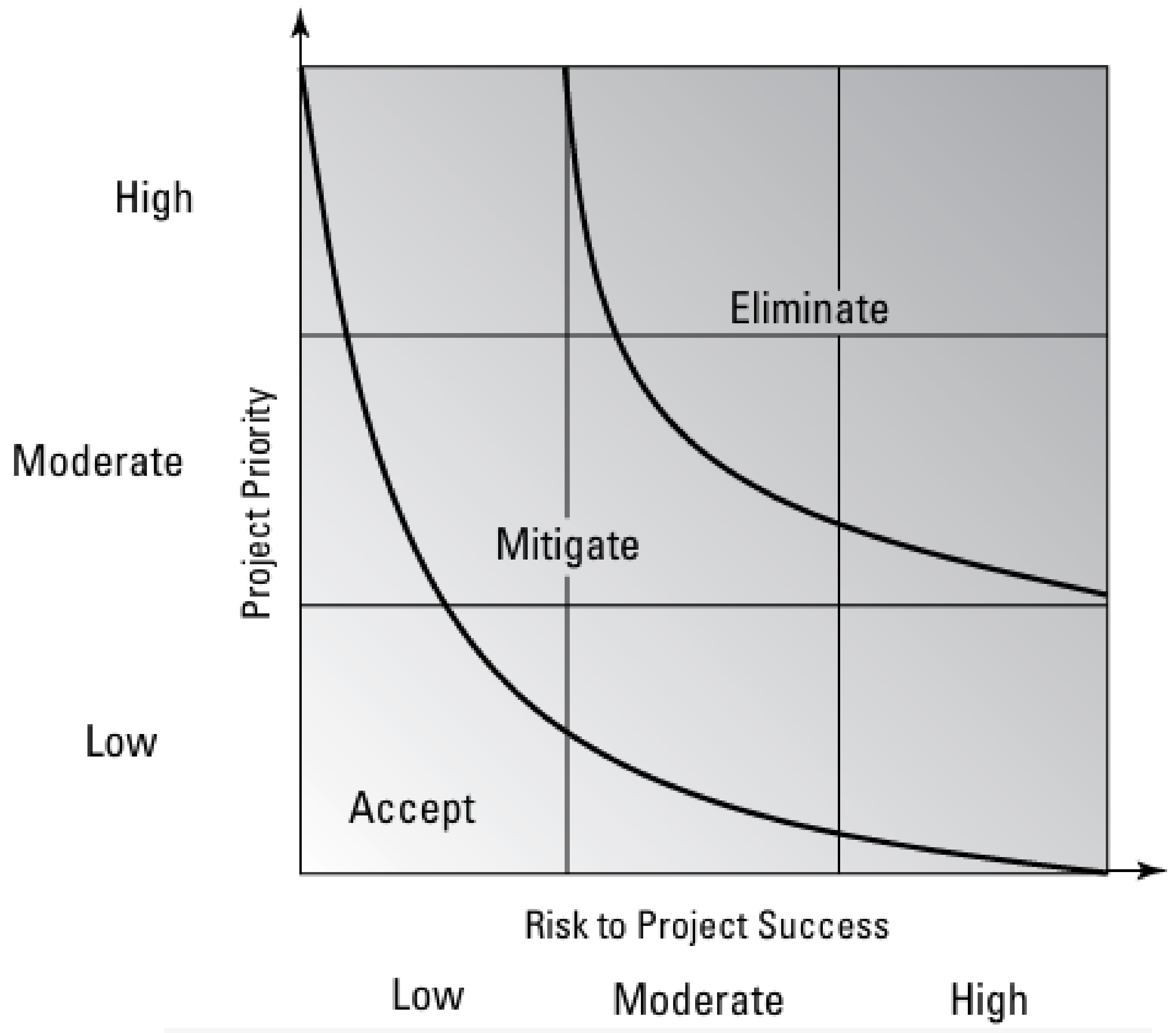
IT PROJECT MANAGEMENT

Muhammad Hamza Ihtisham

Preparing a Risk Response Plan



AVOIDING RISKS

- Changed a project plan to avoid risk interruption
- Used an established approach to software development rather than a newfangled model
- Hired experts to consult the project team during the development process
- Spent additional time with the project stakeholders to clarify all project objectives and requirements

TRANSFERRING RISKS

Transference means that the risk doesn’t go away. It’s just someone else’s responsibility now. You’ve used transference if you’ve ever done any of the following:

- Purchased insurance, such as errors and omissions insurance
- Hired experts to complete a portion of the project work
- Demanded warranties from vendors
- Brought in consultants to test units and builds of your software

MITIGATING RISKS

Risk mitigation is about reducing the impact and/or the probability of risk. Ideally, you'd like to reduce both the impact and the likelihood of a risk occurring, but often when you're mitigating risk you choose to mitigate either one or the other — usually you suck it up and accept the lesser of two evils. You've used mitigation if you've ever done the following:

- Added extra testing, verification, or customer approval activities to ensure that the software conforms to requirements
- Reduced the number of processes, activities, or interactions within a smaller project to streamline and focus project management activities on accomplishing specific project tasks.
- Developed and tested prototypes, used user acceptability testing processes, or launched pilot groups within your organization before releasing the software

ACCEPTING THE RISKS

When you accept certain risks, either the risks are so low that the project can live with them, or the risks are inevitable, but the project must move forward anyway. Sometimes you just know that you can’t prevent an identified risk, you just suck it up, work towards a solution, and deal with it.

EXAMINING RISK RESPONSES AND IMPACTS

The project team and the project manager should determine when the risk response should be implemented. Two terms here to recognize are

- **Risk threshold:** The line of demarcation that signals that a risk is about to come into play and that some response should happen. The risk threshold can be a date for completion, a percentage of the work that is not complete, a failed test, or any other event that signals a pending risk.
- **Risk trigger**: A trigger is an event within the project that triggers a preplanned response to the identified risk

HANDLING THE RIPPLE EFFECT OF RISK RESPONSE

There are two key risks that come from risk responses that should be examined with every risk response:

- **Residual risks:** Residual risks are usually tiny risks that linger after a risk response. These are generally accepted, and the project moves forward.
- **Secondary risks:** Secondary risks are more serious. They occur when a risk response creates significant new project risks.

Planning for Software Quality

Defining quality in software projects

Working with your organization's quality policy

Creating a quality management plan

Identifying how changes in time and cost will affect project quality

Defining Quality

DEFINING QUALITY

- **What customers say:** The software you create lives up to expectations, is reliable, and does some incredible things the customer doesn’t expect (or even think of).
- **What your project team says:** The work is completed as planned and as expected, with few errors — and fewer surprises.
- **What managers say:** The customer is happy, and the project delivers on time and on budget.
- **What you may say:** The project team completes its work according to its estimates, the customer is happy, and management is happy with the final costs and schedule.

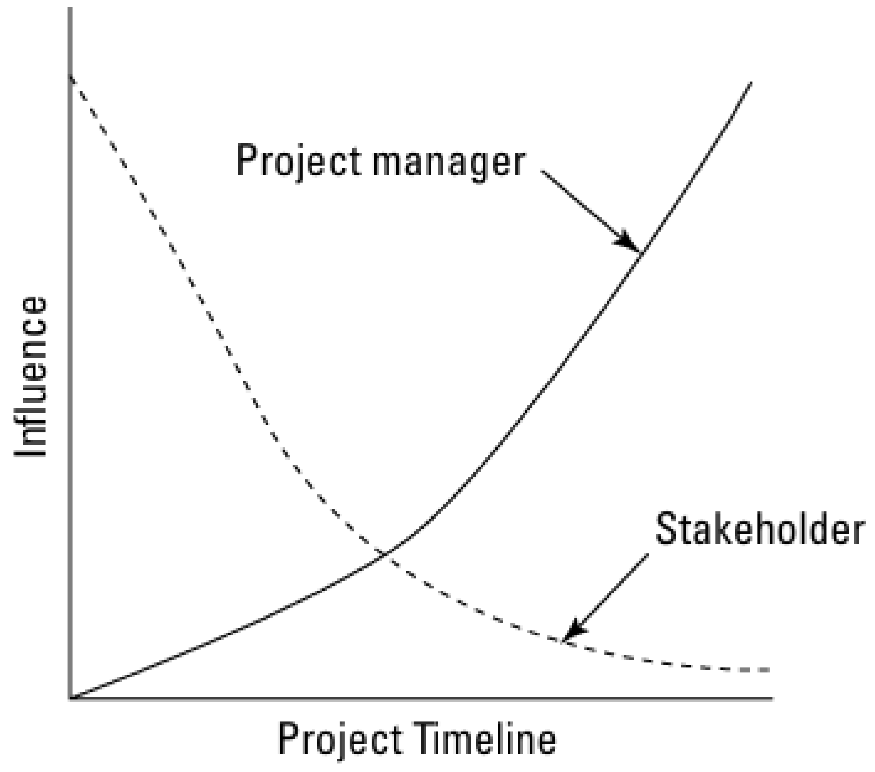
REFERRING TO THE PRODUCT SCOPE

As the project manager, your primary concern is satisfying the product scope. If you work primarily to satisfy the product scope, then you'll be in good shape with satisfying the customer's expectations for quality. But, to satisfy the product scope you must first have several documents:

- **Product scope description document.** This document defines what the customer expects from the project. What are the characteristics of the software? This description becomes more complete as you progress through the project and gather more knowledge.
- **Project requirements document.** This document defines exactly what the project must create without being deemed a failure. What types of functionality should stakeholders be able to perform with the software? This document prioritizes the stakeholders' requirements.
- **Detailed design document.** This document specifies how the project team will create units that meet the project requirements, which in turn will satisfy the product scope.
- **Metrics for acceptability.** Many software projects need metrics for acceptability. These metrics include speeds, data accuracy, and metrics from user acceptability tests. You'll need to avoid vague metrics, such as good and fast. Instead, aim to define accurate numbers and determine how the values will be captured.

REFERRING TO THE PROJECT SCOPE

- The project scope defines all the work (and only the required work) to create the project deliverable. The project scope defines what will and won't be included in the project deliverable. Project scope is different than the product scope, because the product scope describes only the finished deliverable, whereas the project scope describes the work and activities needed to reach the deliverable.
- You must define the project scope so that you can use it as an appropriate quality tool. The project scope draws a line in the sand when it comes to project changes.



AVOIDING GOLD-PLATED SOFTWARE

You create gold-plated software when you complete a project, and the software is ready to go to the customer but suddenly realize that you have money to burn.

- Your customer may be initially happy that you’ve delivered underbudget. Then they’ll wonder whether you cut corners or just didn’t have a clue as to the actual cost of the project.
- The customer may wonder why your estimate and the actual cost of the project deliverable are not in sync.
- The remaining budget will be returned to the customer unless your contract stipulates otherwise.
- Other project managers may not be happy that you’ve created a massive, unused project budget when their projects have been strapped for cash.
- Key stakeholders may lose confidence in your future estimates and believe them to be bloated, padded, or fudged.

WORKING WITH A QUALITY POLICY

WORKING ISO PROGRAMS

- The International Organization for Standardization (ISO) is a worldwide body with 153 members that convenes in Geneva, Switzerland. The goal of the ISO is to set compatibility standards for all industries, to establish common ground, and to maintain interoperability between businesses, countries, and devices.
- There are many different ISO programs, but the most popular is ISO 9000. An ISO 9000-certified organization focuses on business-to-business dealing and striving to ensure customer satisfaction. An ISO 9000-certified organization must ensure that it.
  - Establishes and meets the customer’s quality requirements.
  - Adheres to applicable regulatory requirements.
  - Achieves customer satisfaction throughout the project.
  - Takes internal measures to continually improve performance, not just once

GETTING A TOTAL QUALITY MANAGEMENT WORKOUT

- The U.S. Naval Air Systems Command originated the term Total Quality Management (TQM) as a means of describing the Japanese-style management approach to quality improvement.
- TQM is largely based on W. Edwards Deming’s 14 Points for Quality. Here’s how Deming’s 14 points and TQM are specifically applicable to software development (you can find out more about W. Edwards Deming in the nearby sidebar, “W. Edwards Deming and the soft ware project manager”).

1. **Create constancy of purpose for improving products and services.** Every developer must agree and actively pursue quality in all his or her software creation, testing, and development.
2. **Adopt the new philosophy.** This philosophy can’t be a fad. The software project manager must constantly motivate the project team to work towards quality.
3. **Cease dependence on inspection to achieve quality.** Software development has a tradition of coding and inspection and then reacting to errors. This model is dangerous because developers begin to lean on the testing phase to catch errors, rather than striving to incorporate quality into the development phase. As a rule, quality should be planned into software design, never inspected in.
4. **End the practice of awarding business on price alone; instead, minimize total cost by working with a single supplier.** The idea here is that a relationship will foster a commitment to quality between you and the supplier that’s a bit more substantial than an invoice and a check.
5. **Constantly strive to improve every process for planning, production, and service.** Quality planning and delivery is an iterative process.

6. **Institute training on the job.** If your software developers don’t know how to develop, they’ll certainly create some lousy software. If your team doesn’t know how to do something, you must train them.
7. **Adopt and institute leadership.** The project manager must identify how to lead and motivate the project team, or the team may lead itself, remaining stagnant.
8. **Drive out fear.** Are your software developers afraid of you? If so, how can they approach you with ideas for quality improvements, news on development, and flaws they’ve identified within the software? Fear does nothing to improve quality.
9. **Break down barriers between staff areas** You’ve got to establish relationships, trust, and open communication between the staff areas that interoperate.
10. **Eliminate slogans, exhortations, and targets for the workforce**. Slogans don’t improve quality; they frustrate workers. When you constantly remind people that “Quality is Everyone’s Job!” you underscore the fact that simply talking about quality doesn’t actually improve it.

11. **Eliminate numerical quotas for the workforce and numerical goals for management.** You can tell your developers to churn out 2,000 lines of code a day, and they’ll probably do it. But they won’t guarantee that the code will be any good. A quota is not the same as a demand for quality code.
12. **Remove barriers that rob people of pride of workmanship and eliminate the annual rating or merit system.** Developers should be able to take pride in their work and their accomplishments and be rewarded accordingly.
13. **Institute a vigorous program of education and self-improvement for everyone.** Training, especially in IT, is paramount. Without proper education, how can you expect your team to deliver?
14. **Put everybody in the company to work accomplishing the transformation.** For Deming’s approach to work, everyone must participate. A few folks here and there won’t make much of an impact in most organizations

PDCA

Plan. You and your project team plan the project work, anticipate changes by analyzing strengths, weaknesses, opportunities, and threats (SWOT), and then predict the results for the different facets of your project. Projects fail at the beginning, not the end.

Do. With your plan established, you can execute the work. The work, however, should be completed in small steps in a controlled environment.

Study. This is the quality control portion of the circle. You, the project team, and the subject matter experts must check the work for accuracy.

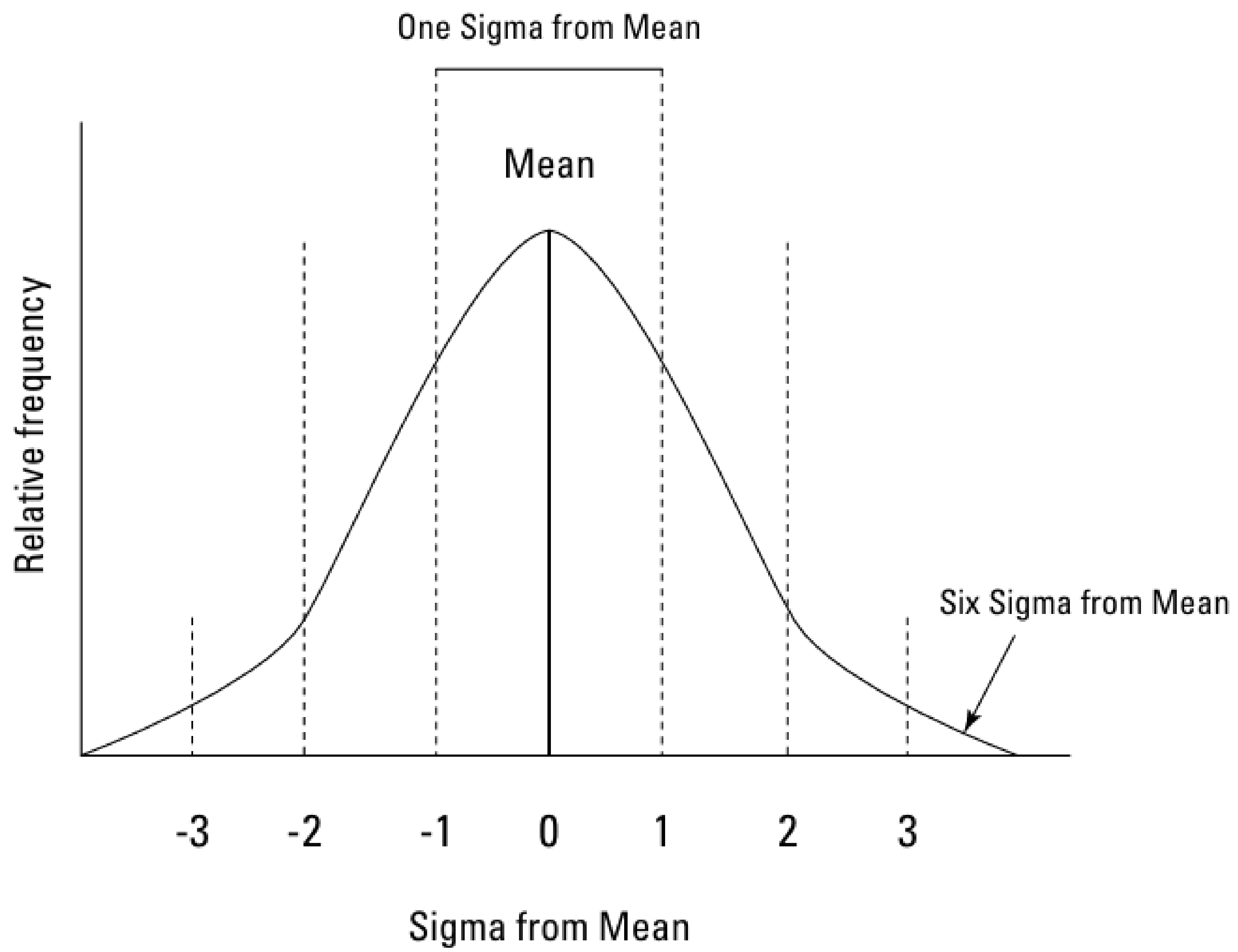
Act. Act to improve the process or standardize your approach in project development. When you’re done, the process starts over.

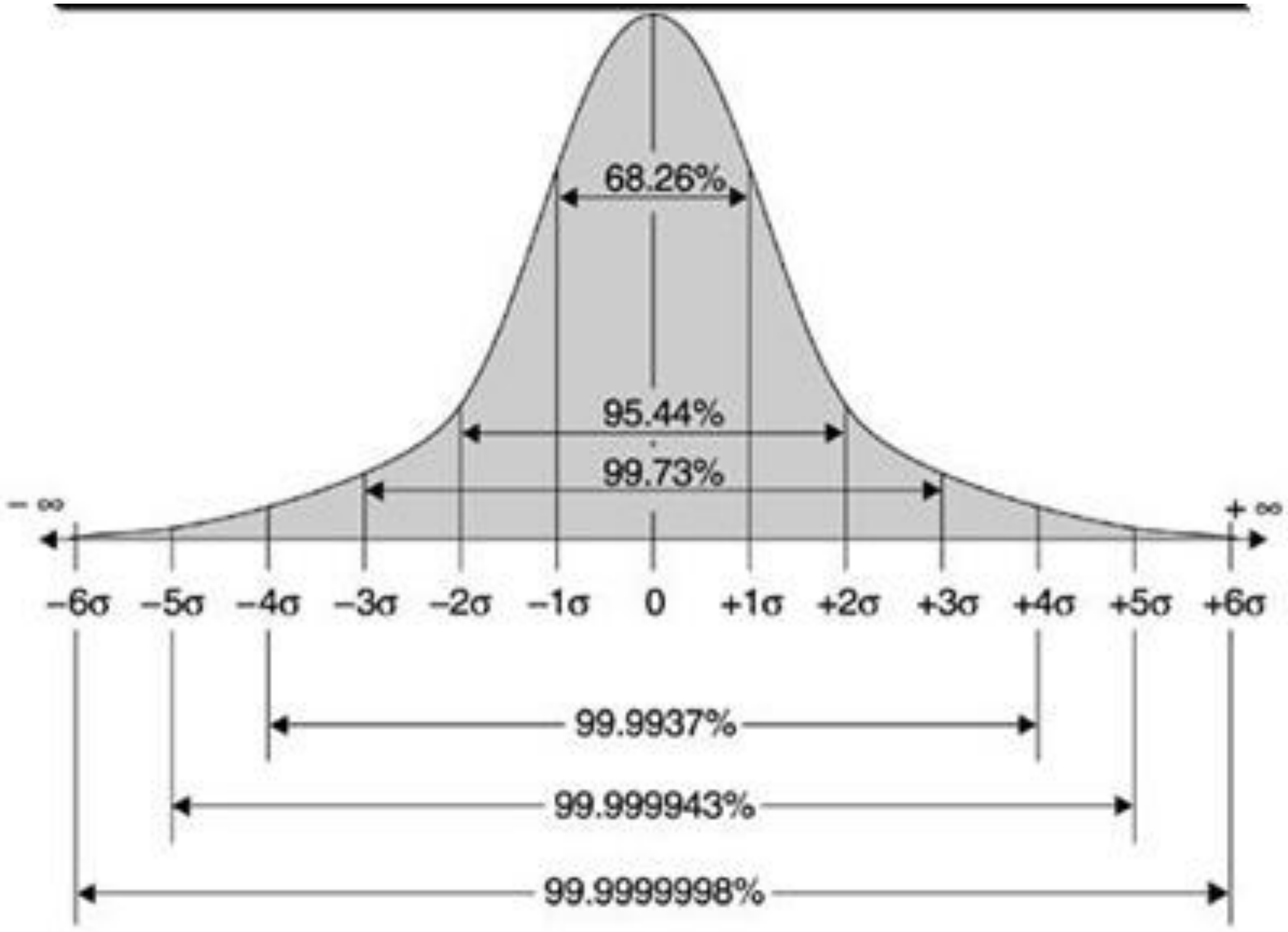
SLIPPING INTO THE SIXTH SIGMA

- Six Sigma is a procedure that strives to reduce waste, errors, and constantly improve quality through the services and deliverables an organization produces. Six Sigma was developed by some really smart people at Motorola who received the Malcolm National Quality Award in 1988 for their Six Sigma methodology.
- Most software is created and tested, then the errors are fixed, patched, or ignored, and then the entire process starts over. Software development, for the most part, focuses on inspection to ensure quality; this is quality control. Six Sigma, however, focuses on preventing the mistakes from entering the process at all; this is quality assurance.
- According to ASQ, most organizations perform at three to four sigma, where they drop anywhere between 20 and 30 percent of their revenue due to a lack of quality. If a company can perform at Six Sigma, it only allows 3.4 defects per million opportunities

SLIPPING INTO THE SIXTH SIGMA

- **We don’t know what we don’t know.** A lack of knowledge keeps organizations trapped in their current environment, losing revenue and preventing progress.
- **We don’t do what we don’t know.** If you don’t know what you should be doing you cannot do it.
- **We won’t know until we measure.** The real action in Six Sigma is to measure to improve.
- **We don’t measure what we don’t value.** Six Sigma looks at what does and does not need to be measured, and then prompts the developer or project manager to act accordingly. If you value your programmers’ time, your software’s errors, and your customer satisfaction, you’ll measure them all.
- **We don’t value what we don’t measure.** This is a call to action! What should you be measuring that you’re not?


One Sigma from Mean
Mean
Relative frequency
Six Sigma from Mean
-3
-2
-1
0
1
2
3
Sigma from Mean


68.26%
95.44%
99.73%
− ∞
+ ∞
−6σ
−5σ
−4σ
−3σ
−2σ
−1σ
0
+1σ
+2σ
+3σ
+4σ
+5σ
+6σ
99.9937%
99.999943%
99.9999998%