

`<title>peer to peer</title>`

an information technology newsletter
volume 4, number 2
summer 2001



why is this man smiling?

>> page 3 nola earns iso registration

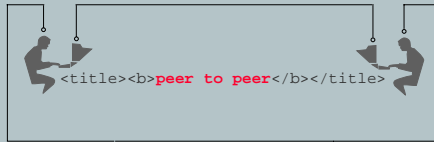
>> page 9 XP with Object Mentor >> page 10 entergy salutes pam seghers >

>> page 11 new puzzler >> page 12 nola help desk established



> page 5 narti kityakara on coherence and coupling >> page 10 usda salutes maria bertucci >> page 6 debugging development





volume 4, number 2
summer 2001

Peer to Peer is published by NOLA Computer Services, Inc. to recognize the accomplishments of our NOLA colleagues and to provide our clients with information concerning developments in the field of information technology.

NOLA offers a broad range of information services to companies and organizations in the public and private sectors. Our services include information management consulting, system architecture, and project management.

For inquiries, please contact us at one of the locations shown below, call toll free at (888) 488-1101, or visit our website at www.nolacom.com.

Peer to Peer wishes to thank the publishers of *Debugging the Development Process* (Microsoft Press) for generously allowing us to reprint portions of this book.

Trish Thomas edits, designs, and manages the publication of *Peer to Peer*. Syed Baber, Bill Gough, and Narti Kitiyakara contributed to this issue. Please send comments to peertopeer@nolacom.com.

www.nolacom.com

Corporate Headquarters
3535 Canal Street
New Orleans, Louisiana 70119
Voice: (504) 488-1111
Facsimile: (504) 488-9955

440 Louisiana Street, Suite 900
Houston, Texas 77002
Voice: (713) 236-7732
Facsimile: (713) 227-0423

Five Concourse Parkway, Suite 3100
Atlanta, Georgia 30328
Voice: (770) 804-5933
Facsimile: (770) 804-5801

1800 Diagonal Road, Suite 600
Alexandria, Virginia 22314
Voice: (703) 684-4454
Facsimile: (703) 548-9446



iso 9001 registered



NOLA
Founders
Stanley Jordan
(right) and
Richard Jackson

dear colleagues,

As you can see by the smiling face on our cover—we have earned ISO 9001 registration! We are truly proud of everyone whose hard work contributed

to obtaining our ISO-9001 registration, but ISO-9001 registration only supplies third-party recognition of the quality services we provide. We must continue to pay close attention to our customers to ensure the most coveted recognition—the recognition by our customers of a job well done.

To support NOLA in sustaining a high level of customer satisfaction, we have established the Corporate System Group. The Group's mission is to deploy processes and systems that enhance the effectiveness and competitiveness of both NOLA and the customers we serve. Among the Group's short-term goals are implementing a continual improvement program, expanding our new help desk to include all functions and projects, and incorporating eXtreme Programming into our software development methodology. Our Commercial Applications Group is presently using XP to develop our leading edge information systems. XP has allowed us to focus the creative energy of our programmers without stifling it. Through pair programming, a fundamental concept of XP, our junior staff is being exposed to lessons learned by our more senior staff members.

In addition to leveraging leading edge technologies in the commercial software products we continue to build, NOLA is engaged in several exciting consulting projects. We are assisting a major refinery with transforming their IT department into a responsive, customer-focused organization. We have designed, developed, deployed, and are currently hosting innovative web-based solutions for several global organizations.

As a result of our dedication to customer satisfaction, NOLA has continued to experience growth. To support this growth—and in the spirit of sustaining an employee-friendly environment—we will soon be expanding our corporate headquarters. In July, we purchased the building at 124 North Genois Street, which is directly behind headquarters at 3535 Canal Street. The North Genois Street building currently comprises residential units, but plans are underway to convert the building to NOLA office space that will accommodate our growing team and exciting new projects.

As we move forward, I hope we can strengthen the bonds between us while we make NOLA a great work environment for a team that is recognized for caring about colleagues as well as customers.

Richard Jackson

We did it!

nola's ISO odyssey by bill gough

We haven't seen much of Bill lately—he divides his time between managing corporate systems at NOLA headquarters and deploying web-based applications at the New York Independent System Operator. He now knows the Delta flight schedules for Albany and New Orleans by heart.



NOLA Computer Services is the first and only ISO 9001-registered software and consulting company based in Louisiana.

how?

It all started in the fall of 1997, when Bruce Woods formed NOLA's Software Engineering Process Group. Composed of NOLA personnel (see sidebar), the SEPG's mandate was to implement a quality system that conformed to the Capability

Maturity Model produced by the Software Engineering Institute. During the year that followed, the SEPG created plans and documents that covered most CMM Level 2 requirements.

In the fall of 1998, I assumed the role of SEPG manager, and Bruce began to investigate ISO registration. He decided that ISO 9001 was, in fact, a better fit than CMM for the full-service organization that NOLA was becoming—ISO 9001 has the broadest scope, spanning initial design, installation, and product support. NOLA executive management agreed.

Bruce researched organizations that could help NOLA achieve ISO 9001 registration, and NOLA chose the Center for International Standards and Quality at Georgia Tech. In July 1999, I returned from a client engagement to manage the registration effort full time. From July to December, Bruce and I attended biweekly CISQ classes where we learned how to document procedures, conduct internal audits, choose an independent registrar, and run a quality system. And so began NOLA's ISO odyssey.

Back in the office, momentum was building. NOLA's executive management team agreed on a quality policy: *We strive to deliver at a level that meets industry standards and exceeds the expectations of our customers.* In August, coach Dennis Kelly visited NOLA to assess the gap between ISO 9001 requirements and our existing implementation. He also supervised the initial internal audit.

In September, management and staff from all areas of the company began work

on a quality manual that would become the bible for our quality system. At a high level, the quality manual described how NOLA would meet ISO 9001 requirements. We also began working on procedures that would serve as the core of NOLA's quality system implementation, including recruiting and hiring, software development, consulting, and contract management.

The year 2000 brought introductory training to all NOLA staff, both on-site and at client sites. César Morataya and I conducted the first of many internal audits in February, then management reviewed the audit results as well as our progress toward completing documentation and implementation.

ISO does not certify companies; ISO certifies independent registrars who, in turn, audit companies like NOLA according to ISO standards. In May 2000, we selected BVQI as our registrar because they offered the best value to the company.

BVQI representative Larry Holt came to NOLA headquarters in August 2000 to conduct a pre-assessment audit, which revealed some implementation weaknesses in our new processes for hiring, contract review, and document and record management. We licked our wounds and made refinements in preparation for the "live" registration audit the following October.

The October audit lasted three days—three of the most nerve-racking days of my life! There we were, sitting right next to auditor David Burnett as he meticulously reviewed each and every one >> page 4

Sponsors
Stan Jordan
Richard Jackson

Internal Auditors
Donnelle Ireland
Clark Lizana
César Morataya (Lead)
Mark Smith
Andrea Washington

Procedure Authors
Paul Augustin
Richard Jackson
César Morataya
Jeff Neupert
Jay Shannon
Andrea Washington
Bruce Woods

Documents and Records Manager
Andrea Washington

Editor
Trish Thomas

SEPG
Shawn Antee
Clive Arlington
Maria Bertucci
Conrad Carriere
Joe Dartez
Bill Gough
Cesar Morataya
Pam Seghers
Trish Thomas
Bruce Woods

who made ISO happen at NOLA?

International standardization began in 1906 with the International Electrotechnical Commission. Twenty years later, the International Federation of the National Standardizing Associations was organized, focusing on mechanical engineering until 1942, when World War II took precedence. After the war, delegates from 25 countries formed a new organization “to facilitate the international coordination and unification of industrial standards.” This organization, ISO, began to function officially on February 23, 1947.

Today, ISO is a worldwide, nongovernmental federation of national standards bodies from 140 countries. ISO’s mission is to promote “the development of standardization and related activities in the world with a view to facilitating the international exchange of goods and services and developing cooperation in the spheres of intellectual, scientific, technological and economic activity.”

we did it! << page 3
of our responses to over 130 ISO requirements. In the end, he found only eight minor nonconformities. We made the necessary refinements and sent our resolutions to BVQI. Based on our responses, David closed the issues and recommended us for registration. Registration was granted on February 12, 2001.

does it all mean
what?

NOLA deliver the best possible solutions to our clients. Top management sets goals and our internal audit team periodically verifies that we’re on track to achieve those goals. Internal auditors frequently review our system to make sure we’re following documented practices and conforming to ISO 9001 standards. They document problems and follow them through to resolution using a corrective and preventive action system. Our corporate systems, staff augmentation, and software product development activities are documented and tracked by project managers and consultants.

Being registered means that NOLA is committed from the top down to continu-

We now have a management system that will help

ally improve the way we do business. It also means we have documented processes for many of our day-to-day operations. We follow proven software development methodologies to ensure accurate cost estimates, reasonable delivery schedules, and improved output quality. We have a nucleus of project managers who take customer ideas and transform them into successful realities. Frequent audits ensure that implementation follows a consistent process. We record, track, and act on customer complaints and internal process inconsistencies. Management meets regularly to make sure our quality system continues to benefit our company and our customers. All of this is at work every day, and it conforms to a standard that is recognized around the world.

I want to thank everyone who—over the last three and a half years—has played a role in building a quality system worthy of ISO 9001 registration. There is no way we could have accomplished this without the hard work and dedication of everyone involved. I especially want to thank our customers who supported our registration efforts with patient understanding when, at times, our off-site staff had to divert attention from their primary duties to participate. Thank you all. </end>

what is iso anyway?

Many people will have noticed a seeming lack of correspondence between the official title when used in full, International Organization for Standardization, and the short form, ISO. Shouldn’t the acronym be “IOS”? Yes, if it were an acronym—which it is not.

In fact, “ISO” is a word, derived from the Greek isos, meaning “equal”, which is the root of the prefix “iso-” that occurs in a host of terms, such as “isometric” (of equal measure or dimensions) and “isonomy” (equality of laws, or of people before the law).

From “equal” to “standard,” the line of thinking that led to the choice of “ISO” as the name of the organization is easy to follow. In addition, the name ISO is used around the world to denote the organization, thus avoiding the plethora of acronyms resulting from the translation of “International Organization for Standardization” into the different national languages of members, e.g. IOS in English, OIN in French (from Organisation internationale de normalisation). Whatever the country, the short form of the Organization’s name is always ISO

—Reprinted with permission from www.iso.ch

coherence, coupling, and refactoring

by narti kitiyakara

Narti Kitiyakara taught himself nearly a dozen programming languages before earning a B.S. and an M.S. in computer science from the American Institute of Computer Sciences. Narti joined the NOLA team in 1998 and is now a lead systems architect in the commercial applications group.



In previous articles I've tried to highlight some fairly concrete technologies and coding techniques that I hope have been of value to both coders and designers. I'd like to take a break from that for this article and discuss a more conceptual issue of importance to all coders during initial development and—especially—during maintenance: coherence and coupling.

The term *coupling* describes how tightly a piece of code is intertwined with the other code in the program. The higher the coupling, the more intertwined the code, and, consequently, the harder to understand. Why? Because the tighter the coupling, the more code you have to consider when you're trying to understand what is being done. Without *some* coupling we'd have a set of unrelated lines of code rather than a program, but your goal should be to keep coupling to a minimum. It's helpful to know the range of coupling that is possible, and what constitutes each point in the range. Roger Pressman, in his book *Software Engineering – A Practitioner's Approach* (McGraw-Hill, 1992), defines a spectrum of coupling ranging from “no direct coupling” at the low-end to “content coupling” at the high-end (see sidebar).

Niklaus Wirth once wrote that a good programmer must be able to jump between different levels of abstraction. It's good *coherence* that allows us to make this jump easily. Coherence describes the single mindedness of a piece of code *at a given level of abstraction*. Note the emphasis in that sentence. At its most detailed, even the simplest “Hello World!” program performs a multitude of different operations, but we almost never deal with the program at that level. Instead, we have compilers or interpreters that add levels of abstraction between us and the binary code that the computer actually executes, and we have code libraries that abstract out the string processing and I/O operations.

My experience has been that programmers tend to use that abstraction without

giving it any thought—yet it's worth thinking about. It's really amazing what that abstraction buys us. I can, for example, open a message box in Visual Basic simply by saying `MsgBox “My Message”, “My Title”` and never have to know that I've really opened separate windows for the title bar, the OK button, the control menu, and a bunch of other things of which I'm not even aware. If we had to know everything that went into opening one of these “simple” message boxes, we'd never do it.

I sometimes think that people want to struggle with their code. I know a lot of programmers (including myself) who feel as though they've scaled Mt. Everest when they get to the end of a major project. Even for the professional, a major programming effort requires the right tools—strong cohesion and low coupling are the two most important. Trying to write a major system without using them properly is like trying to climb Mt. Everest without carrying your oxygen—it can be done, but it's a pretty stupid thing to do.

As part of NOLA's adoption of XP methodology, I recently attended

a class on refactoring. Although I disagreed with the emphasis on mechanical refactoring, I still feel that it offered many useful techniques for raising cohesion and reducing coupling. The class is based on Martin Fowler's *Refactoring: Improving the Design of Existing Code* (Addison-Wesley, 1999), which is a whole cookbook of techniques. The one we seemed to use most in class was “Extract Method,” which helps cohesion and coupling by reducing the size of methods. >> page 8

level	description
NO DIRECT COUPLING	Occurs when there is no relationship between pieces of code.
DATA COUPLING	Occurs when two pieces of code communicate via parameters.
CONTROL COUPLING	Occurs when a value set in one piece of code controls the behavior of another piece of code.
EXTERNAL COUPLING	Occurs when a piece of code affects something external to the system. Examples of external coupling include writing to a file or a screen.
COMMON COUPLING	Occurs when variables are used over too large a scope. This often means global variables, but can also apply to local variables in large methods.
CONTENT COUPLING	Occurs when a piece of code makes use of data contained in another piece of code, or when an unstructured jump into the middle of another piece of code is executed. It is rare to find content coupling in modern programming languages.

“this book might make microsoft sound bad...”

worries author Steve Maguire in his preface to *Debugging the Development Process*—which happens to have been published by Microsoft Press—but he goes on to say, “I think people realize that Microsoft wouldn’t have reached its position of prominence in the software industry if the company were full of bozos. It isn’t.” We include this excerpt from *Debugging the Development Process* (Microsoft Press, 1994), as another in our series from quality assurance specialists.

While working with these teams, I discovered that they were all making the same fundamental errors and that they were perpetually repeating those errors. Not only that, once I’d gotten attuned to the mistakes those teams were making, I saw that even teams on successful projects were making those same fundamental errors—they just made the mistakes less often or had instituted countermeasures to overcome the effects of those mistakes.

In every group I worked with, I found that the project leads were spending nearly all of their time writing code and almost none of their time thinking about the project. The leads didn’t spend time trying to keep schedules on track, they didn’t look for foreseeable problems so that they could circumvent them, they didn’t work to protect other team members from unnecessary work, they didn’t pay particular attention to training other team members, and they didn’t set detailed project goals or create effective attack plans. The leads were spending too much time *working* when they should have been *thinking*.

In many ways, this state of affairs wasn’t really the fault of the leads. The leads hadn’t been trained to be leads. They were programmers who woke up one day to find themselves, or one reason or

another, plunked into lead positions. These new leads knew how to program well, but they didn’t know how to run projects well, so they focused on what they knew best and allowed their projects to run themselves—right into the ground.

on improving the product **focus**

Companies pay programmers to produce useful, high-quality products in a reasonable time frame. But programmers often get sidetracked into doing work that has nothing to do with creating a product. They, or their leads, fail to recognize a basic truth of product development:

Any work that does not result in an improved product is potentially wasted or misguided effort.

If you don’t immediately see why this point is so important, consider two extremes. Which programmer is more likely to produce a useful, high-quality product in a reasonable time frame: the programmer who regularly attends meetings, writes status reports, and is buried in e-mail or the programmer who uses all her time to research, design, implement, and test new features? Is there any question that the first programmer’s schedule will slip whereas the second, much more

focused, programmer not only is likely to finish on schedule but may even finish early?

I’ve found that groups regularly get into trouble because programmers are doing work they shouldn’t be doing. They’re spending too much time preparing for meetings, going to meetings, summarizing meetings, writing status reports, and answering e-mail. Some programmers initiate this kind of activity themselves. More often, such distractions are at the behest of a misguided lead.

One lead with whom I worked required every team member to send a weekly e-mail message reporting on the status of his or her work. The entire team would then meet for an hour or so to rehash what everybody had been doing and to discuss any external issues that had cropped up. After the meeting, anybody who had offered new information would have to write those thoughts down in another piece of e-mail and send it off to the lead.

Now this lead was just trying to be thorough. What he didn’t realize was that he was choking his team with a lot of pointless process work. Was it really necessary to have both status reports and status meetings? And what about those follow-up reports? Were they really necessary, or could they have been elimi-

HIGHLIGHTS

Debugging the Development Process

By Steve Maguire

Taken largely from his experiences with Microsoft development teams whose projects were in trouble, *Debugging the Development Process* is full of simple, practical methods for keeping development projects on schedule.

Companies have hired their programmers to focus on creating high-quality products, but



programmers can't do that if they're constantly pulled away to work on peripheral tasks. Make sure that every team member is focused on strategic work, not on housekeeping tasks; look for and ruthlessly eliminate any

work that does not improve the product.

Unfortunately, some housekeeping work is necessary, at least in larger companies. One way to keep such work to a minimum is to regularly ask the questions "What am I ultimately trying to accomplish?" and "How can I keep the benefits of the task yet eliminate the drawbacks?" Fulfill the need, not some overblown corporate process.

The benefits of establishing specific goals might not be easy to see, but it's easy to see the chaos that ensues in groups that don't set such goals. Yes, creating detailed goals can be tedious; but that up-front work is much less painful than leading a project that slips two days every week.

Every team member needs to know the coding priorities. Is maintainability important? What about portability? Size? Speed? If you want the code to reflect the goals for the product, you must tell programmers what trade-offs to make as they implement features. You must also establish quality bars so that team members won't waste time writing code that will have to be rewritten before you ship. The earlier you define the quality bars, the earlier you'll minimize wasted effort.

From *Debugging the Development Process* by Steve Macquire.
Reproduced by permission of Microsoft Press.
All rights reserved.

nated in 99 percent of the cases if the lead had simply taken better notes during meetings?

Obviously, your answers to such questions will depend on your particular corporate environment, but in the actual case I've just described, the only process work that ever turned out to have any value was the initial status report. I don't remember a single status meeting that was worth the time it took to attend, and every time the lead asked for follow-up reports I winced, thinking, "Why? They just told you what they thought."

I was only an occasional visitor to these regular status meetings, so I wasn't often affected by the status work. I always wondered, though, how much other unnecessary process work that group was routinely saddled with.

In his well-intentioned zeal to be thorough, that group's lead violated what I consider to be a fundamental guideline for project leads: *The project lead should ruthlessly eliminate any obstacles that keep the developers from the truly important work—improving the project.*

There's nothing earth-shattering about this observation, yet how many leads do you know who make it a priority to actively look for and eliminate unnecessary obstacles?

If the lead I've been talking about had been actively trying to eliminate unnecessary work, I'm sure he could have come up with a much simpler and more effective method of determining the state of his project. Having status reports, status meetings, and follow-up reports was overkill. *Don't waste the developers' time on work that does not improve the product.*

there's

always a better way
As a lead, I'm always asking myself one question, in all phases of the project.

What am I ultimately trying to accomplish?

I constantly ask this question because it's so easy to get sidetracked on work that isn't important. If you've ever spent more time formatting a memo—playing with fonts and styles—than you did writing the memo in the first place, you know what I

mean. In the moment, you get caught up because the work seems important, but if you step back and get some perspective, you see that it's the message that's important, not how artistic you can make it. We've already seen an example of misdirected effort in the status meetings and status reports I've talked about. How would you answer this question: *What am I ultimately trying to accomplish by holding status meetings and requiring status reports?*

Isn't the primary goal of gathering project status information to detect, at the earliest possible moment, whether the project is going astray? Think about that. Suppose all projects were finished exactly as scheduled—no project end date ever slipped, and nobody ever worked over time. Would anybody ever gather status information? Of course not. There'd be no reason to.

If the ultimate purpose of status meetings and status reports is to determine whether a project's >> page 11

“... every time the lead asked for follow-up reports I winced, thinking, ‘Why? They just told you what they thought.’”

example 1 refactoring

Listing 1 shows how coherence can make code easier to understand. On the surface, this function appears cohesive; its single purpose is to return an HTML drop-down based on the information in a database. Yet below the surface, it's not as cohesive as it first appears. It really operates at two levels of abstraction. At the high level it simply returns an HTML drop-down list. Look at the details, though, and you'll see that it really comprises two steps: generate a SELECT tag and generate zero or more OPTION tags.

The first step toward improving this code would be to make it more cohesive by separating the two functions. To do this, we would use Extract Method to remove the code that generates an OPTION tag. (See Listing 2.) This leaves us with two methods that are more cohesive than the original. I think everyone would agree that they are easier to understand and will be easier to maintain than the original. By removing the common coupling from getDDLString,

listing 1

```
private function getDDLString(byval strTable, byval code_id, byval_
field_name, byval dbcode, byval dbdescr)
    dim strDDL
    dim rs1
    if code_id = "" then
        code_id = 0
    end if
    SQL = "Select * FROM " & strTable & " order by " & dbdescr
    Set rs1 = Conn.Execute(SQL)
    strDDL = "<select size='1' name='" & field_name & "'>"
    do while not rs1.eof
        strDDL = strDDL & "<option value='" & rs1(dbcode) & "' "
        if cng(rs1(dbcode)) = cng(code_id) then
            strDDL = strDDL & " selected "
        end if
        strDDL = strDDL & ">" & rs1(dbdescr) & "</option>"
        rs1.MoveNext
    loop
    rs1.close
    strDDL = strDDL & "</select>"
    getDDLString = strDDL
end function
```

listing 2

```
private function getOptionTag(byval optionValue, _
byval optionText, byval selectedOption)
    if selectedOption = optionValue then
        getOptionTag = "<OPTION value='" & optionValue & _
        "' selected>" & optionText & "</OPTION>"
    else
        getOptionTag = "<OPTION value='" & optionValue & _
        "' >" & optionText & "</OPTION>"
    end if
end function

private function getDDLString(byval strTable, _
byval code_id, byval field_name, byval dbdescr)
    ...
    do while not rs1.eof
        strDDL = strDDL & _
        getOptionTag(rs1(dbcode), rs1(dbdescr), code_id)
        rs1.MoveNext
    loop
    ...
end function
```

coherence,
coupling,
& refactoring

listing 3

```
private function getDatabaseOptions_
(byval tableName, byval valueField, byval_
textField)
    dim options
    dim rs1

    set options = new Collection
    set rs1 = Conn.Execute("SELECT * FROM " & tableName & _
    " ORDER BY " & textField

    While not rs1.eof
        options.Add New Code Value/Description Pair
        rs1.MoveNext
    Loop

    GetDatabaseOptions = options
end function

private function getDDLString(byval selectName, byval_
options, byval selectedOption)
    dim index

    getDDLString = "<SELECT size='1' name='" & _
    selectName & "'>"

    for index = 0 to options.Count - 1
        getDDLString = getDDLString & getOptionTag(...)
    next

    getDDLString = getDDLString & "</SELECT>"
end function
```

we've also added flexibility to the system, since we can now use getOptionTag for other purposes as well.

We can improve this code even further. The getDDLString method still has external coupling—necessary in any program, but best isolated rather than mixed in with your other code. I've had to resort to pseudocode here due to space considerations, but Listing 3 demonstrates how we can have getDDLString take a collection of name/value pairs rather than directly talking to the database—thus removing the external coupling. This gives us more flexibility, since getDDLString can now return a drop-down list for any set of data instead of insisting that it comes from the database. Furthermore, maintenance will be easier; if the database structure changes, we won't need to consider the getDDLString function. The code changes would be limited to the much simpler getDatabaseOptions function.

We are pleased to announce our training partnership with ObjectMentor—a worldwide leader in software engineering and eXtreme Programming mentoring. To kick off the partnership, ObjectMentor will host a one-day XP overview at NOLA headquarters, 3535 Canal Street in New Orleans, on September 28, 2001. The overview will be suitable for managers, programmers, and anyone else interested in delivering high-quality software.

Acclaimed authors Robert C. Martin and Lowell Lindstrom will cover such XP practices as pair programming, test first design, and continuous integration. Attendees will learn how to tell whether or not their projects and organizations are good candidates for XP. They will hear case studies of both successful and unsuccessful XP projects. Attendees will also participate in a simplified XP planning session and see demonstrations of both XP planning and programming practices.

Programmers will learn the importance of pair programming, short iteration cycles, and estimation. Customers will learn what they can do to drive the development efforts of XP teams.

Register now for this FREE overview at www.objectmentor.com or call NOLA at (504) 488-1111.



example 2 refactoring

Our next example, Listing 4, shows control coupling, external coupling, and a relatively low level of cohesion. This code was not actually part of a subroutine—but, if it were, what would we call it?

ValidateAndSaveData? One of the things that was pointed out in the refactoring course was: if you have to have an “And” in a subroutine name, you’re probably doing too much in it.

Thus, the first step toward improving this code would be separating the validation code from the persistence code, letting the validation routine return a value that indicates whether or not the validation was successful. This improves the coherence of the code and removes some of the external coupling. The validation routine, however, is still less coherent than it could be. All individual validations should really be separate subroutines, making changes easier if the validation rules change. Even after we split those out, we still have the problem of external coupling. A good solution would be: have the overall validation routine (ValidateAllInput in listing 5, page 10) return a collection of errors instead of just a simple flag. When it starts, it empties the collection. Subsequent individual validation routines

listing 4

```
if isnull(l_EmployerName_TXT) then
    l_Error_Num = 1
end if
if isnull(l_StartDate_DTM) then
    l_Error_Num = 1
else
    if not isdate(l_StartDate_DTM) then
        response.write OutputLabel("Invalid Start Date. "&_
            "Use MM/DD/YYYY format<p>",true)
        l_Error_Num = 2
    end if
end if
if not isnull(l_FinishDate_DTM) then
    if not isdate(l_FinishDate_DTM) then
        response.write OutputLabel("Invalid Finish Date. "&_
            "Use MM/DD/YYYY format<p>",true)
        l_Error_Num = 2
    end if
end if
if isnull(l_PositionHeld_TXT) then
    l_Error_Num = 1
end if
if isnull(l_Notes_TXT) then
    l_Error_Num = 1
end if
if l_Error_Num = 0 then
    Set recAddressInformation = Server.CreateObject("ADODB.Recordset")
    With recAddressInformation
        .ActiveConnection = Conn
        .CursorType = adOpenKeyset
        .LockType = adLockOptimistic

        .Source = "SELECT * FROM T_AddressInformation"
        .Open
        .AddNew
        .Fields("PostalAddress1_TXT").value = l_PostalAddress1_TXT
        .Fields("PostalAddress2_TXT").value = l_PostalAddress2_TXT
        .Fields("PostalCity_TXT").value = l_PostalCity_TXT
        .Fields("PostalState_TXT").value = l_PostalState_TXT
        .Fields("PostalCode_TXT").value = l_PostalCode_TXT
        .Fields("PostalCountry_TXT").value = l_PostalCountry_TXT
        .Update
        l_EmployerAddress_NUM = _
        .Fields("AddressIdentifier_NUM").value
        .Close
    end With
end if
```

add errors to the collection. If the collection is empty after ValidateAllInput runs, we know all input was validated. To demonstrate this, I've used a simple string to contain the collection of errors. Each validation routine appends an error message where necessary. More sophisticated systems are certainly possible. Listing 5 shows all of the high-level code and one of the validation routines.

Remember—all refactoring does is strengthen cohesion and reduce coupling after the fact. If your Argo is already between the Symplegades, refactoring is the dove that can bring your ships safely through the crashing rocks. Personally, though, I'd rather avoid those waters entirely by maintaining strong cohesion and low coupling at all times. </end>



Senior Consultant
Pam Seghers

received a G.A.T.O.R. award from SAIC's

Vicki Lueb, manager of O&M support and Capital Projects, and Art Bonneval, customer relational office representative.

Ms. Seghers was recognized for "outstanding customer service" as team lead for the Accounting Code Block Master project at **Entergy Corporation**. Ms. Seghers has since passed the baton of team lead to Mike Joffe of SAIC and assumed the role of project manager for ACBM projects.

Senior Consultant
Maria Bertucci



received a letter of appreciation from Director Joanne Ellis, USDA National Finance Center Information Resources Management Division.

"Your work toward the successful implementation of Remedy has been critical to our success," Ms. Ellis stated. "Your knowledge, leadership, and ability to work well with all employees were key elements in the success of this project."

Listing 5

```

dim strErrorMessages

ValidateAllInput l_EmployeeName_TXT, l_StartDate_DTM, l_FinishDate_DTM, _
l_PositionHeld_TXT, l_Notes_TXT, strErrorMessages

if strErrorMessages = "" then
    SaveEmployeeInformation l_EmployeeName_TXT, l_StartDate_DTM, _
l_FinishDate_DTM, _
l_PositionHeld_TXT, l_Notes_TXT
else
    OutputErrorMessages strErrorMessages
end if

private sub ValidateAllInput(byval strEmployeeName, byval dtmStartDate, _
byval dtmFinishDate, byval strPositionHeld, byval _
strNotes,byref strErrorMessages)
    strErrorMessages = ""

    strErrorMessages = strErrorMessages & _
EmployeeNameValidationMessage(strEmployeeName)
    strErrorMessages = strErrorMessages & _
StartDateValidationMessage(dtmStartDate)
    strErrorMessages = strErrorMessages & _
FinishDateValidationMessage(dtmFinishDate)
    strErrorMessages = strErrorMessages & _
PositionHeldValidationMessage(strPositionHeld)
    strErrorMessages = strErrorMessages & _
NotesValidationMessage(strNotes)
end sub

private function StartDateValidationMessage(byval dtmStartDate)
if isnull(dtmStartDate) then
    StartDateValidationMessage = "Starting date required."
elseif not(isdate(dtmStartDate)) then
    StartDateValidationMessage = "Please use MM/DD/YYYY "& _
"format for start date."
else
    StartDateValidationMessage = ""
end if
end function

```



schedule is in danger of slipping, is it really necessary to pull the development team away from their work to collect this information? I don't think so. I have never held status meetings, and they are the first bit of pointless process I eliminate whenever I become the new lead of a group. I simply don't believe it's necessary to hold status meetings to determine whether a schedule is going to slip—that is, not if you're also collecting status reports.

So what about those status reports? How important are they? I think status reports—of some sort—are a necessary evil. A lead does, after all, need to know when problems occur. But note that while status reports are necessary, they—like status meetings—do not improve the product in any way. When you believe that a task is necessary but see that it doesn't improve the product, you should always ask a more specific form of this general question: *How can I keep the benefits of this task yet remove the drawbacks?*

Status reports do serve a valuable purpose, but they take time to write and can create a negative mindset in the team—at least in the way they have been done in many Microsoft groups.

to the basics stick

If you look back at the points raised in this chapter, you'll see that they boil down to a simple formula for software development: figure out what you're trying to do and how you should do it, and then make sure that every team member stays focused on the project goals, coding priorities, and quality bars you've come up with. Pretty basic stuff.

Now take a look at the teams in your company. How many have detailed goals for their projects? In how many do the programmers understand exactly how they should be writing their code and to what standards of quality? Then ask yourself, "Are the programming teams focused fully on improving their products?"

Now look at the project leads in your company. Do they habitually call meetings

to discuss every little thing, or do they reserve meetings for truly important issues? Do they put obstacles in the programmers' paths—asking them to write questionably useful reports, for instance—or do the leads strive to remove obstacles to development work?

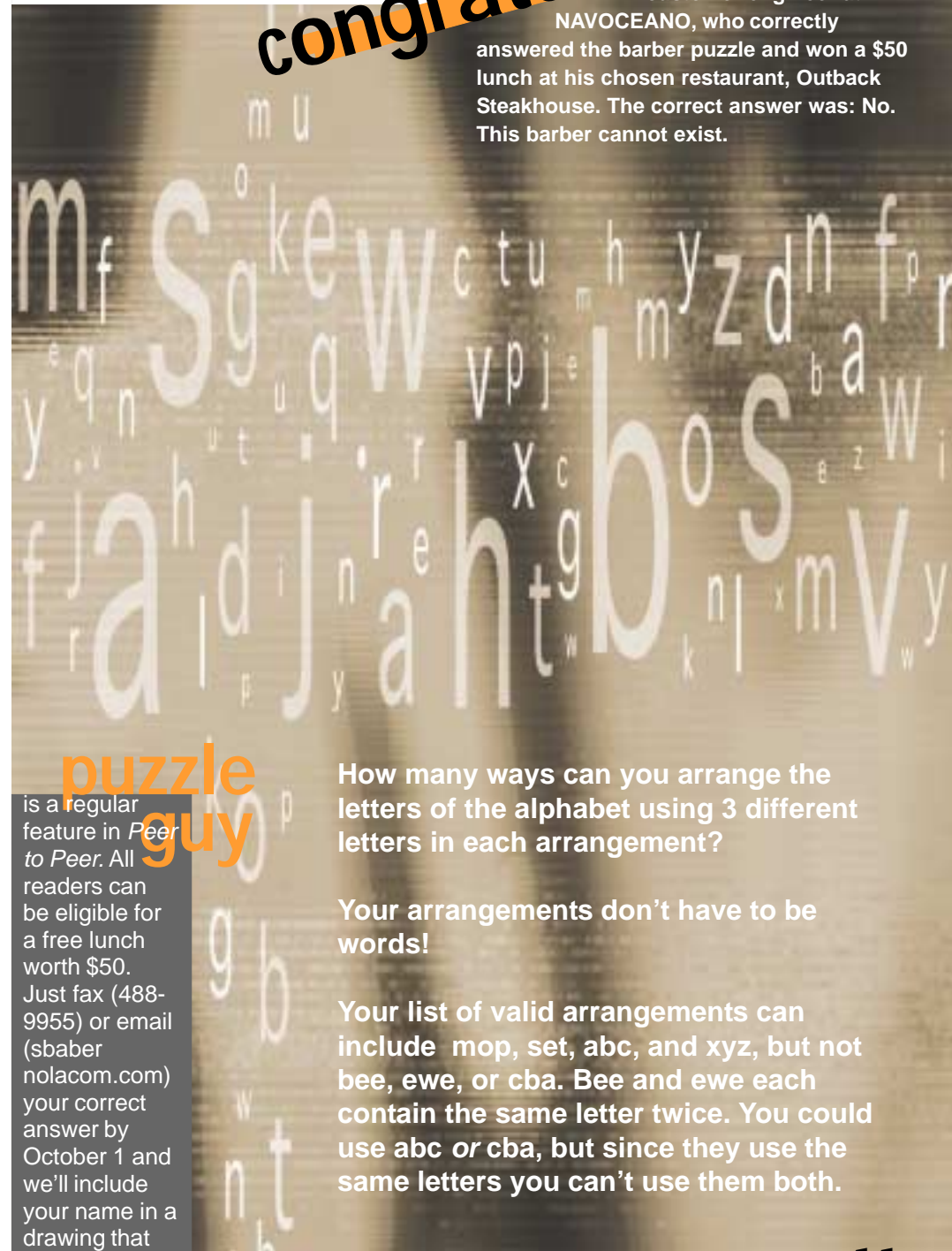
The points in this chapter are basic, but in my experience few groups focus on these fundamental concepts. And that, I believe, is why you can't pick up *InfoWorld* or *MacWEEK* without reading about some project that has slipped another six months or on which the programmers are working so hard that they don't even bother to go home at night. </end>



A native New Orleansian, Puzzle Guy Syed Baber recently graduated from Tulane University, where he earned degrees in computer science and molecular biology. Syed joined the NOLA team in 2000 and is now an eXtreme programmer in the commercial applications group.

congratulations to michael ashmead

NOLA senior customer engineer at NAVOCEANO, who correctly answered the barber puzzle and won a \$50 lunch at his chosen restaurant, Outback Steakhouse. The correct answer was: No. This barber cannot exist.



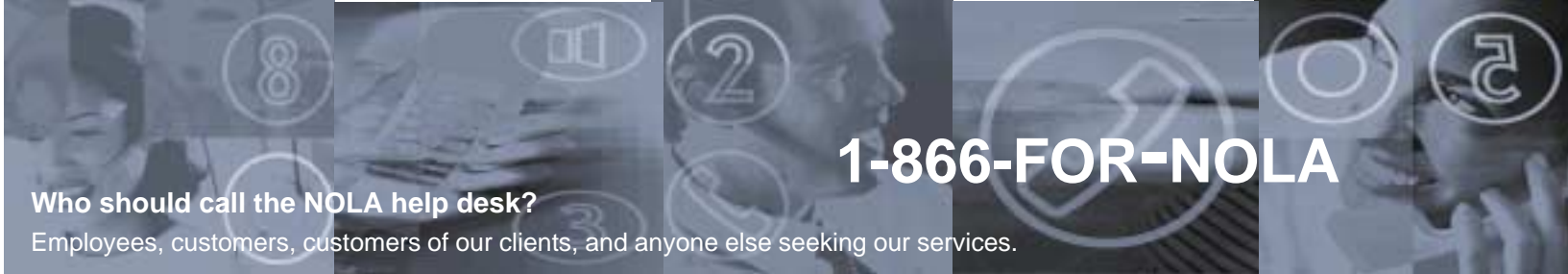
puzzle guy

is a regular feature in Peer to Peer. All readers can be eligible for a free lunch worth \$50. Just fax (488-9955) or email (sbaber nolacom.com) your correct answer by October 1 and we'll include your name in a drawing that will determine the winner.

How many ways can you arrange the letters of the alphabet using 3 different letters in each arrangement?

Your arrangements don't have to be words!

Your list of valid arrangements can include mop, set, abc, and xyz, but not bee, ewe, or cba. Bee and ewe each contain the same letter twice. You could use abc or cba, but since they use the same letters you can't use them both.



1-866-FOR-NOLA

Who should call the NOLA help desk?

Employees, customers, customers of our clients, and anyone else seeking our services.

What should NOLA's help desk be used for?

Requesting service, service changes, and appraisals. Complaints. Observations. Questions. Suggestions.

Where are NOLA's help desk staff located?

The help line—1-866-FOR-NOLA—rings directly to our corporate headquarters.

When is help available?

Monday through Friday between the hours of 8 AM and 5 PM.

Why did NOLA establish the help desk?

The NOLA help desk was established to better serve our customers. At NOLA, our customer group includes our employees because our corporate staff is committed to providing quality service internally as well as externally.

How does the help desk work?

When a call comes in to 1-866-FOR-NOLA, one of our customer service representatives determines what sort of help is required. The rep will provide help personally or dispatch the call appropriately. All calls are tracked from conception to completion in our web-based issue tracking system. We also perform follow-up calls to make sure callers are completely satisfied with the help our reps provide.

BULK RATE
US POSTAGE
PAID
NEW ORLEANS LA
PERMIT NO. 2554



New Orleans, LA 70119-6170

COMPUTER
SERVICES

NOLA