Dependable Systems

Dr. Peter Tröger

Dependable Systems Course

- Introduction into theoretical foundations, common building blocks and example implementations for dependable IT components and systems
- Recommend readings: Collection from standard literature
- Open discussion during the lecture, relation to other courses
- Knowledge gained from the lectures has to be applied in practical project work
  - One topic per student group (2-3 persons)
  - Practical experiments with fault tolerance technology
  - Presentation during the lecture, written report (pass / fail)
- Slides, list of project topics and supervisors on course home page

  https://www.dcl.hpi.uni-potsdam.de/teaching/depend/

Course Topics

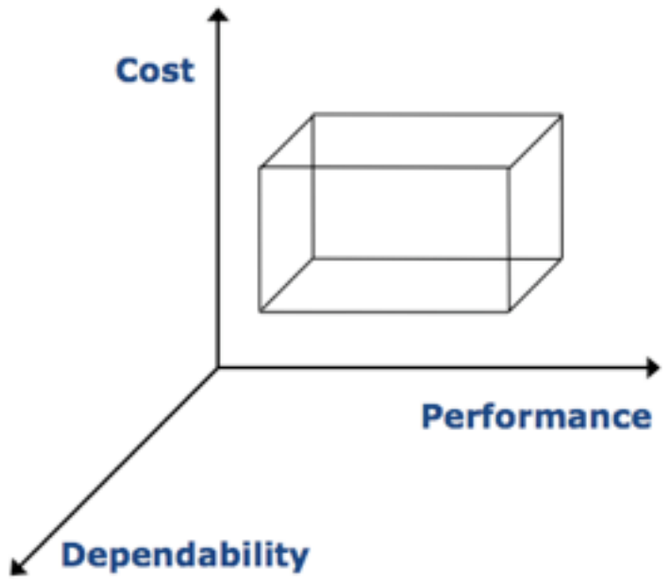
- Definitions and metrics
  - Fault, error, failure, fault models, fault tolerance, fault forecasting, ...
  - Reliability, availability, maintainability, error mitigation, damage confinement, ...
  - Reliability engineering, reliability testing, MTTF, MTBF, failure rate, ...
- Fault tolerance patterns
  - N-out-of-M, voting, heartbeat, ...
- Analytical evaluation
  - Reliability block diagrams, fault tree analysis, ...
  - Petri nets, Markov chains
  - Root cause analysis, risk analysis and assessment

Course Topics

- Software dependability
  - Fault-tolerant programming: simplex, recovery blocks, checkpointing, roll-back, ...
  - Fault-tolerant distributed systems: Transactions, consensus, clocks, ...
  - Testing
- Hardware dependability
  - Failure rates, testing, redundancy approaches
- Advanced approaches
  - Autonomic computing, rejuvenation
  - Online failure prediction
  - Performability

Dependability

- **Umbrella term** for **operational** requirements on a system
  - IFIP WG 10.4: "*[..] the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers [..]*"
  - IEC IEV: "*dependability (is) the collective term used to describe the availability performance and its influencing factors : reliability performance, maintainability performance and maintenance support performance*"
  - Laprie: „ *Trustworthiness of a computer system such that reliance can be placed on the service it delivers to the user* “
- Adds a third dimension to system quality
- General question: How to deal with unexpected events ?
- In German: ‚Verlässlichkeit‘ vs. ‚Zuverlässigkeit‘



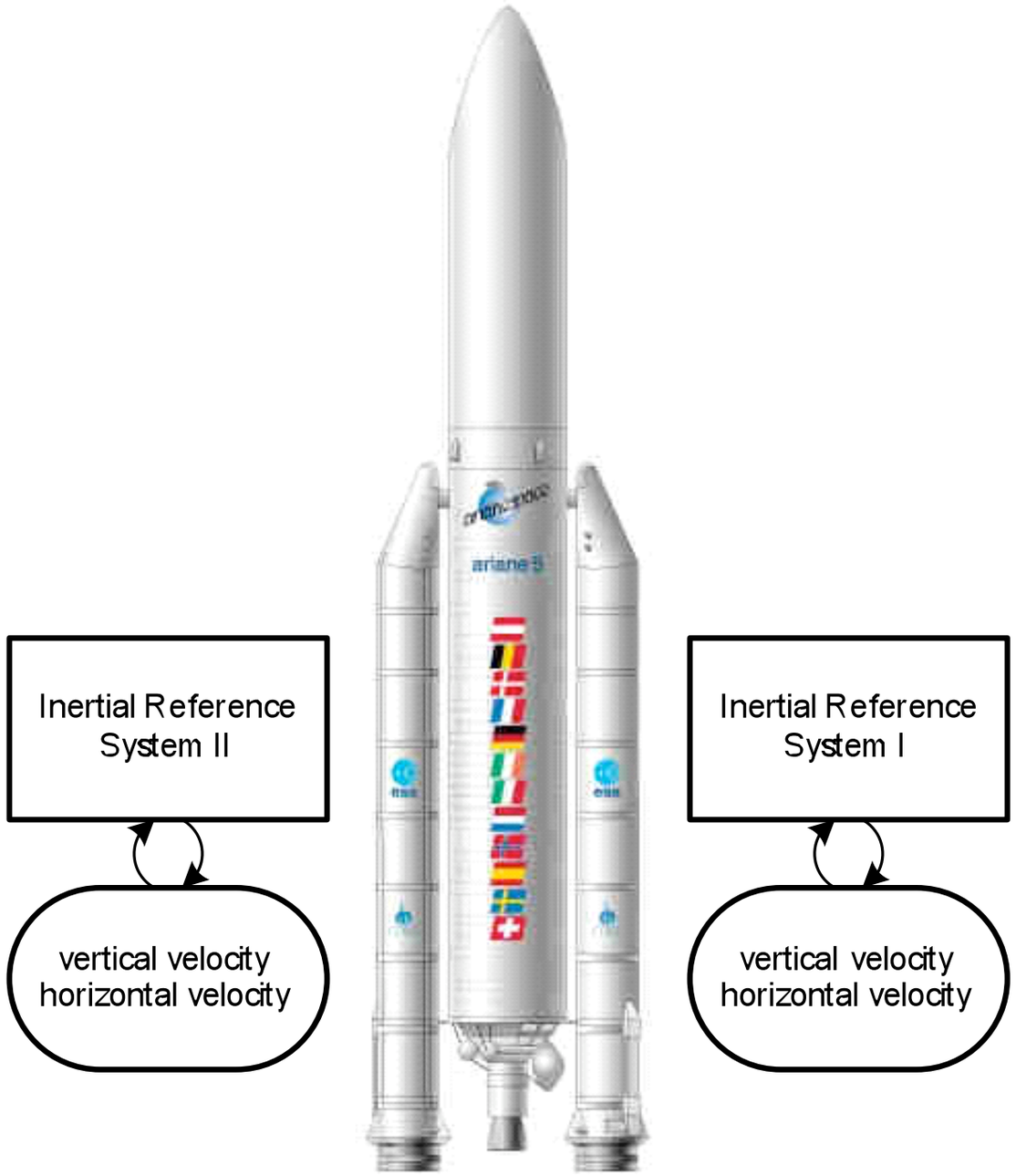
Dependability Examples

- A segmentation **fault** can lead to a system **failure.**
- A **faulty** controller in an RAID array lowers the **reliability** of the storage facility.
- The measurable **availability** of a web server depends on the **fault tolerance** capabilities in all its components.
- A database **fault** creates an **erroneous** state in the authentication component, which can lead to a system **failure** influencing the system **confidentiality.**
- The **integrity** of electronic voting systems depends on the complete absence of **failures** in the voting and the calculation procedures.
- **Fault forecasting** can support the **maintainability** of a system.
- Testing can only prove the presence of **faults**, not their absence.
- Our service level agreement guarantees an **availability** of 99.99%.

Importance of Dependability for Business

- Average costs per hour of downtime (Gartner 1998)
  - Brokerage operations in finance: $6.5 million
  - Credit card authorization: $2.6 million
  - Home catalog sales: $90.000
  - Airline reservation: $89.500
- 22-hour service outage of eBay on June 6th 1999
  - Interruption of around 2.3 million auctions
  - 9.2% stock value drop, $3-5 billion of lost revenues
  - Problems blamed on Sun server software
- Defense systems, air traffic control, banking, telephone system, online games, ...

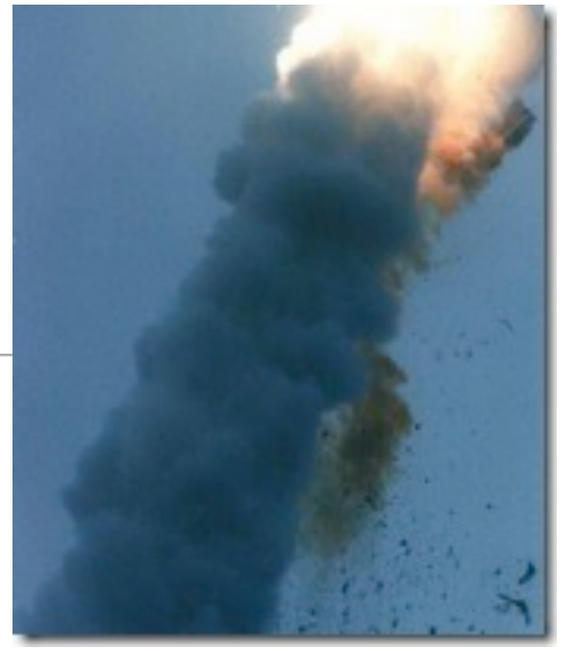
Example: Ariane 5




```
declare
    v_veloc_sensor:     float;      -- 64 bit floating point
    h_veloc_sensor:     float;
    v_veloc_bias:       integer;    -- 16 bit signed integer
    h_veloc_bias:       integer;
begin
    sensor_get (v_veloc_sensor);
    sensor_get (h_veloc_sensor);

    v_veloc_bias := integer (v_veloc_sensor);
    h_veloc_bias := integer (h_veloc_sensor);
exception
    when others => use_irs1();
end;
```

Example: Ariane 5



- June 4 1996, self-destruct firework for $500 million
- Initiated because boosters were ripping from the rocket
  - Reasoned by abrupt course correction for a wrong turn that never happened
  - Reasoned by on-board computer reaction on inertia system data
    - Was not data, but a memory dump due to an overflow error
    - Identical inertia backup system had shut down milliseconds before
- Intertia system from Ariane 4, numbers were never expected to get that high
  - No test of full system due to budget limits
- System was running after lift off for no special purpose
  - Possibility for fast restart with brief hold in the countdown

Example: New York Stock Exchange

- Credit Suisse trading desk, Nov 14th, 2007
  - Automated trading algorithm went mad
  - Reasoned by misplaced double click in the trading software UI
  - Hundreds of thousands of bogus order cancellations
  - DoS nature of fault caused NYSE to freeze up - $150.000 fine

Example: Therac 25

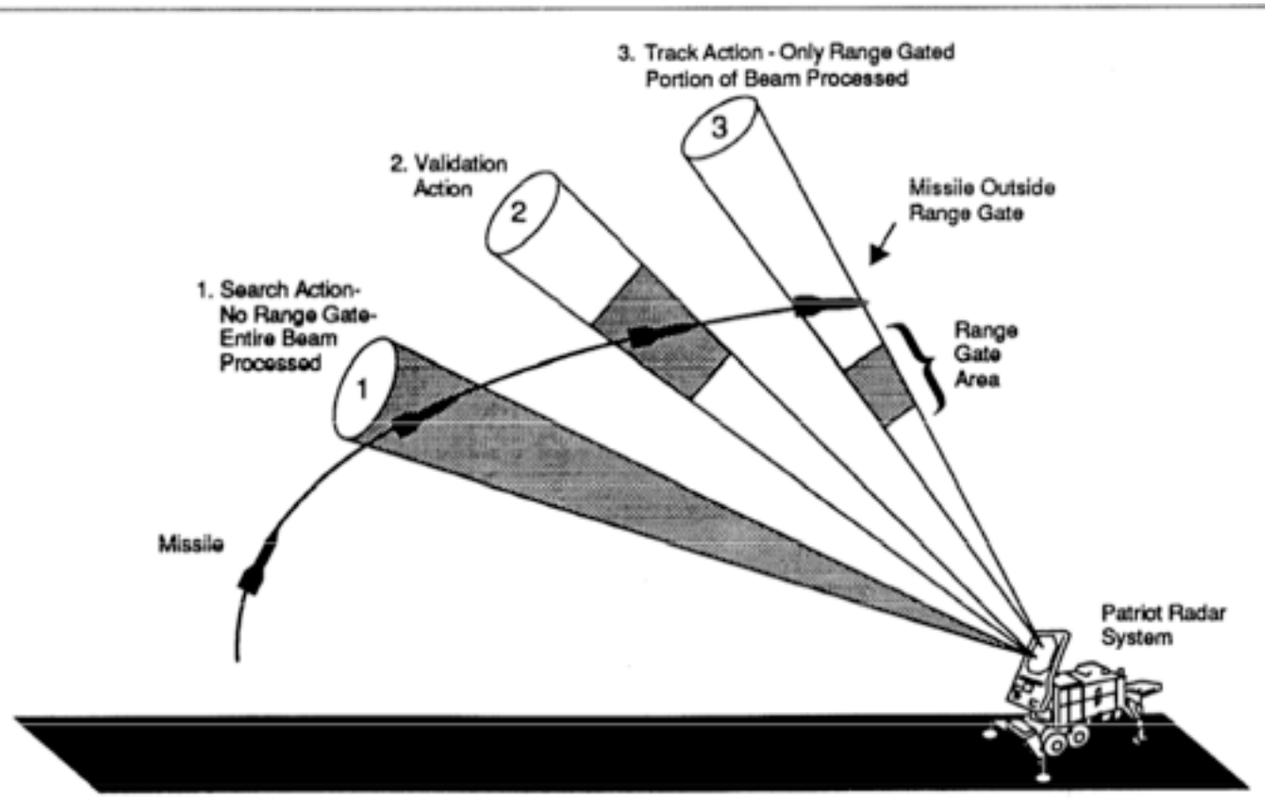
- Radiation therapy engine, involved in 6 accidents between 1985 and 1987
- Two operation modes: High-power spreaded beam, or low power electron beam
- Accident: High-power mode without spreader plate activated
  - Harmful operation not detected due to software flaws
  - Institutional issues: No independent software review, no risk analysis for failure modes, no test of hardware / software assembling
  - Engineering issues: No hardware interlock, software re-use from older models, no sensors for hardware check, arithmetic overflow in the check routines
  - No investigation of single modules

Example: Mars Pathfinder

- Successful landing on Mars at July 4th, 1997
- Spontaneous resets after landing
  - VxWorks operating system, real-time scheduling of data streams and commands
  - Classical priority inversion - long-running medium-communication task, low-priority meterological task, reset triggered by watchdog
  - Antennas performed too good, unexpected timing behaviour in data transmission
  - Problem reproduced with identical Pathfinder on earth
  - Semaphore had priority inheritance protection, which was switched off for performance optimization

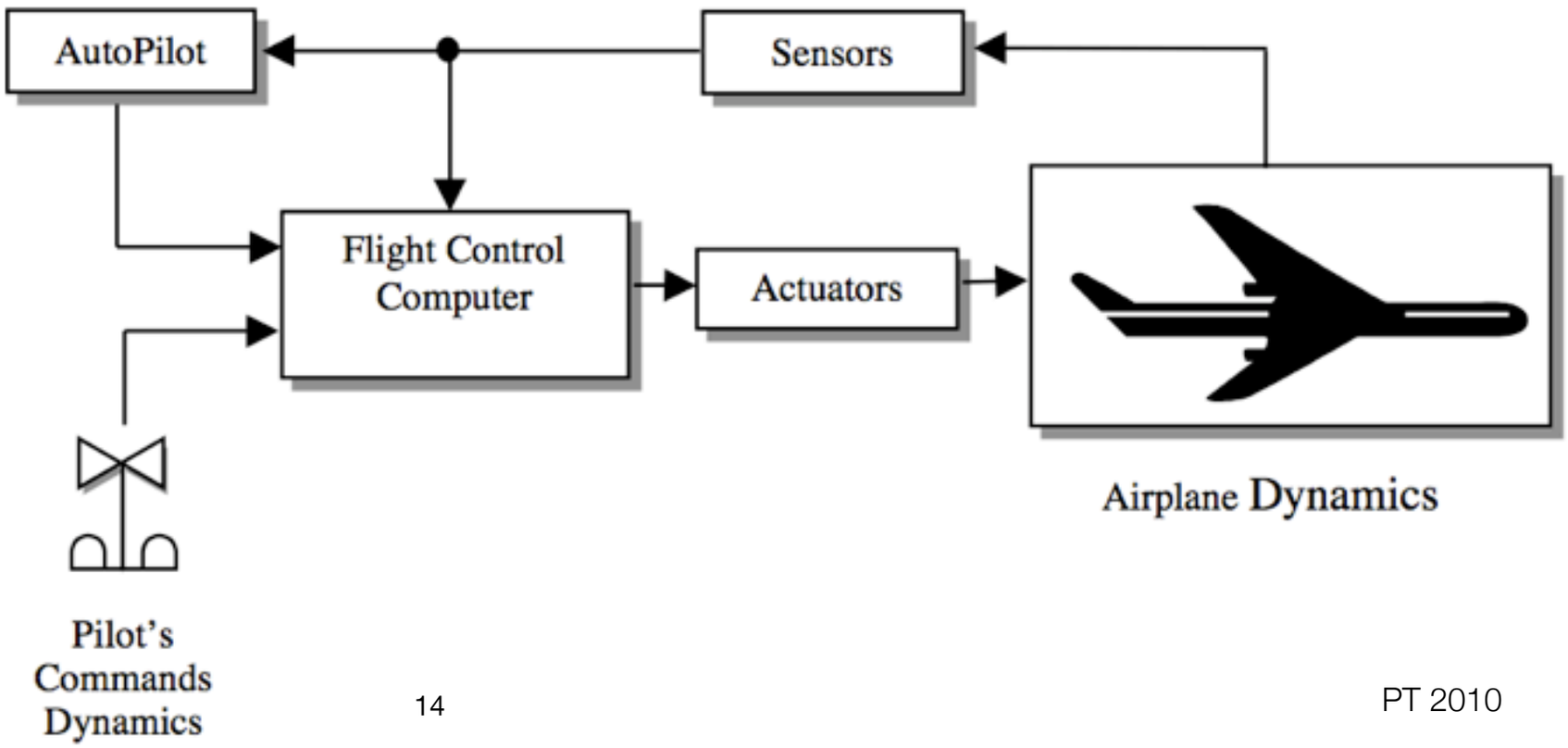
Example: Patriot Missile Launcher

- Mobile missile launcher, designed for a few hours of operation
- February 1991 - Battery in Dharan, Saudi Arabia failed to intercept Scud missile
  - Software aging problem in system‘s weapon control computer
    - Target velocity and time demanded as real values, stored as 24-bit integer
    - Inaccurate tracking computation due to overlong operation (> 100 hours)
    - Modified software reached the base one day after the accident
    - Missile launcher was never designed for Scud defense operation



Example: Boeing 777 Flight Controller

- Fly-by-wire system, requirements
  - Probability 1x10-10 for degration between ‚MIN-OP configuration‘
  - ‚Never give up‘ redundancy management strategy
  - Mean-time-between-maintenance-actions: 25.000 operating hours



Example: Boeing 777 Flight Controller (Buus et al.)

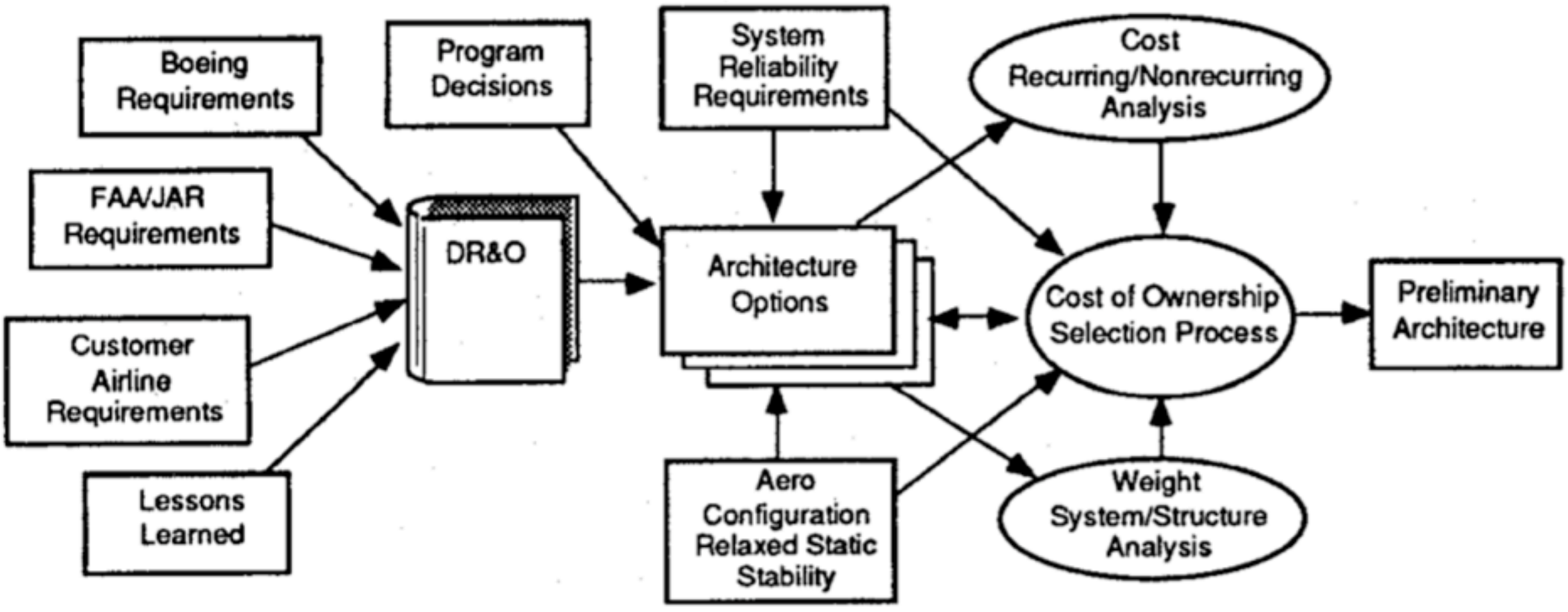


Fig. 2. System architecture design drivers and selection process.

Example: Air France AF 447 (May 2009)

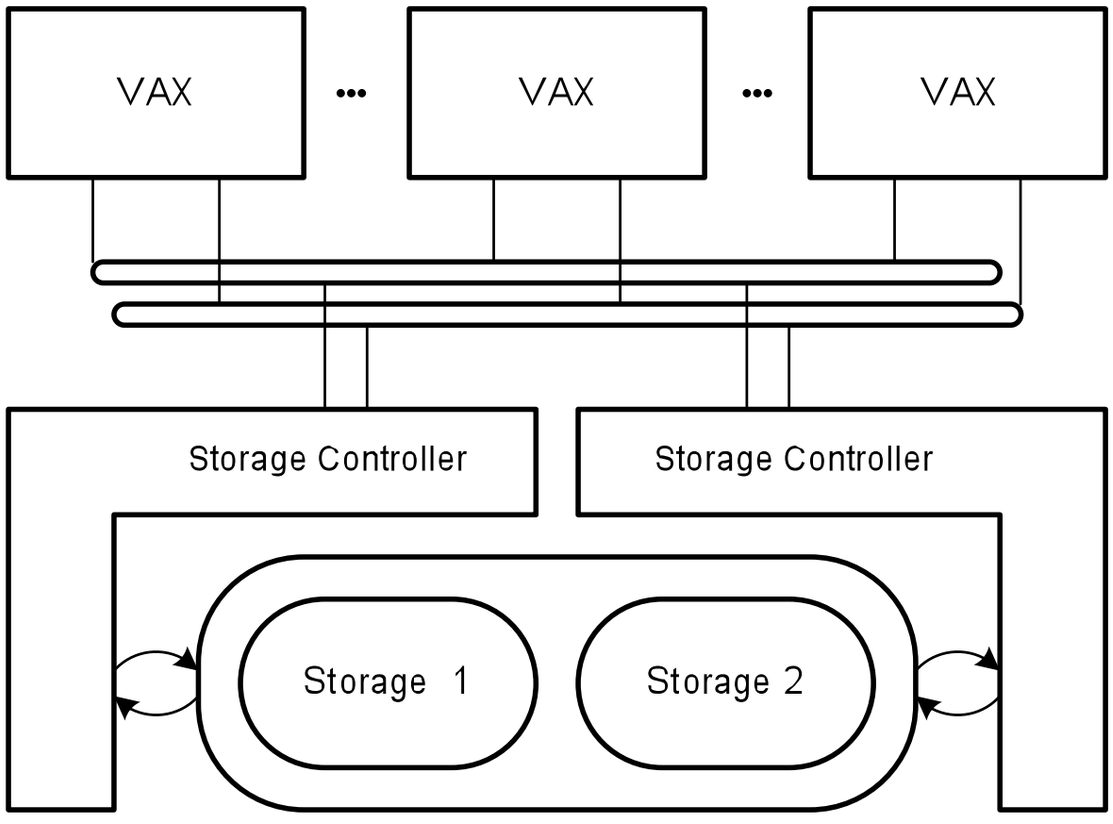
- Airbus A330 flight at 10.000m through turbulent weather conditions
- Ice crystals trigger malfunction in Pitot air pressure sensors
  - Stream of air normally used for speed computation
  - Flight speed instruments shut down, followed by auto pilot
  - Demanded manual angle and thrust level computation by pilots
  - Flight computer switches to emergency mode (‚alternate law 2‘)
    - Still operational (‚fly-by-wire‘), but different flight feeling
  - Pilots lost physical control over machine, restart of flight computer not successful
- Specification for sensors not feasible (written in 1947)
  - Sensor must be operational till -40 degrees / 9000m height

Example: IBM S/390 Outage Sources

- Examination of 16 managed mainframe systems over 6 months (Pfister, 1990)
- 92% planned outages, 8% unplanned outages
  - 32% database backup
  - 24% database reorganization
  - 14% system software maintenance
  - 12% network
  - 8% applications
  - 8% hardware
  - 2% other

Example: VAX Hardware Redundancy

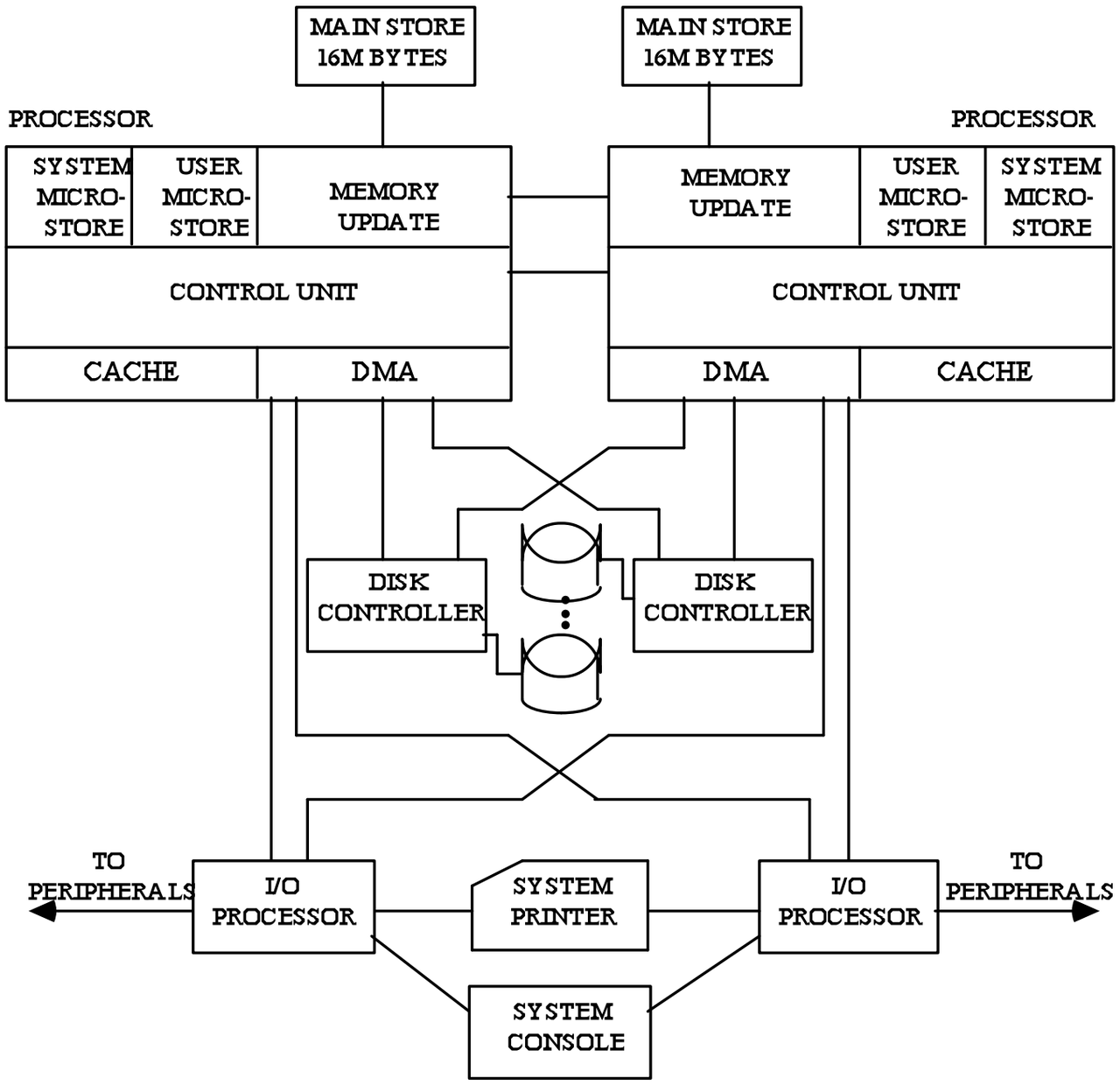
- VMS operating system on VAX hardware, since mid-1970‘s, outgrow of PDP-11
- VAXft in the early 1990‘s



Example: AT&T Electronic Switching System ESS1A

- Downtime for entire system not more than 2 hours over 40 years life
  - System outage < 3 minutes / year
  - 100% availability from user perspective
- Goal achieved, two minutes of down time [Malek]
  - 24 sec hardware faults, 18 sec software problems, 36 sec procedural errors, 42 sec recovery problems
- Full duplication of all critical components
- Automated error detection on hardware and software level
  - Replication checks, timing checks, coding checks, internal checks

Example: AT&T Electronic Switching System ESS1A



The architecture of the AT&T 3B20 Duplex system

(C) Malek

Example: Amazon Cloud Outages

- 2006 - Elevated levels of authenticated requests from multiple users
  - Request volumes are monitored, but cryptographic overhead was not considered
  - Overload on authentication services - performs request processing and validation
  - S3 service (cloud storage service) became inaccessible, but is the only storage facility supported for Amazon-hosted virtual machines
  - Took 3,25 hours to solve the problem - live fixing, started at 3am
- No SLA promises ever, but trust in cloud-only hosted web sited dropped dramatically
- 2009 - 19 hours of outage for bitbucket.org
  - Code hosting service, relies on EC2 and block storage
  - UDP / TCP flooding attack influenced also access to storage facility

Example: Nature vs. Technology