The Cost of Poor Quality Software in the US: A 2018 Report

Herb Krasner
Member, Advisory Board
Consortium for IT Software Quality (CISQ)
www.it-cisq.org
Hkrasner@utexas.edu

Date: September 26, 2018

CISQ

Consortium for IT Software Quality

Contents

1. Forward

The following abbreviations are used widely throughout this report. Basic definitions are provided here with more detailed definitions of each term found in the body of the report.

Abbreviation	Meaning
IT	Information Technology
US	United States of America
CoSQ	Cost of Software Quality
CPSQ	Cost of Poor Software Quality
CGSQ	Cost of Good Software Quality
LOC	Lines of Code (source)
CPDQ	Cost of Poor Data Quality

This report aggregates publicly available source material of the cost of poor software quality in the US today. The report describes how to stimulate software quality improvement programs widely across industry and government.

Our conclusions understand that most IT and software organizations do not now collect Cost of Software Quality (CoSQ) data. Without a defined CoSQ model, most IT leaders lack a basis for estimating the answers to these two pertinent questions:

1. What is the cost of poor-quality software in our organization?
2. How do our investments in software quality affect our overall costs of quality and cost of ownership for software assets?

Previous published studies[1,2,3,4,] have highlighted various aspects of poor-quality software. These studies are lacking because they fail to account for the total cost of poor-quality software across the entire US software industry.

This study performs a systematic review of the available public sources on the topic of the cost of poor-quality software in the US today. A systematic review, critical assessment and evaluation of all found data sources provide a method of locating, assembling, and evaluating the body of public sources. This study takes a comprehensive view of approximating the total cost of poor software quality in the USA today.

2. Executive Summary

This project performs a comprehensive research study, evaluating the cost of software—specifically poor-quality software—on the US economy as a whole. Existing sources of public data were used in this report with all sources cited.

This report fills a gaping hole in our understanding of the financial implications of poor-quality software effecting society today and into the future. This report is primarily for C-suite executives, CTOs, CIO's and other IT professionals who are interested in quantifying their costs of poor-quality software.

The report body describes the primary motivations for doing this study, including software's critical importance to modern society and illuminating the fundamental issues causing problems. The iceberg model is used to show which software quality costs are usually hidden from sight.

Next the landscape of software quality problem areas are described by 1) looking backwards in time, 2) forward into the future, and 3) identifying current issues facing us. The issues described include:

1. Legacy systems that hold our personnel and budgets captive;
2. Technical innovations that attempt to move us forward at accelerating rates;
3. Today's highly vulnerable "Systems of Systems";
4. Today's era of 9-digit software systems' failures and defects; and
5. The growing burden of technical debt.

Once the landscape is defined, the labor force impacts are addressed by covering the following topics:

1. Defining the Information Technology Workforce;
2. Computer and Information Technology Occupations in the US Today (BLS);
3. Impact of the IT gig economy; and
4. Implications for quality and costs.

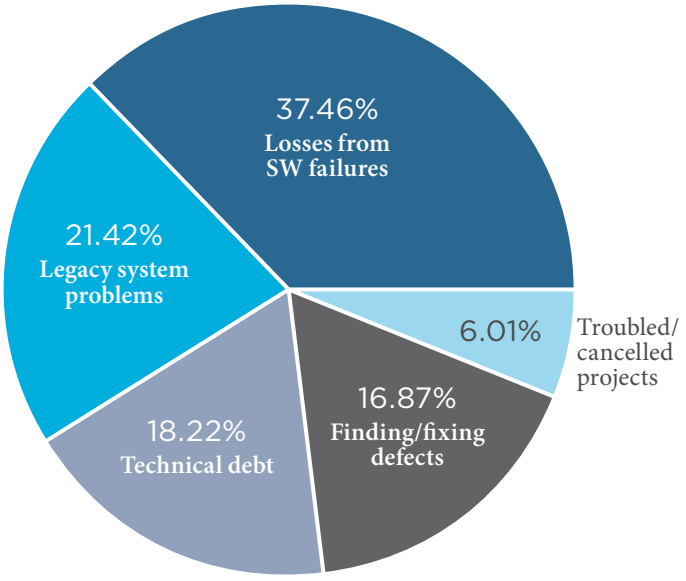
Formal definitions of software quality and the cost of software quality model are introduced by defining:

1. Software Quality;
2. Good versus Poor-quality Software;
3. The cost of software quality model and its evolution;
4. Categories of Cost of Poor Software Quality (CPSQ); and
5. Categories of Cost of Good Software Quality (CGSQ).

The Cost of Software Quality (CoSQ) model identifies the component costs of quality and how those add up to form a notional total. A summary of cost categories for poor-quality software and data, and what these numbers are telling us in order to improve the situation are summarized. The management actions necessary to attack the problems and make a significant improvement in various organizational situations conclude the report.

In summary, the cost of poor quality software in the US in 2018 is approximately **$2.84 trillion**, the main components of which are seen in the following graph. If we remove the future cost of technical debt, the total becomes $2.26 trillion. For simplification, the various cost categories are, at this time, assumed to be mutually exclusive. Clearly a deeper level of intersection analysis is warranted. We therefore view this amount as a potential upper bound. It was our intention to use this result as a starting point for community discussion and future in-depth benchmarking studies.

FIGURE 1: AREAS OF COST RELATING TO POOR IT/SOFTWARE QUALITY IN THE US



The methods for arriving at these category amounts and total is presented in the sections of the body of this report.

General recommendations for improvement depend on each organization's unique situational context. These recommendations include:

1. Find and fix problems and deficiencies as close to the source as possible, or better yet, prevent them from happening in the first place. This is in line with industry movements such as early work product appraisals and continuous testing.
2. Measure the CPSQ. With these numbers in hand, you have the basis for a business case to invest smartly in software quality improvement.
3. Attack the problem by focusing on the different results of good vs. poor software quality in your shop and relevant benchmark organizations.
4. Economic target areas will likely include: cost of ownership, profitability, human performance impact, enabling innovation, and effectiveness of mission critical IT systems.

3. Introduction

IN THIS SECTION:

- The importance of software and its quality in the world today
- Spending on IT software in the world today
- Illuminating a fundamental but unseen problem in IT systems
- Introducing the cost of quality approach adapted to software
- *The Iceberg Model* of hidden software quality costs

"Software is eating the world" —Marc Andreeson, August 20, 2011, Wall Street Journal

The primary motivations for this report are to:

- Define the crucial importance of software and its quality in modern society
- Identify the limitations of previous studies of some of the costs of poor-quality software
- Illuminate the fundamental issues that are causing quality problems with our IT and software-enabled systems

According to Wikipedia, ***software*** is a generic term that refers to a collection of data and/or computer instructions that tell a computing device how to work. A computing device includes any programmable chip, chip set, or collection of such devices. This includes both general purpose and special purpose computing devices, and all types of software that run on them (e.g. everything from firmware to business enterprise software to cloud services to embedded software, etc.).

Software in modern society is ubiquitous. Consider your smart phone. It's a mobile computing device with millions of lines of computer code in it. For example, the average iPhone app has around 10-50 thousand lines of code, while Google's entire code base is two billion lines of code for all its services. The smart phone Operating System (e.g. Android) alone has roughly 12 million lines of code.

A ***line of code (LOC)*** is a single discrete instruction to the computing device, e.g., to perform an operation, or declare a data element, in whatever language that is used. For ease of counting purposes sometimes a source LOC (SLOC) is simply defined as a line of text in the program's source listing. There are many more specific ways of defining a LOC.

By 2021, there will be almost 36 billion Internet-connected devices[16] and over 54%[3] of the world's population will be Internet users; and global internet traffic will reach 3.3 zettabytes[15]. A zettabyte is equal to one sextillion (10^{21}) or, strictly, 2^{70} bytes.

In the USA, when asked "to what extent do information and communication technologies (ICTs) enable access for all individuals to basic services (e.g., health, education, financial services, etc.)?" [1 = not at all; 7 = to a great extent]; the USA scores 5.7 on the 7-point scale. The leading countries score 6.2. This rapidly changes as organizations adapt to a "Bring Your Own Device" BYOD) approach. Your interface to software-enabled services in many cases is now your smart phone.

The most recent survey of computer ownership was conducted in 2012. It revealed that in the three decades since the first survey, the percentage of homes with a computer increased almost tenfold, to nearly 80%. Moreover, a poll conducted in February 2018 by the Pew Research Center found that 77% of all Americans, and 94% of all Americans aged 18-29, own a smartphone. Worldwide smart phone usage is predicted to be 2.53 billion this year.

Computing devices and software are the main tools that enable our personal lives, our society, industry and government; therefore, software quality and software security are among the most important topics of this decade. The importance of both quality and security will increase over the next decade. Computing technology is integral to all today's activities. Software quality matters.

How much are we spending on IT software in the world today?

According to IDC[9], global information technology spending will top $4.8 trillion in 2018, with the US accounting for approximately $1.5 trillion of that market. They state that the United States is the largest technology market in the world, representing 31% of the global total. In the US according to CompTIA, the IT sector is poised for another strong year, 5.0% growth projected. The optimistic upside forecast is in the 7.2% range, with a downside floor of 2.8%.

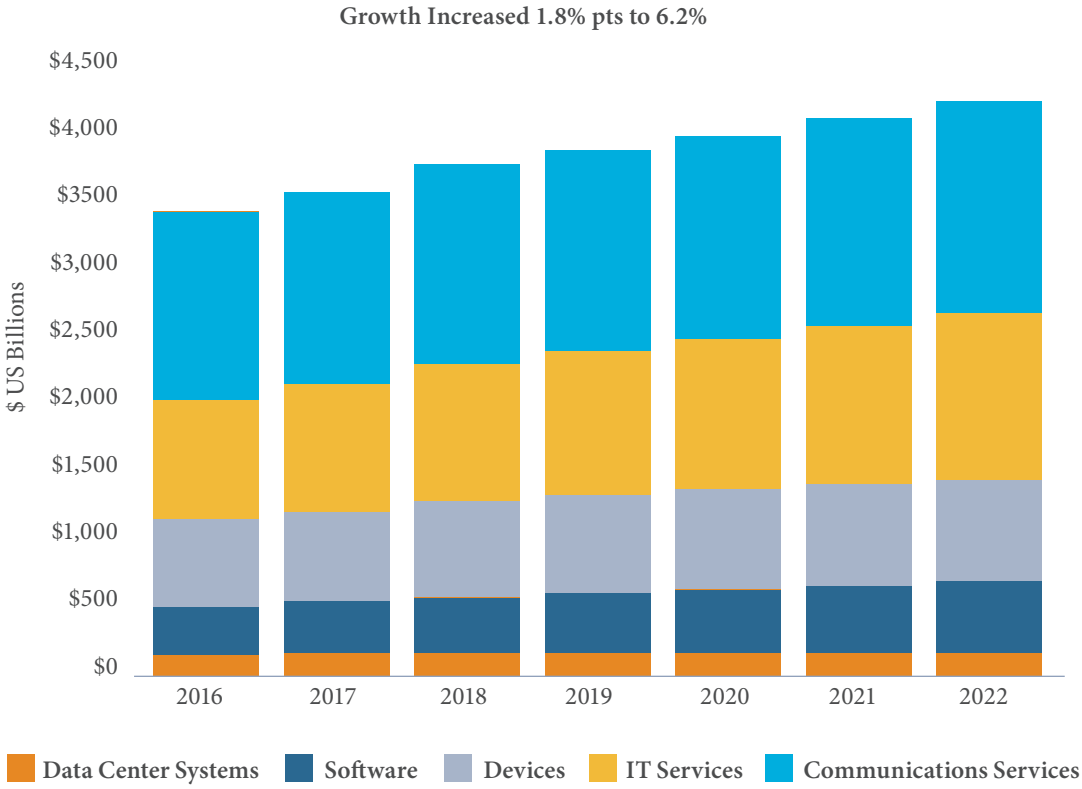
For the most recent year of available data, US exports of tech products and services were an estimated at $309 billion in 2016. Exports account for approximately $1 out of every $4 generated in the US tech industry. Forecasts of IT growth from various sources include:

- Gartner 4.5% worldwide forecast
- IDC 5.3% worldwide forecast
- Forrester 5.8% US forecast
- CompTIA 5.0% worldwide forecast

According to Gartner[5], about $3.7 trillion dollars will be spent worldwide on IT enterprise systems in 2018; an increase of 6.2% over last year. Their study, focusing on purchased products and services, covers the cost of: Data Center Systems, Enterprise Software, Computing Devices, IT Services and Communications Services. Enterprise software spending is forecast to experience the highest growth in 2018 with an 11.1 percent increase, at about $400 billion. Application software spending is expected to continue to rise through 2019, and infrastructure software will also continue to grow, bolstered by modernization initiatives. When considering the additional expenses of labor and support costs, it is asserted by Apptio[7] that global IT spending is actually closer to **$6.3 trillion**—because in most companies, the cost of labor accounts for close to 45% of their IT spending. It would appear that this number covers IT products, services and labor; but probably does not cover things like embedded systems, IoT, lost market share, stock declines, legal costs, etc.—and other costs associated with problematic IT systems and services.

The US share of that $6.3 T would be **$1.953 trillion, or approximately 9.56% of US GDP ($20.4 T)**. It would not be unreasonable to suggest that US total IT spend in these products, services and labor, is about 10% of GDP. The following graph is freely available on the Gartner web site[5].

FIGURE 2: GARTNER'S FORCAST FOR 2018 WORLDWIDE DOLLAR-VALUED IT SPENDING



Accepting Gartner's report with Apptio's enhancement, a more correct representation of total worldwide IT spending is $6.3 trillion in 2018. The US share of that would be $1.95 trillion. Adding in potential missing categories described above, the US IT spending amount for products, services and labor is probably at least **$2 trillion** or about 10% of GDP. US GDP in 2018 is $20.4 trillion, or about 23.3% of the world economy.

Software quality is important—just about every C-suite executive now knows that. But recognizing that concept in the abstract is one thing, while actually investing time and resources toward procuring, developing, releasing and/or evolving high-quality software is quite another. The fact of the matter is that, many executives don't ultimately make software quality a top-level priority goal. This can be a serious mistake. Just ask Equifax!

The reality of the situation is that there are serious costs associated with poor-quality software. It's not just a question of undermining a company/organization's reputation—although that has its own costs—it's also a matter that's directly reflected in the bottom line.

According to Curtis[13], today's software applications have now entered the era of 9-digit (>= $100M) software failures. Not only are these huge figures, but they represent money that is—in a very real way—purely wasted.

Illuminating a fundamental but unseen problem in IT systems

As a purely intellectual product, software is among the most labor-intensive, complex, and error prone technologies in human history. Software is so pervasive in modern society, that we are often unaware of its presence until problems arise. Even though many successful software products and systems exist in the world today, an overall lack of attention to quality has also led to many problematic systems that don't work right, as well as to many software projects that are late, over budget, or cancelled.

What constitutes good quality in software is generally taken to mean that a software product (or system) provides value (e.g. satisfaction) to its users/stakeholders, makes a profit (if that is a goal), generates few serious complaints/problems, and contributes in some way to the goals of humanity (or at least doesn't do harm). Poor-quality is therefore the opposite of that. Another popular definition of the cost of poor software quality (CPSQ) is those costs that would disappear if IT systems, related processes, and included products/components were perfect. More specific definitions are discussed later in this paper.

When these simple questions are routinely asked at the C-Suite level, amazing organizational transformations are possible.

If improving business success through IT software quality is an organizational goal, then answers to a these little asked questions must be derived:

- How much is software costing us, and what are its benefits?
- How much is poor software quality costing us?
- How good is our software?

When these simple questions are routinely asked at the C-Suite level, amazing organizational transformations are possible.

The cost of quality approach adapted to IT software

As a product, software is different from any kind of manufactured object. Some obvious differences are:

- It has high fixed costs and somewhat lower variable costs.
- It doesn't wear out but does require maintenance.
- Additional value can more easily be added in the future (i.e. vs. hardware).
- It is inherently more complex
- It is more intangible and less visible because it is non-physical

CoQ is a proven technique in manufacturing and service industries, both for communicating the value of quality initiatives and for identifying quality initiative candidates. CoSQ offers the same promise for the software industry but could be used more than currently.

Initial uses of CoSQ indicate that it represents a very large percentage of development costs—60 percent and higher for organizations that are unaware of improvement opportunities[14]. CoSQ use demonstrates significant cost savings for software organizations willing to undertake quality improvement initiatives. Perhaps more importantly, the use of CoSQ enables an understanding of the economic tradeoffs that accompany activities and expenditures made for improving the quality of delivered software.

The Iceberg Model

Many of the costs of poor IT software quality are hidden and difficult to identify with formal measurement systems. The iceberg model (figure below) is very often used to illustrate this concept: Only a minority of the costs of poor software quality are obvious—appearing above the surface of the waterline. But there is a huge potential for reducing costs under the waterline. Identifying and improving these costs will significantly reduce the costs of operating a business/organization.

FIGURE 3: THE ICEBERG MODEL

4. The Landscape: Looking Backwards, Forwards and at Present

IN THIS SECTION:

- Legacy systems that hold our personnel and budgets captive
- Technical innovations that attempt to move us forward at accelerating rates
- Today's highly vulnerable "Systems of Systems"
- Today's era of 9-digit software systems' failures and defects
- The growing impact of technical debt

Looking backwards: Legacy systems hold us captive

In 2002, NIST reported that estimates of the economic costs of faulty software in the US range in the tens of billions of dollars per year and have been estimated to represent approximately just under 1 percent of the nation's gross domestic product (GDP). How has that changed in the 16 years since?

In most companies and organizations, the Operation & Maintenance (O&M) of existing IT systems consumes the majority of the IT budget, roughly 75% of the total IT spend per year. For a particular system, software maintenance costs[23] will typically form 75-80% of the Total Cost of Ownership (TCO). In either case this leaves only about 25% for the development of new capabilities, products and systems.

Respondents to a 2013 Forrester Research survey of IT leaders at more than 3,700 companies estimated they spent an average of 72% of their budgets on just keeping-the-lights-on functions. In 2016 the US Government Accountability Office found that 5,233 of the government's almost 7,000 IT projects systems were spending "all of their funds on operations and maintenance".

Legacy IT systems reflect an organization's past and present; they mirror both the complexity of the world they were developed for and that they currently operate in. If you peel away a system's layers you see code and data flows that reflect rules governing the organization—some nuanced, some long forgotten—which determine how the software should process information. As the organization changes, new code is layered over existing code. Embedded systems, starting with military airplanes, ships, motor vehicles, railway signaling, telecommunications, the electricity grid, gas/oil analysis and even traffic lights, contain more software. Legacy systems become unwieldy due to aging, varying by particular type of system.

One reason is the technology itself. The result of different departmental approaches, and inadequate IT strategy and governance leads to an assortment of diverse mainframes, servers, databases, computer languages and packages from multiple vendors. The resulting fragmented architecture—with thousands of interlinked subsystems—becomes costly to maintain and, as it ages, fewer people know how to work on it.

Legacy systems can do real damage to a company or organization:

- Legacy IT strategies aren't prepared for continuous change
- Legacy systems make security worse, not better
- Meeting customers on their terms becomes impossible
- Legacy systems are not cost effective to manage
- Compatibility issues threaten business interactions
- It's unhealthy for employee training
- Proprietary and archaic technology leads to personnel fatigue

Determining the cost of poor-quality software in legacy systems requires a deeper look at what activities are actually consuming the most effort during the O&M phase of an IT systems extended lifetime. Software maintenance costs[41] include the following basic categories:

- **Corrective maintenance:** costs due to modifying software to correct issues discovered after initial deployment (generally 20% of software maintenance costs)
- **Adaptive maintenance:** costs due to modifying a software solution to allow it to remain effective in a changing business environment (generally up to 50% of software maintenance costs)
- **Perfective maintenance:** costs due to improving or enhancing a software solution to improve overall performance or maintainability (generally 25% of software maintenance costs)
- **Preventive maintenance:** costs due to modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults (generally 5% of software maintenance costs).

According to Curtis[24], correcting defects frequently accounts for as much as one third of all post-release O&M work, and time spent understanding the code has been shown to account for as much as half of all the effort expended by maintenance staff. When the overlap between these two activities is removed, as much as two thirds of all maintenance effort can be classified as waste.

The factors above can be used to provide an estimate of the overall cost of poor-quality legacy software in O&M in the USA today. If 75% of all IT dollars are being spent on O&M, and if as much as 2/3 of that could be classified as "waste", that gives us an approximate upper bound of $980 billion on the cost of poor-quality software in O&M from a maintenance perspective. Waste is a lean term that means all non-value-added activities. This waste does not include those additional costs incurred outside of the IT organization. The lower bound using only corrective maintenance in the calculation would be $290 billion. The mid-point between the upper and lower bound would be $635 billion.

The approaches for attacking this part of the problem will be different than the approaches needed for new IT systems acquisition and development.

Looking forward: Tech innovations coming faster and faster

The Fourth Industrial Revolution, representing a transition to a new set of systems, bringing together digital, biological, and physical technologies in new and powerful combinations, is upon us. The term 'Fourth Industrial Revolution' was first used in 2016 at the World Economic Forum. New systems are being built on the infrastructure of the digital revolution (3rd). Just as the digital revolution was built on the heart of the second industrial revolution—electricity, mass communication systems, and modern manufacturing—the new systems that mark the Fourth Industrial Revolution are being built on the infrastructure of the third, digital revolution—the availability of global, digital communications; low-cost processing and high-density data storage; and an increasingly connected population of active users of digital technologies.

The Fourth Industrial Revolution represents new ways in which technology becomes embedded within societies and even the human body. It is marked by emerging technology breakthroughs in a number of fields, including:

- Robotics
- Nanotechnology
- Quantum computing
- Artificial Intelligence (AI)/Machine Learning (ML)
- biotechnology
- blockchain/cryptocurrencies
- location-based platforms
- Internet of Things (IoT)
- virtual/augmented/mixed reality
- e-learning
- BYOD (Bring Your Own Device)
- mobile computing
- wearables/implantables
- e-payment systems
- autonomous vehicles
- digital security technologies (especially multilayer authentication)

These technologies, directly enabled by new computer software, challenge the systems of the past. They have great potential to connect billions of more objects/people to the Internet, drastically improve the efficiency of business and organizations and help regenerate the natural environment through better asset management. They hold unique opportunities to improve human communication and conflict resolution, while at the same time potentially causing large disruptions in our modern societies, especially when they fail massively. E.g., what happens when a self-driving auto kills a pedestrian without stopping? Is that a software flaw? Very likely, YES—probably one of omission—and it's already happened.

The inherent characteristics of these new software systems will have increased: complexity, conformity, changeability and connectedness; requiring us to take a fresh look at how poor-quality software impacts future scenarios of developing and deploying these new technologies. These new technologies are primarily in the research and development (R&D) stage of their lifecycle. Software quality problems frequently occur when research prototypes are prematurely thrust into a product development stream.

Looking at today: Highly vulnerable and deficient systems of systems

On average, software developers make 100 to 150 errors for every thousand lines of code[23]. Of course, this number varies from developer to developer and project to project. Even if only a small fraction—say 10 percent—of these errors are serious, then a relatively small application of 20,000 lines of code will have roughly 200 serious coding errors. Not to place the blame solely on software developers, the Meta Group reports that up to 80 percent of the issues leading to customer dissatisfaction can be traced to poor understanding of requirements. Poor architecture causes a wide array of quality problems including fragility, lack of scalability, and resistance to modification. In summary, the whole software development process is fraught with opportunities to introduce problems and deficiencies.

The main culprits in most problematic IT systems of today are sheer size and complexity. For example[43], the Google codebase includes approximately one billion files and has a history of approximately 35 million commits spanning Google's entire 18-year existence. The repository contains 86 terabytes of data, including approximately two billion lines of code in nine million unique source files. In terms of the largest single product, that's probably Microsoft Windows at 500 million LOC. As of 2017, Microsoft announced what they believe is the world's largest Git repository[44]:

- approximately 3.5M files that
- result in a Git repository of about 300 gigabytes in size
- with 4,000 engineers producing 1,760 daily "lab builds" across 440 branches, plus thousands of pull request validation builds.

Even your smart phone has millions of LOC in it. Not to mention your new automobile with dozens of systems embedded, all interacting with each other.

The era of 9-digit failures and defects[47]

Software nonperformance and failures are expensive. The media is full of reports of the catastrophic impact of software failures. In a recent report[46], software-testing company Tricentis analyzed 606 software fails from 314 companies to better understand the business and financial impact of software failures. The report revealed that these software failures affected 3.6 billion people and caused $1.7 trillion in financial losses and a cumulative total of 268 years of downtime. Software bugs were the most common reason behind these failures. Their stories come from English language news outlets. Their report did not allocate failures to specific countries, but in terms of relative GDP, the US dominates the group of English language speakers (75% of the total). We therefore assume that 75% of these failure totals are attributable to the US. So, the US total for software failures in the news is $1.275 trillion.

And, what about all those failure stories that don't make the news?

According to Curtis[46], when losses from IT malfunctions hit 5 or 6 digits, IT managers are at risk. When losses hit 7 or 8 digits, IT and line-of-business executives are at risk. When losses hit 9 digits, C-Level jobs are at risk. Most often these 9-digit fiascos result from software flaws inside a system. Three trends magnify the impact of software malfunctions, driving business liabilities toward 9 digits.

- With increasing digital transformation, a far greater slice of business operations from sales to delivery is integrated and controlled by software, thus rapidly spreading the effects of a malfunction across the value chain.
- Businesses are now enabled by systems of systems, expanding complexity exponentially and concealing the triggers for 9-digit losses in a thicket of cross-system interactions.
- Increased competition, especially online, has prioritized speed-to-business over operational risk and corrective maintenance costs, a huge gamble for systems not designed to expect and manage failures.

...better architectural and coding practices can implement the internal system safeguards that limit the damage from potentially devastating defects...

If the trend toward multimillion-dollar defects, some reaching 9 digits continues, or even accelerates, the status quo in IT will change, and not from inside the IT community. If the tipping point for greater regulation and centralized control has not been passed, avoiding it will require greater adherence to software best practices that move software development toward a real engineering discipline. For example, better architectural and coding practices can implement the internal system safeguards that limit the damage from potentially devastating defects long before they spiral toward to 9 digits and beyond.

The table on the next page lists the top 2018 IT failures in the news so far, representing just the tip of the cost of poor-quality software iceberg.

TABLE 1: TOP 2018 IT FAILURES IN THE NEWS

Wells Fargo Bank	In early August 2018, Wells Fargo admitted that as many as 400 homeowners were accidentally foreclosed upon after a "calculation error" in their accounting software denied them a mortgage loan modification. In their latest SEC filing, the bank promised to continue to assess any customer harm and provide remediation as appropriate. To that end, they have set aside $8 million for affected customers.
PSA Airlines	In June 2018, at PSA Airlines, issues with a crew-scheduling program caused thousands of flights to be cancelled for days last week. The computer problem was tied to the crew scheduling and tracking system at PSA Airlines, a wholly owned subsidiary that operates flights under the American Eagle brand. Those flights carry passengers to and from regional airports to major hubs like Charlotte, North Carolina. This was a significant IT systems issue that caused both PSA's main systems and backup systems to slow down beyond a usable state. During the outage, American Airlines cancelled about 3,000 flights, with 2,500 of those to and from the Charlotte airport. Those cancellations stranded passengers in Charlotte and elsewhere, while drawing widespread ire from travelers, including on social media. The airline industry has been particularly hard hit with numerous IT failures this past year.
Uber Technologies, Inc.	In March 2018, a self-driving Uber SUV struck and killed a pedestrian in suburban Phoenix, Arizona, in the first death involving a fully autonomous test vehicle. Uber determined the likely cause of the fatal collision was a problem with the software that decides how the car should react to objects it detects. The car's sensors apparently detected the pedestrian, but the software decided it did not need to react right away. Uber executives believe the system was tuned so it would be less responsive to objects in its path, such as plastics bags. How much this will eventually cost Uber internally and externally is yet to be determined. Another crash on March 23, 2018 of a Tesla Model X in Mountain View, California caused that company's stock price to drop 3.3% the following day.
TSB Bank	Millions of TSB customers were locked out of their accounts after an IT upgrade led to an online banking outage. A planned system upgrade was expected to shut internet and mobile banking services down for one weekend in April 2018 but ended up causing weeks of disruption. The problems arose from TSB's move to a new banking platform following its split from Lloyds Banking Group. Immediately after the new system was switched on, many customers experienced problems logging in, while others were shown details from other people's accounts or inaccurate credits and debits on their own. Customers remained locked out of their accounts two weeks after the initial outage. TSB said it was handling their complaints on a case-by-case basis.
Welsh NHS IT failure	Doctors and hospital staff of the Wales NHS experienced a widespread computer failure that led to them being unable to access patient files. According to the National Cyber Security Centre, the failure was due to technical issues as opposed to a cyber-attack, yet it still caused wide disruption as GPs were unable to access blood and X-Ray results. It also caused a backlog as patients could not be contacted to cancel appointments, and notes could not be typed up and saved on NHS systems.

Meltdown & Spectre	At the start of 2018, Google researchers revealed CPU hardware vulnerabilities called Meltdown and Spectre, which affected almost all computers on the market. Meltdown primarily affects Intel processors, while Spectre affects Intel, AMD and ARM processors. Although these are both primarily hardware vulnerabilities, they communicate with the operating system to access locations in its memory space. Meltdown breaks the most fundamental isolation between user applications and the operating system. This allows a program to access the memory, and also the secrets, of other programs and the operating system. Spectre meanwhile breaks the isolation between different applications—it allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. New Spectre flaws are still being discovered.
Hawaii Sends Out State-Wide False Alarm About a Missile Strike	On January 13, 2018 the citizens of Hawaii were notified to take immediate cover in the face of an inbound ballistic missile strike. It turned out to be a false alarm, although it took over 30 minutes (and, presumably, several thousand heart attacks) before the alert was retracted. Investigations found that while the problem was largely due to human error, there were "troubling" design flaws in the Hawaii Emergency Management Agency's alert origination software.
US CBP	For a second year in succession, the US Customs and Border Protection (CBP) computer systems experienced an outage that left thousands of passengers across the United States waiting in long lines to clear customs. This time, the outage was only for about two hours, while last year's incident lasted four hours and affected more than 13,000 passengers on 109 flights, according to a Department of Homeland Security Inspector General report. The 2017 New Year's problem was caused by an inadequately tested software change related to CBP's long-running IT modernization effort. Another report indicated that the main CPB computer system used to screen international passengers has seen its performance "greatly diminished over the past year as a result of ongoing efforts to modernize (its) underlying system architecture." Before this latest outage, there were three other service disruptions in 2017, according to the DHS IG report.

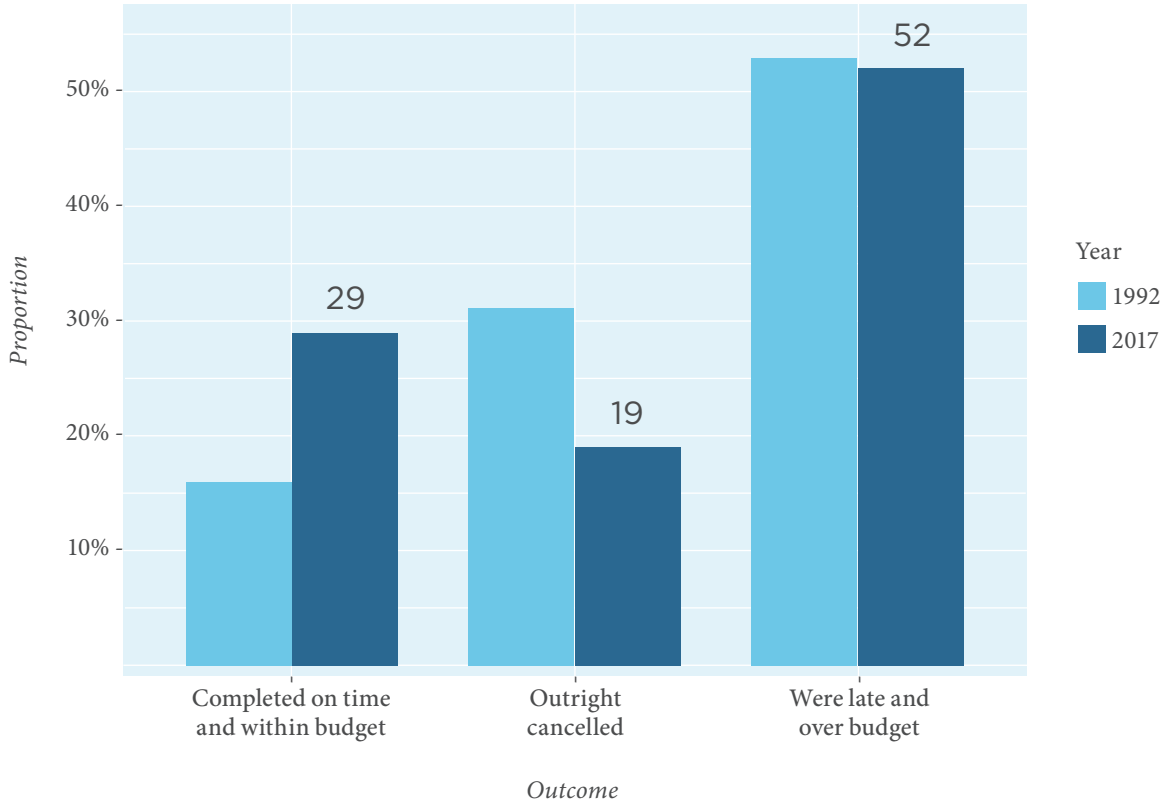
Troubled/challenged projects[30]

There are also thousands of troubled projects within organizations that rarely make the news.

Looking at 25 years of historical projects in the Standish database, in 2017 they reported the following: that only 29% were fully successful with respect to time and budget. Their data says nothing about the quality of the result; presumably those had successful outcomes. The Standish Group research shows a staggering 19% of projects will be cancelled before they ever get completed. Further results indicate 52% of projects will cost 189% of their original estimates.

The number and % of challenged projects (over budget, behind schedule, low quality deliverables) has barely changed over 25 years. The cost of these cancellations and overruns are usually hidden just below the tip of the proverbial iceberg.

FIGURE 3: STANDISH GROUP CHAOS REPORTS: 25 YEARS



Looking at the total IT spend for labor with the assumption that 25% of the total applies to development projects, we can estimate the total dollar amount impacted. Using the percentages above to gauge the cost impact, and assuming that all projects are equally funded (on average), we arrive at the dollar amount. The US labor base is about $1 trillion, with $250 billion in development projects. **Therefore $130 billion is lost in troubled projects, and $47.5 billion in cancelled projects.**

A good example of such a troubled project is seen by the state of Rhode Island, which has been having troubles with the $364 million Unified Health Infrastructure Project (UHIP) public assistance program that it rolled out in September 2016 to great fanfare. Like Phoenix, UHIP was intended to save the state millions of dollars per year by reducing processing and staffing costs. Due to myriad operational problems, the cost of UHIP is now pegged at $492 million, not counting the $85.6 million credited back to the state by prime contractor Deloitte. As a point of reference, UHIP was originally slated to cost between $110 million and $135 million and be ready to go live in April 2015. Since its debut, the barrage of significant errors in UHIP has meant thousands of Rhode Island's neediest families have not received the public assistance payments they were eligible for. Flaws in the system still keep showing up. Last October, for example, it was discovered that thousands of applications for benefits were never processed. There are likely similar troubled projects going on in every state (e.g., Texas OAG T2 project[32]). This project[33] was reportedly $200 million over budget and several years late in 2016. They still will not deliver a working system this year as was promised in their corrective action plan.

Technical debt

Technical Debt is a forward-looking metric that represents the effort required to fix problems that remain in the code when an application is released. The CAST Appmarq benchmarking repository[29] was used to benchmark Technical Debt across different technologies, based on the number of engineering flaws and violations of good architectural and coding practices in that code base. CAST bases the Technical Debt calculation in an application as the cost of fixing the structural quality problems in an application that, if left unfixed, put the business at serious risk. Technical Debt includes only those problems that are highly likely to cause severe business disruption; it does not include all problems, just the most serious ones. Based on this definition and the 2011 analysis of 1400 applications containing 550 million lines of code submitted by 160 organizations, CAST estimates that the Technical Debt of an average-sized application of 300,000 lines of code (LOC) is $1,083,000. This represents an average Technical Debt of $3.61 per LOC; and that is just the principal owed. Java applications were higher at $5.45 per LOC.

Assuming that this debt is generally true of all software applications and systems, then estimating the LOC existing in the US today, identifies the contribution of this area to the total CPSQ!

According to Grady Booch[30] in 2005, when asked "how many LOC are written each year around the world?"; he stated that about 35 billion LOC are written each year. This is based on about 15 million or so software professionals worldwide, of which about 30% actually cut code. Each developer contributes about 7,000 LOC a year. Over history, there are about a trillion lines of code written so far up to 2005!

How much has this number of LOC and technical debt grown over the last 13 years?

If we conservatively assume that the worldwide code growth rate is steady at 35 billion new LOC per year, then in 2018 there would be 1.455 trillion LOC worldwide. And assuming that there is $4 of technical debt per LOC then the debt in 2018 would be $5.82 trillion; and in 2020, $6.1 trillion. And if the US share of that debt is roughly 31% (see the section on IT spending), US debt figures would be $1.8 trillion and $1.9 trillion respectively.

The total amount of source code and the total number of projects double about every 14 months.

Looking at the Open Source area for an example of where we have good data on source code in available repositories, we see that code growth follows an exponential growth pattern. Research in 2008[14] shows that the additions to open source projects, the total project size (measured in LOC), the number of new open source projects, and the total number of open source projects are growing at an exponential rate. The total amount of source code and the total number of projects double about every 14 months. This growth rate may not be indicative of non-open source projects since they have so many contributors.

Figured another way, in 2011 CAST Software[42] estimated that the global IT technical debt was $500 billion and would rise to $1 trillion by 2015. If the doubling period is 4 years, then the debt in 2019 would be $2 trillion; and in 2018 $1.75 trillion. The US share of that would be $.54 trillion—a lower figure than our previous calculations would indicate.

Deciding that the latter technique is probably closer to the truth, we therefore conclude that the amount of IT technical debt in the US in 2018 is approximately $.54 trillion. That represents just the debt principal.

Landscape summary

Other important areas that are contributing major problems to the above landscape are:

- Cybersecurity vulnerabilities and the rapid rise in cybercrime
- Commonly used open source software defects
- Purchased software product and multi product suite deficiencies
- Large systems of systems comprised of custom code mixed with COTS and Open source

These issues are not elaborated here due to report size considerations.

...the CPSQ covering the landscape in 2018 is about $2.4 trillion.

In May 2017, analyst firm Forrester produced a detailed forecast for the US technology market, titled *US Tech Market Outlook For 2017 And 2018: Mostly Sunny, With Clouds And Chance Of Rain.* Key takeaways from the report are that spending on software will grow by nearly 10 percent in 2017 and 2018, thanks mainly to increased cloud adoption, while consulting services and staff budgets will rise by 6-7 percent. However, total US tech spending growth will only be around 5 percent, thanks to barely rising budgets for outsourcing, hardware and telecom services. The outlook is that software is currently the main growth area in enterprise IT spending, while security and privacy remains a major priority.

What is the cost of poor-quality software in these major buckets of the landscape of problem areas facing us right now? In summary, they are:

- Legacy system CPSQ—$.635 trillion
- Massive failures and 9-digit defects—$1.275 trillion
- Troubled and cancelled projects—$.178 trillion
- Technical debt—$.54 trillion

It is not yet clear how the above categories intersect, implying the need for deeper research.

Assuming the above categories are mutually exclusive, the CPSQ covering the landscape in 2018 is about $4.234 trillion.

5. Human Talent Perspective on CPSQ

IN THIS SECTION:

- Defining the Information Technology Workforce
- Computer and Information Technology Occupations in the US Today (BLS)
- Impact of the IT gig economy
- Implications for software quality and its costs

With employer demand for IT talent routinely outstripping supply, the year ahead will force more organizations to rethink their approaches to recruiting, training, and talent management. Additionally, questions surrounding skills gaps, diversity, alternative education/career paths and the future of work will demand more meaningful attention and resources.

How many IT and software professionals are there in the USA? How much do they make? And how does that contribute to the possible costs of poor software quality? Getting good answers depends on who you ask, and how they define the job categories of IT and software professionals.

The following reports from 2017-2018 have attempted to quantify the population of IT and software professionals in the US today:

1. **Evans Data** has reported that there are 4.4M software professionals in the US. Evans Corporation, for instance, counts everyone who's actively involved in the creation of software, from rank and file coders to team leaders and managers, all the way up to CTOs. As of 2016, they estimated that there are 21 million professional software developers worldwide.
2. **Don't Quit Your Day Job** has reported 4.2M. It's worth noting that the 4.2 million includes technical writers, electrical and hardware engineers, CAD programmers, actuaries, statisticians, economists, mathematicians, and generally everyone who writes or reads code on a daily basis, in addition to software developers.
3. **StackOverflow** has reported 4.1M
4. **Wikipedia** reports that there are 3.87M

All of these are roughly in the same ballpark. and somewhat dependent on how they defined a software professional. All of these numbers are actually quite low.

IDC published the "2014 Worldwide Software Developer and ICT-Skilled Worker Estimates" document, a study estimating the number of professional software developers, hobbyist developers and Information and Communications Technology (ICT)-skilled workers in the world at the start of 2014. The 90 countries covered in the study represent 97% of the world's GDP.

According to IDC, there are an estimated "29 million ICT-skilled workers in the world as we enter 2014, including 11 million professional developers." Besides those, there are estimated to be another 7.5M hobbyist software developers around the globe, IDC said that worldwide the US accounts for 19 percent of software developers (both professional and hobbyists), followed by China with 10 percent, and offshore outsourcing powerhouse India with 9.8 percent. The US accounts for 22 percent of IT-skilled workers worldwide, followed by India with 10.4 percent and China with 7.6 percent. That would place **the number of ICT workers in the US at 6.4M**.

Defining the information technology workforce

According to CompTIA[64], an analysis of the tech workforce begins with an important distinction. There are two components to the tech workforce.

All workers employed by US technology companies represent tech industry employment. In 2017, an estimated 6.1 million workers were employed in this category, an increase of 2.0% over 2016. For 2018, the growth outlook should roughly mirror the previous year, reaching 6.22 million.

- Tech industry employment includes technical positions, such as software developers or network administrators, as well as non-technical positions, such as sales, marketing, and HR. Note: CompTIA's IT Industry Outlook includes workers employed by companies with payroll, known as employer firms, plus self-employed technology workers.
- The other component of the tech workforce consists of the technology professionals working outside of the tech industry. While the tech industry is the largest employer of technology professionals, with 44 percent of its workforce meeting this criteria, the majority of technology professionals work in other industry sectors, such as healthcare, finance, media, or government.

In 2017, nearly 5.4 million individuals worked as technology professionals across the US economy.

In 2017, nearly 5.4 million individuals worked as technology professionals across the US economy. This represents an increase of 2.1%, or nearly 110,000 net new jobs. Growth in the tech occupation category is expected to hold steady in the year ahead.

Because technology now permeates every industry sector and an increasing number of job roles, the lines have blurred noticeably, making it more difficult to precisely quantify the tech workforce. The expanded spheres circling the Venn diagram is one way of thinking about the new segments workers that cannot be adequately accounted for due to limitations in government data sources.

Software has been a driving force in the tech sector and the broader economy; a trend that has accelerated in the past few years. From mobile apps and SaaS to open-source languages and platforms, software continues to "eat the world," as noted by Marc Andreesen. As such, demand for software development skills make it the largest category of tech occupation and one of the fastest growing. Arguably, categories such as web developers and data scientists could be included under the software development umbrella, making it even larger. Beyond software, categories such as IT support, CIOs, and cybersecurity analysts are growing at a brisk rate, reflecting the needs of organizations pursuing digital transformation.

All in all, there's not really a way to check how accurate any of the numbers above are. Inconsistent job definitions pose the problem.

Computer and information technology occupations in the US today

Analyzing the US Bureau of Labor Statistics for their most recent report (2016) shows that US employment generally increased by 11.5 million over the 2016-26 decade. This is an increase from 156.1 million to 167.6 million, the US Bureau of Labor Statistics reported on October 24, 2017. This growth—0.7 percent annually—is faster than the 0.5 percent rate of growth during the previous decade. About 9 out of 10 new jobs are projected to be added in the services-providing sector. Healthcare support occupations (23.6 percent) and healthcare practitioners and technical occupations (15.3 percent) are projected to be among the fastest growing occupational groups during the 2016–26 projections decade. Several other occupational groups are projected to experience faster than average employment growth, including personal care and service occupations (19.1 percent), community and social service occupations (14.5 percent), and computer and mathematical occupations (13.7 percent). Among the top ten fastest growing occupations they find that application software developers will grow at a rate of 30.7%.

The following table shows the number of IT jobs in the US, their related salary levels and total wages projected through 2018.

TABLE 2: OVERVIEW OF US IT JOBS IN 2018

IT Occupation	Median Annual Pay (May 2017)	Population (2016)	10 yr growth rate (%)	Wage growth/ yr	Population (2018)	Median Wages (2018)	Wages total 2018 (by category)	Population # 2020	Median wages 2020 by category	Wages total 2020 (by category)
Web developers	67,990	162,900	15	3.40	167,787	70,301	11,795,704,626	172,821	75,082	12,975,746,917
Network architects	104,650	162,700	6	3.70	164,652	108,522	17,868,415,985	166,628	116,552	19,420,966,913
Programmers (code/test)	82,240	294,900	-7	2.75	290,771	84,501	24,570,648,534	286,701	89,149	25,559,125,724
Systems analysts	88,270	600,500	9	3.50	611,309	91,359	55,848,854,020	622,313	97,754	60,833,922,729
Database administrators	87020	119,500	11	3.50	122,129	90,065	10,999,633,875	124,816	96,370	12,028,539,628
Info. Security analysts	95,510	100,000	28	3.50	105,600	98,852	10,438,860,960	111,514	105,772	11,795,077,775
Network and systems administrators	81,100	391,300	6	3.50	395,996	83,938	33,239,276,670	400,748	89,814	35,992,818,349
Network support specialist	52,810	835,300	11	3.50	853,677	54,658	46,660,554,389	872,457	58,484	51,025,182,647
User support specialist	52,810	835,300	11	3.50	853,677	54,658	46,660,554,389	872,457	58,484	51,025,182,647
Application Software developers (engineers)	103,560	1,256,200	30	3.50	1,331,572	107,184	142,724,012,191	1,411,466	114,687	161,877,574,627
Systems software developers (engineers)	103,560	1,256,200	24	3.50	1,316,498	107,184	141,108,268,656	1,379,689	114,687	158,233,168,141
Computer and Info. Research scientists	114,520	27,900	19	3.50	28,960	118,528	3,432,600,377	30,061	126,825	3,812,451,935
Computer and Info. System Managers	139,200	367,600	12	3.50	376,422	144,072	54,231,928,012	385,457	154,157	59,420,838,885
		6,410,300			**6,619,050**		**599,579,312,689**	**6,837,127**		**664,000,596,922**
							Total US IT wages in 2018			**Total US IT wages in 2020**

Notes:
Using US Bureau of Labor Statistics database for base information by occupation—accessed 5/28/2018.
Wage growth per year from conservative online salary & recruiting sources.
Extrpolation to 2018 & 2020 values by formula.

Using BLS base data one can project forward to look at population growth and salary growth over the next decade (2016-2026). This allowed us to calculate the US population of IT professionals, and their approximate total salaries over all IT labor categories to come up with a total level of US IT salary expenditure for 2018 of **$600 billion** for the categories defined by the BLS.

These numbers appear quite low considering the many categories of IT professionals not represented in the BLS database. For example, missing categories include: C-suite IT executives, QA specialists, data scientists, and gig part-timers, etc.

Impact of the IT gig economy[65]

It is predicted that freelancers will comprise the majority of the US workforce within a decade.

The gig economy has exploded in the past five years, with more than 57 million Americans working freelance jobs today—about 36 percent of the US workforce—according to the "Freelancing in America" study conducted last year by Edelman Intelligence. It is predicted that freelancers will comprise the majority of the US workforce within a decade. As it turns out, the top best-paying positions per hour in the gig economy are in IT. The top software developer gigs are: AI/ML, blockchain architect, ethical hacking, augmented and virtual reality, and Amazon web services.

"The gig economy…is now estimated to be about 34% of the workforce and expected to be 43% by the year 2020," Intuit CEO Brad Smith said in May 2017. Conservatively estimating that 33% of the IT workforce are now gig workers, adding roughly another $300 billion in salaries derived from the BLS database, totals an IT workforce of $900 billion/year. Adding in the missing categories from the BLS database, the $1 trillion mark is easily reached.

Implications

In 2018 approximately 6.6 million Americans were working in computer or information technology fields. Most of these jobs pay well, many with benefits. IT professionals typically enjoy good job security as well. This number does not include "gig economy" IT workers, nor non BLS IT labor categories.

According to the US Bureau of Labor Statistics, employment of computer and information technology occupations is projected to grow 13 percent from 2016 to 2026, faster than the average for all occupations. These occupations are projected to add about 832,000 new jobs. Demand for these workers will stem from greater emphasis on emerging technologies like cloud computing, big data analytics, and information security. According to CompTIA research, nearly 4 in 10 US IT firms report having job openings and are actively recruiting candidates for technical positions. Hiring intent is most concentrated among large- and medium-size firms. Among hiring employers, more than half indicate it's due to expansion, while a similar percentage indicate the need for new skills in areas such as software development, IoT, or data is driving the hiring at their firm. Demand for IT talent continues to exceed the supply of such talent.

IT professionals are also experiencing annual salary growth of about 3.4-4.0 %[10]. For example, an IT solutions architect is seeing 3.7%, and web designer at 3.4%.

And with all the jobs in other fields that require some level of software development prowess, it looks like some day software development may well become the new literacy.

With an estimate of the total IT and software population and their total salary expenditures for 2018 and beyond, our attention focuses on the central question of this study: "How much of the total work effort, and those total salary dollars are being lost to poor-quality software?"

The total US **IT professional wage base in 2018 is ~$1 trillion.** Asking all of those IT professionals to write down what activities they actually spend their time on, provides us with a surprising answer. Only a few empirical studies to determine that have ever been done—the commonly held assumption being that the activities done must be in line with the project plan's WBS or the project's defined SDLC process. But what if that was not true?

Several empirical investigations have suggested that software developers actually spend most of their time in the following activities:

1. Knowledge acquisition (especially problem domain and new technology)
2. Finding and fixing problems and deficiencies
3. Rework
4. Communicating with others
5. Dealing with changing expectations and requirements
6. Other waste due to cancelled or challenged projects

Clearly items 2, 3, 6, in the above list are major parts of the cost of poor-quality in software development. These are reflected in our model of the cost of software quality in the next section.

According to Jones[62], the amount of software effort spent on software projects that will be cancelled due to excessive error content appears to absorb more than 20% of the US software work force. In addition, about 60% of the US software engineering work time centers on finding and fixing errors, which might have been avoided. Finally, software schedules for major applications are about 25% longer than they should be due to poor-quality expanding testing intervals. He has dubbed this "wastage".

Unfortunately, IT and software organizations do not collect effort data in this way. Nor do most of them actually collect data on the cost of software quality. There are some, which have done that successfully—but just a few. Most cost tracking systems for software development projects actually omit as much as 50% of the total effort[9]. For example, uncompensated overtime is very rarely captured by formal cost tracking systems, and yet software engineers routinely work 50-60 hours per week. Other omissions include management effort, and the work of part-time specialists such as quality assurance and business analysts.

Laying out the actual state of collecting effort data on IT projects, focuses on how much of technical staff time is spent on dealing with poor-quality software.

In my experience with helping organizations establish a CoSQ initiative from an immature starting point, it is often the case that initially they are spending 50-80% of their time on CoSQ; and at the project level it varies quite a bit depending on type and size of the project. And most of that time is being spent on poor-quality issues. The relationship between process maturity, product quality and cost of poor quality (specifically rework) can be seen in the table below. [58, 59]

TABLE 3: PROCESS MATURITY, PRODUCT QUALITY AND COST OF QUALITY

Process Maturity (characteristic)	Rework (% of total development effort)	Product Quality (defect density)
Immature	>=.50	double digit
Project controlled	.25 – .50	single digit
Defined org. process	.15 – 25	.X
Management by fact	.05 – .15	.OX
Continuous Learning & Improvement	< = .05	< .00X

There is a correlation between six sigma levels and CoSQ[60] as seen in the table below.

TABLE 4: SIGMA LEVEL AND COSQ

Sigma Level	DPMO	CoSQ as % of Sales
2	289,000	>40
3	67,000	24–40
4	6,000	15–25
5	233	5–15
6	3.4	< 1

Assuming that the average performance of a company is 2.5 sigma (10% buggy code), ~ 40 percent of its annual revenue gets chewed up by the cost of quality.

iSixSigma.com

Assuming that the average performance of a company is 3 sigma, 25 percent to 40 percent of its annual revenue gets chewed up by the cost of quality, more than half of which is CPSQ.

The table below indicates the range of CoSQ performance between the best and worst companies I have personally encountered.

FIGURE 4: CoSQ Targets and Benchmarks

Caveat—this is not based on a scientific study—it is my hypothesis based on a few observations of my clients

CoSQ Indicator \ Companies Observed	Worst	Best
% of development resources	> 70%	< = 30%
% of net SW sales*	> 15%	< 1%

*speculative and highly depended on type of business

Given such a wide range of organizational performance, it is difficult to estimate an average. Assuming that the median is closer to best performance, an optimistic estimate the cost of poor-quality software approximates 35% of all IT labor spending. That means that the CPSQ in the US for 2018 (from a technical labor expenditure perspective) is at least **$319 billion**.

6. Cost of Software Quality: Definitions and Model

IN THIS SECTION:

- Definition of software quality
- Good versus poor-quality software
- The cost of software quality model and its evolution
- Categories of CPSQ
- Categories of CGSQ

Definition of software quality

What gets measured, gets improved!

"Quality" can mean different things to different people. The concept and vocabulary of quality is elusive. The meaning differs depending upon circumstances and perceptions. The dictionary definition of Quality (in general)

1. the standard of something as measured against other things of a similar kind; the degree of excellence of something.
2. a distinctive attribute or characteristic possessed by something.

The ISO 8402 standard defines quality as "the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs [now]." Quality is a different concept when focusing on a tangible software product versus the perception of a quality service enabled by software. The meaning of quality is time-based or situational.

Consumers now view quality as a fundamental measure of their total perception/experience with a product or service, as well as of the company, delivery and maintenance network that provides and supports it—a kind of unified "quality-value" metric.

A general definition of software quality was provided in the introduction section of this paper. While sufficing for general discussions, the need for each project to have its own more specific definition is necessary. Software quality is more precisely defined as a combination of the following 4 aspects:

1. **Conformance to requirements**
   - The requirements are clearly stated and the product must conform to it
   - Any deviation from the requirements is regarded as a defect
   - A good quality product contains fewer defects
2. **Fitness for use/purpose**
   - Fit to user expectations: meet user's needs
   - A good quality product provides better user satisfaction

3. **Meeting standards**
   - In many industries and organizations certain external and internal standards must be complied with
   - A good quality product conforms to required standards of quality/process (e.g., ISO 25000, CMMI level)
4. **Underlying aspects, which include**
   - Structural quality (e.g. complexity)
   - Aesthetic quality (e.g. appearance)

Every application or business domain faces a specific set of software quality issues, and software quality must be defined accordingly. A definition fashioned from the above aspects and/or applicable standards should be created for your organization and for each project. The series of standards ISO/IEC 25000, known as SQuaRE (System and Software Quality Requirements and Evaluation), creates a framework for the evaluation of software product quality. An example of how to use the ISO 25000 framework to establish software quality goals can be provided on request.

The key point is having such a definition of software quality for your project that is measurable.

Good versus poor-quality software

If there was a simple measure for "good" software, we'd all be using it, and everyone would demand it.

In practice several metrics are used as indicators, usually in combination. For example:

- Defect trend over time is often used to differentiate—good is a decreasing curve, poor is an increasing curve.
- Testing code coverage has been used as a surrogate—but doesn't speak to the quality of the tests themselves.
- Cyclomatic complexity, depth of inheritance, degree of class coupling, and a few other structural metrics, are indicators of sub-par code.
- The amount of effort that it takes to understand what a piece of code does is another good indicator.

Poor-quality is not an inevitable attribute of software. It results from known causes. It can be predicted and controlled, but only if its causes are understood and addressed. As explained by Curtis[2] enhanced by this author, the primary causes of poor software quality are:

- Lack of domain knowledge (resulting in poor requirements)
- Lack of technology knowledge (resulting in uncertainty about goodness of components)
- Unrealistic schedules (from poor estimation practices)

- Badly engineered software (resulting from immature, undisciplined practices; and using less qualified software engineers)
- Poor acquisition practices
- Communication and coordination breakdowns in teams
- Lack of useful data about the state of software quality

The first two causes distinguish between functional and non-functional quality problems, a critical distinction since non-functional defects are not detected as readily during test and their effects are frequently more devastating during operations. The third and fourth causes have been perennial, although the fourth problem is exacerbated by the increase in technologies integrated into modern applications. The fifth problem is not entirely new but has grown in effect with growth in outsourcing and packaged software. The sixth problem is one that grows as the team gets larger and more dispersed. The seventh problem applies to those organizations that do not collect or report useful software quality data (e.g. tracking defects).

By understanding and addressing these top causes, quality can be designed in from the start, substantially reducing both the 40% of project effort typically spent on rework and the risks to which software exposes the organization.

We now turn our attention to the cost of software quality, and especially the cost of poor-quality software, which includes such things as the costs of lost goodwill, and expenses incurred in recalls, refund, replacement, rework, waste, fixing deficiencies, cancelled projects, etc.

The cost of software quality model and its evolution

In August 2013, Amazon lost $4.8M after going down for 40 minutes due to a software "glitch"; that breaks down to about $120,000 per minute. That number is certainly much higher today. And it is certain that executives in Amazon know their precise CPSQ figure. As a result, Amazon executives put proactive and predictive quality management practices at the forefront.

Cost of quality is a methodology that allows an organization to determine the extent to which its resources are used for activities that directly effect the quality of the organization's products or services, and that result from failures and deficiencies. Having such information allows an organization to determine the potential savings to be gained by implementing process and product improvements.

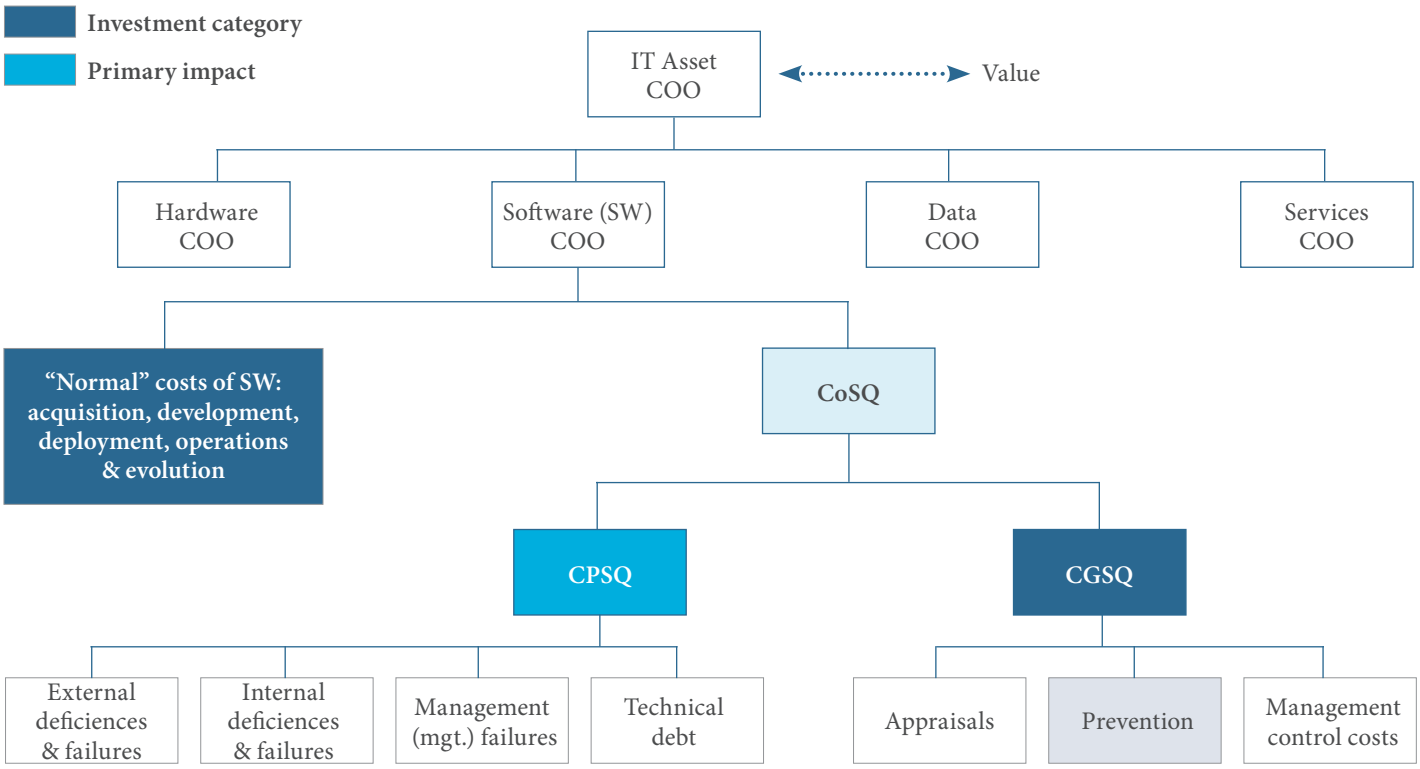
The original model of quality costs was divided into four categories:

1. External Failure Cost
2. Internal Failure Cost
3. Inspection (appraisal) Cost
4. Prevention Cost

This model was primarily applied to manufacturing situations. In that context the focus was on non-conformance to specifications and standards. This was found to be insufficient for software. In the 1990's this author and several others adapted and applied the model to computer software[63, 67, 68].

In our enlightened new model of the cost of software quality we have divided up the old categories and added several new categories to reflect the different nature of software. The rationale for this new model is that it makes the tradeoff between poor and good quality software much more apparent. And it adds new categories such as technical debt and management failures that were not in the original model[1].

FIGURE 5: IT ASSET AND COSQ CONTEXT MODEL



Legend: COO = cost of ownership CoSQ = cost of software quality CPSQ = cost of poor SW quality CGSQ = cost of good SW quality

As we can see in the above figure, the Cost of Software Quality (CoSQ) is a part of the Cost of Ownership (COO) for the software component of a system, which is a part of the total COO for an IT asset that contains software. As we look at process and product improvements, quantifying the "quality" costs to the organization is defined as the Cost of Quality (COQ). Why quantify the quality data? The COQ categorizes these costs, so the organization can see how moving from a quality assurance (control and correction) focus to a focus on prevention helps to reduce the cost of failures and deficiencies. The American Society of Quality (ASQ) uses the following formula to calculate the Cost of Quality (COQ), and we have adapted it to software as:

Cost of Software Quality (CoSQ) =
Cost of Poor Software Quality (CPSQ) + Cost of Good Software Quality (CGSQ)

As highlighted in the figure on the previous page we show that investing in software engineering discipline and in the Cost of Good Software Quality (CGSQ), will dramatically reduce the Cost of Poor Software Quality (CPSQ).

Categories of CPSQ

Cost of poor-quality (COPQ): These are the costs associated with providing poor-quality work products, systems or services. There are four categories: internal failure costs (e.g. costs associated with defects found before the customer receives the product or service), external failure costs (e.g. costs associated with defects found after the customer receives the product or service), appraisal costs (costs incurred to determine the degree of conformance to requirements and quality standards) and management failures (costs incurred by executives and below dealing with the ramifications of poor-quality software).

Internal Failure and Deficiency Costs

These costs are associated with system failures and deficiencies discovered before the system leaves the development organization and is deployed into the operational environment. These deficiencies occur when a system fails to meet a certain requirement, resulting in waste or rework. The deficiencies could be in the work products of development, the development process, and/or components if they fail to meet quality standards and requirements. The largest category of costs here are the professional effort to find and fix all of the defects. The impact of cancelled and delayed projects are also included here. Unfortunately, very few organizations track this category prior to the commencement of testing. Included are:

- **Waste**—performance of unnecessary work or holding of work products as a result of errors, poor organization, or communication, cancelled and challenged projects
- **Scrap**—defective product or material that cannot be repaired, used, or sold
- **Rework or rectification**—correction of defective material or errors, agile refactoring
- **Failure analysis**—activity required to establish the causes of internal failure

External Failure and Deficiency Costs

These costs occur when products or services that fail to reach quality standards are not detected until after transfer into operation or to the customer. External failure/deficiency costs are incurred during customer use and can include defective products, warranty charges, customer complaints, rejections, recalls, returns, patches and repairs. While external costs are the most apparent, these costs sometimes can be difficult to quantify. Therefore, businesses fail to include them in the overall quality costs because failures such as poor installation and usage problems are not always reported by the customer. A large category of costs here are massive failures, and latent defects in the software when delivered. The largest category of costs here are

professional effort to replicate, find and fix all of the fielded defects and re-appraisals to verify fixes. Loss of sales, tarnished reputation, legal/litigation and excessive customer complaint handling, are large costs in this category. Included are: wasted marketing costs, brand damage, and technical support team effort.

Technical Debt

Technical debt in software is a relatively new concept. The term was coined by Ward Cunningham to describe the obligation that a software organization incurs when it chooses a design or construction approach that's expedient in the short term but that increases complexity and is costlier in the long term. There are two basic types of technical debt: intentional and unintentional. One of the important implications of technical debt is that it must be *serviced*, i.e., once you incur a debt there will be interest charges. A good example of this is future refactoring that needs to be done.

Technical debt is measurable. For example, one organization we've heard about maintains a debt list within its defect tracking system. Each time a debt is incurred, the tasks needed to pay off that debt are entered into the tracking system along with an estimated effort and schedule. The debt backlog is then tracked, and any unresolved debt more than 90 days old is treated as critical. Since this is a fairly new concept, there are still questions raised about what kinds of flaws should or shouldn't be classified as Technical Debt. There are others who define it more precisely in order to quantify the level of structural quality problems in the operational system. Structural quality metrics measures how well a system is designed and constructed with respect to best practices.

Management Failures

These are the non-technical costs incurred by an organization who suffers from poor-quality software management practices at the executive level and below. This includes:

- Unplanned costs for professional and other resources, resulting from underestimation of the resources in the planning stage.
- Damages paid to customers as compensation for late project completion, a result of the unrealistic schedule in a project's proposal/plan.
- Damages paid to customers as compensation for late completion of the project, a result of management's failure to recruit qualified team members.
- Damages to other projects planned to be performed by the same teams involved in the delayed projects. The domino effect may induce considerable hidden failure costs.
- Excessive management crisis mode behaviors, like lots of meetings to solve urgent problems.
- Hidden external failure costs, that is, reduction of sales as a result of damaged reputation, increased investments in sales promotion underpricing of tender bidding to counter the effects of significant past delayed completion of projects due to managerial failures in appraisal and/or progress control tasks.

Categories of CGSQ

The cost of good software quality is as variable as the organizations represented. Some groups invest a lot in proactive quality management and planning, while others make do with patchwork systems and reactive programs aimed at solving problems after they occur. The costs of good quality are broadly broken down into management control costs, prevention costs and appraisal costs.

Appraisal Costs

Appraisal costs are those associated with actions designed to find quality problems with measuring, evaluating, inspecting, testing and auditing systems and work products to ensure they adhere to the quality standards and performance requirements. Investing in the resources to identify and ultimately diagnose poor-quality helps an organization reach its strategic objectives and increase the value of the system and overall customer satisfaction. The biggest buckets of cost here are usually testing and QA. Included are: V&V, quality audits, inspections, peer reviews, supplier assessments, etc.

Prevention Costs

Prevention costs are incurred to prevent or avoid quality problems. These investments keep product failure/deficiency costs to a minimum and can help reduce appraisal costs. Eliminating defects before deployment begins reduces the costs of quality and can help companies increase profits. Prevention costs include process planning, review and analysis of quality audits and training employees to prevent future failure. The major buckets are proactive quality management, quality planning, training, and improvement programs.

Management Control Costs

Management can perform several activities to prevent or reduce the costs that result from the types of failure particular to its functions: contract reviews, planning, goal establishment, and progress review and control of the software project. This includes:

- Costs of carrying out contract reviews
- Establishing quality goals, objectives, gating/release criteria and quality standards
- Costs of preparing project plans, including quality management plans
- Costs of periodic updating of project and quality plans
- Costs of performing regular progress review and control
- Costs of performing regular progress control of external participants' contributions to projects

A detailed chart of accounts for our CoSQ model is beyond the scope of this report. The author can be contacted for examples of such.

Understanding Cost of Poor Software Quality in your organization is the first step toward gaining executive buy into quality-led operations. This is fundamental to agile, DevOps as well as Proactive and Predictive Quality Management. With a CPSQ number in hand, you have the basis for a business case to invest smartly in quality. Determining CPSQ may sound daunting, but in fact, it's very achievable and simply requires some tried-and-true methods along with a cross-functional team to get the brainstorming on paper. This author has developed a survey instrument to help organizations get started. You can gain an understanding of the true impact of problems, mistakes, bugs, defects, security gaps, and general sloppiness.

A good example of what can be learned by measuring the cost of software quality can be read[65] which showed that the CoSQ represented 33 percent of the overall project cost, and they were at CMMI Level 3. CoSQ is much higher in immature organizations, perhaps reaching 66%.

7. Conclusions

This report does not represent a new scientific study. It is an aggregation of publically available source material that was found to be pertinent to the first order approximation of the cost of poor software quality in the US today; and then how that knowledge might best be leveraged to stimulate software quality improvement programs widely across industry and government.

We present our conclusions, with the understood caveat that most IT and software organizations do not now collect CoSQ data. We believe this may be true because without an understanding of a defined model of CoSQ most IT leaders would not have a basis for estimating the answers to our questions, e.g.,:

- How much are you spending today on the cost of poor-quality software in your organization?
- How are your investments in good software quality affecting your overall costs of quality and cost of ownership for software assets?

What the various sources have revealed—the cost of poor-quality software

Using our cost of software quality model introduced in the proceeding section, we have broken down the cost of poor software quality into these four main buckets:

1. **External deficiencies and failures** – the largest chunks of which are: finding and fixing operational deficiencies, and massive failure consequences
2. **Internal deficiencies and failures** – the largest chunks of which are: finding and fixing unreleased deficiencies, rework, cancelled projects, and troubled projects.
3. **Technical debt** – the largest chunks of which are the violations of good practices, and fixing problems that may cause future disruptions
4. **Management failures** – the largest chunks of which are unanticipated costs, customer damages, and reactive crisis-mode management behaviors

In the area of external deficiencies and failures we found that:

- From the perspective of what we know about the needs of legacy systems, and the dominant role they play in most IT shops; we have estimated that the cost of poor-quality software in this area is about **$635 billion** this year.
- From the perspective of the current failures that we are seeing in fielded software driven systems, we can observe that massive failures are happening at an increasing rate; and that cybersecurity vulnerabilities are rampant throughout software system infrastructures and technology stacks.

a. The costs of these cyber breaches are not cheap, both in dollars and reputation. Statistics released in late February from a Council of Economic Advisers report on the impact of cyber attacks on US government and industry set the aggregated cost of malicious cyber activity between $57 billion and $109 billion in 2016. In 2018, these cyber attacks are estimated to cost **$126 billion**.

b. In 2012, Gene Kim and Mike Orzen (Lean IT) calculated the global impact of IT failure as being $3 trillion annually. If the USA accounts for about 31% of that global IT spend, we could assume that they suffer 31% of the failure costs as well, which would put US losses at about $.93 trillion. Extrapolating that over 6 years at a conservative growth rate of 3% per year, would mean that the cost of IT failures in the USA in 2018 would be about **$1.11 trillion**.

The sum total of external deficiencies and failures would be $1.76 trillion if there was no overlap in the above categories, or $1.1 trillion if there was complete overlap. We choose to believe that the truth is somewhere between and so choose the mid point as our estimate—**$1.43 trillion**.

In the area of internal deficiencies and failures, we found that:

- From the perspective of the total portfolio of current IT projects, we know that about 1/3 of them will be cancelled or will fail in a significant way. That places about $300 billion of the total US IT labor base at risk. Cancelled projects due to schedule slippage or cost overruns are most frequently caused by poor software quality. We estimate that $130 billion is lost in troubled projects, and $47.5 billion is lost in cancelled projects.
- Finding and fixing internal problems and deficiencies—assuming that about 50% of the US IT development effort is spent in this area, the cost would be about $500 billion.

In the area of technical debt, we found that:

- The largest chunks are the violations of good practices, and fixing known, postponed problems that may cause significant disruptions. We estimate that the CPSQ in this area is $.54 trillion in technical debt in 2018; a huge number. This represents a potential future cost, and there may be many situations in which this debt is not paid or is forgiven.

In the area of management failures, we found that:

According to a study[88] by IBM and Ponemon Institute of compromises to business continuity or security due to IT risks, they estimated the cost of business disruptions to be:

Type of event	Length of disruption (minutes)	Cost (per minute)	Likelihood of event over next 2 years
Minor	20 min.	$53,210	69%
Moderate	112 min.	$38,065	37%
Substantial	442 min.	$32,229	23%

The most telling statistic of the study was that 75% of event costs go to reputational damage and the bottom line. Since we have no simple way of directly converting this into a cost of poor-quality software, we will leave it out of our totals.

Summary of poor software quality costs

If we take all of these different perspectives into account, we can hone in on the approximate total CPSQ in the US this year.

Using the CoSQ model from section 6, the main components of our starting estimate are therefore:

1. External failures and deficiencies – $1.43 trillion
2. Internal failures and deficiencies – $.8 trillion
3. Technical debt – $.54 trillion
4. Management failures – unknown contribution at this time

In summary, the cost of poor-quality software in the US in 2018 is approximately $2.84 trillion, the components of which are seen in Figure 1 on page 5.

One could argue that technical debt should not be included, as it represents a future cost which may or may not be paid back. Often times technical debt laden legacy systems are simply replaced in which case the costs are shifted into new development. We could not find any hard data on what percentage of technical debt gets paid back versus forgiven. If we remove the future cost of technical debt, the total CPSQ becomes $2.26 trillion. It was our intention to use this result as a starting point for community discussion and future benchmarking studies.

If we attempt to project these costs into the future, we must make additional assumptions about the growth factors and growth rate in each area. As we showed in the landscape section, if we use a conservative code growth rate of 35 billion new LOC per year worldwide, and if we attribute 31% of that to the US, we can estimate the CPSQ and the technical debt in future years.

If the IT labor force is continuing to grow at about 2% per year, and if productivity rates have improved, and if salaries for US IT professionals are increasing at about 3.5% per year, then the assumption about stable code growth rates in the calculations above must be revisited. Consequently, the CPSQ and technical debt figures would be higher. Given these considerations it would be reasonable to raise the above cost by approximately 10%.

Other observations

We have observed that the term "quality costs" means different things to different organizations. Whether it's the costs of finding and correcting problems in quality or the costs to attain quality, they can be significant—e.g., nearly 20 to 40 percent of a company's sales, according to Juran's Quality Control Handbook[70]. In software, the difference between poor-quality and good-quality is felt differently for different types of software usage situations and system sizes.

Poor-quality is opportunity lost. Understanding where and why these opportunities exist is a chance to improve the bottom line. Although there's good value to be gained by reducing waste, rework, and warranty costs, there's even greater value in looking for the root causes of these problems, because the return is normally much greater.

The bottom line, according to Capers Jones[69], is that poor-quality software costs more to build and to maintain than good quality software, and it can degrade operational performance, increase users error rates, and reduce revenues by decreasing the ability of employees to handle customer transactions or attract addition clients. For the software industry, not only is quality free, but it benefits the entire economic situations of both developers and clients. The details of exactly how much can be saved through a focus on good quality is seen in the results of industry-wide improvement programs for software.

What to do

When people think about what it takes to build mission-critical and safety-critical software, they usually consider the complexity of the undertaking. There is a belief that in order to have quality, an organization needs to have a cumbersome process with oversight, standards, formality, and documentation. But such assumptions are not correct. There are many ways to achieve better software quality. While some businesses may require Total Quality Management, or Six Sigma, or lean and agile, or pursue Capability Maturity Model certification, most organizations need not adopt such rigorous programs as their first steps.

It is important to recognize that software quality improvement, like software development, is an iterative process. There is no need to accomplish everything with a single step; improvement can be accomplished in incremental steps. Even small changes can make a tangible difference, including adjusting the organizational

The key strategy for reducing the cost of poor software quality is to find and fix problems and deficiencies as close to the source as possible, or better yet, prevent them from happening in the first place.

attitude towards quality. Software quality improvement requires a commitment from business leadership and a mindset change that begins at the top.

The business benefits of good software quality are both broad and deep. Not only does quality facilitate innovation by increasing predictability, lowering risk, and reducing rework; it serves as a differentiator, since it enables a business to set itself apart from its competitors. Most importantly, continuously ensuring quality will always cost less than ignoring quality. Quality is more than free when it is done right.

The key strategy for reducing the cost of poor software quality is to find and fix problems and deficiencies as close to the source as possible, or better yet, prevent them from happening in the first place. This strategy implies that wise investments in software engineering discipline and the cost of good software quality will dramatically reduce the cost of poor-quality, as seen in our cost of quality model in the previous section of this report. The concept of Continuous Testing is gaining traction in the industry for exactly this reason. Continuous Testing is generally defined as making testing an ongoing, automated and constant part of the software development lifecycle—so that defects can be found and corrected as soon as they are introduced. Measuring the cost of software quality in your organization is a recommended first step.

8. Acknowledgements

The author would like to thank the following individuals, whose contributions to this work, either directly or indirectly, has inspired me over my professional career.

Of special note are the industry thought leaders about software engineering empirical studies, Bill Curtis and Capers Jones. Caper's foundational work on the economics of software quality was highly influential, and widely quoted in this report. Bill's work on standard measurements of software quality and technical debt was highly influential, and frequently quoted. I value the four decades of interaction with Bill over important topics in the field of software engineering discipline—thanks, Tex.

Grateful thanks to my professional *posse* for their contributions to my thinking about these subjects, especially Don and Linda Shafer, for their most helpful review comments on the drafts of this report. Also thanks to our project administrator, Tracie Berardi, and our professional designer, Cathleen Schaad.

Of special note is the corporate sponsor of this work, Brendan Hayes of CA Technologies for his leadership in making this report happen.

About the author

Herb Krasner is a retired Professor of Software Engineering at the U. of Texas at Austin. He is an industry consultant for 5 decades; helping organizations baseline and improve their software development capabilities. More recently he has been involved in establishing required measurements for large IT projects in the state of Texas. He is an active member of the CISQ Advisory Board and is well known for his previous research in the empirical studies of software professionals, the ROI of software process improvement, and the cost of software quality. He can be reached at hkrasner@utexas.edu. For more information google his name or see: http://www.ece.utexas.edu/people/faculty/herb-krasner

9. Section References

Forward Section References

1. https://www.standishgroup.com/outline
2. https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf
3. https://www.castsoftware.com/research-labs/crash-reports
4. https://www.tricentis.com/software-fail-watch/

Introduction Section References

5. https://www.gartner.com/technology/research/it-spending-forecast/
6. https://www.apptio.com/emerge/trends/gartners-2018-it-spend-prediction
7. https://www.apptio.com/thankyou/resources/research-reports/2018-state-global-technology-economy?aliId=13732465
8. https://www.comptia.org/resources/it-industry-trends-analysis
9. https://www.idc.com/getdoc.jsp?containerId=prUS42833317
10. *The Economics of Software Quality,* Capers Jones and Oliver Bonsignour, Addision Wesley, 2012
11. *Programming Productivity*, Capers Jones, 1986, Table 3-25, p. 179.
12. https://www.infoworld.com/article/3130217/software/the-era-of-nine-digit-defects.
13. Why Does Software Cost So Much? Initial Results from a Causal Search of Project Datasets, Konrad, et al., 2018, CMU SEI Tech Report to be published in 2018
14. *Using the Cost of Quality Approach for Software*, Krasner, H. and D. Houston, CrossTalk magazine, The War on Bugs, Nov. 1998
15. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html
16. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions), https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
17. Worldwide internet user penetration from 2014 to 2021, https://www.statista.com/statistics/325706/global-internet-user-penetration/

Landscape Section References

18. 2018 Open Source Security and Risk Analysis; Synopsys Center for Open Source Research & Innovation
19. Amy DeMartine, The Forrester Wave™: Software Composition Analysis, Q1 2017, Forrester, 2017.
20. The ROI from Software Quality, Emam, K. E, Auerbach Publications, 2005
21. http://pcdgroup.com/the-pros-and-cons-of-custom-software-vs-off-the-shelf-solutions/
22. Bill Curtis, Oct. 14, 2016 - https://www.infoworld.com/article/3130217/software/the-era-of-nine-digit-defects.html
23. Watts S. Humphrey, A Discipline for Software Engineering, Addison Wesley, 1996
24. An Introduction to Automatable Standards for Software Measurement, Dr. Bill Curtis, June 19, 2018, CISQ Texas Summit
25. https://www.statesman.com/news/state--regional-govt--politics/attorney-general-tech-project-200-million-over-budget/PVxJjXzobcGk9cZyL4EVEN/
26. https://www.tricentis.com/software-fail-watch/

27. OAG T2 link to the news
28. LOC - lines of code, aka source lines of code (SLOC), is a software metric used to measure the size of a computer program by counting the number of lines in the text of a program's source code. – Wikipedia.
29. https://www.castsoftware.com/research-labs/technical-debt-estimation
30. http://www.aspectprogrammer.org/blogs/adrian/2005/03/grady_boochs_ke.html
31. Deshpande, A. and Riehle, D., 2008, in IFIP International Federation for Information Processing, Volume 275; Open Source Development, Communities and Quality; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 197–209. https://link.springer.com/content/pdf/10.1007%2F978-0-387-09684-1_16.pdf
32. https://www.forbes.com/sites/forbespr/2017/05/31/poor-quality-data-imposes-costs-and-risks-on-businesses-says-new-forbes-insights-report/#7d05734d452b
33. https://insidebigdata.com/2017/05/05/hidden-costs-bad-data/
34. http://www.ibmbigdatahub.com/infographic/four-vs-big-data
35. https://thehackernews.com/2018/07/samsam-ransomware-attacks.html?mkt_tok=eyJpIjoiTmpSa05EZGhNREEyTnpCbCIsInQiOiIzQzFJbUJXY2k0Q1B0V1BBeit4REt4c21OZDdxa1RoeG plUGFsQ0hFT2xcLzF5Tkl2UGJHZ2tpZ3FSaVRYcVVpaGRDUnlNYWZIN0YycThRVzFmNWd4aXF5aDdBSmw5cW5 ZWXg1ek9xS1NEWCtHeTZXbG9GOG9QSTJKaDNYdEJZbjQifQ%3D%3D
36. https://zephoria.com/top-15-valuable-facebook-statistics/
37. https://www.tricentis.com/software-fail-watch/
38. https://www.activestate.com/sites/default/files/pdfwp/whitepaper-true-cost-opensource.pdf
39. NIST Cybersecurity Framework - https://www.nist.gov/cyberframework
40. 5." World's Biggest Data Breaches ," Information Is Beautiful , blog, July2018 ; http://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks .
41. *Practical Software Maintenance*, Pigoski, T.M., Wiley, 1997
42. How to Monetize Application Technical Debt, CAST Software Analyst Report, 2011
43. Why Google Stores Billions of Lines of Code in a Single Repository, https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext
44. https://docs.microsoft.com/en-us/azure/devops/git/git-at-scale
45. Bill Curtis, Oct. 14, 2016 - https://www.infoworld.com/article/3130217/software/the-era-of-nine-digit-defects.html
46. https://www.tricentis.com/software-fail-watch/
47. Bill Curtis, Oct. 14, 2016 - https://www.infoworld.com/article/3130217/software/the-era-of-nine-digit-defects.html

Human Talent Section References

48. *"Global Developer Population and Demographic Study 2016 V2". Evans Data Corporation. Retrieved 19 January 2017.*
49. *"United States Labor Force Statistics Seasonally Adjusted"*. Labor Market Information. Rhode Island Department of Labor and Training. October 2016
50. U.S. Census Bureau, DataFerrett, Current Population Survey, Basic Monthly Microdata, March 2015
51. Software Developer Statistics: How Many Software Engineers Are There in the US and the World? Oct 31, 2017.
52. https://www.daxx.com/article/software-developer-statistics-2017-programmers
53. https://www.bls.gov/ooh/occupation-finder.htm
54. https://www.bls.gov/ooh/computer-and-information-technology/home.htm
55. Other web sites have a much more detailed analysis of IT professional salaries in the US. See for example https://www1.salary.com/IT-salaries.html

56. *The Technical and Social History of Software Engineering*, Capers Jones, Addison Wesley to be published in the autumn of 2013
57. Modis Technology 2018 Salary Guide, http://www.modis.com/clients/salary-guide
58. *Using the Cost of Quality Approach for Software*, Krasner, H. ,and D. Houston, 1998, CrossTalk Magazine, the War on Bugs
59. Krasner, H., 1990, Self Assessment Experiences at Lockheed, Proceedings of the SEI/AIAA Software Process Improvement Workshop, November 8, 1990, Chantilly, VA
60. https://www.isixsigma.com/implementation/financial-analysis/cost-quality-not-only-failure-costs/
61. http://theinstitute.ieee.org/ieee-roundup/blogs/blog/toppaying-jobs-in-the-gig-economy-are-in-tech
62. Capers Jones, WASTAGE: THE IMPACT OF POOR QUALITY ON SOFTWARE ECONOMICS, Version 8:0 September 3, 2017

CoSQ Section References

63. Using the Cost of Quality Approach for Software, Krasner, H. and D. Houston, CrossTalk magazine, The War on Bugs, Nov. 1998
64. Curtis, Bill, July 1, 2009 - https://www.datamation.com/entdev/article.php/3827841/Top-Five-Causes-of-Poor-Software-Quality.htm
65. Measuring the Cost of Software Quality of a Large Software Project at Bombardier Transportation: A Case Study, Claude Y. Laporte et al, Software Quality Professional magazine, VOL. 14, NO. 3/© 2012, ASQ
66. http://it-cisq.org/wp-content/uploads/2017/11/IT-Quality-Measurement-Implications-for-Large-IT-Projects-in-Texas-Nov-2017.pdf
67. http://sunset.usc.edu/cse/pub/event/archives/pdfs/Herb_pt2.pdf
68. ftp://itin.sei.cmu.edu/pub/documents/95.reports/pdf/tr017.95.pdf

Conclusion Section References

69. *The Economics of Software Quality*, C. Jones and O. Bonsignour, Addison Wesley, 2012
70. Juran's Quality Handbook: The Complete Guide to Performance Excellence 6/e 6th Edition by Joseph A. Defeo (Author), J.M. Juran (Author)
71. http://www.ibmbigdatahub.com/infographic/four-vs-big-data
72. https://www.zdnet.com/article/worldwide-cost-of-it-failure-revisited-3-trillion/
73. https://raygun.com/blog/cost-of-software-errors/
74. https://hbr.org/2016/09/bad-data-costs-the-u-s-3-trillion-per-year
75. https://www.forbes.com/sites/forbespr/2017/05/31/poor-quality-data-imposes-costs-and-risks-on-businesses-says-new-forbes-insights-report/#7d05734d452b
76. https://insidebigdata.com/2017/05/05/hidden-costs-bad-data/
77. http://www.ibmbigdatahub.com/infographic/four-vs-big-data
78. https://www.statista.com/statistics/263591/gross-domestic-product-gdp-of-the-united-states/
79. Error Cost Escalation Through the Project Life Cycle, Stecklein, et al, 2004 https://ntrs.nasa.gov/search.jsp?R=20100036670
80. Krasner, H. *Ensuring E-Business Success by Learning from ERP Failures*, IT Professional Magazine, Jan./Feb. 2000
81. https://www.eetimes.com/author.asp?section_id=36&doc_id=1330462
82. https://www.bernsteincrisismanagement.com/hard-stats-on-the-cost-of-information-crises/
83. Texas OAG T2 project in the news - https://www.statesman.com/news/state--regional-govt--politics/attorney-general-tech-project-200-million-over-budget/PVxJjXzobcGk9cZyL4EVEN/