Advanced Risk Based Test Results Reporting: Putting Residual Quality Risk Measurement in Motion

By Rex Black and Nagata Atsushi

Introduction

Analytical risk based testing offers a number of benefits to test teams and organizations that use this strategy. One of those benefits is the opportunity to make risk-aware release decisions. However, this benefit requires risk based test results reporting, which many organizations have found particularly challenging. This article describes the basics of risk based testing results reporting, then shows how Rex Black (of RBCS) and Nagata Atsushi (of Sony) developed and implemented new and ground-breaking ways to report test results based on risk.

Analytical Risk Based Testing

Testing can be thought of as (one) way to reduce the risks to system quality prior to release. Quality risks typically include possible situations like slow system response to use input, incorrect calculations, corruption of customer data, and difficulty in understanding system interfaces. All testing strategies, competently executed, will reduce quality risks. However, analytical risk based testing, a strategy that allocates testing effort and sequences test execution based on risk, minimizes the level of residual quality risk for any given amount of testing effort.

There are various techniques for risk based testing, including highly formal techniques like Failure Mode and Effect Analysis (FMEA). Most organizations find this technique too difficult to implement, so RBCS typically recommends—and helps clients to implement—a technique called Pragmatic Risk Analysis and Management (PRAM). You can find a case study of PRAM implementation at another large company, CA, at http://www.rbcs-us.com/images/documents/A-Case-Study-in-Risk-Based-Testing.pdf. While this article describes the implementation of the technique for projects following a sequential lifecycle, a similar approach has been implemented by organizations using Agile and iterative lifecycle models.

In analytical risk based testing, stakeholder interviews, requirements specifications, past defect history, and other sources of information are used to develop a categorized list of quality risk items, which are then assessed to determine, for each risk item, the likelihood of bugs related the risk item and the impact of such bugs should they exist in the system after release. Impact is typically determined from a business perspective, while likelihood involves consideration of technical issues. Likelihood and impact are then combined to determine an overall rating of risk for each risk item (see Rex Black's book *Managing the Testing Process, 3e*, for a detailed description of the process of quality risk analysis).

This catalog of prioritized risk items is then used to develop and execute tests. Tests are developed for each risk item, with the precise number of tests covering each risk item based on the level of risk. As tests are developed, they are given a priority based on the priority of the risk

item they cover. Then, during test execution, tests are run in risk priority order. This provides three immediate benefits:

- Tests effort is tightly calibrated to the level of risk reduction that any given functional or non-functional attribute requires.
- Tests are run in an order that maximizes the chances of discovering the most important bugs early in test execution.
- If necessary due to schedule pressures, less important tests (which are sequenced towards the end of test execution) can be eliminated.

However, if, during test development and execution, the test team uses a test management tool, a database, or even a simple spreadsheet to maintain traceability between risk items, the tests that cover them, the results of those tests, and bugs found during testing, the organization gains a fourth benefit: the ability to make fully-informed, risk-aware decisions about whether to release software based on the residual level of quality risk.

This benefit is of great importance, because a large number of project teams make release decisions without a full understanding of the current quality status of the system under test. This is because the standard test management reporting dashboards, based on charts and tables showing bug status and test pass/fail status, are at best indirect and imperfect measures of quality and quality risk. Like a shadow thrown by a candle in a dark room, the picture given by such charts and graphs is flickering, unclear in the details, and distorted. Risk based results reporting, when done properly, allows everyone, testers and non-testers alike, to see a clear, direct, steady picture of quality risk.

Basic Risk Based Test Results Reporting

How does risk based test results reporting work? There are three basic approaches.

The first approach, and simplest, approach relies on reorganizing the traditional metrics of testing, bug status and test pass/fail status, based on their relationship to risks. An example of this kind of reporting is shown in Table , using a hypothetical e-commerce application.

	Tests				Bugs		
Risk Category	**Total**	**Pass**	**Fail**	**Not Run**	**Total**	**Open**	**Resolved**
Browsing	72	52	0	20	23	5	18
Catalog	81	43	7	31	46	17	29
Shopping Cart	57	47	1	9	41	1	40
Checkout	66	17	2	47	40	12	28
Performance	91	26	6	59	28	27	1
Reliability	69	33	1	35	22	16	6
Usability	75	34	3	38	21	14	7

Table : Simple Risk Categorized Results Report

Based on this table, we can see that risk is relatively high in the non-functional categories of performance, reliability, and usability given the large number of open bugs and tests remaining to run. This situation with non-functional categories also illustrates an advantage of risk based results reporting. The low number of failed tests in these three categories might lead us to conclude—based on test pass/fail status alone—that quality risk is low in these areas, but the juxtaposition of the large number of open bugs in each area shows that the risk is in fact high.

You can also see, in this table, that the risk situation is serious in the functional catalog area. Browsing and shopping cart areas look much better. Checkout, while in better shape than the catalog, still has a significant number of open bugs and unrun tests.

This kind of tabular approach is simple to implement—given the traceability discussed above—but it does suffer from some disadvantages. For one thing, we still have a distorted picture of the quality risk, because there is no indication of the level of risk of the failed tests, the unrun tests, or the open bugs. For example, suppose that the risk items related to the open bugs, failed tests, and unrun tests for catalog are rated very low risk in terms of likelihood and impact? In that case, the residual level of risk is lower for the catalog than this table makes it appear.

To resolve the distortion issue, we can introduce the concept of risk weighting, which is the second basic approach to risk based results reporting. For example, suppose we use a scale from 1 to 5 for both likelihood and impact, where 5 represents the very highest level of risk and 1 the very lowest. (This is an ascending scale; we can also use a descending scale where the smaller numbers represent the higher level of risk.) Multiplying likelihood and impact together gives us a risk score for each risk item, and this risk score can be inherited by each test case and bug report associated (through traceability) with each risk item. This allows us to weight the numbers in the table by risk. To keep the numbers reasonable, we can normalize (divide) the scores by 25, the maximum risk score.

The result might look something like Table . In this table, the risk scores are calculated by adding the risk scores for each test associated with the risk category and for each bug associated with the risk category, and dividing those scores by 25 to keep the numbers from being enormous. Thus the highest risk test or bug report has a score of 1, while the lowest risk test or bug report has a score of 1/25 or 0.04. Thus we show two decimal places of precision in the table.

	Tests Risk Scores	Bug Risk Scores					
Risk Category	**Total**	**Pass**	**Fail**	**Not Run**	**Total**	**Open**	**Resolved**
Browsing	16.20	15.91	0.28	0.01	3.82	0.65	3.17
Catalog	3.75	1.05	0.27	2.43	3.07	1.41	1.66
Shopping Cart	4.32	1.86	0.02	2.44	22.50	0.90	21.60
Checkout	19.20	3.30	1.11	14.79	3.00	0.48	2.52
Performance	3.67	2.82	0.31	0.54	1.32	0.21	1.11
Reliability	95.00	79.20	3.61	12.19	0.88	0.04	0.84

Usability	25.00	9.66	0.89	14.45	0.79	0.17	0.62

Table : Weighted Risk Score Results Report

This report tells us a somewhat different story than the previous one. Catalog still looks risky, with a relatively high score for failed and unrun tests as well as open bugs. Browsing still looks much less risky, but notice that the shopping cart now tells a rather concerning story. We have found very serious bugs in this area. While most of those bugs are resolved, in terms of risk scores, more than half of the risk coverage planned for the shopping cart remains to be run. The testing principle of bug clustering tells us that we can expect to find many more serious problems in this area. Performance and reliability look much less risky, while usability still looks risky.

While this risk weighting of test and bug metrics resolves the distortion in the previous table, there can easily be too much detail. The total number of risk categories can easily exceed 20 or 30, rather than being limited to seven as in this simple example. So, while this level of detail is useful to the tester and test manager as they adjust the risk analysis—and thus the tests—based on a greater understanding of risks, reporting the level of detail shown in Table to project participants and stakeholders beyond the test team is likely to overwhelm them.

To resolve this issue of excessive detail for some stakeholders, we can introduce the concept of risk status classification, which is the third basic approach to risk based results reporting. In risk status classification, we retain the concept of risk weighting. However, instead of using the test status and bug status metrics directly, we use test status and bug status to classify the status of each risk item into one of three groups:

> **Green**: A risk item is classified as green when all associated tests have been run and have passed, and any associated bugs have been resolved (either fixed or deferred).
>
> **Red**: A risk item is classified as red when at least one associated tests has failed and/or at least one associated bugs remains open (i.e., unresolved).
>
> **Black**: A risk item is classified as black if it is neither red nor green, which means that, while no associated tests have failed and no associated bugs remain open, one or more associated tests remain to be run.

Now, by adding the risk scores for the risk items in each classification, we can easily produce a chart such as the one shown in Figure .

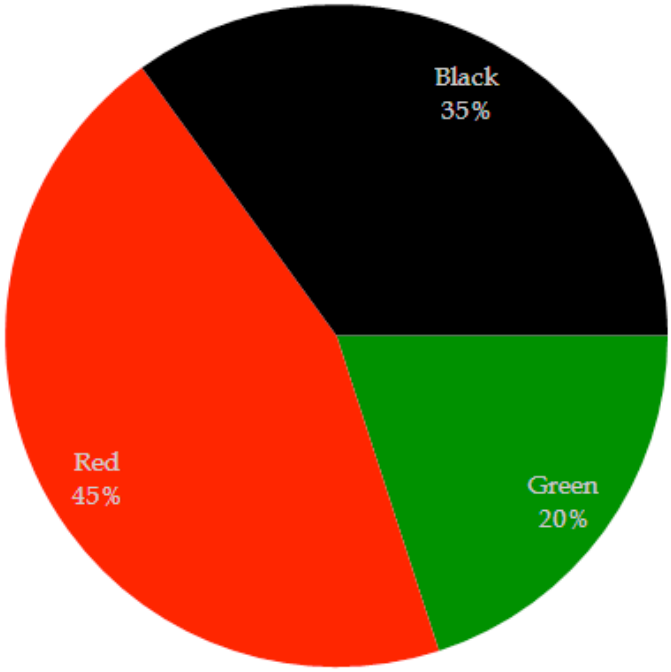

Figure : Risk Score Pie Chart

This chart has a number of advantages over Table . It is much more immediately intuitive, due to the lower level of detail presented. However, at the same time, it presents an even less distorted view of risk than did Table (which was in turn less distorted than Table). Figure presents a less distorted view of risk for two reasons. First, the calculations are done directly on risk items themselves, rather than abstractions like bug and test metrics. Second, the calculations are done at the individual risk item level, not using categories of risk items, thus making the measure more fine-grained. Finally, while the level of detail is reduced at first, it provides the ability to zoom in on any of the risk categories, because the underlying data behind the chart can be used to produce a detailed report of the risk items in each of the three categories, along with the status of the associated tests and bug reports for each risk item.

However, this pie chart representation still has a disadvantage that exists for the tabular presentations as well. We can't easily compare the situation currently with the situation in the past, so we don't know if the trend is positive, negative, or flat in terms of residual quality risk. Of course, we could use small multiples to present a series of pie charts from left to right corresponding to the risk status at chosen periods of time (e.g., daily, weekly, etc.), as shown in Figure . (For a detailed discussion of the concept of small multiples, see Edward Tufte's *Envisioning Information*.)

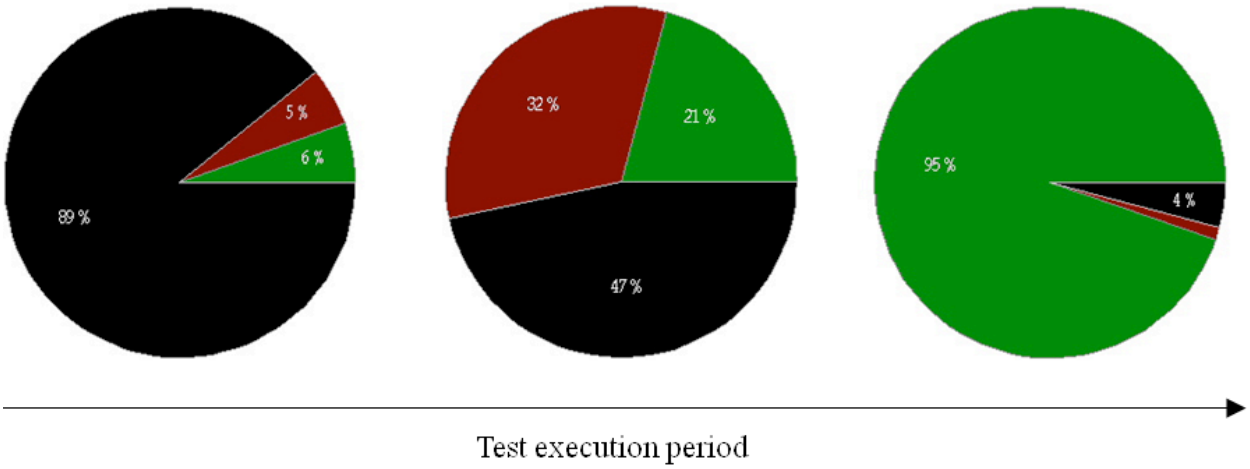

Figure : Multiple Risk Score Pie Charts, Shown Over Time

While Figure does show a trend towards reduction of residual risk as the project continues, the visualization of the trend in such a series of small multiples does lead to clear extrapolation toward the future. So, the three basic approaches to risk based results reporting—categorization of test and bug status, risk weighting of test and bug status, and classification of risk items—progress from coarse-grained and distorted to fine-grained and accurate, but leave us with only limited resolution of trends over time. Now, let's turn our attention to how Sony uses risk based testing, and how Sony's need to see risk trends more clearly lead to a significant advance in risk based results reporting.

Sony's Use of Risk Based Testing

Rex Black had come to Sony a few times to provide the RBCS Advanced Test Manager training class, which includes material and exercises related to risk based testing. Following these courses, Nagata Atsushi was tasks with introduce risk based testing into a QA section. At that time, he started work on implementing the risk identification and risk analysis processes included in the PRAM methodology mentioned earlier. Sony used RBCS' PRAM templates for quality risk analysis (these templates are described in Rex Black's book Advanced Software Testing: Volume 2).. Risk based testing is currently implemented on some teams at Sony, though some challenges remain.

One challenge is reaching agreement with the stakeholders on the identified risks and their level of risk as determined during the risk analysis. Test engineers created the risk analysis tables and then reviewed those with the design team and project management. The project members must understand that the entire team shares the product risks, effectively, because problems with the product are everyone's problems.

Another challenge is ensuring the proper scheduling of development and other supporting functions to enable risk based testing. A test team needs test items (units under test) and test environment (resources) in a timely fashion to support the execution of tests in risk order; otherwise test execution is forced to proceed in a less-than-optimal sequence, due to missing functions, improperly configured environments, and other constraints. Thus, the entire team must

not only accept the results of the risk analysis as a basis for testing, as discussed in the previous paragraph; they must also accept the implications of the risk analysis for the development schedule. Currently, while the QA team enjoys good support from the design team, project management is not fully on-board.

Finally, another challenge involves the connection between the risk analysis and the subsequent test design and test results monitoring. Sony remains focused on better connecting the test design process with the risk analysis results. In this article, we'll focus on the final element, that of risk based test results monitoring and reporting.

Advances in Risk Based Results Reporting at Sony

In a previous section, we mentioned that Sony needed to understand not only the current level of risk, but also trends in terms of the remaining level of risk. They also wanted to see this trend in relation to the testing to be done, so that they could visualize how test execution was affecting the remaining risk. In this section, we'll see the reporting technique devised by Sony to do so, including how the technique was revised over time.

Let's start by considering what the ideal trend would look like. In the ideal risk based test execution approach (after having overcome the logistical challenges mentioned in the previous section), we execute the higher priority test cases earlier in the test execution period, and the lower priority test cases later. This sequencing would result in a rapid reduction in the amount of residual quality risk as test execution begins, with the rate of risk reduction per testing day going down as testing concludes.

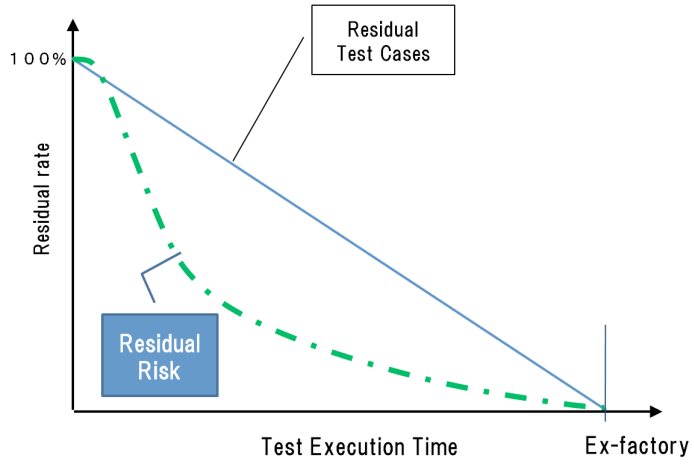

Figure : Ideal Residual Risk Pattern

Figure shows the ideal residual risk pattern. The top line shows the test cases remaining to be run, which go down linearly over time (assuming no blockages and constant test resources available). As we run test cases, once a test case has passed, then the associated level of risk for that test case is reduced. As mentioned above, a test case associated with a higher level of risk takes higher priority in risk based testing. Assuming that we can execute the higher priority test cases earlier and those tests are passing, the amount of residual risk decreases as shown in the lower line in Figure .

For example, at the half-way point of test execution, even though half of the tests remain to run, the residual risk is less than 30%. Therefore, the test team is making good progress in testing. In addition, the project stakeholder can also feel confident about product quality.

When the testing is completed, one might expect that the residual risk would go to zero as shown here, though in a moment we'll explain why that cannot happen. Before getting to this issue, however, let's recognize that ideal situations are seldom achieved. Given the reality of projects at all organization, Sony included, actual testing shows results similar to Figure . Let's consider two key points from this chart.

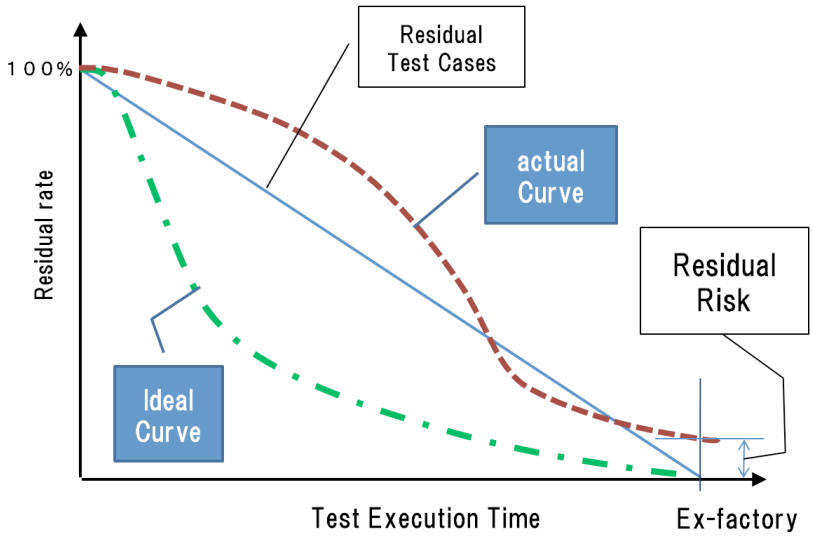

Figure : Actual Residual Risk

First, even if it were possible to achieve a level of zero residual quality risk, doing so might involve more time and resource than an organization would (or should) invest. So, it's better to arrive at an agreement about a level of residual risk that is deemed acceptable for product shipment. This level of acceptable quality risk can be discussed and agreed with the stakeholders, including the project or product manager who is responsible for the final release decision. Notice that a chart such as the one in Figure fulfills the need mentioned at the beginning of this section, to allow project stakeholders to understand not only the current level of risk, as shown in Table and Figure , but also the risk trend in time and the relationship to test execution.

Second, notice that we can see worst-case situations, such as that shown in the first half of the test execution period in Figure , where the level of risk is not falling as rapidly as we would want or expect. When the residual risk curve is above the residual test cases curve as shown here, that is a warning sign that calls for intervention. We should understand what the causes are—which would likely include factors such as a high rate of failing tests, a high number of blocked tests, or the inability to run tests in risk order due to missing functions or environmental constraints—and institute correction actions to change the trend. We can see the effect such corrective actions would have in the second half of the test execution period in Figure .

With this theoretical discussion in hand, let's look at what happened when Nagata introduced this approach to their projects. Figure shows an example of residual risk chart from an actual Sony project.

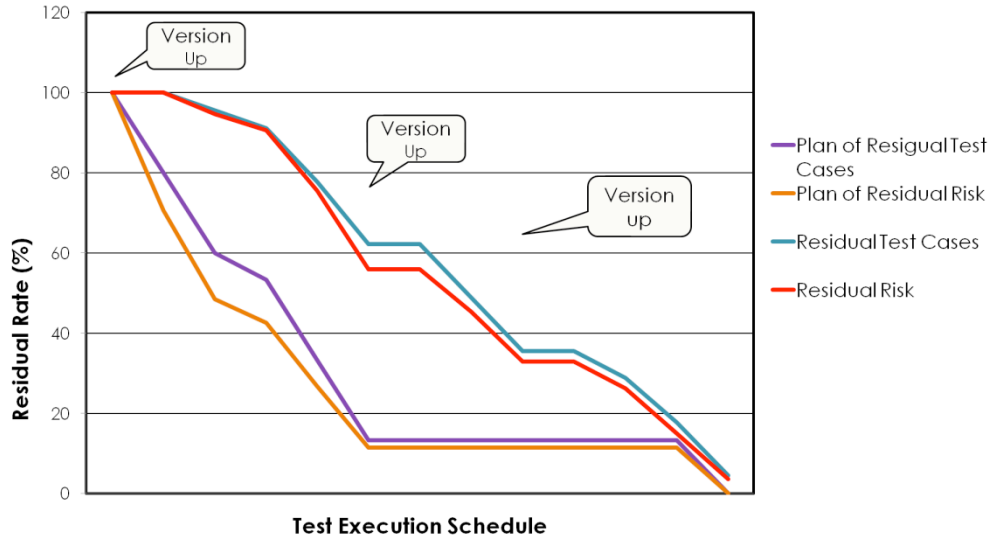

Figure : An Example of the Residual Risk Chart on a Sony Project

As you can see, Figure shows both planned curves and actual curves. This is another useful refinement, because it makes clear expectations and highlights variance from the plan in terms of test run and pass and in terms of quality risks mitigated. The planned curves are the same as in Figure .

As shown, test execution had a problem. The code version was upgraded frequently in response to bugs found and fixed. So, at the beginning of test execution, the test team needed to perform a significant amount of confirmation testing; in addition, the large number of bugs resulted in many tests that could not proceed or were postponed. Since the order of test execution was changed due to delays in delivery of high-risk functional areas, the rate of reduction of the actual residual risk was slower than planned. At the ex-factory point (i.e., release to production), some test cases and residual risks still remained. Fortunately, the level of risk was low. The test team and the other project stakeholders had sufficient confidence, and they agreed to ship the product based on the residual risk analysis.

During a recent consulting trip to Japan, Rex and Nagata reviewed Sony's implementation of risk based testing, including this example of risk based results reporting. We are both pleased with the technique that Sony has developed, which does a fine job of putting risk reduction trends in context with test execution trends and the project schedule. Rex did suggest a refinement, which will go into future reports.

In Figure , the target residual risk level is zero. However, while the calculations can show residual risk as going to zero, in fact it does not. While we can reduce the likelihood of undetected failures through our test coverage, we cannot reduce the possible impact associated with such failures without removing key features from the product. So, some level of residual risk always remains. There is a minimum level of risk which is essentially inherent in any product. The key point is that the risk is at an acceptable level. The revised risk pattern, taking this consideration into account, is shown in Figure .

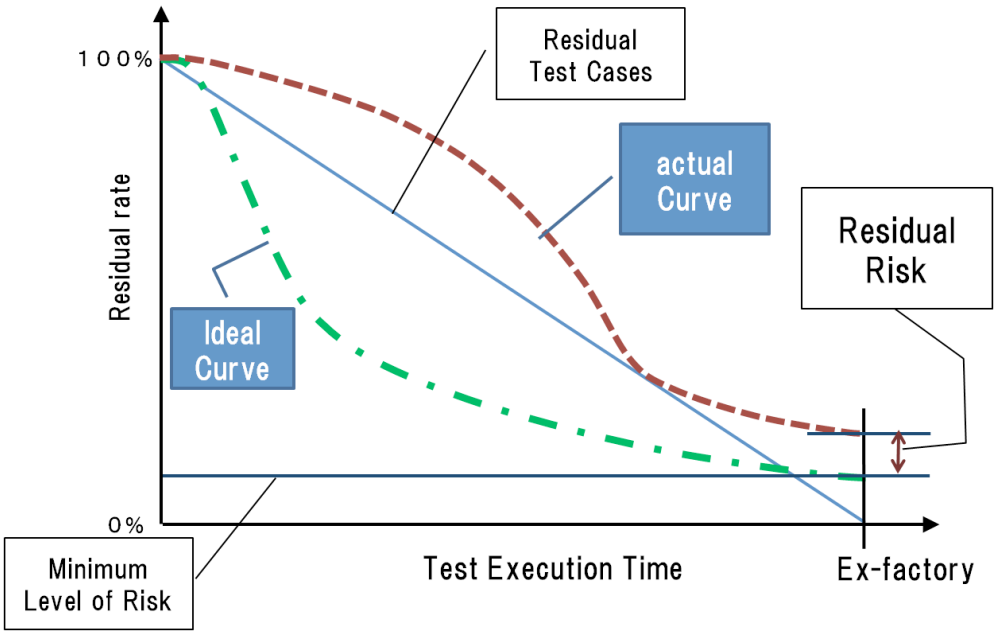

Figure : Revised Residual Risk Chart

Conclusion

In this article, we have discussed ways to help test teams and organizations make risk-aware release decisions through risk based test results reporting. We've shown some of the rudimentary ways in which organizations have done so using tables. This leads to the next logical progression, the use of a pie chart to show residual risk. However, we now believe that a combined test execution and residual risk trend chart, showing planned and actual trends for tests and risks, provides the most fine-grained, accurate, and actionable insight to the project stakeholders. Sony's implementation of such a trend chart constitutes a new and ground-breaking way to report test results based on risk. We encourage other organizations that are using risk based testing to adopt this powerful approach.

Author Biographies

With over a quarter-century of experience, Rex Black is President of RBCS (www.rbcs-us.com), a leader in software, hardware, and systems testing. For sixteen years, RBCS has delivered consulting, outsourcing and training services to clients ranging from Fortune 20 companies to start-ups. Rex has published six books which have sold over 50,000 copies, including Japanese, Chinese, Indian, Hebrew, and Russian editions. He has written over forty articles, presented hundreds of papers, workshops, and seminars, and given about seventy-five speeches at conferences and events around the world. Rex is also the immediate past President of the International Software Testing Qualifications Board and the American Software Testing Qualifications Board. Rex may be reached at rex_black@rbcs-us.com.

Atsushi Nagata is a Quality Engineering Manager in Sony Corporation. Atsushi is in charge of internal consulting on software testing process improvement and software development process

improvement. Atsushi was a member of the team that translated Rex Black's book *Foundations of Software Testing: ISTQB Certification* into Japanese. His current interests are risk based testing, test design methods and software requirements review techniques. He is a certified Agile Inspection Leader and a certified Agile Inspection Process Owner. Atsushi may be reached at atsushi.nagata@jp.sony.com