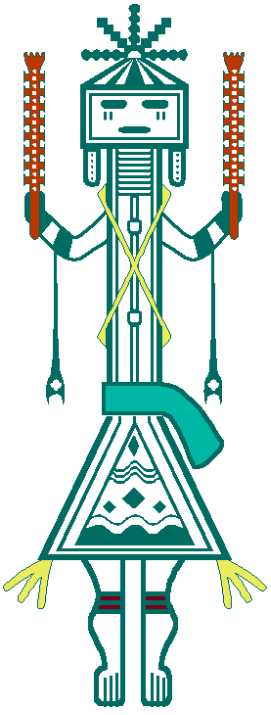
Software Performance Antipatterns in Cyber-Physical Systems (CPS)

ICPE 2020

Connie U. Smith, Ph.D.
L&S Computer Technology, Inc.
Performance Engineering Services
www.spe-ed.com

Overview



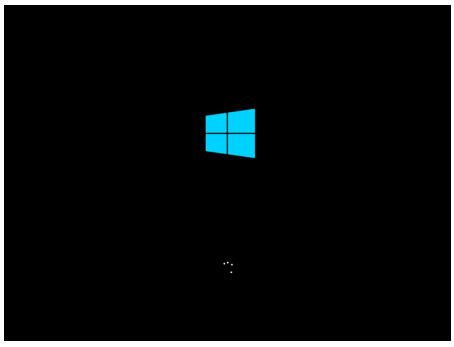
- Brief overview of Software Performance Antipatterns (SPAs)
- 3 New SPAs found in Cyber-Physical Systems (CPS)
- Brief Wrap Up
- Refer to paper
  - Significance and related work in paper
  - 6 Previously defined SPAs common in CPS
  - Other details

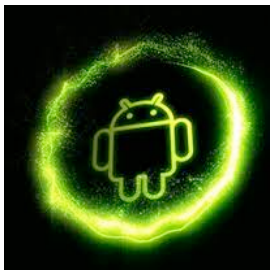
Performance Antipatterns

- Identify potential performance problems in architecture or design and the solution
- Note that:
  - These CPS SPAs may apply to other types of systems
  - Other SPAs may also apply to CPS
  - In our experience these are most common

Poor CPS Performance is Noticeable!

Start-Up Time is Often Problematic





Embedded Systems Then and Now

Key differences in development of today's CPS increase likelihood of performance problems

1. Are We There Yet?

- Often due to polling/reporting
  - Status information, new request arrival, event occurrence
  - High overhead
  - Interval too short relative to changes

1. Are We There Yet? - Solution

- Select platform appropriate for task
  - eg., interrupt-driven or message-based
  - how are notifications implemented?
- Change interval and/or find a better design
  - analysis of polling frequency relative to positive result
  - analysis of time-out interval for usability
  - application characteristics for when to do logging? asynchronous?

2. Is Everything OK?

- Frequency of status checks is far greater than needed for the situation. Significant overhead to activate/ deactivate the processes/tasks doing the check.
  - Excessive checks may deplete the resource.
  - Usability issue for notifications at inappropriate times.

Check	sec.	#times
Battery life	.676	445
Disk space	1.701	445
Update status	0.71	445
Total	3.097	

2. Is Everything OK? - Solution

- Design the platform status checks
  - Event triggered - notify when status reached (e.g., 20% remaining)
  - Time-based - intervals appropriate for resource
  - Event-based - key point(s) in scenario
  - State based - much less frequent when resources are available

3. Where Was I?

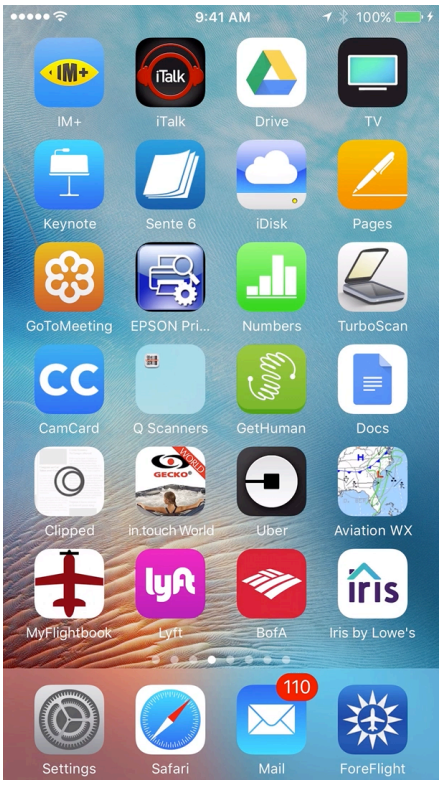
- Process does not remember state so always starts over.
- State is automatically restored but is frequently not the desired state.
- Example: avionics system reports weather predictions - long delay & no results without network connectivity
- Example: avionics - must re-specify all settings not just one

3. Where Was I? - Solution

- Check for connectivity first then display most recent results
- Design for most critical and/or most frequent situations
- Requirement specification for frequent usage & failure modes
  - likely setting changes
  - lack of network connectivity
  - device configuration changes

Example

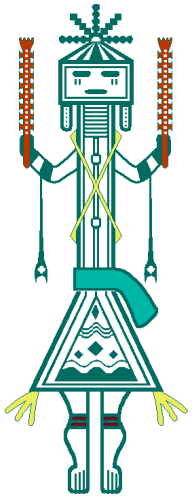
App Connects to Device



Other Applicable SPAs

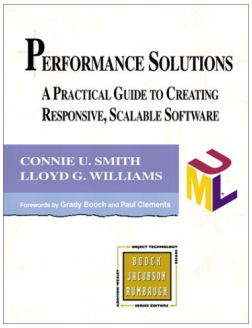
- Unnecessary Processing - executed on the Fast Path but not needed at that time
- How Many Times Do I Have to Tell You? - common method indirectly called many times, but only needed once
- More is Less - too many of a resource results in poorer overall performance
- The Ramp - processing time increases as the system is used
- Museum Checkroom - deadlock from FCFS allocation/deallocation from common queue
- Falling Dominoes - one failure causes performance failures in other components

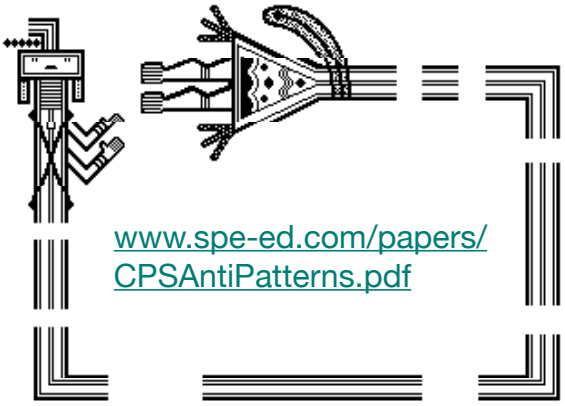
Observations



1. SPAs adapted to specific domains increase likelihood of detection
2. Correlate measurements to design to identify root cause of problems and make auto-correction easier
   - measurements may all look like extensive processing and doesn't explain why
3. Performability -> Falling dominoes and museum checkroom
4. SPAs are better for correcting performance problems than "Bad smells"

Questions?





www.spe-ed.com/papers/CPSAntiPatterns.pdf