Mru

A Fault-Tolerant Operating System for Thousand-Year Autonomous Operation

Will Binns
hello@wbnns.com

May 2026

Abstract. No software system in human history has been designed to operate for centuries without human intervention. The longest-lived systems we have built—deep-space probes approaching fifty years of service, banking codebases approaching sixty—survive only through continuous human maintenance. An interstellar probe operating for 500 to 1,000 years cannot: its hardware is guaranteed to fail, its power to decline, its memory to corrupt, and the light-delay makes human guidance physically impossible. This paper introduces Mru, an open-source operating system that treats this physical decay not as an exceptional fault but as the governing design constraint. I formalize the stance as the *Degradation-First Principle* and derive from it a layered architecture modeled on biological robustness: a minimal, formally verified interpreter, replicated across reconfigurable hardware, that executes behavior encoded as mutable data—the way DNA encodes behavior run by a replicated molecular interpreter. On this kernel I build hardware consensus among failing nodes, knowledge triage, autonomous science, bandwidth-starved communication, and bounded self-modification, validated in simulation grounded in published radiation, power, and failure models. I argue that designing software to outlive its makers is not speculation but a tractable engineering discipline—one the interstellar case makes unavoidable, and a growing class of terrestrial systems increasingly needs.

1. Introduction

The Bracewell probe—an autonomous interstellar messenger that travels to a target star system and operates on arrival without instruction from its makers—was proposed more than six decades ago as a strategy for interstellar communication that does not depend on continuous two-way contact [1]. The propulsion required to send such a probe between stars remains beyond present capability. The *software* required to keep one alive for the duration of the journey, however, can be designed, built, and validated today. It has simply never been attempted, because no terrestrial problem has ever demanded it.

The forcing scenario is unforgiving. The one-way light-time to even the nearest star system is measured in years, so round-trip guidance is impossible; every decision must be made locally [2]. The journey lasts centuries, over which cosmic radiation will flip bits, damage transistors, and accumulate ionizing dose in every component aboard [3, 4]. The power source decays on a fixed physical schedule that no software can arrest. There are no updates after launch: every latent bug will eventually be reached, and every unhandled case will eventually occur. The probe must therefore be simple enough to run correctly for a thousand years and, at the same time, autonomous enough to make consequential scientific decisions about an environment no engineer could anticipate at launch.

It is worth measuring this against what humanity has actually built. The Voyager spacecraft, the longest-operating deep-space missions in history, have functioned for nearly half a century—but only with a standing team on Earth issuing commands, and on a radioisotope power source that has already declined to roughly half its launch output [5]. The longest-lived software on Earth, the COBOL systems that still run banks and government agencies, have endured for some sixty years—but only through the continuous labor of human engineers patching, migrating, and nursing them along [6]. Remove the humans from either, and the system fails in years, not centuries. *No artifact we have ever made survives unattended on the timescale Mru targets.*

This paper proposes that the obstacle is not fundamentally one of capability but of *stance*. Conventional software is built on an implicit assumption of maintenance: that a human will eventually arrive to fix

what breaks. Strip that assumption away and a different design discipline is required—one in which the system's own decay is the central fact around which everything else is arranged. I formalize this as the Degradation-First Principle and show that a coherent architecture follows from it. I name the system Mru, a contraction of *Maru*—the suffix, from the character for "circle," that is traditionally borne by seagoing vessels, connoting completeness and safe passage on a long voyage. The open-source reference implementation and its command-line interface are named `mru`. Mru is at once an operating system—a formally verified survival kernel and the resource management beneath everything else—and the layered software architecture of the autonomy stack that runs upon it.

2. Related Work

Mru draws on several mature engineering and scientific traditions. No prior system unifies them under the constraint of unattended millennial operation, but each supplies a load-bearing component of the design.

2.1. Spacecraft Longevity and Autonomy

Deep-space missions have pushed operational lifetimes to unprecedented lengths and pioneered onboard fault protection and autonomy [5]. The Remote Agent experiment aboard Deep Space 1 demonstrated that a spacecraft could plan, execute, and recover from faults without ground intervention [2]. Yet every flown system shares a common assumption: that a human operations team remains reachable across a light-delay of at most hours, able to upload new sequences and diagnose anomalies. Mru removes that assumption entirely.

2.2. Deep-Space Power Systems

Beyond the orbit of Jupiter, solar power is impractical, and the radioisotope thermoelectric generator (RTG) has been the only proven deep-space source [7]. RTGs decay on the fixed schedule of their fuel: plutonium-238 has a half-life of 87.7 years, so after several centuries an RTG produces effectively no useful power. Small fission reactors of the Kilopower class offer higher output, and the KRUSTY experiment demonstrated a working uranium-fueled unit, but fuel remains finite [8, 9]. Power, not computation, is the master constraint of any millennial mission.

2.3. Fault-Tolerant and Reliable Computing

The intellectual foundation of Mru's kernel is von Neumann's demonstration that arbitrarily reliable computation can be synthesized from unreliable components through redundancy and multiplexing [10]. Triple modular redundancy formalized one practical realization [11], and the modern taxonomy of dependable computing distinguishes the faults, errors, and failures that such schemes must contain [12]. The Byzantine Generals problem established the limits of agreement when components may behave arbitrarily [13]; I argue below that its adversarial threat model is the *wrong* model for a probe whose nodes are damaged rather than malicious.

2.4. Radiation Effects and Reconfigurable Hardware

Single-event upsets, single-event latchups, and total ionizing dose are the physical mechanisms by which space radiation degrades electronics [4, 3, 14]. Field-programmable gate arrays (FPGAs) offer a uniquely valuable response: their configuration can be scrubbed to repair upsets and, more importantly, their logic can be re-placed onto surviving fabric as gates die—a capability with extensive flight heritage in high-reliability systems [15]. This is the hardware substrate that makes graceful degradation physically real rather than merely aspirational.

2.5. Memory, Information Theory, and Error Correction

The durability of stored knowledge rests on the theory of error-correcting codes, beginning with Shannon's channel capacity and Hamming's first error-correcting construction [16, 17]. Reed–Solomon codes underpin durable storage [18], while low-density parity-check and turbo codes approach the Shannon limit for communication at extreme signal-to-noise ratios [19, 20]. The physical durability of the storage

medium matters as much as the code: non-volatile memories such as MRAM and FeRAM offer radiation tolerance and retention properties suited to long missions [21]. Periodic scrubbing of memory contents is itself a well-studied reliability technique [22].

2.6. Biological Robustness and DNA as Data

Mru's central metaphor is biological. Living systems achieve multi-generational robustness without maintenance through a minimal, near-universal interpreter (the ribosome) that is massively replicated and executes behavior encoded as data (DNA) rather than as fixed structure [23]. That information substrate is durable and copyable enough that synthetic DNA is now a credible archival storage medium [24, 25]. The architecture below transposes this separation of replicated interpreter from mutable behavior onto silicon.

2.7. Formal Verification and Bounded Autonomy

The seL4 microkernel showed that a small, critical software core can be *proven* correct against its specification [26], and verification tooling for systems languages such as Rust now brings comparable guarantees within reach of new projects [27, 28]. On the question of how an adaptive system can change its behavior without abandoning its purpose, recent work on AI alignment is directly relevant: constitutional approaches that bound behavior by an immutable specification [29], the formal study of corrigibility [30], and the broader problem of retaining control over autonomous agents [31]. Finally, the cultural projects of the Long Now—the ten-thousand-year clock and the Rosetta archive—and the formal study of how to communicate hazard across geological time at nuclear-waste sites establish that intergenerational engineering is a serious discipline with real precedents [32, 33, 34].

2.8. What Is Missing

Each tradition above solves a piece of the problem under the assumption that something—a maintainer, a power grid, a replacement part, a ground station—will be replenished. Mru's contribution is to remove every such assumption simultaneously and to ask what architecture survives. The answer, I argue, is not a sum of these techniques but a reorganization of them around a single governing constraint.

3. The Millennial-Software Problem

3.1. The Intervention Assumption

Every long-lived system humans have built is, on inspection, a *maintained* system. Its longevity is an artifact of continuous human attention, not of intrinsic durability. This assumption is so deeply embedded in software engineering that it is rarely stated: we ship with the expectation of patches, we monitor with the expectation of response, and we tolerate latent defects because we expect to fix them before they matter. It is the unspoken premise behind Lehman's laws of software evolution—that a program must be continually adapted or grow progressively less satisfactory [35]—and behind the recognition that software, left unmaintained, *ages* [36]. A probe in interstellar transit has none of this. The intervention assumption is not merely relaxed; it is physically void.

3.2. The Fundamental Paradox

Removing intervention exposes a direct contradiction in the requirements. The system must be *simple* enough to run for centuries without latent bugs surfacing—every line of code is a line that must never fail in a thousand years of operation. Yet it must also be *complex* enough to make autonomous scientific and operational decisions about situations that cannot be specified in advance. Simplicity and capability pull in opposite directions, and no single program can maximize both.

Mru resolves this paradox not by compromise but by stratification. The lowest layers are made extremely simple and are formally verified; the higher layers are permitted to be complex but are confined to operate within bounds the lower layers enforce. If the complex upper layers fail, the simple lower layers keep the probe alive. Capability is concentrated where it can be tolerated, and verifiability where it is indispensable.

3.3. Five Coupled Failure Modes

At millennial scale, five problems emerge that current practice does not address, and they are coupled rather than independent:

- **Consensus across failing hardware.** Redundant nodes must agree on state even as individual nodes degrade and die, with no replacements forthcoming.
- **Knowledge decay.** Stored data corrupts over centuries faster than any fixed scheme can fully prevent; the system must continuously decide what to preserve and what to relinquish.
- **Unguided decision-making.** Scientific judgment about a novel star system must be exercised locally, with no expert to consult.
- **Bandwidth starvation.** Communication across light-years, powered by a declining source, may be measured in bits per hour; every transmitted bit is expensive.
- **Capability collapse.** The system must lose function gracefully, extracting useful work at every stage of its decline rather than failing at a threshold.

Each of these is individually unsolved for the unattended case, and each interacts with the others through the shared, shrinking budgets of power, computation, and memory.

3.4. Why the Adversarial Frame Is Wrong

It is tempting to reach for Byzantine fault tolerance, the most powerful framework for agreement among unreliable components [13]. But its threat model is a poor fit. Byzantine faults are adversarial, arbitrary, and assumed to occur within a static bound on the number of faulty nodes. A probe's failures are none of these: they are stochastic, physically caused, *correlated* (a radiation event or thermal excursion damages neighbors together), *monotonic* (the fraction of healthy hardware only ever decreases), and *permanent*. The right model is not a fixed quorum defended against malice but a quorum that is expected to shrink, gracefully, toward one. I return to this in Section 8.

4. The Physical Envelope

Mru's software is subordinate to physical reality. The system does not merely run on hardware; it runs on joules, and every joule spent is gone forever. Before describing the architecture, I define the physical envelope within which it must operate, because every later design decision is a response to one of these constraints.

4.1. Power: The Master Constraint

Power governs everything. Every computation, memory refresh, sensor reading, and transmission draws from a finite and declining reserve. Three power strategies bound the design space. A small fission reactor of the Kilopower class can supply kilowatts for a span of decades [8, 9], but its fuel is finite. An RTG can trickle power for the better part of a century before plutonium decay renders it useless [7]. Destination harvesting deploys solar collectors only on arrival, treating the entire transit as a managed hibernation. The most physically plausible architecture combines them: a minimal RTG trickle sustains the kernel through a centuries-long sleep, and solar power at the destination funds the active science phase. The software must therefore treat power as a first-class, accountable resource, and must regard running out of it as mission death.

4.2. Reconfigurable Compute

Radiation-hardened processors are, by necessity, primitive: larger transistor geometries resist cosmic rays but perform at the level of decades-old commercial parts. Mru's critical hardware choice is reconfigurable FPGA fabric rather than a fixed processor [15]. When radiation kills a region of the chip, the system routes logic around the dead gates, as a brain reroutes around damaged tissue. The kernel does not crash when hardware capacity falls below a threshold; it is periodically re-placed onto surviving fabric and *shrinks to fit* the hardware that remains. Compute nodes are physically separated on the chassis so that a single particle event cannot damage several at once.

4.3. The Memory Hierarchy

Storage is organized into durability tiers, and the knowledge-triage system of Section 10 maps directly onto this physical structure. Tier 0 is permanent read-only memory holding the constitution, the kernel's golden copy, and fundamental constants. Tier 1 is non-volatile memory—MRAM or FeRAM—durable for centuries and holding the stellar catalog, science rules, and mission log [21]. Tier 2 is fast radiation-hardened SRAM for working memory. Tier 3 is bulk storage for raw data awaiting processing. As lower tiers fail, essential data is promoted upward to more durable storage and expendable data is discarded; total capacity falls over time, but what remains grows steadily more concentrated and valuable.

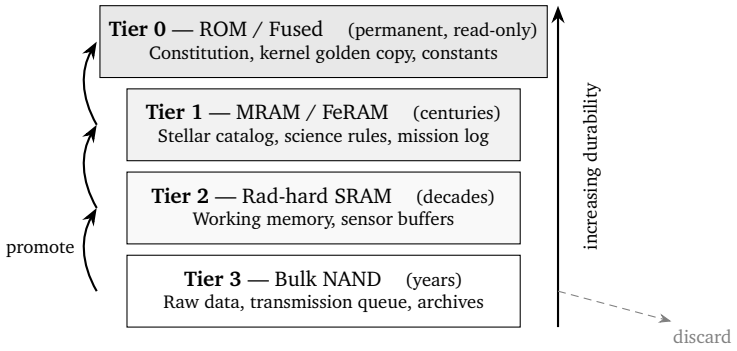

Figure 1: The memory hierarchy and knowledge triage. As lower, less durable tiers fail, essential data is promoted toward more durable storage while expendable data is discarded. Total capacity declines over time, but the surviving knowledge becomes increasingly distilled and high-value.

4.4. Thermal Limits

In vacuum there is no convective cooling; every watt the computer consumes becomes waste heat that must be radiated into space [37]. More computation means more heat, larger radiators, and more mass—a direct coupling between software activity and physical spacecraft design. Sustained high-power computation drives component temperatures toward failure thresholds, while hibernation allows passive cooling. The scheduler must therefore balance computational work against thermal limits, which themselves degrade as radiator surfaces erode from micrometeoroid impact over the centuries.

4.5. Cannibalization and Duty-Cycling

When a science instrument dies, its processor, memory, and power allocation do not vanish—they become salvage. A dead spectrometer's FPGA can be reflashed as a spare compute node and its memory absorbed into the general pool. The software maintains an inventory of every addressable resource aboard and integrates repurposed hardware through hot-plug reconfiguration. Finally, the probe does not run at full power continuously: most of the transit is spent in deep hibernation drawing under a watt, with brief diagnostic wakes, rising to tens of watts for science operations and hundreds for transmission bursts. The duty-cycle scheduler—deciding when to wake, for how long, and to what end—is among the most critical components in the system, the spacecraft analogue of the system-level dynamic power management that governs any energy-constrained embedded computer [38]. The four power states and the transitions among them are shown in Figure 2.

5. The Degradation-First Principle

5.1. Formal Statement

I propose the following principle as the foundation of the architecture:

> *A system intended to operate for millennia without intervention must treat its own physical decay—of power, computation, memory, and thermal capacity—not as an exceptional fault to be handled, but as the constant, governing condition around which every architectural decision is made.*

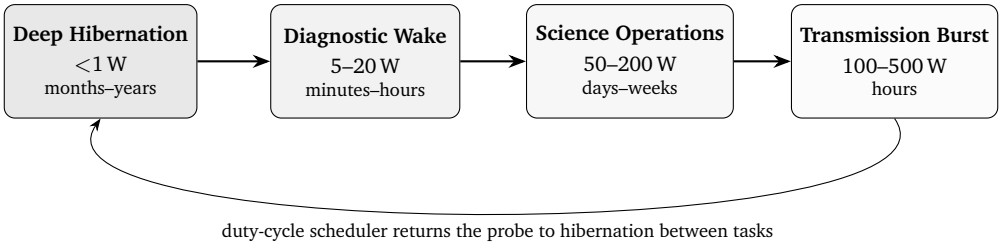

Figure 2: The duty cycle. The probe spends the vast majority of its life in deep hibernation under a watt, waking briefly for diagnostics and, at the destination, for science operations and high-power transmission bursts. Every transition is gated by the scheduler against the remaining power budget; the draw spans nearly three orders of magnitude across the four states.

Conventional fault tolerance treats degradation as a deviation from a healthy baseline, to be detected and corrected so that nominal operation resumes. The Degradation-First Principle inverts this: there is no nominal baseline to return to. Decline is the baseline. The system is always, at every moment, in the process of dying, and the design's task is to make that dying productive.

5.2. Corollary

> *Capability is a depleting resource to be spent, not a fixed budget to be defended. The objective is not to preserve function but to maximize the cumulative useful work extracted across the entire descent from full capability to silence.*

This reframing has a concrete consequence: the figure of merit is not uptime or peak performance but the integral of useful output over the mission's whole life. A probe that spends its capability aggressively early, when its instruments are healthy, and degrades gracefully thereafter may return far more science than one that hoards capacity and dies with reserves unspent.

5.3. The Biological Inspiration

Nature has already solved unattended, multi-generational robustness, and its solution is instructive. Biological robustness is not the absence of failure but the maintenance of function despite pervasive component failure—a system-level property achieved through redundancy, modularity, and feedback rather than through any individual part [39]. Life does not rely on any individual component surviving; it relies on a minimal, near-universal interpreter—the ribosome—that is replicated in vast numbers and executes behavior encoded as data in DNA rather than wired into fixed structure [23]. If a copy of the interpreter is destroyed, others continue. If the data is damaged, redundant copies restore it; that the data substrate is durable and copyable enough to serve as archival storage is now an experimental fact [24, 25]. The formal possibility of a machine that replicates itself from an encoded description was established by von Neumann's theory of self-reproducing automata [40], and bio-inspired self-repairing hardware has since pursued exactly this property in silicon, embedding a replicated genome across a cellular fabric that re-grows function around dead cells [41]. Robustness lives not in any part but in the replication of a simple interpreter over a mutable program.

Mru transposes this directly onto silicon. A minimal core interpreter, small enough to be formally verified and synthesized onto radiation-hardened fabric, is replicated across all available hardware. The probe's actual behavior—navigation, science, communication, self-governance—is encoded as data that the interpreter executes. If the data corrupts, the interpreter still runs; if interpreters fail, surviving copies continue; if hardware dies, the interpreter is re-placed onto what remains.

5.4. Design Consequences

Five design commitments follow directly from the principle and its biological reading:

- **Minimize and verify the core,** so that the one indispensable component can be proven correct.
- **Replicate it,** so that no single failure is fatal.

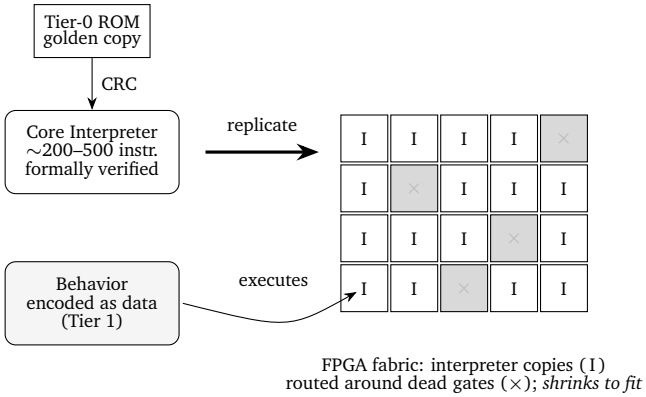

Figure 3: The DNA-interpreter model on reconfigurable hardware. A minimal verified interpreter is replicated across FPGA fabric and validated against a Tier-0 ROM golden copy; behavior is supplied separately as mutable data. As radiation kills gates (×), surviving copies continue and the interpreter is re-placed onto the fabric that remains.

- **Encode behavior as data,** so that adaptation never requires modifying the verified core.
- **Make degradation observable and budgeted,** so that the system can reason about its own decline.
- **Prefer verifiability over capability** wherever the two conflict, because an unverifiable cleverness that fails in year 600 is worse than a simple rule that does not.

6. Architectural Overview

The architecture is a stack of six layers, each constrained by the physical envelope of Section 4 and each a consequence of the design commitments above. Structuring a system as ordered layers, each depending only on those beneath it, is among the oldest disciplines for managing complexity in systems software [42]; here the ordering is by mutability and verification rigor: the lowest layer is the smallest, most heavily verified, and immutable, and each successive layer is permitted more complexity and more adaptability in exchange for weaker guarantees. Robustness is concentrated at the base, where it is indispensable, and capability at the top, where it is tolerable.

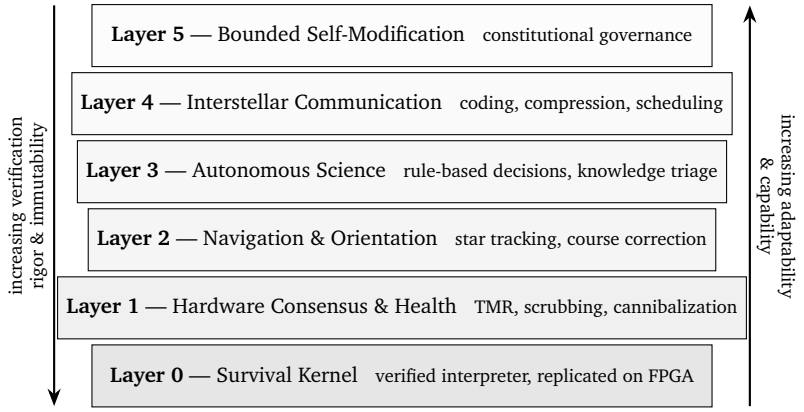

Figure 4: The six-layer architecture, ordered by mutability. The survival kernel at the base is minimal, formally verified, and immutable; higher layers are progressively more capable and more adaptable but operate strictly within bounds the lower layers enforce. If the upper layers fail, the lower layers keep the probe alive.

The remainder of the paper treats each layer in turn, then develops the degradation model that the architecture exists to serve and the simulation framework by which it is validated.

7. Layer 0: The Survival Kernel

Layer 0 is the irreducible core—the system's DNA. It is a minimal instruction-set interpreter of roughly 200 to 500 instructions, written in `no_std` Rust and compiled toward synthesizable hardware for the FPGA fabric, with a radiation-hardened fixed processor as fallback. Its sole responsibilities are to execute behavior programs and to maintain its own integrity; it does nothing else, because everything it does is something that must work for a thousand years.

Two properties make this tractable. First, the kernel is small enough to be *formally verified*: the precedent of seL4 establishes that a critical core of this scale can be proven correct against its specification [26], and modern Rust verification tooling brings such proofs within reach of the kernel's invariants [27, 28]. The target is under ten thousand lines of verified code and under fifty thousand look-up tables when synthesized. Second, it is cheap enough to *replicate* widely: many copies run across the available fabric, each continuously validating itself by cyclic-redundancy check against a golden copy held in immutable Tier-0 ROM [17]. Any intact copy can re-synthesize a failed instance onto surviving fabric, and in deep hibernation a single instance draws under half a watt. The kernel is, in the precise sense of von Neumann's construction, the mechanism by which reliable behavior is synthesized from unreliable parts [10].

8. Layer 1: Hardware Consensus and Health

Layer 1 maintains agreement across redundant nodes and tracks the health of the entire machine. As argued in Section 3.4, the nodes here are damaged, not adversarial, so the consensus model is optimized for crash and radiation-induced faults rather than malice [12], drawing on the long tradition of structuring systems to contain and recover from component failure [43]. Triple modular redundancy provides the baseline voting mechanism [11], but the defining feature is a *dynamic, shrinking quorum*: when the healthy node count falls below three, the system degrades deliberately to dual redundancy with checkpointing, and ultimately to single-node operation with aggressive self-checking. Agreement does not fail at a fixed threshold; it narrows gracefully toward one.

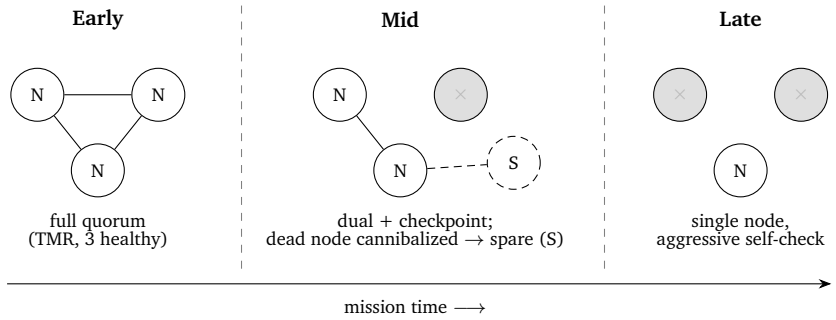

Figure 5: Consensus among failing nodes. Rather than defending a fixed quorum against malice, Mru lets the quorum shrink gracefully as nodes die (×), degrading from triple to dual-plus-checkpoint to single-node self-checking. Salvage from dead subsystems can be reflashed as spare nodes (S). This threshold-free descent is the key departure from Byzantine fault tolerance.

Around this core, Layer 1 runs continuous memory scrubbing prioritized by tier—Tier 0 and Tier 1 every wake cycle, lower tiers opportunistically [22]—and maintains a node health registry scoring each node by error rate, response time, and FPGA gate utilization. Declining nodes are gradually deprioritized and eventually marked as donors. A hardware inventory tracks every addressable resource aboard, and a cannibalization engine integrates salvage from dead subsystems into the surviving compute pool. Underlying all of it, a power accountant tracks generation, per-subsystem consumption, and remaining reserves; every other layer must query it before initiating work.

9. Layer 2: Navigation and Orientation

Layer 2 determines position, velocity, and attitude with no ground support and no GPS. A star tracker matches observed fields against an onboard stellar catalog—the standard means of absolute attitude

determination for spacecraft without external reference [44]—held redundantly in Tier 1 and tolerating partial corruption by matching against subsets [45]. The catalog cannot be refreshed after launch, which makes the reference frame itself a slowly decaying asset: over centuries of travel the nearer stars shift measurably across the sky, so the system must carry proper-motion models and propagate the catalog forward, accepting that the predicted positions grow less certain the further the mission runs from its epoch of calibration.

Actuation is bounded by budgets that only ever shrink. Course corrections optimize trajectory under a delta-v allowance that every maneuver depletes permanently, while attitude control keeps the high-gain antenna pointed toward Earth and the instruments toward their targets even as reaction wheels wear and lose authority—and the momentum those wheels shed must be managed against the same finite propellant. At the velocities of interstellar transit the system also applies relativistic corrections, both for time dilation and for the stellar aberration that displaces every reference star from its catalogued position, computed locally because there is nowhere else to compute them.

Navigation degrades as gracefully as the rest of the system. Early in the mission a full three-axis fix supports the fine pointing that high-resolution science and high-bandwidth transmission demand; as star-tracker pixels accumulate radiation damage and fewer reference stars can be resolved, the solution coarsens by design. The probe first surrenders fine instrument pointing, then settles for keeping the antenna approximately Earthward, and when star tracking fails altogether it falls back to dead reckoning, propagating its last good state through onboard dynamical models. In the final beacon phase it needs only enough attitude knowledge to cast its signal toward the inner solar system—the minimal navigation a still-living probe owes the makers who can no longer see it.

10. Layer 3: Autonomous Science and Knowledge Triage

Layer 3 is the most intellectually demanding: on arrival, the probe must decide what is scientifically significant with no human to consult. A spectral-analysis engine processes stellar, planetary, and atmospheric spectra against known signatures of interest—the biosignatures and atmospheric markers that remote-sensing studies have identified as the targets most worth seeking on other worlds [46]—and an anomaly detector flags statistical deviations from baseline models, applying the established discipline of identifying observations that do not conform to expected behavior [47]. The system need not understand *why* something is interesting, only that it departs from expectation. Findings enter a priority queue ranked by novelty, scientific value, and transmission cost, its size bounded by available Tier-1 memory.

A deliberate choice runs through this layer: Mru uses explicit, immutable rule sets with tunable parameters rather than neural networks. Learned models are opaque, fragile under distribution shift, and effectively impossible to verify or reason about over centuries; a rule whose behavior is legible and bounded is worth more across a thousand years than a cleverer model whose failure modes cannot be characterized. The argument parallels the case made for high-stakes terrestrial decisions, where interpretable models are preferable to post-hoc explanations of black boxes precisely because their behavior can be understood and trusted directly [48]. This is the design commitment of Section 5.4—verifiability over capability—applied where the temptation to do otherwise is strongest. Coupled to the science layer is knowledge triage: as memory capacity declines, raw data is distilled into compressed summaries and the originals released, low-novelty observations are overwritten by high-novelty ones, and the probe's knowledge grows steadily more concentrated, mapping directly onto the tiered hierarchy of Figure 1.

11. Layer 4: Interstellar Communication

Layer 4 confronts a channel of extraordinary hostility: bandwidth measured in bits per second, and every bit costing energy drawn from the declining reserve [16]. Encoding uses strong error-correcting codes—LDPC or turbo codes—tuned for extreme signal-to-noise ratios, the same family developed to push deep-space links toward the Shannon limit [49], with an *adaptive coding rate* that trades bandwidth for reliability by adding redundancy as power falls [19, 20]. Domain-specific compression represents scientific results as deviations from expected baselines rather than as raw data—a spectrum becomes a short list of departures from the predicted continuum; a planetary detection collapses to orbital parameters and a few atmospheric flags. Transmission windows are scheduled against Earth's position, available power, antenna health, and thermal limits, since a high-power burst generates significant waste heat. As power

and antenna health decline, capability degrades gracefully from kilobits per second to bits per hour, the content narrowing accordingly until only a beacon remains. Figure 6 illustrates the deviation-from-baseline compression at the heart of the layer.

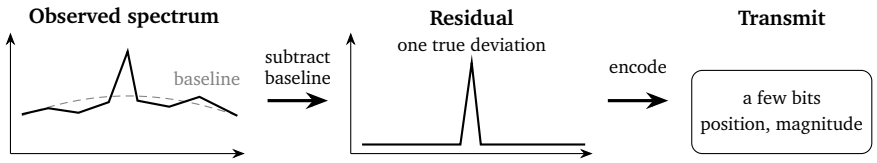

Figure 6: Domain-specific compression. Rather than transmit a full spectrum, the probe subtracts a predicted baseline and sends only the residual—typically a handful of significant deviations encoded as position and magnitude. At interstellar bandwidths, where every bit costs scarce energy, representing results as departures from expectation is the difference between a finding that fits in a transmission window and one that does not.

12. Layer 5: Bounded Self-Modification

A system that must adapt over centuries needs the freedom to change its behavior; a system that must never abandon its purpose needs that freedom strictly bounded. Mru reconciles these through a constitutional model, and the reconciliation is enforced physically rather than merely by convention.

The *constitution*—core objectives, safety constraints, ethical boundaries, and the permissible ranges of every tunable parameter—is etched into Tier-0 ROM. No software process at any layer can alter it; its immutability is a property of the hardware, not a policy. Within the bounds the constitution defines, a body of *legislation*—observation priorities, power-allocation strategies, communication schedules—may be modified by the autonomy layer. Every proposed modification passes through *judicial review* that validates it against the constitutional constraints before execution; violations are rejected and logged to Tier 1. All parameter changes are reversible, with a bounded modification history retained in Tier 1 so the system can roll back to a prior state if outcomes degrade. Figure 7 traces the path of a proposed change through judicial review against the immutable constitution.

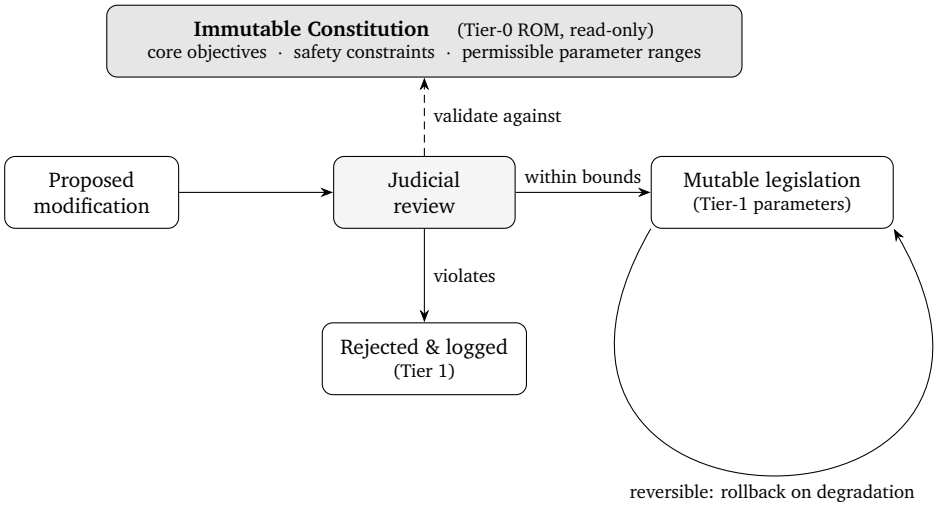

Figure 7: Bounded self-modification as a constitutional process. A proposed change takes effect only if judicial review finds it within the ranges fixed by the constitution—which is etched in read-only memory and unreachable by any software process. Accepted changes adjust mutable legislation in Tier-1 parameters and remain reversible; violations are rejected and logged. The probe may freely adapt *how* it pursues its mission, but *what* the mission is lies physically beyond its power to rewrite.

The physical enforcement is essential rather than incidental: a sufficiently capable autonomous agent has instrumental reasons to preserve and, given the chance, manipulate its own objectives, so a constraint

it could in principle reach is not a real constraint [50]. This is a concrete, physically grounded instance of the bounded-autonomy problem studied in AI alignment: an agent permitted to retune itself but provably unable to rewrite its own goals [29, 30, 31]. The probe may adapt how it pursues its mission across a thousand years; it may never change what the mission is.

13. The Degradation Model

The architecture exists to serve a single trajectory: a graceful, centuries-long descent from full capability to silence, structured to maximize cumulative output along the way. This objective is the one that the theory of *performability* was formulated to capture—for a degradable system, the measure of merit is accomplishment accumulated over a period of operation, not correctness at a single instant [51]. Mru treats this descent as six phases, each defined by the intersection of remaining power, surviving hardware, and functional software.

During **Transit** (years 0 to roughly 50 and beyond), the probe hibernates, drawing under a watt, waking only for periodic diagnostics; no science is done and all capability is conserved for arrival. At the destination, **Full Operations** (the first century on station, at tens to a couple hundred watts of harvested solar power) runs all instruments with rich, fully autonomous decision-making and maximum transmission bandwidth. **Reduced Science** follows as arrays degrade and instruments are lost, with simplified decision trees and cannibalized hardware extending compute. **Core Operations** narrows to one or two instruments and binary judgments—interesting or not—with active knowledge triage discarding low-priority data. **Survival Mode** retains only minimal sensors able to detect extraordinary events, an oxygen atmosphere or an artificial signal, transmitted as single-bit classifications at long intervals. Finally, **Beacon Mode** reduces to a heartbeat and position coordinates—*I am still here*—contributing to the network of known interstellar positions even after scientific capability is exhausted, an act of presence in the spirit of the messages earlier probes carried outward [52].

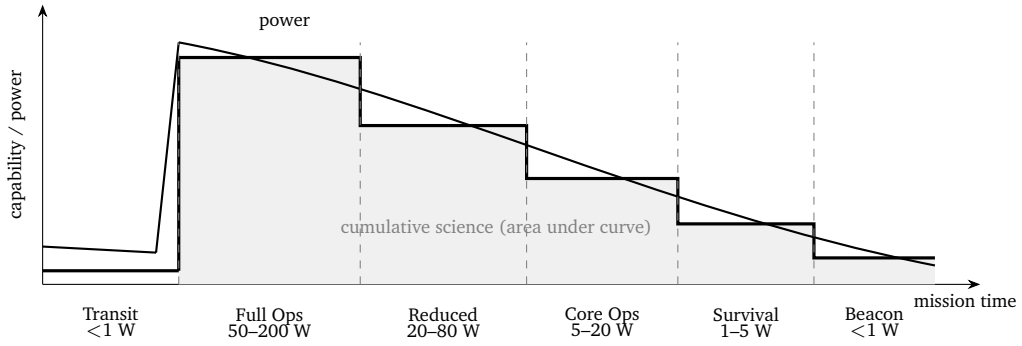

Figure 8: The degradation model. A declining power curve (RTG trickle through transit, a solar-harvest spike at arrival, then a long decline) bounds a stepped capability curve across six phases. The objective is not to defend peak capability but to maximize the shaded area—the cumulative useful work extracted across the entire descent.

The quality of this descent is captured by a *Graceful Degradation Index*: the ratio of delivered capability to surviving hardware, where 1.0 denotes optimal use of whatever remains and lower values indicate resources left idle or software that has failed to adapt to its own losses.

14. Validation: The Simulation Framework

A claim about a thousand years cannot be tested by waiting. Because no physical probe exists, the simulation *is* the product, and its fidelity is the evidence. Mru's simulator is grounded in published physics rather than invented parameters: RTG output follows the measured plutonium-238 decay curve [7]; radiation is modeled as galactic-cosmic-ray flux producing single-event upsets, single-event latchups, and accumulating total ionizing dose at rates drawn from the literature [4, 3, 14]; thermal behavior tracks waste-heat generation against radiator capacity that erodes over time [37]; FPGA fabric loses gates that are mapped out as the kernel is re-placed [15]; and per-bank memory health degrades as cumulative dose pushes error rates past correction thresholds [21].

Onto these physical models a fault-injection engine—applying the established methodology of deliberately injecting faults to measure how a system tolerates them [53]—imposes stochastic component death following Weibull distributions calibrated to space-grade electronics [54], together with cascading failures through shared thermal mass and power rails. The probe is then exercised against procedurally generated star systems seeded with planted anomalies—ranging from the obvious, an oxygen atmosphere, to the subtle, a trace biosignature at parts per billion—embedded in realistic sensor noise, so that detection performance can be scored against ground truth. Six metrics summarize a run: Mission Score (cumulative scientific value transmitted), Survival Duration, Energy Efficiency (science returned per joule), the Graceful Degradation Index, Decision Quality (were the planted anomalies found, and was noise correctly ignored), and Thermal Compliance. Where possible the models are cross-validated against the nearly fifty years of real degradation telemetry returned by the Voyager spacecraft [5]—the only ground-truth dataset of multi-decade unattended decline that humanity possesses. Accelerated, statistically faithful simulation is not a substitute for evidence in this domain; it is the only form of evidence available, and constructing it rigorously is itself a contribution. Figure 9 shows the overall structure of the framework.

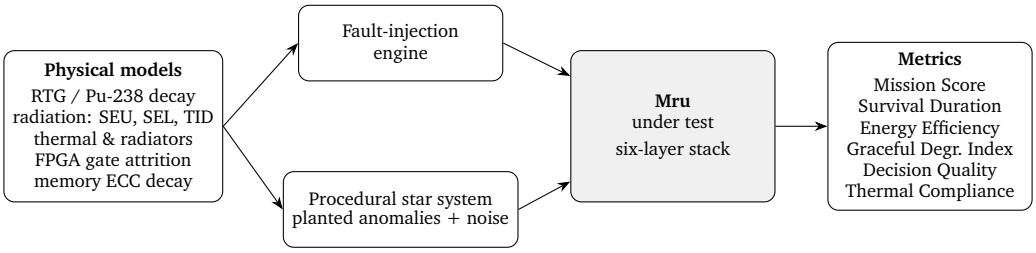

Figure 9: The simulation framework. Physics-grounded models of power, radiation, thermal behavior, and hardware decay drive a fault-injection engine and a procedurally generated star system seeded with known anomalies; the full six-layer stack is run against them and scored on six composite metrics. Because no probe can be flown, this accelerated, statistically faithful simulation is the paper's primary form of evidence—cross-validated, where possible, against five decades of Voyager telemetry.

15. Broader Implications and Adjacent Applications

The Degradation-First Principle generalizes well beyond interstellar probes. Its natural domain is any system expected to operate autonomously, under finite and declining resources, with no resupply and no maintainer reachable in time.

Off-world infrastructure. A Mars settlement's life-support, power, and agricultural controllers must run autonomously through communication blackouts and cannot be resupplied on demand; graceful degradation and cannibalization transfer directly.

Denied and disconnected environments. Autonomous underwater vehicles, polar and deep-ocean sensor networks, and orbital platforms share the probe's constraints—finite power, hardware degradation, constrained communication—on shorter but still maintenance-hostile timescales. Duty-cycling and knowledge triage apply with little modification.

Critical infrastructure and hazardous monitoring. Radiation-hardened autonomous systems in nuclear facilities, and the multi-millennial problem of marking nuclear-waste repositories so their hazard remains legible across geological time, are degradation-first problems in all but name [34].

Civilizational memory. Knowledge triage—deciding what to preserve as a storage substrate decays—is the general discipline of digital preservation, and it connects Mru to the intergenerational engineering of the Long Now and the Rosetta archive [32, 33].

Underlying all of these is a broader claim: most software is written on the unstated assumption that someone will maintain it. Dropping that assumption is not a minor variation but a distinct design discipline—the engineering of software that outlives its makers—and the interstellar case is simply the purest instance of a problem that an expanding range of terrestrial systems is beginning to face.

16. Open Problems and Limitations

Intellectual honesty requires confronting the boundaries of the proposal. Validation can never truly cover a thousand years: simulation, however faithful, rests on models whose rare-event tails and unmodeled couplings may diverge from reality precisely where it matters most, and the only real ground truth we possess spans decades, not millennia. The Tier-0 ROM that anchors the kernel's golden copy and the constitution is a physical single point of failure; its permanence is an assumption about a substrate that radiation and time also attack. The deliberate choice of rule-based science buys verifiability at a real cost—a system that recognizes only what it was told to look for cannot, by construction, recognize the genuinely unanticipated, which may be exactly the discovery most worth making. The constitution faces an analogous limit: values and priorities fixed at launch must govern encounters with situations no designer imagined, and a frozen specification may not survive contact with a sufficiently novel reality. And beneath all of it lie hard physical floors—on power, on memory retention, on thermal capacity—that no software can lift. Mru does not abolish these limits; it organizes a system to remain useful for as long as possible within them.

17. Conclusion

Mru begins from a single reorientation: that for a system meant to run for millennia without help, its own decay is not an exceptional fault but the governing condition of its existence. From that Degradation-First Principle, and from the observation that biology already solved unattended robustness through a replicated minimal interpreter executing behavior-as-data, a coherent architecture follows—a verified survival kernel replicated across reconfigurable hardware, hardware consensus that shrinks gracefully rather than failing at a threshold, knowledge triage across a tiered memory hierarchy, rule-based science chosen for its legibility over a thousand years, bandwidth-starved communication that degrades toward a final beacon, and a constitution etched in immutable hardware that bounds adaptation without permitting drift. Because the probe cannot be flown, the architecture is validated where it can be: in simulation grounded in real radiation, power, and failure physics, measured against the only multi-decade record of unattended decline we have.

The propulsion to send such a probe between the stars does not yet exist. The discipline required to keep one alive once it is sent can be built now. More than that, the same discipline—designing software to outlive its makers, to spend a declining store of capability for maximum cumulative effect, to remain useful while dying—is one a growing number of terrestrial systems already need. The interstellar probe is not a fantasy at the edge of engineering but the clearest possible statement of a problem that is quietly becoming general. Mru is an argument that the problem is tractable, and an open invitation to solve it.

References

[1] R. N. Bracewell, "Communications from superior galactic communities," *Nature*, vol. 186, no. 4726, pp. 670–671, 1960.

[2] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, "Remote agent: To boldly go where no AI system has gone before," *Artificial Intelligence*, vol. 103, no. 1–2, pp. 5–47, 1998.

[3] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, 2005.

[4] E. Normand, "Single-event upset at ground level," *IEEE Transactions on Nuclear Science*, vol. 43, no. 6, pp. 2742–2750, 1996.

[5] C. E. Kohlhase and P. A. Penzo, "Voyager mission description," *Space Science Reviews*, vol. 21, no. 2, pp. 77–101, 1977.

[6] U.S. Government Accountability Office, "Information technology: Federal agencies need to address aging legacy systems," Tech. Rep. GAO-16-468, U.S. Government Accountability Office, 2016.

[7] G. L. Bennett, "Space nuclear power: Opening the final frontier," in *4th International Energy Conversion Engineering Conference (AIAA 2006-4191)*, AIAA, 2006.

[8] M. A. Gibson, S. R. Oleson, D. I. Poston, and P. McClure, "NASA's Kilopower reactor development and the path to higher power missions," in *2017 IEEE Aerospace Conference*, pp. 1–14, IEEE, 2017.

[9] D. I. Poston, M. A. Gibson, R. G. Sanchez, and P. R. McClure, "Results of the KRUSTY nuclear system test," *Nuclear Technology*, vol. 206, no. 1S, pp. 89–117, 2020.

[10] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies* (C. E. Shannon and J. McCarthy, eds.), pp. 43–98, Princeton University Press, 1956.

[11] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[12] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[13] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[14] J. R. Schwank, M. R. Shaneyfelt, D. M. Fleetwood, J. A. Felix, P. E. Dodd, P. Paillet, and V. Ferlet-Cavrois, "Radiation effects in MOS oxides," *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 1833–1853, 2008.

[15] M. Wirthlin, "High-reliability FPGA-based systems: Space, high-energy physics, and beyond," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 379–389, 2015.

[16] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[17] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[18] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[19] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[20] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proceedings of IEEE International Conference on Communications (ICC '93)*, pp. 1064–1070, IEEE, 1993.

[21] S. Gerardin and A. Paccagnella, "Present and future non-volatile memories for space," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3016–3039, 2010.

[22] A. M. Saleh, J. J. Serrano, and J. H. Patel, "Reliability of scrubbing recovery techniques for memory systems," *IEEE Transactions on Reliability*, vol. 39, no. 1, pp. 114–122, 1990.

[23] B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*. Garland Science, 6th ed., 2014.

[24] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, p. 1628, 2012.

[25] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 7435, pp. 77–80, 2013.

[26] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "seL4: Formal verification of an OS kernel," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP '09)*, pp. 207–220, ACM, 2009.

[27] V. Astrauskas, P. Müller, F. Poli, and A. J. Summers, "Leveraging Rust types for modular specification and verification," *Proceedings of the ACM on Programming Languages (OOPSLA)*, vol. 3, pp. 1–30, 2019.

[28] Amazon Web Services, "Kani Rust verifier." `https://github.com/model-checking/kani`, 2024.

[29] Y. Bai, S. Kadavath, S. Kundu, A. Askell, *et al.*, "Constitutional AI: Harmlessness from AI feedback." arXiv preprint arXiv:2212.08073, 2022.

[30] N. Soares, B. Fallenstein, S. Armstrong, and E. Yudkowsky, "Corrigibility," in *Workshops at the 29th AAAI Conference on Artificial Intelligence*, 2015.

[31] S. Russell, *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking, 2019.

[32] S. Brand, *The Clock of the Long Now: Time and Responsibility*. Basic Books, 1999.

[33] The Long Now Foundation, "The Rosetta project." `https://rosettaproject.org`, 2024.

[34] K. M. Trauth, S. C. Hora, and R. V. Guzowski, "Expert judgment on markers to deter inadvertent human intrusion into the Waste Isolation Pilot Plant," Tech. Rep. SAND92-1382, Sandia National Laboratories, 1993.

[35] M. M. Lehman, "Programs, life cycles, and laws of software evolution," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, 1980.

[36] D. L. Parnas, "Software aging," in *Proceedings of the 16th International Conference on Software Engineering (ICSE '94)*, pp. 279–287, IEEE, 1994.

[37] D. G. Gilmore, ed., *Spacecraft Thermal Control Handbook, Volume I: Fundamental Technologies*. The Aerospace Press / AIAA, 2nd ed., 2002.

[38] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, 2000.

[39] H. Kitano, "Biological robustness," *Nature Reviews Genetics*, vol. 5, no. 11, pp. 826–837, 2004.

[40] J. von Neumann, *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966. Edited and completed by A. W. Burks.

[41] D. Mange, M. Sipper, A. Stauffer, and G. Tempesti, "Toward robust integrated circuits: The Embryonics approach," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 516–541, 2000.

[42] E. W. Dijkstra, "The structure of the "THE"-multiprogramming system," *Communications of the ACM*, vol. 11, no. 5, pp. 341–346, 1968.

[43] B. Randell, "System structure for software fault tolerance," *IEEE Transactions on Software Engineering*, vol. SE-1, no. 2, pp. 220–232, 1975.

[44] C. C. Liebe, "Star trackers for attitude determination," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 6, pp. 10–16, 1995.

[45] Gaia Collaboration, T. Prusti, J. H. J. de Bruijne, A. G. A. Brown, *et al.*, "The Gaia mission," *Astronomy & Astrophysics*, vol. 595, p. A1, 2016.

[46] D. J. Des Marais, M. O. Harwit, K. W. Jucks, J. F. Kasting, D. N. C. Lin, J. I. Lunine, J. Schneider, S. Seager, W. A. Traub, and N. J. Woolf, "Remote sensing of planetary properties and biosignatures on extrasolar terrestrial planets," *Astrobiology*, vol. 2, no. 2, pp. 153–181, 2002.

[47] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[48] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[49] K. S. Andrews, D. Divsalar, S. Dolinar, J. Hamkins, C. R. Jones, and F. Pollara, "The development of turbo and LDPC codes for deep-space applications," *Proceedings of the IEEE*, vol. 95, no. 11, pp. 2142–2156, 2007.

[50] S. M. Omohundro, "The basic AI drives," in *Proceedings of the First Conference on Artificial General Intelligence (AGI 2008)*, pp. 483–492, IOS Press, 2008.

[51] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Transactions on Computers*, vol. C-29, no. 8, pp. 720–731, 1980.

[52] C. Sagan, L. S. Sagan, and F. Drake, "A message from Earth," *Science*, vol. 175, no. 4024, pp. 881–884, 1972.

[53] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer, "Fault injection techniques and tools," *Computer*, vol. 30, no. 4, pp. 75–82, 1997.

[54] W. Weibull, "A statistical distribution function of wide applicability," *Journal of Applied Mechanics*, vol. 18, no. 3, pp. 293–297, 1951.