"Delivering Bug-Free Applications:
Strategies & Techniques for Reducing the Testing Burden

Whil Hentzen
Hentzenwerke

Types of Bugs

Programming Errors

Programming errors include syntax errors, mismatched constructs, and other misuse of the language.

Logic Errors

Logic errors include missing branches of decision trees, bad assumptions about program flow and non-tested cases.

Business Rule Errors

Business Rule errors are mistakes or misinterpretations of the rules that govern the relationships between the data sets of the user.

Data Errors

Data errors are caused by data sets containing (or not containing) the data they are assumed to contain. Empty tables, missing values and bad data are three examples.

Version Conflicts, Hardware and Other Annoyances

This area is the catch-all for errors that appear when the application runs on your system but not on the user's machine. To avoid this, the up front specifications should include the version of operating system, FoxPro, and third party tools, and the range of the equipment (processor, memory, video, printers).

An Ounce of Prevention - Avoiding Bugs in the 1st Place

Conventions and Standards

Consistency

Case & continuation

Table and Key design

Field typing

Memvar typing and scoping

Comments and unfinished business

Starting out with a Pristine Environment

Memory

Tables

Output

Keyboard/Mouse

Outside files: Library, Procedure, On Error, Help

Restoring the environment after a crash

Single function in COMMON

Version and platform independent

Mimics FoxPro CONFIG.FP/W and application setup code

Defensive Programming

Declare variables at the top - not on the fly

Save and restore the environment - tables, indexes, record pointers

Code for the case that will “never happen”

Make all comparisons case insensitive

Make no assumptions about the data

Provide the ability for the system to recover from errors

Creating Bug-Free Reusable Functions

Document on the fly, and do it properly - especially calling syntax & parms

Use FoxPro's capability to accept fewer parms than were declared in the function

Overload the parameters to a reasonable degree

Plan for additional parms in future revs

Test the parameters that are passed to the function

Declare all memvars in your function private

Pull references to specific files when possible

Return the environment to where it was when the function started

Test the black box

The best black boxes follow the time-honored "One entrance, one exit" adage

Multiple Debug windows

View

Debug

Trace

“WAIT WINDOW”

Testing with Multiple Data Sets

Directory structure

Switching between test and live data sets

Choosing a data set in use

System looks for the default data set in SYS_SYST.

A different data set can be chosen by passing a parameter when loading the application.

Handling pathing during development and production use

After saving the current directory and path, switch to the desired data directory.

Restoring the original path and directory when finished

The Testing Process

Creating Test data

Reusable data sets

Populating tables

- Empty tables
- Bad data
- Boundary conditions

Large Volumes: Make My Data

A Testing Methodology

Objects

Functions

Data

Rules

Integration Testing

Don’t Test It Yourself!