

# EECE499-01: Computers and Nuclear Energy

**Charles Kim**

**Fall 2012**

# Computers and Risks

## ⌘ Lecture Resource

☑ Chapter 2 of *Safeware* by Nancy Leveson

## ⌘ Contents

☑ Background

☑ The role of computers in accidents

☑ Software Myths

☑ Why Software Engineering is Difficult

☑ The Reality We Face

## Background -1

### ⌘ Invention of computers

### ⌘ General Purpose Computers

- ☒ Design of functions or desired go
- ☒ Writing down the design : Code
- ☒ Code loaded into the computer
- ☒ Instructions executed
- ☒ Software + General-Purpose Computer →  
Special-Purpose Machine
- ☒ Any change required? Then,
  - ☒ Revise the code
  - ☒ No need of building a physically different machine from scratch
  - ☒ Patches and add-on etc

## Background -2

### ⌘ Impact of General Purpose Computers + Software

- ☑ Manufacturing phase is eliminated from the life cycle
- ☑ Design and verification phase only
- ☑ No sense of physics
- ☑ Explosive increase of uses and applications
- ☑ Can this be applied to ALL systems without considering its dangers?

### ⌘ Study subjects

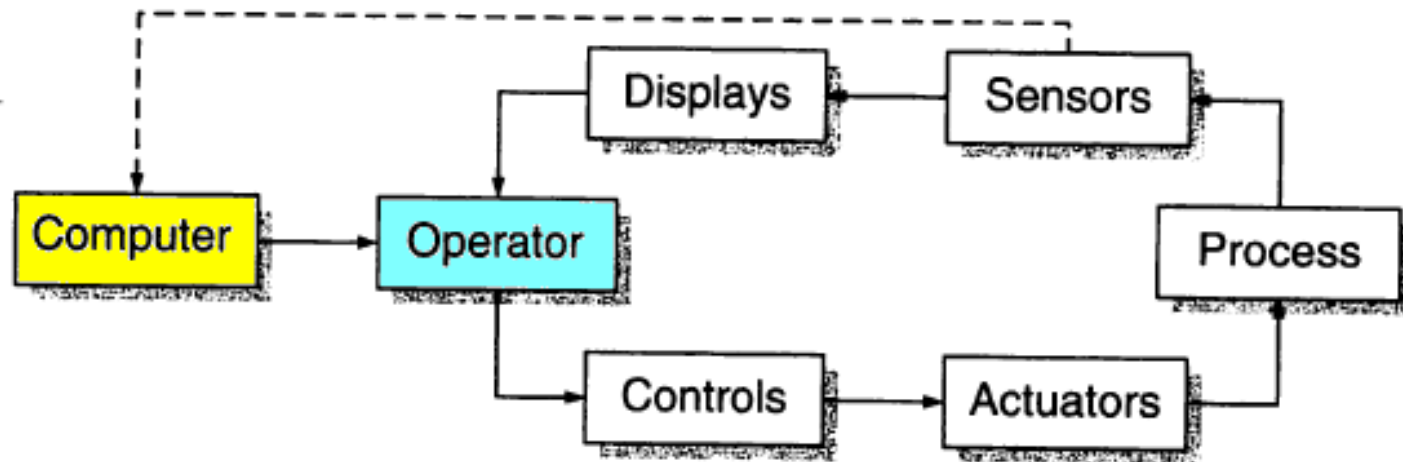
- ☑ Role of computers in accidents
- ☑ Some of the myths related to its use

## The Role of Computers in Accidents

- ⌘ Most control functions are done by computers
- ⌘ Most design and design support are done by computers
- ⌘ Most safety-critical devices are now controlled by computers – replacing traditional hardware **safety interlocks** and protection systems
- ⌘ Software **interlock**?

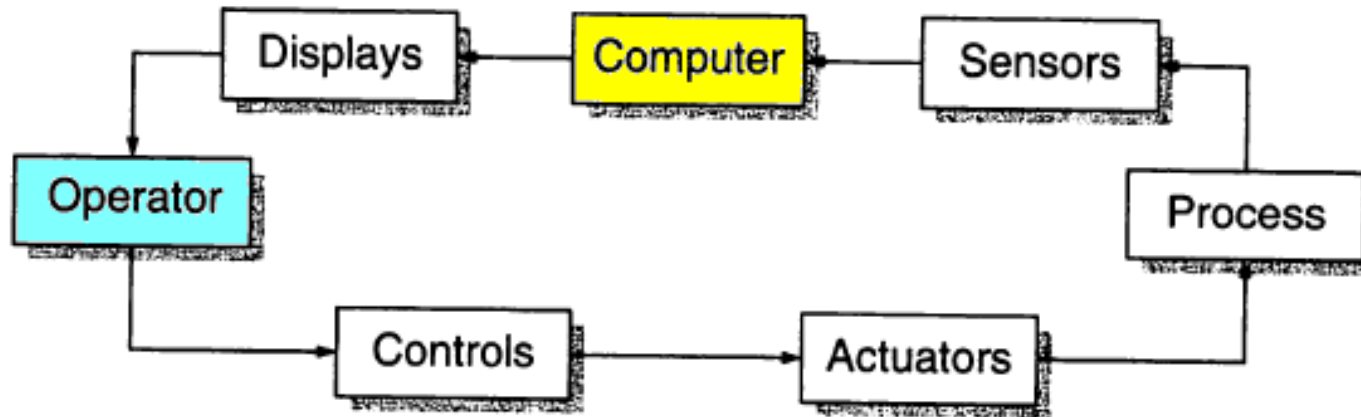
# Computer use in Safety-Critical Loop 1

⌘ Aid to human controller by reading sensors



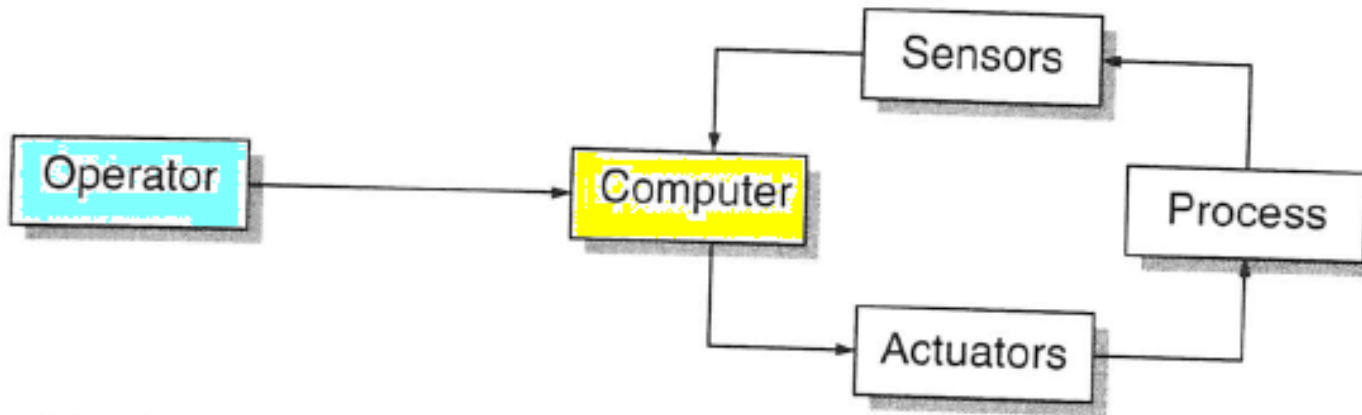
## Computer use in Safety-Critical Loop 2

- ⌘ Aid to human controller by reading sensors and interpreting sensor data



## Computer use in Safety-Critical Loop 3

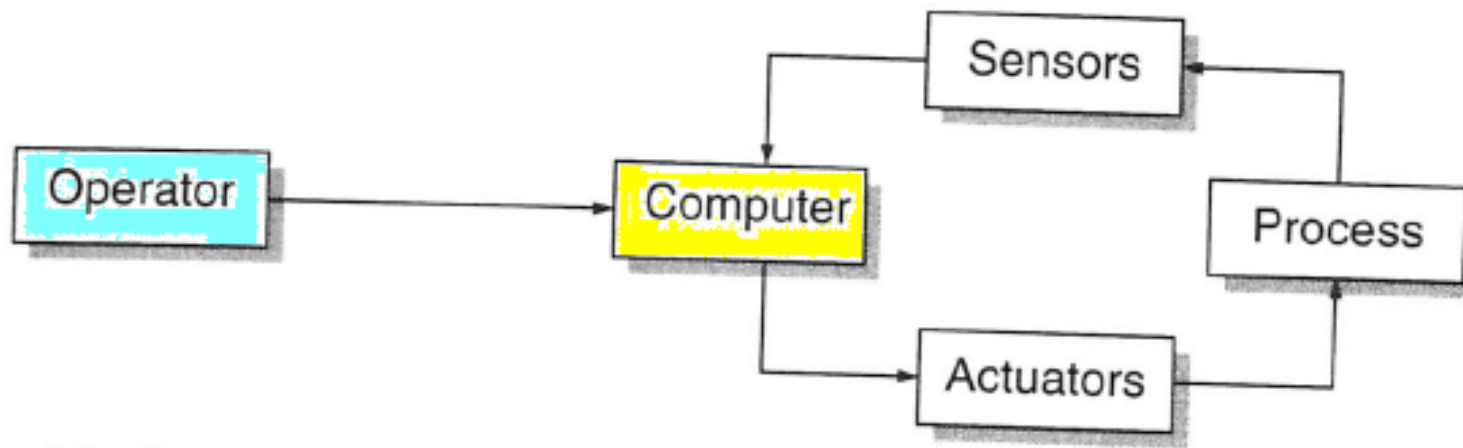
- ⌘ Computer assumes the complete control with operator as adviser





## Computer use in Safety-Critical Loop 4

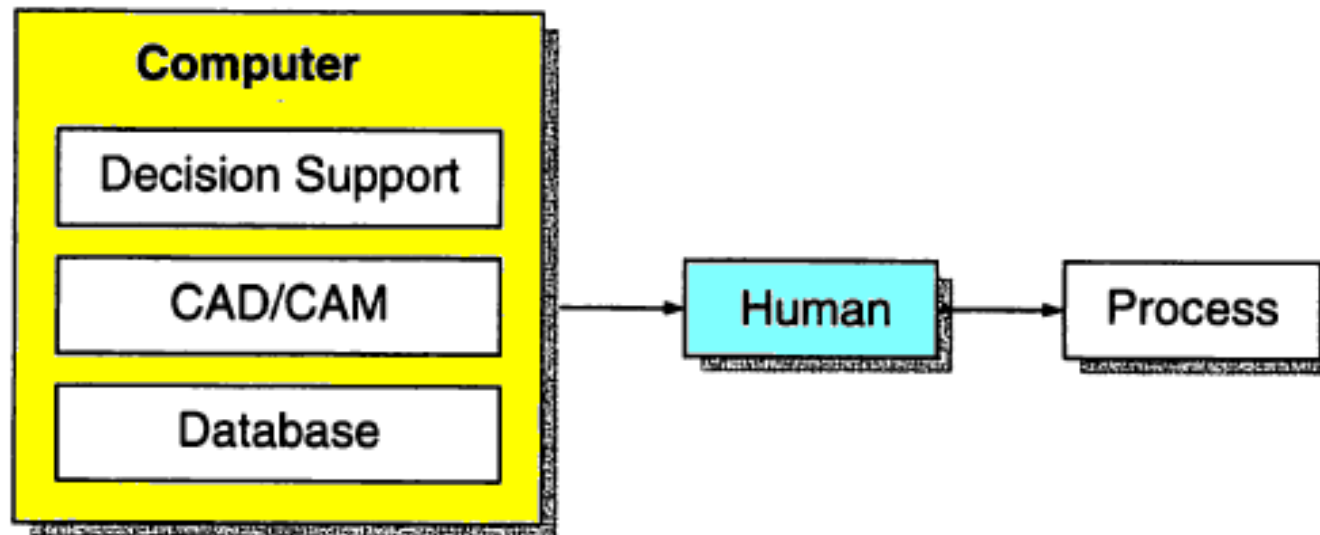
- ⌘ Computer assumes the complete control with operator as adviser



- ⌘ Cf. HAL in "2001 A Space Odyssey"

# Dangers in Computer Control

- ⌘ Software generated data is used to make safety-critical decisions
- ⌘ Software is used in design analysis (**Self-serving?**)
- ⌘ Safety-critical data is stored in computer database (**or Cloud?**)



# Are they reliable?

What Happens When a  
Medical Office Information System Fails

Nuclear Cooling System Software Error



- ⌘ An F-14 drove off the deck of an aircraft carrier on command from its computer-controlled throttle.

# Undetected Nuclear Software Error

## Hitachi Finds Nuclear Software Fault; Undetected for 28 Years

*By Shigeru Sato - April 10, 2008 22:57 EDT*

April 11 (Bloomberg) -- [Hitachi Ltd.](#), Japan's third-largest builder of nuclear reactors, discovered a programming error in software used for almost three decades to measure the impact of earthquakes on pipes at atomic power stations.

The mistake, made by a Hitachi programmer, allows the software to underestimate the quake impact on steel pipes associated with eight nuclear reactors owned by six utilities, including [Tokyo Electric Power Co.](#), Hitachi spokesman [Keisaku Shibatani](#) said by telephone.

Confidence in the safety of Japan's nuclear power plants has been shaken after a 6.8-magnitude earthquake caused a fire and radiation leaks at a Tokyo Electric facility in Niigata prefecture last July. Twelve power producers, responding to a government request, revealed in March 2007 more than 300 cases of improper safety practices. Hitachi reported the software problem to the utilities this week, Shibatani said.

"It was a human error," he said. "We're closely looking into this now."

Hitachi rose 0.5 percent to 648 yen at the close of morning trading in Tokyo compared with a 1.5 percent rise in the benchmark Topix index.

One of the eight plants was built by General Electric Co., which has an alliance with Hitachi, and the rest employ Hitachi technology. Test results using properly programmed software show the pipes can be used safely, Shibatani said.

The owners of the reactors include Tohoku Electric Power Co., [Chubu Electric Power Co.](#), Hokuriku Electric Power Co., Chugoku Electric Power Co., and Japan Atomic Power Co.

# Y2K Bug ?

ITN

## Computer problems hit three nuclear plants in Japan

January 03, 2000

Share  Twitter Email

 Recommend

 One recommendation. [Sign Up](#) to see what your friends recommend.

by Martyn Williams

From...

Tokyo IDG Only a handful of computer problems have been reported in Japan in the new year to date; however, at least three hit systems associated with nuclear power plants, according to the government and power generating companies.

The potentially most serious problem occurred not at midnight but at 858 a.m. local time on Jan. 1 at the Fukushima Number 2 nuclear power plant of Tokyo Electric Power Co. TEPCO. The system that shows the position of the control rods in the reactor core failed, leaving operators unable to gauge the rods positions using the system.

ALSO  
Feds sound Y2K allclear  
Computer problems hit three nuclear plants in Japan  
Y2K bug hits heating system in Korean apartments  
Two glitches hit Microsoft Internet services as New Year rolls over  
Y2K sites around the world

# Missile Launch Control Glitch



## Computer problem blamed for missile site malfunction

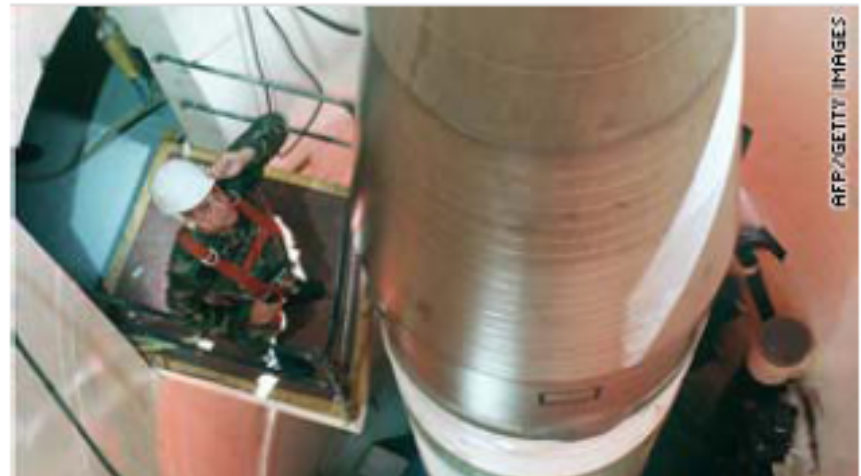
October 27, 2010 | By Larry Shaughnessy, CNN Pentagon Producer

A malfunctioning launch control center for a portion of the nation's nuclear missiles remained offline Wednesday as investigations continued into a weekend computer problem that disrupted communications with more than 10 percent of America's land-based nuclear missiles.

Early indications are that Saturday's disruption to one of the launch control centers linked to Warren Air Force Base in Wyoming lasted longer than an hour, Lt. Col. John Thomas said. The problem appears to be very similar to glitches at two other nuclear missile sites in the late 1990s.

Share  Twitter  Email

 Recommend  114 recommendations. [Sign Up](#) to see what your friends recommend.



An Air Force technician inspects a Minuteman III missile in a silo in North Dakota.

AFP/GETTY IMAGES

# New Challenges of Widespread use of Computers

- ⌘ Methods of **safe computer control** system is **lagged** behind the development of such systems
- ⌘ **Safety engineering** does **not include computers** (software in particular)
- ⌘ Software reliability (or safety) is **not easy** to adopt in to the existing reliability program
- ⌘ Computer engineers (software) are **not prepared** to cope with safety problems
- ⌘ **Communication problems** among system engineers, software engineers, operators, programmers, etc.

## Software Myths

⌘ If there are problems, why are computers being used so widely?

☑ Computer provides power, speed, and control

☑ Computer control is light and small

☑ Economical

⌘ Do these advantages last long?

⌘ Are they only for a short term?

⌘ What are the myths in using computers?



# Myth 1 - Cost

## ⌘ Myth

- ⊞ "The cost of computers is lower than that of analog or electromechanical devices"

## ⌘ Reality

- ⊞ Superficial truth
- ⊞ The cost of certifying highly reliable H/W and S/W, and of maintaining S/W can be enormous
  - ⊗ Example: On-board Space Shuttle software costs NASA \$100M a year to maintain
- ⊞ The lifetime cost of S/W is not cheap

# Myth 2 - Revision

## ⌘ Myth

☒ "Software is easy to change"

## ⌘ Reality

☒ Change is easy

☒ Change without introducing errors is extremely difficult

☒ Verification and re-verification is done at extremely high cost

☒ Running on Old Machine

☒ DMSMS (Diminishing Manufacturing  
& Material Shortages)

☒ ADA and FORTRAN

☒ Revised software contains more errors



# Myth 3 - Reliability

## ⌘ Myth

- ⊞ "Computers provide greater reliability than the devices they replace"

## ⌘ Reality

- ⊞ Mechanical failure and operator error (Random in nature) vs. software error (Design error in nature)
- ⊞ Software errors in highly safety-critical systems
  - ⊞ 10% deviation from original specification
  - ⊞ Discrepancies found even after extensive checking using sophisticated test platforms
  - ⊞ Truncation, round-off errors
- ⊞ The lifetime of software errors
  - ⊞ Over the entire life system lifetime – even after tens or hundreds of thousands of hours of use
  - ⊞ Therac-25 worked correctly thousands of times before the first known overdose
  - ⊞ Numerous errors in NASA space shuttle software
- ⊞ Redundancy and Diversity for hardware reliability but no proven method for software reliability

## Myth 4 – Error Removal

### ⌘ Myth 4

- ☒ “testing software or proving (using formal verification techniques) software can remove all the errors”

### ⌘ Reality

- ☒ Limitation of software testing
- ☒ Exhaustive software testing impossible
- ☒ Mathematical verification is still not practical
- ☒ Overload problems: Three Mile Island, Emergency Dispatch, Cell Phone Dead during/after the Earthquake.
- ☒ More of the requirement than coding problem itself.

# Why Software Engineering is Difficult

- ⌘ Why we have so much trouble in software when the software is replacing electromechanical devices?
- ⌘ Same engineering approach can be applied?
- ⌘ Unique engineering problems in constructing complex software
  - ☑ Analog versus discrete state systems
  - ☑ Curse of flexibility
  - ☑ Complexity and Invisible Interface

# Analog vs. discrete State Systems

- ⌘ Computers simulate the behavior of an analog controller
- ⌘ Translation (conversion) of analog to digital form → inaccuracy and complication are introduced
- ⌘ Continuous functions → Discrete functions
- ⌘ Continuous analysis → Numerical analysis
- ⌘ Big problem areas: time (Patriot Missile Battery example), finite-precision arithmetic, concurrency
- ⌘ Physical range (limited) vs. software range (unlimited) → abnormal behavior → difficult to test
- ⌘ Bizarre way of software failure

**Irish Telecom crash due to 'bizarre' software failures**

# Curse of Flexibility

⌘ Easy change of computer function by each change of software – flexible, quick and with low cost → error introduction, complexity

⌘ Mechanical Construction

- ⊞ Governed by mechanical limitation
- ⊞ Laws of dynamics
- ⊞ Nature imposes discipline of design process
- ⊞ Control complexity

⌘ Software Construction

- ⊞ No physical limitation
- ⊞ Enormously complex design is made
- ⊞ Premature construction before full understanding

⌘ Success and Partial success

⊞ S/W:

- ⊞ Difficult to build one that works under all conditions
- ⊞ Easy to build one that works 90% of the time

⊞ Aircraft:

- ⊞ Difficult to build a plane that flies 90% of the time

# Complexity and Invisible Interfaces

## ⌘ Complex system or design

- ☒ Huge number of components

## ⌘ Large number of interfaces → uncontrollable complexity → human mind is unable to comprehend all the conditions of the interactions

## ⌘ Decomposition is very difficult

- ☒ Cf. Cars and airplanes --- built by dividing into parts and units

- ☒ Software, no physical connections, parts, or units

- ☒ Invisible interfaces

- ☒ Discipline and training needed

## ⌘ "Normal Accident" – Charles Perrow



## The Reality We Face

- ⌘ **New scene** in failures and errors due to computer control with software
- ⌘ **Design error** is hard to find
- ⌘ Perfect software is never there, never can be → **Impossible** goal
- ⌘ Other **solutions and methods** must be developed to cope with computer hardware problems and software problems and challenges

# Difficulties in Software



## Quotation from Embedded Share

Code is often not written "application specific" any more, because it is not time or cost efficient. There are many abstractions in code which invariably lead to design flaws, coding flaws and implementation flaws. Also, as with one incident where I work, one of our analog to digital converters became obsolete. The hardware was re-designed to accommodate the new chip and worked great, but the code still had to run on both old and new hardware. The result is even more complex code to fit the needs of a small variation in the hardware. This can also introduce bugs as again, the hardware is designed for one specific task, and the firmware must be altered to allow for this new component.

In all, I don't think it comes down to the tools, attitude, design philosophies or anything else, it is a combination of everything. Let's face it, this stuff is complex, and firmware/software is expected to run on a target platform with minimal design changes. When was the last time you heard someone say "Hey, great job designing that new piece of hardware, now let's design all new firmware for it as well!". I'll tell you, I haven't heard that one before. Usually, we are looking for ways to fit most of the old code into the new hardware to get the product out on time.

# Homework 2

## ⌘ Subject

☑ **“Forgetting How It Used to be Done”**

## ⌘ Assignment Details

☑ Read the handout (in the web page) carefully

☑ Summarize and write a report

☑ Page 1: 1 paragraph executive summary and conclusion (write like a WP staff writer, and remember a pair of scissors)

☑ Pages 2 -3: Description of your talk on the subject

☑ No picture, no photo, text only

## ⌘ Report Writing Method:

☑ Letter size with 1" margin all sides

☑ No cover page

☑ Your name and ID (header)

☑ Times New Roman 11 pt & single space

☑ MS Word format

☑ File naming convention: YourLastName2.doc (for softcopy submission)

## ⌘ Submission Due

☑ Hardcopy:

☑ Softcopy: \*No PDF format, please

# Homework 3

## ⌘ Subject

- ☒ **"Myths of Correctness"**

## ⌘ Assignment Details

- ☒ Read the handout in the web-page carefully

- ☒ Summarize and write a report

- ☒ Page 1: 1 paragraph executive summary and conclusion (write like a WP staff writer, and remember a pair of scissors)

- ☒ Pages 2 -3: Description of your talk on the subject

- ☒ No picture, no photo, text only

## ⌘ Report Writing Method:

- ☒ Letter size with 1"margin all sides

- ☒ No cover page

- ☒ Your name and ID (header)

- ☒ Times New Roman 11pt & single space

- ☒ MS Word format

- ☒ File naming convention: YourLastName3.doc (for softcopy submission)

## ⌘ Submission Due

- ☒ Hardcopy:

- ☒ Softcopy: \*No PDF format, please

# Homework 4

## ⌘ Subject

☒ “Top 10 Worst Software Project Failure”

## ⌘ Assignment Details

☒ No handout

☒ Power Point Slides

☒ Slid 1: Top 10 list **AND** 1 paragraph executive summary and conclusion (write like a WP staff writer, and remember a pair of scissors)

☒ Slides 2 – 11: summary description of each failure (focused on the main reason of the failure:

☒ **Slide 12: Reference source** must be added at the end of the description)

☒ File naming convention: YourLastName4.PPT(for softcopy submission)

## ⌘ Submission Due

☒ Softcopy: \*No PDF format, please

## ⌘ Presentation

☒ R