BIO

PAPER

T16

November 7, 2002
2:30 PM

IMPROVEMENT IS A JOURNEY: A SOFTWARE TEST IMPROVEMENT ROAD MAP

Karen Rosengren
IBM

International Conference On
Software Testing Analysis & Review
November 4-8, 2002
Anaheim, CA USA

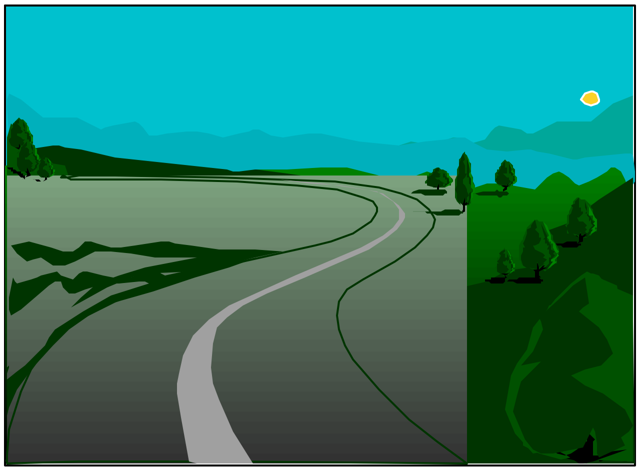
Karen Rosengren

Karen has worked for IBM for 23 years, with over 15 years of test experience. Most of her testing career has been spent testing operating systems, first in Poughkeepsie, New York working on IBM's OS/390, then in Boca Raton, Florida working on OS/2. She is currently the IBM TEST Technology Leader, reporting to the IBM Director of Test. Her mission is to improve testing at IBM through the use of technology and tools. She investigates the latest tools and practices in order to develop the methodology to make testing more effective and efficient, and sets direction for improvement within the test teams across IBM based on that methodology. She works closely with the teams to understand their environments and products, educates them on the latest technology, and drives change into the teams.

Karen has a B.S. degree in Computer Science from The Pennsylvania State University. She lives in Round Rock, Texas with her husband, who also works for IBM, and their two children.

Improvement is a Journey: Strategic Test Improvement Roadmap

STIR



Karen Rosengren

IBM Corporation

krosengr@us.ibm.com

Introduction

So, you want to improve the quality of the testing done by your organization? The test quality improvement journey has several aspects to consider: the identification of improvement actions, which improvement action to start with and how to continue to improve. This document focuses on those improvement actions and ways to implement and improve on an existing set of good practices.

The Strategic Test Improvement Roadmap (STIR) is a tool which our organization has used to identify good testing practices and to provide a roadmap to implementing and improving on those practices. Our "definition" of practice is: a repeatable performance of an activity, capturing the skill gained from experience. In other words, practices are those testing activities that are done on a daily basis. And, as experience is gained, the quality and performance of those activities improves. At first glance, this appears fairly simple. However, there are many good testing practices out there. How do you know where to start?   How do you know which practices you are capable of implementing? How do you know which practices will help you the most?

Background of our Testing Improvement Journey

At this point, let's step back for a moment. In 1998, Bill Woodworth was appointed the IBM Director of Test, a position covering the entire corporation. His mission was to improve the way that testing was being done at IBM. He created the IBM TEST Community Leaders (ITCL), a group of 30 of the top technical people in test (Top Guns), with a corresponding 30 managers, who held key middle management positions in test. The idea behind the ITCL was to bring testers together to share experiences with each other and to strengthen test across IBM by allowing the individual units to grow and improve. The purpose of this group was for the same reason many testers attend conferences like STAR; to find new ideas and to share what is currently being done so that their organization can improve on the way test is being done. This concept sounds simple enough, but in reality we ran into many challenges. We were working with a very diverse set of testers and products, and the culture within each of the groups was different. As stated earlier, the practices are really what are done on a day-to-day basis. Each of the areas represented did testing differently, had different processes, and used a wide variety of practices. They developed products that spanned many functions; microcode, base operating systems, middleware, and applications. On top of that, each area's organization represented a different culture. Some placed a high value on test, while others did not. This made it very difficult to identify "best practices" that were applicable to such a diverse set of testers and gave us many challenges when we actually tried to deploy a set of practices. We noticed that some teams were very successful at implementing and deploying new practices. While other teams floundered and failed miserably, unable to determine where and how to start. In order to make each of our teams successful we needed a structure to communicate what the good practices are and to show them how to deploy those practices in such a way as to maximize their success.

Development of the Strategic Test Improvement Roadmap

As stated earlier, the ITCL was made up of top technical leaders from various test areas. Many of them had more than 10 years of experience in test and found it fairly easy to apply that experience to deploying new practices. However, in many cases, they were not the norm and that worked against us. Finally, Dave Jewell, one of the Top Guns, presented some fairly basic but extremely important insight into deploying practices. He noticed that practices vary by the level of skill and experience that is required to implement them. That is why we saw tremendous success with some of our practices and why only a few groups were able to deploy other practices. We also realized that practices are related to each other. We saw that it was impossible to deploy certain practices until some other practice was successfully deployed.

Based on these observations, we developed a structure that we called the Strategic Test Improvement Roadmap (STIR) to easily define for a tester which practices the ITCL recommends and to provide a logical way to implement them. The concept is that teams start with practices at the basic level, and implement the basic steps within those practices. As they improve, gain skill, and collect data, they can implement practices that start at the intermediate level. They can then revisit practices started at the basic level and implement the intermediate steps for those practices. What this structure allows us to provide is a simple way to show the teams what the ITCL recommended practices are, as well as how to implement them in a way that allows the greatest opportunity for success.

An important point is that STIR is not a maturity model. It’s an organization of practices to help determine which practice should or can be deployed next. A maturity model measures an organization. STIR does not measure the maturity of an organization.

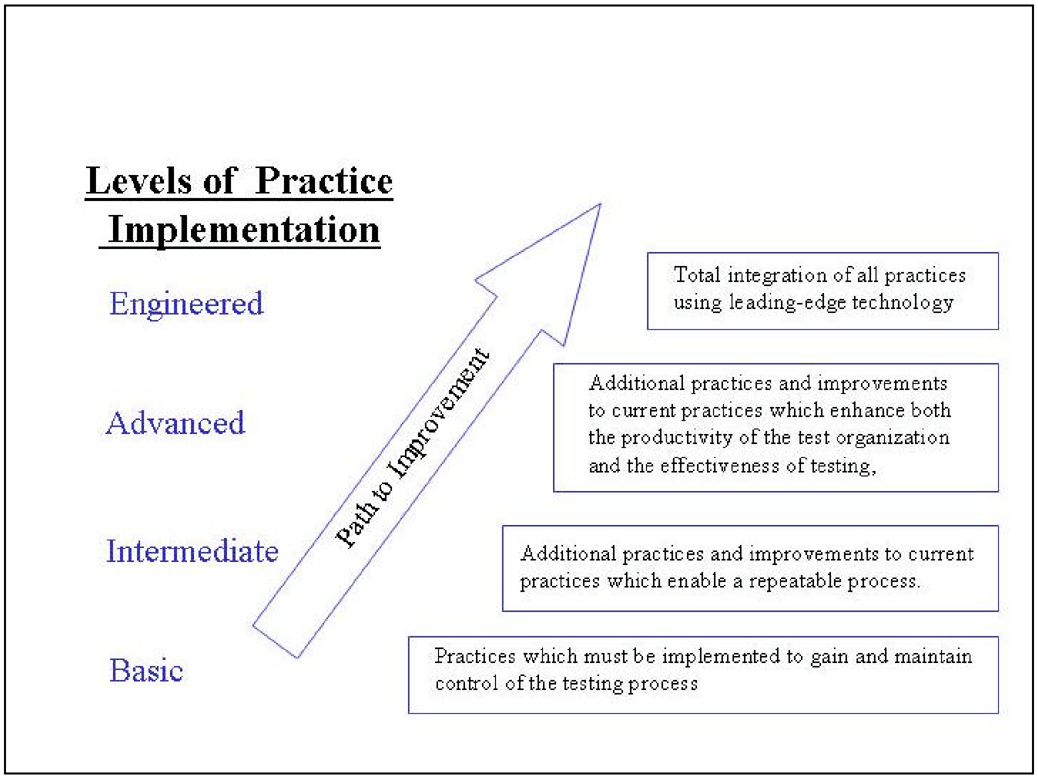

Levels of Practice Implementation
Engineered
Advanced
Intermediate
Basic
Path to Improvement
Total integration of all practices using leading-edge technology
Additional practices and improvements to current practices which enhance both the productivity of the test organization and the effectiveness of testing.
Additional practices and improvements to current practices which enable a repeatable process.
Practices which must be implemented to gain and maintain control of the testing process

Figure 1 Overview of STIR

Test Practice Levels

From a structural point of view, the practices are grouped into levels, from basic to engineered. Those practices which are absolutely necessary to establish control over your projects are at the basic level. These are the easiest practices to implement and ones that have seen deployed in most organizations. Practices that require data from other practices or those that require a more skilled organization to implement are at the intermediate or advanced level. Also, to stress continuous improvement and to aid with deployment, each individual practice has implementation steps that are at the basic to engineered level.

Basic is the starting point. These are practices that are necessary to provide control over projects. They require little skill to implement them and they are the basis for further improvements. The test organization can implement these practices with no involvement from areas outside of test in the development life cycle.

Intermediate practices focus on creating a repeatable process. They start to involve development, but in a reactive way. For instance, an intermediate step of review and inspection would involve reviewing a document after it was written and providing comments.

Advanced, with this level the test organization starts to become more proactive, involved earlier in the development cycle, providing input to the development of requirements, designs and specifications, instead of just reviewing and providing comments. The key focus is on improving productivity and enhancing effectiveness.

At both the intermediate and advanced levels, new practices are introduced.

Engineered, we are still working on what it means to be engineered, but our goal is the total integration of all practices. We do not introduce new practices at this level, but would probably expect a test team to be implementing practices at the engineered level to handle introducing leading edge technology.

Test Domains

We further categorized the practices by "domain". The definition of domain is sphere of activity. Domains are collections of activities that are related, like planning or execution. They are loosely related to steps in a process, but by calling them domains, we did not tie STIR to any specific process. With the variety of test groups that we are supporting, we could not tie STIR to one specific process.

Test Management is the basic process of identifying tasks, tracking status, and moving a project along to completion.

Test Planning encompasses those activities associated with anticipating the tasks that will need to be performed and documenting them. This is the first step and the place where the feedback loop should be closed. As you start a new project, you should always look back to see what worked well and not so well and plan for improvements.

Test Design/Development involves the activities of designing and documenting test cases, test scripts, etc. The work done here is the result of test planning.

Test Execution and Defect Detection is really the heart of the testing. This is where the test cases or scripts are actually executed and results verified, problems researched, i.e. all the real day-to-day work of the test team. Defect detection is included since both static and dynamic testing are covered in this domain. Static testing is reviewing documents and code. This form of testing allows the detection of defects without actual execution.

The Practices

Now that STIR is defined, as well as structure and philosophy behind it, here is the heart of STIR. The ITCL recommended practices listed below are how we initiate our testers into a quality testing structure. We have these practices available on our internal test web site for availability to the entire test organization. For each of the practices, there is a link to a brief, 3-5 page description of the practice and the implementation steps at each of the STIR levels. We have kept the write-ups very short, since they are meant to be a starting point, not the definitive information on the practice. Given this structure, it's straight forward for the tester to see which the easier practices to implement are and which ones require more skill or resource. We developed this list of practices based on research done within IBM and across the IT industry, plus hours of discussion among the Top Guns. It has proved to be a good starting point for us.

The following list shows the implementation steps by STIR level for all of the ITCL recommended practices.

Practices which can be implemented starting at the **basic level**:

- **Management of Test Case Status and Overall Status:** This practice allows the test team to articulate, in a timely manner, where you are in a particular testing phase of the development life cycle.
  Basic: The results of executing test cases are tracked manually. The results are maintained in some kind of repository. This repository could be as simple as a piece of paper, but typically is a spread sheet, database, flat file, etc. Test results are plotted on an S-curve, as is the outstanding defect backlog.
  Intermediate: While the development of tests may still be a manual process, the tracking of their execution is under the control of automation. Automation is used to monitor and/or record the status of test execution, to support linkage to related defects and to enable reporting of test status.
  Advanced: Results are tracked automatically, with defects linked to failed test results. Predictive tooling is implemented to help determine when the appropriate quality level will be achieved, taking defect arrival rates, backlogs, and closure rates into account.
- **Formal Entry/Exit Criteria:** The implementation of this practice allows the tests that have been defined for a product to be executed in an effective and efficient manner and to find the types of defects that they are targeted to find, as well as help determine when testing is finished for each test phase.
  Basic: Simple test entry and exit criteria are established and enforced. Risk assessments are used for all criteria with a simple high/medium/low assessment.
  Intermediate: Metrics gathered are used to enhance and support entry and exit criteria. Historical data is used to hone risk assessments.
  Advanced: Entry and exit criteria use overall product metrics. Entry criteria are used to influence prior process steps.
- **Defect Management:** This practice includes the problem tracking, determination and evaluation mechanisms used during the development life cycle.
  Basic: Use a defect tracking tool to capture defects during a test and track them to resolution.
  Intermediate: Capture defect statistics at the end of each release and use it to produce basic product quality metrics. If doing integration test involving multiple development groups, the system is capable of successfully managing defect tracking and resolution across these groups.
  Advanced: Do defect trending and capture metrics across multiple releases to understand if product or process quality is making progress.

- **Teaming Testers with Developers:** Implementation of this practice involves getting testers and developers to work together to enhance knowledge of processes and products, and to create an atmosphere that is not adversarial. It should ultimately show its benefits in detection of defects earlier in the product cycle, improvements in test effectiveness and product quality, more efficiency in defect analysis and turn-around time as well as a decrease overall in the product development cycle.

  Basic: Obtain test and development leadership team support for the teaming of testers and developers, and get acquainted with each other.

  Intermediate: Become active earlier in the product cycle by becoming more involved with development work items, invite development to take part in test planning activities, and take basic steps to facilitate testers working with developers.

  Advanced: Get more involved with development by providing them with early test feedback and performing tests with them. Start to reach out to other product teams.
- **Testing to Requirements:** The implementation of this practice makes creating test cases and testing more dynamic, as each phase of the software development lifecycle is verified and validated against the requirement set. Requirements based testing is one of the primary ways to ensure that all of the needs of the customer have been coded, verified and delivered.

  Basic: Use requirements documents as the source of testing. Ensure the requirements are testable.

  Intermediate: Requirements can be traced throughout the system. Testers participate in requirements reviews and represent customer's voice.

  Advanced: All documents are traced back to the baseline requirements. Testers are fully involved throughout the process and ensure all activities relate to the original requirement document(s).
- **DBCS Test Method and Approach:** Double Byte Character Set testing is carried out to make sure that a product is suitable for marketing in the local language in Japan, Korea, China and Taiwan. DBCS testing should be considered as a specialized aspect of the overall product test.

  Basic: Test cases verify the ability to input, store, display, and print the DBCS characters correctly.

  Intermediate: Test cases verify the ability to exchange DBCS data across different operating systems and locales. The quality (including aesthetics) of the DBCS output is also verified.

  Advanced: Test cases verify that the product integrates into the entire product stack required for a DBCS solution.
- **Functional Specifications:** Functional specification documentation describes the 'what' information that is required for function and system testing. It should describe the product in terms of external functions, aiding testers in developing test cases. The implementation of this practice, when there are documented specifications, will provide many benefits, i.e. the detection of defects earlier in the product cycle, improvements in test effectiveness and efficiency, improvements in test coverage and increased product quality.

Basic: Obtain, in some form, basic documentation about 'what' it is you are going to test; apply it to your test plans and test cases.
Intermediate: Analyze the functional specification information provided; develop a list of information you need. Participate in document reviews to request additional information that will improve your test.
Advanced: Work with development to create the functional specifications. Use the information you now have to revalidate test projections, resource planning and schedules. Add a testability focus to the functional specifications. Start making plans for future automation of test variations and test case generation.

- **Testware Control:** The control of the test material required to run your test cases (programs, scripts, documentation, data, etc.) is vital to the maintenance of the test's most valuable asset.
  Basic: Testware is held in a central location with backup procedures in place.
  Intermediate: Repositories are used to manage key deliverables (test plans, test cases, scripts, inputs to test, outputs from test). Disciplined change management procedures are implemented to introduce changes on a carefully planned basis.
  Advanced: Intermediate version control is successfully applied to an integration test involving multiple applications, multiple development and/or test groups. Release control processes are in place. Distribution of testware from the repository to the test environments is automated.
- **Reviews and Inspections:** The purpose of this practice is to identify defects in requirements, design, code, test plans, test programs and documentation so they can be corrected before they escape to a later step in the development process.
  Basic: Informal reviews of test deliverables by others on the test team are a common and regular practice.
  Intermediate: Reviews conducted by the test team include key representatives from development and other affected areas. At least some testers take part in requirements and design reviews. Most review meetings are conducted more formally than simple walk-throughs.
  Advanced: Deployment of the reviews and inspections practice is pervasive and fairly consistent throughout the organization, with much greater rigor involved in planning for inspections, conducting the meetings, and ensuring that defects are properly resolved.

Practices which can be implemented starting at the **intermediate level**:

- **Code Coverage:** The main technique for demonstrating that the testing has been thorough is called coverage analysis. Simply stated, the idea is to create, in some systematic fashion, a large and comprehensive list of tasks and check that each task is covered in the testing phase.
  Intermediate: Code coverage is used for unit testing. Team has a tool that supports their efforts and automated test execution practice to allow fullest advantage.

Advanced: Code coverage is used for function and/or system testing where it is appropriate. Where applicable, coverage is used to minimize regression test suites.

- **Defect Analysis:** Problem defect analysis addresses understanding the root cause underlying the defect as well as a few other semantics like: how is was it found, what was done incorrectly, who made the error, when was it made, why wasn't it detected earlier, and how could it have been prevented?

  Intermediate: Defects are classified by a scheme and data is used to justify some decisions.

  Advanced: Defect data is used to implement and measure improvement actions and drive decisions.
- **Design for Testability:** Testability is the degree to which the system aids the establishment of test criteria and the performance of tests to determine whether the criteria have been met. When a product or solution is developed with "ease of test" as an attribute, it becomes easier for the test team to do the verification and validation. The test team can verify the function more efficiently and focus on other test tasks.

  Intermediate: Guidelines for testability exist and are used in reviews.

  Advanced: Test participates in defining the design and driving in testability.
- **Minimizing Regression Test:** This practice looks at techniques which aim to reduce the size of regression test suites. The issue is basically how to select tests from the entire pool of possible tests such that when they are executed the maximum amount of product functionality is exercised by the minimum number of tests.

  Intermediate: Analyze historical defect data to identify test cases which only detect defects which are also detected by other test cases or have a long history of never detecting a defect.

  Advanced: Select test cases from knowledge of how they exercise the underlying product under test, possibly by using a code coverage tool.
- **Customer Information into Test:** This practice describes some methods of bringing information about customers, their concerns and their environments into test.

  Intermediate: Start gathering customer information from internal sources such as people and forums, analysis of field data, and use this information to create use cases and/or implementing patterns in your environment.

  Advanced: Become directly involved with customers by working critical situations and customer site visits.
- **Automation of Test Execution:** The implementation of this practice provides a way for test to leverage their resources, increasing both the coverage and frequency of execution. This deals with the type of automation that most people think of when they hear the phrase "test automation". That is, the specific sequence of actions that are normally taken to execute a test case will be done programmatically instead of manually.

  Intermediate: Determine the short and long term benefits for test execution automation. Educate the test team in the fundamentals of

automated test execution. Develop and use automated scripts and/or programs that execute straight forward function type testing.
Advanced: Develop a test execution automation strategy. Design, develop, implement and deploy an automated test execution system. Maintain the test execution automation system from release to release. Continually expand automated test execution into more advanced areas of testing.

- **Automated Test Case Generation:** This practice seeks to automate the creation of test cases. Test cases are generated automatically rather than being created by hand.
  Intermediate: Test case generation tools are used. These allow testers to work at a higher level of abstraction and create test cases more quickly. Functional specifications and automated test execution are used. Basic defect analysis is in use to evaluate effectiveness of generated tests.
  Advanced: Model-based testing is deployed within the organization. A subset of testers has expertise building and debugging models and generating test suites from them. Effective feedback loops exist connecting the model building team with the development team. Defect analysis is done to facilitate continuous model improvement.
- **Automated Test Environment Control:** This practice details ways that you can reduce the amount of setup time required prior to starting execution of test cases. This practice also includes such things as the monitoring and logging of test case execution status, test system cleanup as well as the controlling and monitoring of all systems under test simultaneously from a central site.
  Intermediate: Incorporate a semi-automated or fully automated process that generates a desired test operating system configuration onto a test machine. Educate the team on fundamentals of automated test environment control.
  Advanced: Develop a test environment automation control strategy. Design, develop, implement and deploy an automated test environment from start to finish. Control and monitor the test system from a central site.

Practices which can be implemented starting at the **advanced level**:

- **Shared Test Environment:** This practice deals with how sharing of complex test environments is successfully accomplished, and addresses issues such as; funding, support and maintenance of the system, change management, configuration management, capacity management, connectivity, coordinating separate and/or concurrent use of the system, and problem determination and resolution of environment problems.
  Advanced: Successfully leveraging a common set of test environment resources across multiple groups and/or locations to support multiple or complex tests. This would include managing issues such as business case or funding, documents of understanding covering usage and service level expectations, support structure and procedures, communications between groups, problem management and test coordination procedures,

communication procedures, process improvement procedures, and how to determine the value-add provided by the shared environment.

The Usage of STIR

So now that STIR is defined, let's take a look at how to use STIR. The first step is to figure out which practices your organization currently uses and how they are used. Within IBM, a self-assessment was developed that asks the testers a set of questions for each recommended practice. The questions on the self-assessment are based on information that is provided in the list above. Additional questions were added to help determine whether or not implementing a practice is or is not a problem, thus invoking some discussion. Nothing that formal needs to be developed, the practice information that is provided can be used to discuss what your team does relative to each practice. The important thing is that you determine what your strengths and weaknesses are, and from there determine what your needs are.

Once you have determined what your needs are, you can identify areas for improvement. The use of STIR helps identify those improvement actions which have the greatest potential to be successful. In other words, if you have determined that you do some of the practices which are classified as basic, but not all of them, then other basic practices should be investigated. At the most, you would look at implementing some of the intermediate practices. You could also improve the way a basic practice is implemented by implementing steps at the intermediate level. What you do not want to do is attempt to implement an advanced practice.

After your improvement action plan is identified, the next step is to determine how to implement it. This includes the resources, skills and schedule available for the implementation, as well as the expectations. Initially, it is important not to be too aggressive; building on small improvements usually works better in the long term. Also, some method needs to be put in place to measure the improvements in order to determine if you have met the expectations.

As you implement your plan, remember to checkpoint your progress along the way to make sure that you stay on track and are actually making progress.

And, of course, remember that this is a journey. You need to periodically reassess yourself to determine your progress. Doing this makes a good, closed-loop process and keeps you focused on continuous improvement.

If you need to add some new practices, use the definitions of the levels that are provided to determine where the new practice fits in relation to other practices. Then break down the implementation of the practice into small steps and determine the levels of the steps by again using the definitions that are provided.

The Current Status of STIR

Just as our implementation of practices is evolving, so is STIR evolving. We have the basic structure down. The test teams are using STIR to identify improvement actions and they find it very helpful. We are very pleased with the enthusiasm and use of STIR.
We have identified 18 very good practices and have documented them. However, just as we encourage our test teams to continuously improve the implementation of each practice, we need to continuously improve the list of practices and not allow them to stagnate. We feel we have some holes; we certainly need more practices to bring in at the advanced levels to push our stronger test groups. Also, you may have noticed that there is no information about the engineered level throughout this document. We are working on exactly what it means for practices to be implemented at the engineered level. We have an idea of what we want, which is total integration of the process. But, in some ways the tool sets are not there yet to support that total integration and we are working on that. As this point we have not identified a test team which is implementing practices at the engineered level as we have defined engineered. So we have some time, but it's an area that we need to focus on to be ready to push the test teams.

A Summary of STIR

STIR is a great tool; it has been very well received by our test teams and has been very effective in helping them improve the way they are doing their testing. By using STIR and the self-assessment, it shows them what they are currently implementing and where they have to improve skills or gather additional data to implement some improvements. But, beware of a trap that is easy to fall into. Certain practices at certain levels require a certain level of skill. However, to turn that around slightly and say that if a team implements a practice at a certain level, the team must be at a certain level is not necessarily true. Or, conversely, to say that if a team does not implement a given practice at all or at a particular level, the team is not at that level or does not have the skill to be at that level is not true. They may choose not to implement a practice or a step for business reasons. STIR does not measure an organization; it measures level of difficulty of a practice or a step within a practice. Don't try to read more into it than is there. If what you really want to do is measure the maturity of your organization, then you need to use something that was developed to measure the maturity of your organization. If what you want is to get your team on a path of continuous improvement, then STIR can help.