ECEN-5623
ASSIGNMENT-5

YASH GUPTE

Date: 26th July 2019

1) [15 points] Research, identify and briefly describe the 3 worst software and hardware computer engineering design flaws and/or software defects of all time [product lack of success can be considered a failure such as Blackberry Storm, Windows Genuine Advantage, Windows 8, Windows ME, Apple Lisa, Pentium FPU bug, as well as mission failures such NORAD false alarms, Mars Express Beagle 2, Challenger and Columbia Shuttle Loss]. State why the 3 you have selected are the worst in terms of negative impact, negligence, and bad decisions made, but why the failure was not really a real-time or interactive systems design flaw. Rank them from worst to least worst.

Ans-1)

Challenger and Columbia Space Shuttle loss: (Link)

On January 28, 1986, Challenger Space Shuttle disintegrated 73s after take-off due to a failure of the SRB leading to the death of all seven astronauts on board. The cause of the failure was due to impairment of O-Rings at low temperatures causing hot combustible gases to escape the booster into the external tank leading to the catastrophe.

On February 1, 2003, Columbia Space Shuttle disintegrated during entry into the earth orbit. This resulted in death of all the crew members. The issue was a result of damage to the carbon-carbon edge during the launch.

1. **Negligence:**
   A. Challenger Space Shuttle
   O Rings were needed to function at low temperatures below 12C. However, O Rings were not tested even after continued requests by the Design Engineers for cost issues as declared by NASA Managers.
   B. Columbia Space Shuttle
   Due to the incurred damage to the Columbia Space Shuttle, the engineers requested for high resolution images of the space craft wings.
2. **Bad Decisions:**
   A. Challenger Space Shuttle
   The NASA managers continually declined the requested to perform tests for the O-Rings seal integrity under 12C.
   B. Columbia Space Shuttle
   NASA's chief Thermal Protection System (TPS) needed the astronauts to leave their Space Vehicle and to inspect the damage to the Columbia Space shuttle which occurred during take off. However, the Space Managers refused the request for the space walk on the grounds of monetary excuses.
3. **Negative Impact:**
   On both these events, the negligence and bad decisions led to the death of the astronauts on board.

Windows Genuine Advantage: (Link)

WGA is system which checks for online validation of authentic windows operating systems' licenses. WGA suddenly started falsely claiming genuine licensed copies as false ones and rendering certain versions of Windows useless.

1. **Negligence:**
   The WGA program could provide false positive that is claim a false copy was genuine. Additionally, genuine copies could be diagnosed to be fake due to improper software updates. The negligence to check for proper diagnoses before launching the WGA was an issue on behalf of Windows.
2. **Bad Decisions:**
   Windows decided to embed a feature within WGA which could automatically call the users to confirm their identity called as "". This occurred in situations where the WGA deemed it safe to confirm the actual identity of the user. This was taken as a spyware like behavior by the users. Some even went to the extent of calling it breach of privacy.
3. **Negative Impact:**
   A lawsuit was filed against Windows for the "phone-home" behavior. Though this lawsuit was later dropped. It added with a bad name and marginal affect to the reputation of Microsoft as a company.

NORAD: (Link)
On 9th November 1979, a stream of constant false warnings spread to two "Continuity Of Government" government bunkers as well as command posts all around the world. A similar false alarm was set on 3 June 1980 sending warnings that a nuclear war was in effect. This time it was due to a computer communications failure.

1. **Negligence:** A technician accidently loaded a test tape into one of the NORAD systems but forgot to change the system status to "test" causing the abovementioned scenario. In the second incident, it was sheer negligence that even after a case of false triggered alarms had occurred, no steps were taken to audit the system in place and check for future faults in the system.
2. **Bad Decisions:** When the alarm was set on 3 June 1980, as per protocol, Pacific Air Forces (PACAF) had their planes loaded with nuclear bombs in the air. On the other hand, Strategic Air Force (SAC) did not, because they knew these were false alarms. This decision of SAC was faced with criticism.
3. **Negative Effects:** With the timing of this incident coinciding with the Cold War, tensions became even more tense between USSR and USA. President Carter's office had to come up with a befitting reply for the USSR and press.

Ranking of critical level of the events:
1. **Space Shuttle incident** – This is because human lives were claimed due to the bad decisions and negligence.
2. **NORAD** – The negligence of the operator as well as complacency of the SAC could have aggravated the tensions between the already volatile situation between USSR and USA. A declaration of war could have caused mass hysteria.
3. **WGA** – While this incident did cause a large loss for Microsoft in terms of upset customers, it did not cause a threat to human lives. The people were still agitated but eventually the patched released by Microsoft made up for this bug.

2) [15 points] Research, identify and briefly describe the 3 worst real-time mission critical designs errors (and/or implementation errors) of all time [some candidates are Three Mile Island, Mars Observer, Ariane 5-501, Cluster spacecraft, Mars Climate Orbiter, ATT 4ESS Upgrade, Therac-25, Toyota ABS Software]. Note that Apollo 11 and Mars Pathfinder had anomalies while on mission, but quick thinking and good design helped save those missions]. State why the systems failed in terms of real-time requirements (deterministic or predictable response requirements) and if a real-time design error can be considered the root cause.

Ans-2)

MARS Observer: (Link)

Brief:

On August 12st 1993, at 01:00 UTC, there was a sudden unexpected loss of contact with the Mars Observer. This was three days prior to the Mars Orbital Insertion. Attempts were made to contact the Observer every 20mins but to no avail. It was a mystery id the MARS orbiter managed to enter the Mars orbit or went in a heliocentric orbit instead.

Issue:

There had been a rupture in the fuel pressurization tank in the spacecraft's propulsion system. This caused leakage of fuel and gas resulting in high spin rate of the spacecraft. This caused the spacecraft to enter Contingency Mode. This caused an interruption in the stored command sequence and as a result failed to turn on the transmitter. This led to the failed communication attempts of the Space Center.

Cause:

The study about the aircraft's communication failure led to some findings. The boards study concluded that the issues with the propulsion system were a result of unintended mixing of Nitrogen Tetroxide (NTO) with MMH and a resulting chemical reaction between them. This reaction occurred within Titanium Pressurization tubing at the time when fuel tanks were being pressurized with helium. The violent reaction led to a rupture in the tubing causing the helium and MMH to get mixed. The led to the catastrophic spin of the aircraft as well as damaging critical electronics circuitry.

Real-time failure cause:

While the MMH leak critically destroyed the electrical components, it is inevitable that this would render the electronics completely useless. However, it there could have been safeguards to such a situation. From a Real-Time perspective, this situation could have been averted. A mechanism needed to be in place which could detect the exceeding values of the spin. For example, the accelerometer or gyroscope on the Space Shuttle should have transmitted this situation back to the control center. The absence of such a mechanism was a Real-Time failure.

Root Cause:

Real-Time failure cannot be considered as a root cause for this failure. This is because the rupture in the pressurization tanks and mixing of the gases was not a real-time issue. However, the time and money spent on understanding the cause of such a failure, could have been prevented by suitable Real-Time response which could alert the ground control.

ARIENE-5)(Link)

Brief:

On 4th of June 1996, 40s after initiation of the flight sequence, Ariane 5 veered off its flight path at about 3700m. Shortly after it broke up and exploded.

Issue:

The Ariane 5 launcher started to break apart at 39s after takeoff as a result of high dynamic loads caused due to a 20degree angle of attack. This attack led to the separation of the Boosters from the main stage. The system was designed to enter a self-destruct mode in case the boosters got separated from the main stage. The 20-degree angle of attack was caused due to full nozzle deflections of solid boosters and the Vulcain engine. The On-Board Computer (OBC) was responsible for commanding the nozzle deflections based on the data received from Inertial Reference System (SRI 2). The data provided to OBC by SRI-2 contained a diagnostic bit pattern of the SRI-2 computer instead of the actual data. This data provided was considered as valid data for nozzle deflections which led to the catastrophe.

Cause:

The reason why SRI-2 was unable to send correct data was since it had declared a failure due to a software exception already. The OBC was designed to switch to SRI-1 in case such a scenario arose. However, SRI-1 was designed to run every 72ms period however it had already ceased to function due to a software exception akin to that in SRI-2. The internal SRI-1 exception had occurred during a data conversion from 64-bit floating point number to a 16-bit signed value. The 16-bit number which was a result of a conversion from float had received a value which exceeded the 16-bit number value. This led to an **Operand Error**. All other conversions in the code were protected from comparable variable conversion errors but only this conversion was left unprotected.

Real-Time Failure Cause:

The error which occurred on the shuttle was in a part of the code which was involved with strap-down inertial platform. This function served a purpose only if the Space vehicle took off. After which it was not needed. The lack of a proper software guard against type conversions led to the entire system to cascade into errors and then fail.

Root-Cause:

Real-Time failure should be considered as a root cause for this failure. This is because, the error did not involve any hardware. The cause originated from an issue in SRI-1 during a floating-point type conversion error which rendered it non-functional. Secondly, SRI-2 experienced a software exception due to the same reason as SRI-1. SRI-2 provided invalid data to OBC. Which was interpreted as valid data and caused system to crash. There could be the following preventions for this scenario. Firstly, checks for proper type conversions should have been in place for SRI-2 and SRI-1. Secondly, OBC should have been notified of such an exception the moment it occurred. Thirdly, there should have been a parity or checking mechanism to ensure the data from SRI-2 is actually valid before being used by OBC.

Toyota ABS Software: (Link, Link-2)

Brief:

On February 3rd, 2010, the NHTSA announced complaints of 102 drivers concerning the problems related to braking system on 2010 model of Toyota Prius. In Japan, 14 additional such reports had been received. Three among these claimed brake problems led to car crashing.

Issue:

The Anti-Lock Braking system is designed to engage and disengage rapidly as it senses and reacts to tire slippage. However, in 2010, Toyota Prius and Lexus HS250h owners reported to have experienced inconsistent brake feel during rough or slick roads while trying to maintain traction.

Cause:

The cause of the temporary braking was that when the car hit a bump, the regenerative braking present in the front 2 wheels used to cut out and was instead replaced by frictional braking. There was however a delay between the cut out of regenerative braking and start of frictional braking. This caused loss of deceleration for a small amount of time. The net impact being a loss of braking and increase in stopping distance.

Real-Time Failure Cause:

The software in the Toyota Prius present in the 2010 model had a new redesigned electronics control system which controlled the Anti-Lock Braking system. The redesigned control system integrated Anti-Lock braking with a combination of regenerative and conventional brakes. The issue arose when there was a switch from the battery charged regenerative system to the conventional friction braking. The owners who reported the brake failure experienced the same on slippery roads especially in Winters. This is because frictional braking is known to be ineffective against a smoother surface.

Root-Cause:

Real-Time failure should be considered as a root cause for this failure. This is because the braking issues which occurred were solely due to the redesigned electronics control system. The delay in switching between ABS and conventional braking systems caused a reduced rate of retardation in the vehicles. Proper realtime mechanisms to reduce this delay could prevent the bad reviews. There is a second face to this argument. Frictional braking has its flaws and a result was replaced by ABS braking back in the day. The decision of the engineers to switch to a more primitive braking system was in order to protect the batteries since they were recharged during regenerative braking. During a bump, there was a chance that the batteries could get damaged. While this does not justify the engineers' choices it does provide an explanation for their actions.

3) [20 points] The USS Yorktown is reported to have had a real-time interactive system failure after an upgrade to use a distributed Windows NT mission operations support system. Papers on the incident

that left the USS Yorktown disabled at sea have been uploaded to Canvas for your review, but please also do your own research on the topic. a) [5 pts] Provide a summary of key findings by Gregory Slabodkin as reported in GCN. b) [5 pts] Can you find any papers written by other authors that disagree with the key findings of Gregory Slabodkin as reported in GCN? If so, what are the alternate key findings? c) [5 pts] Based on your understanding, describe what you believe to be the root cause of the fatal and near fatal accidents involving this machine and whether the root cause at all involves the operator interface. d) [5 pts] Do you believe that upgrade of the Aegis systems to RedHawk Linux or another variety of real-time Linux would help reduce operational anomalies and defects compared to adaptation of Windows or use of a traditional RTOS or Cyclic Executive? Please give at least 2 reasons why or why you would or would not recommend Linux

Ans-3)

Summary

On September 1st of 1997, US Navy ship USS Yorktown stalled in the middle of the ocean for 2 hours and it needed to be brought into a pier for a fix. The article discusses view of different personnel on board the ship during the incident as well as comments of the Navy General and Chief Information Officers. The comments vary in opinion as to what occurred and why it occurred. These discussions involve flaws inherent to the OS on the NAVY ship as well arguments to defend it.

The USS Yorktown was a part of testing the Navy's new Smart Ship program aiming at reducing crew by automated processes. These processes were based on a common Operating System (OS) throughout all the Navy ship programs. This OS was Windows NT.

Yorktown's Standard Monitoring Control System administrator entered zero into the data field for the Remote Data Base Manager program. This led to cascading failure of multiple systems on the ship ultimately leading to all system failures and stalling of the ship.

Statements of various persons on and off the ship were recorded for a clearer understanding of the root cause of the system failure. A Civilian engineer on-board the ship Anthony DiGiorgio stated that use of an inherently flawed system such as Windows NT on a war ship was relying on luck during a war. Captain Richard Ruston admitted to propulsion power breakdown in the system twice before this incident. However, he did not consider Windows NT to be the basis of the problem. The reason was actually blamed on the operator negligence and LAN casualty errors which he said were common during testing.

Ron Redman deputy technical director of the Fleet Introduction Division of the Aegis Program Executive Office stated that he too realized flaws in the Windows NT system and recommended the use of Unix for equipment control and machinery and Windows NT for information and data.

All in all, this article discusses different viewpoints surrounding the event.

Conflicting Viewpoints

In the view of the events of 1st September 1997 and the subsequent findings and discussions, the main topic which took the limelight was the operating system OS which was used on the Navy ship that day; **Windows NT**.

While the viewpoints mentioned did not directly conflict with the finding and arguments stated on the article by Gregory Slabodkin, they did have a different view of the same findings.

Ron Morse mentioned in his discussion with Jerry Pournelle(Link) that while it was true that there had been failures with the Windows NT failures, it was a learning process. He stated that the USS Yorktown was a testbed for the US Navy Smart Ship program. Having failures in such scenarios is imperative to understand the potential flaws at the time of war. He also showed an increasing support for the US Navy Smart Ship program saying that it is one of the most exciting programs which enables engineers to work at a closer proximity with the engines than the bureaucrats and admirals.

The second statement was made by Talin. (Link) He did not outright berate Windows NT. He stated that OS on complex systems especially a Navy Warship were prone to errors and failures. However, he recommended optimal analysis of the system for all possible failures. These analysis hold a lot of importance to make sure these failures are not repeated as well as steps taken to resolve them. His second suggestion was that the code for the systems must be distributed among various personnel involved in the system development process. This enables 'peer review' and as a result, any potential errors or flaws in the logic can be brought forth early on in the development process.

Cause of error

The cause of errors were the inherent flaws in Windows NT. The system failed because the Standard Monitoring Control System administrator entered a zero into the data field for the Remote Database Manager program which caused a buffer overflow. This overflow crashed the entire system as a result the ship stalled in the middle of the ocean.

The OS was not designed to handle such exceptions with grace. As a result, due to this error, the system stopped execution and did not even service the other components of the ship, such as the propulsion system. This led to a complete failure of all systems due to an error in one part of the system. This also suggests that Windows NT may not be the most robust and secure OS.

On reading articles about Windows NT, it became evident that the Windows NT version used in 1997 had some major flaws. The first of these can be gathered from an article by John Leyden (Link) in which he states that the Windows NT systems had a vulnerability. The vulnerability was *"involves the Microsoft's implementation of Remote Procedure Call protocol, more specifically the component that deals with message exchange over TCP/IP. Malformed messages received by the Endpoint Mapper process, which listens on TCP/IP port 135, might cause a server to hang."*

While the error which occurred in 1997 was due to a divide by 0, the entire Navy ship was programmed using Windows NT. Windows NT was also responsible for the Local Area Networks and ship communications. A vulnerability of this kind could have rendered the system useless as well. Moreover, this vulnerability gained further limelight when Microsoft released a statement which stated "*The architectural limitations of Windows NT 4.0 do not support the changes that would be required to remove this vulnerability,"*.

I do not believe that the error was the operator's fault. The system should have been able to handle such a an error gracefully.

Opinion on Linux based systems

In my opinion, use of Linux based systems especially such as Redhawk Linux could have prevented the errors and failures caused on the USS Yorktown on September of 1997. The following are the reasons for using Linux instead.

1. Mechanism to handle divide by 0 error: (Link)
   There is a separate mechanism to handle divide by 0 faults. In order to prevent a system failure as that on the Windows NT system, there is a separate signal SIGPE which is released indicating a Floating-Point Error especially divide by 0 and overflow. An argument in GNU C Library called FPE_INTDIV_TRAP indicates the system in case of SIGFPE signal and suitable actions can be taken. In the above case, this implementation could have prompted the operator to enter a valid value in case of a 0 he initially inserted.
2. Security concerns: (Link) (Link2)
   The lack of security on the Windows NT was exemplified by 2003 attacks on Windows NT based systems using the Windows Metafile Vulnerability. The WMV bug permitted arbitrary code to be executed on host machines without their permission. In Linux, there is a special version called Security Enabled Linux (SELinux) which defines the rights of every user, applications, process and file on the system. It also governs all these functions using a security policy. These features can completely prevent such an attack.

4) [50 points total] Form a team of 1, 2, or at most 3 students and propose a final Real-Time prototype, experiment, or design exercise to do as a team. For the summer RT-Systems class, all groups are asked to complete at time-lapse monitoring design as outlined in requirements with more details provided in this RFP. Based upon requirements and your understanding of the UVC driver and camera systems with embedded Linux (or alternatives such as FreeRTOS or Zephyr), evaluate the services you expect to run in real-time with Cheddar, with your own analysis, and then test with tracing and profiling to determine how well your design should work for predictable response. You will complete this effort in Lab Exercise #6, making this an extended lab exercise, and ultimately you will be graded overall according to this rubric and you must submit a final report with Lab Exercise #6 as outlined here. a) [30 pts] Your Group should submit a proposal that outlines your response to the requirements and RFP provided. This should include some research with citations (at least 3) or papers read, key methods to be used (from book references), and what you read and consulted. b) [20 pts] Each individual should turn in a paragraph on their role in the project and an outline of what they intend to contribute (design, documentation, testing, analysis, coding, drivers, debugging, etc.). For groups of 2 or 3, it is paramount that you specify individual roles and contributions.

Ans-4)

<u>**PART-A**</u>

<u>**Project Selection:**</u>

Time Lapse

<u>**Requirements:**</u>

A. Minimum Requirements

1. The resolution used will be VGA (640x480).
2. Frames will be acquired in 2 separate rates; 1Hz and 10Hz.
3. Images will be stored in PPM P6 RGB format.
4. Every frame which is saved will incorporate a timestamp in the header for that file.
5. A '#' will exist as the suffix for the timestamp for each frame.
6. An additional '#' prefixed system name, "uname -a" will be appended to the file's header.

B. Verification of Minimum Requirements

1. The images captured will be compared with an external wall clock over a duration of 30mins with 1800 frames.
2. The maximum error between all the 1800 frames will be less than 1s. Which means, images captured must be apart by 1s for 1Hz mode and 100ms for 10Hz mode. A maximum deviation of One frame from the intended final frame must be observed.
3. In order to achieve this, initial frames with the target start time will be counted and the will be done once the system finishes capturing the final frames. This is to ensure that, when the final set of frames are captured, they do not show a value one second less than the intended time.
4. Mechanism to measure the average frame jitter will be used such that after every frame which is captured, a difference from the start time will be taken. A difference in time value which exceeds a multiple of the selected frequency (1Hz or 10Hz) will indicate a frame jitter. This accumulated jitter will be calculated over a fixed number of frames and will be adjusted with the timer which triggers the scheduler.
5. An additional time lapse images of a process such as melting of an ice cube will be captured and the same will be converted to a video file MPEG2/MPEG4. This will be attached in the final project folder.

C. Target goals

Continuous download of frames over Ethernet so that you can run indefinitely and never run out of space on your flash file system which should maintain only the last 2000 frames.

D. Stretch Goals

Run at a higher frame rate of 10 Hz with continuous download, this time for 9 minutes, which will produce up to 6000 frames (about 6GB uncompressed for 640x480) and repeat the jitter and accumulated latency verification at this higher rate.

Framework:

1. A readthread running on Core-0 will capture frames at a fixed rate (10-40 fps) and write to a circular buffer of fixed size (100).
2. A scheduler will run at either 1hz or 10Hz as specified by user on the same core as Read frame but at higher priority.
3. RGB processing, socket transfer and grey scale(*) on same core running as RM. With priorities RGB first, Socket transfer 2nd and Greyscale third.
4. On a third core, Image read from Ring buffer will take place. These images will be written to the specified format in the bit-field stored within each entry of the circular buffer. This will all run on Core-1. The date, time as well as 'uname' will be added to the header of the image.
5. All these threads will run till 1800 frames have been completed by RGB processing.

Testing :

There will be tests to check for:

1. If the initial images captured are blurry.
2. The frame jitter between frames.
3. Packets received via sockets.

Procedure for testing:

1. Before the initiation of the threads, there will be a check for obtaining frames with a stable clock hand position I.e. non blurry. The time at this moment will be recorded as the start time and the threads will begin execution.
2. When every frame is captured time difference wrt to the first frame will be taken. If it is greater than the multiple set sample frequency of 1Hz or 10Hz, the difference will be subtracted from the Scheduler timing mechanism (interrupt or nano sleep)
3. The packets received via Sockets on the client PC will be checked for blurrs and time stamp issues.

Bibliography and references:

A. Methods

There are three main features of this real-time project which will be needed for proper implementation of the required guidelines.

1. **Coordinated Universal Time** (UTC) – **UTC** is the primary standard of setting clock times all around the world. It lies within 1 second of mean solar time. UTC is also known as Greenwich Mean Time (GMT). UTC is based on International Atomic Time (TAI) with an addition of leap seconds to compensate for the earth's slowing rotation. Leap seconds are added to keep the UTC within 0.9seconds of the UT1 variant of universal time.
2. **International Atomic Time** (TAI) - **TAI** is a weighted average of the time kept by over 400 atomic clocks in more than 50 national laboratories worldwide. The definition of a second is based on Cesium clocks is a part of International System of Units (**SI Units**). These clocks are compared with

GPS signals and with two-way satellite time and frequency transfer. TAI is an order of magnitude more stable than its best constituent clock due to signal averaging.

3. **Network Time Protocol (**NTP**) – NTP** provides clock synchronization between computer systems over packet switched and variable latency data networks. NTP is a networking protocol intended to Synchronize all participating computers to within a few milliseconds of UTC.
4. **Sockets** – On a computer network, a node is an internal endpoint for receiving data within a node. Sockets will be used to transfer data between Nvidia and the Linux system. This transfer will occur for 6000 frames. After this the frames will be overwritten.

B. References:

1. https://en.wikipedia.org/wiki/Network_socket
2. https://en.wikipedia.org/wiki/Network_Time_Protocol
3. https://en.wikipedia.org/wiki/International_Atomic_Time
4. https://en.wikipedia.org/wiki/Coordinated_Universal_Time

PART-B)

This is an individual project, as a result all the programming, testing, verification and report writing will be done by me.