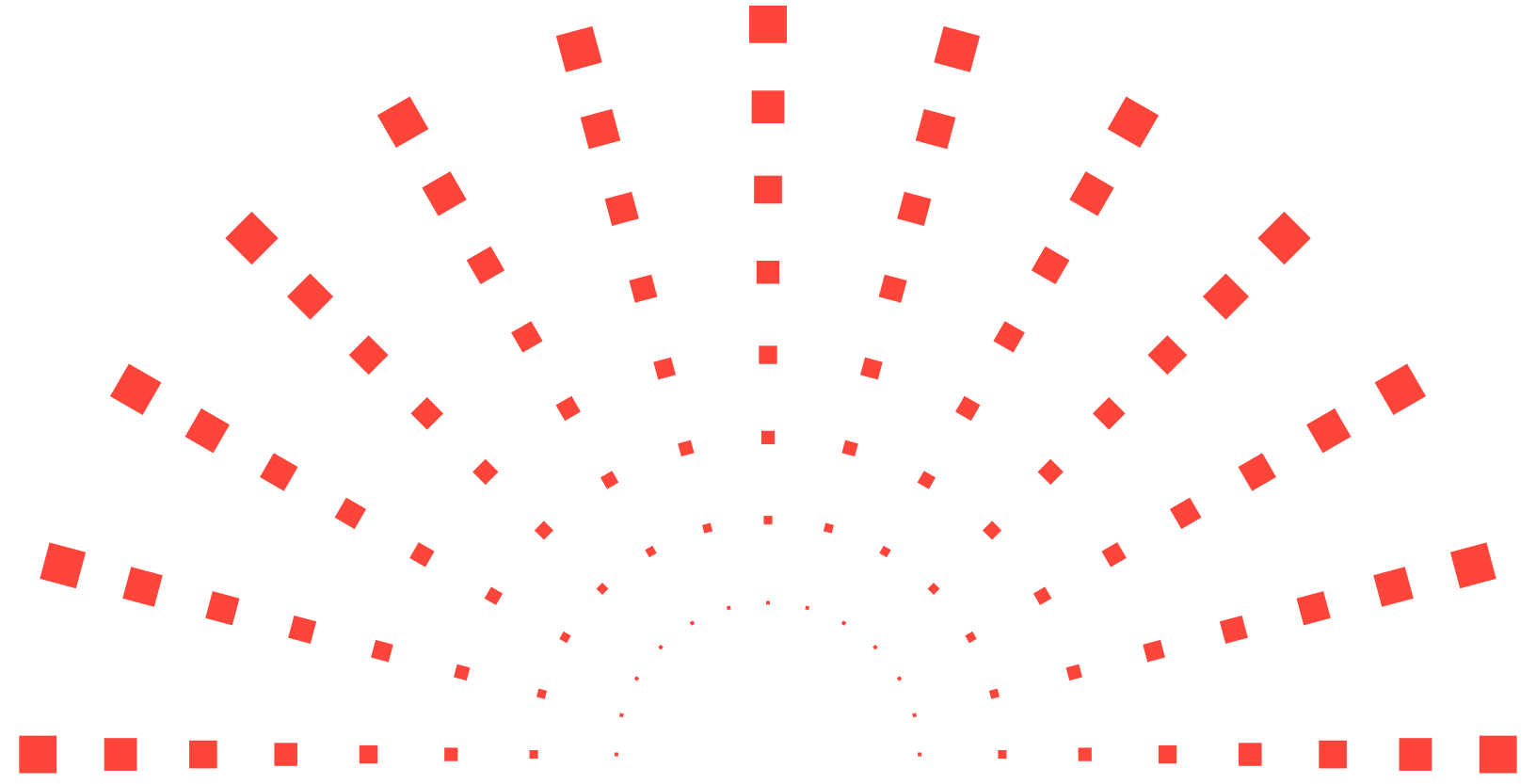
QNX

White Paper

Ensuring Functional Safety in Robot Systems Development

The 12 Key Components for Safety-Critical Robot Systems



Introduction

Imagine stepping into a factory alive with the whir of robotic arms, tirelessly welding steel with laser precision, or a hospital where a surgical robot threads a needle through tissue too delicate for human hands. Picture a bustling warehouse where autonomous carts glide past workers, dodging obstacles with a dancer's grace, or a home where a companion bot lifts a child's toy without a hint of menace. These aren't futuristic fantasies – they're snapshots of today, powered by robotics that have leapt from lab benches to the heart of our lives.

While the possibilities for robotics to change the world as we know it seem endless, a huge challenge in developing these systems is ensuring that they operate correctly, safely, and securely. Beneath the marvel lies a razor-sharp truth – these machines wield power that can build or break, heal or harm, save or shatter. Functional safety is the invisible thread stitching this trust together, ensuring that every gear, sensor, and line of code aligns to banish risk – not just in theory, but in the chaotic, unpredictable grind of reality. An area of particular concern is assuring and certifying embedded software, which is compounded by the trend of increasing software content and complexity in embedded systems. Additionally, the convergence of enablers such as advanced autonomous technologies, edge computing, and the Internet of Things has resulted in embedded systems that are increasingly interconnected and intelligent. These systems are expected to operate reliably in real-time environments while meeting new and evolving functional safety and cybersecurity standards.

Functional safety isn't just about slapping on a kill switch or a warning light – it's about engineering robots that think ahead, shrug off failures, and stand as sentinels against threats, whether from a frayed wire or a hacker's keystrokes. As robots evolve from niche tools to indispensable partners – lifting tons in shipyards, navigating storms for deliveries, cradling patients in care homes – the stakes skyrocket. A single misstep could topple a 500kg load, nick an artery, or crash a production line costing millions. Functional safety isn't a luxury; it's the bedrock of this robotic revolution, a discipline that fuses physics, math, and grit to make machines not just smart, but safe. This paper delves into the 12 pillars that forge this assurance, from the microsecond precision of real-time systems to the long-game vigilance of lifecycle care. It's a journey through the guts of robotics – where brilliance meets responsibility, and where safety isn't a checkbox, but a foundation.

The urgency is palpable. On November 8, 2023, a South Korean worker was crushed to death by an industrial robot at a vegetable packaging plant in Goseong, per AP News, when the machine grabbed him and pressed him against a conveyor belt, failing to identify the human being. This isn't a fluke – it's a flare, signaling that safety must evolve with innovation. Robots today aren't static anymore – they're adaptive, connected, and ubiquitous, pushing engineers to rethink how we harness their power. Whether it's an industrial robot swinging steel or an AMR (autonomous mobile robot) threading a hospital hallway, functional safety is the glue that bonds the capability to trust.

The Essentials of Functional Safety

Functional safety is a robot's solemn vow: "I will not falter, I will not harm." It's not a feature tacked on late – it's the pulse coursing through every circuit, every algorithm, every weld. Imagine an articulated robot pausing mid-motion as a worker nears, its sensors humming to catch a shadow or an AMR rerouting around a spilled cart. This is safety as a living system – reliability ensuring the gears mesh, security locking out digital saboteurs, and resilience catching the wild pitches life throws: a power surge, a toddler's tug, a storm's static. It's a symphony where every note – hardware, software, human – must harmonize, or the tune turns to discord.

At its core, functional safety is about foresight baked into design. It's the math proving a brake fires, the redundancy doubling a sensor's eyes, the training turning a worker's panic into reflex. Take a surgical robot: its 0.1mm precision isn't luck – it's a 1ms control loop, a 10^{-7} failure rate, and a surgeon's muscle memory, all locked in from day one. Security's the silent guard – encryption shields its commands from a hacker's spoof, while reliability keeps its motors purring through a 12-hour shift. This isn't passive defense; it's calculated engineering, turning chaos into control.

Why does this matter? Because robots aren't toys – they're like forces of nature (e.g., a 200kg arm at 2m/s carrying 400J of kinetic energy or a 40kg mobile robot with a load of 15kg topples down two stories of stairs, is enough to cripple a worker). It's the difference between a partner and a peril, a lifeline and a liability. This sense isn't abstract – it's visceral, a feeling in the hum of a factory line or the beep of a care bot. As robotics weaves deeper into our world, safety isn't just engineering – it's a pact, forged in steel and code, that promises machines magnify our potential without menacing our existence. It's the soul of robotics, and it's electrifying.

Key Components for Safety-Critical Robot Systems

In the high-stakes arena of robotics, where machines wield raw power alongside human lives, functional safety isn't a sideline – it's the main event. Every weld, every pivot, every autonomous glide hinges on a system engineered to pre-empt disaster, whether it's a 200kg arm swinging in a factory or a surgical robot slicing within a hair's breadth of a nerve. The difference between triumph and tragedy lies in a meticulous web of principles that govern how robots think, react, and endure. This isn't a simple checklist; it's a battle-tested playbook, forged in the crucible of real-world risks.

The following are 12 pillars that form the backbone of safety-critical robot systems, each a vital thread to form the tapestry of trust. From the ironclad rules of international standards to the predictive pulse of risk analysis, these pillars span the lifecycle of a robot – from its first spark of design to its final decommissioning. They tackle the nuts and bolts – redundancy in sensors, jitter-free like timing, code that won't crack – while wrestling with the human element, the cyber threats, and the relentless wear of time. Together, they don't just mitigate hazards; they redefine what it means to build a machine that's as dependable as it is daring. Let's dive into the 12 pillars essential for building safe and secure robotic systems.

1. Applying Safety Standards and Certifications

Standards like IEC 61508, ISO 10218, ISO 3691-4, and ISO 13482 aren't mere regulatory gatekeepers, they're the battle-hardened charters of robot safety, carved from decades of hard-won lessons in steel, sweat, and code. IEC 61508 sets a relentless benchmark with Safety Integrity Levels (SILs) (e.g., a SIL 3 system must keep dangerous failures below 10^{-7} per hour, a thin margin that demands perfection in every circuit, weld, and code). ISO 10218-1:2025, the industrial robot's guiding star, governs articulated robot arms in industrial settings, enforcing safety functions like speed and separation monitoring when a worker is in close proximity (e.g., changing the speed of an articulated arm from 2m/s to 0.5m/s when a worker nears within 1m). For AMRs acting as driverless industrial trucks, like a warehouse bot moving 500kg pallets, ISO 3691-4 takes the helm, mandating collision avoidance with 0.3m stops at 1m/s. Meanwhile, ISO 13482, the sentinel of personal care robots, governs AMRs in public spaces ensuring safety where the public roams. These standards aren't optional; they're the steel spine of safe design, calibrated to the millisecond, the newton, and the human touch.

The 2025 landscape crackles with evolution. ISO 10218-1:2025 introduces Class I and Class II robot classifications for functional safety, with Class I articulated arms facing stringent tests to cap manipulator force at 50N, its mass at 10kg, or speed at 250mm/s – anything above, Class II. It tackles cybersecurity, requiring manufacturers to provide detailed information in the instruction handbook about implemented features – like secure communications to shield networked robots from spoofed commands that could turn a 1m/s arm into a 10m/s hazard – alongside the need for user policies to bolster protection. ISO 10218-2:2025 embeds collaborative requirements like power and force limiting (PFL), speed and separation monitoring. ISO 3691-4 ensures transport AMRs have safe operating speeds in shared spaces with robust obstacle detection, while ISO 13482, undergoing revisions in 2025, adapts the standard for personal care robots to service robots. Compliance is the starting line; the vanguard goes further by building robots that don't just pass today's tests but are primed for tomorrow's uncharted frontiers. It's a high-stakes chess match where every move fortifies safety.

Certification is a robot's passport to deployment. A IEC 61508 SIL 2-rated articulated arm, proven through 10^5 grueling cycles, welds car frames flawlessly, while an ISO 10218-compliant industrial robot performs collaborative tasks with near-zero incidents. An ISO 3691-4-certified AMR moves pallets through a warehouse without a hitch, and an ISO 13482-compliant hospital bot, delivers trays of food and medicine with responsive avoidance, in a bustling ward. Fail to certify, and you're constrained –

these standards are the foundation for building trust with society, especially when lives hang in the balance. Safety regulations aren't about bureaucracy; it's the bedrock of trust, ensuring every robot is an upholder of safety, ready to face the chaos of a factory floor, a collaborative cell, or a public space with unwavering reliability.

2. Building The Safety Case

A safety case is a robot's legal brief to the universe: "I'm safe, prove me wrong." It's a fortress of evidence. For an industrial robot, it might show how dual encoders guarantee a 0.01° stop, and tools like Monte Carlo simulations help throw virtual wrenches to prove how the system holds.

This isn't a one-and-done document, it's alive. Swap a motor, and the case recalculates: does the new torque curve spike collision risk? Wear a gear down 15%, and fatigue models adjust the odds. A warehouse AMR's safety case might evolve from dodging static racks to weaving through moving forklifts, each shift validated by fresh sensor fusion tests. The thrill is in the chase: spotting a $1/10^6$ software bug before it bites, like a navigation glitch sending a bot left instead of right. It's detective work with a magnified safety lens.

A robust safety case builds confidence in a robot's reliability. It ensures a surgical robot can operate within 0.1mm of a nerve every time or an AMR can transport 500kg past workers without incident. Overlook it, and you risk unexpected failures that could disrupt operations or require justification to stakeholders. It's the engineer's commitment, grounded in evidence, to deliver a system that's dependable.

3. Conducting a Hazard and Risk Analysis

Hazard and Risk Analysis (HARA) is a robot's crystal ball, peering into possible "what if" scenarios to uncover and eliminate or reduce the threat of dangers before they strike. It's a systematic process that identifies hazards across a robot's operation - whether it's a robot's arm swinging too fast, risking a collision that could bruise a worker or an operator misinterpreting a warning signal, leading to an unintended start. Drawing from frameworks like ISO 12100, HARA evaluates each hazard by assessing its severity (e.g., a minor injury versus a critical one), the likelihood of occurrence (e.g., rare versus frequent), and the ability to avoid or control the risk (e.g., easily mitigated versus challenging to manage). With these assessments, mitigation strategies are then crafted with precision (e.g., using dual IMUs to cross-check yaw to help ensure an AMR stays on course; or implementing a 2m/s speed cap to shrink an AMR's stopping distance to 1m with a max load. This isn't just a checklist, it's a proactive shield turning potential chaos into calculated control.

The importance of HARA cannot be overstated, it's the foundation of a robot's safety strategy, ensuring risks are caught early and addressed comprehensively. Without it, hazards lurk undetected, waiting to disrupt operations or cause harm. HARA drives iterative design improvements: on day one, it might flag a blind spot in a robot's sensor array, prompting the addition of a redundant ultrasonic sensor; months later, it could detect a new risk from a vibrating joint, leading to a torque sensor pinging to monitor stability. Tools like Fault Tree Analysis (FTA) map out failure cascades while hazard and operability workshops brainstorm edge cases, such as a power flicker mid-weld that could jolt an arm into motion. By anticipating these scenarios, HARA ensures robots are engineered to handle the unexpected, safeguarding both equipment and human lives. It's a continuous process, evolving with the robot's lifecycle, adapting to new environments, and keeping safety at the forefront of every operation.

HARA is the difference between a reliable system and a costly failure. Skip it, and you risk operational setbacks like an AMR colliding with Wi-Fi dropout. HARA's insights empower engineers to build robots that perform under pressure, ensuring smooth operations and protecting workers. It's a disciplined approach that transforms uncertainty into assurance, making safety not just a goal, but a guarantee.

4. Redundancy and Fault Tolerance

Redundancy in software acts as a robot's safety net, ensuring critical functions persist even when code falters. For an AMR navigating a warehouse, redundant path-planning algorithms run in parallel fusing decisions to ensure a seamless trajectory even if one fails. Fault tolerance digs deeper: Unlike a monolithic operating system kernel, QNX's microkernel, for instance, isolates tasks, so if a navigation software crashes, the braking system remains alive, thanks to process isolation. Recovery shines in software design – a watchdog timer detects a stalled navigation thread, rebooting in 20 µs while a failover algorithm engages a simplified backup path, keeping the AMR moving at a reduced 0.5 m/s. This isn't just coding – it's software engineering with a safety-first mindset, ensuring uninterrupted operation.

The challenge lies in balance, as overloading redundancy can backfire. Triple software modules might seem airtight, but they inflate memory usage and processing overhead, risking latency spikes. Diversity is key: using varied algorithms reduces shared failure modes, unlike duplicating the same code. Fault tolerance in software also stands out, but it's a balance – enough redundancy to safeguard a system, but not too much that it slows the system down.

So why would you invest in software resilience like a microkernel RTOS? Because it ensures a robot's reliability in critical moments. A well-designed safety net lets a surgical robot maintain 0.1mm precision during a software hiccup, or an AMR transport 500 kg through a warehouse without missing a beat. Neglect it, and a single software fault – like an uncaught navigation error – could halt operations, leading to costly downtime. Software redundancy and fault tolerance transform potential disruptions into manageable blips, keeping systems operational and workers safe. It's the coder's promise, woven into every line, to deliver a robot that doesn't just function – it endures.

5. Formal and Semi-Formal Methods

Formal methods are like a robot's truth serum like mathematical techniques that prove it will behave correctly. Model checking with tools like UPPAAL explores millions of possible states (e.g. ensuring that a cobot's arm engages its brakes within a millisecond if it detects a 100N force spike). Semi-formal tools, such as timed Petri nets, visualize timing constraints (e.g., ensuring a surgical robot's movements don't jitter more than 0.1 milliseconds, where any deviation could be catastrophic). These rigorous approaches catch subtle bugs that would be impossible to spot just by inspecting the code.

The toughest challenge is finding the edge cases. Picture an AMR navigating through fog where formal proofs ensure its sensor fusion logic correctly favors radar over LIDAR, allowing it to dodge a crate while moving at 1 m/s. Petri nets can shine here too (e.g. a factory robot's weld sequence – where say the torch must stay on for exactly 2 seconds before shutting off - is modeled to prove it won't overlap with a human's reach-in window). But pure math hits its limits with dynamic chaos, like a worker unexpectedly falling in front of a robot. That's where hybrid approaches come in, combining formal methods with machine learning validation (e.g., neural net stress tests) to bridge the gap. It's a brain-bending blend of rigor and messy reality.

When a 1-in-10^6 glitch (e.g., a floating-point overflow swinging an arm 90 degrees off course) won't show up in normal tests, it can still wreck lives. A robot's safety and compliance rely on this level of proof, verified down to the bit. It's painstaking work, but it's the details that save lives.

6. Real-Time Constraints and Jitter Management

Real-time performance is a robot's lifeline, ensuring actions happen precisely when needed. In a surgical robot, a control loop must execute within a 1ms window to maintain 0.1mm precision during a delicate procedure; even a 2ms delay can disrupt the operation. A real-time operating system (RTOS) like QNX tackles this by prioritizing critical tasks, using preemption to ensure high-priority functions, such as robotic control, execute without delay. By minimizing latency, an RTOS ensures a robot's response aligns with its environment, keeping operations smooth and predictable in dynamic settings.

Jitter management requires careful design to maintain consistency. Strategies like efficient interrupt handling reduce delays caused by external events, while scheduling techniques, such as fixed-priority scheduling, ensure critical tasks meet their deadlines. For example, QNX uses low-latency scheduling and interrupt handling to keep timing variations in check, allowing robots to maintain steady navigation updates even under heavy computational loads. Environmental factors, like temperature fluctuations, can also impact performance, but predictive adjustments – like throttling non-critical logging tasks when CPU temperatures hit 75°C keep a 1000Hz control loop stable for robots assembling parts. These approaches ensure a robot's control systems remain responsive, preserving accuracy across diverse operating conditions.

Why focus on this? Because reliable timing underpins a robot's ability to perform safely and efficiently. A well-managed system allows a surgical robot to execute precise movements without hesitation, or an AMR to navigate a warehouse without unexpected pauses, ensuring seamless operations. Neglecting jitter and latency can lead to operational inefficiencies, such as a factory bot missing a 50ms deadline, delaying the production line by tens of seconds per cycle, and costing hours over a shift. By prioritizing real-time constraints, engineers ensure robots deliver consistent performance, fostering trust in their deployment across industries. It's a foundational discipline that keeps robotic systems in sync with the real world, driving reliability in every cycle.

7. Error Detection and Handling

Errors in robotic systems can disrupt operations, but strong detection and handling mechanisms keep them in check. Microkernel architecture RTOS, such as QNX OS 8.0 helps here by isolating system components, if a driver or service crashes, it doesn't take down the whole robot. Instead, the faulty piece can restart while everything else keeps running. For example, if a robotic arm's motor controller freezes, the rest of the robot stays active, ready to recover without a full reboot.

Watchdog timers play an important role in this too, monitoring system health and kicking in if a process stalls. If a mobile robot's localization module gets stuck, a watchdog can detect it and trigger a safe state, like halting movement or switching over to a backup navigation system, keeping the mission on track without missing a beat.

QNX also uses message-passing between system components, which means it's easy to spot communication breakdowns. If a mapping service stops responding, the robot immediately knows something's wrong and can either try to reconnect or pivot to a fallback plan. This kind of quick error detection keeps a warehouse bot from freezing up mid-task just because one module hit a glitch.

More advanced techniques can add extra layers of safety. Anomaly detection can flag unusual patterns that hint at hardware starting to fail. And if something goes wrong, the QNX microkernel RTOS's flexible design lets the robot reallocate resources, shift into reduced-capability modes, or reroute communications, helping it stay operational even when things aren't perfect.

Communication errors are caught early, too, thanks to built-in error checking like CRC validation for network packets. If a mobile robot loses clean data from a sensor or hits network noise, QNX helps make sure those errors don't ripple through the system and cause bigger problems.

Finally, tools like QNX's TraceEvent give developers real-time visibility into what happened leading up to a fault. If a robot's vision system glitches, engineers can quickly replay the system's steps and find the root cause, speeding up

fixes and minimizing downtime.

By building error detection and recovery into the core of the system, a dependable RTOS helps robots stay adaptable, safe, and dependable - whether they're navigating crowded warehouses, assisting in hospitals, or working side-by-side with people on the factory floor.

8. Security as a Safety Prerequisite

Security in robotics isn't an add-on, it starts at the foundation. For secure robots, security begins with trusted hardware, an immutable anchor that ensures the system boots on a known-good foundation.

Before the operating system even comes online, the robotic system authentication, using public key infrastructure and asset management system, verifies device identity and code authenticity. Every robot, every update, every key component is cryptographically signed and checked, stopping counterfeit devices or rogue firmware before they ever get a chance to run.

Only then does the operating system load. Built on QNX's microkernel architecture, it keeps attack surfaces small by isolating services and minimizing what runs with privileged access. If an application fails or misbehaves, it can't easily take down the whole system.

Higher up, layers of protection stack together. Runtime integrity checks watch system behavior for anything suspicious. Advanced clustering controls CPU and memory use, preventing processes from stepping outside their boundaries. Application security and access controls tighten permissions even further, limiting what code can do and where it can reach.

Tamper protection and secure boot verify system integrity on every restart, while encrypted file systems protect critical data from compromise.

Importantly, these defenses aren't just internal designs - QNX systems are built to meet certification standards like IEC 62443, which defines security requirements for industrial automation and control systems. Achieving compliance shows that security is not only layered, but formally validated against international benchmarks.
The risks are serious - a compromised robot can move unpredictably, disrupt operations, or endanger people. Events like the 2023 insider breach reported by Industrial Cybersecurity Insider highlight why defense in depth, combined with verified compliance, is critical.

By layering security from hardware to runtime, and aligning with standards like IEC 62443, robots stay safe, reliable, and ready, even when facing today's evolving cybersecurity challenges.

9. Static Analysis and Code Coverage Analysis

Static analysis and code coverage analysis are essential for ensuring a robot's software is free of critical flaws. Static analysis tools scan codebases to identify potential issues, such as memory leaks or logic errors, before deployment, while code coverage ensures all paths in the software are tested, verifying that a robot's navigation or control logic functions as expected. In addition, static analysis tools like LDRA's tool suite can measure compliance with coding standards such as MISRA C/C++, CWE, and CERT. These standards specify numerous coding rules designed to ensure software reliability, safety, and security.

While static analysis is performed on software before compilation, dynamic analysis is performed on software that has been compiled and is operating in a physical or virtual system. Compliance with functional safety standards typically requires requirements-based testing (RBT) to be performed. The goal of RBT is to demonstrate that each software requirement has been properly implemented and verified. RBT usually consists of testing at the unit, integration, and system levels.

Code coverage analysis measures the extent to which software has been tested. Performing code coverage analysis while running RBT makes it easy to see if and where there is any untested code that could result in unintended functionality, including a security risk or safety hazard. While code coverage analysis is usually performed on source code, for highly critical software it

may also need to be performed on object code.

Despite their power, these methods aren't foolproof – runtime issues like race conditions can still emerge, requiring additional testing like fuzzing to uncover edge cases. By thoroughly analyzing and testing code, engineers can address hazards and vulnerabilities that might otherwise disrupt a robot's performance in real-world scenarios.

A robust codebase is the backbone of a robot's reliability, enabling consistent performance in applications like surgical robotics or warehouse automation. A 2022 incident reported by Robotics Today highlighted the impact of untested code, where a software bug caused unexpected behavior in a factory boot. Static and code coverage mitigate such risks, ensuring robots operate smoothly and safely, ready to meet the demands of their challenging environments.

10. Integration Testing and Tool Chain Qualification

Integration testing ensures a robot's components work together seamlessly, verifying that sensors, actuators, and software interact as intended in real-world conditions.

Hardware-in-the-loop simulations mimic operational environments, like a factory's electrical noise, to confirm system reliability, while tool chain qualification ensures the software development process – compilers, simulators, and scripts – produces accurate results, avoiding errors introduced by faulty tools.

Real-world challenges are a key focus, with simulation scenarios like connectivity drops to test a robot's response, ensuring it can adapt without failure. Post-deployment, continuous integration pipelines validate updates, confirming new features don't introduce catastrophic outcomes due to tool errors. This rigorous testing ensures a robot's components align, maintaining performance across its lifecycle.

Thorough integration and validation are crucial for a robot's reliability, ensuring it can handle complex tasks without disruption, whether in a factory or a public space. Without these checks, a small error – like a miscalibrated sensor – could lead to operational failures, halting production or requiring costly fixes. By prioritizing these processes, engineers build robots that perform consistently, ready to tackle the challenges of their environments.

In addition to validation of the robotic system, functional safety standards require the tools used to develop, validate, and verify the system to be qualified for use or 'fit for purpose.' Also, the tools used to automate the development and testing of software have to be qualified as they present a potential safety risk and security hazard. There are several generally accepted approaches to tools' qualification including verification, proof in use, and evaluation of the tool development process. In many cases, tools can be pre-qualified by independent certifying agencies such as SGS TÜV SAAR and TÜV SÜD, like those offered by QNX and LDRA. In the most safety and security critical cases, tools must be qualified through a rigorous verification process within the context of the robotic system being developed.

11. Human Factors and Training

Human-robot interaction is a critical aspect of safety, requiring intuitive interfaces and thorough training to ensure effective collaboration. A robot's control panel must be clear – distinct colors for "go" and "stop" commands – reducing the risk of errors. Ergonomic design and feedback mechanisms, like haptic alerts, further enhance operator awareness, ensuring safe operation.

Training is equally vital, equipping operators to handle robots confidently. Virtual and mixed reality simulations allow workers to practice managing a robot in a factory environment, mastering emergency protocols to respond quickly. Numerous studies have found that improved training reduces operator errors significantly, highlighting the impact on safety. These efforts ensure humans and robots work in harmony, minimizing risks in collaborative tasks.

Focusing on human factors ensures robots are practical and safe partners, whether in a surgical suite or a

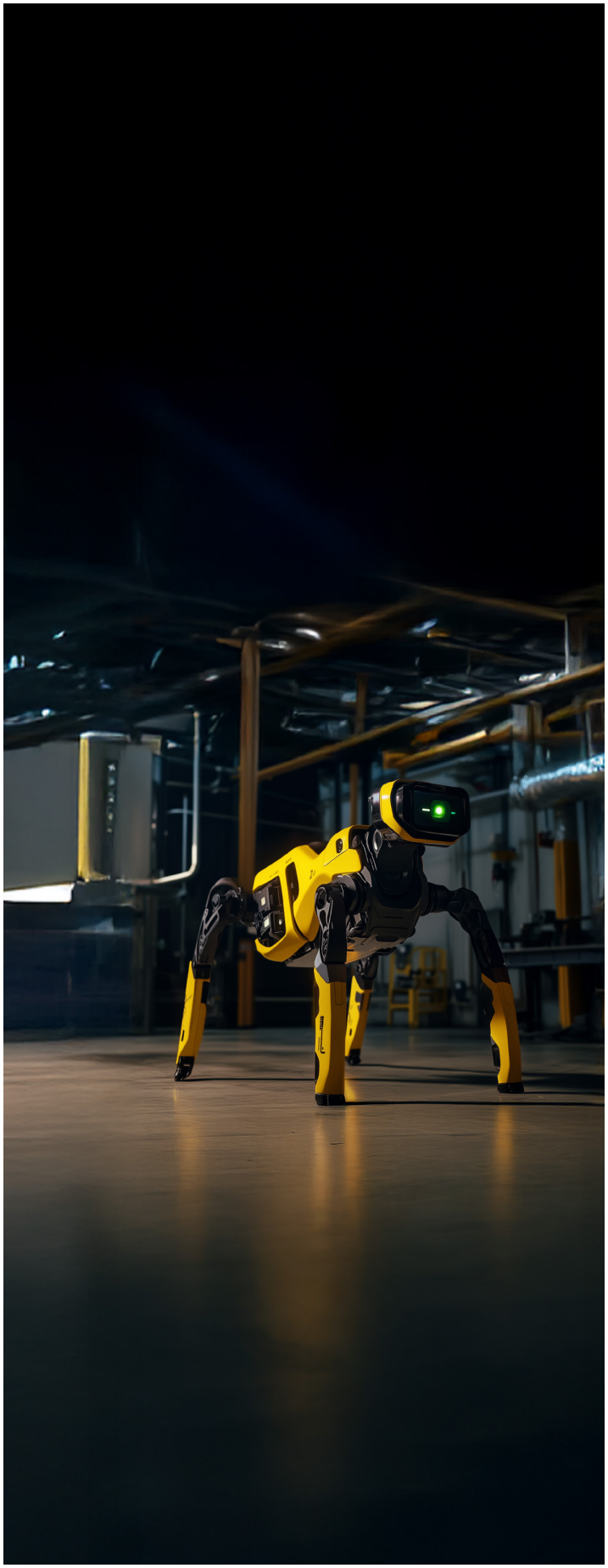
warehouse. Poorly designed interfaces or inadequate training can lead to mistakes, disrupting operations and requiring manual intervention. By prioritizing usability and education, engineers create systems that operators can trust, fostering seamless collaboration and reliable performance in every interaction.

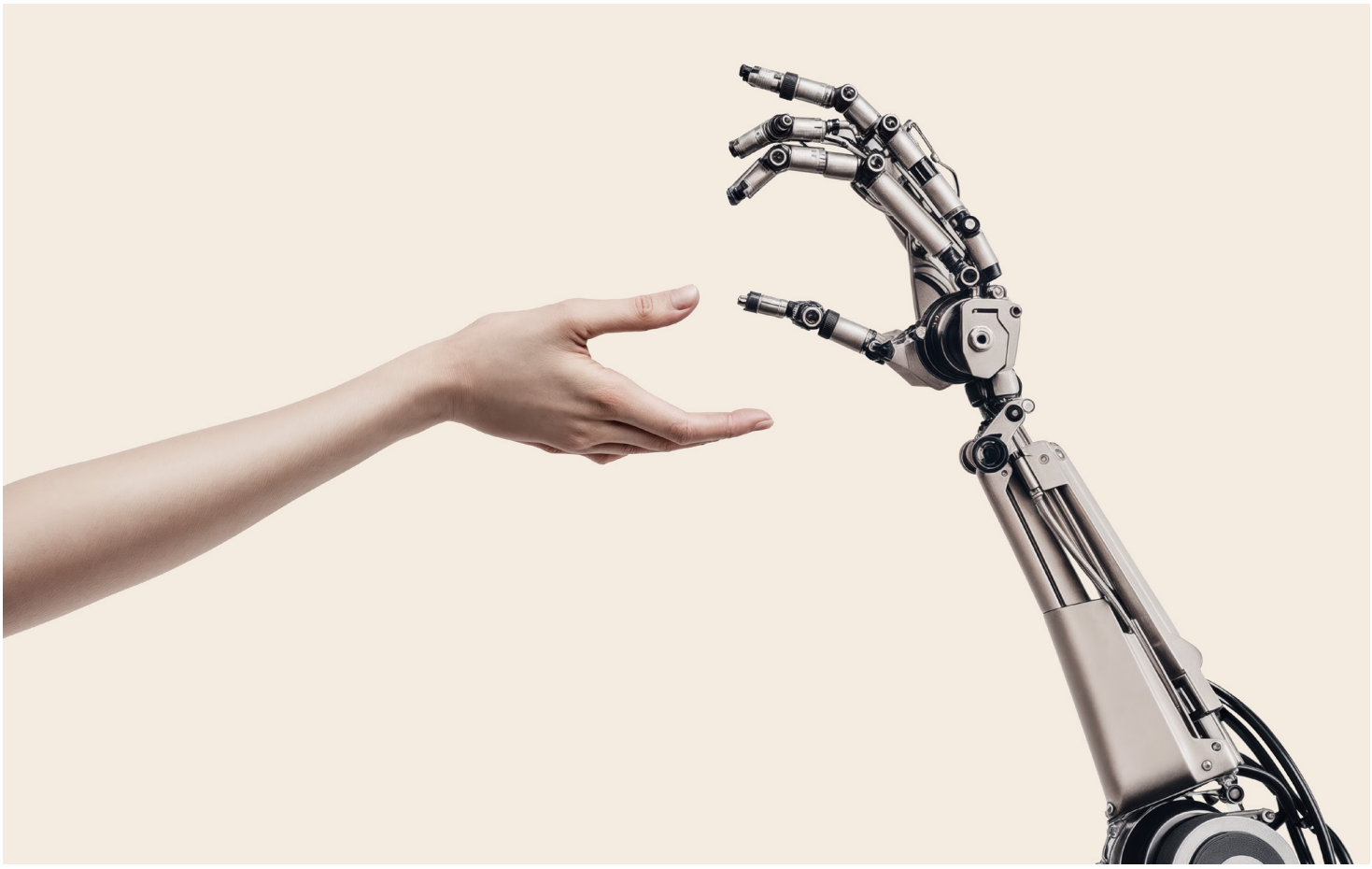
12. Lifecycle Management and Decommissioning

Lifecycle management ensures a robot remains reliable throughout its operational life, monitoring performance and addressing wear over time. Connected systems can track component health, flagging issues like sensor degradation for timely maintenance, while software updates enhance functionality, adapting to new challenges like improved obstacle detection. This proactive approach keeps robots performing optimally, even as conditions change.

Decommissioning is the final step, requiring careful handling to prevent future risks. Secure data erasure, following standards like NIST 800-88, protects sensitive information, while safe disassembly ensures components are disposed of responsibly, often with audited hand-offs to new systems. These steps ensure a robot's end-of-life process is as safe as its operation, leaving no loose ends.

Effective lifecycle management and decommissioning prevent operational disruptions, ensuring a robot remains dependable from start to finish. Neglecting these processes can lead to failures, like a degraded sensor causing a navigation error, or environmental hazards from improper disposal. By managing the full lifecycle, engineers create machines that are not only innovative but also dependable, ready to meet the challenges of an ever-changing world.





Future of Functional Safety

Artificial intelligence (AI) is reshaping functional safety, enabling robots to anticipate and adapt to risks in real time. AI-driven systems can predict potential issues – like a worker's movement in a warehouse – using advanced analytics, enhancing responsiveness in dynamic environments.

However, AI introduces complexities, as its adaptive nature can lead to hallucinations or erroneous outputs that may compromise safety. To address the drawbacks of AI in real-world deployments, future safety frameworks may combine formal verification methods with software-defined guardrails for AI-driven systems, ensuring robots and humanoids remain predictable while still leveraging intelligent insights. This evolution promises a new era of safety – one where robots don't just react but proactively protect, blending human oversight with machine intelligence to push the boundaries of what's possible.

January 2026

QNX

About QNX

QNX, a division of BlackBerry Limited, enhances the human experience and amplifies technology-driven industries, providing a trusted foundation for software-defined businesses to thrive. The business leads the way in delivering safe and secure operating systems, hypervisors, middleware, solutions, and development tools, along with support and services delivered by trusted embedded software experts. QNX® technology has been deployed in the world's most critical embedded systems, including more than 275 million vehicles on the road today. QNX® software is trusted across industries including automotive, medical devices, industrial controls, robotics, commercial vehicles, rail, and aerospace and defense. Founded in 1980, QNX is headquartered in Ottawa, Canada.

Learn more at qnx.com →