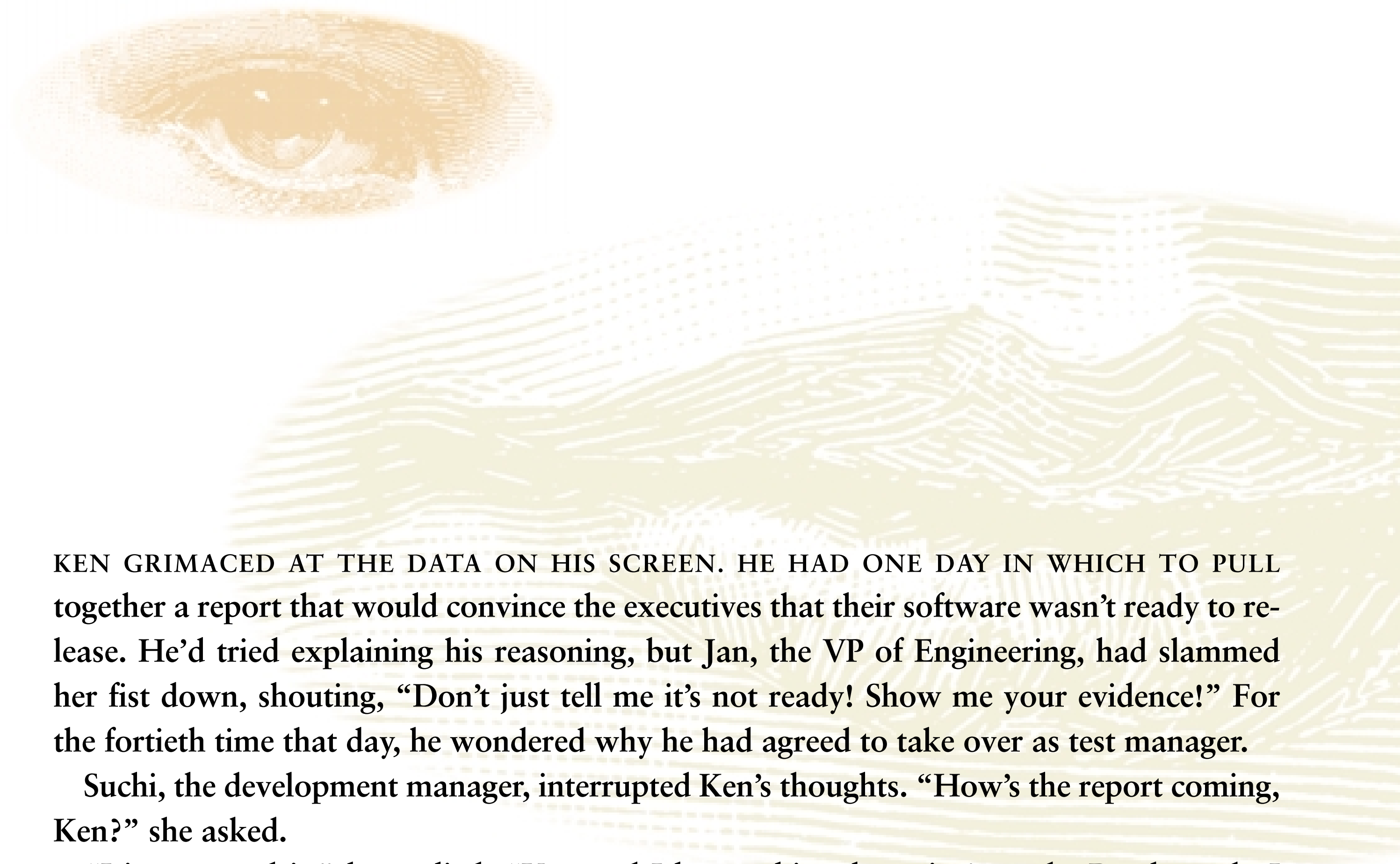
INFO TO GO

- Data can tell multiple stories; it's up to you to decide which one to tell and the best way to present it.
- Stacked line and stacked bar charts show more information, more clearly than a chart with multiple lines.
- Use intuitive colors whenever possible.
- Less is more—avoid cluttering the chart with 3-D effects, fancy patterns, or decoration.
- Labels and call outs are more readable than legends.

SHOW *and* TELL

Six simple things you can do to make your graphs more effective

BY KATHY IBERLE AND ELISABETH HENDRICKSON



KEN GRIMACED AT THE DATA ON HIS SCREEN. HE HAD ONE DAY IN WHICH TO PULL together a report that would convince the executives that their software wasn't ready to release. He'd tried explaining his reasoning, but Jan, the VP of Engineering, had slammed her fist down, shouting, "Don't just tell me it's not ready! Show me your evidence!" For the fortieth time that day, he wondered why he had agreed to take over as test manager.

Suchi, the development manager, interrupted Ken's thoughts. "How's the report coming, Ken?" she asked.

"I just started it," he replied. "You and I know this release isn't ready. But how do I make the numbers show that?"

"First, let's try to nail down *how* we know it's not ready. Then, let's figure out what we can *show* them so that what we *tell* them rings true," explained Suchi.

1 Plot Your Story

As testers, we live and breathe data. From bug statistics to test coverage metrics to performance stats, we often find ourselves up to our elbows in spreadsheets trying to help others visualize what we've found. There's a saying that data doesn't lie. It may be true that raw data doesn't lie, but it doesn't tell a story either. When we convert raw data into charts and graphs, we imbue the data with meaning. And like it or not, how we design our graphs has a big effect on the story that the data tells.

First we have to decide which of the many possible stories we want to tell. As the old 1950s television series "Naked City" said, "There are eight million stories in the naked city—here is just one of them." Any reasonably comprehensive set of data could enable us to tell multiple stories from multiple perspectives. The trick is to choose only the stories that say what needs to be said in a given situation.

One way to focus the story that your data tells is to use the **Goal-Question-Metric** (GQM) technique. Victor Basili and his associates at the University of Maryland created the GQM technique in the 1980s. In the words of Johanna Rothman (in her chair report from ASQ Boston in July 1997), "The fundamental assertion behind GQM is that in order to measure something, you must have a *goal.* (There is no point to measuring something you are not going to use.) To illuminate the goal, you ask *questions* that help you figure out when you have met the goal. Those questions have answers that can be *measured.*"

As you formulate your questions, you should also begin thinking about how to combine those measures into illuminating charts. Time, tests, status, bug counts, severity, and modules are all variables that could be measured. Charts are a way to compare or show a relationship between two or more variables. For example, charts may show the number of bugs found over time, the number of tests failing compared to the whole, or the severity of bugs found by module.

If we were designing a metrics program from scratch, we might have a lot of leeway in how we turned the questions into metrics. In Ken's case, he has only the defect and test execution data that has been gathered on the project to date. He cannot retroactively gather more. So he has to determine which of the available measurements will address the questions. Let's listen in on more of the conversation between Ken and Suchi.

Ken sighed. "Suchi, do you remember Jan's pep talk today? She put up a chart based on numbers I had sent her, comparing the number of bugs found and resolved each week of the project. She pointed to it proudly, saying, 'The find rate is slowing down and we're fixing more bugs than ever! We're almost done. Another week to test and we can ship!'

What would convince Jan that things aren't as good as they look?"

He pulled his attention back to the numbers currently on his screen. "Maybe we should show the rate at which serious bugs are being found. We may not be finding as many overall bugs as before, but the ones we are finding are very serious, and tell me there may be a lot more beneath the surface." Ken scrolled through the defect data, showing Suchi showstopper after showstopper.

"The deferral rate bothers me too," Ken continued. "Sure, we all agreed to defer the bugs on a case-by-case basis, but I'm not sure that anyone on the project team is aware of how many bugs we've added to the backlog."

"What about test execution status?" Suchi asked. She'd been alarmed by how slowly the test effort seemed to be going.

Ken, Suchi, and Jan all share the same goal: releasing a solid product as soon as possible. During their conversation, Ken and Suchi asked a number of questions about progress towards the goal: Are serious issues still being found? Is test progress on track? Will the defects be fixed by the release date?

To answer these questions, Ken and Suchi realized that they would need to show:

- The bug find rates for serious defects as compared to less serious problems
- The effect of deferrals on the defect backlog and overall product quality
- The planned vs. actual test rates

But how do you turn these numbers into compelling charts? Ken pulled out the line chart Jan had made and gave it to Suchi. "How can we make this chart say all that?" he asked her.

2 Choose the Right Data

Suchi looked at Jan's chart. It looked good, but it didn't feel right. Jan's words came back to her, "The find rate is slowing down and we're fixing more bugs than ever." Then she realized what was bothering her. "Ken, this resolved line includes *all* the bugs we've moved off our plate, including the rejected, deferred, and fixed defects. Of course these numbers look good! We were trying to address the backlog in this release, so we've resolved more bugs than we've found. But are we fixing enough of our new defects?"

Ken suggested, "Let's add a line for the fixed defects and see what it looks like." He quickly penciled in a line for the defects that were really fixed, rather than deferred to a later release or rejected as duplicates. Now the chart showed a quite different story. The fix rate had

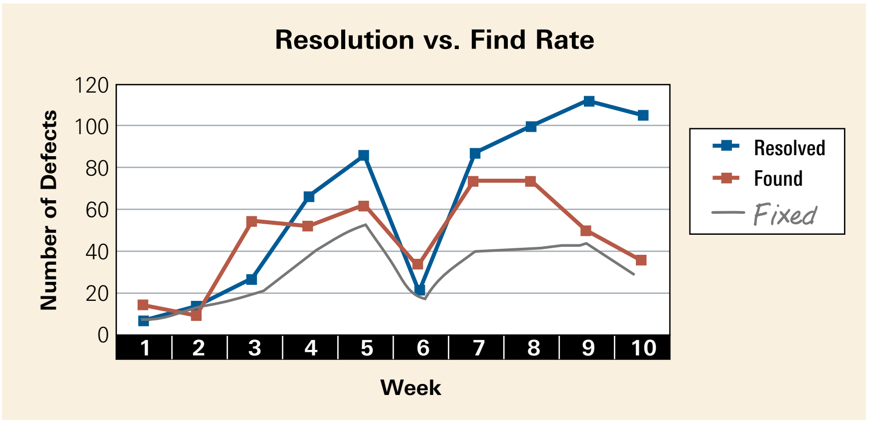


Figure 1: **Jan was basing her view on a chart that showed the number of bugs resolved, including those that were rejected and deferred. Ken penciled in the number of defects actually fixed by this project and suddenly the story changed.**

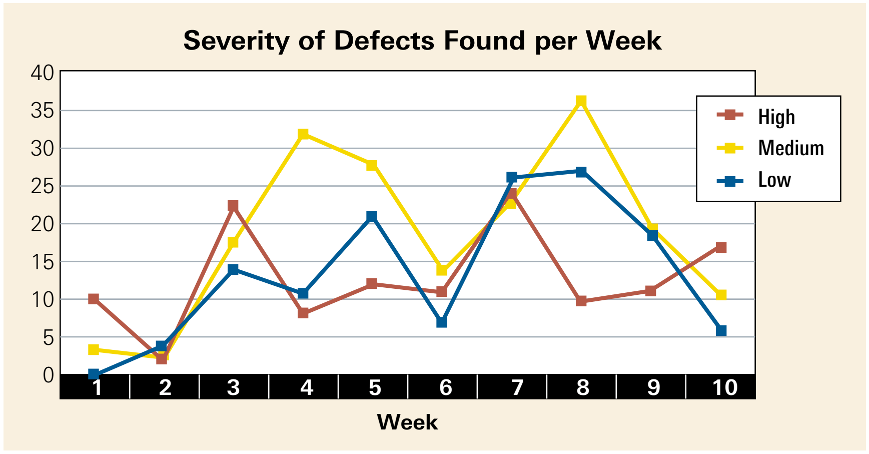


Figure 2: **Each line represents the number of defects found by severity category. Not only is the graph difficult to read, but it also doesn't show the total number of bugs found each week.**

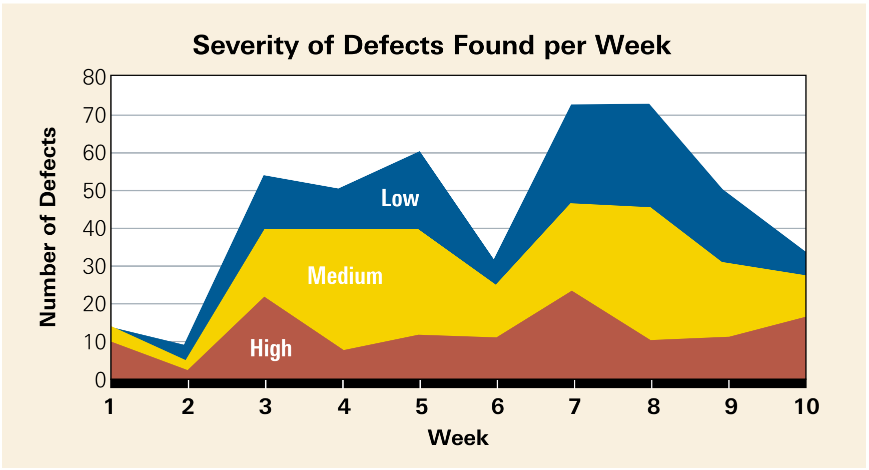


Figure 3: **This chart shows the same data as in Figure 2, but by using a stacked line chart, Ken can show additional information. For instance, the chart now shows the total number of bugs found each week.**

ANNIE BISSETT

been below the find rate for the entire course of the project (see Figure 1).

3 Choose the Right Chart for the Job

Ken returned to his original concern: last week's sudden rise in serious defects found. In order to tell this story, he had to plot the find rate for each severity separately. Ken used a line chart to show a numerical trend over time—the defects found per week by severity (see Figure 2). But the resulting chart looked like a jumble of lines and didn't get his point across. How could he make it better?

Ken converted his graph to a **stacked line chart.** A stacked line chart (also called an area chart or a geology chart) is a rich way of presenting the same data in a format that allows multiple comparisons. Ken placed the "High" defects on the bottom so he could see them most clearly. This format also shows some additional relevant data—the top line is the total number of defects found each week, and a slice down through the layers gives an approximate picture of the distribution of severities in a given week.

Spreadsheets offer numerous types of graphs, from simple two-dimensional line, bar, and pie charts to three-dimensional charts, stock charts, and bubble graphs that incorporate more complex data. Before you blithely accept the default chart type, consider your options. Another type of graph may serve your needs better.

The stacked line chart used in Figure 3 now clearly shows the rise in serious defects found over the last few weeks of testing. However, Ken was still bothered by Jan's assertion that the development team was resolving bugs quickly. "Suchi, I know your team is resolving bugs quickly, but an awful lot are just being pushed off to the next project. I think the backlog is rising, not falling. How can I show that?"

"Well, we've been using a pie chart to show the percentage of defects that have been fixed, deferred, or are still open each week. But how do we show that over time? What if we use a stacked bar chart instead?" suggested Suchi.

Pie charts are an excellent choice for showing percentages of a whole at a single point in time. However, it's difficult to see the progression from week to week by looking at a series of pie charts, or worse, trying to remember what last week's pie chart looked like. A bar chart is a better choice for showing a trend in percentages over time. The **stacked bar chart** in Figure 4 shows the proportion of defects that were fixed, rejected, or deferred each week, which reveals a trend.

Suchi looked at the chart. "Well, we're telling more of our story now. You can easily see that at the beginning of the project we were fixing most of the bugs we found, but at the end of the project we are deferring most of the bugs. I had a feeling that was happening. What did that do to the overall number of defects in the product?"

Ken said, "For that, we need to show the number of defects that were in the deferred state every week, not the number that were *moved* into deferred that week. Oh, and the open defects count also."

Ken went back to the spreadsheets for the correct data and created a stacked line chart to show the trend over time. He added a line showing the sum of open and deferred defects (see Figure 5).

"Look, Suchi, the total number of known defects in the product has not been dropping. The project started with a backlog, and we have not reduced that backlog at all."

"You're right," added Suchi. "And one of the goals of the release was to reduce the backlog of bugs, so the release isn't meeting that goal."

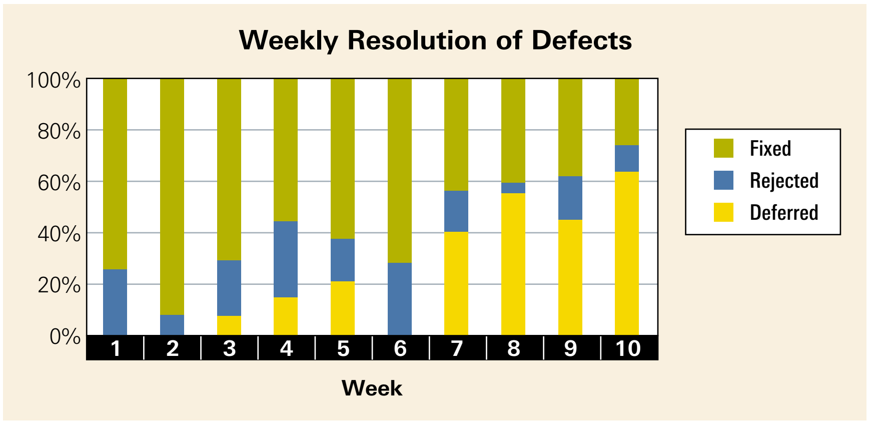


Figure 4: A stacked bar chart shows that the percentage of defects deferred is growing, while the proportion of bugs fixed each week is shrinking.

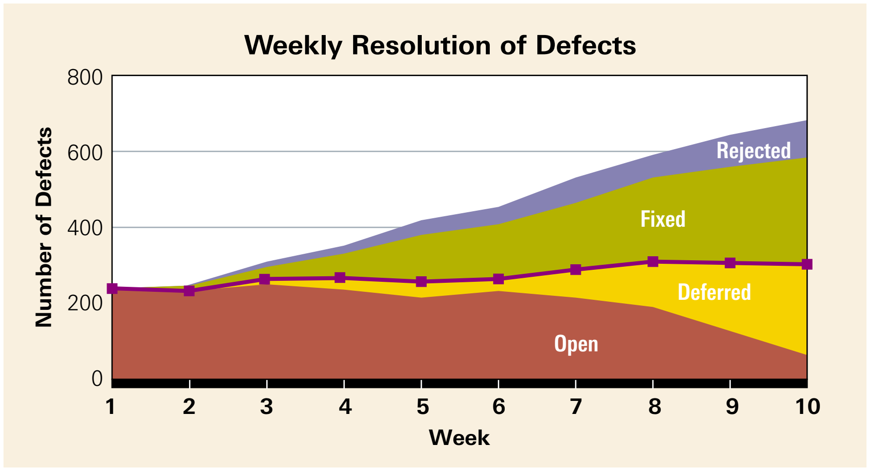


Figure 5: This stacked line chart shows the disposition of bugs and the total number of bugs. The purple line emphasizes that the backlog of bugs for this product has remained relatively constant.

ANNIE BISSETT

4 Choose the Axis Values

Now the graphs showed a great deal more. But the story was still not complete. "Ken, I'm still worried about the test execution status."

"Hmm. The testers kept telling me that they couldn't complete tests because of blocking defects—which is one reason we're behind, I think. Let's look at blocked tests over time."

His first graph had a huge spike in the second week, which dominated the entire graph.

Ken said to Suchi, pointing at the spike, "Hey, do you remember what caused this?"

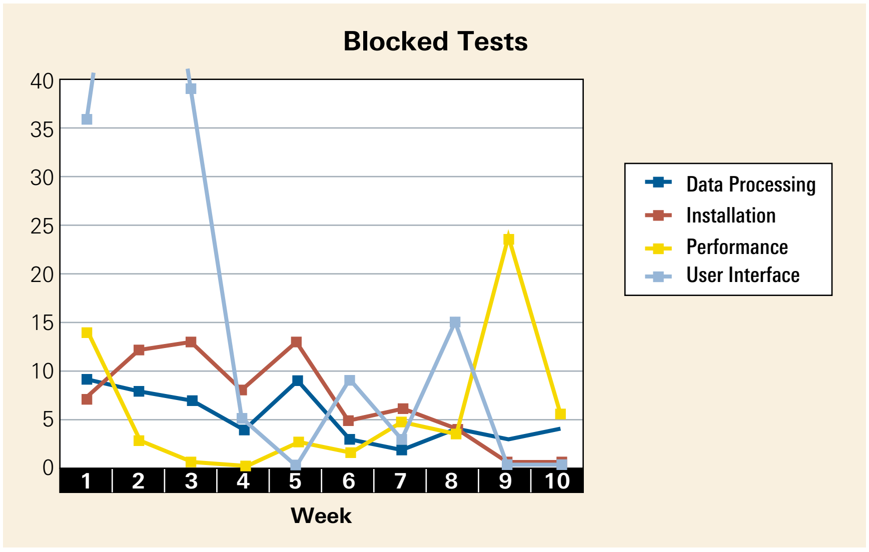


Figure 6: **Note that the spike in week two is literally off the chart. Ken chose to adjust the scale of the chart so that this anomaly would not dominate the chart and make it difficult to see the impact of blocking bugs later in the project.**

Suchi thought for a moment. "Wasn't that the week that the wrong version of the main menu file got into the build, so that a whole big chunk of the user interface couldn't even be reached?"

Ken responded, "Oh, yeah, so the entire test suite was blocked. That's not really all that interesting—let's chop off the top part of the scale so we can take a better look at the later weeks."

The resulting chart cut off the top part of the spike in week two, but it made the spikes in later weeks much more prominent (see Figure 6). With a change in the range of the y-axis, it was more obvious that performance testing was blocked toward the end.

When deciding how to best display values, consider changing the axis values to reflect the story that you want to tell. You'll want to choose the scale and range for numerical data, and consider categorizing your data.

Scale When dealing with numerical data, you'll typically use a linear scale, meaning that each interval along an axis represents the same range of values. However, there are occasions when you may use a logarithmic scale where each interval is some multiple of the one that preceded it. You may also want to reverse the numbers so that, for example, the bars on a bar chart hang down rather than reach up.

Range By default, graphing packages typically create the y-axis with a range of values from zero to slightly above the maximum. This can have unrealistic side effects such as percentage graphs going up to 120 percent. Choose your scale deliberately, not by default. It's usually best to start at zero, but when you're dealing with small changes in large values, you may want to focus the graph on the upper end of the range.

Categories Look for ways to categorize your data. For example, when you're looking at variables over time, you may want to group days into weeks or months. Or you may want to group data into distinct time periods. Projects have a distinct beginning, middle, and end during which the team is focused on different activities.

5 Use Colors and Lines

Suchi looked at the graph of blocked tests. "I see that the performance testing hasn't been going very well. But what about the overall test coverage? How many of the tests have actually been run?" Ken quickly put together another graph, using the spreadsheet defaults (see Figure 7).

Ken's goal is not to design a *pretty* graph but rather to illustrate relationships. For instance, a dashed line can represent a goal, while a thicker-than-usual line can draw attention to particularly important data. Similarly, color can draw attention and provide meaningful information at a glance. International traffic standards have given the colors red, yellow, and green meanings that people who live near paved roadways will understand immediately. To maintain the connection between the color and its meaning across a set of graphs, use colors consistently. Don't use blue to represent low-severity defects on one chart and green for low severity on another chart.

Whatever colors and line styles you choose, make sure they are clearly distinguishable from one another in all the formats in which the graphs will be presented. For example, if you'll be printing them in black and white and then photocopying them, run a test page first to ensure that the colors haven't all faded to a uniform gray.

If you need more than eight colors or line styles, your graph is too busy. The point is to tell an accurate story—not to impress the audience with your artistic talents.

Ken changed the colors on the test coverage chart to draw attention to the worrisome data and emphasized the planned number of tests with a heavy line.

6 Add Labels and Remove Extraneous Details

"I like the story it's telling, Ken," said Suchi, "but the legend is using up so much space. And won't people ask what happened in week six?"

Ken deleted the legend in favor of labels on the data. He also added a call out to point out that week six was the Thanksgiving holiday week (see Figure 8).

It's far too easy to create impenetrable charts. Unless you specify labels (including units) for your axes, no one will know what your charts measure.

When there are multiple categories, as in stacked line or bar charts, a legend or labels are necessary to identify them. Edward Tufte suggests replacing legends with self-descriptive data. He argues that legends are difficult to translate and add nothing to a well-constructed graph anyway. Many of the charts in this article use labels right on the data rather than forcing you to decipher what area represents what data from a tiny square of the same color.

Spreadsheets tend to add unnecessary details to graphs. In particular, consider whether you can simplify your chart by removing gridlines, 3-D effects, and legends. Anything that clutters up your chart will get in the way of your audience's understanding of the story you're trying to show.

ANNIE BISSETT

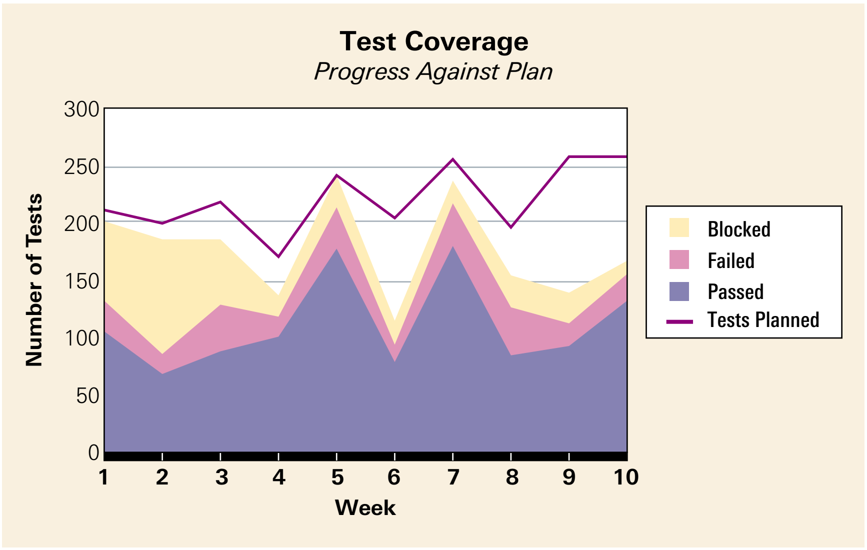


Figure 7: This mixed-type chart shows the tests planned and the status of tests actually executed. This chart is displaying the data using the defaults of the graphing program. The data is all there, but the chart doesn't have the "punch" that Ken wants it to have.

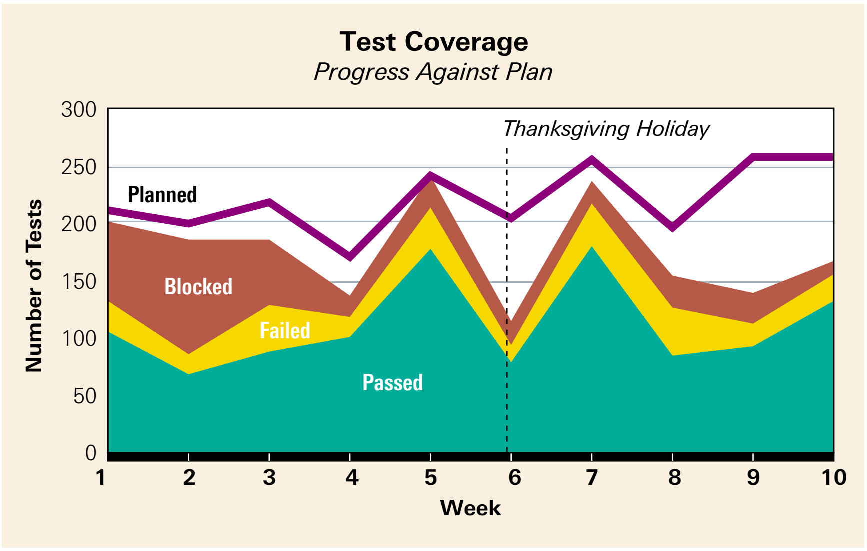


Figure 8: Ken made conscious choices on how to use color, line weight, and background on this chart. He eliminated clutter and added contextual information to increase the information conveyed by the chart.

After removing any clutter, you may want to add annotations. Use call outs to explain unusual features of your data. A gridline may be useful to emphasize a limit, a goal, or a point where the data values changed significantly.

Ken and Suchi looked at the test coverage chart again. Suchi said, "I can see that testing is picking up speed, but this makes it clear that there are a lot of tests that haven't been run at all. When you combine this chart with the others you've made, it's obvious that we're going to need more time for testing. We did it."

ANNIE BISSETT

The Real Story: We're Not Finished Yet

Ken and Suchi put the finishing touches on their entire presentation just before the big meeting. Ken walked the executive team through his charts, one by one.

Ken used the GQM method once again, this time focusing on the executive team's goals. He walked the executive team through the story, question by question.

Goal: release with fewer known open defects than the previous release.

- Are we fixing more defects than we're finding? Suchi showed Figure 1 with Ken's added "Fixed" line.
- Are we reducing the total number of open defects in the backlog? Suchi showed Figure 5 illustrating that the backlog had grown slightly.

Goal: Find the important defects before the customers do.

- Are we running out of serious defects to find? Ken showed Figure 3, pointing to the uptick in high-priority bugs found in the last two weeks.
- Have we looked for defects everywhere we should look? Ken showed Figure 8 illustrating that they had not yet run all the tests they'd planned.

Ken and Suchi could tell by the look on Jan's face in the executive staff meeting that they'd made their point. Even Jan had to admit that their detailed charts showed a much different view of the project than her superficial ones. She agreed with the executive staff's decision to hold off on releasing for another three weeks. She stopped Ken after the meeting.

"I was sure I knew what was going on with the project," she began. "I guess I owe the two of you an apology. But what I don't understand is how two charts of the same data could paint such a different picture of the health of the project?"

Ken smiled as he responded, "Data doesn't lie, but it sure can tell stories." STQE

Kathy Iberle (www.kiberle.com) is a senior software quality engineer with eighteen years of experience at Hewlett-Packard. She also spent seven years studying chemistry and drawing lots of graphs the old-fashioned way.

Elisabeth Hendrickson (esh@qualitytree.com) is a consultant, author, speaker, and teacher in the field of software quality and testing. You can read more of her thoughts on software quality at www.qualitytree.com.