




Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 1 of 9

1.0 POLICY/PURPOSE

It is the policy of Santa Barbara Applied Research (SBAR) to employ consistent programming practices to produce high-quality software. Good programming practices are an essential part of reducing costs and increasing quality. The purpose of this guideline is to provide recommendations for creating and developing software that follow good programming practices. Following these practices will reduce the time necessary to develop and maintain software.

2.0 SCOPE

This guideline applies to all SBAR personnel developing software in any programming or scripting language. It is particularly important for software delivered to a customer, but applies equally well to programs developed for internal use.

3.0 REFERENCES AND DEFINITIONS

3.1 References

ISO 9001: Quality Management Systems-Requirements, Third Edition (2000-12-15)

- ISO 9001 Element 7.3.2 (Design and Development Inputs)


3.2 Definitions

Functional Area Manager (FAM): A senior supervisory individual who is responsible for the leadership, direction, and overall success of an area of the company, such as finance, human resources, contract administration, engineering, operations and maintenance, logistics, quality, special projects/contracts, etc.

Quality Assurance (QA): All the planned and systematic activities implemented within the quality system and demonstrated, as needed, to provide adequate confidence that an entity will fulfill its quality requirements.

Software Life Cycle: The steps that a software development project goes through, starting with the requirements phase and ending with the software acceptance and delivery phase.



Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 2 of 9

4.0 RESPONSIBILITIES

4.1 Functional Area Manager (FAM)

The FAM is responsible for the overall management of software projects and programming efforts. The FAM:

- Provides guidance and direction on software development and design.
- Assigns programming tasks, as necessary.
- Acts as liaison between SBAR and client.

4.2 Programmer/Analysts

Programmer/Analysts are responsible for the design, creation and maintenance of software applications. Programmer/Analysts:


- Provide technical expertise in software-related areas
- Provide analysis of software design
- Aid in cost estimates for contractual purposes
- Provide training and instruction to personnel as required
- Design, create, and maintain software
- Meet with clients for reviewing purposes or demonstrations of the product.

4.3 Programmers

Programmers are responsible for the creation and maintenance of software applications. Programmers:

- Provide support in the design of software.
- Provide support in writing programs as directed.
- Make revisions to software as directed.
- Test and provide quality assurance (QA) for applications.
- Meet with clients for reviewing purposes or demonstrations of the product.



Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 3 of 9

5.0 REQUIREMENTS/PROCEDURES:

5.1 General

The practices outlined below ensure that programs are written clearly; aiding in the design, development, and maintenance of the software life cycle. The methods listed below provide guidelines in developing well-written, robust applications.

5.2 Program Division

Programs shall be divided into meaningful modules or objects. These modules/objects encapsulate a particular functionality or procedure, providing programs that are less complex and easier to develop and maintain. Should portions of code become too large for clear understanding, they should be further subdivided into smaller functions or procedures. This guideline should be strictly adhered to unless program functionality prevents it; for example, if program speed is greatly affected.

5.3 Naming Conventions


Names given to variables, functions, procedures, buttons, icons, etc. shall be specific to their functionality. Currently, most programming environments allow the use of long variable names; this should be taken advantage of, as practical. These names should be easy to understand by someone looking at the program for the first time. All buttons, icons, etc. should be given names and not left as the default, e.g. "button id 193" is changed to "next button". This guideline adds clarity to the program and makes software maintenance more manageable.

5.4 Header Comments

Comments shall be provided at the beginning of each procedure, function, handler, etc. These comments tell what the procedure, function, handler, etc. does, but not necessarily how it does it. Comments can also include inputs and outputs expected, name of person writing the code, date code was written, and tracking information for version or change made, if applicable.

5.5 Inline or Code Comments

Comments shall be provided in any area of code that might be confusing or where understanding of the program can be enhanced.

Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 4 of 9

5.6 Indentation

Indentation of two or more spaces is used to enhance program readability. Code that is too long to fit on one line has all subsequent lines indented at the same level of indentation under the first line. All control structures (if then else, while loops, etc.) have internal statements and expressions indented. The contents of procedures, functions, handlers, etc. shall also be indented. This guideline helps in the reading and understanding of programming logic. For ToolBook applications, see Attachment 1, “Software Style Guides for ToolBook.”

5.7 Whitespace

Blank lines shall be used wherever practical to separate procedures, functions, control structures, etc. These lines shall be used to separate portions of code that logically should be kept together from the rest of the code. This guideline is followed whenever practical; however, sometimes programming space does not allow for this. If used effectively, whitespace can aid in quickly being able to understand program flow and logic.

5.8 Data Extraction

All important data used to verify the correct operation of the computer program must be available. If the data is not readily apparent, provisions for data extraction must be included in the program (e.g., debug statements that can be switched off using a compiler switch). Switched debugging statements must contain only code to display or record values. Data can be displayed to a screen or recorded in a separate data extraction file. If possible, data extraction files shall be in a user-friendly format.


When required by contract, data extraction is non-switch code that records predetermined data to one or more data extraction files in the predetermined format.

5.9 Tracking Changes

Whenever changes are made to existing code that has been placed under configuration management, the following apply:

- Comments delineate the beginning and end of the code change.
- Previous code is commented out instead of being deleted.
- Comments record the programmer and date of change.
- If available, comments record the reason for change, such as a software change proposal (SCP).



Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 5 of 9

5.10 Programming Style

While concise code is preferred, the follow-on programmer may not be as well versed in the programming language. Therefore, the following style guidelines shall be used.

- Each programming statement shall do only one thing. Avoid compound statements.
- Each programming statement is placed on a separate line.
- Compiler “features” or “bugs” shall not be used.
- Architecture “features” or “bugs” shall not be used.
- Higher-level languages shall be used unless the lower level language or assembly language offers significant advantages.


5.11 Records

Review of this guideline should be recorded on the Design Control Checklist for every software project as a “Design and Development Input”.

5.12 Training

Training for this guideline is required for anyone writing software within the scope of this procedure. Training records shall be maintained in the Training Attendance Form Binder, and/or the individual’s personnel file.



Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 6 of 9

Attachment 1. Software Style Guides for ToolBook

SBAR is in the business of selling software. That software should not only function properly, but also should look professionally manicured and standardized. In addition to the satisfied clients that we will help create, many of these guides will tend to speed up integration and debugging. Implementing these guides costs little or no extra time if they are implemented during the design phase of a project. The following guides are a start toward that end.

1. "END" statements: All "END" statements should be followed by text denoting what it is ending.


EG: END if
 END step
 END linkdll
 END conditions
 END while
 END buttonclick
 END in

This is especially valuable for long structures that cannot be totally seen on the CRT when integrating and/or debugging. Additionally, TOOLBOOK does a much better job of locating syntax errors when this text is available.

2. Indenting. Each of the "END"s in item 1 constitute the end of a structure. Each statement within a structure should be indented by using the tab key (equivalent to four spaces).

EG: TO HANDLE buttonclick
 IF a = 1
 put "1" into text of field "abc" of page "stores"
 put "1" into text of field "abc" of page "sStores"
 ELSE
 put "0" into text of field "abc" of page "stores"
 put "0" into text of field "abc" of page "sStores"
 END if
 END buttonclick

Special Case: CONDITIONS
 WHEN <condition 1>
 <action 1>
 WHEN <condition 2>
 <action 2>
 ELSE
 <action 3>
 END conditions

Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 7 of 9

3. Continuation Lines: Statements that are too long to be typed on one line should be split up at logical breakpoints (see example). Subsequent lines should be indented by two spaces instead of the four spaces generated by the tab key.

```
IF (<long condition>)\
    and (<long condition>)\
    and (<long condition>)\
    <action>
END if
```

4. Capitalization: The eyes naturally tend to gravitate to upper case characters. All code should be lower case except key words as follows:

- a) The word “STEP” and the word “END” for looping structures.

```
EG:          STEP n from 1 to 2
              hide rectangle “abc” of page(storespg[n])
              END step
```

- b) The words “TO HANDLE” and the associated “END”.

```
EG:          TO HANDLE buttonclick
              code .....
              END buttonclick
```

- c) The word “LINKDLL” and the associated “END”.

```
EG:          LINKDLL “tb40win.dll”
              code .....
              END linkdll
```

- d) The words “IN VIEWER” and the associated “END”.


```
EG:          IN VIEWER “smallmdi”
              code .....
              END in
```

- e) The word “WHILE” and the associated “END”.

```
EG:          WHILE a < 10
              code .....
              END while
```

- f) The word “CONDITIONS” and the associated “END”.

```
EG:          CONDITIONS
              WHEN a is 0
                  code .....
              WHEN a is < 0
                  code .....
              ELSE
                  code .....
              END conditions
```

Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 8 of 9

5. Use of quotation marks:

Preferred EG: hide rectangle "abc" of page "sStores" ----preferred.


The following examples are not preferred for the following reasons:

- a) These statements will execute slightly slower. Without the quotes, TOOLBOOK considers abc & sStores to be variables, causing one extra level of processing.
- b) Global edits are more dangerous without the quotes and in some cases may be impossible.
- c) The "of page < >" reduces the possibility of program errors especially in the case of the PTT software where many duplicate item names exist on different pages.
- d) The "of page < >" adds to the self-documenting capabilities of TOOLBOOK.

Non-Preferred: hide rectangle abc of page sStores

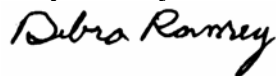
Never: hide rectangle abc



Approved:		Date: 19 May 2004
Title: Software Programming	Rev NC	Page 9 of 9

PREPARATION, REVIEW, AND APPROVAL OFFICIALS

Prepared By:

Debra Ramey
Software Systems Manager

Reviewed By:

Blake Monson
CBT Specialist

Reviewed By:

Ralph Chapman
Quality Assurance Officer, Ventura

Approved By:

M.T. Schmoll
Director, Corporate Programs

Approved By:

Grace Vaswani
President/CEO

CONTROLLED DISTRIBUTION LIST

Copy No.	Copy Custodian
Master (Electronic)	Software Systems Manager
Copies	SBAR Web Site