# TTIT62 Real-time Process Control

## Lecture 7: Dependability

**Simin Nadjm-Tehrani**

Real-time Systems Laboratory

Department of Computer and Information Science
Linköping university



## Dependability

- If a system has to provide its service in real-time, it has to provide a service at all!
- How do things go wrong?
- Why?
- What can we do about it?
- Basic notion in **dependable** and **fault-tolerant systems** (sv. pålitliga system)



## Is not this typical?

**January 25, 2009:**

- the Mars rover Spirit failed to perform the day's instructions, and the rover also may have suffered amnesia, as it failed to record its activities in its non-volatile memory. Spirit appears to be operating normally now, but the rover's controllers are still not sure what happened. "At this time, we don't know whether the problem was a one-time event--whether it was induced by a cosmic ray--or whether it might be an indicator of aging hardware," says NASA project manager John Callas.



## Failures are not unusual!

- **TCAS** is a system for avoidance of mid-air crashes between passenger airplanes. On 3rd February 1994 two commercial airplanes came as close as 1.6 km from each other over Oregon in USA.



## FT - June 16, 2004

- "If you have a problem with your Volkswagen the likelihood that it was a software problem is very high. Software technology is not something that we as car manufacturers feel comfortable with."

Bernd Pischetsrieder, chief executive of Volkswagen



## October 2005

- "Automaker Toyota announced a recall of 160,000 of its Prius hybrid vehicles following reports of vehicle warning lights illuminating for no reason, and cars' gasoline engines stalling unexpectedly."

*Wired 05-11-08*

- The problem was found to be an embedded software bug

## February 2, 2004

- Angel Eck, driving a 1997 Pontiac Sunfire found her car racing at high speed and accelerating on Interstate 70 for 45 minutes, heading toward Denver
- ... with no effect from trying the brakes, shifting to neutral, and shutting off the ignition.



## Even worse

- Between June 85 - January 87 six patients in USA and Canada got very high doses of radiation from a cancer treatment system **Therac 25**. The doses varied from 15.000-20.000 radiation units instead of the normal levels (ca 200 units). Three of the patients died due to overdose and the following complications.



## What is dependability?

*Property of a **computing system** which allows reliance to be justifiably placed on the service it delivers.*

[Avizienis et al.]



## Attributes of dependability

IFIP WG 10.4 definitions include:

- **Safety**: non-occurance of harm to people and environment
- **Availability**: the readiness for usage
- **Confidentiality**: non-occurance of unauthorized disclosure of information
- **Reliability:** continuity of correct service



## Reliability

[Sv. Tillförlitlighet]

Means that the system (functionally) behaves as specified, and does it continually over measured intervals of time.

Typical measure in aerospace: $10^{-9}$

Another way of putting it: MTTF - One failure in $10^9$ flight hours.



## Faults, Errors & Failures

- **Fault**: a defect within the system or a situation that *can* lead to failure
- **Error**: manifestation (symptom) of the fault - an unexpected behaviour
- **Failure**: system not performing its intended function

## Examples

- Year 2000 bug
- Bit flips in hardware due to cosmic radiation in space
- Loose wire
- Air craft retracting its landing gear while on ground

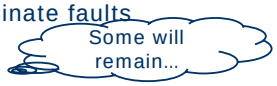
Effects in time:

Permanent/ transient/ intermittent



## Fault ⇒ Error ⇒ Failure

- Goal of system verification and validation is to eliminate faults




- Goal of safety/risk analysis is to focus on important faults
- Goal of fault tolerance is to reduce effects of errors if they appear - *eliminate or delay failures*



## Fault tolerance

- Means that a system provides a minimal acceptable function
  - Even in presence of (a class of) faults
  - During a period defined by certain model assumptions
- Foreseen or unforeseen?



## External factors

The film …



## Means to achieve

… dependability according to [IFIP 10.4]:

1. Fault prevention
2. Fault removal
3. Fault tolerance
4. Fault forecasting



## On-line fault management

- Fault detection
  - By program or its environment
- Fault tolerance using redundancy
  - software
  - hardware
  - Data
- Fault containment by architectural decisions

## Redundancy

From D. Lardner: Edinburgh Review, year 1824:

*"The most certain and effectual check upon errors which arise in the process of computation is to cause the same computations to be made by separate and independent computers*; and this check is rendered still more decisive if their computations are carried out by different methods."*

* *people who compute*



## Static Redundancy

Used all the time (whether an error has appeared or not), just in case…

- SW: N-version programming
- HW: voting systems
- Data: parity bits, checksums



## Dynamic Redundancy

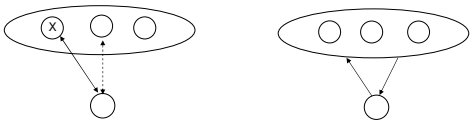
Used when error appears and has to be treated

- SW: Recovery methods
- HW: Switching to back-up module
- Data: Self-correcting codes
- Time: Re-computing a result



## Server replication models

- Primary backup
- Active replication






## Error recovery

Backward:

- Roll back the system to a safe state that was reached before the error showed up (when did it show up?)
- Restart with an alternative module (how are results of earlier module's computations affected?)



## Error recovery

Forward:

- "Treat the error" and continue as if nothing has happened
- Redundancy used where the "treat" is

## Dependability & Real-time

- How do methods for increasing dependability affect real-time performance?

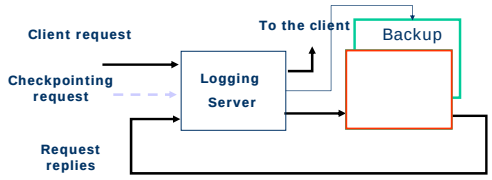


## Dependability –- Real-time

- Faults and their treatment affect timing
- Detection & Removal:
  - Consider faults when performing timing analysis, watchdogs, monitors
- Tolerance:
  - Redundancy in time (transient faults)
  - Redundancy in space (permanent faults)
  - Both static and dynamic redundancy have impacts on timing
- Forecasting: Overload management



## Checkpointing




Challenge: to checkpoint often enough but not too often

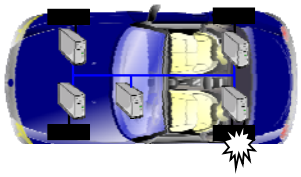


## Dependability & Real-time

- How important is determinism in the time domain for achieving dependability?



## Brake-by-wire





## Support for time determinism

- Hardware
  - I/O devices, memory access, cache
- Operating system
  - Interrupts & context switches, drive routines
- Real-time communication
  - Bounded communication in TDMA buses
- Programming language
  - Explicit treatment of faults
  - Inclusion of fault-treatment code in timing analysis

## Exception management

- Every program should test for validity range of its computations
  - so, why exceptions?
- Mechanism to support recovery models in programming languages
  - e.g. Ada´s exceptions support backward error recovery via the termination model
  - See example program



```
package Temp_Control is
subtype Temperature is integer range 0..100;
     Sensor_Dead, Actuator_Dead: exception;
...
end Temp_Control;
package body Temp_Control is

     procedure Set_Temp(...) is
     begin
          -- set new value for actuator
          if No_Response then
               raise Actuator_Dead
          end if;
     end Set_Temp;
...
end Temp_Control;
```



```
     function Read_Temp return Temperature is
     begin
          -- read sensor value
          if No_Response then
               raise Sensor_Dead
          end if;
          -- return the value
          exception
               when Constraint_Error =>
               -- too high a temperature
               -- take appropriate action
     end Read_Temp;
...
end Temp_Control;
```



```
begin
     -- initialize
     Set_Temp(...);
     when  Actuator_Dead =>
          -- take some action
     ...
end Temp_Control;
```