
  ' is located in the path "e:\training\prototype\database". Change the drive
  ' and the path if necessary.
  ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
  ' calls.
  ' The opened database will be stored in the "VenDB" object handle
declared in
  ' the module "VenDBStor".
  Dim rc As Integer

  If VenDB_Trace Then
    MsgBox "VenDB_DLL:Execute openVenDB activity with Access
Selection " & VenDB_AccSel
  End If

  ' Open the database
  Select Case VenDB_AccSel
    Case 1
      ' Return result code for the activity.
      rc = DAO_openVenDB
      If VenDB_Trace Then
        MsgBox "VenDB_DLL:DAO_openVenDB result = " & rc
      End If
    Case Else
      MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
      ' Return result code for the activity.
      rc = -1
  End Select
  openVenDB = rc
End Function


Public Function closeVenDB() As Integer
  ' This function closes the Vendor database.
  ' It will make use of the DAO provided by MS Access 7.0 and use the
ODBC driver
  ' calls.
  ' The opened database object handle is stored in the variable declared in
  ' the corresponding module that handles the access methods.
  Dim rc As Integer

  If VenDB_Trace Then
    MsgBox "VenDB_DLL:Execute closeVenDB activity"
  End If

  ' Close the database
  Select Case VenDB_AccSel
    Case 1
      ' Return result code for the activity.
      rc = DAO_closeVenDB
      If VenDB_Trace Then
        MsgBox "VenDB_DLL:DAO_closeVenDB result = " & rc
      End If
    Case Else
      MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
      ' Return result code for the activity.
      rc = -1
  End Select
  closeVenDB = rc
End Function
```

```
Public Function recExistVN(ByVal vna As String, ByRef exist As Boolean) As
Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Vendor DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If VenDB_Trace Then
      MsgBox "VenDB_DLL:Execute MPN Uniqness activity "
   End If

   Select Case VenDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recExistVN(vna, exist)
      Case Else
         MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recExistVN = rc
End Function


Public Function recCreate(ByVal vna As String, ByVal va As String, _
                  ByVal vpn As String, ByVal vno As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Vendor DB.
   ' Returns TRUE if such record exists, FALSE otherwise
   Dim rc As Integer

   If VenDB_Trace Then
      MsgBox "VenDB_DLL:Execute Create Ven Record activity"
   End If

   Select Case VenDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recCreate(vna, va, vpn, vno)
      Case Else
         MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recCreate = rc
End Function

Public Function getRecInfo(ByVal vna As String, ByRef va As String, ByRef
vpn As String) As Integer
   ' This function retrieve the whole record and return
   ' the field values that are existed in the Vendor DB.
   Dim rc As Integer

   If VenDB_Trace Then
      MsgBox "VenDB_DLL:Execute Get Item Info activity"
   End If

   Select Case VenDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_getRecInfo(vna, va, vpn)
      Case Else
         MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   getRecInfo = rc
End Function


Public Function recEditVPN(ByVal vpn As String, ByVal vna As String, ByVal
new_vpn As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Vendor DB.
   ' If the part exists, assign the new item quantity value
   Dim rc As Integer

   If VenDB_Trace Then
      MsgBox "VenDB_DLL:Execute Edit VPN activity"
   End If

   Select Case VenDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recEditVPN(vpn, vna, new_vpn)
      Case Else
         MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recEditVPN = rc
End Function

Public Function recEditVA(ByVal vna As String, ByVal va As String, ByVal
vpn As String) As Integer
   ' This function check if a record containing the fieldname
   ' is already existed in the Vendor DB.
   ' If the part exists, assign the new item sale price value
   Dim rc As Integer

   If VenDB_Trace Then
      MsgBox "VenDB_DLL:Execute Edit VA activity"
   End If

   Select Case VenDB_AccSel
      Case 1
         ' Return result code for the activity.
         rc = DAO_recEditVA(vna, va, vpn)
      Case Else
         MsgBox "VenDB_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   recEditVA = rc
End Function


Public Sub setVenDBTrace(ByVal ToTrace As Boolean)

 VenDB_Trace = ToTrace

End Sub
```

VenDB_Jet_DAO.bas

```
Option Explicit

' This module is to implement the database access using the VB built in Jet
engine
' and the DAO model.
Private VenWS_handle As Workspace
Private VenDB_handle As Database


Function DAO_openVenDB() As Integer
   ' Using the OpenDatabase Method in DAO
   ' It uses the OpenDatabase method to open the Microsoft Jet-based
Vendor database.

   ' Create Microsoft Jet Workspace object.
   ' Set wrkJet = CreateWorkspace("", "admin", "", dbUseJet)
   Set VenWS_handle = DBEngine.Workspaces(0)

   ' Open Database object from saved Microsoft Jet database
   ' for exclusive use (i.e. option = True).
   On Error GoTo openDbErr
      If VenDB_Trace Then
         MsgBox "DAO:Open database"
      End If
      Set VenDB_handle = VenWS_handle.OpenDatabase("c:\CMMS
Newest\salessystem.2\project\data\CMMS.mdb", True)
      DAO_openVenDB = 0
      Exit Function

openDbErr:
   MsgBox "DAO:Open Database error : " & Err.Number
   DAO_openVenDB = -1
   Exit Function

End Function


Function DAO_closeVenDB() As Integer
   ' Use the Close Method in DAO
   VenDB_handle.Close
   VenWS_handle.Close
   DAO_closeVenDB = 0
End Function


Function DAO_recExistVN(ByVal vna As String, ByRef exist As Boolean) As
Integer
   ' Check to ensure that the Vendor DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
```

```
    Dim VenRec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Vendor DB handle resource is secured.
    If VenDB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Vendor Database not available"
        exist = False
        DAO_recExistVN = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If VenDB_Trace Then
        MsgBox "***** Start recExistPN *****
    End If
    strQuery = "PARAMETERS [new_vna] String; "
    strQuery = strQuery & "SELECT * FROM Vendors WHERE VNA =
[new_vna];"
    Set qdf = VenDB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_vna") = vna
    Set VenRec = qdf.OpenRecordset(dbReadOnly)
    If VenRec.EOF Then
        ' Record not found.
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " NOT found"
        End If
        exist = False
        DAO_recExistVN = 0
    Else
        ' Record found. No need to search further
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " found"
        End If
        exist = True
        DAO_recExistVN = 0
    End If
    VenRec.Close
    qdf.Close

    If VenDB_Trace Then
        MsgBox "***** Exit recExistVN  *****
    End If

End Function

Function DAO_recCreate(ByVal vna As String, ByVal va As String, _
                ByVal vpn As String, ByVal vno As String) As Integer
    ' Check to ensure that the Vendor DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim VenRec As Recordset

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Vendor DB handle resource is secured.
    If VenDB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Vendor Database not available"
        DAO_recCreate = -1
        Exit Function
    End If

    Set VenRec = VenDB_handle.OpenRecordset("Vendors", dbOpenTable)

    If VenDB_Trace Then
        MsgBox "DAO:Add new record"
    End If
    VenRec.AddNew
    VenRec!vna = vna
    VenRec!va = va
    VenRec!vpn = vpn
    VenRec!vno = vno
    VenRec.Update
    VenRec.Close

    DAO_recCreate = 0
End Function


Function DAO_getRecInfo(ByVal vna As String, _
                ByRef va As String, ByRef vpn As String) As Integer

    ' Check to ensure that the Vendor DB is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim VenRec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Vendor DB handle resource is secured.
    If VenDB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Vendor Database not available"
        ' exist = False
        DAO_getRecInfo = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If VenDB_Trace Then
        MsgBox "***** Start getRecInfo *****
    End If
    strQuery = "PARAMETERS [new_vna] String; "
    strQuery = strQuery & "SELECT * FROM Vendors WHERE VNA =
[new_vna];"
    Set qdf = VenDB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_vna") = vna
    Set VenRec = qdf.OpenRecordset(dbOpenDynaset)
    If VenRec.EOF Then
        ' Record not found.
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " NOT found"
        End If
        DAO_getRecInfo = -1
    Else
        ' Record found. No need to search further
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " found"
        End If
        ' Assign the new value to the IQ field of the record
        va = VenRec!va
        vpn = VenRec!vpn
        DAO_getRecInfo = 0
    End If
    VenRec.Close
    qdf.Close

    If VenDB_Trace Then
        MsgBox "***** Exit getRecInfo  *****
    End If

End Function


Function DAO_recEditVPN(ByVal vpn As String, ByVal vna As String, ByVal
new_vpn As String) As Integer
    ' Check to ensure that the Customer DB  is opened for
    ' access.
    ' If the DB is not opened, fail the operation by
    ' displaying an error message box.
    Dim VenRec As Recordset
    Dim strQuery As String
    Dim qdf As QueryDef

    ' Ensure the database is availabile for access. The
    ' check should never fail because the coordinator should
    ' have ensure the Customer DB handle resource is secured.
    If VenDB_handle Is Nothing Then
        MsgBox "DAO:Validation Fail : Vendor Database not available"
        ' exist = False
        DAO_recEditVPN = -1
        Exit Function
    End If

    ' *** Create a SQL query ***
    If VenDB_Trace Then
        MsgBox "***** Start recEditCPN *****
    End If
    strQuery = "PARAMETERS [new_vna] String, [new_vpn] String; "
    strQuery = strQuery & "SELECT * FROM Vendors WHERE VFN =
[new_vfn] AND VNA = [new_vna];"
    Set qdf = VenDB_handle.CreateQueryDef("", strQuery)
    qdf.Parameters("new_vna") = vna
    qdf.Parameters("new_vpn") = vpn
    Set VenRec = qdf.OpenRecordset(dbOpenDynaset)
    If VenRec.EOF Then
        ' Record not found.
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " VPN=" & vpn & " NOT
found"
        End If
        DAO_recEditVPN = 0
    Else
        ' Record found. No need to search further
        If VenDB_Trace Then
            MsgBox "DAO:Record with VN=" & vna & " VPN=" & vpn & " found"
        End If
```

```
      ' Assign the new value to the IQ field of the record
      VenRec.Edit
      VenRec!vpn = new_vpn
      VenRec.Update
      DAO_recEditVPN = 0
   End If
   VenRec.Close
   qdf.Close

   If VenDB_Trace Then
      MsgBox "***** Exit recEditVPN  *****
   End If

End Function

Function DAO_recEditVA(ByVal vna As String, ByVal va As String, ByVal
vpn As String) As Integer

   ' Check to ensure that the Vendor DB is opened for
   ' access.
   ' If the DB is not opened, fail the operation by
   ' displaying an error message box.
   Dim VenRec As Recordset
   Dim strQuery As String
   Dim qdf As QueryDef

   ' Ensure the database is availabile for access. The
   ' check should never fail because the coordinator should
   ' have ensure the Vendor DB handle resource is secured.
   If VenDB_handle Is Nothing Then
      MsgBox "DAO:Validation Fail : Vendor Database not available"
      ' exist = False
      DAO_recEditVA = -1
      Exit Function
   End If

   ' *** Create a SQL query ***
   If VenDB_Trace Then
      MsgBox "***** Start recEditCPN *****
   End If
   strQuery = "PARAMETERS [new_vna] String, [new_vpn] String; "
   strQuery = strQuery & "SELECT * FROM Vendors WHERE VNA =
[new_vna] AND VPN = [new_vpn];"
   Set qdf = VenDB_handle.CreateQueryDef("", strQuery)
   qdf.Parameters("new_vna") = vna
   qdf.Parameters("new_vpn") = vpn
   Set VenRec = qdf.OpenRecordset(dbOpenDynaset)
   If VenRec.EOF Then
      ' Record not found.
      If VenDB_Trace Then
         MsgBox "DAO:Record with VN=" & vna & " VPN=" & vpn & " NOT
found"
      End If
      DAO_recEditVA = 0
   Else
      ' Record found. No need to search further
      If VenDB_Trace Then
         MsgBox "DAO:Record with VN=" & vna & " VPN=" & vpn & " found"
      End If
      ' Assign the new value to the IQ field of the record
      VenRec.Edit
      VenRec!vna = vna
      VenRec!va = va
      VenRec.Update
      DAO_recEditVA = 0
   End If

   VenRec.Close
   qdf.Close

   If VenDB_Trace Then
      MsgBox "***** Exit recEditCA  *****
   End If

End Function
```

POFormClass.cls

```
Option Explicit

Public POForm_AccSel As Integer

Public Function pofOpen() As Integer

   Dim rc As Integer

   If POForm_Trace Then
      MsgBox "POForm_DLL:Execute pofOpen activity with Access Selection
" & POForm_AccSel
   End If

   ' Open the database
   Select Case POForm_AccSel
      Case 1
         ' Return result code for the activity.
         rc = XL_pofOpen()
         If POForm_Trace Then
            MsgBox "POForm_DLL:XL_pofOpen result = " & rc
         End If
      Case Else
         MsgBox "POForm_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   pofOpen = rc

End Function

Public Function pofClose() As Integer

   Dim rc As Integer

   If POForm_Trace Then
      MsgBox "POForm_DLL:Execute pofClose activity with Access Selection
" & POForm_AccSel
   End If

   ' Open the database
   Select Case POForm_AccSel
      Case 1
         ' Return result code for the activity.
         rc = XL_pofClose()
         If POForm_Trace Then
            MsgBox "POForm_DLL:XL_pofClose result = " & rc
         End If
      Case Else
         MsgBox "POForm_DLL:Other Access Selection chosen but not
supported yet"
         ' Return result code for the activity.
         rc = -1
   End Select
   pofClose = rc

End Function

Public Function pofSubmit(ByVal poid As Date, ByVal vna As String, ByVal
va As String, ByVal vpn As String, _
                  ByVal atno As String, ByVal pqty As Integer, ByVal atc As
Double, ByRef ist As Double, _
                  ByVal atno2 As String, ByVal pqty2 As Integer, ByVal atc2 As
Double, ByRef ist2 As Double, _
                  ByVal atno3 As String, ByVal pqty3 As Integer, ByVal atc3 As
Double, ByRef ist3 As Double, _
                  ByVal atno4 As String, ByVal pqty4 As Integer, ByVal atc4 As
Double, ByRef ist4 As Double, _
                  ByVal etno As String, ByVal pqty5 As Integer, ByVal etc As
Double, ByRef ist5 As Double, _
                  ByVal etno2 As String, ByVal pqty6 As Integer, ByVal etc2 As
Double, ByRef ist6 As Double, _
                  ByVal etno3 As String, ByVal pqty7 As Integer, ByVal etc3 As
Double, ByRef ist7 As Double, _
                  ByVal etno4 As String, ByVal pqty8 As Integer, ByVal etc4 As
Double, ByRef ist8 As Double, _
                  ByVal satno As String, ByVal pqty9 As Integer, ByVal satc As
Double, ByRef ist9 As Double, _
                  ByVal satno2 As String, ByVal pqty10 As Integer, ByVal satc2
As Double, ByRef ist10 As Double, _
                  ByVal satno3 As String, ByVal pqty11 As Integer, ByVal satc3
As Double, ByRef ist11 As Double, _
                  ByVal satno4 As String, ByVal pqty12 As Integer, ByVal satc4
As Double, ByRef ist12 As Double, _
                  ByRef post As Double, ByRef pst As Double, ByRef fst As
Double, ByRef tpc As Double) As Integer

   Dim rc As Integer

   If POForm_Trace Then
      MsgBox "POForm_DLL:Execute pofSubmit activity with Access
Selection " & POForm_AccSel
   End If

   ' Open the database
   Select Case POForm_AccSel
      Case 1
         ' Return result code for the activity.
         rc = XL_pofSubmit(poid, vna, va, vpn, _
                  atno, pqty, atc, ist, _
                  atno2, pqty2, atc2, ist2, _
                  atno3, pqty3, atc3, ist3, _
                  atno4, pqty4, atc4, ist4, _
```

```
                etno, pqty5, etc, ist5, _
                etno2, pqty6, etc2, ist6, _
                etno3, pqty7, etc3, ist7, _
                etno4, pqty8, etc4, ist8, _
                satno, pqty9, satc, ist9, _
                satno2, pqty10, satc2, ist10, _
                satno3, pqty11, satc3, ist11, _
                satno4, pqty12, satc4, ist12, _
                post, pst, fst, tpc)

        If POForm_Trace Then
            MsgBox "POForm_DLL:XL_pofSubmit result = " & rc
        End If
    Case Else
        MsgBox "POForm_DLL:Other Access Selection chosen but not
supported yet"
        ' Return result code for the activity.
        rc = -1
  End Select
  pofSubmit = rc

End Function


Public Function pofPrint() As Integer

  Dim rc As Integer

  If POForm_Trace Then
    MsgBox "POForm_DLL:Execute pofPrint activity with Access Selection "
& POForm_AccSel
  End If

  ' Open the database
  Select Case POForm_AccSel
    Case 1
      ' Return result code for the activity.
      rc = XL_pofPrint()
      If POForm_Trace Then
        MsgBox "POForm_DLL:XL_pofPrint result = " & rc
      End If
    Case Else
      MsgBox "POForm_DLL:Other Access Selection chosen but not
supported yet"
      ' Return result code for the activity.
      rc = -1
  End Select
  pofPrint = rc

End Function


Public Sub setPOFormTrace(ByVal ToTrace As Boolean)

 POForm_Trace = ToTrace

End Sub
```

XL_POForm.bas

```
Option Explicit

Private pofApp  As Object
Private pofSheet As Object

Function XL_pofOpen() As Integer
  ' This function will open the PO Invoice Form
  If POForm_Trace Then
    MsgBox "Set Excel Application Object"
  End If
  On Error GoTo pofErr1
  Set pofApp = CreateObject("Excel.Application.5")
  ' pofApp.Visible = True

  If POForm_Trace Then
    MsgBox "Open file Invoice.xls"
  End If
  On Error GoTo pofErr1
  pofApp.WorkBooks.Open filename:="c:\cmms
newest\salessystem.2\project\data\Purchase_Order.xls"

  If POForm_Trace Then
    MsgBox "Set Excel Worksheet Object"
  End If
  On Error GoTo pofErr2
  Set pofSheet = pofApp.ActiveSheet

  If POForm_Trace Then
    MsgBox "The name of the pofsheet object is " & pofSheet.Name
  End If
  XL_pofOpen = 0
  Exit Function

pofErr1:
  Set pofSheet = Nothing
pofErr2:
  Set pofApp = Nothing

  XL_pofOpen = -1

End Function

Function XL_pofClose() As Integer

  ' Check to ensure that the worksheet object is not NULL.
  If pofApp Is Nothing Then
    MsgBox "Error:Excel Application object not available"
    Exit Function
  End If

  If pofSheet Is Nothing Then
    MsgBox "Error:Worksheet object not available"
    Exit Function
  End If

  If POForm_Trace Then
    MsgBox "Close the worksheet"
  End If
  pofApp.ActiveWorkbook.Close saveChanges:=False

  If POForm_Trace Then
    MsgBox "Exit Excel Application"
  End If
  pofApp.Quit

  Set pofSheet = Nothing
  Set pofApp = Nothing

  XL_pofClose = 0

End Function

Function XL_pofSubmit(ByVal poid As Date, ByVal vna As String, ByVal va
As String, ByVal vpn As String, _
                ByVal atno As String, ByVal pqty As Integer, ByVal atc As
Double, ByRef ist As Double, _
                ByVal atno2 As String, ByVal pqty2 As Integer, ByVal atc2 As
Double, ByRef ist2 As Double, _
                ByVal atno3 As String, ByVal pqty3 As Integer, ByVal atc3 As
Double, ByRef ist3 As Double, _
                ByVal atno4 As String, ByVal pqty4 As Integer, ByVal atc4 As
Double, ByRef ist4 As Double, _
                ByVal etno As String, ByVal pqty5 As Integer, ByVal etc As
Double, ByRef ist5 As Double, _
                ByVal etno2 As String, ByVal pqty6 As Integer, ByVal etc2 As
Double, ByRef ist6 As Double, _
                ByVal etno3 As String, ByVal pqty7 As Integer, ByVal etc3 As
Double, ByRef ist7 As Double, _
                ByVal etno4 As String, ByVal pqty8 As Integer, ByVal etc4 As
Double, ByRef ist8 As Double, _
                ByVal satno As String, ByVal pqty9 As Integer, ByVal satc As
Double, ByRef ist9 As Double, _
                ByVal satno2 As String, ByVal pqty10 As Integer, ByVal satc2
As Double, ByRef ist10 As Double, _
                ByVal satno3 As String, ByVal pqty11 As Integer, ByVal satc3
As Double, ByRef ist11 As Double, _
                ByVal satno4 As String, ByVal pqty12 As Integer, ByVal satc4
As Double, ByRef ist12 As Double, _
                ByRef post As Double, ByRef pst As Double, ByRef fst As
Double, ByRef tpc As Double) As Integer

  Dim strFormula As String

  ' Check to ensure that the worksheet object is not NULL.
  If pofApp Is Nothing Then
    MsgBox "Error:Excel Application object not available"
    Exit Function
  End If

  If pofSheet Is Nothing Then
    MsgBox "Error:Worksheet object not available"
    Exit Function
  End If


  ' Assign the values to the cells.
  ' Let the worksheet perform the calculations.
  pofSheet.Range("C4").Value = poid
  pofSheet.Range("C6").Value = vna
  pofSheet.Range("C7").Value = vpn
  pofSheet.Range("C8").Value = va
```

```
    strFormula = "=$E$24*" & 0.08
    pofSheet.Range("E25").Formula = strFormula
    strFormula = "=$E$24*" & 0.07
    pofSheet.Range("E26").Formula = strFormula


    pofSheet.Range("B15").Value = atno
    pofSheet.Range("C15").Value = pqty5
    pofSheet.Range("D15").Value = atc
    ist5 = pofSheet.Range("E15").Value

    pofSheet.Range("B16").Value = atno2
    pofSheet.Range("C16").Value = pqty6
    pofSheet.Range("D16").Value = atc2
    ist6 = pofSheet.Range("E16").Value

    pofSheet.Range("B17").Value = atno3
    pofSheet.Range("C17").Value = pqty7
    pofSheet.Range("D17").Value = atc3
    ist7 = pofSheet.Range("E17").Value

    pofSheet.Range("B18").Value = atno4
    pofSheet.Range("C18").Value = pqty8
    pofSheet.Range("D18").Value = atc4
    ist8 = pofSheet.Range("E18").Value

    pofSheet.Range("B11").Value = etno
    pofSheet.Range("C11").Value = pqty
    pofSheet.Range("D11").Value = etc
    ist = pofSheet.Range("E11").Value

    pofSheet.Range("B12").Value = etno2
    pofSheet.Range("C12").Value = pqty2
    pofSheet.Range("D12").Value = etc2
    ist2 = pofSheet.Range("E12").Value

    pofSheet.Range("B13").Value = etno3
    pofSheet.Range("C13").Value = pqty3
    pofSheet.Range("D13").Value = etc3
    ist3 = pofSheet.Range("E13").Value

    pofSheet.Range("B14").Value = etno4
    pofSheet.Range("C14").Value = pqty4
    pofSheet.Range("D14").Value = etc4
    ist4 = pofSheet.Range("E14").Value

    pofSheet.Range("B19").Value = satno
    pofSheet.Range("C19").Value = pqty9
    pofSheet.Range("D19").Value = satc
    ist9 = pofSheet.Range("E19").Value

    pofSheet.Range("B20").Value = satno2
    pofSheet.Range("C20").Value = pqty10
    pofSheet.Range("D20").Value = satc2
    ist10 = pofSheet.Range("E20").Value

    pofSheet.Range("B21").Value = satno3
    pofSheet.Range("C21").Value = pqty11
    pofSheet.Range("D21").Value = satc3
    ist11 = pofSheet.Range("E21").Value

    pofSheet.Range("B22").Value = satno4
    pofSheet.Range("C22").Value = pqty12
    pofSheet.Range("D22").Value = satc4
    ist12 = pofSheet.Range("E22").Value


    post = pofSheet.Range("E24").Value
    pst = pofSheet.Range("E25").Value
    fst = pofSheet.Range("E26").Value
    tpc = pofSheet.Range("E27").Value

    XL_pofSubmit = 0

End Function


Function XL_pofPrint() As Integer

    If POForm_Trace Then
        MsgBox "Print the Invoice"
    End If
    pofSheet.PrintOut

    XL_pofPrint = 0

End Function
```

13. Annex F - Local UI Code

ADDInventoryItem.cpp

```
// AddInventoryItem.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "AddInventoryItem.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CAddInventoryItem a_inv_i_pkg;
typedef int (*AddInvItemProc)(
Cstring m_mptd,
Cstring m_mpn,
Cstring m_sn,
Cstring m_atno,
Cstring m_etno,
Cstring m_satno,
short m_qr,
Cstring m_nur,
short m_iqty,
Cstring m_isl,
short m_msl
                );

int invoke_a_inv_i()
{
int rc = -1;
AddInvItemProc
lpfnDllFunc1;
// Function pointer

TRACE("Invoke_a_inv_i\n");
if (hinstDLL != NULL)
{
lpfnDllFunc1 = (AddInvItemProc)GetProcAddress(hinstDLL,
"Req_add_inventory_item");
if (!lpfnDllFunc1)
                {
// handle the error
//AfxFreeLibrary(hinstDLL);
TRACE("Invoke_a_inv_i: GetProcAddress fail\n");
                }
else
                {
// call the function
rc = lpfnDllFunc1
(a_inv_i_pkg.m_mptd , a_inv_i_pkg.m_mpn , a_inv_i_pkg.m_sn ,
a_inv_i_pkg.m_atno, a_inv_i_pkg.m_etno , a_inv_i_pkg.m_satno ,
a_inv_i_pkg.m_qr , a_inv_i_pkg.m_nur , a_inv_i_pkg.m_iqty ,
a_inv_i_pkg.m_isl , a_inv_i_pkg.m_msl  );
TRACE("Invoke_a_inv_i: rc=%d\n", rc);
//AfxFreeLibrary(hinstDLL);
                }
}
return rc;
}

/////////////////////////////////////////////////////////////////////////////
// CAddInventoryItem dialog

CAddInventoryItem::CAddInventoryItem(CWnd* pParent /*=NULL*/)
: CDialog(CAddInventoryItem::IDD, pParent)
{
//{{AFX_DATA_INIT(CAddInventoryItem)
m_mptd = _T("");
m_mpn = _T("");
m_sn = _T("");
m_atno = _T("");
m_etno = _T("");
m_satno = _T("");
m_nur = _T("");
m_isl = _T("");
m_iqty = 0;
m_qr = 0;
m_msl = 0;
//}}AFX_DATA_INIT
}

void CAddInventoryItem::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAddInventoryItem)
DDX_Text(pDX, IDC_EDIT_MPTD, m_mptd);
DDV_MaxChars(pDX, m_mptd, 80);
DDX_Text(pDX, IDC_EDIT_MPN, m_mpn);
DDV_MaxChars(pDX, m_mpn, 25);
                DDX_Text(pDX, IDC_EDIT_SN, m_sn);
                DDV_MaxChars(pDX, m_sn, 25);
                DDX_Text(pDX, IDC_EDIT_ATNO, m_atno);
                DDV_MaxChars(pDX, m_atno, 25);
                DDX_Text(pDX, IDC_EDIT_ETNO, m_etno);
                DDV_MaxChars(pDX, m_etno, 25);
                DDX_Text(pDX, IDC_EDIT_SATNO, m_satno);
                DDV_MaxChars(pDX, m_satno, 25);
                DDX_Text(pDX, IDC_EDIT_NUR, m_nur);
                DDX_Text(pDX, IDC_EDIT_ISL, m_isl);
                DDX_Text(pDX, IDC_EDIT_IQTY, m_iqty);
                DDX_Text(pDX, IDC_EDIT_QR, m_qr);
                DDX_Text(pDX, IDC_EDIT_MSL, m_msl);
                //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CAddInventoryItem, CDialog)
//{{AFX_MSG_MAP(CAddInventoryItem)
// NOTE: the ClassWizard will add message map macros here
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAddInventoryItem message handlers
```

AddInventoryItem.h

```
#if
!defined(AFX_ADDINVENTORYITEM_H__096E4BC1_8F1A_11D1_9946_00
6097CC972B__INCLUDED_)
#define
AFX_ADDINVENTORYITEM_H__096E4BC1_8F1A_11D1_9946_006097CC
972B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// AddInventoryItem.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CAddInventoryItem dialog

class CAddInventoryItem : public CDialog
{
// Construction
public:
CAddInventoryItem(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
                //{{AFX_DATA(CAddInventoryItem)
                enum { IDD = IDD_ADD_INVENTORY_ITEM };
                CString         m_mptd;
                CString         m_mpn;
                CString         m_sn;
                CString         m_atno;
                CString         m_etno;
                CString         m_satno;
                short           m_qr;
                CString         m_nur;
                short           m_iqty;
                CString         m_isl;
                short           m_msl;
                //}}AFX_DATA


// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAddInventoryItem)
protected:
virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
// Generated message map functions
//{{AFX_MSG(CAddInventoryItem)
// NOTE: the ClassWizard will add member functions here
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

extern CAddInventoryItem a_inv_i_pkg;
extern int invoke_a_inv_i();

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.
```

```
#endif //
!defined(AFX_ADDINVENTORYITEM_H__096E4BC1_8F1A_11D1_9946_00
6097CC972B__INCLUDED_)
```

CreatePurchaseOrder.cpp

```
// CreatePurchaseOrder.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "CreatePurchaseOrder.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define C_SUBMIT                0
#define C_CONFIRMPRINT          1

CCreatePurchaseOrder create_PO_pkg;

/////////////////////////////////////////////////////////////////////////////
// CCreatePurchaseOrder dialog


CCreatePurchaseOrder::CCreatePurchaseOrder(CWnd* pParent /*=NULL*/)
: CDialog(CCreatePurchaseOrder::IDD, pParent)
{
//{{AFX_DATA_INIT(CCreatePurchaseOrder)
        m_atc = 0.0;
        m_atc2 = 0.0;
        m_atc3 = 0.0;
        m_atc4 = 0.0;
        m_atno = _T("");
        m_atno2 = _T("");
        m_atno3 = _T("");
        m_atno4 = _T("");
        m_etc2 = 0.0;
        m_etc3 = 0.0;
        m_etc4 = 0.0;
        m_etno = _T("");
        m_etno2 = _T("");
        m_etno3 = _T("");
        m_etno4 = _T("");
        m_fst = 0.0;
        m_ist = 0.0;
        m_ist10 = 0.0;
        m_ist11 = 0.0;
        m_ist12 = 0.0;
        m_ist2 = 0.0;
        m_ist3 = 0.0;
        m_ist4 = 0.0;
        m_ist5 = 0.0;
        m_ist6 = 0.0;
        m_ist7 = 0.0;
        m_ist8 = 0.0;
        m_ist9 = 0.0;
        m_poid = time_t(0);
        m_pon = 0;
        m_post = 0.0;
        m_pqty = 0;
        m_pqty10 = 0;
        m_pqty11 = 0;
        m_pqty12 = 0;
        m_pqty2 = 0;
        m_pqty3 = 0;
        m_pqty4 = 0;
        m_pqty5 = 0;
        m_pqty6 = 0;
        m_pqty7 = 0;
        m_pqty8 = 0;
        m_pqty9 = 0;
        m_pst = 0.0;
        m_satno = _T("");
        m_satno2 = _T("");
        m_satno3 = _T("");
        m_satno4 = _T("");
        m_satc = 0.0;
        m_satc2 = 0.0;
        m_satc3 = 0.0;
        m_satc4 = 0.0;
        m_tpc = 0.0;
        m_va = _T("");
        m_vna = _T("");
        m_vno = _T("");
        m_vpn = _T("");
        m_status = _T("");
        m_etc = 0.0;
//}}AFX_DATA_INIT
}


void CCreatePurchaseOrder::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CCreatePurchaseOrder)
        DDX_Text(pDX, IDC_EDIT_ATC, m_atc);
        DDX_Text(pDX, IDC_EDIT_ATC2, m_atc2);
        DDX_Text(pDX, IDC_EDIT_ATC3, m_atc3);
        DDX_Text(pDX, IDC_EDIT_ATC4, m_atc4);
        DDX_Text(pDX, IDC_EDIT_ATNO, m_atno);
        DDX_Text(pDX, IDC_EDIT_ATNO2, m_atno2);
        DDX_Text(pDX, IDC_EDIT_ATNO3, m_atno3);
        DDX_Text(pDX, IDC_EDIT_ATNO4, m_atno4);
        DDX_Text(pDX, IDC_EDIT_ETC2, m_etc2);
        DDX_Text(pDX, IDC_EDIT_ETC3, m_etc3);
        DDX_Text(pDX, IDC_EDIT_ETC4, m_etc4);
        DDX_Text(pDX, IDC_EDIT_ETNO, m_etno);
        DDX_Text(pDX, IDC_EDIT_ETNO2, m_etno2);
        DDX_Text(pDX, IDC_EDIT_ETNO3, m_etno3);
        DDX_Text(pDX, IDC_EDIT_ETNO4, m_etno4);
        DDX_Text(pDX, IDC_EDIT_FST, m_fst);
        DDX_Text(pDX, IDC_EDIT_IST, m_ist);
        DDX_Text(pDX, IDC_EDIT_IST10, m_ist10);
        DDX_Text(pDX, IDC_EDIT_IST11, m_ist11);
        DDX_Text(pDX, IDC_EDIT_IST12, m_ist12);
        DDX_Text(pDX, IDC_EDIT_IST2, m_ist2);
        DDX_Text(pDX, IDC_EDIT_IST3, m_ist3);
        DDX_Text(pDX, IDC_EDIT_IST4, m_ist4);
        DDX_Text(pDX, IDC_EDIT_IST5, m_ist5);
        DDX_Text(pDX, IDC_EDIT_IST6, m_ist6);
        DDX_Text(pDX, IDC_EDIT_IST7, m_ist7);
        DDX_Text(pDX, IDC_EDIT_IST8, m_ist8);
        DDX_Text(pDX, IDC_EDIT_IST9, m_ist9);
        DDX_Text(pDX, IDC_EDIT_POID, m_poid);
        DDX_Text(pDX, IDC_EDIT_PON, m_pon);
        DDX_Text(pDX, IDC_EDIT_POST, m_post);
        DDX_Text(pDX, IDC_EDIT_PQTY, m_pqty);
        DDX_Text(pDX, IDC_EDIT_PQTY10, m_pqty10);
        DDX_Text(pDX, IDC_EDIT_PQTY11, m_pqty11);
        DDX_Text(pDX, IDC_EDIT_PQTY12, m_pqty12);
        DDX_Text(pDX, IDC_EDIT_PQTY2, m_pqty2);
        DDX_Text(pDX, IDC_EDIT_PQTY3, m_pqty3);
        DDX_Text(pDX, IDC_EDIT_PQTY4, m_pqty4);
        DDX_Text(pDX, IDC_EDIT_PQTY5, m_pqty5);
        DDX_Text(pDX, IDC_EDIT_PQTY6, m_pqty6);
        DDX_Text(pDX, IDC_EDIT_PQTY7, m_pqty7);
        DDX_Text(pDX, IDC_EDIT_PQTY8, m_pqty8);
        DDX_Text(pDX, IDC_EDIT_PQTY9, m_pqty9);
        DDX_Text(pDX, IDC_EDIT_PST, m_pst);
        DDX_Text(pDX, IDC_EDIT_S_ATNO, m_satno);
        DDX_Text(pDX, IDC_EDIT_S_ATNO2, m_satno2);
        DDX_Text(pDX, IDC_EDIT_S_ATNO3, m_satno3);
        DDX_Text(pDX, IDC_EDIT_S_ATNO4, m_satno4);
        DDX_Text(pDX, IDC_EDIT_SATC, m_satc);
        DDX_Text(pDX, IDC_EDIT_SATC2, m_satc2);
        DDX_Text(pDX, IDC_EDIT_SATC3, m_satc3);
        DDX_Text(pDX, IDC_EDIT_SATC4, m_satc4);
        DDX_Text(pDX, IDC_EDIT_TPC, m_tpc);
        DDX_Text(pDX, IDC_EDIT_VENDOR_ADDRESS, m_va);
        DDX_Text(pDX, IDC_EDIT_VENDOR_NAME, m_vna);
        DDX_Text(pDX, IDC_EDIT_VENDOR_NO, m_vno);
        DDX_Text(pDX, IDC_EDIT_VPN, m_vpn);
        DDX_Text(pDX, IDC_STATUS, m_status);
        DDX_Text(pDX, IDC_EDIT_ETC, m_etc);
        //}}AFX_DATA_MAP
}

// Implement the override function to initialize the PO entry form fields.
BOOL CCreatePurchaseOrder::OnInitDialog( )
{
        COleDateTime timeNow;
        BOOL rc;

        rc = CDialog::OnInitDialog();
        timeNow = COleDateTime::GetCurrentTime();
        m_poid.SetDate(timeNow.GetYear(), timeNow.GetMonth(),
timeNow.GetDay());
        create_PO_pkg.UpdateData(FALSE);
        return(TRUE);
}

BEGIN_MESSAGE_MAP(CCreatePurchaseOrder, CDialog)
        //{{AFX_MSG_MAP(CCreatePurchaseOrder)
        ON_BN_CLICKED(IDSUBMIT, OnSubmit)
        ON_BN_CLICKED(IDCONFIRMPRINT, OnConfirmPrint)
        ON_BN_CLICKED(IDCLEAR, OnClear)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////
// CCreatePurchaseOrder message handlers Utilities
//

void init_create_PO_pkg()
{
        COleDateTime timeNow;

        create_PO_pkg.m_atc = 0.0;
        create_PO_pkg.m_atc2 = 0.0;
        create_PO_pkg.m_atc3 = 0.0;
        create_PO_pkg.m_atc4 = 0.0;
        create_PO_pkg.m_atno = _T("");
        create_PO_pkg.m_atno2 = _T("");
        create_PO_pkg.m_atno3 = _T("");
        create_PO_pkg.m_atno4 = _T("");
        create_PO_pkg.m_etc = 0.0;
        create_PO_pkg.m_etc2 = 0.0;
        create_PO_pkg.m_etc3 = 0.0;
        create_PO_pkg.m_etc4 = 0.0;
        create_PO_pkg.m_etno = _T("");
        create_PO_pkg.m_etno2 = _T("");
        create_PO_pkg.m_etno3 = _T("");
        create_PO_pkg.m_etno4 = _T("");
        create_PO_pkg.m_fst = 0.0;
        create_PO_pkg.m_ist = 0.0;
        create_PO_pkg.m_ist10 = 0.0;
        create_PO_pkg.m_ist11 = 0.0;
        create_PO_pkg.m_ist12 = 0.0;
        create_PO_pkg.m_ist2 = 0.0;
        create_PO_pkg.m_ist3 = 0.0;
        create_PO_pkg.m_ist4 = 0.0;
        create_PO_pkg.m_ist5 = 0.0;
        create_PO_pkg.m_ist6 = 0.0;
        create_PO_pkg.m_ist7 = 0.0;
        create_PO_pkg.m_ist8 = 0.0;
        create_PO_pkg.m_ist9 = 0.0;
        create_PO_pkg.m_poid = time_t(0);
        create_PO_pkg.m_pon = 0;
        create_PO_pkg.m_post = 0.0;
        create_PO_pkg.m_pqty = 0;
        create_PO_pkg.m_pqty10 = 0;
        create_PO_pkg.m_pqty11 = 0;
        create_PO_pkg.m_pqty12 = 0;
        create_PO_pkg.m_pqty2 = 0;
        create_PO_pkg.m_pqty3 = 0;
        create_PO_pkg.m_pqty4 = 0;
        create_PO_pkg.m_pqty5 = 0;
        create_PO_pkg.m_pqty6 = 0;
        create_PO_pkg.m_pqty7 = 0;
        create_PO_pkg.m_pqty8 = 0;
        create_PO_pkg.m_pqty9 = 0;
        create_PO_pkg.m_pst = 0.0;
        create_PO_pkg.m_satno = _T("");
        create_PO_pkg.m_satno2 = _T("");
        create_PO_pkg.m_satno3 = _T("");
        create_PO_pkg.m_satno4 = _T("");
        create_PO_pkg.m_satc = 0.0;
        create_PO_pkg.m_satc2 = 0.0;
        create_PO_pkg.m_satc3 = 0.0;
        create_PO_pkg.m_satc4 = 0.0;
        create_PO_pkg.m_tpc = 0.0;
        create_PO_pkg.m_va = _T("");
        create_PO_pkg.m_vna = _T("");
        create_PO_pkg.m_vno = _T("");
        create_PO_pkg.m_vpn = _T("");
        create_PO_pkg.m_status = _T("");

        timeNow = COleDateTime::GetCurrentTime();
        create_PO_pkg.m_poid.SetDate(timeNow.GetYear(),
timeNow.GetMonth(), timeNow.GetDay());
}

typedef int (*CreatePOProcessProc)(

                int                     po_select,

                double          *m_atc,
                double          *m_atc2,
                double          *m_atc3,
                double          *m_atc4,
                CString         m_atno,
                CString         m_atno2,
                CString         m_atno3,
                CString         m_atno4,
                double          *m_etc,
                double          *m_etc2,
                double          *m_etc3,
                double          *m_etc4,
                CString         m_etno,
                CString         m_etno2,
                CString         m_etno3,
                CString         m_etno4,
                double          *m_fst,
                double          *m_ist,
                double          *m_ist10,
                double          *m_ist11,
                double          *m_ist12,
                double          *m_ist2,
                double          *m_ist3,
                double          *m_ist4,
                double          *m_ist5,
                double          *m_ist6,
                double          *m_ist7,
                double          *m_ist8,
                double          *m_ist9,

                COleDateTime            m_poid,
                int             m_pon,
                double          *m_post,
                short           m_pqty,
                short           m_pqty10,
                short           m_pqty11,
                short           m_pqty12,
                short           m_pqty2,
                short           m_pqty3,

                short           m_pqty4,
                short           m_pqty5,
                short           m_pqty6,
                short           m_pqty7,
                short           m_pqty8,
                short           m_pqty9,
                double          *m_pst,
                CString         m_satno,
                CString         m_satno2,
                CString         m_satno3,
                CString         m_satno4,
                double          *m_satc,
                double          *m_satc2,
                double          *m_satc3,
                double          *m_satc4,
                double          *m_tpc,
                CString         *m_va,
                CString         m_vna,
                CString         m_vno,
                CString         *m_vpn,
                CString& m_status

                );

int create_PO_process(int po_select)
{
int                                                     rc = 0;
CreatePOProcessProc
lpfnDllFunc1;
// Function pointer
Cstring strMsgText;
TRACE("create_PO_process\n");
if (ui_msg) {
AfxMessageBox("create_PO_process");
        }

if (hinstDLL != NULL)
        {
lpfnDllFunc1 = (CreatePOProcessProc)GetProcAddress(hinstDLL,
"Req_create_po");
if (!lpfnDllFunc1)
        {
// handle the error
TRACE("create_PO_process: GetProcAddress fail\n");
AfxMessageBox("Error:Coord Interface not Found");
        }
        else
        {
// call the function
if (ui_msg) {strMsgText.Format("create_PO_process: Call coord DLL VN=%s
%s VPN=%s ETNO=%s PQTY=%d ETC=%f IST=%f ",
create_PO_pkg.m_vna, create_PO_pkg.m_vpn, create_PO_pkg.m_etno,
create_PO_pkg.m_pqty,create_PO_pkg.m_etc, create_PO_pkg.m_ist);

AfxMessageBox(strMsgText);
                }


rc = lpfnDllFunc1
(po_select,

        &create_PO_pkg.m_atc,
        &create_PO_pkg.m_atc2,
        &create_PO_pkg.m_atc3,
```

```
                &create_PO_pkg.m_atc4,
                create_PO_pkg.m_atno,
                create_PO_pkg.m_atno2,
                create_PO_pkg.m_atno3,
                create_PO_pkg.m_atno4,
                &create_PO_pkg.m_etc,
                &create_PO_pkg.m_etc2,
                &create_PO_pkg.m_etc3,
                &create_PO_pkg.m_etc4,
                create_PO_pkg.m_etno,
                create_PO_pkg.m_etno2,

                create_PO_pkg.m_etno3,
                create_PO_pkg.m_etno4,
                &create_PO_pkg.m_fst,
                &create_PO_pkg.m_ist,
                &create_PO_pkg.m_ist10,
                &create_PO_pkg.m_ist11,
                &create_PO_pkg.m_ist12,
                &create_PO_pkg.m_ist2,
                &create_PO_pkg.m_ist3,
                &create_PO_pkg.m_ist4,
                &create_PO_pkg.m_ist5,
                &create_PO_pkg.m_ist6,
                &create_PO_pkg.m_ist7,
                &create_PO_pkg.m_ist8,
                &create_PO_pkg.m_ist9,
                create_PO_pkg.m_poid,
                create_PO_pkg.m_pon,
                &create_PO_pkg.m_post,
                create_PO_pkg.m_pqty,
                create_PO_pkg.m_pqty10,
                create_PO_pkg.m_pqty11,
                create_PO_pkg.m_pqty12,
                create_PO_pkg.m_pqty2,
                create_PO_pkg.m_pqty3,
                create_PO_pkg.m_pqty4,
                create_PO_pkg.m_pqty5,
                create_PO_pkg.m_pqty6,
                create_PO_pkg.m_pqty7,
                create_PO_pkg.m_pqty8,
                create_PO_pkg.m_pqty9,
                &create_PO_pkg.m_pst,
                create_PO_pkg.m_satno,
                create_PO_pkg.m_satno2,
                create_PO_pkg.m_satno3,
                create_PO_pkg.m_satno4,
                &create_PO_pkg.m_satc,
                &create_PO_pkg.m_satc2,
                &create_PO_pkg.m_satc3,
                &create_PO_pkg.m_satc4,
                &create_PO_pkg.m_tpc,
                &create_PO_pkg.m_va,
                create_PO_pkg.m_vna,
                create_PO_pkg.m_vno,
                &create_PO_pkg.m_vpn,
                create_PO_pkg.m_status
                                        );
if (ui_msg) {strMsgText.Format("create_PO_process: rc=%d, etc
=%d",rc,create_PO_pkg.m_etc);
AfxMessageBox(strMsgText);

                        }
return rc;
}

}


else {
AfxMessageBox("Error:Cannot load Coord.dll");
return -1;
}
return -1;
}

void CCreatePurchaseOrder::OnSubmit()
{
int rc = 0;

TRACE("..Invoke PO Submit container\n");
create_PO_pkg.UpdateData(TRUE);

// Update Status bar
create_PO_pkg.m_status = "Submitting, please wait ...";
if (create_PO_pkg.UpdateData(FALSE))
TRACE(".....PO form updat successful\n");
else
TRACE(".....PO form update failed\n");

rc = create_PO_process(C_SUBMIT);
TRACE("..Return valu = %d\n", rc);

if (rc != 0) {
TRACE(".....Invoke create_PO submit result fail\n");
return;
// Implement roll-back code here later
}

create_PO_pkg.m_status = "PO order submitted, click CONFIRM to commit
and print";
if (create_PO_pkg.UpdateData(FALSE))
TRACE(".....P form update successful\n");
else
TRACE(".....PO form update failed\n");

}

void CCreatePurchaseOrder::OnConfirmPrint()
{
// TODO: Add your control notification handler code here
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..create_PO ConfirmPrint container\n");
create_PO_pkg.UpdateData(TRUE);
// Update Status bar
create_PO_pkg.m_status = "Confirming, please wait ...";
if (create_PO_pkg.UpdateData(FALSE))
TRACE(".....O form update successful\n");
else
TRACE(".....PO form update failed\n");

rc = create_PO_process(C_CONFIRMPRINT);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
TRACE(".....create_PO ConfirmPrint result fail\n");
return;
// Implement roll-back code here later
}

create_PO_pkg.m_status = "PO order Confirmed and Printed";
if (create_PO_pkg.UpdateData(FALSE))
TRACE(".....POo form update successful\n");
else
TRACE(".....PO form update failed\n");

}

typedef int (*CreatePOClearProc)();

void CCreatePurchaseOrder::OnClear()
{
int rc = 0;
CreatePOClearProc lpfnDllFunc1;
// Function pointer
Cstring strMsgText;

create_PO_pkg.UpdateData(TRUE);
if (hinstDLL != NULL)
{
lpfnDllFunc1 = (CreatePOClearProc)GetProcAddress(hinstDLL,
"Req_PO_Order_Clear");
if (!lpfnDllFunc1)
                {
// handle the error
TRACE("clear create_PO: GetProcAddress fail\n");
AfxMessageBox("Error: Clear Form Proc Not Found");
                }
                else
                {
// call the function
rc = lpfnDllFunc1();
if (ui_msg) {
strMsgText.Format("clear create_PO: rc=%d", rc);
AfxMessageBox(strMsgText);
                        }

init_create_PO_pkg();
create_PO_pkg.m_status = "PO order Cleared";
if (create_PO_pkg.UpdateData(FALSE))
TRACE(".....fform update successful\n");
else
TRACE(".....PO form update failed\n");
                }
}
else {
AfxMessageBox("Error:Cannot load Coord.dll");
return;
}
return;
```

```
}
```

CreatePurchaseOrder.h

```
#if
!defined(AFX_CREATEPURCHASEORDER_H__5B707F62_9014_11D1_99
46_006097CC972B__INCLUDED_)
#define
AFX_CREATEPURCHASEORDER_H__5B707F62_9014_11D1_9946_0060
97CC972B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// CreatePurchaseOrder.h : header file
//

/////////////////////////////////////////////////////////////////////////
// CCreatePurchaseOrder dialog

class CCreatePurchaseOrder : public CDialog
{
// Construction
public:
        CCreatePurchaseOrder(CWnd* pParent = NULL);  // standard
constructor

// Dialog Data
        //{{AFX_DATA(CCreatePurchaseOrder)
        enum { IDD = IDD_CREATEPO };
        double      m_atc;
        double      m_atc2;
        double      m_atc3;
        double      m_atc4;
        CString     m_atno;
        CString     m_atno2;
        CString     m_atno3;
        CString     m_atno4;
        double      m_etc2;
        double      m_etc3;
        double      m_etc4;
        CString     m_etno;
        CString     m_etno2;
        CString     m_etno3;
        CString     m_etno4;
        double      m_fst;
        double      m_ist;
        double      m_ist10;
        double      m_ist11;
        double      m_ist12;
        double      m_ist2;
        double      m_ist3;
        double      m_ist4;
        double      m_ist5;
        double      m_ist6;
        double      m_ist7;
        double      m_ist8;
        double      m_ist9;
        COleDateTime            m_poid;
        int         m_pon;
        double      m_post;
        short       m_pqty;
        short       m_pqty10;
        short       m_pqty11;
        short       m_pqty12;
        short       m_pqty2;
        short       m_pqty3;
        short       m_pqty4;
        short       m_pqty5;
        short       m_pqty6;
        short       m_pqty7;
        short       m_pqty8;
        short       m_pqty9;
        double      m_pst;
        CString     m_satno;
        CString     m_satno2;
        CString     m_satno3;
        CString     m_satno4;
        double      m_satc;
        double      m_satc2;
        double      m_satc3;
        double      m_satc4;
        double      m_tpc;
        CString     m_va;
        CString     m_vna;
        CString     m_vno;
        CString     m_vpn;
        CString     m_status;
        double      m_etc;
        //}}AFX_DATA


// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CCreatePurchaseOrder)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
virtual BOOL OnInitDialog( );

//}}AFX_VIRTUAL

// Implementation
protected:

// Generated message map functions
//{{AFX_MSG(CCreatePurchaseOrder)
afx_msg void OnSubmit();
afx_msg void OnConfirmPrint();
afx_msg void OnClear();

//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

extern CCreatePurchaseOrder create_PO_pkg;
//extern int create_PO_submit();


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_CREATEPURCHASEORDER_H__5B707F62_9014_11D1_99
46_006097CC972B__INCLUDED_)
```

CreateWO.cpp

```
// CreateWO.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "CreateWO.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define C_EQ_INFO               0
#define C_EMP_INFO              1
#define C_SAVE                  2
#define C_WOEDIT                3
#define C_WOUPDATE              4


CCreateWO create_WO_pkg;


/////////////////////////////////////////////////////////////////////////
// CCreateWO dialog


CCreateWO::CCreateWO(CWnd* pParent /*=NULL*/)
        : CDialog(CCreateWO::IDD, pParent)
{
        //{{AFX_DATA_INIT(CCreateWO)
        m_won =                 _T("");
        m_sn =                  _T("");
        m_id =                  time_t(0);
        m_ib =                  _T("");
        m_sd =                  time_t(0);
        m_tcd =                 time_t(0);
        m_combo_en =            _T("");
        m_combo_en2 =           _T("");
        m_combo_en3 =           _T("");
        m_combo_en4 =           _T("");
        m_dmp =                 _T("");
        m_ff =                  _T("");
        m_location =            _T("");
        m_mt =                  _T("");
        m_etna =                _T("");
        m_wad =                 _T("");
        m_status =              _T("");
        m_tna=                  _T("");
        m_tl =                  _T("");
        m_cd =                  time_t(0);
        m_ds =                  time_t(0);
```

```
        //}}AFX_DATA_INIT
}

void CCreateWO::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CCreateWO)
        DDX_Control(pDX, IDC_COMBO_EN4, m_combo_en4_ctrl);
        DDX_Control(pDX, IDC_COMBO_EN3, m_combo_en3_ctrl);
        DDX_Control(pDX, IDC_COMBO_EN2, m_combo_en2_ctrl);
        DDX_Control(pDX, IDC_COMBO_EN, m_combo_en_ctrl);
        DDX_Text(pDX, IDC_WON, m_won);
        DDX_Text(pDX, IDC_SER_NO, m_sn);
        DDX_Text(pDX, IDC_ID,m_id);
        DDX_Text(pDX, IDC_IB, m_ib);
        DDX_Text(pDX, IDC_SD,m_sd);
        DDX_Text(pDX, IDC_TCD,          m_tcd);
        DDX_CBString(pDX, IDC_COMBO_EN, m_combo_en);
        DDX_CBString(pDX, IDC_COMBO_EN2, m_combo_en2);
        DDX_CBString(pDX, IDC_COMBO_EN3, m_combo_en3);
        DDX_CBString(pDX, IDC_COMBO_EN4, m_combo_en4);
        DDX_Text(pDX, IDC_DMP, m_dmp);
        DDX_Text(pDX, IDC_FF, m_ff);
        DDX_Text(pDX, IDC_LOCATION, m_location);
        DDX_Text(pDX, IDC_MT, m_mt);
        DDX_Text(pDX, IDC_ETNA, m_etna);
        DDX_Text(pDX, IDC_WAD, m_wad);
        DDX_Text(pDX, IDC_STATUS, m_status);
        DDX_Text(pDX, IDC_TNA,          m_tna);
        DDX_Text(pDX, IDC_TL, m_tl);
        DDX_Text(pDX, IDC_CD, m_cd);
        DDX_Text(pDX, IDC_DS, m_ds);
        //}}AFX_DATA_MAP
}

// Implement the override function to initialize the WO entry form fields.
BOOL CCreateWO::OnInitDialog( )
{
COleDateTime timeNow;
BOOL rc;

rc = CDialog::OnInitDialog();
timeNow = COleDateTime::GetCurrentTime();
m_id.SetDate(timeNow.GetYear(), timeNow.GetMonth(), timeNow.GetDay());
create_WO_pkg.UpdateData(FALSE);
return(TRUE);
}

BEGIN_MESSAGE_MAP(CCreateWO, CDialog)
//{{AFX_MSG_MAP(CCreateWO)
ON_BN_CLICKED(IDEQ_INFO, OnEq_Info)
ON_BN_CLICKED(IDC_EMP_INFO, OnEmp_Info)
ON_BN_CLICKED(IDC_SAVE, OnSave)
ON_BN_CLICKED(IDC_CLEAR_WO_FORM, OnWOClear)
ON_BN_CLICKED(IDC_WEDIT, OnEdit)
ON_BN_CLICKED(IDC_UPDATE, OnUpdate)

//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCreatePurchaseOrder message handlers Utilities
//

void init_create_WO_pkg()
{
COleDateTime timeNow;

create_WO_pkg.m_won             = _T("");
create_WO_pkg.m_sn              = _T("");
create_WO_pkg.m_id              = time_t(0);
create_WO_pkg.m_ib              = _T("");
create_WO_pkg.m_sd              = time_t(0);
create_WO_pkg.m_tcd             = time_t(0);
create_WO_pkg.m_combo_en        = _T("");
create_WO_pkg.m_combo_en2       = _T("");
create_WO_pkg.m_combo_en3       = _T("");
create_WO_pkg.m_combo_en4       = _T("");
create_WO_pkg.m_dmp             = _T("");
create_WO_pkg.m_ff              = _T("");
create_WO_pkg.m_location        = _T("");
create_WO_pkg.m_mt              = _T("");
create_WO_pkg.m_etna            = _T("");
create_WO_pkg.m_wad             = _T("");
create_WO_pkg.m_status          = _T("");
create_WO_pkg.m_tna             = _T("");
create_WO_pkg.m_tl              = _T("");
create_WO_pkg.m_cd              = time_t(0);
create_WO_pkg.m_ds              = time_t(0);
timeNow = COleDateTime::GetCurrentTime();
create_WO_pkg.m_id.SetDate(timeNow.GetYear(), timeNow.GetMonth(),
timeNow.GetDay());
}

typedef int (*CreateWOProc)(int wo_select,

        CString             m_won,
        CString             *m_sn,
        COleDateTime        *m_id,
        CString             *m_ib,
        COleDateTime        *m_sd,
        COleDateTime        *m_tcd,
        CString             *m_combo_en,
        CString             *m_combo_en2,
        CString             *m_combo_en3,
        CString             *m_combo_en4,
        CString             *m_dmp,
        CString             *m_ff,
        CString             *m_location,
        CString             *m_mt,
        CString             *m_etna,
        CString             *m_wad,
        Cstring             & m_status,
        CString             *m_tna,
        CString             *m_tl,
        COleDateTime        *m_cd,
        COleDateTime        *m_ds
        );

int create_WO_process(int wo_select)
{
int                                                     rc = -1;
CreateWOProc lpfnDllFunc1;
// Function pointer
Cstring strMsgText;

TRACE("create_WO_process\n");
if (ui_msg) {
AfxMessageBox("create_WO_process");
        }

if (hinstDLL != NULL)
        {
lpfnDllFunc1 = (CreateWOProc)GetProcAddress(hinstDLL,
"Req_create_wo");
if (!lpfnDllFunc1)
        {
// handle the error
TRACE("create_WO_process: GetProcAddress fail\n");
AfxMessageBox("Error:Coord Interface not Found");

        }
else
        {
// call the function
if (ui_msg) {strMsgText.Format("create_WO_process: Call coord DLL SN=%s
IB=%s ", create_WO_pkg.m_sn, create_WO_pkg.m_ib);
AfxMessageBox(strMsgText);
        }


rc = lpfnDllFunc1
(wo_select,
        create_WO_pkg.m_won,
        &create_WO_pkg.m_sn,
        &create_WO_pkg.m_id,
        &create_WO_pkg.m_ib,
        &create_WO_pkg.m_sd,
        &create_WO_pkg.m_tcd,
        &create_WO_pkg.m_combo_en,
        &create_WO_pkg.m_combo_en2,
        &create_WO_pkg.m_combo_en3,
        &create_WO_pkg.m_combo_en4,
        &create_WO_pkg.m_dmp,
        &create_WO_pkg.m_ff,
        &create_WO_pkg.m_location,
        &create_WO_pkg.m_mt,
        &create_WO_pkg.m_etna,
        &create_WO_pkg.m_wad,
        create_WO_pkg.m_status,
        &create_WO_pkg.m_tna,
        &create_WO_pkg.m_tl,
        &create_WO_pkg.m_cd,
        &create_WO_pkg.m_ds                                     );
if (ui_msg) {strMsgText.Format("create_WO_process: rc=%d, rc");
AfxMessageBox(strMsgText);
                }

if(!create_WO_pkg.m_combo_en.IsEmpty())
```

```
{
CString strCombo = create_WO_pkg.m_combo_en;
strCombo = strCombo.Left(strCombo.GetLength()-1);
create_WO_pkg.m_combo_en_ctrl.ResetContent();
create_WO_pkg.m_combo_en2_ctrl.ResetContent();
create_WO_pkg.m_combo_en3_ctrl.ResetContent();
create_WO_pkg.m_combo_en4_ctrl.ResetContent();
int idx =strCombo.ReverseFind('.');
while(idx != -1)
int ilen = strCombo.GetLength();
CString sRightStr = strCombo.Right(ilen-(idx+1));
create_WO_pkg.m_combo_en_ctrl.AddString(sRightStr);
create_WO_pkg.m_combo_en2_ctrl.AddString(sRightStr);
create_WO_pkg.m_combo_en3_ctrl.AddString(sRightStr);
create_WO_pkg.m_combo_en4_ctrl.AddString(sRightStr);
strCombo = strCombo.Left(idx);
idx =strCombo.ReverseFind('.');
                                        }
if(!strCombo.IsEmpty())
                                        {
create_WO_pkg.m_combo_en_ctrl.AddString(strCombo);
create_WO_pkg.m_combo_en2_ctrl.AddString(strCombo);
create_WO_pkg.m_combo_en3_ctrl.AddString(strCombo);
create_WO_pkg.m_combo_en4_ctrl.AddString(strCombo);
                                        }
create_WO_pkg.m_combo_en = strCombo;
create_WO_pkg.m_combo_en2 = "";
create_WO_pkg.m_combo_en3 = "";
create_WO_pkg.m_combo_en4 = "";
int iComboSize = create_WO_pkg.m_combo_en_ctrl.GetCount();
switch (iComboSize)
                                                {
case 2:
create_WO_pkg.m_combo_en3_ctrl.ResetContent();
create_WO_pkg.m_combo_en4_ctrl.ResetContent();
break;
case 3:
create_WO_pkg.m_combo_en4_ctrl.ResetContent();
break;
default:
break;
;

}
return rc;
}

}


else {
AfxMessageBox("Error:Cannot load Coord.dll");
return -1;
}
return -1;
}



void CCreateWO::OnEq_Info()
{
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..Invoke WO Eqpt Info container\n");
create_WO_pkg.UpdateData(TRUE);

// Update Status bar
create_WO_pkg.m_status = "Getting Eqpt Info, please wait ...";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......WO form update successful\n");
else
TRACE("......WO form update failed\n");


rc = create_WO_process(C_EQ_INFO);
TRACE("..Return value = %d\n", rc);
if (rc != 0) {
TRACE("......Invoke create_WO submit result fail\n");
return;
// Implement roll-back code here later
}

create_WO_pkg.m_status = "Eqpt Info retrieved, click EQ_INFO to Get
Employee Info";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Eqpt Info retrieved\n");
else
TRACE("......Get Eqpt Info failed\n");
}

void CCreateWO::OnEmp_Info()
{
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..create_WO Emp_Info container\n");
create_WO_pkg.UpdateData(TRUE);

// Update Status bar
create_WO_pkg.m_status = "Getting Employee Info, please wait ...";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Retrieval of Employee Info successful\n");
else
TRACE("......Retrieval of Employee Info failed\n");

rc = create_WO_process(C_EMP_INFO);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
                TRACE("......get Emp Info result fail\n");
                return;
                // Implement roll-back code here later
}

create_WO_pkg.m_status = "WO order confirmed";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Retrieval of Employee Info successful\n");
else
TRACE("......Retrieval of Employee Info failed\n");

}

void CCreateWO::OnSave()
{
// TODO: Add your control notification handler code here
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..Save WO container\n");
create_WO_pkg.UpdateData(TRUE);

// Update Status bar
create_WO_pkg.m_status = "Saving, please wait ...";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Save successful\n");
else
TRACE("......Save failed\n");
rc = create_WO_process(C_SAVE);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
TRACE("......Save fail\n");
return;
// Implement roll-back code here later
}

create_WO_pkg.m_status = "WO Saved";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......WO Saved\n");
else
TRACE("......Save failed\n");
}

void CCreateWO::OnUpdate()
{
// TODO: Add your control notification handler code here
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..Update WO container\n");
create_WO_pkg.UpdateData(TRUE);

// Update Status bar
create_WO_pkg.m_status = "Updating, please wait ...";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Update successful\n");
else
TRACE("......Update failed\n");
rc = create_WO_process(C_WOUPDATE);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
TRACE("......Update fail\n");
return;
// Implement roll-back code here later
}

create_WO_pkg.m_status = "WO Updated";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......WO Updated\n");
else
```

```
TRACE("......Update failed\n");

}

void CCreateWO::OnEdit()
{
// TODO: Add your control notification handler code here
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..Edit WO container\n");
create_WO_pkg.UpdateData(TRUE);

// Update Status bar
create_WO_pkg.m_status = "Retrieving, please wait ...";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......Retrieve successful\n");
else
TRACE("......Retrieve failed\n");

rc = create_WO_process(C_WOEDIT);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
TRACE("......Retrieve fail\n");
return;
// Implement roll-back code here later
}

create_WO_pkg.m_status = "WO Retrieved";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......WO Retrieved\n");
else
TRACE("......Retrieve failed\n");

}


typedef int (*CreateWOClearProc)();

void CCreateWO::OnWOClear()
{
int rc = 0;
CreateWOClearProc lpfnDllFunc1;
// Function pointer
Cstring strMsgText;
create_WO_pkg.UpdateData(TRUE);
if (hinstDLL != NULL)
{
lpfnDllFunc1 = (CreateWOClearProc)GetProcAddress(hinstDLL,
"Req_WO_Order_Clear");
if (!lpfnDllFunc1)
        {
// handle the error
TRACE("clear create_WO: GetProcAddress fail\n");
AfxMessageBox("Error: Clear Form Proc Not Found");
        }
        else
        {
// call the function
rc = lpfnDllFunc1();
if (ui_msg) {
strMsgText.Format("clear create_WO: rc=%d", rc);
AfxMessageBox(strMsgText);
        }

init_create_WO_pkg();
create_WO_pkg.m_status = "WO order Cleared";
if (create_WO_pkg.UpdateData(FALSE))
TRACE("......WO form update successful\n");
else
TRACE("......WO form update failed\n");
        }
}
else {
AfxMessageBox("Error:Cannot load Coord.dll");
        return;
}
return;
}
```

CreateWO.h

```
#if
!defined(AFX_CREATEWO_H__14239641_9741_11D1_9946_006097CC97
2B__INCLUDED_)
#define
AFX_CREATEWO_H__14239641_9741_11D1_9946_006097CC972B__INC
LUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// CreateWO.h : header file
//

/////////////////////////////////////////////////////////////////////////
// CCreateWO dialog

class CCreateWO : public CDialog
{
// Construction
public:
CCreateWO(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
        //{{AFX_DATA(CCreateWO)
        enum { IDD = IDD_WO };
        CComboBox m_combo_en4_ctrl;
        CComboBox m_combo_en3_ctrl;
        CComboBox m_combo_en2_ctrl;
        CComboBox m_combo_en_ctrl;
        CString                 m_won;
        CString                 m_sn;
        COleDateTime            m_id;
        CString                 m_ib;
        COleDateTime            m_sd;
        COleDateTime            m_tcd;
        CString                 m_combo_en;
        CString                 m_combo_en2;
        CString                 m_combo_en3;
        CString                 m_combo_en4;
        CString                 m_dmp;
        CString                 m_ff;
        CString                 m_location;
        CString                 m_mt;
        CString                 m_etna;
        CString                 m_wad;
        CString                 m_status;
        CString                 m_tna;
        CString                 m_tl;
        COleDateTime            m_cd;
        COleDateTime            m_ds;
        //}}AFX_DATA


// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CCreateWO)
protected:
virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
virtual BOOL OnInitDialog( );
//}}AFX_VIRTUAL

// Implementation
protected:

        // Generated message map functions
        //{{AFX_MSG(CCreateWO)
        afx_msg void OnEq_Info();
        afx_msg void OnEmp_Info();
        afx_msg void OnSave();
        afx_msg void OnWOClear();
        afx_msg void OnEdit();
        afx_msg void OnUpdate();

//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

extern CCreateWO create_WO_pkg;
extern int create_WO_process();

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_CREATEWO_H__14239641_9741_11D1_9946_006097CC97
2B__INCLUDED_)
```

DeleteInventoryItem.cpp

```
// DeleteInventoryItem.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "DeleteInventoryItem.h"

#ifdef _DEBUG
#define new DEBUG_NEW
```

```
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

CDeleteInventoryItem d_inv_i_pkg;
typedef int (*DeleteInvItemProc)(
CString        m_mpn

                                );

int invoke_d_inv_i()
{
int rc = -1;
DeleteInvItemProc lpfnDllFunc1;
// Function pointer

TRACE("Invoke_d_inv_i\n");
if (hinstDLL != NULL)
{
lpfnDllFunc1 = (DeleteInvItemProc)GetProcAddress(hinstDLL,
"Req_del_inventory_item");
if (!lpfnDllFunc1)
                {
// handle the error
//
AfxFreeLibrary(hinstDLL);
TRACE("Invoke_d_inv_i: GetProcAddress fail\n");
                }
else
                {
// call the function
rc = lpfnDllFunc1
                (
d_inv_i_pkg.m_mpn
                );
TRACE("Invoke_d_inv_i: rc=%d\n", rc);
//
AfxFreeLibrary(hinstDLL);
                }
                }
                return rc;
}


/////////////////////////////////////////////////////////////////////////////
// CDeleteInventoryItem dialog


CDeleteInventoryItem::CDeleteInventoryItem(CWnd* pParent /*=NULL*/)
                : CDialog(CDeleteInventoryItem::IDD, pParent)
{
                //{{AFX_DATA_INIT(CDeleteInventoryItem)
                //m_checkdelete = FALSE;
                m_mpn = _T("");
                //}}AFX_DATA_INIT
}


void CDeleteInventoryItem::DoDataExchange(CDataExchange* pDX)
{
                CDialog::DoDataExchange(pDX);
                //{{AFX_DATA_MAP(CDeleteInventoryItem)
                //DDX_Check(pDX, IDC_CHECK_DELETE, m_checkdelete);
                DDX_Text(pDX, IDC_EDIT_MPN, m_mpn);
                DDV_MaxChars(pDX, m_mpn, 25);
                //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CDeleteInventoryItem, CDialog)
//{{AFX_MSG_MAP(CDeleteInventoryItem)
// NOTE: the ClassWizard will add message map macros here
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDeleteInventoryItem message handlers
```

DeleteInventoryItem.h

```
#if
!defined(AFX_DELETEINVENTORYITEM_H__5B707F61_9014_11D1_9946
_006097CC972B__INCLUDED_)
#define
AFX_DELETEINVENTORYITEM_H__5B707F61_9014_11D1_9946_006097
CC972B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// DeleteInventoryItem.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CDeleteInventoryItem dialog

class CDeleteInventoryItem : public CDialog
{
// Construction
public:
CDeleteInventoryItem(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
//{{AFX_DATA(CDeleteInventoryItem)
enum { IDD = IDD_DELETE_INVENTORY_ITEM };
//              BOOL        m_checkdelete;
CString        m_mpn;
//}}AFX_DATA


// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CDeleteInventoryItem)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:

// Generated message map functions
//{{AFX_MSG(CDeleteInventoryItem)
// NOTE: the ClassWizard will add member functions here
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

extern CDeleteInventoryItem d_inv_i_pkg;
extern int invoke_d_inv_i();

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line

#endif //
!defined(AFX_DELETEINVENTORYITEM_H__5B707F61_9014_11D1_9946
_006097CC972B__INCLUDED_)
```

WO_Statu.cpp

```
// WO_Statu.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "WO_Statu.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//#define C_WO_STATUS


CWO_Statu WO_Status_pkg;
/////////////////////////////////////////////////////////////////////////////
// CWO_Statu dialog


CWO_Statu::CWO_Statu(CWnd* pParent /*=NULL*/)
                : CDialog(CWO_Statu::IDD, pParent)
{
                //{{AFX_DATA_INIT(CWO_Statu)
                m_wono =                        _T("");
                m_cs =                          _T("");
                m_sdate =                       time_t(0);
                m_lcdate =                      time_t(0);
                m_ldate =                       time_t(0);
                m_comp_ques =                   _T("");
                m_late_start_ques =             _T("");
                m_late_comp_ques =              _T("");
                m_started_ques =                _T("");
                m_idate =                       time_t(0);
//}}AFX_DATA_INIT
}


void CWO_Statu::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
```

```
//{{AFX_DATA_MAP(CWO_Statu)
DDX_Text(pDX, IDC_WONO, m_wono);
DDX_Text(pDX, IDC_Completion_Status, m_cs);
DDX_Text(pDX, IDC_StartD,m_sdate);
DDX_Text(pDX, IDC_TargetCD, m_tcdate);
DDX_Text(pDX, IDC_Today_Date, m_tdate);
DDX_Text(pDX, IDC_CompletedQues,m_comp_ques);
DDX_Text(pDX, IDC_Late_StartQues, m_late_start_ques);
DDX_Text(pDX, IDC_Late_CompQues, m_late_comp_ques);
DDX_Text(pDX, IDC_StartedQues, m_started_ques);
DDX_Text(pDX, IDC_IDate, m_idate);
//}}AFX_DATA_MAP
}

// Implement the override function to initialize the WO Status Today's Date
field.
BOOL CWO_Statu::OnInitDialog( )
{
COleDateTime timeNow;
BOOL rc;

rc = CDialog::OnInitDialog();
timeNow = COleDateTime::GetCurrentTime();
m_tdate.SetDate(timeNow.GetYear(), timeNow.GetMonth(),
timeNow.GetDay());
WO_Status_pkg.UpdateData(FALSE);
return(TRUE);
}


BEGIN_MESSAGE_MAP(CWO_Statu, CDialog)
//{{AFX_MSG_MAP(CWO_Statu)
ON_BN_CLICKED(IDGET_STATUS, OnGet_Status)
ON_BN_CLICKED(IDCLEAR_FORM, OnWOStatusClear)

//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////
// CWO_Statu message handlers
void init_WO_Status_pkg()
{
COleDateTime timeNow;

WO_Status_pkg.m_wono =                  _T("");
WO_Status_pkg.m_cs =                    _T("");
WO_Status_pkg.m_sdate =                 time_t(0);
WO_Status_pkg.m_tcdate =                time_t(0);
WO_Status_pkg.m_tdate =                 time_t(0);
WO_Status_pkg.m_comp_ques =             _T("");
WO_Status_pkg.m_late_start_ques =       _T("");
WO_Status_pkg.m_late_comp_ques =        _T("");
WO_Status_pkg.m_started_ques =          _T("");
WO_Status_pkg.m_idate =                 time_t(0);

timeNow = COleDateTime::GetCurrentTime();
WO_Status_pkg.m_tdate.SetDate(timeNow.GetYear(), timeNow.GetMonth(),
timeNow.GetDay());
}

typedef int (*WOStatusProcessProc)(

            CString             m_wono,
            CString             *m_cs,
            COleDateTime        *m_sdate,
            COleDateTime        *m_tcdate,
            COleDateTime        *m_tdate,
            CString             *m_comp_ques,
            CString             *m_late_start_ques,
            CString             *m_late_comp_ques,
            CString             *m_started_ques,
            COleDateTime        *m_idate

                    );

int WO_Status_process()
{
int rc = 0;
WOStatusProcessProc lpfnDllFunc1;
// Function pointer
Cstring strMsgText;

TRACE("WO_Status_process\n");
if (ui_msg) {
AfxMessageBox("WO_Status_process");
}

if (hinstDLL != NULL)
{
lpfnDllFunc1 = (WOStatusProcessProc)GetProcAddress(hinstDLL,
"get_WO_Status");

if (!lpfnDllFunc1)
        {
// handle the error
TRACE("WO_Status_process: GetProcAddress fail\n");
AfxMessageBox("Error:Coord Interface not Found");

        }
        else
        {
// call the function
if (ui_msg) {strMsgText.Format("WO_Status_process: Call coord DLL
WON=%s ",WO_Status_pkg.m_wono);
AfxMessageBox(strMsgText);
                }

rc = lpfnDllFunc1
(/*wo_status,*/

            WO_Status_pkg.                  m_wono,
            &WO_Status_pkg.                 m_cs,
            &WO_Status_pkg.                 m_sdate,
            &WO_Status_pkg.                 m_tcdate,
            &WO_Status_pkg.                 m_tdate,
            &WO_Status_pkg.                 m_comp_ques,
            &WO_Status_pkg.                 m_late_start_ques,
            &WO_Status_pkg.                 m_late_comp_ques,
            &WO_Status_pkg.                 m_started_ques,
            &WO_Status_pkg.                 m_idate
                                                    );
if (ui_msg) {
strMsgText.Format("WO_Status_process: rc=%d, rc");
                                        AfxMessageBox(strMsgText);
        }
return rc;
}

}


else {
AfxMessageBox("Error:Cannot load Coord.dll");
return -1;
        }
return -1;
}

void CWO_Statu::OnGet_Status()
{
int rc = 0;

// TODO: Add your control notification handler code here
TRACE("..Invoke WO Status Info container\n");
WO_Status_pkg.UpdateData(TRUE);

// Update Status bar
//
create_WO_pkg.m_status = "Getting Eqpt Info, please wait ...";
if (WO_Status_pkg.UpdateData(FALSE))
TRACE("......Get WO Status Info successfu\n");
else
TRACE("......Get WO Status Info failed\n");

rc = WO_Status_process(/*C_WO_STATUS*/);
TRACE("..Return value = %d\n", rc);

if (rc != 0) {
TRACE("......Invoke WO Status submit result fail\n");
return;
// Implement roll-back code here later
}

//
create_WO_pkg.m_status = "Eqpt Info retrieved, click EQ_INFO to Get
Employee Info";
if (WO_Status_pkg.UpdateData(FALSE))
TRACE("......Eqpt Info retrieved\n");
else
TRACE("......Get Eqpt Info failed\n");

}



typedef int (*WOStatusClearProc)();

void CWO_Statu::OnWOStatusClear()
{
int rc = 0;
WOStatusClearProc lpfnDllFunc1;
// Function pointer
Cstring strMsgText;
```

```
WO_Status_pkg.UpdateData(TRUE);
if (hinstDLL != NULL)
{
lpfnDllFunc1 = (WOStatusClearProc)GetProcAddress(hinstDLL,
"Get_WO_Status_Clear");
if (!lpfnDllFunc1)
{
// handle the error
TRACE("clear WO_Status: GetProcAddress fail\n");
AfxMessageBox("Error: Clear Form Proc Not Found");
                    }
else
{
// call the function
rc = lpfnDllFunc1();
if (ui_msg) {
strMsgText.Format("clear WO_Status: rc=%d", rc);
AfxMessageBox(strMsgText);
          }

init_WO_Status_pkg();
//WO_Status_pkg.m_status = "WO Status Cleared";
if (WO_Status_pkg.UpdateData(FALSE))
TRACE(".....WO Status successful\n");
else
TRACE(".....WO Status failed\n");
          }
}
else {
AfxMessageBox("Error:Cannot load Coord.dll");
return;
}
return;
}
```

WO_Statu.h

```
#if
!defined(AFX_WO_STATU_H__2345BFA1_F0DC_11D1_9946_006097CC9
72B__INCLUDED_)
#define
AFX_WO_STATU_H__2345BFA1_F0DC_11D1_9946_006097CC972B__IN
CLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// WO_Statu.h : header file
//

/////////////////////////////////////////////////////////////////////////
// CWO_Statu dialog

class CWO_Statu : public CDialog
{
// Construction
public:
CWO_Statu(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
          //{{AFX_DATA(CWO_Statu)
          enum { IDD = IDD_WO_STATUS };
          CString              m_wono;
          CString              m_cs;
          COleDateTime         m_sdate;
          COleDateTime         m_tcdate;
          COleDateTime         m_tdate;
          CString              m_comp_ques;
          CString              m_late_start_ques;
          CString              m_late_comp_ques;
          CString              m_started_ques;
          COleDateTime         m_idate;
          //}}AFX_DATA


// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CWO_Statu)
protected:
virtual void DoDataExchange(CDataExchange* pDX);   // DDX/DDV support
virtual BOOL OnInitDialog( );
//}}AFX_VIRTUAL

// Implementation
protected:

// Generated message map functions
//{{AFX_MSG(CWO_Statu)
afx_msg void OnGet_Status();
afx_msg void OnWOStatusClear();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

extern CWO_Statu WO_Status_pkg;
extern int WO_Status_process();

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_WO_STATU_H__2345BFA1_F0DC_11D1_9946_006097CC9
72B__INCLUDED_)
```

NMaint.cpp

```
// NMaint.cpp : implementation file
//

#include "stdafx.h"
#include "LocalUI.h"
#include "NMaint.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

#define C_WK                    0
#define C_MON                   1


NMaint NM_pkg;
/////////////////////////////////////////////////////////////////////////
// NMaint dialog


NMaint::NMaint(CWnd* pParent /*=NULL*/)
          : CDialog(NMaint::IDD, pParent)
{
          //{{AFX_DATA_INIT(NMaint)
          m_today = time_t(0);
          m_etno = _T("");
          m_location = _T("");
          m_snm = _T("");
          m_snw = _T("");
          m_status = _T("");
          m_nmm = time_t(0);
          m_nwm = time_t(0);
          //}}AFX_DATA_INIT
}


void NMaint::DoDataExchange(CDataExchange* pDX)
{
          CDialog::DoDataExchange(pDX);
          //{{AFX_DATA_MAP(NMaint)
          DDX_Text(pDX, IDC_Today, m_today);
          DDX_Text(pDX, IDC_ETNO, m_etno);
          DDX_Text(pDX, IDC_LOCN, m_location);
          DDX_Text(pDX, IDC_SNM, m_snm);
          DDX_Text(pDX, IDC_SNW, m_snw);
          DDX_Text(pDX, IDC_STATUS, m_status);
          DDX_Text(pDX, IDC_NMMAINT, m_nmm);
          DDX_Text(pDX, IDC_NWMAINT, m_nwm);
          //}}AFX_DATA_MAP
}


// Implement the override function to initialize the NM's Today's Date field.
BOOL NMaint::OnInitDialog( )
{
          COleDateTime timeNow;
          BOOL rc;

          rc = CDialog::OnInitDialog();
          timeNow = COleDateTime::GetCurrentTime();
          m_today.SetDate(timeNow.GetYear(), timeNow.GetMonth(),
timeNow.GetDay());
          NM_pkg.UpdateData(FALSE);
          return(TRUE);
}


BEGIN_MESSAGE_MAP(NMaint, CDialog)
          //{{AFX_MSG_MAP(NMaint)
                    ON_BN_CLICKED(IDGet_WK, OnGetNMW)
                    ON_BN_CLICKED(IDGet_MON, OnGetNMM)
```

```
                    ON_BN_CLICKED(IDCLEAR_FORM,
OnNMClear)

//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////
// NMaint message handlers
void init_NM_pkg()
{
        COleDateTime timeNow;

        NM_pkg.m_today =            time_t(0);
        NM_pkg.m_etno =             _T("");
        NM_pkg.m_location =         _T("");
        NM_pkg.m_nwm =              time_t(0);
        NM_pkg.m_nmm =              time_t(0);
        NM_pkg.m_snm =              _T("");
        NM_pkg.m_snw =              _T("");
        NM_pkg.m_status =           _T("");


        timeNow = COleDateTime::GetCurrentTime();
        NM_pkg.m_today.SetDate(timeNow.GetYear(),
timeNow.GetMonth(), timeNow.GetDay());
}

typedef int (*NMProcessProc)(

        iint nm_select,
        COleDateTime        *m_today,
        CString                     m_etno,
        CString                     m_location,
        COleDateTime        *m_nwm,
        COleDateTime        *m_nmm,
        CString                     *m_snm,
        CString                     *m_snw,
        CString&            m_status
        );

int NM_process(int nm_select)
{
        int
        rc = -1;
        NMProcessProc               lpfnDllFunc1;   //
Function pointer
        CString
        strMsgText;

        TRACE("NM_process\n");
        if (ui_msg) {
        AfxMessageBox("NM_process");
        }

        if (hinstDLL != NULL)
        {
        lpfnDllFunc1 = (NMProcessProc)GetProcAddress(hinstDLL,
        "get_NM");
        if (!lpfnDllFunc1)
                {
        // handle the error
        TRACE("NM_process: GetProcAddress fail\n");
        AfxMessageBox("Error:Coord Interface not Found");
                }
                else
        {
        // call the function
        if (ui_msg) {
strMsgText.Format("NM_process: Call coord DLL ETNO=%s
LOCATION=%s",

        NM_pkg.m_etno, NM_pkg.m_location);

        AfxMessageBox(strMsgText);
                }
        rc = lpfnDllFunc1
        (nm_select,
        &NM_pkg.m_today,
        NM_pkg.m_etno,
        NM_pkg.m_location,
        &NM_pkg.m_nwm,
        &NM_pkg.m_nmm,
        &NM_pkg.m_snm,
        &NM_pkg.m_snw,
        NM_pkg.m_status

                                                    );
        if (ui_msg) {
        strMsgText.Format("NM_process: rc=%d, rc");
        AfxMessageBox(strMsgText);
        }
        return rc;
}
}

        else {
        AfxMessageBox("Error:Cannot load Coord.dll");
        return -1;
        }
        return -1;
}

void NMaint::OnGetNMW()
{
        int rc = 0;

        TRACE("..Invoke NM Info container\n");
        NM_pkg.UpdateData(TRUE);

        // Update Status bar
        NM_pkg.m_status = "Getting status, please wait ...";
        if (NM_pkg.UpdateData(FALSE))
        TRACE("......Get NM Info successful\n");
        else
        TRACE("......Get NM Info failed\n");

        rc = NM_process(C_WK);
        TRACE("..Return value = %d\n", rc);

        if (rc != 0) {
        TRACE("......Invoke NM submit result fail\n");
        return;
        // Implement roll-back code here later
        }

        NM_pkg.m_status = "status retrieved";
        if (NM_pkg.UpdateData(FALSE))
        TRACE("......NM Info retrieved\n");
        else
        TRACE("......Get NM Info failed\n");

}

void NMaint::OnGetNMM()
{
        int rc = 0;

        // TODO: Add your control notification handler code here
        TRACE("..Invoke NM Info container\n");
        NM_pkg.UpdateData(TRUE);

        // Update Status bar
        NM_pkg.m_status = "Getting Status, please wait ...";
        if (NM_pkg.UpdateData(FALSE))
        TRACE("......Get NM Info successful\n");
        else
        TRACE("......Get NM Info failed\n");

        rc = NM_process(C_MON);
        TRACE("..Return value = %d\n", rc);

        if (rc != 0) {
        TRACE("......Invoke NM submit result fail\n");
        return;
        // Implement roll-back code here later
        }

        NM_pkg.m_status = "Status retrieved";
        if (NM_pkg.UpdateData(FALSE))
        TRACE("......NM Info retrieved\n");
        else
        TRACE("......Get NM Info failed\n");
}

typedef int (*NMClearProc)();
void NMaint::OnNMClear()
{
int             rc = 0;
NMClearProc             lpfnDllFunc1;   // Function pointer
CString                                         strMsgText;

NM_pkg.UpdateData(TRUE);
if (hinstDLL != NULL)
        {
lpfnDllFunc1 = (NMClearProc)GetProcAddress(hinstDLL,
"Get_NM_Clear");
if (!lpfnDllFunc1)
        {
// handle the error
TRACE("clear WO_Status: GetProcAddress fail\n");
AfxMessageBox("Error: Clear Form Proc Not Found");
        }
```

```
            else
            {
// call the function
rc = lpfnDllFunc1();
if (ui_msg) {
strMsgText.Format("clear NM: rc=%d", rc);
AfxMessageBox(strMsgText);
            }

init_NM_pkg();
//
WO_Status_pkg.m_status = "WO Status Cleared";
if (NM_pkg.UpdateData(FALSE))
TRACE(".....NM successful\n");
else
TRACE(".....NM failed\n");
                        }
}
else {
AfxMessageBox("Error:Cannot load Coord.dll");
return;
}
return;
}
```

LocalUI.cpp

```
// LocalUI.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "LocalUI.h"

#include "MainFrm.h"
#include "LocalUIDoc.h"
#include "LocalUIView.h"
#include "TraceView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CLocalUIApp
typedef void (*SetTraceWndHandleProc)(HWND hwnd            );

BEGIN_MESSAGE_MAP(CLocalUIApp, CWinApp)
//{{AFX_MSG_MAP(CLocalUIApp)
ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
// Standard file based document commands
ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
// Standard print setup command
ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLocalUIApp construction

CLocalUIApp::CLocalUIApp()
{
m_hwndTrace = NULL;
}

// Declare the handler to the Coord DLL library.
HINSTANCE hinstDLL = NULL;
BOOL                    ui_msg = FALSE;

/////////////////////////////////////////////////////////////////////////////
// The one and only CLocalUIApp object

CLocalUIApp theApp;

// This identifier was generated to be statistically unique for your app.
// You may change it if you prefer to choose a specific identifier.

// {1D779142-366F-11D1-9A80-444553540000}
static const CLSID clsid =
{ 0x1d779142, 0x366f, 0x11d1, { 0x9a, 0x80, 0x44, 0x45, 0x53, 0x54, 0x0,
0x0 } };

/////////////////////////////////////////////////////////////////////////////
// CLocalUIApp initialization

BOOL CLocalUIApp::InitInstance()
{
// Initialize OLE libraries
if (!AfxOleInit())
{
AfxMessageBox(IDP_OLE_INIT_FAILED);
return FALSE;
}

AfxEnableControlContainer();

// Standard initialization
// If you are not using these features and wish to reduce the size
//  of your final executable, you should remove from the following
//  the specific initialization routines you do not need.

#ifdef _AFXDLL
Enable3dControls();
// Call this when using MFC in a shared DLL
#else
Enable3dControlsStatic();  // Call this when linking to MFC statically
#endif

// Change the registry key under which our settings are stored.
// You should modify this string to be something appropriate
// such as the name of your company or organization.
SetRegistryKey(_T("Local AppWizard-Generated Applications"));

LoadStdProfileSettings();  // Load standard INI file options (including MRU)

// Register the application's document templates.  Document templates
//  serve as the connection between documents, frame windows and views.

CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CLocalUIDoc),
                RUNTIME_CLASS(CMainFrame),
// main SDI frame window
                RUNTIME_CLASS(CTraceView));
                AddDocTemplate(pDocTemplate);

// Connect the COleTemplateServer to the document template.
//  The COleTemplateServer creates new documents on behalf
//  of requesting OLE containers by using information
//  specified in the document template.
m_server.ConnectTemplate(clsid, pDocTemplate, TRUE);
// Note: SDI applications register server objects only if /Embedding
//   or /Automation is present on the command line.

// Parse command line for standard shell commands, DDE, file open
CCommandLineInfo cmdInfo;
ParseCommandLine(cmdInfo);

// Check to see if launched as OLE server
if (cmdInfo.m_bRunEmbedded || cmdInfo.m_bRunAutomated)
{
// Register all OLE server (factories) as running.  This enables the
//  OLE libraries to create objects from other applications.
COleTemplateServer::RegisterAll();

// Application was run with /Embedding or /Automation.  Don't show the
//  main window in this case.
return TRUE;
}

// When a server application is launched stand-alone, it is a good idea
//  to update the system registry in case it has been damaged.
m_server.UpdateRegistry(OAT_DISPATCH_OBJECT);
COleObjectFactory::UpdateRegistryAll();

// Dispatch commands specified on the command line
if (!ProcessShellCommand(cmdInfo))
return FALSE;

// Load the Coordinator DLL library here
hinstDLL = AfxLoadLibrary("Coord.dll");


// The one and only window has been initialized, so show and update it.
m_pMainWnd->ShowWindow(SW_SHOW);
m_pMainWnd->UpdateWindow();

SetTraceWndHandleProc  lpfnDllFunc;
// Function pointer
lpfnDllFunc = (SetTraceWndHandleProc)GetProcAddress(hinstDLL,
"Req_Set_TraceWindowHandle");

if(lpfnDllFunc)
{
lpfnDllFunc(m_hwndTrace);
}
else
```

```
{
AfxMessageBox("Cannot find SetTraceWndHandleProc in Cord.dll");
}


return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
CAboutDlg();

// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA

// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
//{{AFX_MSG(CAboutDlg)
// No message handlers
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
//{{AFX_DATA_INIT(CAboutDlg)
//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
CDialog::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CAboutDlg)
//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
//{{AFX_MSG_MAP(CAboutDlg)
// No message handlers
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CLocalUIApp::OnAppAbout()
{
CAboutDlg aboutDlg;
aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CLocalUIApp commands

int CLocalUIApp::ExitInstance()
{
// TODO: Add your specialized code here and/or call the base class
if (hinstDLL != NULL)
{
        AfxFreeLibrary(hinstDLL);
        TRACE("Free Library Coord\n");
}

return CWinApp::ExitInstance();
}
```

LocalUI.h

```
// LocalUI.h : main header file for the LOCALUI application
//

#if
!defined(AFX_LOCALUI_H__1D779147_366F_11D1_9A80_444553540000_
_INCLUDED_)
#define
AFX_LOCALUI_H__1D779147_366F_11D1_9A80_444553540000__INCLU
DED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
#error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols


// Declare the handler to the Coord DLL library.
extern HINSTANCE        hinstDLL;
extern BOOL                     ui_msg;


/////////////////////////////////////////////////////////////////////////////
// CLocalUIApp:
// See LocalUI.cpp for the implementation of this class
//

class CLocalUIApp : public CWinApp
{
public:
CLocalUIApp();
HWND m_hwndTrace;
// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CLocalUIApp)
public:
virtual BOOL InitInstance();
virtual int ExitInstance();
//}}AFX_VIRTUAL

// Implementation
COleTemplateServer m_server;
// Server object for document creation
//{{AFX_MSG(CLocalUIApp)
afx_msg void OnAppAbout();
// NOTE - the ClassWizard will add and remove member functions here.
//    DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_LOCALUI_H__1D779147_366F_11D1_9A80_444553540000_
_INCLUDED_)
```

LocalUI.odl

```
// LocalUI.odl : type library source for LocalUI.exe

// This file will be processed by the MIDL compiler to produce the
// type library (LocalUI.tlb).

[ uuid(1D779143-366F-11D1-9A80-444553540000), version(1.0) ]
library LocalUI
{
importlib("stdole32.tlb");


//  Primary dispatch interface for CLocalUIDoc

[ uuid(1D779144-366F-11D1-9A80-444553540000) ]
dispinterface ILocalUI
{
properties:
// NOTE - ClassWizard will maintain property information here.
//    Use extreme caution when editing this section.
//{{AFX_ODL_PROP(CLocalUIDoc)
//}}AFX_ODL_PROP

methods:
// NOTE - ClassWizard will maintain method information here.
//    Use extreme caution when editing this section.
//{{AFX_ODL_METHOD(CLocalUIDoc)
//}}AFX_ODL_METHOD

};

//  Class information for CLocalUIDoc

[ uuid(1D779142-366F-11D1-9A80-444553540000) ]
coclass Document
{
[default] dispinterface ILocalUI;
```

```
};

//{{AFX_APPEND_ODL}}
//}}AFX_APPEND_ODL}}
};
```

LocalUIDoc.cpp

```
// LocalUIDoc.cpp : implementation of the CLocalUIDoc class
//

#include "stdafx.h"
#include "LocalUI.h"

#include "LocalUIDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CLocalUIDoc

IMPLEMENT_DYNCREATE(CLocalUIDoc, CDocument)

BEGIN_MESSAGE_MAP(CLocalUIDoc, CDocument)
//{{AFX_MSG_MAP(CLocalUIDoc)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

BEGIN_DISPATCH_MAP(CLocalUIDoc, CDocument)
//{{AFX_DISPATCH_MAP(CLocalUIDoc)
// NOTE - the ClassWizard will add and remove mapping macros here.
//     DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_DISPATCH_MAP
END_DISPATCH_MAP()

// Note: we add support for IID_ILocalUI to support typesafe binding
//  from VBA.  This IID must match the GUID that is attached to the
//  dispinterface in the .ODL file.

// {1D779144-366F-11D1-9A80-444553540000}
static const IID IID_ILocalUI =
{ 0x1d779144, 0x366f, 0x11d1, { 0x9a, 0x80, 0x44, 0x45, 0x53, 0x54, 0x0,
0x0 } };

BEGIN_INTERFACE_MAP(CLocalUIDoc, CDocument)
        INTERFACE_PART(CLocalUIDoc, IID_ILocalUI, Dispatch)
END_INTERFACE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLocalUIDoc construction/destruction

CLocalUIDoc::CLocalUIDoc()
{
// TODO: add one-time construction code here
EnableAutomation();

AfxOleLockApp();
}

CLocalUIDoc::~CLocalUIDoc()
{
AfxOleUnlockApp();
}

BOOL CLocalUIDoc::OnNewDocument()
{
if (!CDocument::OnNewDocument())
return FALSE;

// TODO: add reinitialization code here
// (SDI documents will reuse this document)

return TRUE;
}



/////////////////////////////////////////////////////////////////////////////
// CLocalUIDoc serialization

void CLocalUIDoc::Serialize(CArchive& ar)
{
if (ar.IsStoring())
{
// TODO: add storing code here
}
else
{
// TODO: add loading code here
}
}

/////////////////////////////////////////////////////////////////////////////
// CLocalUIDoc diagnostics

#ifdef _DEBUG
void CLocalUIDoc::AssertValid() const
{
CDocument::AssertValid();
}

void CLocalUIDoc::Dump(CDumpContext& dc) const
{
CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLocalUIDoc commands
```

LocalUIDoc.h

```
// LocalUIDoc.h : interface of the CLocalUIDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_LOCALUIDOC_H__1D77914E_366F_11D1_9A80_444553540
000__INCLUDED_)
#define
AFX_LOCALUIDOC_H__1D77914E_366F_11D1_9A80_444553540000__IN
CLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000


class CLocalUIDoc : public CDocument
{
protected: // create from serialization only
CLocalUIDoc();
DECLARE_DYNCREATE(CLocalUIDoc)

// Attributes
public:

// Operations
public:

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CLocalUIDoc)
public:
virtual BOOL OnNewDocument();
virtual void Serialize(CArchive& ar);
//}}AFX_VIRTUAL

// Implementation
public:
virtual ~CLocalUIDoc();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
//{{AFX_MSG(CLocalUIDoc)
// NOTE - the ClassWizard will add and remove member functions here.
//    DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

// Generated OLE dispatch map functions
//{{AFX_DISPATCH(CLocalUIDoc)
// NOTE - the ClassWizard will add and remove member functions here.
//    DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_DISPATCH
DECLARE_DISPATCH_MAP()
DECLARE_INTERFACE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_LOCALUIDOC_H__1D77914E_366F_11D1_9A80_444553540
000__INCLUDED_)
```

LocalUIView.cpp

```
// LocalUIView.cpp : implementation of the CLocalUIView class
//

#include "stdafx.h"
#include "LocalUI.h"

#include "LocalUIDoc.h"
#include "LocalUIView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView

IMPLEMENT_DYNCREATE(CLocalUIView, CView)

BEGIN_MESSAGE_MAP(CLocalUIView, CView)
//{{AFX_MSG_MAP(CLocalUIView)
// NOTE - the ClassWizard will add and remove mapping macros here.
//    DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
// Standard printing commands
ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView construction/destruction

CLocalUIView::CLocalUIView()
{
// TODO: add construction code here

}

CLocalUIView::~CLocalUIView()
{
}

BOOL CLocalUIView::PreCreateWindow(CREATESTRUCT& cs)
{
// TODO: Modify the Window class or styles here by modifying
//  the CREATESTRUCT cs

return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView drawing

void CLocalUIView::OnDraw(CDC* pDC)
{
CLocalUIDoc* pDoc = GetDocument();
ASSERT_VALID(pDoc);

// TODO: add draw code for native data here
}

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView printing

BOOL CLocalUIView::OnPreparePrinting(CPrintInfo* pInfo)
{
// default preparation
return DoPreparePrinting(pInfo);
}

void CLocalUIView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
// TODO: add extra initialization before printing
}

void CLocalUIView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
// TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView diagnostics

#ifdef _DEBUG
void CLocalUIView::AssertValid() const
{
CView::AssertValid();
}

void CLocalUIView::Dump(CDumpContext& dc) const
{
CView::Dump(dc);
}

CLocalUIDoc* CLocalUIView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument-
>IsKindOf(RUNTIME_CLASS(CLocalUIDoc)));
        return (CLocalUIDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CLocalUIView message handlers
```

LocalUIView.h

```
// LocalUIView.h : interface of the CLocalUIView class
//
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_LOCALUIVIEW_H__1D779150_366F_11D1_9A80_44455354
0000__INCLUDED_)
#define
AFX_LOCALUIVIEW_H__1D779150_366F_11D1_9A80_444553540000__I
NCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CLocalUIView : public CView
{
protected: // create from serialization only
CLocalUIView();
DECLARE_DYNCREATE(CLocalUIView)

// Attributes
public:
CLocalUIDoc* GetDocument();

// Operations
public:

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CLocalUIView)
public:
virtual void OnDraw(CDC* pDC);  // overridden to draw this view
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
//}}AFX_VIRTUAL

// Implementation
public:
virtual ~CLocalUIView();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
//{{AFX_MSG(CLocalUIView)
// NOTE - the ClassWizard will add and remove member functions here.
//    DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in LocalUIView.cpp
inline CLocalUIDoc* CLocalUIView::GetDocument()
{ return (CLocalUIDoc*)m_pDocument; }
```

```
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_LOCALUIVIEW_H__1D779150_366F_11D1_9A80_44455354
0000__INCLUDED_)
```

Mainfrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "LocalUI.h"
#include "MainFrm.h"
#include "CreatePurchaseOrder.h"
#include "AddInventoryItem.h"
#include "DeleteInventoryItem.h"
#include "CreateWO.h"
#include "WO_Statu.h"
#include "NMaint.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
//{{AFX_MSG_MAP(CMainFrame)
ON_WM_CREATE()
ON_COMMAND(ID_LOCALUI_TRACE_ENABLE, OnLocaluiTraceEnable)
ON_COMMAND(ID_LOCALUI_TRACE_DISABLE, OnLocaluiTraceDisable)
ON_COMMAND(ID_CMMSSCREENS_PURCHASEORDERS_CREATE,
OnCmmsscreensPurchaseordersCreate)
ON_COMMAND(ID_ADD_INVENTORY_ITEM, OnAddInventoryItem)
ON_COMMAND(ID_DELETE_INVENTORY_ITEM, OnDeleteInventoryItem)
ON_COMMAND(ID_CREATE_WO, OnCreateWO)
ON_COMMAND(ID_WO_STATUS, OnWOStatus)
ON_COMMAND(ID_CMMSSCREENS_WORKORDERS_NEXTMAINTENAN
CE, OnNM)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
ID_SEPARATOR,
// status line indicator
ID_INDICATOR_CAPS,
ID_INDICATOR_NUM,
ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
// TODO: add member initialization code here

}
CMainFrame::~CMainFrame()
{
}
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
return -1;

if (!m_wndToolBar.Create(this) ||
!m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
TRACE0("Failed to create toolbar\n");
                return -1;      // fail to create
                }
if (!m_wndStatusBar.Create(this) ||
!m_wndStatusBar.SetIndicators(indicators,
  sizeof(indicators)/sizeof(UINT)))
                {
TRACE0("Failed to create status bar\n");
return -1;      // fail to create
                }

m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);

return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
return CFrameWnd::PreCreateWindow(cs);
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

void CMainFrame::OnLocaluiTraceEnable()
{
// TODO: Add your command handler code here
ui_msg = TRUE;
}

void CMainFrame::OnLocaluiTraceDisable()
{
// TODO: Add your command handler code here
ui_msg = FALSE;
}

void CMainFrame::OnCmmsscreensPurchaseordersCreate()
{
//              int rc = 0;

// TODO: Add your command handler code here
if (create_PO_pkg.DoModal() == IDOK) {
TRACE("..Exit Create PO Dialog\n");
                }
}
void CMainFrame::OnAddInventoryItem()
{

int rc = 0;
if (a_inv_i_pkg.DoModal() == IDOK) {
TRACE("..Exit Add Inv Item Dialog\n");
rc = invoke_a_inv_i();
TRACE("..Return value = %d\n", rc);
}

}

void CMainFrame::OnDeleteInventoryItem()
{

int rc = 0;

if (d_inv_i_pkg.DoModal() == IDOK ){
TRACE("..Exit Del Inv Item Dialog\n");
rc = invoke_d_inv_i();
TRACE("..Return value = %d\n", rc);
                }
}
void CMainFrame::OnCreateWO()
{

//int rc = 0;
if (create_WO_pkg.DoModal() == IDOK ){
TRACE("..Exit Create WO Dialog\n");
}
}

//void CMainFrame::OnCreateWo()

void CMainFrame::OnWOStatus()
{
```

```
//              int rc = 0;

if (WO_Status_pkg.DoModal() == IDOK ){
TRACE("..Exit WO Status Dialog\n");
                }

}
void CMainFrame::OnNM()
{

//              int rc = 0;
if (NM_pkg.DoModal() == IDOK ){
TRACE("..Exit NM Dialog\n");
                }
}
```

Mainfrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if
!defined(AFX_MAINFRM_H__1D77914C_366F_11D1_9A80_444553540000
__INCLUDED_)
#define
AFX_MAINFRM_H__1D77914C_366F_11D1_9A80_444553540000__INCLU
DED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
CMainFrame();
DECLARE_DYNCREATE(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CMainFrame)
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
//}}AFX_VIRTUAL

// Implementation
public:
virtual ~CMainFrame();
#ifdef _DEBUG
virtual void AssertValid() const;
virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
CStatusBar  m_wndStatusBar;
CToolBar    m_wndToolBar;

// Generated message map functions
protected:
//{{AFX_MSG(CMainFrame)
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void OnLocaluiTraceEnable();
afx_msg void OnLocaluiTraceDisable();
afx_msg void OnCmmsscreensPurchaseordersCreate();
afx_msg void OnAddInventoryItem();
afx_msg void OnDeleteInventoryItem();
afx_msg void OnCreateWO();
afx_msg void OnWOStatus();
afx_msg void OnNM();
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_MAINFRM_H__1D77914C_366F_11D1_9A80_444553540000
__INCLUDED_)
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
//              LocalUI.pch will be the pre-compiled header
//              stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

StdAfx.h

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if
!defined(AFX_STDAFX_H__1D77914A_366F_11D1_9A80_444553540000_
_INCLUDED_)
#define
AFX_STDAFX_H__1D77914A_366F_11D1_9A80_444553540000__INCLUD
ED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from
Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                     // MFC support for
Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_STDAFX_H__1D77914A_366F_11D1_9A80_444553540000_
_INCLUDED_)
```