Considerations for Architecting High Reliable systems

Jacques Prinsloo

Kairos Virtual Instruments
Kairos background

Owners

Jacques Prinsloo
• Kairos Sales Manager / Partner
• NI ISR/FSE – 8 years

John Gush
• Kairos Technical Manager / Partner
• NI AE – 5 years

Andries Bolleurs
• Kairos Operational Manager / Partner
• Kairos Control Systems - Owner
Kairos VI info

- Est. 2010
- Systems Integrator
- Registered NI Alliance Member
- 2x Certified LabVIEW Programmers
- 4x Software Developers
- Consulting, s/w Development, Hardware
- Not more than 20% income from specific client
Clients

MEGCHEM

SABS

DENEL

AEROSUD AVIATION (Pty) Ltd

CSIR

VERTEX Automation (Pty) Ltd
Variety of expertise
Kairos CS info

• Est. 2006
• Systems Integrator
• PLC & SCADA based systems
• Approved Siemens Partner
• 2012 prize for best WinCC/TI Portal project
• Mining, Feedmills, Food & Beverage
• 8x Software Engineers
Agenda

• System Dependability
• System Reliability
• System Availability
• Hardware Considerations
• Software Considerations
• Examples
RELIABILITY

Even the Empire had material problems... stupid "Lowest Bidder" contracting rules!
System Dependability

*Dependability* is defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers. The service delivered by a system is its behavior as it is perceptible by its user(s); a user is another system (human or physical) interacting with the former.

Dependability has these Attributes:

- **Reliability**
- **Availability**
- Robustness
- Safety
- Confidentiality
- Integrity
- Maintainability
In general, **reliability** (systemic def.) is the ability of a person or system to perform and maintain its functions in routine circumstances, as well as hostile or unexpected circumstances.

"Reliability is, after all, engineering in its most practical form"

as once stated by James R. Schlesinger, Former US Secretary of State for Defense. [3]
Reliability may be defined in several ways:

- The idea that a system is fit for a purpose with respect to time.
- The capacity of a **designed** system to perform as required over time.
- The capacity of a **maintained** system to perform as required over time.
- The resistance to failure of a system over time.

**Every Reliable system has a Program plan**

- A reliability program plan is used to document exactly what "best practices" (tasks, methods, tools, analyses and tests) are required for a particular (sub)system, as well as clarify customer requirements for reliability assessment.
- A Reliability Program Plan may also be used to evaluate and improve Availability of a system.
Reliability

The probability of NOT failing in a particular environment for a particular mission time.

\[ R(t) = e^{-\lambda t} \]

- \( R(t) \) is the probability of success
- \( t \) is the “mission time” or in other words, the time the system must be executing without an outage
- \( \lambda \) is the constant failure rate over time (N failures/hour)
- MTTF (Mean Time To Failure) is \( 1/\lambda \)
$\lambda$ (Failure Rate) Over Time

“Bathtub Curve”

Reliability Testing
“Probability of not failing during its Useful Life”

Durability Testing
“How long will it last”
Factors That Affect R(t)

- Technology
- Design
- Components Used
- Manufacturing
- People
- Customer Assembly
- Usage
- Dirt

- Dirty Power
- Power Cycles
- Voltage Transients
- Temperature
- Humidity
- Shock
- Vibration
- Chemicals
- etc...
How To Improve Hardware:

Reliability:  *Make it tougher!*
- Fault Prevention
  - Robust designs that leverage derated parts
- Fault Removal
  - Test beyond the specs

Availability:  *Repair it faster!*
- On-Site Spares
- Fault Monitoring and Diagnostics
- Redundancy
Burn-In - Useful Life - Wearout

Driven by three very different physical failure domains

Pre-release Testing - Useful Life (w/upgrades) - Obsolete

Driven by effectiveness of software defect detection and repair processes over the span of many upgrades

- As complexity moves to SW so do the reliability issues
- Industry standard is that there are 10:1 SW failures to HW failures
Factors That Affect SW R(t)

- Technology
- Design
- Components Used
  - Manufacturing
  - People
  - Customer Assembly *(Customer Code)*
- Usage
- Dirt
- Dirty Power
- Power Cycles
- Voltage Transients
- Temperature
- Humidity
- Shock
- Vibration
- Chemicals
- etc...
How To Improve Software:

Reliability:  *Get the bugs out!*

- Fault Prevention and Removal (Review & Test)
  - Unit Testing
  - System Testing
  - Customers need to test their application

- Fault Tolerance

Availability:  *Reset it faster!*

- Fault Monitoring and Diagnostics
- VERY-FAST reset capability
# Differences in HW and SW Reliability

<table>
<thead>
<tr>
<th>Attributes</th>
<th>HW</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cause of defects</td>
<td>Dev &amp; Mfg</td>
<td>Dev</td>
</tr>
<tr>
<td>Complexity over time</td>
<td>Stays the same</td>
<td>Grows with updates</td>
</tr>
<tr>
<td>Wear-out</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Repair</td>
<td>Expensive</td>
<td>Low-cost</td>
</tr>
<tr>
<td>Affected by environment</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>HALT/HASS testing</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Standardized components</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Availability

*Availability* of a system is the percentage of time that a system is available for service.

Availability = \( \frac{\text{Uptime}}{\text{(Uptime + Downtime)}} \) = \( \frac{\text{MTTF}}{\text{(MTTF + MTTR)}} \)

- MTTF (Mean Time To Failure) represents the expected time for a single failure to occur.
- MTTR (Mean Time To Repair or Mean Time to [operational] Restoration) represents the expected amount of time to repair the system and restore it to an operational state, i.e., the duration of the outage.
Availability

• You assume system failures or outages will occur
  (of course you want to limit the number of failures as much as possible)

• The goal is to limit the down time, thus make the MTTR (Repair Time) as small as possible.
  1. Detect failure
  2. Diagnose the failure
  3. Repair the system
  4. System is back in operation
MTBF vs. MTTF

• MTBF (Mean Time Between Failure)
  – Assumes repair and reuse
  – Commonly used for HW that is cost effective to repair

• MTTF (Mean Time To Failure)
  – Assumes that you do not repair and reuse
  – Commonly used for HW that is not cost effective to repair
  – Commonly used for SW because after a reset you start new

• We will use MTTF for this session, because:
  – Our customers frequently replace old HW with new HW after a HW failure
  – System reliability is mainly affected by SW
  – Thus, for our purposes we can assume MTBF = MTTF
## Availability, The Game of 9s

<table>
<thead>
<tr>
<th>Availability Class</th>
<th>Availability</th>
<th>Unavailability Range (downtime)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Unmanaged</td>
<td>90% (1 nine)</td>
<td>36.5 – 38.3 days/year (52,560 – 55,188 min/year)</td>
</tr>
<tr>
<td>2 – Managed (good web servers)</td>
<td>99% (2 nines)</td>
<td>3.65 – 5.48 days/year (5,256 – 7,884 min/year)</td>
</tr>
<tr>
<td>3 – Well-managed</td>
<td>99.9% (3 nines)</td>
<td>8.76 – 13.14 hours/year (526 – 788 min/year)</td>
</tr>
<tr>
<td>4 – Fault Tolerant (better commercial systems)</td>
<td>99.99% (4 nines)</td>
<td>52.56 – 78.84 minutes/year</td>
</tr>
<tr>
<td>5 – High-Availability (High-reliability products)</td>
<td>99.999% (5 nines)</td>
<td>5.26 – 7.88 minutes/year</td>
</tr>
<tr>
<td>6 – Very-High-Availability</td>
<td>99.9999% (6 nines)</td>
<td>31.54 – 47.30 seconds/year (2.63 – 3.94 mins/5 years)</td>
</tr>
<tr>
<td>7 – Ultra-Availability</td>
<td>99.99999% (7 nines) to 99.99999999% (9 nines)</td>
<td>3.16 seconds/year to 31.54 milliseconds/year (15.8 – 23.7 secs/5 years or less)</td>
</tr>
</tbody>
</table>
How Reliability and Availability Play Together

• You can have a relatively low MTTF, but if you have a very short MTTR your availability will be high, and your customer will perceive high system reliability.

Example: If your system can operate uncontrolled for a 20 second window without any issues and you could detect a failure and reset your system within this time window, then you could have several failures per month and never have an outage!

  – For SW failures use a fast reset strategy to quickly repair the system
  – For HW failures use redundancy and switchover to standby HW; this could also save you from a SW failure.

• Work on keeping the MTTR as short as possible!
System Reliability = Hardware Reliability and Software Reliability
Hardware Considerations:

• Selection of Hardware
• Correct Installation & assembly
• Hardware interlock circuits
• Reliable communications / networks
• Redundancy
Selection of Hardware

• Correct controller size, speed etc.
  – Bigger is better. Most resources
  – Budget does play a role

• Correct specification for other switch/control gear
  – Relay, contactor sizing
  – Correct voltage/current ratings

• Embedded vs. Desktop/Windows systems
Correct installation & assembly

• Keeping to manufacturer standards
• Correct cabling sizes
• Not exceeding component specifications
• Use the right tools
• Drawings & guidelines (fault finding)
• Environmental considerations
Hardware Interlock circuits

• Unsafe or disastrous conditions
• Safety relays / contactors
• Not relying on software or controller
• Safely/reliably shut down system/component
• Notify control system of problem
Reliable communications / networks

• Hard-wired vs. Wireless
  – Wireless could be more susceptible to interference

• RS-232 vs. RS-485
  – Noise susceptibility

• Choose the network or comm. that would suit your application for both ease of use and reliability
Redundant system

• Software may cater for sensor failures
• Additional hardware more reliable if failures occurs.
• Example:
  – Separate set of Temp probes
  – Standalone temp controller with contactor to cut power.
• Generally inexpensive, depending on component(s) that will fail.
• Comes down to cost of add. Hardware vs. cost of replacing failed components.
Redundant system Examples:
Example 2: Timed contactor

- Main incoming power – timer starts
- Timer elapses after $x(t)$
- Contactor switch off if no reset signal is received from Control system.
- Unsafe conditions / controller faulty
- Do not want to power up system if nothing is being controller.
- Valves could be open, gas leaks etc.
Software Considerations

• Software compatibility with controller
• Memory & HDD space
• Controller speed & size
• Watchdogs & Heartbeats
• Communication protocols
• Running out of resources
Software compatibility

• Did you allow in hardware decision for enough cpu/memory?
• Software version compatible with hardware version
• Interfacing to 3\textsuperscript{rd} party hardware/software
• NI : RT & FPGA
  – Size, version
Memory & HDD space

- Windows systems normally have sufficient RAM.
- Developers may take this for granted when dev. in Windows environments.
- Embedded systems usually have less RAM
- Porting Win code to RT system
Watchdog timers (RT level)

• Built into all NI controllers (cRIO, PXI)
• Hardware clock – expires if not reset
• Reset by software
• Clock will reset/restart controller if not reset
• Startup states important:
  – SW: code startup – set to safe states
  – HW: some hardware have safe states
• Some watchdogs turn off power. Most only resets hardware (like NI controllers)
Heartbeats / Handshaking

- Not for critical/safety systems
- Between:
  - Windows – RT or Win – Win
  - PC – Tablet
  - LV – OPC – PLC
- Set/reset of a bit on both sides
- Take action when heartbeat is lost
Communication Protocols

• UDP vs. TCP/IP

• UDP : connectionless protocol.
  – When you send a data or message, you don't know if it will get there. There may be corruption while transferring a message. (Streaming media / DNS / VoIP)

• TCP : connection-oriented protocol.
  – When a file or message is sent, it will get delivered unless the connection fails. If the connection is lost, the server will request the lost part. There is no corruption while transferring a message. (WWW / email / FTP)
Running out of Resources

- NI cRIO controllers does not gracefully retire when out of RAM/memory
- VxWorks not cRIO hardware.
- Tip: If designing apps. that stores/log data in TDMS files, continuously monitor the RAM usage to take action before failure.
- TDMS – very good storing mechanism
- TDMS use more resources to contain the format.
Remember

• System Reliability = Hardware and Software Reliability

• Reliability has NO absolutes!
  – You are playing the odds; put them in your favor

• Availability = Fast Repair
Summary

• Choose components correctly
• Plan / Architect system correctly
• Consider resources
• Have fail safes in place:
  – H/W interlocks
  – Redundant systems
  – Watchdogs / Heartbeats
• Overcompensate (if financially viable) but don’t overkill
Summary

• This is why s/w design becomes more expensive today
• Making a system works might be relatively easy
• Making it work reliably takes longer and better planning
Kairos Consulting

• We can assist in drawing up specifications
• Selection of hardware components
• Software architecture
• Best practices
• Consultation on hourly rates or project based
Contact Details

http://www.kairos-cs.co.za/kvi

SALES & PROJECT CONTACT:
Jacques Prinsloo
Cell: 083 462 3976
Email: jprinsloo@kairos-cs.co.za

TECHNICAL & SPECIFICATION CONTACT:
John Gush
Cell: 083 408 9497
Email: jgush@kairos-cs.co.za