Object-Oriented Programming

Software Design Flaws and the CrowdStrike 2024 Outage

Rigidity

Rigidity occurs when even a small change requires updates in many different parts of a system.

- Classic example: Legacy ERP systems, where changing a tax rule or discount calculation forces edits across dozens of modules because the business logic is scattered everywhere.

- CrowdStrike 2024 example: The single update pushed in January 2024 that caused widespread Windows crashes across hospitals, airlines, and banks shows how rigid systems amplify risk. The security agent was so deeply embedded in critical Windows processes that a small change propagated massive, global failures.

- Analogy: Like a tightly wound knot—tug on one thread and the whole thing pulls apart.

Fragility

Fragility means that altering one part causes unexpected failures elsewhere.

- Classic example: A hospital EHR update to the patient registration form unexpectedly broke prescription ordering because the two modules were secretly intertwined.

- CrowdStrike 2024 example: The update was meant to improve threat detection, but instead broke unrelated workflows like airline check-ins, hospital admissions, and government services, because core system processes were disrupted.

- Analogy: Like a pane of glass—tap one corner and cracks spread across the whole surface.

Immobility

Immobility happens when useful components cannot be extracted or reused because they are too entangled with their environment.

- Classic example: An e-commerce shopping cart tightly bound to the website's UI, preventing reuse in a mobile app.

- CrowdStrike 2024 example: The security agent design was so locked into Windows internals that it couldn't be easily isolated or sandboxed. If it had been a more modular, platform-agnostic component, the damage might have been contained instead of bringing down mission-critical services.

- Analogy: Like a heavy boulder cemented in place—valuable, but impossible to move.

Object-Oriented Programming

Viscosity

Viscosity describes environments where it's faster and easier to hack in a bad fix than to follow good design principles.

- Classic example: In a large PHP web app, developers copy-paste the same validation code into 20 files because refactoring it into a shared module would require too much effort.

- CrowdStrike 2024 reflection: Under pressure to deliver fast updates against rapidly evolving cyber threats, engineers may face incentives to take quick but risky paths. If testing, sandboxing, and staged deployment are harder than a global push, viscosity makes the "wrong" approach the default — which is exactly what allowed a single flawed update to cascade worldwide.

- Analogy: Like trudging through a tar pit—it's so hard to move properly that taking a shortcut feels irresistible.

Conclusion

The January 2024 CrowdStrike outage wasn't just a failure of one update — it was a case study in software quality issues that OOP and modular design aim to solve. Rigidity magnified the risks, fragility spread failures into unrelated domains, immobility prevented safe isolation, and viscosity made it easier to push a flawed update globally than to stage it carefully.

This real-world event shows why principles like encapsulation, modularity, and dependency management are not just academic — they are the difference between resilient systems and catastrophic global outages.