Types of software failures

Frequently, software bugs serve as a motivation for computer scientists to learn, understand, and apply specific software engineering techniques. Certain literature uses failures to motivate approaches not only for the development of software but also for its application and the interpretation of related results. This document summarize certain classes of bugs of different aspects of software problems

1. Machine Numbers, Precision, and Rounding Errors

Are bugs related to working with numbers, as well as their representation. The finite representation of numbers in a computer is a notorious source of errors. The set of integers or real numbers contains infinitely many elements. But every computer is finite, and thus every integer and real number has to be stored in a data format using only a finite number of bits.
Storing a number in an inappropriate data format represents an error and typically results in unwanted behavior or even failure of the corresponding computer program

The Failure of the Ariane 5

• Field of application: near-Earth spaceflight
• What happened: self-destruction in view of unstable flight behavior
• Loss: more than $500 million for satellites
• Type of failure: integer overflow

Y2K and Data Formats

• Field of application: various
• What happened: fear of widespread breakdown of computer systems on January 1, 2000
• Loss: billions of dollars
• Type of failure: too-short data format in legacy software, Bug was an error caused by coding the year only by its last two digits: 99 versus 1999.

Vancouver Stock Exchange

• Field of application: financial market
• What happened: wrong computing of stock market index
• Loss: —
• Type of failure: accumulated rounding error, software was already used to compute the indexes and other economic data at stock exchanges

The Patriot Missile Defense Incident

• Field of application: military defense
• What happened: failure to intercept incoming missile
• Loss: 28 dead soldiers and more than 90 injured
• Type of failure: rounding error

2. Mathematical Modeling and Discretization

Solving physical problems on the computer first requires a mathematical description of the phenomenon (frequently called modeling) that has to be restated in a form suitable for computing. This is achieved by a discretization of the physical domain and the underlying Functions.

Mathematical equations modeling real-world phenomena depend on parameters that have to be set by the user in the program. In physical applications these parameters are constant, such as the speed of light or the gravitational constant. Financial mathematics tries to describe the future development of stock prices or options, e.g., by mathematical models similar to physical systems. But the corresponding parameters are nearly constant only as long as the economic circumstances are stable. In the event of a market crash, all these predictions are more or less worthless.

Loss of the Gas Rig Sleipner A

• Field of application: civil engineering
• What happened: complete collapse of the concrete structure prepared for the gas rig Sleipner A in 1991
• Loss: $180–$250 million concerning the concrete structure and an estimated $700 million to $1 billion of total costs including costs for the delayed start of operation
• Type of failure: erroneous usage of simulation software

London Millennium Bridge

• Field of application: civil engineering
• What happened: wobbling bridge
• Loss: £5 million
• Type of failure: insufficient input assumptions in FE (finite element method) analyses. The analysis of the excitation (The influence of wind and pedestrians on the stability of the bridge) showed no serious problems, and tests with a few people confirmed the calculated results.

Weather Prediction

• Field of application: weather forecast
• What happened: underestimating the thunderstorm Lothar in Europe in 1999
• Loss: €5 billion and up to 60 deaths from thunderstorm Lothar in Europe
• Type of failure: wrong input and coarse time window in the context of data assimilation

Mathematical Finance

• Field of application: financial markets
• What happened: contribution to global financial crisis, missing stopping criterion for stock orders
• Loss: billions of dollars

• Type of failure: careless (greedy) use of mathematical models and computer (On Wall Street, a large amount of stock market trading is realized automatically via algorithms and computers, when certain market trends prevailed.

3. Design of Control Systems

Numerical aspects are relevant to designing control logic and larger control systems such as real-time programs for the control of aircraft or cars.
Autonomously driving cars also depend on numerical input gathered by cameras or radar, e.g., to generate a virtual representation of the environment that is necessary to control the vehicle to avoid accidents. Mistakes in designing software for determining obstacles can lead to fatal accidents. Similarly, errors in the control structures of code can have fatal consequences in certain applications, such as activation of airbags.

Fly-by-Wire

• Field of application: aviation
• What happened: crash during landing in Warsaw in 1993
• Loss: 2 dead and 56 injured
• Type of failure: unreasonable conditions for allowing braking during landing. In modern airplanes, manual flight controls are replaced by an electronic interface and a computer system. Such fly-by-wire systems use digitized sensor data and pilot commands as well as software to decide on the necessary actions to operate the elevator, rudder, etc. Under normal conditions the computer can avoid dangerous situations where the flight could become unstable due to certain commands of the pilot, but in extreme situations, the software may prohibit actions that are dangerous but necessary to avoid a crash.

Automotive Bugs

• Field of application: automotive industry
• What happened: deployment of deactivated airbag; autonomous driving accidents
• Loss: several deaths
• Type of failure: faulty extension of legacy software; incorrect identification of obstacles

4. Synchronization and Scheduling

In order to solve specific tasks in a distributed manner, communication between and timing of jobs is necessary. In this way, time-dependent arrangements are important when connecting different computers.

Space Shuttle

• Field of application: space flight
• What happened: delay of launch
• Loss: —
• Type of failure: synchronization failure, when used five computers: four identical in software and hardware and one different in software. The four identical computers were running the same processes in parallel and allowed an intermediate replacement of a failing computer. The fifth computer served as the backup that had to be able to take over control in the event of a consistent failure of the other systems. This had to listen to all input and some output data traffic from the four-

computer syste with an asynchronous design, while the other computer was implemented as a fully synchronous system.

5. Software-Hardware Interplay

Computer hardware is notoriously impacted by flaws due to its complicated design. Manufacturers such as Intel, AMD, and NVIDIA typically struggle to create robust systems, but bugs continue to appear

• Field of application: chip design
• What happened: Intel's Pentium processor delivered inaccurate division results for certain input values
• Loss: $475 million
• Type of failure: missing data in lookup table

Mars Rover Spirit Flash Memory Problem

• Field of application: space flight
• What happened: unwanted reboots of the control computer
• Loss: time and, thus, data due to fewer days of operation
• Type of failure: overflow related to file system and storage management (FAT32)

6. Complexity

Software is becoming more and more elaborate and voluminous. Considering operating systems (OS), Windows 10 had about 50 million lines of code, Mac OS X 10.4 about 86 million lines, and the Linux kernel 3.6 about 15.9 million lines of code. This increases the probability and the number of errors in a program, and, thus, also the vulnerability of the OS.
Especially in the beginning of the computer era, software was considered safe and reliable compared to hardware. This led to accidents in which medical devices killed people due to wrong radiation settings, as in the case of the infamous Therac-25.

Therac-25

• Field of application: medical engineering
• What happened: malpractice by radiation overdose
• Loss: at least two persons killed and four badly injured due to overdose
• Type of failure: error-prone software design in context of a complex system (race conditions, integer overflow).
1. Race conditions are related to shared resources used by different concurrent tasks. Problem of insufficient mechanism to detect editing
2. A combination of an integer overflow in the variable used for indicating proper settings

Bibliography

Thomas Huckle, Bits and Bugs: A Scientific and Historical Review of Software Failures in Computational Science, febrero, 2019.