Thirty Years of Software Problems in the News

Andrew J. Ko
The Information School
University of Washington, Seattle, USA
ajko@uw.edu

Bryan Dosono
School of Information Studies
Syracuse University, USA
bdosono@syr.edu

Neeraja Duriseti
AT&T, USA
nduriseti@yahoo.com

ABSTRACT
How have the consequences of software problems changed over the past 30 years? To begin to answer this question, we analyzed 386,381 news articles reporting on software problems published between 1980 and 2012, spanning widely circulated newspapers to small trade magazines. Our results show that after an increase in reporting just prior to Y2K, news on software problems has declined in North America, but increased in the rest of the world. Most articles only report minor consequences such as frustration, confusion, anger, or at worst, having to delay some activity for a few hours, usually due to service outages in government, transportation, finance, and information services. However, about once per month, the news reports at least one death, injury, or threatened access to food or shelter due to software problems. Reports of these severe consequences are also increasing, due primarily to stories about transportation and government software.

Categories and Subject Descriptors
K.4.1 [**Computers and Society**]: Public policy issues — abuse and crime, human safety.

General Terms
Human Factors, Reliability.

Keywords
Software failures, news, news clustering.

1. INTRODUCTION
The ubiquity of software has transformed society in innumerable ways, reshaping globalization [2], social relationships [29], work [21], social identity [11,1,7], and countless other aspects of society. With these changes, however, have also come a dramatic shift in how the society works: behind much of our modern infrastructure is now the automated digital logic of algorithms, often supplementing or even replacing the slow subjectivity of human decision making.

Such automation naturally has tradeoffs: it can greatly increase productivity and lower costs, but it can also fail spectacularly [3]. Most water, gas, and electricity failures now involve software failures [24], stock markets can now crash in a flash [27], race conditions can leave millions without power [31], and usability problems can cost lives [16]. From a productivity perspective, software problems can also be quite expensive: one estimate of the annual costs in lost time and labor due to software problems is nearly $40 billion per year [26] and companies like Microsoft regularly spend hundreds of millions of dollars removing defects from a single release of Windows [20].

CHASE'14, June 2 – June 3, 2014, Hyderabad, India

Studies show that software problems in particular industries are also increasing in frequency and severity over time as the causes of system failures have shifted from hardware to software [8]. Security breaches are now more prevalent and more costly than ever before [7,10]; spreadsheet errors are leading to an increasing number of financial calamities [23]; software infrastructure failures rose between 1994 and 2005 [24]; and cars are recalled more frequently than ever because of software defects [18].

While these studies teach us much about how the consequences of software failure have changed in specific industries, they leave open many questions about the broader impact of software problems on human experience. Are software problems in government, education, and other industries experiencing from similar increases in software problems? Are other regions of the world suffering from software problems in the same way that North America has? What consequences, besides death and financial loss, have software problems posed to society? Answers to these questions are critical to guiding and prioritizing software engineering research and practice. They are also important as a record of the past 30 years of digitization of modern society.

No one study can hope to answer these questions definitely. In this paper, we begin to understand answer them at the macro scale, investigating the trends in software problems and their consequences by mining archives of failures. Perhaps the most extensive archive of software problems and their consequences is *news* archives. News is probably the largest record of the acute effects of software failure on society. Of course, news is inevitably biased in many ways: it only reports on a small subset of the software problems that occur in the world, it only covers problems that journalists become aware of, and only then focusing on problems that are "newsworthy" in some way. Journalists also tend to speculate about the causes of software problems, reporting little about technical and human causes underlying a failure [12].

With these caveats in mind, we present an analysis of 386,381 news articles, pulled from news sources both large and small in circulation. We explore a broad notion of software problems that encompasses both defects that led to failures, but also requirements defects, and the fear of categories of defects that might eventually lead to consequences (such as Y2K defects). Through both quantitative and qualitative methods, we investigate the problems reported, the consequences they have had on people and society, and how the consequences reported in the news have changed over the last 30 years.

2. METHOD
The focus of our analysis was on software problems, which we define as one of two things: (1) events in which a software system behaved in some problematic way or (2) defects that people feared would eventually cause some problematic behavior. This broad notion of software problems includes functional defects, usability problems, security flaws, requirements defects, and any other type of undesirable behavior of a software system. We use the word "problem" instead of "failure" since software behavior can be problematic even when it works as intended [4,13,14]. We

Table 1. Terms considered for retrieving articles. Based on a random sample of 260 articles for each term from 1980-2012. Only ProQuest counts were included because it was the only system that provided search result counts for large queries.

term	% about software	# articles with term	# about software	dominant topics
glitch	60%	49,537	~30,000	software problems
bug	11%	182,150	~20,000	insects, illness
malfunction	8%	24,154	~2,000	mechanical problems
defect	3%	92,300	~3,000	taxes, birth defects
flaw	3%	106,240	~3,000	construction, decisions
error	2%	629,832	~12,000	sports, news, medicine
mistake	1%	586,720	~6,000	decision making
fault	1%	265,383	~3,000	politics

explicitly do not use the words *failure*, *fault* and *error*, as they refer to details that few journalists can observe.

We retrieved news from ProQuest, LexisNexis, and Factiva, the three major news archives. Our focus was on English news, as evidence shows that the majority of published news articles is written in English [5], but we included all digitally available news sources from around the globe reporting on events that occurred at local, regional, national, and international levels. Using each of these database's metadata services, we determined that these databases included full text articles for 25,181 news sources. These databases included full text articles for 93 of the top 100 newspapers in the United States (based on data from the Audit Bureau of Circulations, http://abcas3.accessabc.com/ecirc/, retrieved March 31st, 2012). Of the 25 most circulated news sources in the US, 23 were available in full text from 1995 and on, and 14 from 1990 and on. The archives also included thousands of trade magazines for specific domains (e.g., Aviation Safety, Banking and Credit News, Health Data Management).

To retrieve articles about software problems, we began by searching for words commonly referring to software problems, specifically searching for bug, defect, error, fault, flaw, malfunction, glitch and mistake. We selected a random sample of search results for each of these terms to assess their viability, randomly selecting one article per quarter, per database, per term, from 1980 to 2012, for a total of 260 articles per term (2,080 articles overall). We then analyzed each article for whether it referred to a software problem or some other type of problem. As seen in Table 1, the only phrase that consistently referred to software problems was glitch. Only 11% of articles with the word bug and 9% of articles with the word malfunction referred to software: across the sources we sampled, these terms typically referred to insects, illness and mechanical failures.

The only phrase with a non-negligible number of articles with uses referring to software was bug. To better understand how excluding these articles would bias our analysis, we investigated usage of the term, finding that 65% of the uses of bug described the "Y2K bug" or "millennium bug", that 30% of these articles also used the word glitch, and that 96% of uses of the word "bug" to refer to software problems occurred before 2000 (which, as we show later, was less than 10% of our data). We therefore decided to focus only on articles with the word glitch.

There were a total of 386,381 articles published between January 1980 and July 2012 containing the word glitch, including duplicates. We manually grouped news sources titles that referred to the same news source with small variations in formatting, casing, and subsection titles. The resulting set included 6,882 unique news sources. This data set may represent one of the most comprehensive samples of stories about software problems in history, including the RISKS Forum (*http://catless.ncl.ac.uk/Risks/*), which began archiving software problems in 1985.

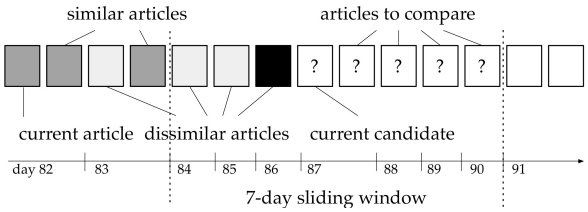

Figure 1. We clustered news articles incrementally using standard TF-IDF cosine similarity and a sliding 7-day window.

2.1. Clustering Articles into Stories

Because our unit of analysis was software problems and not an individual article, we needed to group articles according to the problems on which they reported. While there are many news clustering techniques, all were inappropriate for our analysis: k-means clustering on a word vector space and Latent Dirichlet Allocation, used both in automated news clustering approaches [9,6,30] and studies of blogs [18], require a specific number of topics to be selected a priori, but we had no basis for choosing such a number. These techniques also ignore time: stories in the 1980s should not be clustered with problems from the 2000s just because they share a topic.

We developed a new algorithm to overcome these issues (Figure 1). We converted each article's words into a vector of tf-idf scores [32] and then iterated through all articles in order of publication date, comparing each article to all subsequent articles published within a 7-day window. If the cosine similarity of any two article's tf-idf term vector was .25 or greater (representing a modest similarity), then an edge was created between the two articles, creating a graph of related articles. When a similar article was found, the comparison window was then set to 7 days ahead of the date of the new relevant article. This allowed articles to be linked across weeks, months, and years, as long as at least one similar article was published within a week. After comparisons for each article were complete for a day, all disjoint graph components of the article graph whose latest dated article was more than 7 days before the current date were removed from the graph and treated as a story cluster. We chose 7 days because of the weekly news cycle.

"Briefings," "Headlines," and other composite articles that reported on multiple events were problematic, as they contained relevant words for multiple clusters, they created edges between multiple distinct clusters that reported on different problems. We therefore decided to exclude these composite articles, detecting them with a set of text formatting patterns [6]. Our filters excluded any story with multiple full caps headers, sequences of bulleted paragraphs, recurring paragraph prefixes such as Twitter usernames, time and date stamp paragraph prefixes that tended to report on distinct stories, and long sequences of dashes or periods that segmented distinct stories. The false positive rate of these heuristics was less than 1%.

The application of our algorithm grouped the 386,381 articles into 58,577 story clusters. In the rest of this paper, article will refer to a single news report from a single news organization, whereas story will refer to the set of articles identified by our clustering algorithm as reporting on the same software problem. To assess the accuracy of clustering, the authors independently read all articles in a random sample of 550 story clusters spanning 6,935 articles, judging each article as either relevant to the cluster or off-topic. These relevance judgments had a Krippendorff's alpha of 0.59, indicating that the clustering was moderately accurate: of these coded stories, 80% contained 100% relevant articles; the mean proportion of relevant articles was 92%.

2.2. Story Measurements

We measured several aspects each story. To select a date to represent when a story cluster's problem occurred, we chose the earliest published date of all articles in the set. We also measured the attention a story was given by the press by computing *normalized article count*, which was the number of articles in a story cluster, excluding duplicates published in different sources, divided by the number of news sources that were available in the news databases in the year the story was published. This measured how many news organizations committed effort to reporting on the story, while controlling for the confounding factor of full text availability over time.

To measure the geographical location from which a story was reported, we began by assigning a world region, country, and state (for US stories) to each news source that appeared more than 5 times in our data set. The world regions we used were North America, Europe, Australia, Asia, India, Africa, Latin America, the Middle East, South America, and Russia. To assign a region, country, and state to a news source, we performed research on each of the 6,882 source's current editorial headquarters. If we could not find the editorial headquarters of a news source, it was labeled "unknown"; this was typically the case for blogs, print magazines, obscure newsletters, and other exclusively online news sources. Using the locations of news sources, we then analyzed the frequency of all news sources' locations in a story, selecting the most frequent region, country, and state.

All other aspects of stories were qualitative measurements, which we obtained by coding a sample of story clusters and their articles using content analysis methods [22,25]. Our sample included the subset of the 58,577 story clusters with 6 or more articles and a random sample of stories with fewer articles, to focus our analysis on the stories that received the most attention from the press. This sample included 6,705 stories (11% of story clusters) spanning 130,189 of the retrieved articles (34% of all articles). Reading these articles spanned about 1,000 person-hours across 2 months.

Table 2 shows the variables that we coded. We first coded whether the story was about a problem with a software-based system or something else (this was the same analysis we did to select search terms, as in Table 1). We counted three types of stories as about software problems: reports on 1) a problem that was or could have been caused by a software or requirements defect, 2) an expected software problem that did not happen, or 3) a defect that might later cause a problem. All other stories were excluded.

We also measured the consequences reported in the story. There are many aspects of consequences that might be measured about software problems, especially their indirect effects, these are difficult to measure using news articles, as articles rarely provide information about the frequency or prevalence of a problem. Instead, we focused on a categorical measurement of the consequences of a problem to human needs leveraging frameworks of universal human needs, such as Maslow's hierarchy [19] and its more modern counterparts [28]. As shown in Table 2, we selected from these needs five major consequences to code individually: death, physical harm, loss of access to food or shelter, a permanent loss of property (physical goods, money, or data), and forced delay in some activity. Our measurement therefore says little about the cumulative consequences of defects, but rather the qualitative kind of consequence that software problem events imposed on people. We considered other consequences to human needs, such social status, love, achievement, and autonomy [28] and emotional consequences such as fear and frustration, but these were either too rare, could not be coded reliably, or were not reported reliably by journalists.

Table 2. Codes used to classify each story. Krippendorf's alphas were computed across three independent raters.

code	description	alpha
software	**true** if a system failure occurred that was or could have been caused by a software defect.	0.79
death	true if an article indicated that one or more people **lost their lives** due partly to a software problem.	1.00
harm	true if an article indicated that one or more people were **physically harmed** due partly to the software problem.	1.00
basic	true if an article indicated that one or more people **lost access to food or shelter** because of the problem.	1.00
property	true if an article indicated that one or more people **permanently lost material goods, money, or data** due partly to the problem.	0.87
delay	true if an article indicated that one or more people had to **postpone an activity** due partly to the software problem.	0.69
industry	**information**. Production and distribution of information. **transportation**. Moving people and things. **natural resources**. Extracting materials from Earth. **sales**. Exchanging money for products. **construction**. Creating built environment. **manufacturing**. Creating products from materials **utilities**. Power, gas, steam, water, and sewage services. **finance**. Manipulating and moving money for profit. **knowledge**. Education, research, and space exploration. **health**. Healthcare, health insurance, and food industries. **entertainment**. Arts, sports, hospitality, tourism, etc. **government**. Politics, defense, justice, taxes, public services, etc. **multiple**. More than one industry was affected.	0.79

The last story variable that we coded was the industry in which the software problem occurred. We derived this measurement from the North American Industry Classification System (*http://www.census.gov/eos/www/naics/*, retrieved May 1st, 2012), maintained by the US Department of Commerce. Our adaptation included the 12 industries in the last part of Table 2 and a "multiple" category for problems that affected multiple industries. Our rule for selecting an industry was to choose the industry that the misbehaving software was intended to support. For example, problems with federal tax return software was coded as government and not finance because the software was intended to support government tax activity and not finance activities such as savings or investment.

To assess the reliability of each variable in Table 2, the authors iteratively and redundantly coded random samples of 100 stories, each time computing Krippendorff's alpha [15] for each variable and discussing disagreements. After three rounds of redundant coding (covering 5% of all story clusters), we reached satisfactory agreement on all variables. The authors then split this sample into three sets and coded each story cluster independently, reading each article in a cluster. For each cluster with more than 100 articles, we read a random sample of at least 10% of stories to identify the dominant story topic.

Of the 6,705 stories that we coded, only 3,977 (59%) concerned software problems. This suggests that only about 35,000 stories of the original 58,577 concerned problematic software behavior. However, since we do not know which of stories concerned software problems and which did not, we used all 58,577 stories for our quantitative measures, assuming the out of scope stories were randomly distributed over time, country, and industry.

Table 3. Top 10 stories by normalized article count.

the software problem reported	year	# articles
The aftermath of the year 2000 millennium bug	2000	17,193
E-voting defects in the 2004 US elections	2004	2,509
Preparations for the year 2000 millennium bug	1998	1,745
Medicare drug eligibility defect	2006	1,983
Toyota Prius sudden acceleration bug	2010	3,782
Delayed e-voting tallies in the 2008 US elections	2008	1,796
Incorrect voting tallies from e-voting machines	2006	1,236
Incorrect New Mexico voting tally	2000	1,003
Atlanta Olympics results delivery to news delayed	1996	447
E-voting defects in the 2002 US mid-term elections	2002	636

3. RESULTS

In this section, we explore trends in the topics, and consequences of software problems in the news. For statistical tests, α=.05. We analyzed all categorical variables with non-parametric chi-squared tests and logistic regressions, using Bonferroni corrections for all post-hoc tests. We report odds ratios from logistic regressions of categorical variables such as country, consequence and industry against time (odds ratios are commonly used measures of effect size, measuring the ratio between the odds of an event occurring in one group to the odds of the same event occurring in a different group. Odds ratios above 1 indicate an increase in the odds of a problem over time and odds ratios below 1 indicate a decrease).

3.1. Notable Story Topics

Journalists have reported a variety of problems and consequences in the past 30 years. English pensioners have been mistakenly marked as dead, halting their benefits; defective breathalyzer source code nearly sent a man to prison; sexual assault victims have received automated calls mistakenly telling them that their attackers had been released. Toddlers appeared on the US Transportation Security Administration's "no fly" lists; firefighters went to the wrong houses because of defects in GPS routing algorithms; Kaiser Permanente, a US health care provider, mislabeled prescription drugs for months, threatening the lives of thousands of patients. The US Army sent letters to the families of dead soldiers with the salutation "Dear John Doe"; tens of thousands of people were locked out of their Facebook accounts for having "inauthentic" names; and 350 Rhode Island residents were mistakenly arrested because of incorrect names on arrest warrants. Stories like these illustrate that many of the most stories reflected mismatches between reality and models of reality.

While the notable stories varied greatly in their consequences, the top 100 stories receiving the most press (as measured by normalized article count of each story) were quite homogenous (Table 3 shows the top 10). The biggest story was by far the Y2K millennium defects, an issue that afflicted software that only used two digits to represent the year. Two clusters of Y2K stories emerged: one including over 17,000 articles after January 1st, 2000, covering hundreds of industries that did and did not experience problems. The second included 1,745 stories covering the fears of Y2K problems and its many potential consequences, including costs to fixing them and stock market anxiety about the potential costs and consequences of Y2K failures. Another 13 stories covered fears and consequences of problems with e-voting software, spanning the decade from 1998 to 2008: tallies were delayed and incorrect, machines froze, digital voter registration records were missing names, and votes were lost.

There were several other recurring problems in the top 100; 12 described software problems that delayed NASA shuttle launches, Mir space station missions, and Mars robot missions. Another 10 stories reported problems with airport computer systems, leading to delayed and canceled flights or damaged and lost luggage. Yet another 10 covered stock market outages in the US, UK, Japan, and Australia (including the recent Facebook IPO NASDAQ problems recently dominating news). Other recurring topics concerned phone and e-mail outages, the Olympics, problems on the first day of school, and tax refund delays. Some stories covered culturally significant problems such as a popular Formula 1 racer who lost because of a friction control system defect in 2001, and an accidental censorship of LGBT media at Amazon.

Most of the consequences reported in these top 100 stories were delay and property loss: 66 stories involved people being forced to wait to vote, launch, fly, trade, shop, etc.; according to the reporters, most of these delays were hours long. In 16 stories, the delay also led to property loss, such as extra costs, lost business, or lost data. Only one story reported death and physical harm (the Toyota acceleration defect) and only one reported a threat to basic needs (a weeklong banking outage in Australia that left many of its 11 million customers without access to cash). The Y2K defect famously had few notable consequences. The most heavily reported stories, therefore, were often more curious in their consequences than they were consequential.

3.2. Reporting Frequency of Problems

Figure 2 shows the number of stories reported per month, separated by region. As the plot shows, there were few problems reported in the 1980s, with a rapid rise in the number of stories approaching the year 2000. After the year 2000, however, the frequency of reporting appears to have slowly declined overall. As shown in the bottom of Figure 2, this trend does not appear to be influenced by full text availability, which has only increased.

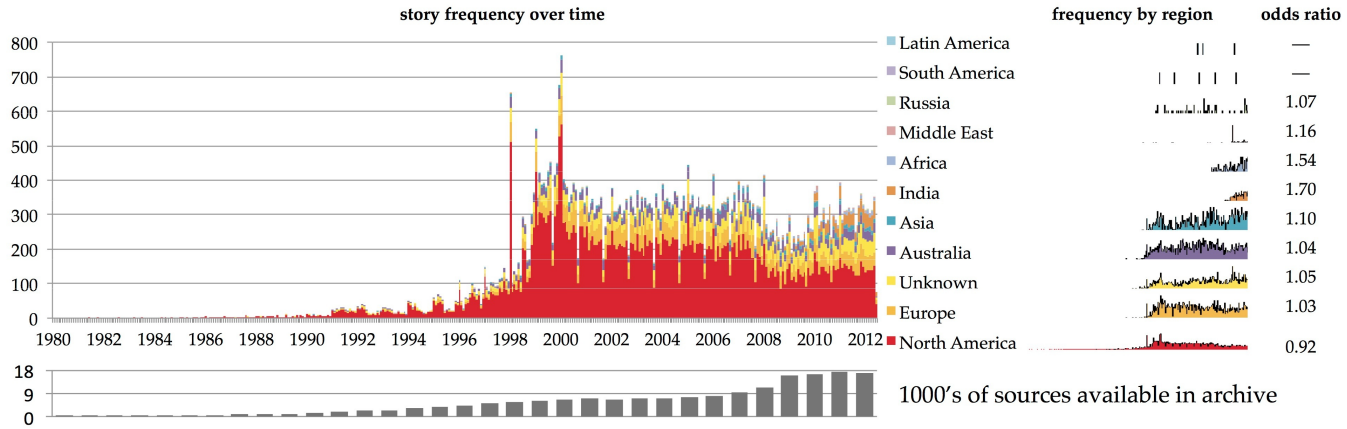

Figure 2. Left: story count each month from Jan. 1980–June 2012, by region. Right: story count for each region over time and odds ratios of significant logistic regressions for stories in a region by time. Bottom: stories available in the news archives over time.

Table 4. Problem frequency by industry. Odds ratios are from logistic regressions of whether a story was in a specific domain versus year. Bold values are significant.

industry	% of stories	frequency by year	odds ratio
government	22%		1.00
information	16%		0.97
finance	14%		1.00
entertainment	12%		**1.03**
transportation	11%		**1.05**
knowledge	10%		**0.94**
sales	4%		**1.04**
health	3%		**1.06**
utilities	3%		**1.05**
multiple industries	3%		**0.85**
manufacturing	2%		1.00
construction	1%		1.00
natural resources	1%		1.06

Most of the stories in the data set were published in North America (63% of stories), and that within the US, the most active states included the New York (29% of articles), Pennsylvania (7%), Florida (7%), California (6%), and DC (6%). Europe (13%) and Australia (8%). The data did include stories from most countries, except for several African and middle eastern countries. Interestingly, however, as seen on the right of Figure 2, North American stories are declining in frequency, ($\chi 2(1, N=58{,}576)=2260.8$, $p<.001$, odds ratio=.916) from several hundred per month to only about 100 per month. While North American stories have declined, there have also been significant increases in the likelihood of software problem stories in nearly every other region of the world, especially Asia, India, and Africa.

In interpreting these trends, it is important to remember that declines in stories in North America may represent declines in US journalism budgets, as opposed to declines in software problem frequency. To investigate this possibility, we considered the 2,182 unique news sources represented in the stories and found that just 190 sources were responsible for 80% of the articles written. Of the top four of these based on circulation (The Associated Press, The McClatchy Company, The Washington Post, and The New York Times) all have declined in the number of stories about software problems since 2000, except for McClatchy, which has increased. This suggests that even the most well-resourced news organizations in the world are still reporting on fewer software problems now than in past decades.

As shown in Table 4, most of the stories reported on problems in government, IT, finance, entertainment, transportation, and knowledge industries (science and education). These trends are changing, however: problems are now more likely to be reported in transportation, entertainment, health, sales, and utilities.

3.3. The Consequences of Software Problems

Figure 3 shows the proportion of stories reporting each of the consequences that we investigated and the proportion of stories that reported none of these five. In this section, we discuss each of these five consequences, as well as the stories that lacked these consequences, presenting qualitative characterizations of when, where, and why software problems were consequential.

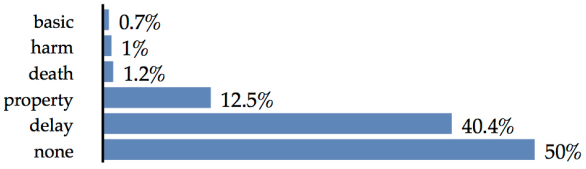

Figure 3. Proportion of stories reporting each consequence.

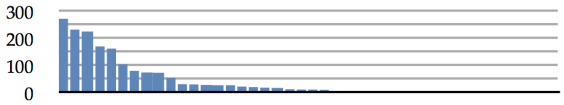

Figure 4. Death counts reported in 47 stories.

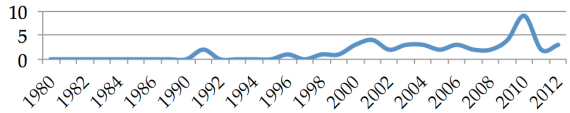

Figure 5. Frequency of stories reporting death.

3.3.1. Reports of Death

Death was rare. Among our coded sample, 47 stories (1.2%) reported one or more deaths related to software. Extrapolating this to the ~35,000 stories in our data, about 400 stories over 32 years likely reported deaths (about 1 story per month worldwide). Of these, 77% were reported in North America, with the others reported in Europe, India, and Australia. As shown in Figure 4, the minimum number of deaths reported in the stories ranged from 1 to 270, with a median of 8, though three of the stories did not report death counts. This is a total of at least 1,691 deaths across 47 stories; extrapolated to all ~35,000 stories, the number of reported deaths blamed partly on software since 1980 is likely about 15,000 (or about 40 people per month).

A third of reported deaths were due to collisions in planes and cars, with most plane crashes accounting for most death counts above 50. On land, transit lines and railroads that used the same signaling system have repeatedly failed to detect trains traveling the wrong way. Toyota recalled its 2010 Hybrid Prius after discovering a design flaw that required a software change to fix the anti lock brake-system problem; the LA times reported in March 2010 that 102 people had died from this and other problems with sudden acceleration. In the air, plane crashes stemmed from pilots receiving faulty readings on cockpit screens, engines and thrusters being reversed mid-flight, and air traffic controllers guiding dangerous landings.

Explosions occurred in approximately a tenth of the stories. For example, a computer program that controlled a Russian nuclear submarine's fire suppression system had malfunctioned, killing 20 people in 2008. In 1991, a requirements problem in radar detection led to the deaths of 28 US soldiers when an unexpected scud missile struck an American base near Saudi Arabia.

There were some instances of death in health care and natural disasters. In 1999, software failed to capture information about the whereabouts of a missing Alzheimer's patient when the patient crossed the state border and was not seen again. In 2010, one hospital's faulty bar-coding system deceived a nurse; she mistook a bag of epidural painkiller for penicillin and hooked it up to an IV line that pumped the painkiller into the bloodstream of a patient who was in the middle of delivering a baby. The baby survived, but the mother's heart collapsed. Several stories reported over-dependence on software-based early warning systems. One was a 2005 software problem that disabled tornado warning broadcasts in Paducah, Kentucky. The other was a 2012 malfunction with a warning system during the Colorado wildfires. In both cases, residents had relied exclusively on the automated phone calls to know when to evacuate, but did not receive them, preventing them from evacuating.

Figure 5 shows that the proportion of stories reporting deaths has risen since 1980, but not significantly ($\chi 2(1, N=3{,}977)=2.1$, $p>.05$). Deaths were more likely in government ($\chi 2(12, N=3{,}977)=62.1$, $p<.0001$), typically during disaster response, and transportation. Deaths were also overrepresented in India and the Middle East ($\chi 2(9, N=3{,}977)=20.2$, $p<.05$).

Figure 6. Distribution of injury counts (in decreasing order).

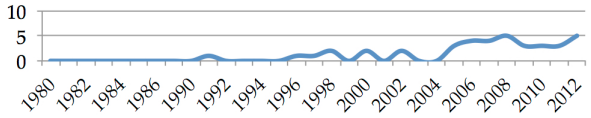

Figure 7. Frequency of stories reporting physical harm.

3.3.2. Reports of Physical Harm

Injury was also rare. In our sample, 39 stories (1.0%) reported physical harm, which suggests that around 350 stories of the ~35,000 reported harm (at a rate of about 1 story per month). Of these, 77% were reported in North America, with the others reported in Europe, Asia, and India. Fewer than half of the stories reported the number injured; of those that did, the number of injuries ranged from 1 to 270, with a median of 19 (as shown in Figure 6), for a total of 1,064 injured across 17 stories. Extrapolating this rate to all 35,000 stories, the number of injured since 1980 is likely about 12,500 (or about 30 people per month).

Half of the stories reported injuries sustained on boats and ships because of bumps and tilts. In 1997, autopilot problems led a double-decker tourist sightseeing boat to ram into a bridge on the Seine River in Paris, injuring 28 people. A similar problem with the autopilot system on a Crown Princess cruise in 2006 caused the month-old ship to tilt harshly to one side, harming 240 passengers. The tilt was so extreme that even the casino's slot machines and the gymnasium's exercise equipment tipped over or slid across the floor, causing serious injury to 20.

Some stories reported injuries in entertainment. In 1998, authorities blamed software for an automated fireworks display that fired all fireworks at once; two technicians were hospitalized with burns. Other stories reported injuries in entertainment, health, and manufacturing. On Broadway, a Spiderman stunt actor was injured after plunging 30 feet due to a computer-controlled harness failure. In 2008, a big television screen at the Piccadilly Gardens fan zone failed to show a highly anticipated football game as planned, resulting in thousands of fans violently rioting against the police, leading to injuries in both parties.

Software related injuries in healthcare were rarely reported. A problem in the government's new computerized finance system between pharmaceutical clinics and the national health department led to drug shortages in the Virgin Islands, affecting dozens of people affected with HIV.

As seen in Figure 7, the likelihood of a story reporting harm has increased over time ($\chi2(1, N=3,977)=6.7$, $p<.05$, odds ratio 1.09). Harm was also more likely in transportation and significantly less likely in finance and information industries ($\chi2(12,N=3,977)=71.0$, $p<.0001$), but not more likely in any particular region of the world ($\chi2(9, N=3,977)=5.4$, $p>.05$).

3.3.3. Reports of Lost Access to Food and Shelter

Threats to food and shelter were even less common than death and physical harm, appearing in only 28 of coded stories (.7%). This suggests that about 250 stories across the 32 years involved a threat to basic needs (at a rate of about 1 story every two months). Of these, 54% were reported in North America, with the others reported in Europe, Australia, and India. Of the 28 stories, 17 reported the number of people affected; as shown in Figure 8 counts, these counts ranged from 1 to 2 million, with a median of 55,000 and a total of 5,451,582 people across 17 stories.

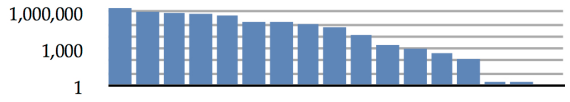

Figure 8. Number who lost access to food or shelter, based on 17 stories reporting head counts, on a logarithmic scale.

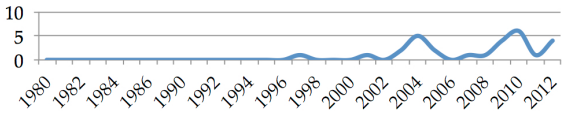

Figure 9. Frequency of threats to basic needs over time.

Extrapolating this to all 35,000 stories, the number of people reported to have lost access to food and/or shelter since 1980 is likely to be about 80 million (30 per month).

A third of stories involved problems that resulted in the delay of government aid to vulnerable populations. Some involved welfare recipients that qualified for food stamps and students waiting for financial aid that was incorrectly denied or backlogged. Five stories reported on faulty natural disaster detection systems, including overlooked tornado warnings, missed wildfire notification dispatches, and 911 emergency calls that did not reach an operator. Errors in these citywide emergency systems have been reported periodically from 2003 to 2012. One other common threat to basic needs was ATM and banking outages, preventing people without credit cards from acquiring cash to buy food. Most stories represented single points of failure.

As seen in Figure 9, the likelihood of a story reporting a threat to basic needs has increased significantly over time ($\chi2(1, N=3,977)=12.1$, $p<.001$, odds ratio 1.16). There was again a relationship between basic needs threats and industry, with basic needs were more likely to be threatened in government (50% of stories concerning basic needs) and utilities (25% of stories). Threats to basic needs were related to region ($\chi2(9, N=3,977)=18.6$, $p<.05$), with basic needs losses overrepresented in the US.

3.3.4. Reports of Property Loss

The loss of money, data, and material goods was much more common, reported in 498 of coded stories (12.5%), and therefore about 4,400 stories over the 32 years. Of these, 68% were reported in North America, with the others reported primarily in Europe (17%) and Australia (9%).

There were three dominant types of property loss. About 20% of stories reported data loss, often e-mails, court documents, or data that NASA probes failed to transmit. For example, in 2001, millions of White House e-mails could not be recovered from backup tape due to an archiving defect, preventing investigation of an obstruction to justice case. Another 20% were stories about companies losing revenue because customers had problems with the company's software (e.g., a *buy.com* coupon led to accidental discounts of a quarter million dollars). In other cases, airlines

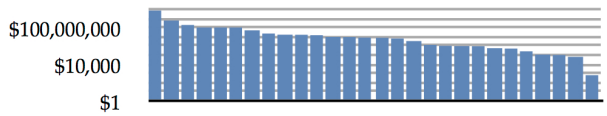

Figure 10. Money lost due to problems on a random sample of 28 stories, on a logarithmic scale (in decreasing order).

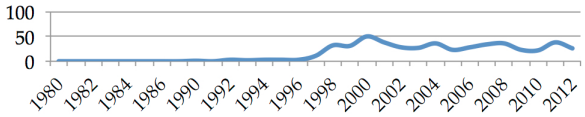

Figure 11. Frequency of property loss over time.

blamed outages for revenue losses. Another 20% reported the costs of additional software testing; for example, a Pennsylvania county invested in a 911 response system, but had to delay its launch and pay for further testing. The other 40% of property losses included incorrect charges that were never reimbursed, regulatory fines due to software defects, stock market loss, and security breaches.

While it was difficulty to quantify the loss of data and physical goods, we were able to quantify loss money, as many articles explicitly reported it. Of a random sample of 100 stories, 28 reported an amount lost due to the software problem. We converted all of these amounts to USD using the average exchange rate for the month the story was published. As seen in Figure 10, the resulting range was from $625 to $7 billion, with a median of $7.7 million lost.

As seen in Figure 11, property loss has not significantly changed in likelihood over time ($\chi 2(1, N=3,977)=0.1$, $p>.05$). Property loss was associated with industry ($\chi 2(12, N=3,977)=97.6$, $p<.0001$), with health, manufacturing, sales, transportation, and utilities industries significantly more likely to involve loss. Property loss was significantly related to region ($\chi 2(8, N=3,977)=21.1$, $p<.05$), with loss significantly overrepresented in Australia and Europe.

3.3.5. Reports of Delay

Delay was the most common consequence, mentioned in 40.4% (14,160 stories). Classifying the topics in a sample of 160 of these revealed several recurring delays. The most common were outages in government services, forcing people to wait days (13%). Software delays were common and typically reported on widely publicized projects such as new government websites or utility services (12%). Other common types of delays occurred due to outages in stock market trading (11%), banks and ATMs (9%), flights (9%), voting and elections (7%), NASA launches (6%), phone service (6%), public transportation (4%), and web sites (4%). The shortest delays concerned emergency responses to fires, injuries, and other time-critical services, where a few minutes of delay was a life and death matter.

People rarely waited long. In a random sample of 160 of these stories, 134 stories explicitly indicated a time scale of delay. Figure 12 shows this distribution, revealing a typical delay of hours or days; the delays of months and years were largely software release delays.

As seen in Figure 13, the likelihood of a story reporting delay has not changed significantly since the late nineties ($\chi 2(1, N=3,977)=3.1$, $p>.05$). There was a relationship between delay and industry ($\chi 2(12, N=3,977)=197.1$, $p<.0001$), with the finance, knowledge, natural resources and transportation industries significantly more likely to suffer delay. Reporting on delays were overrepresented in Asia, Europe, and Russia and underrepresented in North America ($\chi 2(9, N=3,977)=29.2$, $p<.001$).

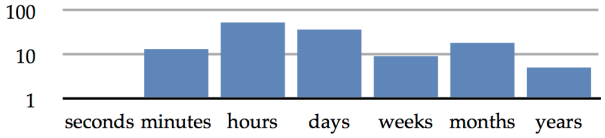

Figure 12. Time lost due to software problems on a random sample of 134 stories explicitly indicating a time scale of delay.

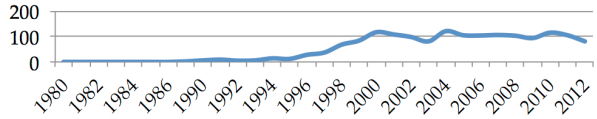

Figure 13. Frequency of delays over time.

4. DISCUSSION AND CONCLUSION

We began this paper by observing that critical software problems in security [7,10], financial calamities [23], software-based infrastructure [24], and automotive recalls [18] are on the rise. Our findings provide a broader picture for interpreting these domain-specific trends:

- Worldwide, news stories on software problems increased substantially in the 1990s, but reached a peak in 2000 and have declined since. Most of this decline is due to reporting declines in North America, with the rest of the world, particularly Asia, India, and Africa, increasing reporting on software problems.
- The problems that have received the most attention by the news tend to involve voting problems, NASA failures, stock market outages, Y2K defects and a diverse array of culturally relevant, but essentially minor defects that generally led to brief delays.
- About once per month on average, the news reports death, physical harm, or threatened access to food or shelter due partly to software problems, for an estimated 32 year cumulative total of 15,000 reported deaths, 12,500 reported injuries, and 80 million people reported to have temporarily lost access to food or shelter. The rate at which reports of physical harm and lost access to food and shelter occurs is increasing slowly, primarily due to failures in transportation and government software.

The central question in interpreting these results is whether news is actually a representative sample of software problems. News, after all, is an imperfect and inevitably skewed sample of the software problems that occur in the world. It is likely *not* a reliable indicator of the frequency and prevalence of problems that occur. Our decision to focus only on news reports that contain the word glitch may have overlooked problems with privacy or security, as those stories tend to not use the world glitch. Our focus on English news also limits our results' generalizability.

There are some reasons to suspect, however, that the trends in our data are to be believed. After all, software testing of all forms has matured in the past 30 years, and our data show that news reports of problems are increasing in every country in our data set except for North America and Europe, two of a few regions where software development practices are arguably most mature. Our data also show that some of the most critical consequences—physical harm and loss of basic needs—are on the rise, but generally only in countries that are introducing *new* kinds of software infrastructure. This interpretation of the data relies on the assumption that it is the novelty of software systems to an industry and the industries maturity in software engineering practices that determines the frequency of critical failures.

Another interpretation of our data is that software problems are just as or more frequent as they have ever been, but journalists have lost interest in reporting them. Perhaps software problems were surprising in the 1990s because fewer people had experiences with how software fails, making any problem, no matter how minor, a newsworthy story. Now that most people have experience with computers, people (and journalists) may see software problems as routine, making them less newsworthy. The increases in reported problems in Asia, India, and Africa, may therefore be the same novelty effect that North America experienced in the 1990's.

A different, but similarly pessimistic interpretation of our results is that journalists are still interested in reporting on software problems, but that they are simply less willing or able to report on them. For example, the only way for journalists to know that a problem occurred is for some person or organization to report it. It

may therefore be that the institutions who have an obligation for transparency—government, publicly traded companies, etc.—are the only ones visible to journalists, and that problems are increasingly common inside of organizations who have no need to announce them to the world. Private organizations may simply be getting better at hiding the problems that occur from the press.

What do these results and interpretations mean for software engineering research and practice? One implication is that, because our results showed that it was typically new software systems that were responsible for most of the news on software problems, we should think critically about which industries and which populations can safely take the risk of migrating to new software systems. Consumer markets might be able to tolerate constant device upgrades and continuous improvements to websites, but as critical infrastructures begin to depend on these consumer applications (e.g., the shift toward "bring your own device policies" in organizations), we may risk bringing increased failure rates into stable (if aging) information systems.

Our results also have many implications for society more broadly, and specifically for society's expectations and understanding of software. For example, our results show that many of the consequences reported in the news are not due directly to software, but to the expectations of software. This suggests that part of mitigating software risks is not only education for engineers about better practices to prevent failure, but also better education for society, calibrating expectations about software reliability and correctness. For example, one could imagine computing literacy courses that explain common ideas in risk concepts such as *single point of failure*, so that people who create processes that rely on software are more aware of the need to anticipate and plan for failure.

Whatever the solution, our results show that software problems will be part of modern society, and increasingly in the developing world, for the foreseeable future. Research on how to design software systems that avoid failure not just in technical ways, but sociotechnical ways, has never been more important or timely.

5. REFERENCES

[1] Acquisti, A., Friedman, A., & Telang, R. (2006). Is There a Cost To Privacy Breaches? An Event Study. *Int'l Conf. on Info. Sys.*

[2] Castells, M. (2009). *The Information Age: Economy, Society, and Culture*. Wiley-Blackwell, 2nd edition.

[3] Charette, R.N. (2005). Why software fails. *IEEE Spectrum*.

[4] Chilana, P.K., Ko, A.J., & Wobbrock, J.O. (2010). Understanding Expressions of Unwanted Behaviors in Open Bug Reporting. *IEEE VL/HCC*, 203-206.

[5] Clausen, L. (2003). *Global News Production*. Copenhagen Business School Press, 1st edition.

[6] Clifton, C. & Cooley, R. (2000). TopCat: Data Mining for Topic Identification in a Text Corpus. *IEEE KDE*, 16(8), 1–33.

[7] Garrison, C.P. & Ncube, M. (2011). A Longitudinal Analysis of Data Breaches. *Information Management & Computer Security*, 19(4), 216-230.

[8] Gray, J. (1990). A Census of Tandem System Availability Between 1985 and 1990. *Tandem Technical Report 90.1*.

[9] He, Q., Chang, K., Lim, E.P., & Banerjee, A. (2009). Keep It Simple with Time: A Re-examination of Probabilistic Topic Detect Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(10), 1795-1808.

[10] Im, G.P. & Baskerville, R.L. (2005). A Longitudinal Study of Information System Threat Categories: The Enduring Problem of Human Error. *SIGMIS Database*, 36(4), 68-79.

[11] Jakobsson, M. & Myers, S. (2006). *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Wiley-Interscience, 1st edition.

[12] Johnson, C.W. (2003). Newspaper and Online News Reporting of Major Accidents: Concorde AFR 4590 in The Times, The Sun and BBC Online. *Univ. of Glasgow, Technical Report*.

[13] Ko, A.J. & Chilana, P. (2010). How Power Users Help and Hinder Open Bug Reporting. *ACM CHI*, 1665-1674.

[14] Ko, A. J. and Chilana, P.K. (2011). Design, Discussion, and Dissent in Open Bug Reports. *iConference*, 106-113.

[15] Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology*. Sage, 2nd ed.

[16] Levinson, N.G. & Turner, C.S. (1993). An Investigation of the Therac-25 Accidents. *IEEE Computer*, 26(7), 18-41.

[17] MacDuffie, J.P, & Fujimoto, T. (2010). Why Dinosaurs Will Keep Ruling the Auto Industry. *Harvard Business Review*.

[18] Mark, G., Bagdouri, M., Palen, L., Martin, J., Al-Ani, B., & Anderson, K. (2012). Blogs as a Collective War Diary. *ACM CSCW*, 37-60.

[19] Maslow, A.H. (1943). A Theory of Human Motivation. *Psychological Review*, 50(4), 370-96.

[20] Murphy, M. & Levidow, B. (2000). Windows 2000 Dependability. *Microsoft Research Technical Report*, MSR-TR-2000-56.

[21] Olson, J.M. & Olson, J.S. (2008). *The Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications*. Lawrence Erlbaum Associates.

[22] Patton, M.Q. (2002). *Qualitative Research and Evaluation Methods*. Sage, 3rd edition.

[23] Powell, S.G., Baker, K.R., Lawson, B. (2008). A Critical Review of the Literature on Spreadsheet Errors. *Decision Support Systems*, 46(1), 128-138.

[24] Rahman, H.A., Beznosov, K., & Martí, J.R. (2009). Identification of Sources of Failures and Their Propagation in Critical Infrastructures from 12 Years of Public Failure Reports. *Int'l J. of Critical Infrastructures*, 5(3), 220-244.

[25] Speed, J.G. (1893). Do Newspapers Now Give the News? *The Forum*, 15(1), 705-711.

[26] Tassey, G. (2002). The Economic Impacts of Inadequate Infrastructure for Software Testing. *National Institute for Standards and Technology*.

[27] Tamai, T. (2009). Social Impact of Information System Failures. *IEEE Computer*, 42(6), 2-3.

[28] Tay, L. & Diener, E. (2011). Needs and Subjective Well-Being Around the World. *J. of Personality and Social Psychology*, 101(2), 354–365.

[29] Turkle, S. (2011). *Alone Together: Why We Expect More from Technology and Less from Each Other*. Basic Books, 1st ed.

[30] Vadrevu, S., Teo, C.H., Rajan, S., Punera, K., Dom, B., Smola, A.J., Chang, Y., & Zheng, Z. (2011). Scalable Clustering of News Search Results. *ACM WSDM*, 675–684.

[31] White, D., Roschelle, A., Peterson, P., Schlissel, D., Biewald, B., & Steinhurst, W. (2003). The 2003 Blackout: Solutions that Won't Cost a Fortune. *The Electricity Journal*, 16(9), 43-53.

[32] Wu, H.C., Luk, R.W.P., Wong, K.F., & Kwok, K.L. (2008). Interpreting TF-IDF Term Weights as Making Relevance Decisions. *ACM TIS*, 26(3), Article 13.