

The Benefits of Object-Based Authoring Systems for Small Businesses

Carl E. Keller, Jr.

Assistant Professor of Accounting
Coastal Carolina University

Benjoe A. Juliann

Assistant Professor of Computer Science
Coastal Carolina University

Abstract

Object-oriented authoring systems present many opportunities for small companies to develop their own customized software. This article describes object-oriented authoring systems, gives examples of how they simplify programming, and concludes with several illustrations of how small businesses may take advantage of these software development systems.

Introduction

Managers of small businesses may have the following wishes: "I wish that I didn't have to take time out to train a new employee again," "I wish that we could do a multimedia presentation to land that large contract next week," or "I wish that our company had a computer program designed to keep only the records that we need, in a format that we could understand." Such concerns are common to both large and small businesses, but in the past, only large companies had the capital and the manpower to develop software programs that would allow these wishes to become reality. However, object-based authoring systems are becoming a cost effective tool used by small companies to write their own software applications.

In the past, most small businesses purchased software "off the shelf," believing that customized software programs would be too expensive. The typical small business could not afford to hire a programmer, nor contract the project out to a programming company. Furthermore, most small businesses did not possess employees that had enough programming skill to write even the simplest application packages. Programming skill was required due to the complexity of the programming languages and their related applications, not to mention the difficulties of debugging a program. Even if a business had a skilled programmer, typically the company could not afford the capital investment (in terms of man-hours) to allow the employee to write the program.

The advent of object-based authoring systems allows small companies to overcome these obstacles to owning their own customized software. Many authoring systems use a fourth generation programming language which is much easier to use than earlier programming languages. These fourth generation languages are structured to be similar to people's speech patterns and are more accepting of different

terms to execute commands. The object/event orientation of these authoring systems allows some programming to be done with a mouse. Thus, programmers can visually see what they are creating as they are writing the program. This fact allows authoring systems to decrease the initial need for programming skill and to reduce the time to write an application. In addition, the more programming that you do with an authoring system, the faster you become at writing applications. Best of all, while there is the initial cost of buying an authoring system, (probably between \$300 and \$4,000), the rights to any programs that you write with the system are usually owned by you!

The first section of this paper explains the technical aspects of an object-based authoring system. The second section provides examples of how easy programming can be when a person is using an object-based authoring system. Finally, the paper discusses potential uses and benefits of authoring systems for small businesses.

Object-Based Authoring Systems

Several authoring systems are available to the public, although you will not usually find these packages on the shelves of your local computer software store. Exhibit 1 displays a partial list of authoring systems. Each authoring system operates uniquely and has different capabilities. The following paragraphs describe the basics of object-based authoring systems.

An authoring system is computer software that allows developers to create programs/applications with writing a minimal amount of programming code. *Object-based systems* and *object-based programming languages* usually refer to systems with *objects* with *inheritance* (Cardelli, 1985). The objects are the building blocks of any program created using an object-based system. An object can be any individual, identifi-

able item, unit, or entity that exists in the application. An object has a state, a behavior, and an identity. Examples of objects that typically exist in a program are navigation buttons, text fields, data or record fields, and graphic objects.

So how do these objects make life easier for unskilled programmers? Suppose you wanted to create a navigation button that would allow a user of your program to go from one screen (a.k.a. "page") to the next screen, in a sequential manner. Some authoring systems (such as Macromedia's Authorware 3.0) may have an icon that you select to insert the forward navigational button. Thus, you simply click the mouse where you want the button to appear, then click the mouse on the appropriate icon, and the button will appear ready for you to test. These are referred to as icon-driven authoring systems.

Other authoring systems require a little more work, but they may grant more flexibility. For example, to create a forward navigation button with a different authoring system (such as Asymetrix's Toolbook 3.0), the programmer would follow these steps:

1. Click the mouse to select a type of button from the tool palette (there are several types to choose from).

2. Use the mouse to point to where the button will be located, then "click and drag" to shape the button to the desired size.

3. Select "button properties" from the "Object" menu. Type in a name for the button (e.g., "Forward1") and a caption to appear on the button for users (e.g., "Forward") in the pop-up menu. Then select the "Script" button to type in the following code:

```
to handle buttonClick
    send next page
end buttonClick
```

4. Select Update and Save Script with the mouse. Then exit the button properties menu and you have created your forward navigation button.

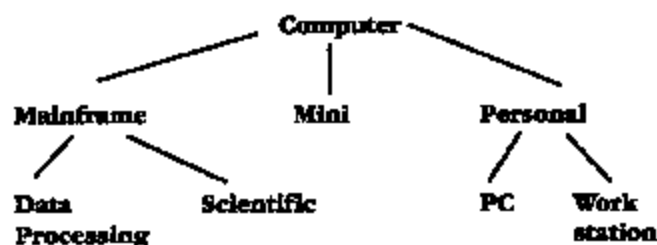
Note two things about programming with this second authoring system. First, one can easily see that the object (e.g., the button) has the three characteristics mentioned earlier. The button has a specific identity via the name of "Forward1." The button is either in an activated or an inactivated state, and performs the specific behavior of sending the user to the next screen (page) when activated. Second, simply by writing a little bit of code, a programmer can gain a lot more flexibility. To illustrate this point, suppose you would like a program that would either:

1. Automatically play music for the user as they view the next screen,
2. Record the amount of time the user spends reading the new screen, or
3. Automatically print a document as the user goes to the next page.

Any of these events (or any combination of them) can be activated by the same forward navigation button with only a few extra lines of code using the second

authoring system. Icon-driven authoring systems would not allow the navigation button to control any of these events because the icon generates the code. Thus, most authoring systems are set up to require some programming code to provide the programmer the ability to customize each application that they develop.

Within an object-oriented environment, the structure and behavior of similar objects are defined in their common *class*. A class also specifies the *inheritance* of an object. Inheritance provides a natural classification for objects. The naturalness comes from the fact that we use concepts, classification, and generalization to understand and deal with the complexities of the real world. (See the example below on using computers.)



Inheritance is a relationship between classes where one class is the parent (a.k.a. base, superclass, or ancestor) class of another. In a classical object-oriented environment, inheritance is a relationship between classes only. Thus, most object-based authoring systems contain a hierarchical organization (often referred to as the *object hierarchy*). Exhibit 2 displays the object hierarchy for the Asymetrix's Toolbook authoring system. Object hierarchy is important because parent objects control child objects to some extent. For example, a page (parent object) controls buttons (child objects) that are located on it. If you cannot view the page, you cannot use the button. Furthermore, while all objects can send and receive event messages to each other, the messages have to pass through the object hierarchy. Therefore, the hierarchy becomes important when the order of events is significant. The authoring system will perform events based upon the order the objects receive their messages. Thus, in the Toolbook object hierarchy, an event that sends separate messages to a page and a button on that page will find that the page message is performed first.

Ease of Programming

The traditional programming environment creates applications by first defining data and then writing the related procedures to handle or manipulate the data. In the object-oriented environment, a program is developed by first creating objects, and then defining the object's behavior by writing program code for that specific object. Recalling the previous example of the forward navigation button, the reader should

note that first the graphical features (size, shape, and style) of the button were created, and then the code was written to tell the system what to do when the button was activated. Thus, the object-oriented environment does not seem as abstract to the programmer as traditional programming environments.

However, object-oriented authoring systems may contain many other features that make programming easier. Returning to the earlier example of the navigation button, suppose you wish to change the button's location. Simply click on the button and drag the button to the new location. The button will look and work the same as before the move. If you have other pages that require similar buttons, you do not have to go through all the initial steps again. Simply click on the button, then select the copy command from the menu bar, move the cursor to the locations where similar buttons are desired, and select the paste command from the menu bar. The new buttons will have all of the same properties as the original button. The new buttons should be given new names (instead of all of them being "Forward 1", perhaps "Forward 2," "Forward 3," etc. . . .) from the Object/Button properties menu.

The programming languages of authoring systems contribute to the ease of new software development. For example, the code for the navigation button was:

```
to handle buttonClick
    send next page
end buttonClick.
```

Instead of using the words "send next page," the same results could be achieved with "send next," "go to next page," or "go to page 2." Thus, these newer programming languages are very close to our everyday language. Furthermore, this illustration demonstrates several ways of writing code that will still allow the computer to perform the same desired function, which makes writing programs less tedious.

In addition, some authoring systems have the ability to allow you to create your own graphics, or import them from another source. You can create text within the application under a variety of fonts and styles. If your system has multimedia capabilities you can incorporate sound and video within your application. Some authoring systems even have programming recorders. A recorder will write the program code as you perform the desired actions with the mouse and keyboard. Recorders are particularly useful when creating animation or printing documents.

For example, the creation of a document (including record fields) is fairly easy with some authoring systems. However, writing the code to automatically print the document when a user clicks on a button becomes tedious. With a programming recorder activated by clicking the mouse on an icon, one merely clicks on a button that has been created, goes through the print menu (similar to selecting options in word processing software), and then deactivates the recorder. The recorder will generate

code similar to Exhibit 3.

Conclusion

Even if people find that developing programs with object-based authoring systems is easy, can small businesses still find practical uses for authoring systems? Consider the following situations where small companies may find creating their own programs useful:

1. Applications can be developed to train, educate, and even test the employees. For example, programs can be written to teach employees about company products, office procedures, and information protocol (from journal entries in the accounting system to database management). The advantage to the small business is that training can be consistent across employees. Even more important, the training will be accomplished on the computer, therefore the time of a supervisor (or another employee) is not required for the training.

2. Presentations made by company employees can be enhanced, particularly if the company has a system with multimedia capabilities. A multimedia presentation can be an effective lecture that integrates sound, video, animation, and text. The authoring system would allow you to choose the music, the video, and the text to present your topic (or product) in the most appropriate manner.

3. A small business may require specialized records or documents that are not easily developed through typical database software. Some authoring systems will allow the programmers to develop their own record fields, documents, and functions. Other authoring systems will allow users to integrate the new program with "off the shelf" software (such as word processing, spreadsheet, or database packages). In any case, the company should be able to develop an application that should meet their needs.

4. A small business may find other uses for authoring systems. As authoring systems are becoming more flexible and powerful, a company will find that they are limited only by their imagination and time.

In addition, object-based development systems have several programming advantages for small businesses. Object-based authoring systems use the naturalness of the "object concept," which helps to increase the speed of application development, provide a consistent and seamless package, and increase the program's quality. Object-based authoring systems emphasize modeling real world situations, which helps provide practical applications. Furthermore, object-oriented development systems have a natural resistance to change: system objects change infrequently while processes and procedures can be changed frequently, which supports the reuse of software (or software designs) and reduces development risks for complex systems.

Exhibits

EXHIBIT 1 Partial List of Authoring Systems*

Act III-Informatics Group, Incorporated	(203) 953-4040
Authorware-Macromedia, Incorporated	(800) 326-2128
Director-Macromedia, Incorporated	(800) 325-2128
Express Author 1.1-Research Triangle Media, Incorporated	(800) 282-2425
GUIDE Author-InfoAccess, Incorporated	(206) 747-3203
HyperCard-Apple Computers, Incorporated	(800) 776-2333
Icon Author-AimTech Corporation	(800) 289-2884
KSS: Author-Comware, Incorporated	(513) 791-4224
LinkWay Live!-EduQuest (An IBM Company)	(800) 426-4338
Media Master-Vision Imaging	(800) 292-4264
MediaMax-Videodiscovery	(800) 548-3472
Multimedia Toolbook-Asymetrix Corporation	(800) 448-6543
TBT Author-HyperGraphics Corporation	(800) 369-0002
Toolbook-Asymetrix Corporation	(800) 448-6543

*Adapted from *AuthorBase (NLM and ARI, 1995)*

EXHIBIT 2

Object Hierarchy* (in descending order)

1. Toolbook development software
2. System programs
3. Program currently being developed
4. Backgrounds
5. Pages
6. Groups
7. Buttons, field, graphic objects

*Adapted from *Toolbook User Manual*
(Asymetrix, 1994)

EXHIBIT 3

Programming Code from a Recorder

```
to handle buttonClick
    set printerStyle to pages
    set printerMargins to 1440, 1440, 1440, 1440
    set printerGutters to 360, 360
    set printerScaling to actual
    set printerBorders to false
    set printerArrangement to 1, 1
    set printerPageBitmap to true
    start spooler
        go to page 14
        print 1
    end spooler
end buttonClick
```

References

Asymetrix Toolbook User Manual. Version 3.0. 1994. Asymetrix Corporation, Bellevue, Washington.

Booch, Grady. 1986. "Object-oriented development." *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2, pp. 211-221.

Booch, Grady. 1994. *Object-Oriented Analysis and Design with Applications*, Second Edition. Benjamin-Cummings Publishing Company.

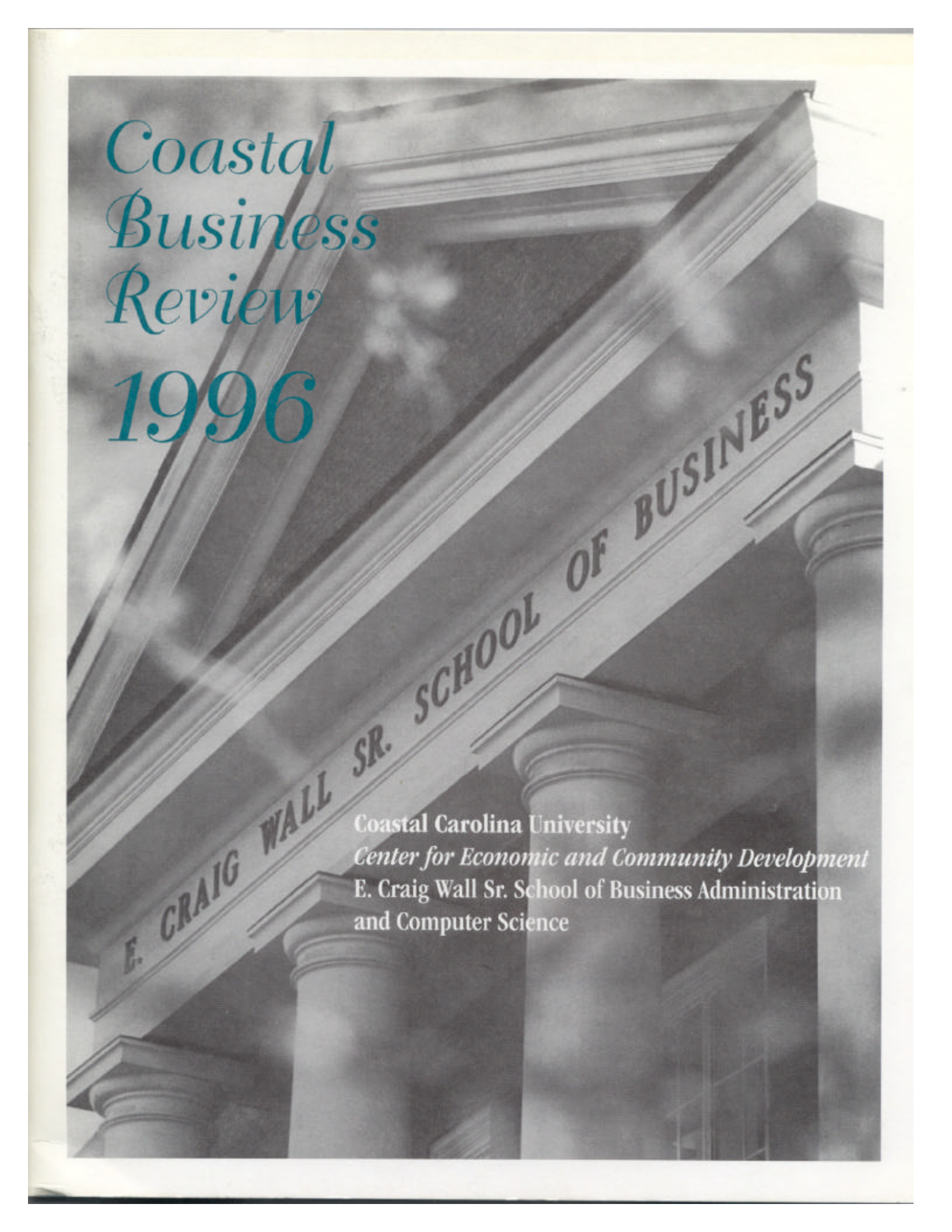
Cardelli, Luca. 1985. Amber. "In Cousineau, Curien and Robinet" (Ed.), *Combinators and Functional Programming Languages*, LNCS 242, Springer-Verlag, pp. 21-47.

Cardelli, Luca. 1988. "A semantics of multiple inheritance." *Information and Computation*, Vol. 76, pp. 138-164.

Jacobson, Ivar, Magnus Christerson, Patrik Jonasson, and Gunnar Overgaard. 1995. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Revised printing. Workingham, England: Addison-Wesley Publishing Company.

National Library of Medicine and the U.S. Army Research Institute. 1995. *The AuthorBase Alphabetical Listing*. (Downloaded from the Internet, <http://www.ncbi.nlm.nih.gov/authorb/idx/index.html>).

Stroustrup, Bjarne. 1987. "What is 'Object-Oriented Programming?'" In J. Bezivin, J-M Hullot, P. Cointe and H. Lieberman (Ed.), *Proceedings of ECOOP '87, LNCS 276*, Paris, France: Springer-Verlag, pp. 51-70.



*Coastal
Business
Review
1996*

Coastal Carolina University
Center for Economic and Community Development
E. Craig Wall Sr. School of Business Administration
and Computer Science

Coastal Business Review

edited by
Gregory L. Krippel, Ph.D.
Assistant Professor of Accountancy
Coastal Carolina University

Center for Economic and Community Development
E. Craig Wall Sr. School of Business Administration and Computer Science
Peter B. Barr, D.B.A., Dean

Coastal Carolina University
Conway, South Carolina

Volume 5-1996

From the Editor

This is the fifth annual issue of the *Coastal Business Review* as published by Coastal Carolina University's E. Craig Wall Sr. School of Business Administration and Computer Science. We have made every effort in the hope that this issue will be as well received as the previous issues have been.

This year we have provided an outlet for meaningful, interesting research with a varied range of articles from authors located in South Carolina and the Southeast.

We have articles of interest to a broad range of South Carolina and Grand Strand businesses. The articles are organized according to four broad interest areas: tourism, small business, medium to large business and business education.

We would like to invite readers of this journal to submit a paper for possible inclusion in the 1997 edition.

Gregory L. Krippel, Ph.D.

Editor

Assistant Professor of Accountancy

1996 Editorial Review Board

Robert B. Burney-Finance

James F. Eason-Accounting

Robert D. Nale-Management

Dennis A. Rauch-Marketing

The deadline for submissions for the 1997 edition is December 15, 1996.

The maximum length of papers submitted should be 20 double-spaced pages.

Please submit materials to:

Dr. Greg Krippel

Coastal Business Review

E. Craig Wall Sr. School of Business Administration and Computer Science

Coastal Carolina University

P.O. Box 261954

Conway, South Carolina 29528-6054

Table of Contents

Tourism Interest Articles

3

Marketing the Myrtle Beach Area:
Methodology and Preliminary Results of Its Success
Donna Smaldone • Tom Russo

9

Horry County—Emergence of Its Thriving
Tourism/Retirement-Based Economy
Robert E. Pugh • Robert T. Barrett

13

Predicting the Number of Tourists That Visit
A Vacation Destination
Thomas Secrest

19

Uncertainty in Economic Predictions
Benjoe A. Jullano

Small Business Articles

25

The Adoption of Document Imaging
Processing Systems in the Small Business Community
Gregory B. Turner • Mark Hartley

29

The Benefits of Object-Based Authoring
Systems for Small Businesses
Carl E. Keller, Jr. • Benjoe A. Jullano

33

The Home Office Deduction Post-Solimon
James R. Hasselback • Sberri Kraftchick

Medium to Large Business Interest Articles

41

Relative Efficiency of Electric Cooperatives in South Carolina:
An Application and Test of Data Envelopment Analysis
William B. Tankersley • Julia E. Tankersley

49

South Carolina's Governor's Quality Award and Its First Recipient
Lilly M. Lancaster

Educational Interest Articles

53

A Research Note On An Ongoing Examination
of Literacy Expectations
Robert D. Nale • Dennis A. Rauch

57

The Effectiveness of Undergraduate Education in Business Administration
Based on the Perceptions of Internal and External Constituencies
Virginia B. Leusen • Wilbur L. Garland

The Center for Economic and Community Development

The Center for Economic and Community Development was established in April 1989 as a separate unit of the E. Craig Wall Sr. School of Business Administration and Computer Science. The center's mission is consistent with the expressed goals of the university, to conduct programs of basic and applied research and programs of public service, to provide leadership, to act as a resource, and to improve the quality of life throughout the service area. To accomplish these goals, the center functions as a bridge between the university and the community, drawing upon the unique talents and expertise of faculty and students to serve economic and community needs. It provides five types of assistance: planning and decision support; technical assistance/applied research; educational programs; information, counseling, and referral services; and student involvement. Funds are provided by the Horry County Higher Education Commission and external sources ranging from private companies to nonprofit and government entities. These funds have enabled the center to undertake research and programs which provide major economic and social benefits to all of the citizens of the Waccamaw region.

Since its inception, the center has initiated or participated in more than 100 projects and investigations. Among the most significant are:

- Development of computer software and a computerized decision support system to assist planning efforts of the former Horry County Economic Development Board (now PARTNERS Economic Development Corporation) and management information systems for the Waccamaw Regional Planning and Development Council
- Assessing the economic effects of the closure of the Myrtle Beach Air Force Base
- Applied research efforts for Grand Strand Water & Sewer Authority, Burroughs and Chapin Company, Inc., WCI Investments, the cities of Conway Myrtle Beach, and many others
- Perceptual studies for nonprofit entities such as the United Way and Horry Cultural Arts Council
- Humanitarian efforts such as an assessment of living conditions in the Backport and Burgess communities and economic impacts of Hurricane Hugo
- Assessment and analysis of Horry county government operations for the Horry County Council and development of computerized data bases that contain regional economic data and statistics which are made available to the public upon request.

The Center continues to be an informed advocate of economic growth of Horry County as well as a participant in improving the overall quality of life in the Waccamaw region. The results of these activities continue to be valuable resources to all of the citizens of our service area.

The Advisory Board for the Coastal Center for Economic and Community Development includes local business people and those with substantial interest in community economic development issues. Serving on this board are:

Ronald B. Ingle
John P. Idoux
Peter B. Barr
Jack Hutchison
George Magrath, Jr.
Nancy Lee
Richard Smith
Deborah Brooks
Ruth Kearns
Pat Williams

Board of Visitors

E. Craig Wall Sr. School of Business Administration and Computer Science

Peter B. Barr, Dean

William J. Baxley, Jr.
William Benson
Lawton Benton
Virginia Biddle
John Coté, Jr.
W. Jennings Duncan
John Gandy
Michael C. Gerald
Stan Gibson
John Gilbertson

John C. Griggs
Gary Hadwin
W. Thomas Hale
Donald Hardee
Keith Hinson
Kenneth R. Hinson
J.J. Johnson
Charles Jordan
George Kosko

Nancy Lee
George N. Magrath, Jr.
Louis Henry Mense
J. Edward Norris, III
Clyde W. Port
Dr. Lee Proctor
Hayden Quattlebaum
Ernest W. Rahon
Charley Ray

Fred Richardson
Mack Singleton
Howard B. Smith, III
Brenda Spadoni
Samuel R. Spann, Jr.
Walter E. Standish, III
Frank Thompson, II
E. Craig Wall, Jr.
Charles L. Watson
Ralph Wilson