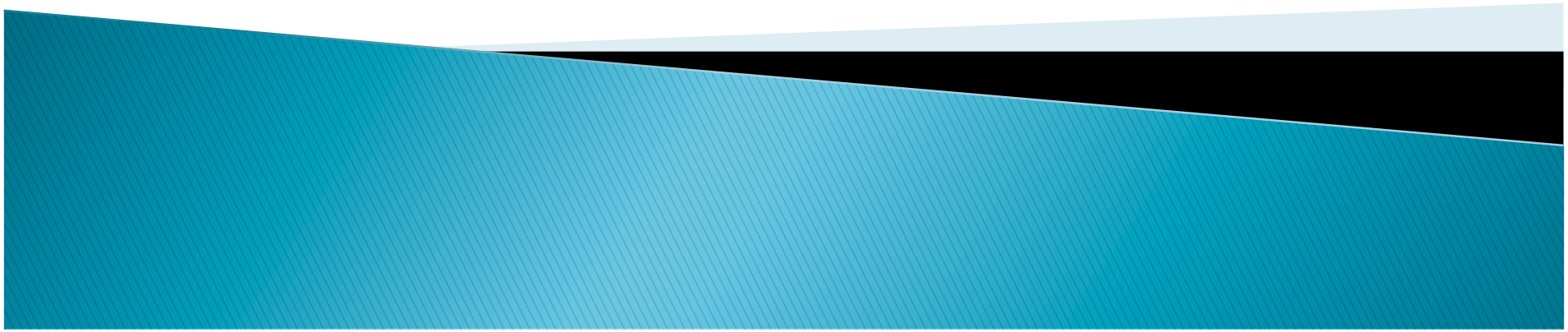
Software Disasters

By: Aditya Bhave

CSCI-5828 Foundations of Software Engineering

03-19-2010



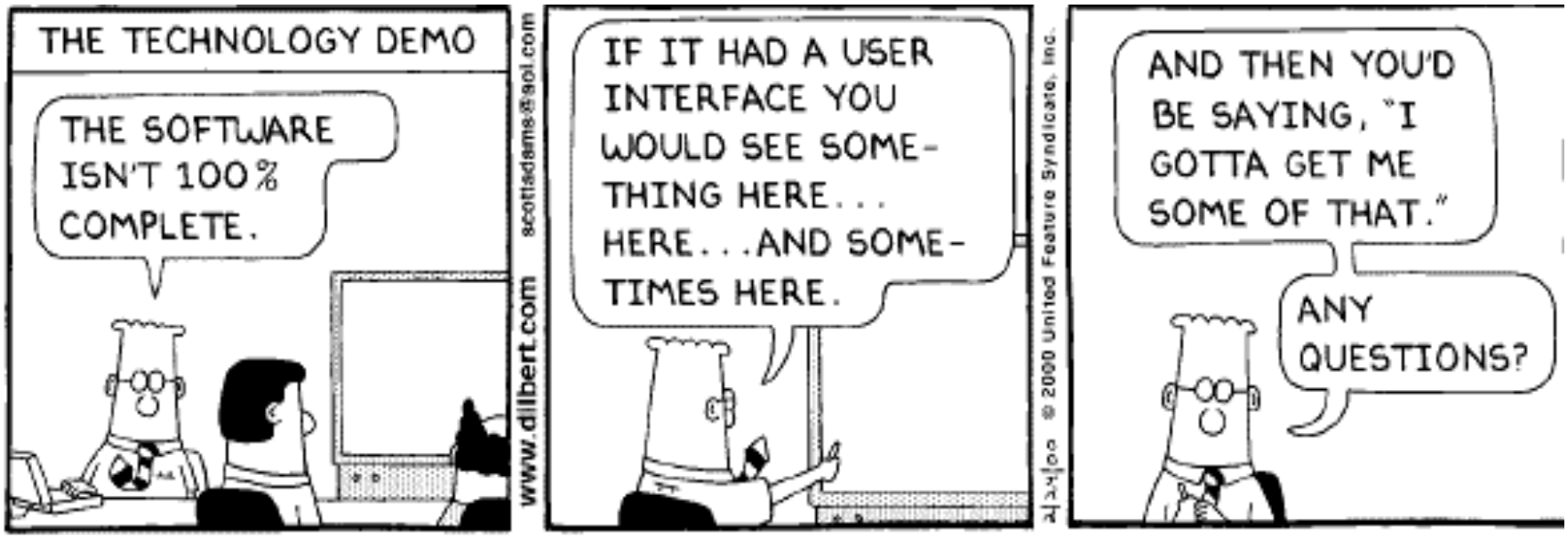
About me

- Embedded systems software background
- Software Engineer
  - Sasken Technologies – multimedia systems device drivers
- Bachelors degree in Electrical engineering
- Currently in my first year of MS in Electrical Engineering.
- Working with Prof. Mark Rentschler (Mech Engr) on medical imaging projects.

Agenda

- Software Disasters
  - What do we mean by disasters ?
  - Main culprits
  - Some examples…
- Case study : DIA baggage handling system
  - Key decisions that went wrong !
- Conclusion
- References

- Dilbert – Scott Adams



- If debugging is the process of removing bugs, then programming must be the process of putting them in ...

– Edsger Dijkstra

Software Disasters

- Caused by
  - Erroneous software
  - Erroneous management techniques
- Consequences :
  - Rework
  - Lost productivity
  - Actual damage
  - Expensive, embarrassing, destructive and deadly

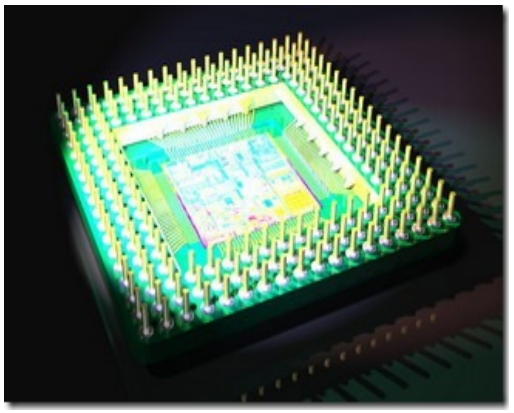
The main culprits…

- Poor
  - Requirements gathering
  - Project planning
  - management methodology
- Inability to understand project technology
- Insufficient expertise –coding errors
  - Example: outsourcing software projects
- Lack of risk management

Some examples …

- Pentium's Long Division (1993)
- Mars Climate Orbiter(1999)
- The Ariane 5 crash (1996)
- AT&T Lines Go Dead (1990)
- Mars Pathfinder(1996-97)

Disaster #1



- Pentium's Long Division (1993)
  - Cost: $475 million,
    corporate credibility
  - Mistakes in dividing floating-point numbers within a specific range
    - Example: 4195835.0/3145727.0 =1.33374
      instead of 1.33382, an error of 0.006%
  - Intel replaced the chips for anyone who complained.
- A flawed division table in the divider in Pentium FPU :
  - missing about five out of a thousand entries

Disaster #2

- Mars Climate Orbiter(1999)
- Cost: $125 million approx.
- Causes:
  - Inconsistent usage of metric system and English system units within the same software
  - Overall process failure to detect such a misuse

Disaster #3

- The Ariane 5 crash (1996):
  - Rocket capable of hurling a pair of three-ton satellites into orbit
  - Project worth ten years and $7 billion approx.
  - Exploded in less than a minute after launch
- Reason
  - small computer program trying to stuff the sideways rocket velocity (a 64-bit number) into a 16-bit space

Disaster #3 Contd…

- Digging deeper:
  - Assuming technical consistency with previous versions
    - Ariane 5 was faster than Ariane 4

      → thus velocity field overflowed
  - The buggy code actually served no purpose after launch !
  - One of the unnecessary “special features” of the system

Disaster #4

- AT&T Lines Go Dead (1990)
  - Cost: 75 million phone calls missed,
    200 thousand airline reservations lost
- One switch suffered a minor mechanical problem
  - Result : Center shut down
  - Effect :
    - This center caused all other stations to shut down
    - Brought down the entire AT&T network for 9 hours.

Disaster #5

- Mars Pathfinder(1996-97)
  - Unconventional landing
  - Total system resets
  - Data losses
- Reason:
  - Priority inversion created for a few tasks
  - Underlying VxWorks not used properly

Disaster #5 Contd…

- Causes of priority inversion:
  - Preemptive priority scheduling of tasks
  - Synchronized access to the “information bus”
  - Low priority meteorological data gathering task
  - Medium priority communications task
  - High priority ISR requiring the information bus
- Action taken :
  - Disaster avoided as the system could be remotely debugged and fixed

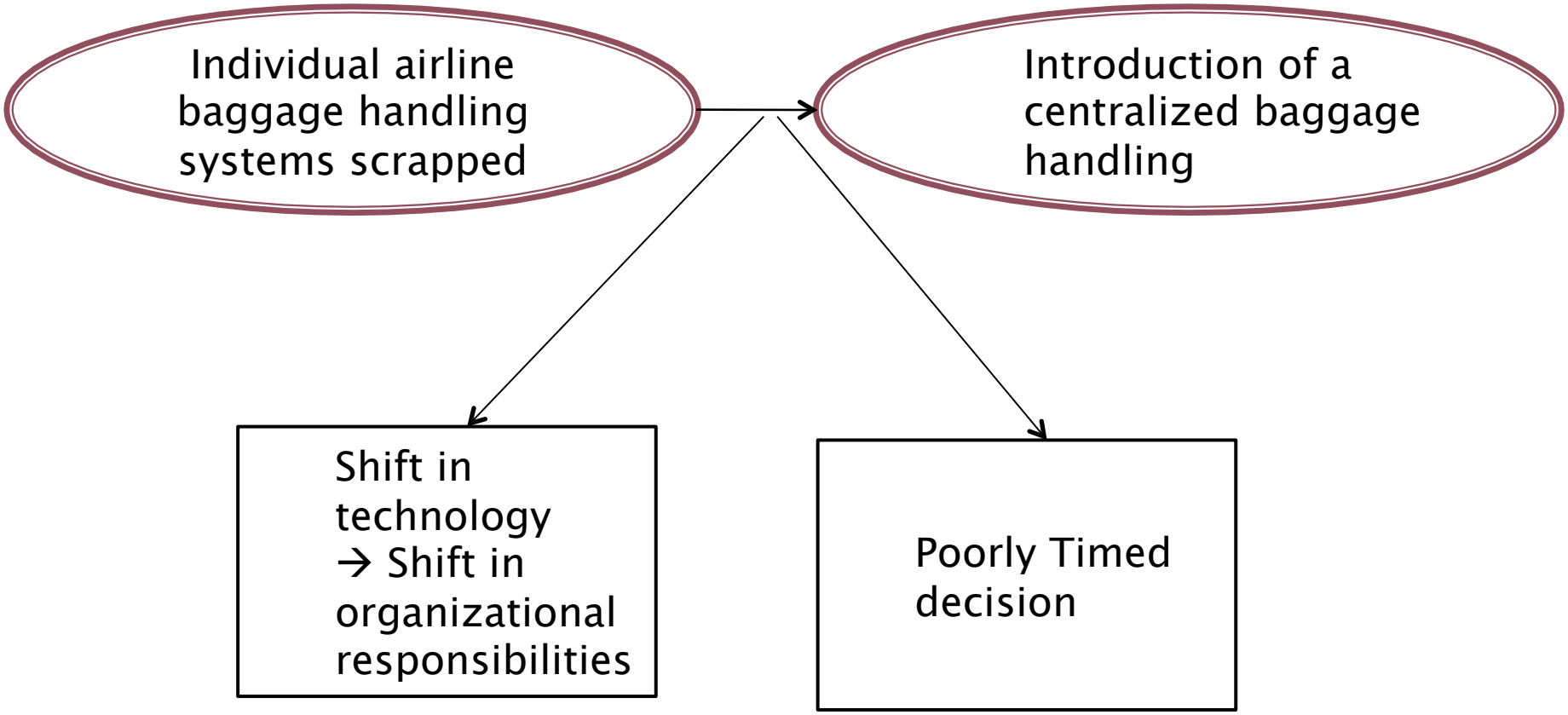
Case Study

- DIA Baggage handling system (1994)
  - Extremely complex and expensive
  - The airport sat idle for 16 months
  - Delay cost: $560M USD approx.
  - $1M monthly maintenance cost
  - The System never functioned properly and was scrapped(2005)

Key Decisions that Led to Disaster

- Change of Strategy
- Why did they still proceed?
- Schedule, scope and budget commitments
- Acceptance of change requests
- Laxity in taking actions
- Architectural and design issues

Change of Strategy



Effect of Change of Strategy

- Failure to
  - link the airport's overall strategy (the goal of having one of the world's most efficient airports) with the sub-strategy of how to build the baggage system
  - ask the critical question of where the responsibility for development of the baggage system needed to be

Why did they still proceed ?

- The 1990 Breier Neidle Patrone Associates report
  - The complexity was too high for the system to be built successfully
- Analysis of the three bids received indicated that none of the vendors could build the system in time for the Oct 1993 opening
- Munich airport experts:
  - The much simpler Munich system had taken 2 full years to build and that it had run 24 / 7 for 6 months prior to opening to allow bugs to be ironed out

Contd…

- Failure of
  - both airport management and BAE's (Boeing) Senior Management team to heed the advice they were receiving
  - the airport's Project Management team to have the BAE proposal and prototype independently reviewed (by a third consultant party)
- Underestimating complexity due to lack of background in related fields, leading to false assumptions

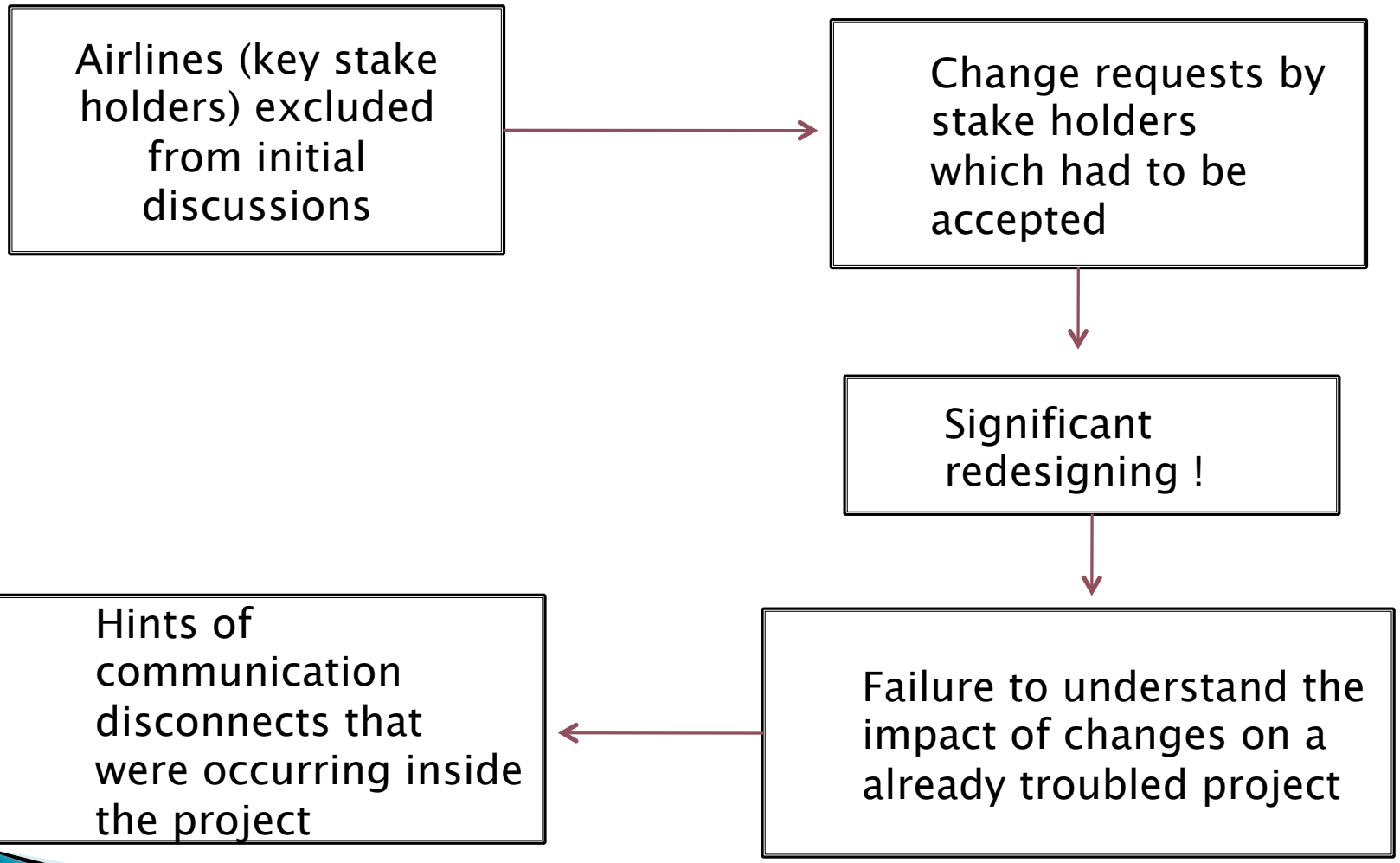
Schedule, scope and budget commitments

- BAE committed to deliver the complete system under a fixed commitments of :
  - Scope
  - Schedule
  - Budget

BAE's top management failed to recognize the level of RISK in this.

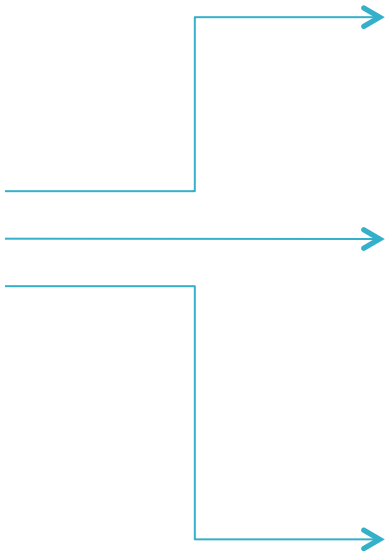
- Hasty decisions:
  - contractual conditions and scope of work were hammered out in just three "intense" working sessions

Acceptance of change requests



Laxity in taking actions

- LATE



Damage control after 4 missed deadlines and 6 months of running behind schedule

Decision to call upon consultants to review the situation

Decision to scrap the automated system and build a manual system

- The highest level had little idea what the true status of the project was

Architectural and design issues

- Design chosen – complex and error prone

100+ individual PCs networked together. Failure of any one of the PCs could result in an outage

Distributed design added to the difficulty of resolving problems when they arose

System kept piling up more and more bags in case of a jam making it worse

Schedule pressure was excessive; making the team settle for the first design they thought of…

Conclusion

The Denver debacle is a template for failure that many other projects have followed. A few of those are:

- The underestimation of complexity
- A lack of planning resulting in subsequent changes in strategy
- Excessive schedule pressure
- Making firm commitments in the face of massive risks and uncertainty
- Poor stakeholder management

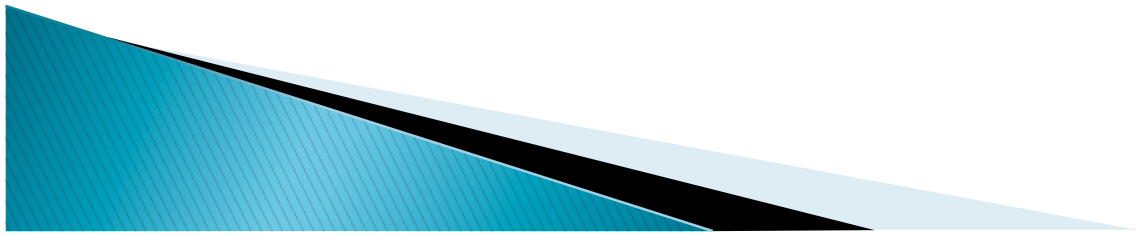
Failures

- Communications breakdowns
- People working in silos
- Poor design
- Failure to perform risk management
- Failure to understand the implication change requests might have
- Lack of management oversight

Failures

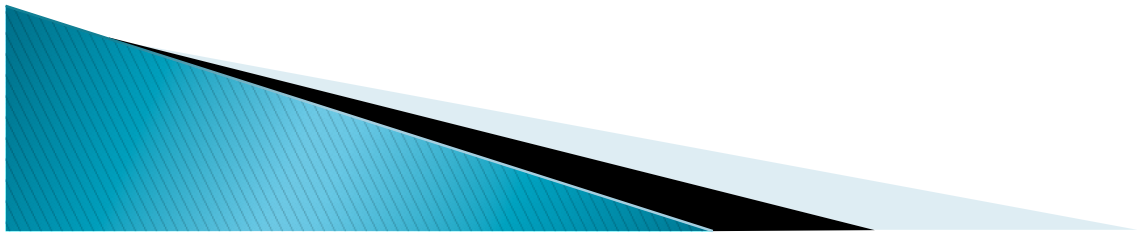
References

- Case Study – Denver International Airport Baggage Handling System – An illustration of ineffectual decision making Calleam Consulting Ltd, 2008.
- http://www.embedded.com/2000/0005/0005feat2.htm
- http://www.around.com/ariane.html
- http://research.microsoft.com/en-us/um/people/mbj/Mars_Pathfinder/Mars_Pathfinder.html
- *Software Runaways : Lessons Learned from Massive Software Project Failures*, Robert L. Glass, Prentice-Hall, 1998.
- Dilbert – Scott Adams http://www.dilbert.com/



Typical Coding Errors…

- Typo errors
  - A simple x==2 instead of x=2
  - Typing a!=b instead of a=!b
- Assumptions about underlying hardware and OS
  - ```
    #include <stdio.h>
    #include <time.h>
     int main()
     {
        time_t biggest = 0x7FFFFFFF;
        printf("biggest = %s \n", ctime(&biggest) );
        return 0;
     }
    ```
    Will the output vary for PC's and UNIX systems?
  - ```
    unsigned int zero = 0;
    unsigned int compzero = 0xFFFF; //1's complement of zero
    ```
    The right way to do this ?



And some more …

- ```
  #define MIN(A,B) ((A) < = (B) ? (A) : (B))
  least = MIN(*p++, b);
  ```
  Whats wrong with this approach ?

- ```
  void foo(void) {
  unsigned int a = 6; int b = -20;
  (a+b > 6) ? puts("> 6") : puts(" < = 6");
  }
  ```
  - What would the output be ?
    - Understanding the integer promotion rules in C

- Using Non-portable coding techniques: Bit fields for example

Back