24. VBCE and Its Application in Time Study of Timber Harvesting Machines

24.1 VBCE

The Visual Basic for Windows CE (VBCE) Toolkit is Microsoft's answer to simplifying the process of creating applications for handheld computers. By augmenting their popular Visual Basic product with the VBCE add-in, Microsoft has provided a tool that enables anyone experienced with VB to immediately begin to develop handheld applications.

VBCE toolkit is an add-in for the Visual Basic design environment. It modifies and augments the VB IDE to provide the tools that you will use to create CE applications.

Software Requirements

In addition to the VBCE Toolkit you will need the following software:

- Windows NT 4.0
- Service pack 3 for NT
- VB V5.0 (either Enterprise Edition or Professional Edition)
- MS CE Services

The VBCE Toolkit is designed for use with Windows NT operating system. While you can install the Toolkit under Windows 95/98, this is not a supported configuration, thus will be error prone and lacking in several of key features.

What is Included with the VBCE Toolkit?

Once the VBCE Toolkit is installed on the system, the Visual Basic integrated design environment, or IDE, is modified to provide a set of tools for creating CE applications. Remember that VBCE Toolkit only adds functionality to the VB IDE and it does not remove all existing functionality that is not applicable for use when creating CE applications.

New Project Type

The first thing that you will have with the VBCE Toolkit occurs when you start Visual Basic. As the VB IDE is loaded the New Project dialog is displayed. This dialog, shown below, contains new project types:

- Windows CE Formless Project
- Windows CE HPC Project
- Windows CE HPC Pro Project

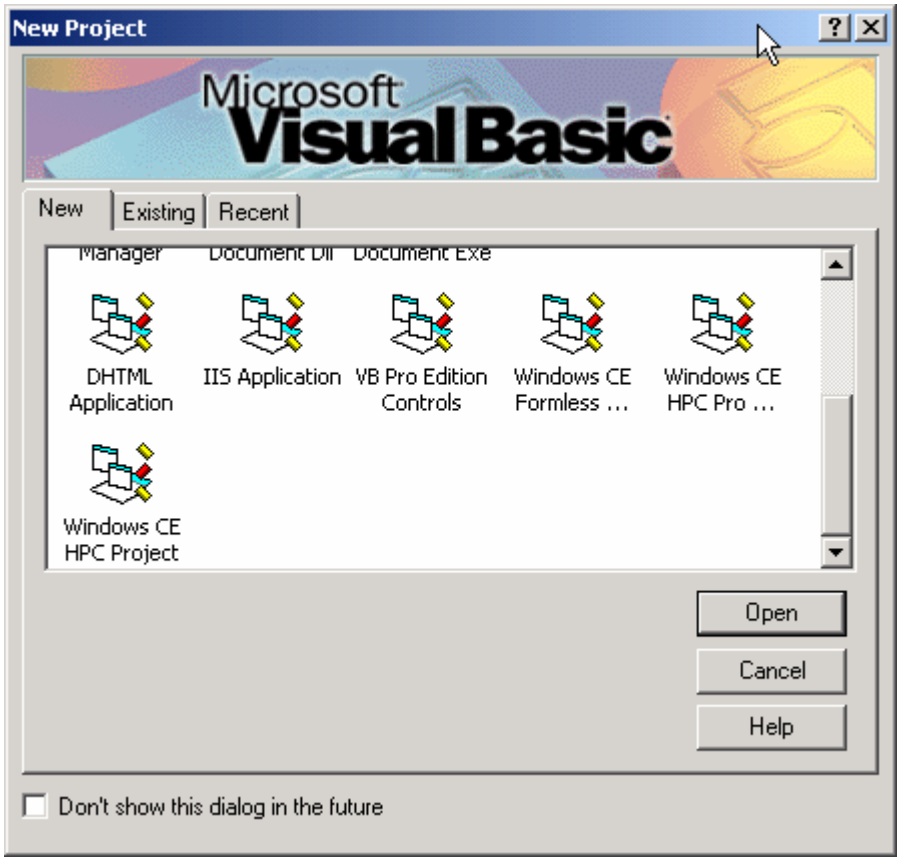

Figure 1. VB new project dialog.

VB Menu Structure Modifications

Of all of the modifications that the VBCE Toolkit makes to the Visual Basic IDE the most notable are those that are made to the menu structure. Several of the menus – the File, Project, Debug, Run, and Tools menu contain new items. Furthermore, there is a whole new menu, Windows CE, which incorporates a set of tools that aid in the development of CE applications. Remember that the modified menu structure only appears when you are working with a Windows CE project.

Differences Between VB and VBCE

Is VBCE a new programming language? Yes, it dose share a common core language with VB. However, there are many differences between the two languages. These differences will impact how you go about creating CE applications. There are quite a few differences out of there. Here are some of them:

(1) VB and VBCE use different ActiveX controls – there are a number of controls included in VB that are missing from VBCE. The controls themselves are physically different and in many cases have different

properties, methods, and events. The point is that do not assume that the VBCE control will work just like the equivalent VB control does.

(2) Single data type in VBCE – VBCE only supports a single data type – Variant. Variants are special in the fact that they can hold a variety of types of data. However, we can use Is functions such as IsDate and IsNumeric to check data types.

(3) Use different method to exit your application – End in VB and App.end in VBCE.

(4) Use a single stand module (.bas) in VBCE.

(5) There is no MDI form in VBCE.

(6) Limited error and debugging capabilities in VBCE.

24.2 VBCE Example

The process of creating a VBCE application has many similarities to the process to create the VB application for Windows 98/2000 or NT. Tasks such as creating the project, building the user interface and coding event procedures are identical between VB and VBCE. However, some steps such as testing in the emulator and on the handheld are unique to the CE environment.

The follows are the basic steps to create a VBCE application:

(1) Create a new project
(2) Set the project properties
(3) Construct the user interface
(4) Set the properties of forms and controls
(5) Add code to procedures
(6) Run and test the program

To create a new VBCE project:

(1) Start Visual Basic - VB IDE will be displayed along with the New Project Dialog, select Windows CE HPC Pro Project, and then click Open button (Figure 2).

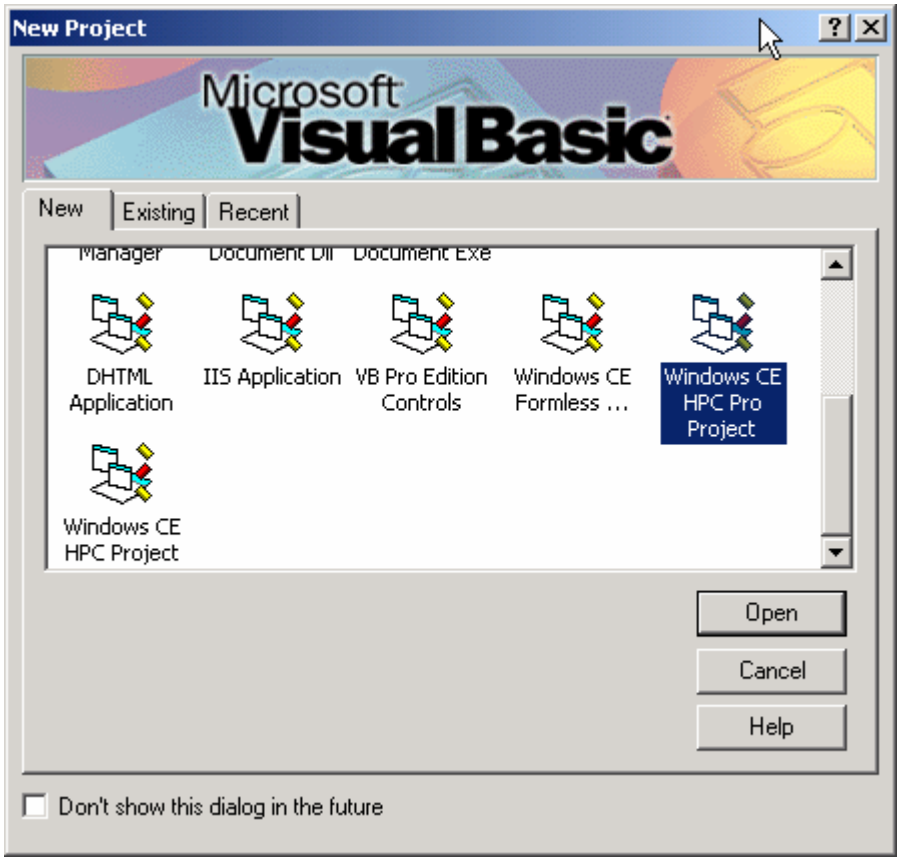

Figure 2. Start a new VBCE project.

(2) Setting the project properties – The Project Properties dialog will be displayed upon creation of a new VBCE project (Figure 3). You can return to this dialog later by selecting the Project Properties menu from the Project menu.

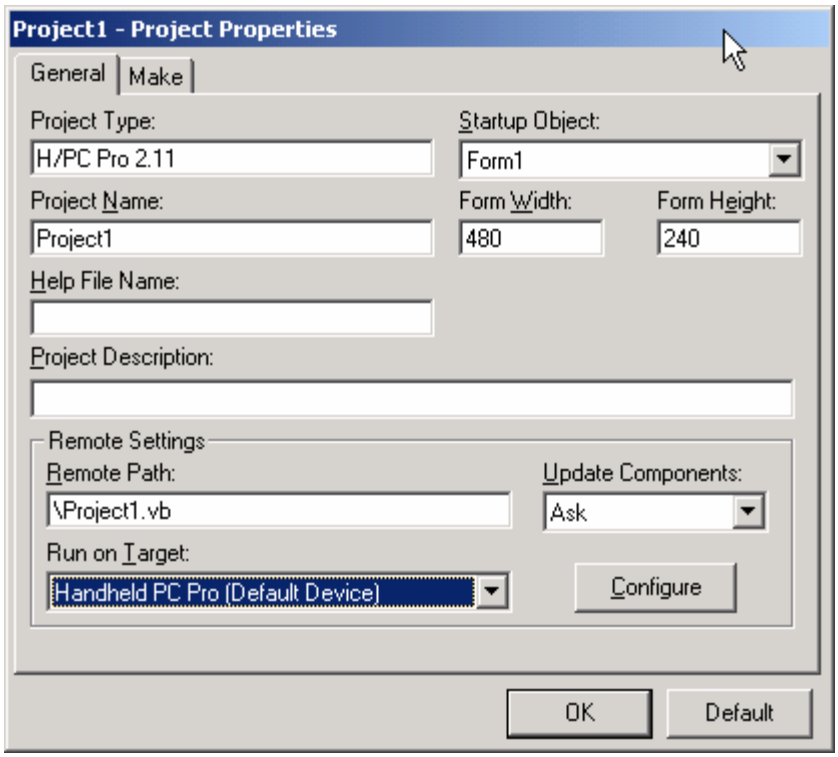

Figure 3. Dialog of project properties.

(3) Constructing the user interface – After specifying the project name and others, click OK, you will have a new project that is composed of a single form (Figure 4).

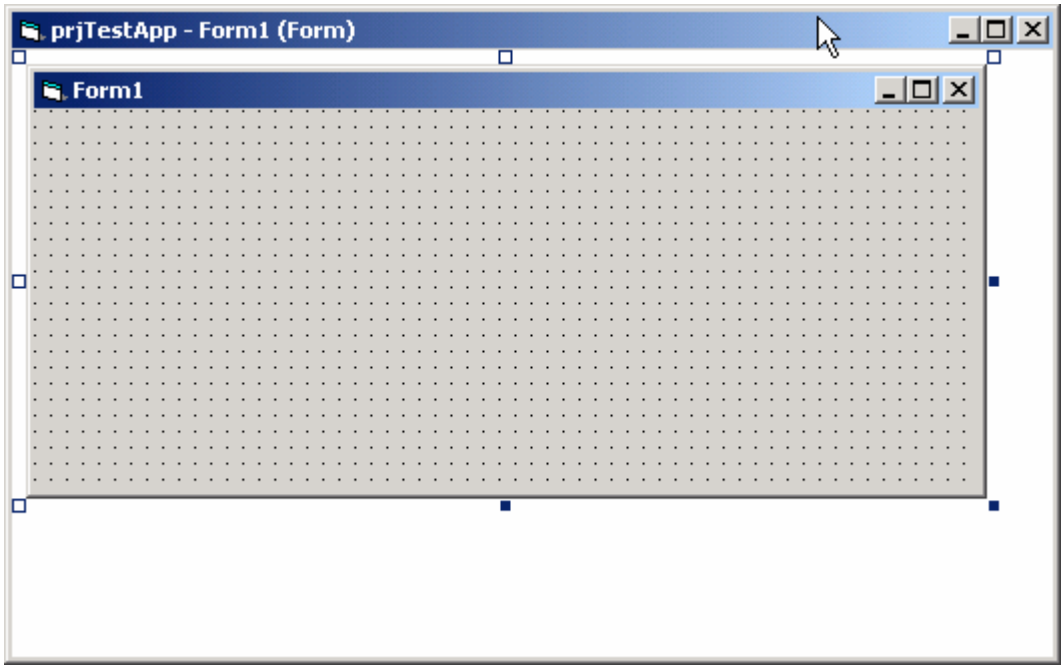

Figure 4. VBCE interface.

(4) Coding procedures – it is the same as programming in VB.
(5) Running and testing the program – there are two ways to test your program:
    a. To run on your handheld
    b. To run in emulator

24.3 VBCE Application in Time Study

Time study is "a set of procedures for determining the amount of time required, under certain standard conditions of measurement, for tasks involving some human, machine, or combined activity". Throughout the years, time studies have been conducted in different way:

(1) Stopwatches and paper
(2) Video camera
(3) DOS-based handheld
(4) Global Positioning Systems (GPS)

Objectives

The objectives of this application are to:

(1) develop a handheld time study system with MS Windows-based graphical user interface (GUI) for timber harvesting operations,
(2) adapt a relational database as the backend for time and factor data storage in the system, and
(3) build an interface module for time and factor data communication between desktop PC and handheld PC (HPC) using ActiveX Data Object (ADO).

System Structure

This time study system consists of three major parts: the handheld system, a GUI on the desktop PC, and a data storage component (Figure 5). The handheld system is used to edit species, design harvesting functions and variables, and collect site, elemental time, and variable data. The GUI component on the desktop provides the interfaces and functions needed to transfer data between the handheld and the desktop PC, and to manipulate and export the data for later analyses. The data storage component is a typical relational database containing tables of the time study data.

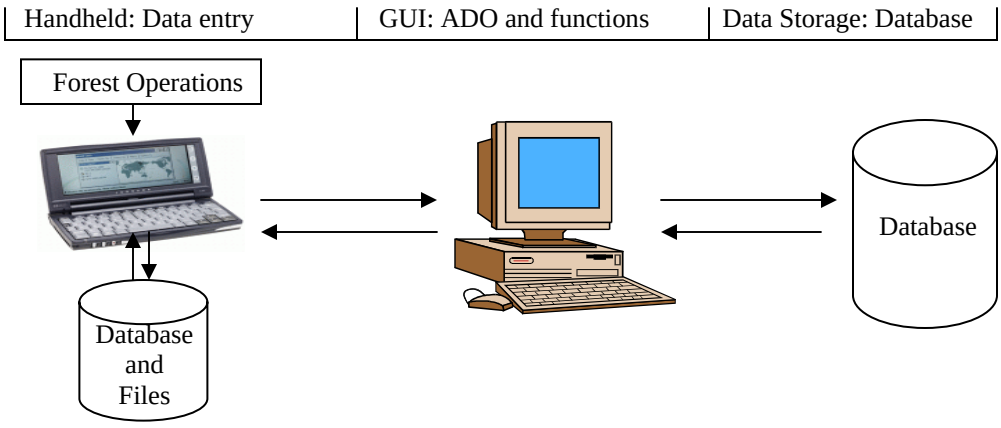

Figure 5. Architecture of the time study system.

The handheld system was written with Microsoft VB CE, which runs under Microsoft Windows CE environment. It contains two modules – design and collect (Figure 6). The design module in this system includes functions that allow the design work to be done on either the PC or the handheld. It provides the user with the option to enter or edit tree species to be used in the study site.

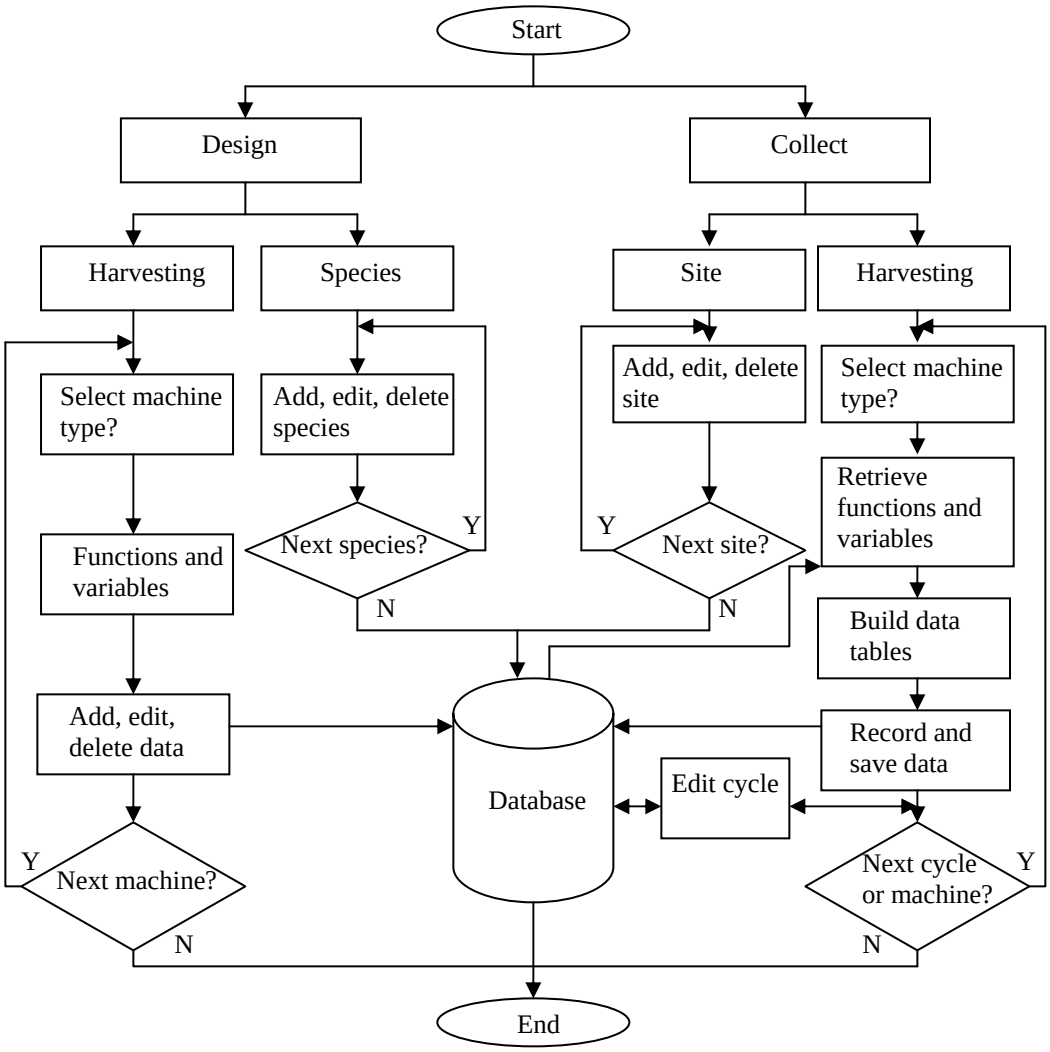

Figure 6. Flowchart of the handheld-based time study system.

Harvesting functions refer to the procedures or steps involved in the work cycle of a harvesting machine. For example, chainsaw felling may require four steps: walk to tree, acquire, fell, and delimb/top. The system allows the users to define functions for a specific machine. Harvesting variables are the factors that affect harvesting operations and elemental times. For example, DBH, height of the tree, and distance between harvested trees are the variables for chainsaw felling in addition to site effects. Once the time study design is completed, the collect module can be invoked and will retrieve the

information entered in the design module. Supporting help files using html-based architecture are also provided for this handheld time study system.

The data transfer interface module was written with MS VB V6.0 under MS Windows 98 or NT environment. ADO CE application programming interface (API) was employed to conduct data transfer via a dynamic link library (DLL) – adofiltr.dll (8). This DLL contains two functions, DesktopToDevice() and DeviceToDesktop(), which are used to transfer data or copy tables. It runs on the desktop PC, not the handheld. The desktop initiates and controls the transfer process. The key requirement for this transfer process is to have the same table schemas on both desktop and handheld. The ADO CE data transfer feature has a solid set of tools for transferring data. While the manual method for copying tables does not offer the control needed by most applications, the programmatic method does. This feature allows the transfer of complete tables between devices rather than requiring the synchronization of individual records.

A relational database model was used for holding harvesting functions, variables, and time study data in the handheld system, which was implemented based on the entity-relationship (ER) model (Figure 7). The relational database model presents the data as a collection of tables. Instead of modeling the relationships in the data according to the way that it is physically stored, the structure is defined by establishing relationships between data entities. Basically there are six data entities in the model - *harvesting functions, variables, site, species, time track*, and *felling/skidding/forwarding/yarding*. Each entity has its own attributes. For example, the *harvesting functions* entity has function ID and name, and machine type attributes. Entities are related using relationships such as “has,” “contains,” and “associates” in the model. A derived attribute *Elapsed_time* was used in time track entity, which is derived from *Start_time* and *End_time* attributes. Attributes belonging to a key are underlined for an entity set. Cycle number, machine type, function start time, end time, elapsed time, and associated harvesting variables are automatically recorded. Harvesting functions, variable data, and species are stored in separate data tables in the design module, which are identified by their primary keys and harvesting machine types. In the collect module, harvesting functions and variables can be queried and retrieved for a specific machine type on which another data table is created for storing functions, variables, and elemental times. Species information is also retrieved for data entry. The site data table contains general information such as site number, name, location, slope, and weather conditions about the logging site. Site number is used as a foreign key to associate site information with other data tables created in collect module.

The *time track* entity is used to track the start and end times of each element in a work cycle. It can also be used as a backup data table for felling, skidding/forwarding, or yarding data entities. The *felling/skidding/forwarding/yarding* entity is designed to store time study data of felling, skidding, forwarding, or yarding depending on the type of logging operation being studied. Data schemas of the main data storage on the desktop PC are the same as those used in the handheld system in order to facilitate the data transfer.

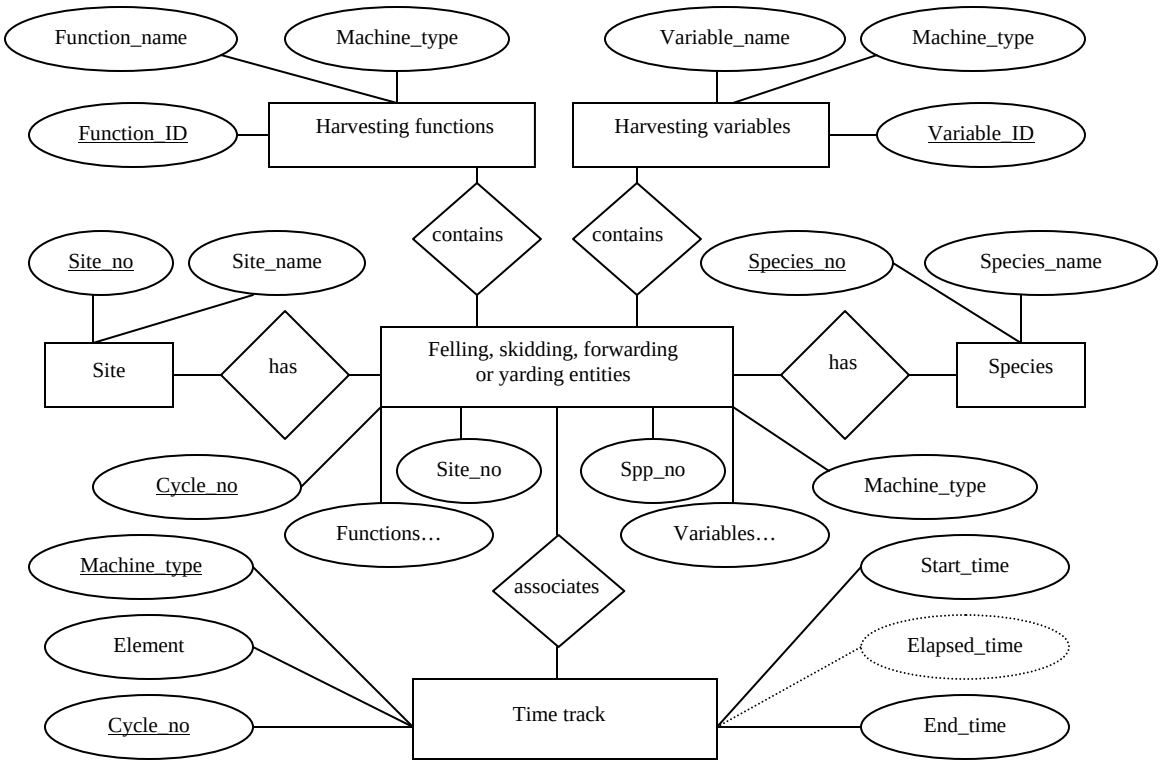

Figure 7. ER data model of the handheld time study system.

System Implementation

Design Module

Three functions were implemented in this module for designing and editing species, harvesting functions, and variables (Figure 8). Data were edited and saved into the database in the design phase and are then retrieved for uses in the data collection phase. The system also allows the user to navigate the database for a species, function, and variable. While navigating to a specific data record in the list, the user can edit or delete the current record.

Harvesting functions and variables were implemented to allow the user to design and edit their names and associate them with machine types. Data fields, including function/variable ID, function/variable name, and harvesting type, are used in such a data table.

Collect Module

There are three required fields: site name, site slope, and study date to collect data for a site (Figure 8). The user is required to fill out these three data fields for this site. The site number will automatically be increased and recorded when a new site is added. The "*Site No.*" will be retrieved later when the user starts to collect time study data and will be saved together with these data as a foreign key in the database.

Collecting elemental times and variable data is the ultimate objective of time studies. When invoking the collect module to collect harvesting elemental times and variable data, all the useful information entered under the design module can be retrieved and employed (Figure 8).

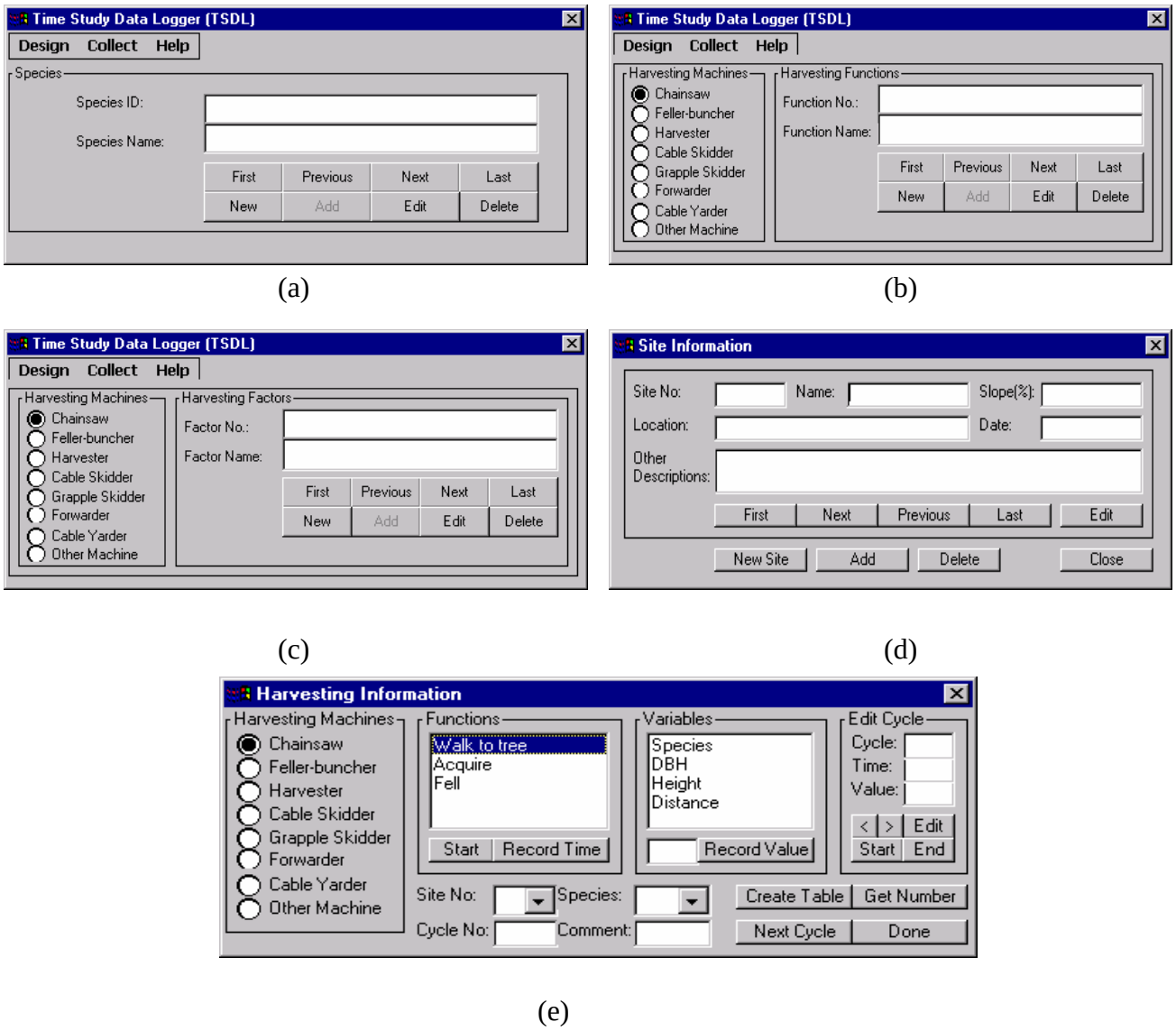

Figure 8. Main forms in the handheld-based time study system.

Elemental times and variables are saved in a database table whose data structure is created based on the parameters entered in the design module. The handheld system can check the data table status to determine whether the table was created or if the existed table's data structure is consistent with the current data. If the table does not exist or its structure is not consistent, a new table can be created.

In order to associate the site information with the time study data, a site number in the "*Site No.*" combo box must be selected. To record elemental time for a function, the user needs to select the function from the "*Function*" list box by simply clicking this

function. Hit "Start" when this function starts and hit the "Record Time" button once the function ends. Repeat the above procedures for any other functions in the list. If the user is not sure what function the logger is going to perform next, the program allows the user to click the "*Start*" button first, then select the correct function when it is identified, and finally click "*Record Time*" when the function is completed. The system also allows the user to record a function multiple times in sequence and the elapsed time for this function is then accumulated accordingly. To record a value for a variable, the user needs to select the variable in the "*Variable*" list box using the same procedures outlined for selecting a function. Then, the user can simply type a value in the text box beside the "*Record Value*" button and click this button. The value for the selected variable is recorded. Repeat the procedures for other harvesting variables. A brief comment can also be recorded in the system, which is especially useful to record a comment for a cycle or the delay reason in that cycle.

Another option is also provided to enter the variable's value. If the handheld does not have a keyboard or the user does not like to use the keyboard on the handheld, she/he can use the "*Species*" combo box to select a species by clicking the species required if the variable is tree species. If the variable is numeric, the user can click the "*Get Number*" button and a data input form will be invoked. Then, the user simply clicks number buttons and the "*Enter*" button to get the required number.

Once the recording is completed for the current work cycle, the user would click the "*Next Cycle*" button to save the current work cycle data to the database. Elemental time is recorded in seconds and can be converted to minutes when the user exports the data for analysis. Units for harvesting variables can be defined by the user.

An editing function for the data in a cycle is provided in the data collection module. The system is implemented to allow the user to go back to any previous work cycle being recorded, then edit the elemental times and variables or fill out the missed data in the cycle. Since it requires connecting to the same data object from two different processes at the same time, the system was implemented not to allow performing two or more data manipulation events concurrently, but rather sequentially in order to avoid mutating table errors.

This functionality is especially useful for the time study of chainsaw felling. For example, the sawyer may harvest one tree and complete all the functions before walking to the next tree to be harvested. In some cases, however, the sawyer might acquire and fell a tree, then walk to another tree and acquire and fell it, and then go back to the first tree to delimb and top it. To edit the cycle data, the user first needs to navigate to a cycle using "<" or ">" buttons and then use the following procedures: (1) select the function or variable from the list boxes, (2) modify their values in "*Time*" and "*Value*" boxes, and (3) click "*Edit*" button to update the cycle data (Figure 4e). The user can also use the "*Start*" and "*End*" buttons in Edit Cycle frame box to record the elapsed time for the selected function instead of entering a value in "*Time*" box.

Transfer Data

A data transfer module was implemented on the desktop PC, which provides three basic functions: (1) transfer data from HPC to PC, (2) update main data storage, and (3) empty data tables copied on the HPC. For the sake of data security, this module was

designed to run the above events sequentially. First, the data tables on the HPC are copied to a temporary database on the PC. Then, the tables in the temporary database are appended to the related tables in the main database. Finally, the system empties the data tables on the handheld in preparation for the next time study. Meanwhile, the user can decide whether or not the event should be activated in each step.

Coding Procedures

Data Manipulation

ADOCE data objects are used in the project for data manipulation.

```
'Data objects in General Declaration
Option Explicit
Dim rdFactor As ADOCERecordset
Dim rdFellF As ADOCERecordset
Dim nFunction, nTempFunc As Integer
Dim nFactor, nTempFact As Integer
Dim blAddNew, blFtAddNew As Boolean

'Define species
Dim rdSpecies As ADOCERecordset
Dim nSpp As Integer

'In Form_load procedures
Private Sub Form_Load()

   On Error Resume Next

   Set rdFellF = CreateObject("ADOCE.RecordSet")
   Set rdFactor = CreateObject("ADOCE.RecordSet")
   rdFellF.Open "SELECT * FROM tblFunction ORDER BY FID ASC", "\My
Documents\TimeStudy\TimeStudy.cdb", 1, 3, 1
   rdFactor.Open "SELECT * FROM tblFactor ORDER BY FtID ASC", "\My
Documents\TimeStudy\TimeStudy.cdb", 1, 3, 1
   'Call GetNofFunction
   nFunction = getRecMaxNumber(rdFellF)
   Call ShowFFunction
   nFactor = getRecMaxNumber(rdFactor)
   Call ShowFactor(rdFactor)
   cmdAdd.Enabled = False
   cmdFtAdd.Enabled = False
   fraFellF.Visible = False
   fraFactor.Visible = False
   blAddNew = False
   blFtAddNew = False
   'Define species
   Set rdSpecies = CreateObject("ADOCE.RecordSet")
  rdSpecies.Open "SELECT * FROM tblSpecies ORDER BY SpID ASC", "\My
Documents\TimeStudy\TimeStudy.cdb", 1, 3, 1
   nSpp = getRecMaxNumber(rdSpecies)
   Call ShowSpecies(rdSpecies)
```

```
    fraSpecies.Visible = False
    cmdSpAdd.Enabled = False

    Call ErrorHandler

End Sub
```

References

Roof, L. 1998. Professional Visual Basic Windows CE Programming. Wrox Press Ltd., Birmingham, UK.

Wang, J., J. McNeel, and J. Baumgras. 2003. A computer-based time study system for timber harvesting operations. Forest Products Journal. 53(3): 47-53.