Uncovering Risk in Your ICD-10 Conversion

Key Risk & Effort Metrics for ICD Data Testing

Abstract

If you are implementing ICD-10 support in your software applications, many things are important to your management of the process. Good development and test practices include: requirement and design reviews, configuration management, and completion of a thorough set of functional (requirements based) tests. However, to reduce risk, a critical and often forgotten area is code coverage, leading to the question…

Do you know whether you are testing all of your ICD data locations in the code?

Are you trusting someone's best guess that each instance of ICD-related data in your source code, whether modified for new ICD-10 data formats or staying the same, is being tested? The good news is that you can know objectively and definitively, with the help of tools like McCabe IQ.

Code coverage has been a best practice for software development for decades, especially for critical applications and projects. If your ICD conversion is critical to get right, then you should analyze code coverage of your ICD data usage in source code.

To explore this further, after assuring that functional (requirements-based) tests are working correctly, the two most important risk metrics to have reported to you, with respect to your ICD data locations testing, are these:

- **Risk %age** – The percentage of ICD data locations that have not been tested
- **Risk Volume** – The count of ICD data locations that have not been tested

Sample Report at Summary Level of
Key Metrics with respect to ICD Data References Testing Completion

Program	Risk%-wrtIDT	RiskVol-wrtIDT	RiskVolRel-wrtIDT	EffortInd-wrtIDT	Paths	ICDDataRefs	TestedICDDataRefs	%TestedICDDataRefs
ovid-domain_McCabe	5.14	11	17	9	4871	214	203	94.86

After having that information both in summary and in detail for each method/class, do you also want to have indicators of the level of your risk as factored by reliability, and of the level of effort that might be required to create additional tests to test the untested ICD data locations in each method/class? **If you want further information, then also use the following metrics:**

- **Risk Volume Factored by Reliability** – The count of ICD data locations in the code that have not been tested, with each factored by an estimated reliability factor (based upon complexity) for the method in which they exist
- **Effort Completion Indicator** – A count of paths within methods containing untested ICD data for which test cases need to be considered to test untested ICD data locations

In short, you can know and manage your risk related to testing of ICD data in your code.

Introduction

This paper discusses how to uncover risk in your ICD-10 conversion, relative to the testing of source code locations using ICD data. The paper is organized as follows:

- Risk of Software Failures Related to ICD Data Usage in Code
- Code Coverage as a Best Practice for Software Risk Mitigation
- Code Coverage of ICD Data as a Best Practice for ICD Conversion Risk Mitigation
- Key Risk and Effort Metrics with respect to ICD Data References
- Changes to Software Processes for ICD Conversion to Reduce Risk

Knowing whether you have tested source code locations containing ICD data references, and having risk metrics related to this testing, can help you better manage your limited resources and time.

If you wish to get a general understanding of this topic, read the first three sections and the fifth, but skip the fourth. If you wish to further explore the details of a set of several risk metrics, read the fourth section on Key Risk and Effort Metrics with respect to ICD Data References.

Risk of Software Failures Related to ICD Data Usage in Code

What is your risk of software failures related to your conversion project for ICD-10? One place to focus is the locations of ICD data usage within the source code. Let us consider your risk related to ICD data usage in terms below.

Some Key Software Risk Factors Applied to ICD Data Usage in Code

Criticality To Mission	The success of your ICD conversion is partially related to how thoroughly you test each usage location of ICD data within your source code. It is true that some locations may be more critical to your success than others, and therefore you may want to analyze which components and which classes/methods are more critical so you can focus more resources on them.
Code Without Tests	ICD data usage locations in your code that do not have thorough tests are risks for causing a failure. It is almost impossible to completely test all combinations of logic and data in methods containing ICD data usage for anything but small methods. Therefore, we recommend that you test each such location at least one time. Testing of each ICD data reference at least one time is essentially branch level code coverage for those locations. For locations within code with greater criticality, you might require branch level coverage for the entire method containing such data, or even Basis Path or Boolean coverage levels.
Change History	For your ICD conversion, you will have some source code that will change. The code that changes has an increased risk of failure relative to unchanged code; even though unchanged code that uses ICD data is still important to your success because it may be used slightly differently when ICD-10 enhancements are made.
Complexity	There is a greater risk of having undetected defects in code that has a greater complexity, primarily because of untested branches and paths. Therefore, if you have ICD data usage in methods/classes that are complex, there is a greater probability that undetected defects exist that impact your ICD conversion success. The code complexity of a method can be measured in multiple ways, one of which is the renowned McCabe cyclomatic complexity metric. We suggest that if your ICD data usage is within methods with a high cyclomatic complexity, then the risk of failure is greater.
Correction History	You may not have information available on correction (defects that were corrected) history of specific code methods/classes or components. However, if you do have information that shows that certain methods/classes or components have had significant corrections (especially those of a high priority), then consider that those methods/classes and components might have a greater risk of failure.

Testing with a good set of functional (requirements-based) tests is important, and does not require you knowing where your ICD data references locations exist in the source code. However, taking action on the above factors does require that you know those locations. There is value in considering these factors for many reasons, the greatest of which may be the allocation of resources/funds during your ICD conversion project.

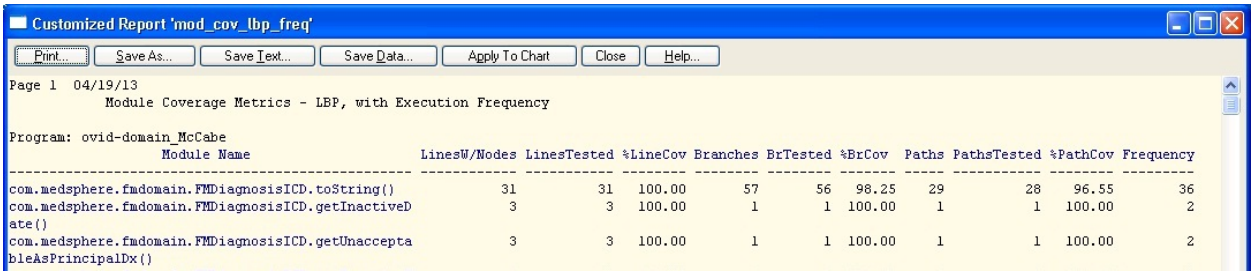
Code Coverage as a Best Practice for Software Risk Mitigation

Code coverage has been a best practice in the software industry for decades, and is particularly recommended for software critical to your success. Code coverage is the analysis of whether a set of tests actually executes specific source code. The term white box testing is often used to describe it, because the inside of the "box" of your built software application can be viewed. (We will not attempt to distinguish here between white box, glass box, and various other terms used in the software industry.) Note that black box testing commonly means your functional (requirements based) tests, because during such tests you are analyzing whether the functionality works without regard for what source code is executed or not, so the inside of the "box" is black (not visible) to the tester.

Code coverage analysis can be performed at many levels. The primary levels include:

- **Line coverage** (Asks the question "has at least something on each line of code been executed at least once?", sometimes called code coverage)
- **Branch coverage** (Asks the question "has each branch outcome been tested at least once?", statement coverage and block coverage can be similar to this)
- **Boolean coverage** (Related to testing conditions in a decision, multiple variations exist)
- **Basis Path coverage** (Asks the question "have at least a set of independent paths equal to the cyclomatic complexity value been tested?", sometimes called McCabe cyclomatic path testing)

Sample Report Showing Line, Branch, and Path Level Coverage Reporting



Areas of code (applications, components, or classes/methods) that are more likely to have hidden defects, and may cause a significant impact to your mission/business, should have more resources/funds allocated to their development and to their testing, in order to reduce risk in those areas. Some guidelines recommend that code coverage usage increase based on the impact of failure to an application or component. Our general recommendations might be summarized as follows:

Code Coverage – General Recommendations

Impact of Failure to Component or Method/Class	Code Coverage Level Minimum Recommendation
Catastrophic impact	Basis Path level or Boolean level
Moderate impact	Branch level
Little or no impact	None required

Code Coverage of ICD Data as a Best Practice for ICD Conversion Risk Mitigation

Because code coverage has been a best practice in the software industry for decades, and because your ICD conversion project is critical to your organization's success, we recommend that you apply a form of code coverage to your effort.

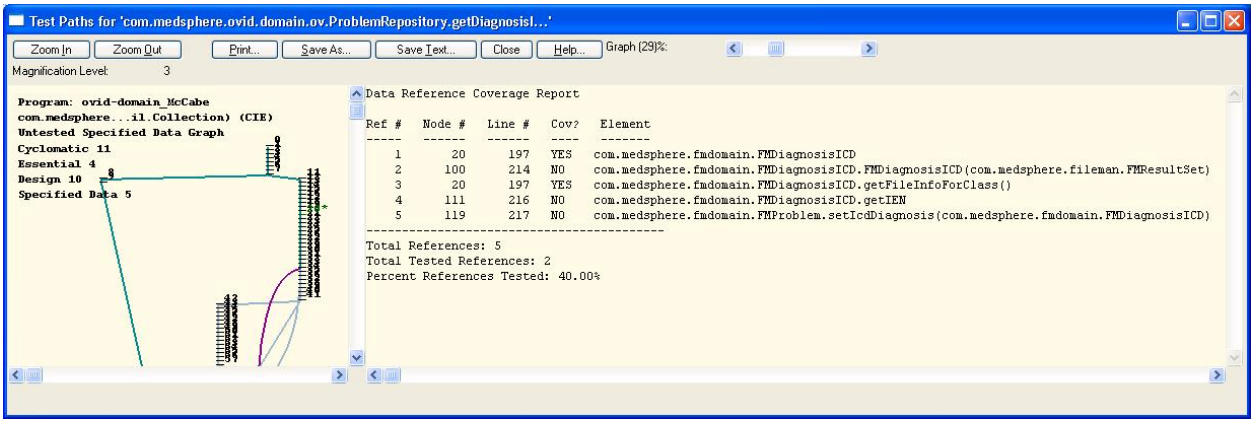
More specifically, we recommend the following:

Code Coverage for ICD Conversion – General Recommendations

Impact Of Failure to Component or Method/Class Containing ICD Data Usage	Code Coverage Level Minimum Recommendation
Catastrophic impact	Branch level (for all branches of every method containing ICD Data References)
Moderate impact or UNKNOWN impact	ICD Data References Testing (each usage tested at least once, which is similar to branch level coverage for those data usages)
Little or no impact	None required

If the impact of failure of specific components or classes/methods is unknown, we recommend that you use the moderate impact level recommendation, which is testing every usage of ICD data at least once.

Sample Report Showing Coverage Status ("Cov?" = YES or NO) for each ICD Data Reference Within a Method



Key Risk and Effort Metrics with respect to ICD Data References Testing

We recognize that good project management requires being aware of and managing with multiple metrics. For an ICD-10 conversion project, your thoroughness in the testing of the source code locations using ICD related data will impact your success. Therefore, we propose that **the two most important metrics relative to the testing of those data locations to have reported to you are these:**

- **Risk %age** – The percentage of ICD data locations that have not been tested
- **Risk Volume** – The count of ICD data locations that have not been tested

And, after having that information both in summary and in detail for each method/class, **the next most important metrics to have available may be:**

- **Risk Volume Factored by Reliability** – The count of ICD data locations in the code that have not been tested, with each factored by an estimated reliability factor (based upon complexity) for the method in which they exist
- **Effort Completion Indicator** – A count of paths within methods containing untested ICD data for which test cases need to be considered to test untested ICD data locations

This section describes four key metrics to facilitate management of ICD data references testing status. The information is organized as follows:

- Table summarizing the four metrics
- Sample reports using the four metrics – provided are a summary level report and a detail (module, i.e., method) level report; a class level report could also be generated
- Description of each of the four metrics in detail, including:
  - **Purpose** – the purpose of the metric for the user
  - **Calculation** – the formula used to calculate the metric
  - **Examples** – example(s) of the value of the metric for a specific case(s)

Key Metrics with respect to ICD Data References Testing Completion

	Metric Title	Summary	Calculation
1	Risk %age	Untested ICD Data References as a percentage of Total ICD Data References	Risk%-wrtIDT: = %_untested_sdr
2	Risk Volume	Untested ICD Data References count	RiskVol-wrtIDT: = untested_sdr
3	Risk Volume Factored by Reliability	Untested ICD Data References count with a Reliability Factor included (based on complexity of the method in which they exist)	RiskVolRel-wrtIDT: If vg < 11, then = untested_sdr, Else = untested_sdr x (ln (sqrt (vg)))
4	Effort Completion Indicator	Untested paths impacted by untested ICD Data References	EffortInd-wrtIDT: If untested_sdr < untested_sdv, Then = untested_sdr Else = untested_sdv

Table Legend:

wrt - with respect to; IDT - ICD Data References Testing; sdr - # of specified data references, which is equivalent to # of ICD data references locations; sdv - specified data complexity or # of specified data paths, which is equivalent to # of data paths impacted by ICD data references within a method; vg - the McCabe cyclomatic complexity or # of cyclomatic paths within a method.

Sample Report at Summary Level of Key Metrics with respect to ICD Data References Testing Completion

Program	Risk%-wrtIDT	RiskVol-wrtIDT	RiskVolRel-wrtIDT	EffortInd-wrtIDT	Paths	ICDDataRefs	TestedICDDataRefs	%TestedICDDataRefs
ovid-domain_McCabe	5.14	11	17	9	4871	214	203	94.86

Sample Report at Detail (Module, i.e., Method) Level of Key Metrics with respect to ICD Data References Testing Completion

ICD Data Coverage plus Risks & Effort (Filtered ICDDataRefs >= 1 & %TestedICDDataRefs < 100) (Sort By Risk% RiskVol)

Program: ovid-domain_McCabe

Specified Data Set no title

File name : *
Module name : *
Element name: *icd* | *ICD* | *iCD*
Element type: *
Declaration : *

Module Name	Risk%-wrtIDT	RiskVol-wrtIDT	RiskVolRel-wrtIDT	EffortInd-wrtIDT	Paths	ICDDataRefs	TestedICDDataRefs	%TestedICDDataRefs
com.medsphere.ovid.domain.ov.PatientVisitRepository.getDetail(java.util.HashMap,java.util.Collection)	100	5	10.4	4	64	5	0	0
com.medsphere.fmdomain.FMICD_Diagnosis.__Anon_001_.getFields()	100	2	2	1	1	2	0	0
com.medsphere.ovid.domain.ov.ProblemRepository.getDiagnosisInfo(java.util.Collection)	60	3	3.6	3	11	5	2	40
com.medsphere.fmdomain.FMDiagnosisICD.toString()	1.75	1	1.68	1	29	57	56	98.25
Total:	261.75	11	17.68	9	105	69	58	138.25
Average:	65.44	2.75	4.42	2.25	26.25	17.25	14.5	34.56
Maximum:	100	5	10.4	4	64	57	56	98.25
Minimum:	1.75	1	1.68	1	1	2	0	0
Std Dev:	46.46	1.71	4.07	1.5	27.71	26.54	27.68	46.46
Rows in Report:	4							

1. Risk %age with respect to ICD Data References Testing Completion

Purpose:

To provide management with a metric for percentage progress of test completion for ICD Data References Testing (IDT). The lower the value, the lower the risk. The goal might be 0% untested ICD data references, or the goal might be a bit higher for methods or components for which the impact is minimal to the organization if an ICD-related software failure occurs in production.

Calculation:

Risk%-wrtIDT

= %_untested_sdr

This is the percentage of untested ICD data references, where references were identified by a product user (possibly a developer or tester using string searches and other techniques, or using locations identified otherwise) and testing is that for which code coverage data has been obtained. This is calculated for each level by summing the untested_sdr (i.e., untested specified data references, which are your untested ICD data references) at that level, and dividing that by the sum of sdr at that level.

Examples:

Assume that the component under analysis for ICD Data Testing has a total of 200 ICD data references identified (sdr), and your testing has tested 160 of them, leaving 40 untested (untested_sdr). Then, this metric's value is 20.0 (meaning 20.0% of ICD data references have not been tested).

Untested ICD Data References	Total ICD Data References	Risk%-wrtIDT
40	200	20.0

2. Risk Volume with respect to ICD Data References Testing Completion

Purpose:

To provide management with a metric for progress by volume of test completion for ICD Data References Testing (IDT). A volumetric progress calculation is especially valuable when comparing the progress of one component vs. another (e.g., one method vs. another, one class vs. another, or one project vs. another, etc.). The lower the value, the lower the risk. The goal might be zero ICD data references untested, or the goal might be a bit higher for methods or components for which the impact is minimal to the organization if an ICD-related software failure occurs in production.

Calculation:

RiskVol-wrtIDT

= untested_sdr

This is the count of untested ICD data references, where references were identified by a product user and testing is that for which code coverage data has been obtained. This is calculated for the module/method level; for higher level summary values, the values for all modules/methods are summed.

Examples:

Assume that the component under analysis for ICD Data Testing has a total of 200 ICD data references identified (sdr), and your testing has tested 160 of them, leaving 40 untested (untested_sdr). Then, this metric's value is 40 (meaning 40 ICD data references have not been tested).

Untested ICD Data References	RiskVol-wrtIDT
40	40

3. Risk Volume factored by Reliability with respect to ICD Data References Testing Completion

Purpose:

To provide management with a metric for progress of remaining volume for test completion for ICD Data References Testing (IDT), where that remaining volume is factored for reliability. A volumetric progress with a reliability factor calculation is especially valuable when comparing the progress of one component vs. another (e.g., one method vs. another, one class vs. another, or one project vs. another, etc.) when the locations of the ICD data references are within components of differing complexities. The lower the value, the lower the risk. It has been shown that high cyclomatic complexity can correlate to lower reliability, therefore the location of ICD data references that remain untested present a greater risk when they are within methods of high complexity. This metric provides a means to consider that additional risk. The goal might be zero ICD data references untested and therefore also zero value for this metric, or the goal might be a bit higher for methods or components for which the impact is minimal to the organization if an ICD-related software failure occurs in production.

Calculation:

RiskVolRel-wrtIDT

If vg < 11, then = untested_sdr

Else = untested_sdr x (ln (sqrt (vg)))

This is the count of untested ICD data references, where references were identified by a product user and testing is that for which code coverage data has been obtained; with each multiplied by a Reliability Factor for the method containing them. The Reliability Factor is a factor of cyclomatic complexity (vg), because that metric has been shown to correlate well to reliability (probability of a latent defect existing); the greater the cyclomatic complexity the greater the risk of leaving a potential defect untested. The Reliability Factor is not something that can be defined with precision, especially in this case for specific data references (ICD data references) within a method. Therefore, we have determined to use the following calculation as an estimate, for several reasons - because it results in no additional risk for methods with cyclomatic complexity of less than 11; because it increases risk value very minimally as the cyclomatic complexity increases; and because it uses functions that are available within the McCabe IQ product, i.e., square root (sqrt) and natural logarithm (ln). Thus, where vg is the cyclomatic complexity of the method,

If vg < 11, Reliability Factor = 1,

Else Reliability Factor = ln (sqrt (vg))

This is for module/method level; for higher level summary values, the metric values for each module/method (inclusive of the Reliability Factor) are calculated then summed.

The following table shows values of the Reliability Factor for certain values of cyclomatic complexity. Note that the factor increases slowly.

Cyclomatic Complexity (vg)	Approximate Reliability Factor
< 11	1.00
12	1.24
15	1.35
20	1.50
25	1.61
40	1.84
60	2.05
100	2.30
200	2.65

Examples:

Assume that the component under analysis for ICD Data Testing has a total of 200 ICD data references identified (sdr), and your testing has tested 160 of them, leaving 40 untested (untested_sdr). And assume that the cyclomatic complexity of every method containing those 40 references is as in the second column of the following table. Then, this metric's value is in the last column of the table.

Untested ICD Data References	Cyclomatic Complexity	Reliability Factor	RiskVolRel-wrtIDT
40	4	1	40
40	9	1	40
40	11	1.199	47.96
40	16	1.386	55.44
40	25	1.609	64.36
40	49	1.946	77.84
40	100	2.303	92.12

4. Effort Completion Indicator with respect to ICD Data References Testing Completion

Purpose:

To provide management with a metric that estimates the relative effort required to complete your ICD Data References Testing (IDT), that is, complete testing of remaining untested ICD data references locations. This volumetric indicator is in units of a number of paths impacted, not in units of test team labor hours. However, this metric is especially valuable when comparing the

remaining test effort for one component vs. another (e.g., one method vs. another, one class vs. another, or one project vs. another, etc.). The lower the value, the lower the effort that might be required to analyze how to create additional test cases to test the remaining untested ICD data references. For example, if a set of 10 untested ICD data references are all within one path in one method, the effort to test them is much less than if the 10 are within 10 different paths within one method or are within 10 different methods. The goal might be zero for the this effort completion indicator, or the goal might be a bit higher for methods or components for which the impact is minimal to the organization if an ICD-related software failure occurs in production.

Calculation:

EffortInd-wrtIDT

If untested_sdr < untested_sdv,

Then = untested_sdr,

Else = untested_sdv

This is an estimate of the number of paths (sdv) within methods that a person will need to consider to create additional test cases to test the remaining untested ICD data references. The untested_sdv metric is a count of cyclomatic paths impacted by untested ICD data references (as counted by untested_sdr), where references were identified by a product user and testing is that for which code coverage data has been obtained. The calculation value is set to the number of untested_sdr when the untested number of ICD data references is less than the untested_sdv, to account for the McCabe IQ tool's techniques that may recommend the testing of an additional path for a method when one or more data references in that method are within one side of a decision. This calculation is for module/method level; for higher level values, sum the values for all modules/methods. Note that the sdv metric is calculated as a reduction of a method's graph by removing structured constructs containing no ICD data references, then calculating the cyclomatic complexity of the resulting graph. Note also that when the user is collecting code coverage data only below the Basis Path (cyclomatic path) level, such as at the line level or branch level, then this metric has some value, but is more limited in value. This limitation is due to the fact that when there are some ICD data references tested in a method but not all, the effort indicator metric value is not adjusted downward until the number of untested ICD data references is less than the number of paths containing all ICD data references.

Examples:

Assume that the component under analysis for ICD Data Testing has a total of 200 ICD data references identified (sdr), and your testing has tested 160 of them, leaving 40 untested (untested_sdr). And assume that the untested_sdv (paths impacted by those untested data references) is as in the second and third columns of the following table. Then, this metric's value is in the last column of the table.

Untested ICD Data References	Exist in # Modules	Untested Sdv for Every Method Containing the Untested ICD Data References	EffortInd-wrtIDT
40	1	1	1
40	1	20	20
40	1	40	40
40	5	3	15
40	10	1	10
40	10	4	40
40	40	1	40

Changes to Software Processes for ICD Conversion to Reduce Risk

Let us assume that you have determined that it seems reasonable to perform ICD Data References Testing (IDT) for your ICD conversion project, which involves code coverage analysis of your ICD data usage locations. Then, what would the process be to perform such analysis and manage by it?

In essence there are several basic steps, each of which might be performed multiple times depending on various considerations:

- **ICD Data References Identification** – The exact locations within source code of ICD related data usage must be identified by some means. Commonly this would be performed just once. Tools such as McCabe IQ have several string search techniques that can be used in combination with user determinations of exactly which found data references are actually ICD related. If an organization has identified such locations by some other means, that information might be brought into a tool.
- **Code Coverage Collection** – A process of preparing the application to collect coverage data, then running tests related to ICD (possibly including all of your regression tests) to collect coverage data, must be performed. When using tools such as McCabe IQ, this can be done independently of the ICD data references identification; that is, it doesn't matter which step you perform first.
- **Reporting ICD Data References Coverage & Related Risk/Effort Metrics** – This step is simply reporting on the information already available from the two prior steps.
- **Managing Additional Test Creation To Test Untested ICD Data References Locations** – You may decide to have new tests created to reduce your risk, and may re-collect coverage data afterwards to report on the new status.

Note that you may want to perform this analysis and obtain these metrics PRIOR to implementing applications enhancements for ICD-10, to assure that your existing test cases thoroughly test your existing ICD data locations now. And/or, you may want these metrics AFTER implementing applications enhancements for ICD-10, to assure that your test cases thoroughly test your final (new/changed) ICD data locations.

Conclusion

Your organization could be put at risk if the ICD-10 implementation is not correct or is not timely. McCabe IQ can help you do it right by providing objective measures of your testing thoroughness. And McCabe IQ can help you do it on time, by pointing where to focus testing resources for reduced risk and maximum reliability. If you are managing ICD-10 implementation, the risk and effort estimation metrics described herein can help you assess your risk with respect to testing of your ICD data references, and help you plan your resources wisely.

About McCabe Software, Inc.

McCabe Software provides Software Quality Management and Software Configuration Management solutions worldwide. "McCabe IQ" (Integrated Quality) is used to analyze and visualize the security, quality, and testing of mission, life, and business critical applications. *McCabe IQ ICD-10 Edition* helps you reduce risk in your ICD-10 conversion efforts by identifying ICD data locations in your software and assuring that tests exist to cover ICD data locations in your software before and after having remediated code. McCabe Software has offices in the United States and distribution worldwide and is online at www.mccabe.com.