RobotRSI: Toward Full-Stack Recursive Self-Improvement of Robots through Autonomous Research

A Survey and Perspective

Pengsong Zhang and OpenAGS

Abstract

Robots are increasingly able to collect their own experience, update policies, infer models of their bodies, search designs, and participate in automated experiments. In parallel, AI scientists and engineering agents are beginning to formulate hypotheses, write and execute code, coordinate simulation tools, analyse results, and propose revised designs. These developments are usually studied in separate communities and at different layers of the robotics stack. As a result, *self-improving robot* may denote anything from online adaptation of a fixed controller to the automated design and manufacture of a successor machine, while recursive self-improvement is often invoked without evidence that one update made subsequent research more effective. This Survey and Perspective introduces **RobotRSI**, a unifying framework for autonomous research across robot engineering and for the verified return of research outputs to robots, robot collectives, or successor systems. We organize the field along three orthogonal dimensions: the engineering object being improved, the research activity being automated, and the robot context in which the improvement is produced and evaluated. The engineering scope spans morphology, materials, actuation, sensing, energy, electronics, perception, self- and world models, control, planning, data, learning, simulation, experimentation, manufacturing, deployment, and the research process itself. We distinguish AI scientists for robotics from robot scientists for robotics, and separate the research executor, research target, and improvement recipient. On this basis, we synthesize progress in autonomous hardware, software, and system-level engineering; analyse how constraints vary across general-purpose and specialized robots; and define evidence requirements for local improvement, system update, loop closure, and recursive benefit. The central conclusion is that important local loops already exist, including autonomous policy improvement, self-model revision, morphology search, and physical experimentation, but present evidence is fragmented across system boundaries. Full RobotRSI therefore remains a systems-research agenda rather than a demonstrated monolithic capability. We propose an evaluation framework, reference architecture, and testable roadmap for connecting these local loops without conflating automation, autonomy, embodiment, and recursion.

Keywords: recursive self-improvement; autonomous robotic research; AI scientist; robot scientist; robot co-design; evolutionary robotics; autonomous experimentation; self-improving robots; embodied intelligence; agentic engineering.

Contents

1 Introduction: The Vision of RobotRSI

1.1 From robots that learn to robots that research

Robotics has long pursued machines that improve through interaction. A controller may adapt to a changed payload, a perception model may learn from deployment data, a robot may infer an altered body model after damage, and an evolutionary process may generate a better morphology. These are consequential forms of improvement, but they do not all automate the same work. Some update task behaviour inside a fixed engineering envelope; some search an engineer-defined design space; some close a physical experiment loop; and only a small subset returns a validated result to a deployed robot or a manufactured successor. The distinction matters because robot engineering is not a single optimization problem. It is a coupled process of specifying requirements, selecting representations, designing hardware and software, executing computational and physical experiments, diagnosing failures, integrating components, and validating an updated system.

Early work already exposed several parts of this wider loop. Automated evolution and fabrication connected virtual morphology search to physical robot construction [1]. A modular "mother robot" later assembled and evaluated successive physical phenotypes, making the experiment-and-selection loop tangible while keeping the producing robot distinct from the robots it produced [2]. Continuous self-modelling showed that a robot could use interaction to infer aspects of its own body and recover behaviour after damage [3]. More recent automatic design methods greatly shorten the search for physically realizable robot bodies [4], while real-robot learning systems reduce repeated human supervision by learning rewards, practising reversible tasks, or curating autonomous experience [5, 6]. Together, these works demonstrate that robot improvement can involve bodies, models, data, objectives, and experimental procedures—not only policy weights.

At the same time, automated science has changed the plausible coordinator of this work. The Robot Scientist lineage made a strong claim unusually early: Adam connected machine-generated functional-genomics hypotheses to automatically selected and physically executed experiments, and later work emphasized formal records and reproducibility; Eve extended the architecture to drug-repositioning campaigns [7–10]. Self-driving laboratories (SDLs) broaden this digital–physical pattern through algorithmic experiment selection, robotic execution, and iterative learning. Examples range from thin-film optimization and mobile chemical search to language-model-mediated chemistry and autonomous inorganic synthesis [11–14]. These systems are central precedents for embodied research, but their research target is ordinarily a biological, chemical, or materials system; the laboratory robot is an executor, not necessarily the artifact being improved.

Contemporary AI-scientist systems extend research automation in computational domains by generating directions, modifying code, running experiments, maintaining notes, writing manuscripts, and conducting machine-mediated review [15, 16]. Recent surveys organize this rapidly moving area by scientific workflow, agent architecture, capability, autonomy, and application domain [17–21]. Engineering agents likewise connect language models or search procedures to finite-element analysis, control design, data curation, and materials simulation. None of these capabilities, by itself, constitutes a self-improving robot. Their importance is compositional: they supply research functions that may be connected to robot-specific models, simulators, test facilities, manufacturing systems, and deployment telemetry.

This paper asks what happens when these lines of work are treated as one engineering problem. We use **RobotRSI** to denote the research programme in which robot-engineering activities are increasingly automated, their validated outputs update a robot system or its successors, and selected updates measurably improve the system's capacity to conduct later improvement. RSI retains its established expansion—**recursive self-improvement**—but its use here is deliberately operational rather than rhetorical. An update is recursive only when its downstream contribution to later R&D can be identified and evaluated. Repeated gains on the same training objective, self-adaptation during task execution, or a faster base model do not by themselves establish such a feedback relation.

1.2 Why a full-stack engineering view is necessary

The word *full-stack* describes the space of eligible research targets, not a requirement that every iteration simultaneously redesign every component. A robot may be limited by sensing rather than morphology, by

thermal capacity rather than planning, or by experimental reset rather than the learning algorithm. A credible self-improving system must therefore decide *where* to spend a limited research budget and must propagate local changes through coupled constraints. A lighter link can reduce actuator load yet increase compliance and model uncertainty. A higher-resolution tactile sensor can improve manipulation yet change wiring, compute, calibration, power, and thermal requirements. A new controller may perform well in simulation but fail because the available machine cannot reproduce the assumed geometry or contact conditions. Practical work on evolvable hardware makes this point sharply: power delivery, connectivity, manufacturing, and assembly define a viable phenotype space that can differ substantially from the unconstrained space imagined by an optimizer [22].

This coupling also explains why data, learning, simulation, and experimentation should not be treated as a fourth engineering stack parallel to hardware, software, and system engineering. They are cross-cutting means and assets. Atomistic simulation can support a material decision; finite-element analysis can support a mechanism; system identification can update a controller model; autonomous data collection can update a perception or policy model; and a physical test can reject any of these outputs. The research activity should be described where its principal engineering object changes, while common orchestration, evidence, and hand-off mechanisms are analysed separately.

A full-stack view is also broader than morphological evolution. The proposal of multi-level evolution—from molecular and material building blocks to morphology and sensorimotor organization—established an important cross-layer precedent [23]. Adaptive-morphogenesis research further examines machines whose structural and actuation systems can express useful body changes on demand [24]. RobotRSI includes both perspectives but asks a different systems question: which *research activities* produced a change, who or what made the relevant decisions, how was the result validated and adopted, and did the resulting system become better at subsequent research? The novelty claim is therefore not "the first full-stack view of robots," but an evidence-oriented connection between full-stack robot engineering, autonomous research, versioned system updates, and recursive research benefit.

Prior surveys already connect several engineering layers. Li et al. organize automated morphology, controller, and vision design through modular graphical representations [25]. Howison et al. describe the exchange between virtual design search and large-scale physical experiments in reality-assisted evolution [26]. Alberts et al. systematically map software architecture-based self-adaptation in robotics, separating the managing system, managed robot, adaptation mechanism, and evaluation strategy across 37 primary studies [27]. Reviews of SDLs additionally analyse enabling hardware and software, laboratory integration, scale, safety, provenance, and performance metrics [28–34]. A recent AI-focused RSI survey distinguishes bounded self-refinement from changes to policies, evaluators, and the research process [35]. RobotRSI builds on all of these connections. Its additional question is how autonomous research targets the versioned robot-engineering stack and how an accepted output changes the system that performs subsequent engineering work, across both computational and physical activities. This motivates tracing the research executor, update recipient, and later research benefit together, rather than claiming novelty for combining body and controller design, for naming a closed experiment loop, or for relabelling runtime adaptation as research.

1.3 Four claims that should not be collapsed

The literature uses *adaptive*, *autonomous*, *self-improving*, *self-evolving*, and *recursive* with overlapping meanings. To prevent labels from deciding the conclusion, we separate four claims:

1. **Task adaptation:** a robot changes state, parameters, a model, or behaviour in response to experience or damage and improves an operational metric.

2. **Autonomous R&D:** a system performs at least one consequential research decision—such as selecting a candidate, experiment, model, design, or diagnostic—within an engineering process.

3. **System self-improvement:** a validated research artifact is accepted into the defined robot system, collective, or successor version and improves an evaluated property.

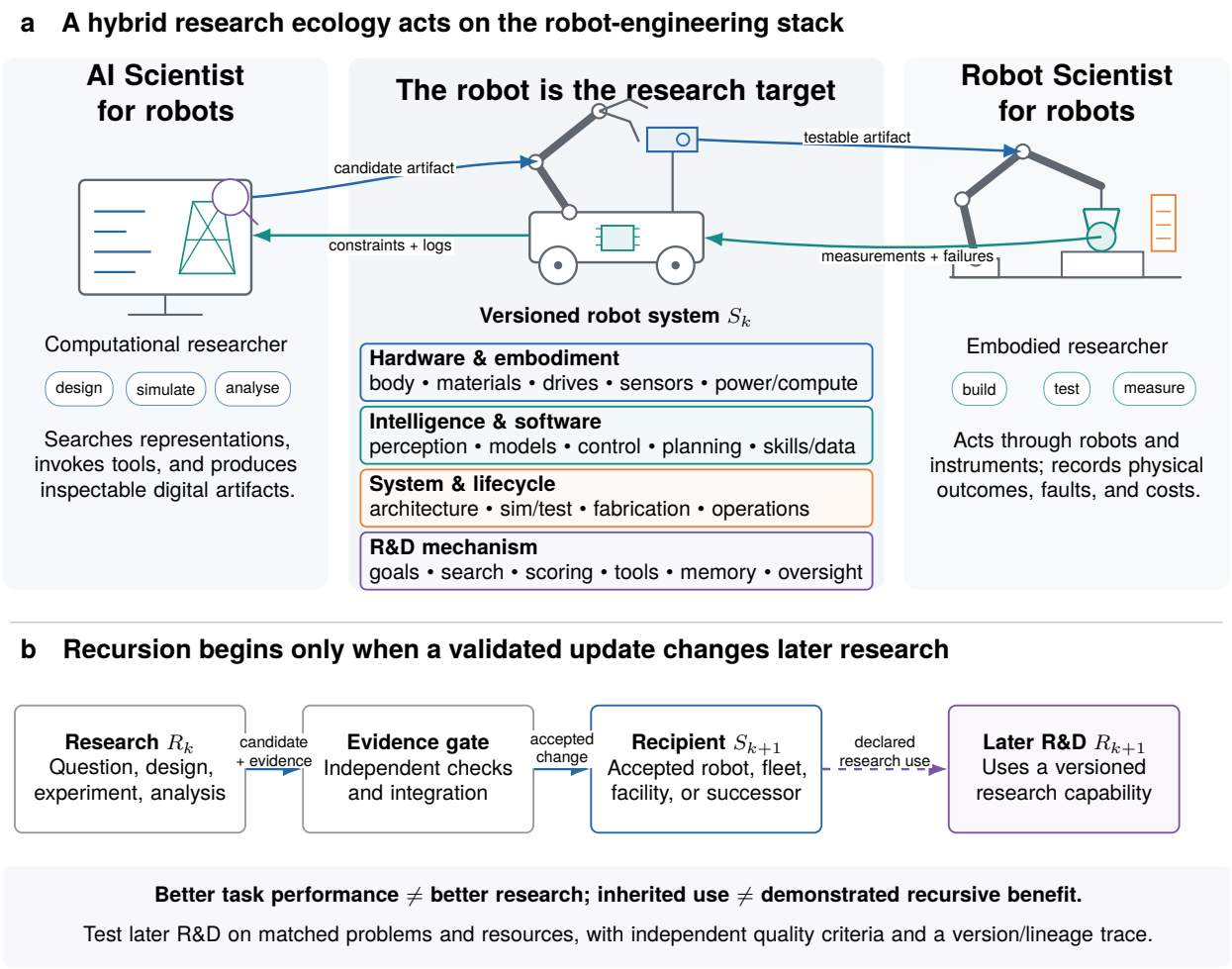

Figure 1: RobotRSI treats robot engineering as the object of a hybrid autonomous-research system. (a) The central robot stack is researchable from embodiment through the R&D mechanism itself. An AI Scientist role produces computational artifacts; a Robot Scientist role builds, manipulates, and measures physical artifacts. Blue arrows carry candidate or testable artifacts toward an executor or recipient; teal arrows return recorded constraints and evidence. The roles can be distributed across agents, robots, instruments, facilities, and people rather than residing onboard one robot. (b) A candidate becomes system self-improvement only after validation and adoption by a declared recipient. The dashed purple arrow records later use of an inherited research capability; a matched downstream comparison is additionally required to claim recursive benefit. The diagram specifies interfaces and evidential conditions, not an existing end-to-end implementation.

4. **Recursive research improvement:** the accepted update causally or credibly contributes to better performance on later research or engineering tasks under a stated comparison.

These claims can coexist, but none is inferred automatically from the previous one. Self-improving semantic perception during deployment, for example, provides strong evidence for autonomous model update and downstream localization benefit [36]. It need not include autonomous selection of a new learning algorithm. Likewise, a robot that adapts to damage by searching a behavioural repertoire demonstrates rapid operational recovery [37], not autonomous redesign of the damaged component. Conversely, an AI system may automate substantial research planning and computational experimentation while never updating a robot. The framework in Figure 1 makes these distinctions explicit.

1.4 Scope and research questions

We study autonomous research *on robot engineering*. This includes research executed entirely in software, research that uses robots as experimental instruments, and research conducted by a distributed system containing AI agents, robots, simulators, test equipment, manufacturing tools, cloud services, and people. We

do not require the physical robot to contain every research component. Such a requirement would obscure realistic system boundaries and would exclude successor construction, fleet learning, and remotely supported field robots. Instead, every case records three roles: the **research executor**, the **research target**, and the **improvement recipient**. A laboratory manipulator that tests an underwater propulsor is an experimental executor; it is not thereby the robot being improved. A cloud agent that updates a fleet policy can belong to the defined R&D system if that boundary and its dependencies are reported.

The survey is guided by seven questions:

- **RQ1:** What distinguishes robot adaptation, autonomous robotic R&D, system self-improvement, and recursive self-improvement?
- **RQ2:** Which research activities have been automated for each hardware, software, and lifecycle engineering object?
- **RQ3:** How do research decisions, computational tools, physical experiments, and human expertise interact?
- **RQ4:** How do robot morphology, task breadth, scale, environment, and system organization change the feasible improvement loop?
- **RQ5:** How are local research outputs validated, integrated, and inherited by robots or successor systems?
- **RQ6:** What evidence and benchmarks are required to evaluate autonomy, improvement, loop closure, and recursion?
- **RQ7:** Which missing interfaces and experiments most constrain progress toward sustained RobotRSI?

1.5 Contributions and organization

This Survey and Perspective makes four contributions. First, it introduces a three-dimensional taxonomy—**engineering object × research activity × robot context**—that preserves full-stack coverage without turning every method or robot class into a separate chapter. Second, it proposes a reference architecture and evidence model that traces artifacts from problem formulation through computation and physical experimentation to system adoption. Third, it separates evidence for operational capability, R&D autonomy, physical realization, system update, and recursive benefit, yielding a benchmark design that reports human and resource dependencies. Fourth, it derives a research agenda centred on testable missing connections rather than forecasts of unbounded or inevitable improvement.

Section 2 defines the scope, historical foundations, taxonomy, and review method. Section 3 describes common mechanisms of autonomous robotic R&D. Sections 4 and 5 survey hardware/embodiment and intelligence/software, respectively; Section 6 connects them through system-level engineering. Section 7 compares robot types and operating contexts. Section 8 analyses loop closure and recursion, Section 9 develops evaluation protocols, and Section 10 presents the research agenda. Section 11 states the resulting perspective and its limitations.

2 Scope, Foundations, and Review Methodology

2.1 Unit of analysis: research transitions, not paper labels

The fundamental unit in this survey is a **research transition**:

$$\tau = \langle S_i, q, a, e, z, u, S_j \rangle,$$

where S_i is the source system version, q is an engineering question or requirement, a is a sequence of research decisions and actions, e is the resulting evidence, z is a candidate research artifact, u is the acceptance and integration operation, and S_j is the recipient version. A transition can be partial: a paper may end at a simulated candidate z, at a physical experiment e, or at deployment in S_j. Keeping partial transitions is

essential for a survey of an emerging system: excluding them would hide the components from which fuller loops must be built. At the same time, the missing edge is recorded rather than imagined.

A **research artifact** is any persistent output that can change a system or a later R&D process: a material recipe, CAD model, morphology, actuator specification, sensor layout, calibration, controller, policy, model, data set, reward, curriculum, simulator parameterization, test procedure, manufacturing plan, diagnosis, code patch, experiment log, or distilled research memory. An artifact becomes an **update** only after its acceptance into a recipient version has been demonstrated. Publication of a candidate or success in an isolated simulator is not acceptance.

A recursive relation exists when an update u_k changes the effectiveness of a later research transition τ_{k+1}. For research performance measure J_R, a minimal empirical claim has the form

$$\Delta J_R = J_R(\tau_{k+1} \mid S_{k+1}, B, Q) - J_R(\tau_{k+1} \mid S_k, B, Q) > 0,$$

where B is a matched resource and human-support budget and Q is a new or held-out distribution of research problems. This notation does not imply that research quality is one-dimensional. It makes the comparison conditions explicit: if the improved system receives a larger model, more experiments, easier questions, or additional human intervention, those changes must be reported rather than attributed silently to recursion.

2.2 Operational definitions

Table 1 defines the principal terms. The definitions are intentionally evidence-oriented. They do not attempt to settle every philosophical meaning of *self* or *autonomy*; they state what evidence is required for the comparative claims made in this paper.

2.3 What is the "self" in RobotRSI?

Robotic systems are distributed. Onboard compute may run control while a remote server trains a model; a fleet may aggregate data; a fabrication cell may construct a successor; and people may set goals, service equipment, or approve deployment. We therefore define the self by an explicit **system boundary**, not by the surface of a robot body. Three common boundaries are legitimate:

1. **Individual boundary:** one robot and its onboard resources. This is the strongest interpretation for in-situ adaptation but excludes many practical training and manufacturing dependencies.
2. **Collective boundary:** a fleet or modular collective that shares experience, compute, components, or control. Improvement may be inherited by many recipients.
3. **R&D-system boundary:** robots, AI services, simulation, laboratories, manufacturing cells, databases, and authorized human roles form a versioned socio-technical system.

No boundary receives an automatic maturity advantage. The individual boundary may conceal human-prepared environments; the wider boundary may truthfully include indispensable infrastructure. What matters is that external dependencies crossing the boundary are reported. Figure 2 visualizes the roles that are often collapsed by the phrase *a robot improves itself*.

2.4 Three complementary classification dimensions

2.4.1 Engineering objects: what changes

We identify 18 object families. Five concern physical embodiment: morphology/mechanisms/materials (O01), actuation/transmission (O02), sensing hardware (O03), energy/thermal systems (O04), and compute/electronics/communications (O05). Five concern the intelligence and software stack: perception/state estimation (O06), self/world/interaction models (O07), control/motion generation (O08),

Table 1: Operational definitions and minimum evidence requirements.

Term	Operational definition	Minimum affirmative evidence	Insufficient by itself
Adaptation	A change that maintains or improves operation under a changed task, body, or environment.	Pre/post or matched comparison on the stated operational property.	The method is called adaptive; parameters change without an evaluated consequence.
Autonomous optimization	Candidate variables are changed and evaluated by a predefined algorithm with bounded objectives and constraints.	Explicit variables, evaluator, search operation, and selected outcome.	A person manually chooses among generated candidates.
Autonomous R&D	The system makes one or more consequential decisions in problem selection, proposal, experiment choice, interpretation, verification, or integration.	Activity-level record of the machine decision, inputs, tools, outputs, failures, and human role.	One code completion, tool invocation, or unattended predetermined script.
AI scientist for robotics	A computational research system performs scientific or engineering reasoning for a robot-related target.	Robot-engineering question plus traceable computational activities and artifacts.	A chatbot answers a robotics question without execution or validation.
Robot scientist for robotics	A system involving physical robots autonomously executes or participates in robot-engineering experiments.	Physical intervention/measurement, decision boundary, result, and robot-engineering target.	A robot performs an application task; a lab robot studies a non-robot object.
Robot self-improvement	A validated artifact is incorporated into the defined robot, collective, or successor and improves a target property.	Source/recipient versions, acceptance operation, and post-update evaluation.	A design is only simulated; a related later paper reports a better robot.
Recursive robot self-improvement	An accepted update improves the capability or efficiency of subsequent autonomous R&D.	Two linked research transitions and a matched comparison of later R&D.	Repeating one training loop, increasing task reward, or receiving an external model upgrade.
Open-ended improvement	Useful novelty continues beyond a fixed finite target set without a predetermined terminal solution.	Long-horizon novelty, utility, retention, resource, and failure evidence under changing challenges.	More iterations on a fixed benchmark or a claim of indefinite scaling.

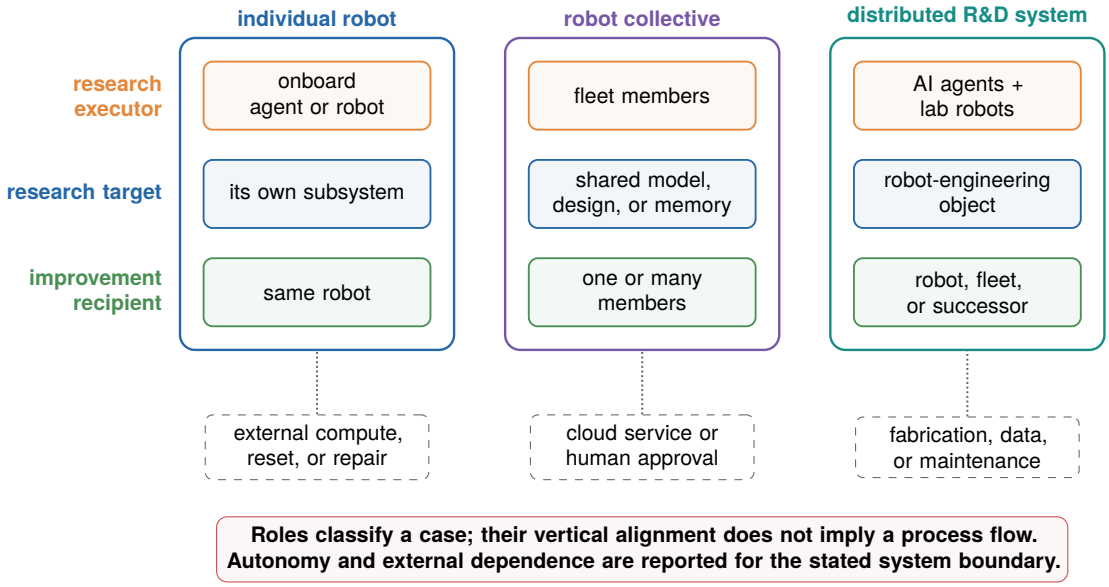

Figure 2: Role separation under three possible system boundaries. Research executor, research target, and improvement recipient are independent descriptors: they may coincide in an individual robot or be distributed across a fleet and an R&D facility. Dotted, non-directional links identify dependencies outside the declared boundary; they do not imply that a dependency autonomously performs research.

skills/planning/interaction (O09), and software architecture/runtime (O10). Three capture reusable learning assets and choices: data/memory/knowledge (O11), models/learning algorithms (O12), and objectives/rewards/curricula (O13). Four concern experimental realization and the lifecycle: simulation/digital experiments (O14), physical experiments/test facilities (O15), manufacturing/assembly/reconfiguration (O16), and deployment/maintenance/fleet updates (O17). The final family is the R&D mechanism itself (O18): problem selection, research planning, tool use, verification, budgeting, and scientific memory.

These tags are non-exclusive, but an object tag denotes what the research changes, not every tool or subsystem it uses. A tactile-sensor project can change material (O01), sensor hardware (O03), estimation software (O06), and electronics (O05); using an existing test fixture does not by itself make facility design (O15) an intervention. We record supporting instruments separately. For narrative clarity a study receives one primary location, while its records preserve additional changed objects when supported by the source.

2.4.2 Research activities: what the autonomous system does

Engineering objects should not be confused with research activities. Across any object, an autonomous R&D system can: (A01) formulate a problem or requirement; (A02) retrieve and organize knowledge; (A03) propose hypotheses, designs, or implementations; (A04) select experiments and allocate resources; (A05) execute computation; (A06) execute physical experiments; (A07) interpret results and diagnose failure; (A08) verify, reproduce, or compare; (A09) accept and integrate an artifact; and (A10) preserve and transfer research memory. This sequence is not always linear. A failed physical test can change the model, design space, experiment plan, or even the problem definition.

2.4.3 Robot context: where improvement must work

The third dimension records constraints rather than producing a catalogue of application areas. We separate intended task breadth from demonstrated coverage; morphology and locomotion mode; rigid, soft, hybrid, biohybrid, and micro-scale realization; task domain; environment; and individual, collective, or distributed organization. A humanoid morphology does not imply a general-purpose system, and a generalist policy does not imply that the robot's hardware, lifecycle, or R&D process is general. Conversely, a specialized underwater or surgical robot can support deep and sustained self-improvement within a narrow mission.

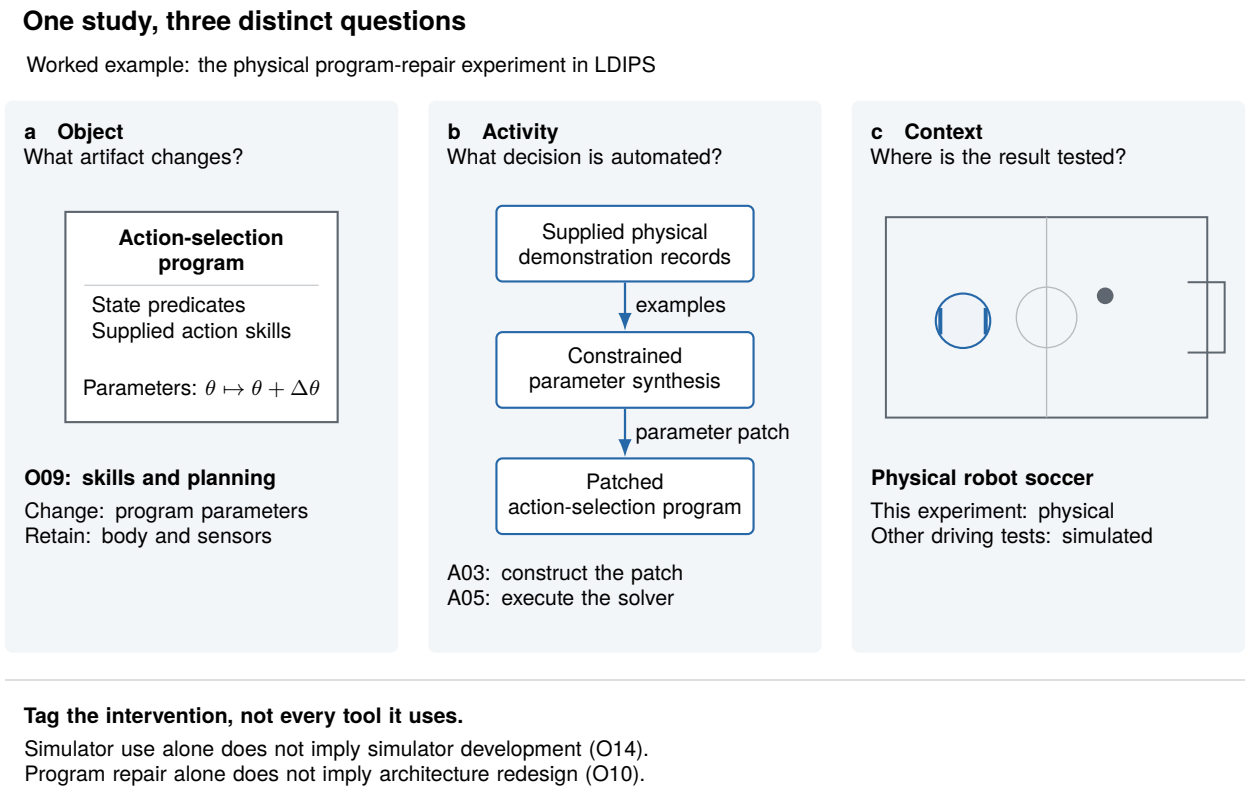

Figure 3: Applying the object–activity–context taxonomy to a reported study. LDIPS's physical repair experiment [38] is read through three complementary views. The object tag concerns the changed action-selection program; the activity tags identify patch construction and solver execution; the context specifies the physical soccer test. Supplied demonstrations do not establish autonomous data collection; physical-trial execution and human roles must be recorded separately. The paper's simulated driving tests are a different experimental context, and neither setting establishes general-purpose competence. The two arrows inside (b) carry demonstration examples and the resulting parameter patch, not a causal ordering between taxonomy dimensions. The program and field drawings are original abstractions, not copied code or measured trajectories. A study may contain additional interventions requiring their own records.

The dimensions answer different questions; they are not assumed statistically independent. Figure 3 illustrates their use with the physical program-repair part of LDIPS [38]. The changed object is an action-selection program, the consequential computation is constrained parameter synthesis, and the tested recipient is a soccer robot. Physical demonstrations are supplied inputs, not evidence that the system autonomously chose and collected them. Other experiments in the same paper retain their own context and realization labels.

2.5 Historical foundations and adjacent literatures

RobotRSI is an intersection, not a replacement label. At least six traditions contribute necessary concepts and mechanisms.

Evolutionary and automated robot design treats bodies and controllers as search variables and provides representations, variation operators, quality-diversity mechanisms, and fabrication loops [1, 2, 23]. Its strengths are population-level exploration and explicit heredity; its physical bottlenecks include evaluation cost, viable phenotype constraints, and the reality gap [22].

Self-modelling, resilient robotics, and adaptive control use interaction to estimate a robot's changed dynamics or search compensatory behaviour [3, 37, 39]. These works supply an immediate feedback route from embodiment to models and action, although the update target is usually behaviour or a predictive model rather than the engineering method that will design the next robot.

Continual, self-supervised, and autonomous robot learning reduces dependence on curated demonstrations and human resets. Examples include continual semantic perception during deployment [36],

Table 2: Dimensions used to classify RobotRSI evidence.

Dimension	Principal values	Reason for separation
Engineering object	O01–O18, from materials and morphology to deployment and R&D mechanisms	Identifies the artifact and subsystem that actually changes.
Research activity	Problem, knowledge, proposal, experiment choice, digital/physical execution, interpretation, verification, integration, memory	Distinguishes autonomous decisions from unattended execution.
Research role	Executor, target, improvement recipient	Prevents an experimental robot from being mistaken for the robot under improvement.
Change form	Parameter, weights, program, data, geometry, material, component, architecture, process	Exposes update magnitude and reversibility.
Evidence environment	Analysis, simulation, component test, physical robot, field deployment	Prevents simulated feasibility from becoming physical loop closure in synthesis.
Robot context	Breadth, embodiment, scale, domain, environment, organization	Enables comparison without forcing mutually exclusive robot classes.
Human/external dependence	Goal, initialization, filtering, reset, repair, approval, external services	Makes autonomy conditional on the actual system boundary.
Feedback status	Candidate, evaluated, accepted, reused in R&D, recursive benefit evaluated	Separates local automation from loop closure.

task/undo-task practice and reward inference [5], and foundation-model-assisted collection and evaluation of autonomous instruction-following data [6]. This tradition offers scalable experience generation and persistent learning, while raising retention, evaluator reliability, distribution shift, and safety questions.

Embodied intelligence and adaptive morphology emphasize that behaviour is produced by coupled materials, morphology, sensing, actuation, and environment. Multi-level evolution proposes design across these layers [23], and adaptive morphogenesis focuses on useful structural change across time scales [24]. RobotRSI uses these accounts to avoid software exceptionalism, but adds the R&D activities and evidence trail that produce, test, and accept a change.

Robot scientists and self-driving laboratories integrate hypothesis handling, experiment planning, robotic execution, analysis, and knowledge revision. Adam and Eve established physically grounded, formally recorded Robot Scientist cycles in biology and drug discovery [7–10]. Later SDLs joined closed-loop optimization or active learning to modular laboratory automation in thin films, chemistry, and inorganic materials [11–14]. Reviews of this literature expose issues that transfer directly to robotic R&D: platform modularity, experimental-space definition, precision, throughput, human labour, safety, interoperability, provenance, and cross-laboratory scaling [28–34]. A Robot Scientist *for robotics* must additionally manage the hazards, resets, metrology, wear, calibration, and version identity of the target machines. The capitalization is intentional here when referring to the historical Adam/Eve programme; the generic role elsewhere need not be a humanoid or even one physical machine.

AI scientists and agentic engineering offer general-purpose research planning, coding, tool orchestration, and evaluation [15, 16]. Adjacent surveys variously organize the field around research-process stages, agent mechanisms, autonomy levels, scientific domains, or a progression from knowledge acquisition to verification and evolution [17–21]. These are complementary views of the researcher; they do not make the engineered robot their coverage unit. Their current evidence is strongest in computational domains. Transferring them to robotics changes the cost and consequence structure: physical experiments are slow, stateful, destructive, difficult to reproduce, and coupled to human and facility safety. RobotRSI therefore treats them as research coordinators whose outputs require domain solvers, instruments, validators, and controlled deployment.

No single axis separates all of these literatures. Three comparisons are especially important. First, the modular design automation survey covers morphology, controllers, vision, and their integration, including swarm-control design [25]; RobotRSI therefore cannot claim novelty merely from spanning body and control. Second, the reality-assisted evolution review organizes model-based search, physical experimentation, and the

return of real data to virtual models [26]; a digital–physical loop is likewise not sufficient. Third, AI-focused RSI work asks whether an AI system improves behaviour, policy, evaluators, or the research process [35], but does not make the coupled hardware–software–lifecycle robot stack or embodied engineering hand-offs its central unit. Table 3 positions the present survey using four discriminators: what is being changed, whether physical realization is constitutive, which research decisions are delegated, and whether later R&D benefit is part of the claim. The table reports organizing criteria of the cited syntheses, not absence of individual counterexamples.

2.6 Review design and evidence coding

Retrieval strategy and scope. We use an object-led structured survey rather than treating the label RobotRSI as a search term that defines a literature. The target scope is the eighteen engineering-object families above. Retrieval combines robotics-specific searches, searches for transferable engineering methods, and citation tracing. Automation terms include evolutionary design, Bayesian optimization, active experimentation, automatic synthesis, and agentic research; an LLM is not an inclusion requirement. This choice admits older engineering automation while preserving the distinction between fixed-objective search and autonomous research direction.

Discovery in the present revision has two traceable components. First, targeted web and citation searches are checked against publisher, proceedings, author, institutional, or other authoritative sources. Second, a fixed open-index snapshot covers 42 query families spanning O01–O18 and eight robot contexts. On 8 September 2026, the executable search retrieved the top 50 relevance-ranked results per query where available from Crossref and arXiv. All raw responses are retained with hashes, request URLs, times, provider-reported totals, and errors. DOI/title deduplication reduced 2,695 returned records to 2,423 candidates. A deliberately strict title rule placed 525 candidates in a priority queue by requiring both a robot/embodiment expression and an engineering-development expression. This rule reduces workload; it is not an eligibility filter and is not used to exclude indirect titles.

The snapshot does not constitute a completed search of every database named in the planning protocol. Crossref's permissive bibliographic relevance search produces very large reported hit totals, while arXiv's conjunctive all-term search is narrower; neither total is treated as the size of the field. Search rankings and per-query caps are not a probability sample, and candidate overlap across query axes prevents interpretation as object prevalence. Table 4 reports what was executed and what remains open. The accompanying configuration, raw responses, screening queues, scripts, and evidence records preserve reproducibility without converting retrieval output into included-study claims.

Eligibility and use in the synthesis. A robot-related development study is eligible when it identifies an engineering object and implements at least one consequential development operation: proposing or searching a design, choosing an experiment, learning from feedback, generating a development artifact, or updating a recipient. Ordinary task execution without such a development relation is not core evidence. A manually designed system can still enter as a foundation or comparison, provided that its role is explicit.

We assign non-exclusive use roles: A, robot-related development; B, transferable engineering methods with a stated robotics connection; C, foundations and infrastructure; and D, evaluation and reproducibility. These are not quality grades. A battery-chemistry laboratory can offer strong physical experimental evidence in B without demonstrating robot self-modification; an FPGA design method can supply C evidence even when its original implementation was manual. Domain relevance, automation, and physical realization are assessed separately.

Publication families and evidence units. Bibliographic records identify sources, not independent experiments. Preprints and final articles are linked; changed titles or venues do not alone create a new study. Follow-up systems remain distinct when they add a method, a recipient, or a new experiment. The Robomorphic Computing–RoboShape progression illustrates why this matters: a proposed automation route and a later

Table 3: RobotRSI at the intersection of adjacent survey traditions. Entries identify each literature's organizing criterion; they do not deny the existence of individual cross-over systems or imply a maturity ranking.

Synthesis or tradition	Primary unit of analysis	Principal change target	Relation to physical realization	Consequential research decision	Later-R&D benefit as a criterion?
Modular robot-design automation [25]	Modular design representation	Morphology, controller, vision, and their integration	Computational and robotic design examples	Search over supplied graph grammars, module libraries, or parameters	No; cross-layer design is the focus
Reality-assisted evolution [26]	Coupled virtual–physical evolutionary loop	Models, controllers, and especially soft or evolvable bodies	Real experiments correct or enrich virtual search	Candidate and experiment selection	No; repeated reality feedback is the focus
Evolutionary robot design [1, 2]	Candidate, population, or lineage	Morphology and controller within an encoded phenotype space	Ranges from simulation to fabricated generations	Variation and selection within the representation	No
Continual and autonomous robot learning [5, 6, 36]	Deployed robot, fleet, or learned model	Data, representation, reward, skill, or policy	Often uses real-robot experience; hardware can remain fixed	Data collection, labelling, practice, or update choices	No; retained task capability is usually the endpoint
Multi-level evolution and adaptive morphogenesis [23, 24]	Cross-scale embodied system	Materials, components, morphology, sensing, actuation, and behaviour	Constitutive: embodiment and environment produce capability	Evolutionary exploration or designed adaptation	No; research-process inheritance is not the organizing question
Robot Scientist lineage [7, 8, 10, 40]	Hypothesis–experiment–knowledge cycle	Scientific hypotheses and formal knowledge	Laboratory robots execute biological experiments	Hypothesis and experiment selection within a formalized domain	No robot-engineering update is required
Self-driving-laboratory reviews and perspectives [28, 30, 31, 34]	Autonomous experimental platform or network	Molecule, material, process, experiment, and learned model	Constitutive: digital decisions are coupled to laboratory automation	Experiment selection, workflow orchestration, and analysis	Usually no; scale, provenance, access, and discovery are central
SDL performance and safety perspectives [32, 33]	Experimental platform and its governance boundary	Precision, throughput, algorithm, human intervention, and safeguards	Physical hazards and measurement quality are explicit	Bounded experiment selection under controls	No; provides evaluation requirements that RobotRSI inherits
LLM scientific-agent surveys [17–21]	Agent capability, architecture, workflow, or scientific domain	Literature, hypotheses, plans, code, analyses, and reports	Physical execution appears in some domains but is not the common unit	Planning, tool use, experiment design, interpretation, and critique	Evolution or reflection may appear; measured downstream R&D benefit is not generally required
AI recursive self-improvement [35, 41, 42]	AI system and its improvement loop	Outputs, policies, evaluators, code, tools, or research process	Usually computational; embodiment is not constitutive	Self-modification, evaluation, and research direction under varying closure	Yes in stronger formulations, but not specifically for a robot-engineering recipient
Robotics software architecture self-adaptation [27]	Managed robot, managing system, and runtime feedback loop	Software architecture, components, task logic, and parameters	Mapped cases include real, simulated, ambiguous, and conceptual systems	Runtime monitoring, analysis, planning, and reconfiguration within supplied objectives	No; mission adaptation and later research benefit are distinct endpoints
RobotRSI	Versioned robot R&D system and transition between research episodes	Hardware, software, integration, lifecycle, and the R&D mechanism itself	Graded from digital evidence to fabrication, bench test, field update, and successor inheritance	Coded activity by activity, with human authority and resource dependencies	**Yes for a recursive-benefit claim; no for inclusion in the survey**

Table 4: Executed open-index discovery snapshot (8 September 2026). Counts describe retrieved records and screening workload, not eligible studies or field prevalence.

Operation	Fixed setting	Count	Interpretive boundary
Crossref discovery	42 object/context queries; relevance-ranked; at most 50 records per query	2,099 returned	Broad top-k candidate discovery; reported database hit totals are not used because the bibliographic search is intentionally permissive.
arXiv discovery	Same 42 query families; all-term query; relevance-ranked; at most 50 records per query	596 returned	Open preprint discovery; zero or low yield for a query is not evidence that an object family lacks research.
Candidate reconciliation	Normalized DOI when present, otherwise normalized title	2,695 $\rightarrow$ 2,423	Removes exact DOI/title duplicates only; conference–journal families and substantive extensions still require manual reconciliation.
Automated workload triage	Title contains both a robot/embodiment expression and an engineering-development expression	525 queued	Prioritization only. It may omit eligible indirect titles and retain ordinary task-autonomy papers.
Targeted source inspections	Eight earlier revision batches using primary or authoritative sources	71 records	Claim-bounded revision checks; not 71 distinct studies and not uniform full-text extraction.
Manual candidate batches	High-priority strict-title candidates; title/abstract decisions plus selected primary/full-text checks	29 decisions: 25 retained, 2 excluded, 2 deferred	Two purposeful batches, not a random sample; 496 strict-title candidates and indirect-title candidates remain undecided.
Current completion boundary	Manual eligibility, publication-family reconciliation, full-text coding, and independent checking	Open	No PRISMA-style included-study count, corpus frequency, or exhaustiveness claim is made from this snapshot.

implemented generator should neither be merged into one undated capability nor counted as two independent demonstrations of the same automated flow.

The coding scheme distinguishes the paper, system, version, research activity, experiment, artifact, and supported claim. For each study-level comparison, the minimum extract is the changed object, decision mechanism, feedback signal, evaluated endpoint, realization boundary, and human-supplied choices. More detailed transition coding adds the source and recipient versions, integration action, later use, resource budget, and evidence locator. An absent transition is not supplied by connecting two otherwise compatible papers.

Source inspection and uncertainty. Publisher metadata and an abstract support a bounded description of a method, but not a claim that its full implementation was audited. Records distinguish such checks from selected full-text inspection and completed method/appendix coding. The current supplement is a set of revision checks, not uniformly completed coding of the bibliography. Unknown implementation details remain unresolved.

We distinguish not reported, explicitly not tested, not applicable, unavailable, and conflicting evidence. In particular, a physical test after computational search differs from physical feedback used during search. A generated hardware design differs from a fabricated device; a fast kernel differs from an improved robot mission. Tables preserve those endpoints rather than assigning every work a single autonomy or maturity score.

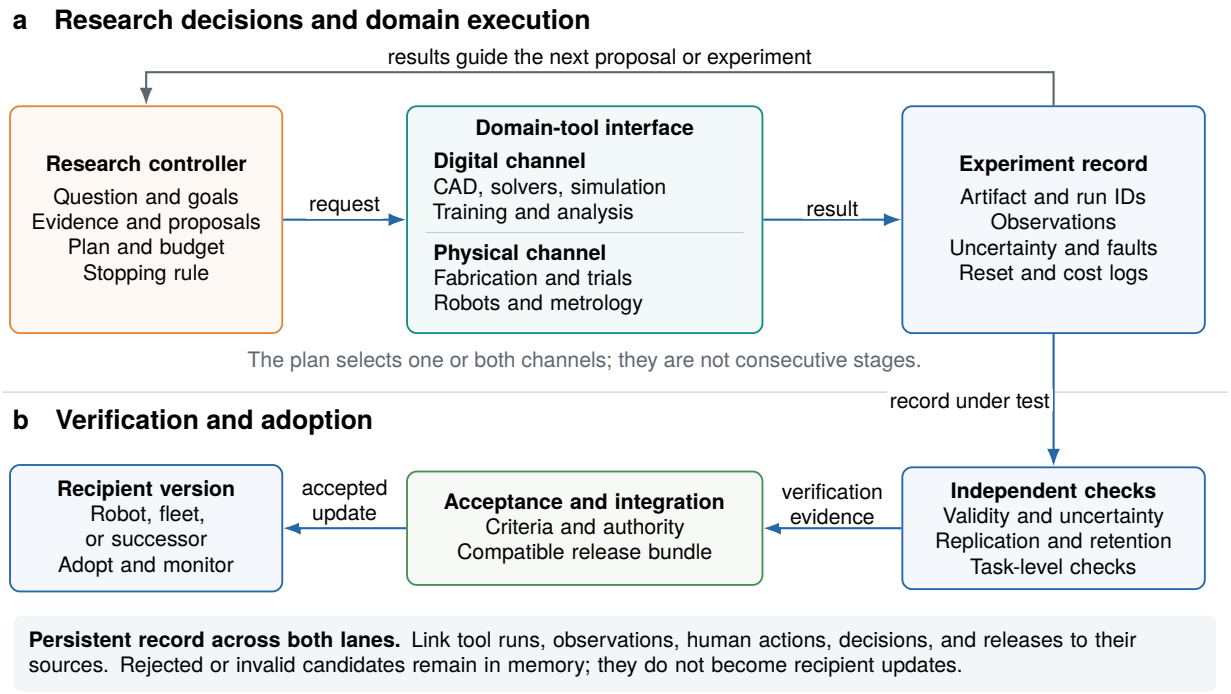

Figure 4: Proposed reference architecture for autonomous robotic R&D. The upper lane separates research decisions from digital and physical execution. An experiment record can guide another research iteration before its candidate is accepted. The lower lane separates recorded results from verification and adoption: only an accepted, compatible change set reaches the recipient. Arrow labels identify the transmitted artifact, except for the explicitly labelled proposal-revision feedback. The persistent-record strip applies to both lanes and is not another sequential stage. The diagram specifies functional roles; it does not assert an implementation that automates all of them.

Synthesis limits. Methods are compared within engineering-object families and then across research arrangements. We do not pool heterogeneous task scores, wall-clock times, and hardware budgets into one effect estimate. Claims about recursive benefit additionally require an identified later-research use and an appropriate comparison, as developed in Sections 8 and 9.

Complete screening, full-corpus coding, and independent evidence checking remain open in this working manuscript. All present extraction records were prepared by the same writing agent; no inter-rater agreement or independent replication is claimed. Consequently, bibliography totals and the number of inspected records are production counts, not evidence of exhaustive coverage. A review-flow figure or corpus-frequency map will require the corresponding completed retrieval and screening records.

3 Common Mechanisms of Autonomous Robotic R&D

3.1 Comparing research systems through artifact hand-offs

The commonality among a morphology optimizer, a program synthesizer, a controller-design agent, and a robotic test facility is not their use of the same model. Each turns a specified representation and a source of feedback into an engineering decision. Their differences concern what is already supplied, what the machine may change, and which observations determine the next change. This section compares those mechanisms before examining their engineering objects in Sections 4–6.

Figure 4 provides a reference architecture for locating these decisions. Its research-and-execution lane produces candidates and observations; its verification-and-adoption lane determines whether a candidate reaches a recipient. A paper can contribute to one hand-off without implementing the whole architecture. A solver that constructs a controller, a fuzzer that finds a counterexample, and a robot that measures a specimen therefore receive different descriptions, even if each runs without intervention after initialization.

The activities in Table 5 are an analytical vocabulary, not a requirement that every system execute ten

Table 5: Activities, artifacts, and characteristic failure modes in autonomous robotic R&D.

Activity	**Consequential machine decision**	**Persistent artifact**	**Failure or hidden dependence**
Problem formulation	Select bottleneck, objective, scope, and constraints	Requirement and research question	Easy proxy, omitted coupling, or human-chosen real question
Knowledge organization	Select and reconcile prior evidence	Evidence graph, assumptions, design rules	Citation error, version confusion, invalid domain transfer
Candidate proposal	Choose model, program, material, geometry, experiment, or process	Executable candidate and rationale	Invalid representation, infeasible design, novelty without utility
Experiment selection	Choose next query and allocate resources	Plan, budget, stopping and abort rules	Unsafe test, incomparable budget, inaccessible measurement
Digital execution	Configure and run solver, simulation, training, build, or analysis	Logs, checkpoints, digital result	Tool misuse, numerical failure, simulator exploitation, nondeterminism
Physical execution	Prepare, reset, intervene, and measure	Data, specimen/robot state, execution trace	Hidden reset, calibration drift, wear, damage, changing environment
Interpretation	Relate observation to hypotheses and subsystem causes	Finding, diagnosis, revised belief	Confounding, post-hoc account, evaluator-generator coupling
Verification	Select baseline, replicate, test robustness and regression	Acceptance report and uncertainty	Self-evaluation bias, cherry-picked condition, no held-out task
Integration	Decide whether and how to update the recipient	Versioned update package and rollback state	Interface breakage, lost capability, no deployment, unsafe propagation
Research memory	Select reusable evidence, failure, and provenance	Curated memory and lineage	Error accumulation, leakage, lost negative results, incompatible versions

consecutive stages. Likewise, the research controller need not be an LLM: evolutionary search, Bayesian optimization, constraint solving, differentiable optimization, and language-agent orchestration make different kinds of decisions. The domain executor supplies the admissible computation or intervention. Separating them helps identify whether a contribution improves proposal generation, experiment selection, execution, interpretation, or integration.

3.2 Problem formulation and research direction

Research systems enter an engineering problem at different points. In parameter optimization, the objective and search space can already be fixed. In program synthesis, examples and a language define which implementations are acceptable. In agentic design, a task description may require decomposition into representations and tool calls before optimization is possible. These entry conditions determine the meaning of autonomy: selecting a solution within a supplied problem is not the same decision as identifying which problem should be solved.

ControlAgent illustrates requirements-conditioned design rather than open-ended problem discovery [43]. The user supplies a plant model and stability, settling-time, and phase-margin requirements. Task-specific agents use encoded control expertise, while a Python executor evaluates proposed controllers and returns numerical feedback. Its ControlEval benchmark contains 500 computational tasks, with feasible requirements curated by experts. This design exposes the benefit of combining language-mediated revision with domain computation, but it does not test discovery of the plant, mission objective, or permissible physical intervention.

Cross-disciplinary design introduces another layer of delegation. The mechatronics framework of Wang et al. distributes a vessel-design objective among mechanical, simulation, electronics, and embedded-software roles [44]. Section 6 examines its resulting artifacts and human dependencies. Such decomposition makes the scope of a proposal broader than tuning one controller; it does not establish that the resulting subsystem requirements are jointly satisfiable.

For comparison, we record the supplied mission, model, variables, constraints, examples, and evaluator separately from the decisions made by the system. Cross-stack research direction can then be stated as a specific additional capability: choosing whether evidence warrants a change to sensing, control, embodiment, an experiment, or the requirements themselves. A grasp failure alone does not identify that choice; different interventions may explain or compensate for the same observation.

3.3 Representations, hypotheses, and candidate construction

A representation determines both the inventions available to a researcher and the errors its tools can reject. Parameterized structures permit economical numerical search but exclude changes outside the parameterization. Assembly grammars encode constructive restrictions before evaluation, as in RoboGrammar [45]. Differentiable shape and control representations expose local derivatives, as in DiffAqua [46]. A freely generated program has a different burden: it may express a wider change, but executability and intended semantics must still be checked.

Constraint-based synthesis demonstrates how domain knowledge can reduce that burden without a language model. LDIPS constructs action-selection programs from demonstrations, combining parameter solving, feature enumeration, and predicate synthesis [38]. Types and physical dimensions prune inadmissible expressions; bounded search and supplied primitives define the available program space. Its guarantee concerns consistency with the encoded examples and constraints, not arbitrary physical correctness. Figure 3 uses its robot-soccer repair experiment to show how changed artifact, research activity, and tested context are coded separately.

These mechanisms differ in where engineering knowledge enters. A grammar excludes constructions, dimensional constraints exclude expressions, a differentiable model supplies sensitivity, and an agent prompt guides proposals. None dominates independently of the problem. Stronger structure can make search and repair tractable while excluding useful candidates; weaker structure increases expressive freedom and places more work on validation.

Hypothesis generation also differs from candidate generation. A controller parameter is a proposed intervention, whereas an explanation of why the robot failed predicts observations under alternative tests. A search process may improve performance without resolving that explanation. We consequently distinguish the artifact proposed, the explanation offered, and the experiment capable of discriminating it from alternatives.

3.4 Experiment selection, budget allocation, and orchestration

Experiment selection allocates resources between obtaining information and improving a candidate. Bayesian physical design, multi-fidelity selection, and nested body–controller optimization make this trade-off in different units. A printed specimen, a simulator query, and a complete controller-training run cannot be compared by counting each as one experiment. Tables 6 and 7 identify the feedback and supplied structure behind representative decisions.

Budget allocation can change which robot appears promising. Arza et al. compare single-phase and two-phase body–controller search in four simulation environments [47]. Their study uses 60 runs per setting and bootstrap confidence bands. Reducing training per body and subsequently retraining the selected body's controller improves the reported mean over both single-phase comparators in all four environments, although the difference is not statistically significant in gym-rem 2D. The choice between retraining only at the end and retraining every new incumbent is budget- and environment-dependent; there is no uniformly superior schedule. The same study associates lower per-design training with simpler designs, measured through controllable degrees of freedom.

The implication is methodological: early performance can measure ease of training as well as eventual design quality. A researcher selecting bodies from partially trained controllers should report both the screening procedure and the final training used to assess the selected body. Increasing the number of designs explored is not automatically a fair or effective use of a fixed budget.

Physical orchestration adds state that a query interface does not erase. A failed grasp can leave a different scene, a test can damage its specimen, and calibration or battery state can change during a campaign. MEDAL++ makes task and undo behaviour part of autonomous practice [5]; physical facilities must similarly distinguish a completed trial from readiness for the next trial. Figure 8 develops this state-management requirement. Tool completion, experiment validity, and facility availability are different outcomes.

An orchestration record therefore needs the artifact and configuration, actual start/stop condition, observations, recovery action, and consumed resources. The appropriate stopping decision can be a successful design, exhausted budget, unresolved model discrepancy, or a request for assistance. These outcomes should not be silently reduced to the fraction of successful tool calls.

3.5 Computational and physical experimentation

The studies employ several evidence modalities: analysis, digital experiments, component tests, complete-robot tests, and operational observations. They answer different questions rather than forming a universal quality ladder. A simulation can expose an infeasible candidate economically, while a poorly controlled field trial may not identify why a change worked. What matters is whether the experiment supports the particular design, transfer, or deployment claim.

BEAR selects printed structures from measured mechanical behaviour, whereas FAST combines gripper simulation and physical grasp measurements in a multi-fidelity design process [48, 49]. The former supports specimen-level design decisions; the latter additionally couples a predictive model to a component's task-relevant response. Their physical interfaces impose different preparation, testing, and recovery costs. Neither a successful specimen test nor a calibrated surrogate, by itself, establishes improvement of an integrated recipient robot.

Table 6: Optimization mechanisms compared by representation, decision signal, and experimental interface. These are complementary mechanisms, not entries in a shared performance ranking. Physical claims refer to the inspected source scope; program and agentic mechanisms are compared separately in Table 7.

Tool / example	Representation	Next-decision signal	Experiment interface	Main comparison issue
Grammar search: RoboGrammar [45]	Graphs generated by assembly rules	Learned value of partial designs	Terrain-based computational design/control evaluation	Feasibility depends on the grammar; encoded fabricability is distinct from fabrication
Nested co-design: Evolution Gym; budget study [47, 50]	Structure and learned controller	Task return after inner-loop controller training	Shared simulation tasks; budget-allocation comparisons	Screening effort and final retraining both affect the selected body
Differentiable co-design: DiffAqua [46]	Shape interpolation and control parameters	Derivatives of simulated objectives	Virtual soft swimmers; no physical swimmers in the reported study	Local minima, parameter scaling, and hydrodynamic approximation
Learned latent dynamics [51]	Task-sensitive reduced state representation	Experience gathered during task optimization	Particle–grid soft-body simulation	Representation learning changes the information available to optimization
Physical experiment selection: BEAR [48]	Parameterized printed structures	Bayesian selection from measured mechanical properties	Automatic printing and mechanical testing	Specimen-level evidence; robot integration is a separate experiment
Multi-fidelity design: FAST [49]	Gripper geometry and a discrepancy-aware surrogate	Selection of simulation or physical test	Finite-element analysis and robotic grasp tests	Account for preparation, mounting, and the cost of both fidelities

Digital–physical consistency concerns the variables that survive realization: geometry, material and process, sensing and actuator response, timing, contact, and environmental conditions. Practical evolvable hardware illustrates how build and connection constraints restrict the candidate space [22]. DrEureka illustrates a complementary software route, where reward-informed randomization changes the conditions used for training before physical transfer [52]. These are different interventions on the relationship between model and robot, not interchangeable forms of validation.

A physical platform named in an experiment does not identify who selected, prepared, or judged the trial. In the cases below, a program may be repaired automatically from supplied demonstrations, or firmware may be generated automatically and measured by a person. Recording those interfaces preserves the demonstrated development contribution without converting a partially automated workflow into an unattended laboratory.

Table 7: Program-development and agentic mechanisms: supplied structure, machine decisions, and the evidence interface. Roles can be shared with people; neither a generated artifact nor a reported physical measurement implies an unattended end-to-end workflow.

Method	Supplied structure	Automated revision	Feedback / evidence	Interpretation boundary
LDIPS [38]	Action primitives, operators, demonstrations, bounded language	Dimensional pruning, feature/predicate synthesis, parameter repair	Example constraints; simulated tasks and physical soccer transfer	Program development, not hardware redesign or autonomous demonstration collection
SoRTSketch [53]	Finite-state sketch and parameter priors	Bayesian update of transition parameters	Corrected interaction traces; physical study with ten users	User actions supply corrections; missed corrections can degrade the update
TARL [54]	ROS/Python code, monitor, allowed constant mutations	Taint-based localization and RL-guided patch selection	Utility under nominal and modified simulated terrain	Small simulated feasibility case, not demonstrated physical ROS repair
ControlAgent [43]	Plant, design requirements, control expertise	Agent-directed controller revision with Python evaluation	Model-based response criteria; ControlEval	Computational controller design; plant identification and physical deployment are separate
Mechatronics agents [44]	Vessel objective, available electronics, expert feedback	Cross-domain proposals and firmware generation	CAD/solver outputs and physical logic-analyzer traces	Expert-assisted simulation and validation; no integrated navigation benchmark in inspected results

3.6 Interpretation, verification, and research memory

Verification criteria follow the representation and intended claim. A constraint solver can test whether a program matches examples; a controller evaluator can test numerical stability and response criteria; a mechanical experiment can test the manufactured specimen. These checks are valuable but not equivalent. Trace consistency does not establish correct intent, and a controller that satisfies a supplied model may still fail on the actual robot.

Program repair makes the distinction concrete. TARL localizes changed utility along instrumented code and searches mutations; SoRTSketch revises transition parameters from corrected interaction traces [53, 54]. Their diagnostics and patch criteria are therefore different. Section 5 compares the simulated ROS feasibility study with the physical social-robot interaction study, including the feedback people supply. A common comparison should identify the fault model, modifiable program region, observed repair target, and retained behaviour, not merely whether the output compiles.

Memory likewise has several mechanisms. ControlAgent stores controller parameters and measured performance rather than an entire conversation [43]. SoRTSketch's Bayesian variant carries a parameter posterior between interactions rather than re-optimizing against an indefinitely growing collection of traces [53]. An evolutionary archive, a posterior, and a retrieval-based evidence store preserve different information. More memory is not uniformly better: old evidence can be incompatible with a changed robot or environment, while discarding it can erase regressions.

For the proposed RobotRSI architecture, experiment provenance is a separate requirement from whichever memory the search algorithm uses. The lineage record should connect an accepted artifact to its measurements, conditions, human actions, and rejected alternatives. It supports later inspection without implying that every recorded observation was available to the learner. This separation is particularly important when protected evaluation data must not leak into proposals.

Finally, scientific reporting is not robot acceptance. AI-scientist systems automate substantial computational research and manuscript production, while the later submission study also documents human filtering and validation [15, 16]. Writing a credible report, passing an experimental criterion, and deploying a compatible robot update are three distinct outputs.

3.7 Humans, AI scientists, and robot scientists

Human involvement changes the information available to research. Demonstrations can specify behaviour that an objective omits; interaction corrections can disambiguate a transition; an expert can supply solver boundary conditions or identify an infeasible design. These contributions should be compared by their content, timing, and frequency, rather than described collectively as minimal supervision.

The mechatronics case shows why task-level accounting matters: structured human assistance was needed for turbulent-flow setup and physical signal verification, even though other design and software operations were generated by agents [44]. In interactive program repair, user correction is itself the learning signal [53]. Removing such involvement without replacing its informational function changes the method being evaluated.

AI scientists and robot scientists are complementary functional roles, not exclusive model classes. Computational agents can coordinate physical instruments, and a robot scientist can use conventional search or statistical inference. The experimental robot can also be distinct from the robot whose body, controller, or firmware is being developed. The consequential questions are who selects the intervention, who executes and validates it, which recipient changes, and what remains supplied from outside the stated system.

3.8 Section synthesis

Three comparisons organize autonomous robotic R&D. First, supplied structure determines the search space: parameters, grammars, program sketches, physical models, and expert instructions admit different revisions. Second, feedback determines the next decision: derivatives, task return, measurement uncertainty, constraint consistency, trace corrections, and solver diagnostics carry different information. Third, the experimental

Table 8: Reporting human and external dependence.

Stage	Examples of human or external input	Required report
Scope	Mission, values, forbidden modifications, target metrics	Who defines and may change each constraint
Initialization	Demonstrations, CAD, templates, datasets, priors	Quantity, provenance, expertise, and reuse across tasks
Candidate filtering	Feasibility, aesthetics, novelty, or relevance	Number before/after filtering and rationale
Experiment	Setup, reset, calibration, repair, safety supervision	Frequency, time, trigger, and effect on data
Analysis	Label correction, result selection, statistical choices	Machine proposal versus human acceptance or modification
Integration	Review, manufacturing, assembly, deployment approval	Exact artifact and recipient; validation and rollback owner
Infrastructure	Cloud models, databases, solvers, machine shops	Availability, version, cost, and boundary status

interface determines cost and validity: computation, demonstrations, components, and physical robots require different preparation and checks.

The resulting research mechanisms already automate more than policy-weight updates. They construct representations and programs, allocate design evaluations, revise models or parameters, test software, and connect disciplinary tools. The remaining compositional problem is to preserve these decisions and their evidence across engineering interfaces. Sections 4–6 therefore examine both the local method and the artifact it can actually hand to the next development activity.

4 Autonomous Research on Robot Hardware and Embodiment

Hardware makes the strongest version of the RobotRSI thesis both meaningful and difficult. A policy can be copied into memory; a new body must be geometrically valid, manufacturable, assembled, calibrated, powered, and shown not to violate safety margins. Hardware changes also alter the data distribution and the meaning of software interfaces. A longer limb changes inertia and actuator demand; a compliant finger changes contact, sensing, and the useful controller; a higher-resolution camera changes power, bandwidth, compute, latency, and thermal load. Autonomous research on embodiment is therefore not a collection of independent optimizers. It is a constrained, multi-fidelity process that has to preserve a system-level contract.

The literature already supplies important pieces of this process. Evolutionary robotics established automatic morphology generation and physical descendants; robot co-design connects form and behaviour; autonomous laboratories close experimental loops over materials; and recent engineering agents invoke CAD, finite-element, topology, motor, or electronic-design tools. The crucial survey question is not whether each piece can be called "AI for design." It is where the loop begins and ends: what is allowed to change, which decisions are autonomous, which physical activities are external, what evidence validates the artifact, and whether a robot version actually receives it.

4.1 The hardware search space is a system contract

Let a hardware candidate be

$$h = (m, p, g, k, a, s, e, c, n), \tag{1}$$

where m denotes material choices, p manufacturing processes, g geometry and morphology, k mechanisms and transmissions, a actuation, s sensing, e energy and thermal design, c onboard computation/electronics, and n communication and physical interfaces. A controller, estimator, or learned policy π is generally required

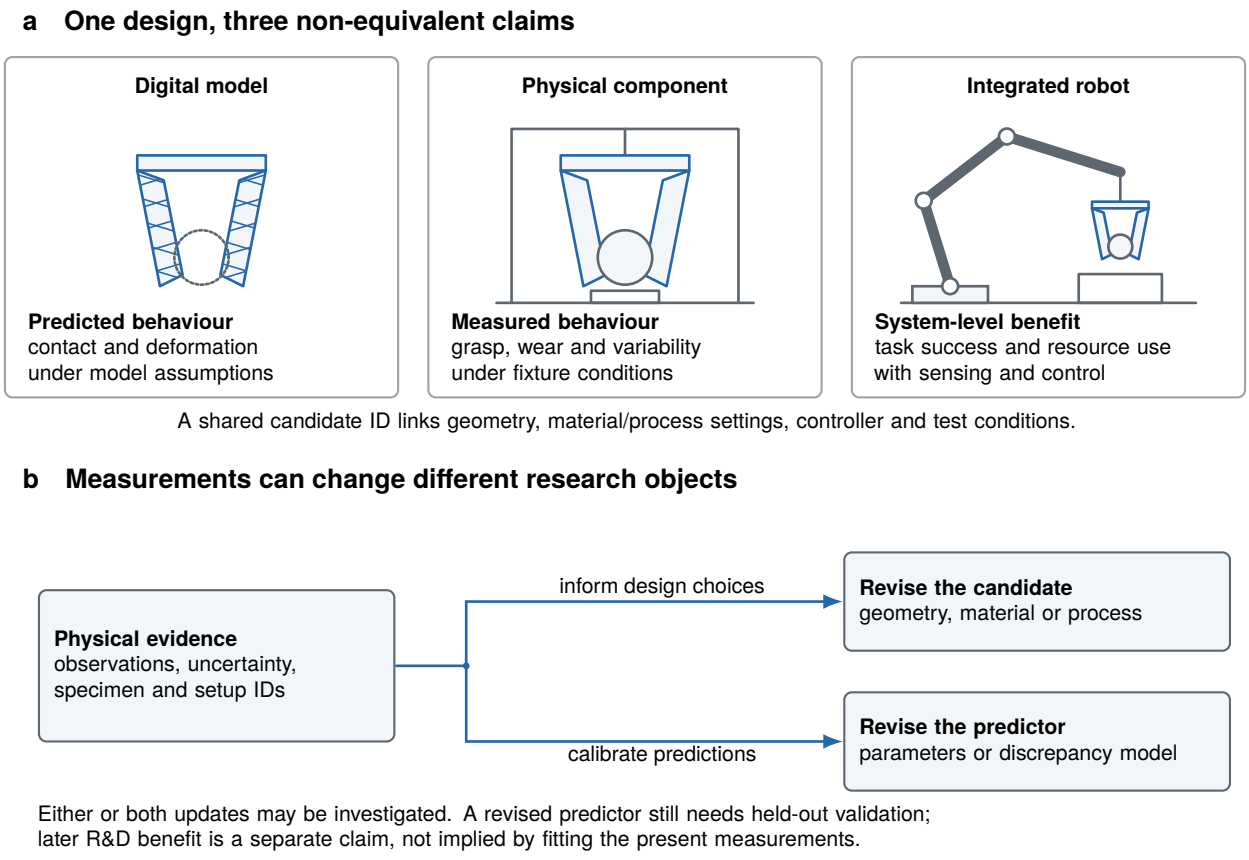

Figure 5: Hardware research must distinguish a predicted design, a tested component, and an improved robot. (a) The same illustrative gripper is shown in a numerical model, a test fixture, and an integrated robot. These are parallel evidence views, not an automatic fabrication pipeline: each claim requires its own realization and evaluation. (b) Physical observations can inform candidate changes, predictive-model changes, or both. The blue arrows denote evidence use, not demonstrated gains or recursive improvement. Drawings are original conceptual schematics, not measured results; no unattended manufacture, integration, or validation is asserted.

even to evaluate h. Hardware research is thus more faithfully written as constrained co-design

$$\underset{h,\pi}{\text{maximize}}\ \mathbf{J}(h,\pi;\mathcal{T},\mathcal{E}) \quad \text{subject to} \quad \mathbf{g}_{\text{safe}} \le 0,\ \mathbf{g}_{\text{make}} \le 0,\ \mathbf{g}_{\text{resource}} \le 0, \tag{2}$$

where the vector objective $\mathbf{J}$ may include task capability, reliability, mass, energy, cost, maintainability, and uncertainty over tasks $\mathcal{T}$ and environments $\mathcal{E}$. This formulation exposes why a scalar task reward is an unsafe acceptance criterion. A high-performing simulated morphology can be impossible to manufacture, overheat its motor, obscure its sensors, or consume the payload margin.

The admissible set is itself a research artifact. It contains available materials and components, fabrication tolerances, connector standards, service access, tool reachability, calibration procedures, and certification rules. Work on practical evolvable hardware and manufacturability-aware initialization shows that physical infrastructure changes which designs can be explored [22, 55]. Consequently, a RobotRSI system must be able to distinguish three outcomes: an intrinsically poor design, a promising design outside the current production capability, and a design whose apparent failure is caused by an invalid test or model. Collapsing all three to a low reward teaches the wrong engineering lesson.

Figure 5 separates three claims that a hardware workflow can otherwise conflate. A numerical prediction concerns a design under model assumptions; a component measurement concerns a realized specimen and fixture; an integrated robot trial also depends on sensing, control, and system resources. The shared candidate identity permits comparison but does not make these endpoints interchangeable. Measurements may motivate a new physical design or a better predictive model. Neither update is automatically validated by the data

that produced it. This distinction places data, simulation, and experimentation within hardware research: they support investigation of the engineering objects rather than form a separate competing layer. Execution, acceptance, and release are treated in Figures 4 and 11.

4.2 Materials, processes, and embodied properties

Robot materials are functional parts of computation and interaction. Stiffness, damping, friction, conductivity, permeability, optical response, fatigue, self-healing, and biocompatibility may determine what the robot can sense or do. The research variables span composition, microstructure, geometry, process parameters, curing or heat treatment, and environmental history. The appropriate evaluator may therefore be a tensile test, cyclic fatigue rig, impedance measurement, contact experiment, or complete task—not merely a database-predicted property.

Autonomous materials laboratories demonstrate that physical hypothesis–experiment–measurement loops are technically feasible. ARES combined an experimental planner, automated reactor, in-situ Raman measurement, and iterative model updates for carbon-nanotube growth; its report also makes the human boundary explicit: people selected the domain and feedback signal, initialized the data, supplied objectives, and periodically changed them [56]. CAMEO coupled physics-informed active learning to synchrotron characterization and retained a human expert inside parts of the loop [57]. AtomAgents illustrates a complementary computational pattern in which multiple agents coordinate retrieval, physics calculations, and alloy analysis [58]. These are strong precedents for an *AI scientist for robot materials*, but they are transferable evidence until a robot-specific requirement drives the campaign and a resulting material/process is fabricated, qualified, and integrated into a robot.

FAST moves closer to that connection. It uses a multi-fidelity Bayesian "scientist" to choose soft-gripper candidates for finite-element simulation or physical evaluation, automatically returns grasp data, and uses those results in later design decisions [49]. Across the reported experiments, the platform collected 50,608 simulated grasps and 8,532 physical grasps, and used physical data to characterize and reduce simulation discrepancy. This is unusually direct evidence of adaptive physical experimentation on a robot component. Its boundary remains instructive: batches of designs were printed, post-processed, and mounted before automatic testing. FAST therefore closes candidate selection, simulation, physical test, and model-update edges more strongly than it closes unattended fabrication or installation into a successor robot.

For RobotRSI, the next connection is a requirements-to-material pipeline. A field robot might identify seal degradation, infer whether the causal lever is geometry, material, or maintenance, commission accelerated ageing tests, and propose a qualified replacement. Acceptance would require not only the target property but process repeatability, environmental compatibility, supply constraints, and regression tests on the recipient. The research output is a package—material specification, process recipe, uncertainty, provenance, and allowable operating envelope—rather than a material name.

4.3 Morphology, mechanisms, and fabrication

Morphology research reveals three choices that shape an autonomous engineering method: the representation of allowable bodies, the procedure for finding a controller for each body, and the source of evidence used to select a design. These choices separate several methodological families that are often grouped under "robot evolution." The soft-robot design review by Pinskier and Howard makes the resulting bottleneck especially visible: a rich material–morphology–control space cannot be explored autonomously without compatible simulators, manufacturing throughput, and tests [59].

Population search and constructible design spaces. Early body–controller evolution connected virtual candidates to manufactured electromechanical forms [1]. Mother Robot moved candidate construction and evaluation into a physical modular assembly process [2]; later Bayesian experiments compared joint morphology–control search with control-only optimization [60]. Their defining design choice is a physically executable repertoire. A module library limits structural novelty but makes candidate construction and

repeated evaluation tractable. A restricted representation is simultaneously a search bias and an experimental interface. The relevant comparison is how much useful design variation can be realized and evaluated for a given facility budget.

RoboGrammar supplies a computational alternative: a graph grammar describes admissible assemblies, and a learned heuristic predicts the promise of partially expanded designs to guide search [45]. The resulting robots and controllers are evaluated on terrain tasks. This separates structural feasibility encoded in the grammar from performance learned during search; encoding fabricable arrangements does not establish that those candidates have been fabricated. More generally, representation quality concerns excluded designs as well as achieved reward. An optimizer cannot discover an actuator arrangement that its grammar cannot express.

Nested design and controller learning. Evolution Gym standardizes voxel bodies and simulated tasks, with baseline methods that place design search outside a controller-learning loop [50]. The comparison includes genetic search, Bayesian optimization, and CPPN-NEAT, paired with PPO for control. This exposes an evaluation issue: a body's score depends on how well its controller has been trained. Our interpretation is that a design that learns quickly can outrank a more capable body under a short training allowance. RobotRSI comparisons should therefore account for controller-training effort as well as candidate count.

Differentiable co-design. Gradient methods exploit derivatives of simulated task performance. DiffAqua parameterizes soft-swimmer shapes and actuation through interpolation between base designs, then optimizes geometry and control jointly [46]. Its ablations identify sensitivity to gradient scaling and initialization, and its limitations reserve physical fabrication for future work. This is computational co-design in an underwater context. Differentiable methods support efficient local improvement when their models and parameterizations are informative; discrete changes and model discrepancy remain separate problems. Matthews et al. complement this family with rapid automatic body design and physical tests of selected bodies [4].

Quality-diversity methods change the question from finding one maximum to retaining a repertoire of differently structured solutions. Zardini et al. compare MAP-Elites, viability evolution with NEAT, and a double-map method that maintains separate morphology and controller descriptors for soft tensegrity modular robots [61]. Both tasks are evaluated in simulation. The resulting archive can support later selection under preferences not represented by a single scalar score, but neither archive diversity nor encoded modularity establishes fabrication or recipient integration.

Learning the representation and generating proposals. Learning-in-the-loop optimization updates a latent description of soft-body dynamics as task optimization proceeds [51]. The state representation becomes a learned object rather than a fixed input. Language-guided approaches such as RoboMoRe instead broaden how candidate descriptions and objectives enter a co-design process [62]. These approaches operate at different interfaces: latent learning changes the representation of experience, whereas language-based proposals change how designs or objectives are specified. An integrated researcher could use both, but their individual demonstrations do not establish that combination.

Across these families, the trade-off is between expressiveness, search cost, and fidelity. Graph rules encode discrete assembly knowledge; voxel encodings admit many topologies but require repeated control learning; differentiable representations support local search but inherit simulator assumptions; physical construction tests real interactions at higher experimental cost. Table 9 separates the reported artifacts and physical evidence. The comparison concerns these interfaces, rather than an assumed progression from evolution to language agents.

Mechanism research adds discrete topology, kinematic singularities, collision, tolerance stack-up, wear, lubrication, cable routing, and serviceability. General mechanical-design agents can translate requirements into parameterized structures and invoke finite-element tools [63]; topology-optimization agents aim to orchestrate geometry, meshing, boundary conditions, solving, and repair [64]. Their value to RobotRSI is not that language replaces mechanics, but that it can coordinate typed tools and preserve the rationale between

requirements, solver inputs, failures, and revisions. Solver success alone is insufficient: boundary conditions can be wrong, an optimized mesh may be unmanufacturable, and a minimum-mass part may have unacceptable fatigue or impact behaviour.

Implemented robot-specific workflows span different parts of this chain. Pastor et al. join genetic optimization of a manipulator's kinematic structure to iterative automated CAD in a five-degree-of-freedom case [65]; the inspected publication record supports a digital design workflow, not physical construction. RobotSmith instead proposes task-specific tool geometries with collaborating vision–language agents, generates use trajectories, and jointly refines geometry and use in simulation [66]. Selected tools are 3D printed and executed by an XArm7 in a multi-stage pancake task. That physical demonstration validates a designed tool–plan pair, while printing, mounting, and acceptance remain outside the reported autonomous pipeline; the paper's 50.0% average success figure belongs to its simulation comparison rather than the physical demonstration.

Fabrication is therefore part of the experiment, not a printer icon between CAD and testing. A complete record includes machine/material batch, orientation, support strategy, toolpath, post-processing, measured dimensions, rejected builds, assembly operations, and calibration. The digital candidate and physical specimen need linked identities. If a human silently reworks, selects, or mounts the successful specimen, that intervention is part of the evidence and cost. A future self-modifying robot may use a distributed workshop rather than carry manufacturing onboard; this is still a valid RobotRSI architecture if the workshop, human roles, logistics, and acceptance authority are inside the declared system boundary.

BEAR offers a useful contrast with simulator-led morphology research. It combines Bayesian experiment selection with automatic additive manufacture and mechanical testing of structures [48]. Here, fabrication and measurement generate the observations on which the next design decision depends. The reported objects are mechanical specimens, so the system contributes a transferable experimental method rather than evidence of a robot receiving a new body. The connection matters for robot structures and protective components, where manufacturing variation is part of the property being optimized.

4.4 Actuation, transmission, and compliant dynamics

Actuation research changes both a force-producing device and the operating conditions under which it is useful. Motor winding, transmission ratio, elastic elements, pneumatic chambers, valves, and drivers expose different search variables. Peak force alone is therefore an incomplete objective: a design can improve displacement while consuming more energy, loading a bearing differently, or reducing its continuous operating envelope.

Search geometry together with its actuation. Raeisinezhad et al. compare firefly search and deep reinforcement learning for pneumatic chamber design [67]. Analytical and finite-element environments support different geometric freedoms. Selected silicone actuators are then fabricated and measured using digital image correlation. The measured objective concerns the balance between horizontal and vertical displacement, not simply maximum motion. Some designs remain computational only. This makes the work a design-search study with subsequent physical validation, rather than an unattended fabrication-and-test loop.

Kaczmarski et al. use a reduced-order active-filament model for a complementary problem: choose fiber-based actuator designs with low actuation energy while reaching a specified target [68]. An inner activation search supplies the cost used by outer Bayesian design optimization. Evaluation is computational and assumes quasi-static motion; the comparison with random search concerns the tested target-reaching problem. The method illustrates how a cheaper mechanistic representation can expand design exploration, while its assumptions determine which dynamics the search can discover.

Both approaches expose a consequential choice: whether to spend computation on a richer mechanics model, a more expressive design space, or more candidate evaluations. A fair comparison needs the same task constraint and a reported controller/activation budget. Otherwise, a body can appear superior because more effort was spent finding how to actuate it.

Select components under mission-derived loads. Liu et al. combine human architectural decisions with numerical design for a heavy-duty quadruped [69]. Their motor–gear selection minimizes component mass subject to torque, speed, and thermal constraints derived from task demands; the candidate catalogue is assembled with human input. A physical prototype provides system-level testing. This is a useful hybrid engineering case: computational selection sits inside a broader human-directed development process, rather than replacing it.

At a different interface, a multi-agent motor-design system couples finite-element evaluation with a surrogate to search motor parameters [70]. It expands computational design orchestration, whereas catalogue selection chooses among already available devices. These should not be assigned the same manufacturing boundary. Changing a motor design can also require a new inverter or cooling arrangement; changing a catalogue choice may preserve manufacturing but still alter integration and control.

The resulting evaluation sequence runs from model identification to component tests and then integrated duty cycles. Its purpose is to locate the source of an apparent gain: geometry, control effort, component selection, or a changed operating limit. Overload and fatigue also modify specimens during testing, so experimental history is part of the observation rather than an interchangeable repetition.

4.5 Sensing, electronics, onboard compute, and communication

The observation channel begins before model training. Modality, placement, optics, sampling, synchronization, packaging, and calibration determine the signal available to learning. Observability-driven sensor placement on a soft continuum arm evaluates reconstruction consequences on physical hardware [71]. Task-and-sensor co-learning instead chooses sparse measurements jointly with learned representations for tactile sensing and proprioception [72]. The first relies on the specified model's information structure; the second relies on the training tasks and data. Their common comparison should hold sensor budget and recipient conditions explicit.

CODEI connects sensor choice and placement to the perception pipeline, planning, computation, body, cost, mass, and energy in a mobile-robot design case [73]. This moves the objective beyond isolated reconstruction error: a sensor that improves estimation can still be a poor system choice if its mass or compute demand compromises the mission. Physical installation and calibration remain different operations from selecting a configuration.

From a robot model to a compute design. Robomorphic Computing identifies a direct relation between robot morphology and accelerator architecture [74]. Link/joint structure determines useful parallelism and matrix sparsity. The paper evaluates an FPGA implementation and synthesized ASIC results, but explicitly states that its proof-of-concept design flow was performed manually and leaves automation to future work. We therefore use it as a design foundation, not as an implemented autonomous generator.

RoboShape subsequently supplies an executable generation framework [75]. It parses robot descriptions, derives topology-based schedules and matrix operations, and generates Verilog within computing-resource constraints. Its evaluation includes FPGA accelerators for iiwa, HyQ, and Baxter models. Computation-only timing and host-transfer timing are separate comparisons. These are hardware-computation tests for robot workloads, not evidence that all three physical robots autonomously replaced their onboard computers.

AutoSoC expands the coupling from a robot-derived accelerator to an algorithm–accelerator workflow for aerial obstacle avoidance [76]. Its Air Learning component trains and validates candidate end-to-end policies, while FlexACL generates accelerator candidates and executes synthesis and place-and-route. Six reported candidates expose performance, area, and power choices. The endpoint is a chip layout that could be taped out: the study does not report tape-out, physical chip measurement, or installation on an aircraft. This is consequential digital co-design across O05, O08, and O12, with the production and recipient edges still open.

This progression is important for RobotRSI: a domain insight becomes a repeatable engineering transformation when the representation, generation procedure, and validation interface are implemented. It need not

be expressed as a conversational agent. A robot-body change could trigger regeneration of a compatible accelerator, but deciding whether that change is worthwhile and validating the full control stack require additional research decisions.

Tool orchestration and interface scope. ChatEDA operates at another level, decomposing requirements and invoking tools through a digital implementation flow [77]. In contrast to RoboShape's specialized mapping, it orchestrates a broader tool sequence. The two approaches can be complementary: an agent can choose and configure a domain generator, while deterministic checks establish properties that a language response alone cannot.

Communication changes similarly span software and physical interfaces. Protocol configuration, task allocation, transceiver choice, and link placement are distinct research objects. The examples above establish neither autonomous radio design nor field qualification of a communication system. For distributed robotics, subsequent studies must connect packet loss, delay, and resource use to estimator and control outcomes. This is an outstanding coverage obligation, not evidence that communication research lacks automated methods.

4.6 Energy, thermal design, packaging, and lifecycle load

Energy research includes at least three different intervention levels: changing a material or device, optimizing how an existing device is operated, and configuring the complete robot's resource allocation. Conflating them obscures both autonomy and benefit. A charging policy can improve a cell's useful life without changing its chemistry; a more conductive electrolyte need not improve a robot battery pack; a lower-power accelerator can change feasible control timing rather than merely extend endurance.

Experimental duration and charging-protocol search. Attia et al. combine early prediction of battery cycle life with Bayesian selection of charging protocols [78]. Short cycling experiments provide the feedback used to select subsequent tests, followed by a separate validation campaign. The early predictor itself relies on prior cells cycled to failure. Thus, reduced campaign duration depends on a previously acquired measurement model and its training cost; it is not cost-free access to lifetime labels. This is physical optimization of an operating protocol, transferable to robot energy systems but not a robot-level deployment study.

Composition search and downstream validation. Clio couples robotic liquid handling and metrology to the Dragonfly experiment planner [79]. Conductivity guides search over electrolyte mixtures; selected solutions undergo pouch-cell tests outside Clio. The preparation of feeder solutions and the separate cell-construction procedure delimit the automated portion. This separates three artifacts: an experimentally selected composition, its measured transport properties, and a battery's downstream behaviour. For RobotRSI, the corresponding hand-off would additionally require a compatible pack and a specified robot duty cycle.

The contrast between these studies is methodological. Early outcome prediction shortens each trial, whereas automated preparation and measurement change how many candidates can be tested with a given apparatus. Either can improve experimental throughput, but their costs and errors enter different parts of a campaign. A survey should report which bottleneck is reduced instead of treating every speedup as a common measure of research autonomy.

Thermal geometry and coupled constraints. Chen et al. integrate parametric modelling, meshing, numerical evaluation, and Bayesian search for a wavy-fin heat sink [80]. The objective combines heat transfer and pressure-drop effects; the reported optimization is computational. This supplies a transferable thermal-design workflow, not robot cooling validation. It complements the task-derived thermal limits used in motor–gear selection: one changes the cooling structure, while the other constrains component selection under a load model.

At system level, CODEI exposes coupled resource choices [73]. These computational comparisons need operating histories to support endurance or lifetime claims. Battery age, ambient conditions, transient loads, cooling auxiliaries, and charging logistics can change the preferred design. The unresolved step is not simply

to attach an optimizer to a thermal solver, but to connect that solver's assumptions and predictions to the recipient's measured mission profile.

Table 10 compares these component and resource pathways, including whether evidence comes from numerical analysis, a fabricated device, an instrumented campaign, or an integrated robot. It also keeps foundational manual workflows distinct from implemented automation.

Table 9: Selected body-design and materials-research methods. The comparison separates the search representation from physical realization; computational feasibility, fabrication, and integrated robot benefit are different endpoints. Material/specimen studies provide transferable methods when no robot recipient is demonstrated.

Study	Research object	Search / experiment mechanism	Evaluated endpoint	Boundary
Lipson and Pollack [1]	Body and controller	Computational evolution; manufacture of selected forms	Fabricated electromechanical bodies	Post-design realization is distinct from designer improvement
Mother Robot [2]	Modular child morphology	Robotic assembly and physical evaluation	Constructed and tested generations	Supplied modules and bounded arena
Rosendo et al. [60]	Morphology–control parameters	Bayesian search over physical construction and trials	Joint-design versus control-only comparison	Fixed parameterization and facility
Manufacturable seeds [55]	Feasible design population	Diversity-aware initialization for evolution	Search convergence and task performance	Encoded manufacturability is not an observed fabrication campaign
RoboGrammar [45]	Graph-structured body	Grammar expansion with learned heuristic search	Simulated terrain-task performance	Grammar constrains the reachable designs
Evolution Gym [50]	Voxel body and controller	Outer design search; inner policy learning	Simulated benchmark comparisons	Body ranking depends on control-training effort
DiffAqua [46]	Soft-swimmer shape and actuation	Differentiable co-design	Computational task performance and ablations	Fabrication deferred; simulator and initialization dependence
Zardini et al. [61]	Soft tensegrity morphology and controller	Quality-diversity co-evolution with separate morphology/controller maps	Simulated goal-reaching and passage-squeezing repertoires	Archive diversity; no fabricated recipient in the inspected evidence
Matthews et al. [4]	Locomotor geometry	Gradient-based automatic design	Physical tests of selected bodies	Physical validation follows computational search
Learning in the loop [51]	Soft-body dynamics representation	Latent learning during task optimization	Optimization with an updated representation	Representation learning is not arbitrary body fabrication
RoboMoRe [62]	Morphology and control objectives	Language-guided proposal and co-design	Reported co-design tasks	Proposal breadth does not establish unattended realization
BEAR [48]	Mechanical specimens	Bayesian selection, printing, mechanical testing	Physical structural experiments	Transferable specimen laboratory; no robot-body recipient
FAST [49]	Soft-gripper geometry and simulator	Adaptive simulation/physical selection; automated grasp trials	Printed gripper tests and discrepancy modelling	External printing, post-processing, and mounting
RobotSmith [66]	Task-specific tool geometry and use plan	VLM proposals; simulated joint geometry/trajectory optimization	Selected 3D-printed tools executed by an XArm7	Printing/mounting external; aggregate success is a simulation result
ARES / CAMEO [56, 57]	Material synthesis / characterization	Active experiment selection and instrument feedback	Physical materials campaigns	Transferable methods; human goals or expert involvement

Table 10: Component, compute, energy, and manufacturing-process research. A denotes robot-related development, B a transferable engineering method, and C a design foundation. Roles are not autonomy levels. Physical realization and automated scope are recorded separately.

Study / role	Changed object	Decision or transformation	Evaluated artifact / endpoint	Boundary
Raeisinezhad et al. [67] (A)	Pneumatic chamber geometry	Firefly / reinforcement-learning search in analytical or FE environments	Selected fabricated actuators; measured displacement balance	Physical validation follows search; some designs simulation-only
Kaczmarski et al. [68] (A)	Fiber actuator design	Outer Bayesian search; inner activation optimization	Computed energy under target-reaching constraint	Quasi-static model; no physical design campaign
Liu et al. [69] (A)	Motor–gear choice	Catalogue optimization under task and thermal constraints	Selected components and tested quadruped prototype	Human architecture and catalogue preparation
Motor-design agents [70] (B)	Motor parameters	Agent orchestration, finite-element evaluation, surrogate search	Computational motor design	Manufacture and robot integration not established
Sensor optimization [71, 72] (A)	Sensor placement and representation	Observability analysis or task-and-sensor co-learning	Reconstruction / task evidence on soft systems	Model- or task-dependent measurement choice
CODEI [73] (A)	Sensing, compute, body, decision pipeline	Constrained task-driven co-design	Computational mobile-robot configurations	Configuration selection is not physical installation
Robomorphic Computing [74] (C)	Morphology-specific accelerator	Manually realized proof-of-concept design method	FPGA computation; synthesized ASIC results	Automation explicitly deferred in the paper
RoboShape [75] (A)	Accelerator implementation	Robot topology and resource limits generate schedules and Verilog	FPGA workload tests for three robot models	Kernel / host-I/O timing, not robot field deployment
AutoSoC [76] (A)	Policy and aerial-robot accelerator layout	Policy training/validation plus synthesis and place-and-route	Six digital accelerator candidates with area/power/performance trade-offs	No tape-out, chip measurement, or aircraft integration
ChatEDA [77] (B)	Digital implementation flow	Language-model tool orchestration	EDA workflow benchmarks	Fabricated robot electronics not established
Attia et al. [78] (B)	Charging protocol	Early lifetime prediction guides Bayesian experiment selection	Cell cycling and separate validation campaign	Prior predictor-training cost; no robot recipient
Clio [79] (B)	Electrolyte composition	Robotic measurement and Bayesian search	Conductivity campaign; later pouch-cell tests	Stock preparation and cell testing outside automated loop
Chen et al. [80] (B)	Heat-sink geometry	Automated modelling, meshing, evaluation, Bayesian search	Numerical thermal / pressure-drop objective	Computational result; robot packaging untested

Continued on the next page

Table 10: Component and resource research (continued).

Study / role	Changed object	Decision or transformation	Evaluated artifact / endpoint	Boundary
AM ARES [81] (B)	Printing parameters	Image-scored Bayesian process optimization	Repeated physical single-layer print trials	User-defined target, bounds, and stopping; not robot assembly

4.7 From a promising component to an accepted embodiment

Hardware evidence should be read as a ladder of claims rather than a maturity score. Digital analysis can show that a candidate satisfies a model. A manufactured specimen can show geometric and process feasibility. A component test can establish a property under a protocol. An integrated robot can expose interface and emergent effects. Field trials can reveal environment and maintenance constraints. Higher rungs do not erase lower-rung uncertainty, and a poorly controlled field demonstration is not automatically stronger than a replicated bench test.

For each candidate, the acceptance package should include: (i) the requirement and causal hypothesis; (ii) immutable and modified variables; (iii) CAD, bill of materials, software/firmware interfaces, and manufacturing record; (iv) calibration and metrology; (v) model predictions and physical residuals; (vi) component, integration, and regression results with uncertainty; (vii) resource and human-intervention accounting; and (viii) recipient version, rollback plan, and post-deployment monitoring. Safety-critical changes additionally require independent hazard analysis and an authority that is not rewarded solely for accepting its own proposal.

This package distinguishes four levels of closure. *Design closure* produces an executable specification. *physical closure* fabricates and measures the artifact. *system closure* installs it in a robot and passes acceptance. *recursive closure* demonstrates that the updated robot or infrastructure performs later R&D more effectively under a controlled comparison. Most existing work contributes strongly to one or two levels; combining them is a system-integration agenda rather than a missing optimization algorithm.

4.8 Section synthesis

Autonomous hardware research is already plausible in pieces and partially demonstrated in physical loops. The strongest evidence includes automatic physical phenotype development and adaptive soft-robot component testing; autonomous materials laboratories and agentic engineering provide transferable mechanisms for other hardware domains. The main gaps are typed hand-offs, unattended physical preparation, cross-domain verification, and recipient-level acceptance. RobotRSI should therefore treat materials, fabrication, morphology, mechanisms, actuation, sensing, energy, electronics, compute, communication, and packaging as coupled research objects, while treating data, simulation, experimentation, and memory as mechanisms that run across them. The next section applies the same activity-and-artifact analysis to intelligence and software, where updates are easier to deploy but no less dependent on embodied evidence.

5 Autonomous Research on Robot Intelligence and Software

Software research can change either a robot's behaviour or the procedure by which that behaviour is developed. The distinction is visible in the experiments: tuning a controller evaluates candidate gains; designing a reward evaluates policies trained with candidate rewards; generating simulation tasks evaluates executable environments and the learners they support. These are different levels of automation, not interchangeable instances of policy learning. They also predate language-model agents: probabilistic modelling, experimental design, automated reinforcement learning, and continual adaptation supply important foundations.

This chapter follows objects O06–O13. Data, simulation, and experimentation appear within their respective engineering loops rather than as a parallel application domain. A symptom alone does not identify the intervention: the same task failure can arise from observation, behaviour, objective, or runtime defects. We therefore compare what each method changes, the feedback used to choose the next step, its physical realization, and the development choices that remain supplied. Later research benefit is considered separately in Section 8.

5.1 Perception, representation, and state estimation

Perception research has at least three intervention points: the observations and supervision, the learned representation, and the estimator that consumes it. A calibration change can correct systematic geometry without

retraining a network; new labels can address a semantic error without changing the sensor; a fusion or uncertainty model can change how unreliable observations affect decisions. These interventions require different evidence. Framewise prediction accuracy, pose error, and task completion should not be treated as equivalent objectives.

Blum et al. connect online semantic learning to indoor localization on real robots [36]. Their system combines self-supervision with continual updates during deployment and evaluates memory replay to reduce forgetting across environments. The important empirical connection is not simply a better segmentation model: the updated perception also benefits localization in 3D floorplans. This is a deployment-grounded alternative to selecting a model solely on a fixed offline dataset. Its learning construction remains supplied, while the observations and model state change with use.

A complementary intervention is to automate development around the model. LABELING COPILOT addresses computer-vision data curation, whereas AutoML-Agent organizes planning, implementation, and verification across a computational learning pipeline [82, 83]. These studies concern different artifacts from on-robot continual learning. They supply transferable methods for choosing data operations and model experiments, but their cited evaluations do not establish downstream robot localization or manipulation.

The resulting research question is how to connect those choices to an embodied error diagnosis. For example, comparing relabelling, recollection from a new viewpoint, and a representation update requires a shared downstream test and separate accounting of annotation and robot time. Temporally adjacent observations should remain grouped during evaluation; otherwise a curation method can appear effective through near-duplicate train–test exposure. These are requirements of the proposed transfer, not additional capabilities attributed to the computational agents.

5.2 Self-models, world models, and interaction models

A learned model can serve two distinct purposes: predicting a robot's operational consequences and selecting informative experiments about the robot. Continuous self-modelling by Bongard et al. uses physical interaction to infer body structure and recover behaviour after damage [3]. Egocentric visual self-modelling instead learns a predictive representation from onboard observations, with physical adaptation on a 12-degree-of-freedom platform and additional simulation-specific evaluations [39]. Both connect observations to a revised model and behaviour, but they differ in representation and sensing assumptions; they should not be collapsed into one universal body-identification algorithm.

This distinction also prevents an ambiguity in the word "self." The egocentric system autonomously updates a model of the robot and uses detected anomalies to adapt behaviour, which is a consequential same-platform development operation. It does not restore a damaged component, expand the permissible model class, or show that the identification procedure itself becomes better at later research. Operational resilience is therefore relevant evidence below the recursive-R&D threshold, not an exception to that threshold.

PILCO provides a complementary model-based policy-search route [84]. A Gaussian process predicts dynamics, model uncertainty enters approximate long-horizon policy evaluation, and analytic gradients improve the policy. New interaction then updates the dynamics data. The physical cart-pole experiment reports 17.5 seconds of system interaction before successful swing-up and balancing; the unicycle study is simulated. The interaction time is a task-specific data-efficiency result, not a complete accounting of computation, setup, or autonomous R&D labour.

The location of the probabilistic model matters. PILCO models state transitions; the controller-tuning methods below can instead model a scalar experimental cost. The first representation supports counterfactual rollouts but must propagate modelling error through time. The second avoids modelling the full plant but obtains information only through evaluated controller settings. Neither representation is intrinsically a more autonomous researcher: they expose different experiment choices and assumptions.

For research use, predictive accuracy is an intermediate endpoint. An updated contact model might alter design rankings, while an updated body model might change the excitation selected for diagnosis. The informative comparison tests that downstream choice, including conditions outside the model's fitting trajectories. Persistent model state can therefore enter a recursive feedback route even when the outer learning code is

unchanged; the downstream benefit still needs its own evidence.

5.3 Control and motion generation

Direct experimental tuning. Marco et al. combine a nominal linear-quadratic regulator (LQR) design with Entropy Search over experimentally measured cost [85]. Tunable LQR design weights parameterize controllers, while the performance objective remains fixed. A Gaussian-process belief guides trials toward information about the optimum, demonstrated on a pole-balancing robot arm. The nominal model structures the search; it is not re-identified during tuning. Failed trials can be interrupted, so nominal LQR construction should not be mistaken for a guarantee that every physical experiment is stable.

Widmer et al. make trial safety part of Bayesian controller optimization for legged locomotion [86]. Their evaluation includes Unitree Go1 hardware across gait settings. Compared with unconstrained informative exploration, restricting trial selection to a probably safe region changes which uncertainty can be resolved and how far the search can move from its starting controller. The controller family, parameterization, and objective are still specified. This boundary is a useful design choice for scarce hardware, not a reason to disregard the autonomous experimental decisions.

Rai et al. use simulation differently: rather than transfer a controller directly, they learn a domain-informed feature transform that structures Bayesian controller optimization [87]. Experiments include physical ATRIAS trials and simulated bipeds under deliberately increasing model mismatch. This makes simulation a prior over which controllers are similar, while measured hardware cost remains the optimization endpoint. The result supports sample-efficient tuning within expert-specified controller families; it does not automate objective selection or body design.

These approaches trade representational breadth for testability. A low-dimensional controller family gives experiments a clear identity and makes comparisons repeatable, but cannot discover a missing feedback structure outside that family. Model-based policy search opens a different parameterization but introduces model and policy-class assumptions. The comparison should therefore report the search envelope alongside trial count and performance, rather than rank methods by physical evaluations alone.

Agentic controller construction. ControlAgent uses domain expertise and computational tools within an LLM-agent workflow to solve control-design tasks [43]. It can address construction and refinement decisions upstream of a supplied parameter search. Its computational evaluation, however, is a different experimental object from a legged robot operating under contacts, delays, and actuator limits. A generated controller's simulated response does not establish compatibility with a robot's estimator, discretization, or runtime.

This suggests a compositional development route: an agent proposes a controller family or implementation, numerical tools check and tune it, and a bounded experimental method evaluates the physical candidate. Such composition is a perspective, not an end-to-end system established by joining results from separate papers. Section 6 examines the integration interfaces.

5.4 Planning, skills, and interaction

Generating executable behaviour. Code as Policies translates language instructions into programs that compose supplied perception and control APIs [88]. Hierarchical code generation defines missing helper functions and supports reactive or waypoint-based behaviour on real robots. This changes the executable task program without necessarily retraining perception or low-level control. Its recursive function generation is a property of program construction, not evidence that successive robot R&D systems improve themselves.

This is an important boundary case for the survey. Generating a program to satisfy the current instruction is task automation. Retaining, comparing, and validating candidate program abstractions for later tasks would add a development loop. The latter requires evidence about reuse, preconditions, and failures, rather than simply a longer generated program.

Synthesizing and repairing action selection. Executable policies can also be synthesized and repaired within a constrained language. In LDIPS, physical soccer tests report attacker success of 50% before parameter repair and 64% after it; deflector success changes from 52% to 72% [38]. Repair uses ten physical demonstration runs. These descriptive results concern supplied skills and program-parameter changes, not new hardware or autonomous collection of repair demonstrations.

RADIUM couples failure discovery more directly to repair [89]. Approximate Bayesian inference searches for likely high-dimensional failure conditions, with gradients from differentiable simulation or a gradient-free alternative, and the same framework adjusts design or control variables to reduce predicted failure. Selected repairs transfer to hardware. In the reported race-car comparison, the nominal and repaired policies fail in 25% and 5% of 20 trials, respectively; a Robotarium experiment supplies a second physical setting. Because simulation produces the search and repair feedback, these tests establish bounded transfer of a repaired candidate, not autonomous physical diagnosis, a new failure specification, or full-stack robot redesign.

RoboScribe widens the editable representation from parameters to a control program [90]. Starting from raw, unsegmented demonstrations, abstraction refinement derives state predicates and abstract actions, then iteratively synthesizes task programs with recurring subroutines. Its reported long-horizon benchmark results concern supplied demonstrations and simulated robot environments. The method removes a predefined-abstraction requirement, but it does not choose the task, collect its own demonstrations, or establish physical deployment. RADIUM and RoboScribe therefore automate different research decisions: one searches a failure distribution and repairs within selected variables; the other changes the symbolic interface through which behaviour is synthesized.

Selecting experience to develop skills. Active selection long predates foundation-model orchestration. SAGG-RIAC chooses parameterized goals from competence progress while learning inverse models on an arm, a quadruped, and an arm manipulating a flexible fishing rod [91]. Hangl et al. instead use autonomous play to compose sensing and preparatory behaviours around a human-taught basic skill [92]. Their physical manipulation trials provide success statistics, while the reported roughly 30% acceleration from active selection is estimated in simulated convergence experiments driven by those statistics. Both systems choose useful experience inside a supplied task/skill representation; neither autonomously invents the development objective.

SOAR uses foundation models to organize and evaluate autonomous robot experience and then improve an instruction-following policy [6]. It reports 25,000 real trajectories across five tabletop environments. Here the planning machinery helps determine what the robot practices and how its experience is assessed; the recipient of training is a policy, not a newly invented planning algorithm.

AutoRT extends experience orchestration to mobile-manipulator fleets in varied office settings [93]. Scene descriptions support task proposals, feasibility filtering, and selection among scripted, learned, and teleoperated execution. Its collected episodes are consequently a mixture, not all autonomous demonstrations. Besides diversity analyses, the paper evaluates RT-1 fine-tuning on two manipulation tasks. Those learning comparisons are more directly relevant to development than diversity scores alone, but do not establish a general improvement across the entire proposed task distribution.

The contrast is between composing an action program and organizing an experience campaign. Both use task semantics, but the latter also determines the distribution on which later learning depends. Broad language instructions do not by themselves establish broad robot competence. A survey of general-purpose development must preserve the distinction between proposed tasks, feasible tasks, executed episodes, and capabilities validated on the recipient.

5.5 Data, learning algorithms, rewards, and curricula

Searching the learning procedure. AutoRL demonstrates that upstream learning design need not depend on an LLM [94]. Its navigation pipeline first searches reward coefficients using task completion and then searches actor–critic layer widths with the reward fixed. Policies are trained in simulation and evaluated

both virtually and on a mobile robot. The architecture search varies layer widths at fixed depth; it is not arbitrary program synthesis. Each outer candidate entails policy training, unlike controller tuning in which an evaluation can be a physical rollout of an existing controller.

Constructing and refining rewards. Eureka opens the reward program itself [95]. Its evolutionary loop generates executable code, trains policies, and feeds reward-component statistics and task scores back into subsequent proposals. This exposes richer revision information than a single final score. Its main evaluation is simulated; human input is also supported. The learned policy is the task artifact, whereas the reward program is an upstream development artifact.

Text2Reward makes an interactive route explicit [96]. A compact environment abstraction grounds dense reward code, execution feedback helps correct code errors, and users can critique rollout videos to revise rewards. Physical Franka manipulation experiments demonstrate transfer in bounded tasks. Compared with automated training statistics, human feedback can specify an intended behaviour that the initial task description left ambiguous. It also remains a human research contribution, even when no manual reward programming is needed.

DrEureka further couples reward construction to domain-randomization design [52]. A reward-trained policy is tested across individual simulation-parameter values to obtain a reward-aware physics prior; those ranges ground LLM proposals for randomization distributions. The reported transfer includes quadruped locomotion and dexterous manipulation. This differs from simply widening every parameter range: what is useful for training depends on the task and policy. Physical transfer evaluates that combined choice, rather than validating reward text in isolation.

These methods optimize learning outcomes through intermediate artifacts. Reward quality is conditional on the learner and training budget, while randomization quality also depends on the physical conditions encountered later. Comparisons should separate code execution, learning progress, intended behaviour, and physical transfer. A syntactically valid reward addresses only the first of these endpoints.

Generating a curriculum or a task implementation. GoalGAN learns a generator of goals at an appropriate level of difficulty from policy success feedback [97]. Its simulated goal-reaching experiments vary a parameterized subset of the state space while preserving the goal semantics and sparse success test. It automates the distribution of practice, rather than writing a new environment implementation.

GenSim instead generates task code and retains a library used for demonstrations, policy training, and task-generator fine-tuning [98]. The generator evaluation reports improved coding pass rates on held-out tasks. The physical policy evaluation uses real-data adaptation and is a separate experiment. Human checking and language–implementation mismatches remain relevant: passing runtime and demonstration-generation checks does not ensure that the written task implements the intended instruction.

GenSim2 extends the approach toward articulated-object tasks with multimodal task proposals, planning/RL solvers, and point-cloud policy learning [99]. Human keypoint annotation and solver verification remain in the pipeline. Its eight-task physical comparison reports mean success of 36.3% with limited real data, 42.5% with simulation data, and 57.5% with combined data, using ten evaluation episodes per task. Individual tasks do not all improve. These within-study outcomes support the generated-data pipeline under its tested configuration; they do not rank it against GenSim's different tasks or establish uncertainty-free transfer.

CurricuLLM generates a sequence of subtasks and compiles each subtask into reward and goal-distribution code, while an evaluator model uses rollout trajectories to select policies [100]. The method is compared across simulated manipulation, navigation, and locomotion settings, and a learned locomotion policy is transferred to a Berkeley Humanoid. For that physical experiment, predefined safety and desirability rewards and a termination condition supplement the generated curriculum. The implemented contribution is therefore automatic curriculum construction with a bounded sim-to-real endpoint, not autonomous definition of the safety envelope.

Sustaining physical practice and adaptation. MEDAL++ addresses reward supervision and environment reset together [5]. From initial demonstrations, it learns task and undo behaviour, infers rewards, and practices on real robots without an externally imposed episodic reset after every attempt. Reported improvements over behavioural cloning are task-dependent, spanning 30–70 percentage points. The feedback loop changes policy, reward, and recovery behaviour within a supplied learning procedure. Reset competence is therefore part of the learning mechanism, not merely background laboratory support.

Never Stop Learning studies a complementary offline fine-tuning route [101]. A pretrained robotic grasping system is adapted with target-domain data under visual and morphology changes, including sequential adaptation experiments. The study also exposes an unresolved stopping problem: excessive optimization on limited data can damage performance and subsequent adaptation. This is evidence against assuming that additional training or another update must improve a robot. An autonomous researcher needs a way to select checkpoints using meaningful evidence, not only an instruction to continue training.

Continual learning is consequently an evaluation problem as well as an update mechanism. Lesort et al. organize robotic continual-learning settings around sequences of experience, resource limits, retention, and embodied evaluation, while observing that much of the field was still evaluated on simulations or static datasets [102]. Tree Learning provides a recent humanoid example of hierarchical parameter inheritance and reports complete skill retention in its specified Unity comparisons [103]. That percentage is bounded to the reported simulated tasks and baselines; without physical deployment, open-world task order, and longer-horizon regression tests, it cannot be read as a general absence of forgetting.

RoboCat feeds experience from task-specialized policies back into generalist training [104]. Its controlled self-data ablation and its comparison of later adaptation answer different questions. The latter evaluates a broader-trained generalist, so the contribution of self-generated data is not isolated from all other dataset changes. Nonetheless, later learning is actually assessed; it should not be treated as absent because the training algorithm remains fixed.

Self-Improving Embodied Foundation Models turns predicted steps-to-go into a dense reward and success detector, allowing several robots to collect experience for reinforcement-learning updates [105]. In three real LanguageTable campaigns, one person monitored three or four stations and performed exceptional or periodic resets without supplying success labels. The reported policy success increases from approximately 62–63% to 87–88% after collecting additional Block2Block experience equivalent to about 3% of the full dataset. This is strong evidence for a real fleet-mediated policy-update loop, but it is not unattended: goals and stopping conditions are supplied, the initial stage uses imitation data, and real-world ALOHA self-improvement remained incomplete.

Open X-Embodiment supplies a different foundation: shared data and models across robot embodiments [106]. Such resources can initialize these loops, but collaborative collection does not imply autonomous collection. Together, these cases distinguish data inheritance, autonomous practice, learner adaptation, and task-generator training. Their common use of experience does not make their research outcomes equivalent.

5.6 Robot software architecture and runtime

The runtime layer determines whether a proposed algorithm can become a robot artifact. Relevant research objects include message interfaces, sensor clocks, drivers, firmware, scheduling, build configurations, and deployment dependencies. Failures here can invalidate conclusions drawn about a model: a delayed observation, incompatible message, or incorrect actuator conversion can produce a task failure without a defective learning algorithm.

The architecture-based self-adaptation literature is the closest established synthesis at this layer. Alberts et al. code 37 primary studies with 32 parameters spanning the managed robot, managing system, adaptation mechanism, and evaluation strategy [27]. Their unit is a runtime response that changes architectural structure or parameters under uncertainty. RobotRSI inherits that managed/managing separation but adds a different evidential edge: a research episode must generate or select a development artifact, establish its adoption into an identified version, and—for recursion—test its effect on later R&D. A component swap that preserves the current mission can be important self-adaptation without satisfying either added edge.

RoboFuzz provides an empirical example of automation at this layer [107]. It mutates ROS messages, executes the system, and monitors developer-specified correctness oracles. Semantic feedback, including sensor inconsistency or control error, guides further input selection because ordinary code coverage can poorly distinguish robot behaviours. The framework supports simulated and physical execution modes, with target-specific evaluation across robot software stacks. This automates counterexample discovery; it does not automatically repair the reported bugs or independently invent the correctness specification.

ROSA changes runtime configuration rather than test input [108]. Its ROS 2 managing subsystem reasons over an application knowledge base to select task-and-architecture co-adaptation plans, evaluated in an underwater-robotics use case. This is a stronger deployment interface than switching a controller parameter alone: components and task logic can be reconfigured together. Yet the knowledge model, permissible actions, and goals are provided by developers. Runtime architectural adaptation is an object for autonomous research and a possible adoption mechanism, not by itself an agent that invents or improves the architecture.

The method contrasts with code-generation agents in the direction of search. A generator proposes implementations intended to satisfy a task; a fuzzer proposes inputs intended to expose invalid behaviour. They could share a development loop, but their evidence is not interchangeable. A generated test suite can miss the same requirement as its code generator, while a failure-inducing input still needs reproduction, diagnosis, and a validated patch. Simulation-discovered failures should retain their realization boundary until physical relevance is checked.

From fault localization to a bounded patch. TARL instruments ROS/Python data-flow paths and estimates their utility using a modified SARSA procedure [54]. Differences between nominal and changed-environment utility identify a candidate faulty line; constant mutations guarded by the relevant condition are then explored with an ϵ-greedy strategy. The demonstration is a small simulated shuttle task, not a physical deployment, despite its use of real ROS code. The performance monitor and mutation space are supplied. This is a concrete repair mechanism, but its single-line feasibility result does not establish general middleware repair or a new correctness specification.

Interaction-driven program repair. SoRTSketch takes a different route: the programmer supplies a finite-state sketch, and Bayesian inference updates uncertain transition parameters from corrected interaction traces [53]. A ten-participant study used a physical tablet-faced robot over four repair iterations. Users' button presses supplied state corrections; sensing and uncorrected interaction errors could lead to ineffective changes. Thus, implicit feedback here does not mean feedback-free learning. Compared with TARL's utility discrepancy, the target is agreement with corrected interaction, not recovery against the same engineering monitor. Both operate within a supplied program structure, whereas broader source-code generation also changes that structure.

Figure 6 compares these mechanisms with LDIPS at the level of representation and feedback. Its parallel panels do not imply a maturity order or a common performance scale.

For robot deployment, the release unit is consequently larger than a model checkpoint or source commit. Code, dependencies, device configuration, calibration, and model state must be compatible. This chapter identifies the development mechanisms; system-level commissioning, release, and recovery are treated in Section 6, avoiding a separate acceptance checklist for every software object.

5.7 Synthesis of software evidence

Tables 11 and 12 organize the evidence around two complementary views: learning from robot experience, and changing the tools or procedures that produce robot software. Membership can overlap. They compare mechanisms and supplied assumptions, not a scalar autonomy rank or pooled performance score.

Three distinctions recur. First, the *search representation* determines what can change: controller weights, dynamics models, reward programs, task distributions, environment code, or runtime inputs. Broader representations allow different inventions but also increase the burden of checking validity. Second, the *feedback signal* determines what the process can learn from: transition data, scalar return, component-level training

Table 11: Experience-based software development: changed artifacts, feedback, and reported robot outcomes. Supplied design choices delimit each experiment; they do not negate the autonomous decisions made within it. Task scores and interaction budgets are not pooled across studies.

Study	Changed artifact	Feedback and next decision	Reported realization / endpoint	Supplied design choices
Blum et al. [36]	Semantic model; replay memory	Deployment self-supervision updates perception	Physical indoor localization and forgetting evaluation	Learning construction; floorplan-localization setting
Bongard / Hu et al. [3, 39]	Self-model; compensatory behaviour	Physical interaction or egocentric observations revise a predictive model	Physical damage/adaptation examples; additional simulated settings in Hu et al.	Representation, sensing interface, and adaptation procedure
PILCO [84]	Dynamics model; policy	Transition data update a GP; uncertain model rollouts guide policy gradients	Physical cart-pole: 17.5 s learning interaction; unicycle simulated	Cost, policy class, and probabilistic approximation
SAGG-RIAC / autonomous play [91, 92]	Inverse model; goal/behaviour selection	Competence progress or model-guided play selects experience	Three robot setups; real manipulation statistics plus simulated selection-speed analysis	Goal/skill representation, basic behaviour, and learner
Automatic LQR tuning [85]	Controller design weights	GP over measured cost; Entropy Search selects informative trials	Physical pole-balancing arm; bounded parameter search	Nominal model, objective, search bounds, and interruption mechanism
Safe locomotion tuning [86]	Controller/gait parameters	Safe Bayesian optimization restricts and selects trials	Simulation and physical Go1 locomotion	Controller family, objective, and safety assumptions
Rai et al. [87]	Controller parameters	Simulation-learned feature transform structures Bayesian hardware trials	Physical ATRIAS and simulated biped experiments under model mismatch	Expert policy family, cost, simulator, and search bounds
MEDAL++ [5]	Task/undo policies; reward	Autonomous practice supplies learning and reset experience	Physical manipulation; task-dependent gains over initial imitation	Demonstrations, tasks, and learner
Never Stop Learning [101]	Adapted grasping policy	Offline fine-tuning with target-domain experience	Physical visual/morphology shifts and sequential adaptation; stopping failure studied	Pretraining, fine-tuning recipe, and data-collection protocol
Tree Learning [103]	Hierarchically inherited skill parameters	Root/branch reuse expands simulated humanoid skills	Unity locomotion, interaction, and navigation; no physical humanoid result	Task order, reward shaping, simulator, and skill interface
SOAR [6]	Experience data; instruction policy	Foundation models organize and evaluate autonomous practice	Physical tabletop collection and subsequent policy improvement	Base models and collection/training pipeline
AutoRT [93]	Selected experience; fine-tuned policy	Scene-grounded proposals select tasks and execution mode	Mobile-manipulator fleet; mixed autonomous/teleoperated data; bounded learning tests	Robot capabilities, constitution, and human supervision
RoboCat [104]	Generalist weights; trajectories	Adapted specialists collect data for generalist training	Self-data ablation and later adaptation are separate comparisons	Demonstrations, architecture, tasks, and training procedure
Self-improving EFM [105]	Policy, replay data, reward labels	Predicted steps-to-go supplies reward/success feedback for fleet practice	Real LanguageTable update on three/four stations; simulated ALOHA	Imitation initialization; supplied goals/stopping; one human monitors and resets

Table 12: Automation upstream of robot policies and in software development. Rows distinguish the object generated or searched from the endpoint actually evaluated. Code as Policies is a task-programming boundary case; computational engineering agents are transferable methods, not physical robot evidence.

Study	Development object	Search / revision signal	Evaluated endpoint	Boundary
AutoRL [94]	Reward coefficients; network widths	Outer search evaluates trained policies	Simulated and physical navigation	Fixed RL algorithm and network depth
Eureka [95]	Reward program	Evolutionary code generation with component-level training feedback	Simulated robot policy performance	Supplied simulator, learner, and task fitness
Text2Reward [96]	Dense reward code	Execution errors and human rollout critique	Simulation and bounded physical manipulation transfer	Human feedback; expert environment abstraction
DrEureka [52]	Reward and randomization distribution	Policy-conditioned physics ranges ground LLM proposals	Physical locomotion/manipulation transfer	Supplied simulator and permitted parameters
GoalGAN [97]	Goal distribution	Policy success guides an adversarial goal generator	Simulated goal-reaching curriculum	Fixed goal parameterization and success criterion
GenSim [98]	Task code/library; generator	Execution, reflection, and library-based training	Held-out task coding; physical policies after real-data adaptation	Semantic mismatch; human checks
GenSim2 [99]	Tasks, solvers, demonstrations	Multimodal proposals and planning/RL solvers	Generated data; physical sim-only and combined-data policy tests	Keypoint annotation and solver verification
CurricuLLM [100]	Subtask sequence, reward code, goal distribution	LLM generation; rollout-based policy evaluation	Simulated multi-domain study; humanoid policy transfer	Physical safety/desirability rewards and termination supplied
Code as Policies [88]	Executable task program	Language-conditioned composition of APIs and helper functions	Physical execution of instructed behaviours	Runtime synthesis is not a later-R&D comparison
RADIUM [89]	Failure distribution and policy/design parameters	Approximate inference identifies failures; simulator feedback drives repair	Physical race-car and Robotarium tests of selected repairs	Failure criterion, simulator, and repair variables supplied
RoboScribe [90]	State/action abstractions and task program	Demonstrations drive abstraction refinement and synthesis	Long-horizon simulated robot environments	Task and demonstrations supplied; no physical deployment established
ROSA [108]	Runtime task logic and software architecture	Knowledge-based reasoning selects a reconfiguration plan	ROS 2 underwater-robotics use case	Knowledge model, goals, and permissible actions supplied
RoboFuzz [107]	Failure-inducing test input	ROS-message mutation with semantic feedback	Correctness violations in robot software stacks	Developer-defined oracles; no automatic repair claim
ControlAgent [43]	Controller design/code	Agent/tool refinement with domain expertise	Computational control-design tasks	Physical closed-loop validity needs separate testing
AutoML-Agent [83]	Learning pipeline and model	Agent planning, implementation, verification	Computational ML tasks	No robot recipient in the cited evaluation
LABELING COPILOT [82]	Curated visual data	Agentic data discovery, synthesis, and labelling	Computer-vision curation tasks	Robot-data dependencies and downstream effects untested here

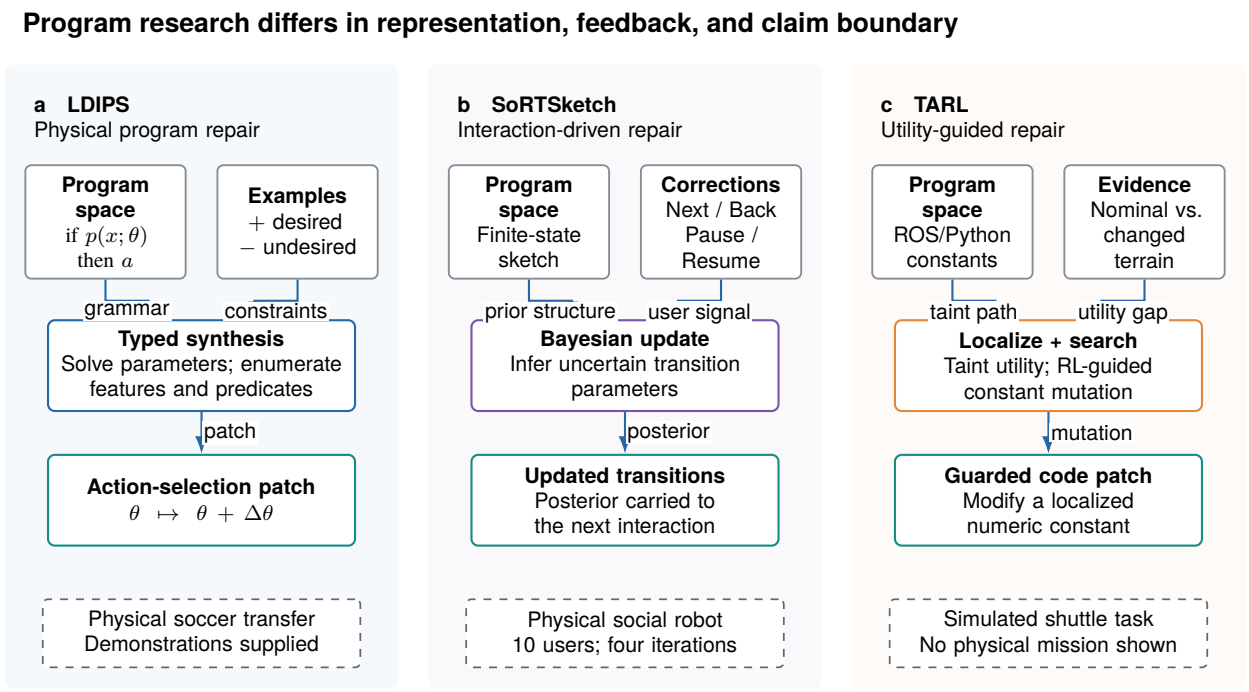

Figure 6: Three bounded mechanisms for researching robot programs. LDIPS combines a supplied language and demonstrations to synthesize or repair action selection [38]; SoRTSketch updates transition parameters from user corrections [53]; TARL localizes and mutates a numeric constant using monitored utility differences [54]. Arrows stay within each method and label the artifact or evidence handed to the next operation. The panels compare mechanism, not performance: tasks, endpoints, budgets, and realization differ. Physical evaluation in the first two panels does not make demonstration collection or user correction autonomous, while the use of ROS in the third does not make its simulated mission a physical test.

statistics, human critique, or semantic failure signatures. More elaborate feedback is useful only if it addresses the relevant error. Third, the *experimental unit* determines the budget: a rollout, a full policy-training run, a physical data campaign, and a software fuzzing campaign have different costs and failure modes.

The reviewed cases already exceed a picture of robots merely updating policy weights. They include automated controller experiments, reward and curriculum construction, task-code generation, fleet experience selection, and correctness testing. At the same time, these components do not establish a single general researcher across perception, planning, learning, and runtime. The principal integration question is whether the system can diagnose an unfamiliar failure, choose the appropriate intervention level, and validate the resulting compatible robot update. The next chapter considers that system-level change set.

6 Autonomous System-Level Robot Engineering

The system-level problem in RobotRSI is to turn separately developed artifacts into a working robot. A new gripper may demand different calibration and control; a perception model may exceed the compute or thermal budget; a lighter body may change vibration and localization; a controller may be valid only for a hardware revision no longer deployed. Sections 4 and 5 examined local research objects. This section asks how their artifacts become one operable, maintainable, and identifiable robot version.

System-level autonomy is not achieved by putting independent optimizers in parallel. It requires allocation of coupled requirements, typed hand-offs, configuration control, integrated experiments, acceptance authority, and lifecycle feedback. The central unit is a *system change set*: a versioned group of hardware, software, calibration, process, and documentation artifacts whose joint effects are evaluated on a declared recipient.

6.1 Requirements, architectures, and resource allocation

A mission statement such as "operate longer in clutter" underdetermines the engineering response. Possible levers include battery, mass, route planning, sensor duty cycle, compute scheduling, actuator efficiency, thermal limits, or task allocation. Autonomous research direction must translate the statement into measurable capability, environment, hazards, invariants, and budgets, then preserve traceability as requirements propagate to subsystems.

Requirements allocation is a composition problem. Two components may each meet a nominal latency limit while communication and scheduling violate the end-to-end deadline. Conversely, accepting a slower local computation can be appropriate if it reduces sensing, transmission, or downstream planning cost. The relevant evidence concerns the combined configuration under the mission workload, not only independent subsystem specifications.

Architecture selection determines which changes are possible. Modularity, standard interfaces, accessible test points, replaceable parts, signed software bundles, simulation models, and telemetry can reduce the cost of later R&D. They can also impose mass, complexity, attack surface, and performance penalties. An architecture for self-improvement should therefore be evaluated against a distribution of future modifications, not assumed universally superior because it is modular.

Wang et al. provide a concrete agent-assisted cross-domain example motivated by a water-monitoring vessel [44]. The reported artifacts include propeller and hull designs, reuse of supplied motor-control electronics, and embedded firmware assessed through physical logic-analyzer traces. The results section also records expert assistance for turbulent-flow configuration and further validation needs. These are useful design and component-level hand-offs. The inspected evaluation does not provide an integrated vessel-navigation benchmark from which autonomous whole-system performance or development-time savings can be established. Its lesson is to inspect the artifacts exchanged between roles, not infer integration from the agent hierarchy.

Resource allocation is part of research reasoning. Mass, volume, power, thermal capacity, bandwidth, compute, human labour, robot hours, materials, and risk are shared budgets. CODEI demonstrates a task-driven computational formulation that exposes choices across body, sensor, algorithm, compute, energy, weight, and cost [73]. RobotRSI generalizes this idea from design selection to research selection: which subsystem should be changed, at what fidelity should it be tested, and which budget is worth spending given the expected system-level information?

6.2 Cross-component co-design

Figure 7 separates reported cross-domain artifacts from the additional evidence required for a system-level claim. A mechanical design, generated firmware, simulation output, and physical signal trace can each be a legitimate intermediate result while leaving integration, commissioning, and mission performance unresolved. The system change set is therefore an evidential unit, not merely a list of files produced by different agents.

Cross-component co-design may be simultaneous, nested, or alternating. These are not obligatory successive stages. Simultaneous search exposes interaction but grows rapidly and makes attribution difficult. Nested design optimizes a controller for every body candidate, but can hide unequal inner-loop budgets. Alternating updates are practical yet can cycle or lock into an early architecture. The chosen scheme and its stopping rule are therefore part of the method, not implementation detail.

Existing work supplies useful patterns. Morphology–control co-optimization demonstrates that a controller optimized for an unsuitable body need not recover the best joint solution [60]. RoboMoRe treats morphology and reward as coupled design targets [62]. CODEI joins sensor and algorithm selection with planning and compute [73]. FAST couples physical component experiments, simulation, and surrogate calibration [49]. These systems span different evidence modalities and scopes; together they motivate cross-stack research, but they do not constitute one demonstrated universal co-designer.

Three additional cases clarify what "system" can mean. Bosio and Mueller co-optimize the attachment layout and controller of a cooperative aerial payload system, then compare optimized and suboptimal configurations on fleets of three or four quadcopters [109]. AutoSoC produces a policy and placed-and-routed

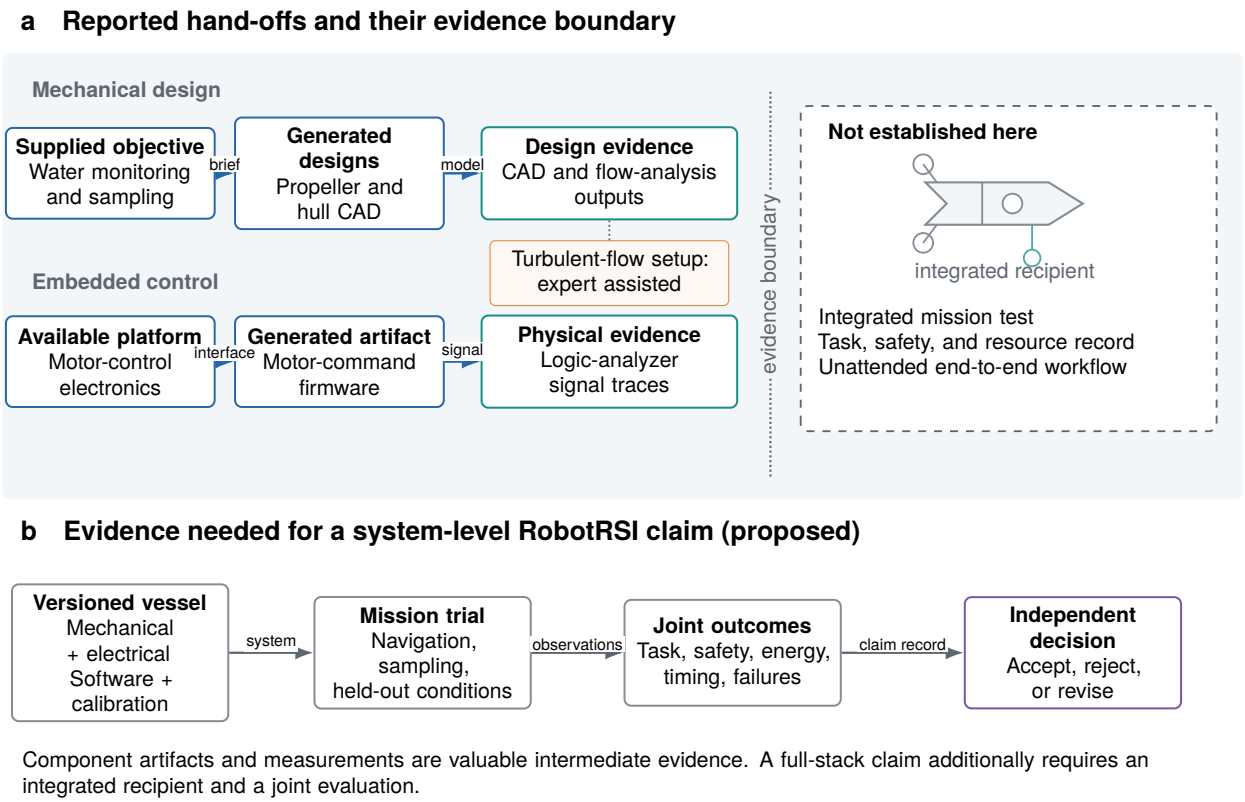

Figure 7: From cross-domain artifacts to a system-level evidence boundary. Panel (a) is a bounded reading of the agent-assisted vessel case reported by Wang et al. [44]: propeller/hull designs, reused motor-control electronics, generated firmware, computational design outputs, and physical logic-analyzer traces are reported, while turbulent-flow analysis requires expert setup. The dashed boundary is the survey's interpretation of what the inspected results do not establish; it is not a failed experiment in the source. Panel (b) is our proposed evidence chain for a stronger system-level claim. Its arrows label required transfers, not outcomes already demonstrated by that study. Multiple subsystem artifacts do not by themselves establish an integrated, unattended, or recursively self-improving robot.

accelerator candidates but stops before chip fabrication and aircraft installation [76]. RobotSmith physically combines a generated tool with a generated use plan, while the manipulator platform and production operations remain supplied [66]. These are genuine cross-component relations with different recipients and missing hand-offs; none should be inflated into autonomous whole-robot engineering.

Two further computational cases expose scale and representation boundaries. Vanteddu et al. keep CAD geometry inside the iRonCub co-design loop: a multi-objective evolutionary search changes selected links for flight performance and automated finite-element analysis rejects candidates that violate a structural margin [110]. SwarmCoDe jointly evolves morphology, planning, heterogeneity, and population composition for simulated swarms under fabrication budgets [111]. The first strengthens the hand-off from control-oriented parameters to mechanically inspectable geometry; the second changes the recipient from one body to a population design. Neither abstract establishes automated fabrication or commissioning, so both remain digital system change sets rather than realized successors.

Attribution requires disciplined comparisons. If body, policy, data, and compute all change, an end-to-end gain supports the change set but not a claim that each part improved. Factorial or staged ablations, interface-level measurements, and matched-budget baselines can locate contributions. In some settings the interaction is the result: neither a new body nor controller works alone. The evidence model must represent such non-additivity instead of forcing credit to one artifact.

Co-design also needs fidelity allocation. A planner can screen thousands of candidates with analytic or learned models, test tens in high-fidelity simulation, and fabricate a few. Physical residuals should update both candidate beliefs and the fidelity model. A different morphology may have a different reality gap, so a single global simulator correction can be misleading. The ARE framework highlights this heterogeneous

reality-gap problem for evolving populations [112].

6.3 Manufacturing, assembly, and reconfiguration

Manufacturing converts design uncertainty into physical variation. Autonomous system engineering must plan processes, schedule equipment, inspect output, handle build failure, assemble and route components, install firmware, and link measured artifacts to digital versions. A printer that runs unattended automates one operation; it does not close the chain if a person selects parts, removes supports, reworks interfaces, mounts electronics, or decides which build is acceptable.

AM ARES makes process development, rather than only part production, an explicit automated activity [81]. It adjusts printing parameters using image-based feedback against a user-defined single-layer target. Campaign execution includes allocating unused substrate locations and cleaning the dispensing tip. The user supplies the parameter bounds, objective, and termination conditions. This is a physical process-optimization loop with an engineered reset mechanism, not autonomous robot assembly. Compared with geometry search over a fixed manufacturing process, it changes the process that realizes a candidate; those two intervention levels can therefore require different validation records.

Automatic manufacture of evolved forms and Mother Robot established early links from computational search to physical descendants [1, 2]. More recent practical evolvable-robot work makes the facility constraints explicit [22]. The ARE project describes RoboFab, in which an arm retrieves components from fixed locations, uses printers for custom body parts, and assembles from a genome; the reported arm operates without vision using predetermined placements [112]. This is valuable infrastructure evidence and also a warning: fixture accuracy and predefined component banks are part of the phenotype representation. The 2024 framework presents continued software–hardware orchestration as a direction; it should not be cited as proof that unrestricted continuous physical evolution is already closed.

Physical growth and replacement versus autonomous design choice. Robot Metabolism uses extendable Truss Links with magnetic connectors to demonstrate morphology formation, damage recovery, replacement of a depleted module, and robot-assisted assembly [113]. All physical experiments are operator controlled through direct commands or preprogrammed scripts; slopes and obstacles are part of the designed apparatus. The contribution is a physical mechanism for bodily change, not an autonomous discovery loop. It complements evolutionary construction systems by changing who can perform assembly: another robot of the same module family can help realize a configuration.

The comparison suggests two distinct research questions. One concerns selecting a useful morphology; the other concerns whether available robots and equipment can realize its transitions. A constructible design may have an inaccessible assembly sequence, and a physically possible transition may be hard to discover or repeat. For RobotRSI, assembly affordances, fixtures, and recovery procedures should therefore be evaluated with the design representation. They are not incidental preparation hidden between a digital candidate and its fitness score.

6.4 Calibration, commissioning, and system validation

Calibration is an especially concrete form of robot-directed experimentation: the target is an identifiable robot model, the intervention is a pose or excitation, and the returned artifact changes how subsequent motions or measurements are interpreted. Three method families make different parts of this workflow economical.

Choose measurements using a predictive model. Daş and Burdick learn residual kinematic errors with Gaussian processes and use GP-UCB to select measurement configurations [114]. This avoids restricting correction to a revised Denavit–Hartenberg parameter vector. Comparisons against random sampling, expected improvement, and a D-optimal method use three simulated arm geometries. The lander motivation and named Barrett WAM geometry do not make these physical or space-deployment experiments. The contri-

bution is adaptive error-model learning; its sensing assumptions and workspace coverage delimit the reported correction.

Yang et al. instead select eye-in-hand calibration views by predicted reduction in parameter uncertainty [115]. The reported comparison uses synthetic data and image/pose datasets acquired with an EPSON manipulator, with held-out measurements for evaluation. The study assumes unconstrained transitions between candidate views and does not charge for path planning. Camera intrinsics are calibrated separately. Thus, informative view selection is demonstrated, but collision-aware autonomous execution and complete commissioning are additional interfaces.

Selingue et al. close more of the physical calibration loop by combining geometric identification with a Gaussian-process model of residual position error and an active-learning stopping rule [116]. Experiments on two manipulators report a 97% error reduction with fewer than 30 measurements for GPR training. The design parameters and stopping condition were tuned on an existing measured dataset, so the result supports data-efficient execution of a supplied hybrid method rather than autonomous invention of the method. Kolakowski provides a different recipient boundary: a mobile robot maps a furnished apartment and collects radio fingerprints to automate calibration of an indoor positioning system [117]. There the robot is the experimental executor, while the calibrated infrastructure and its learned radio map are the principal recipients.

These methods optimize different quantities. An error-correction model needs samples that explain residual behaviour; a parameter-estimation method needs measurements that constrain its chosen geometric variables. Neither objective automatically minimizes total calibration time. A distant informative pose can be expensive or unreachable, and a small posterior uncertainty can coexist with an incorrect model class. Comparisons should therefore separate measurement count, travel/execution cost, predictive accuracy, and downstream task performance.

Change the measurement apparatus. MUKCa takes a complementary route: a two-socket fixture constrains the end-effector, and optimization adjusts the nominal kinematic model using recorded joint configurations [118]. Users guide the robot in a back-drivable mode and trigger observations. Evaluation spans Franka, KUKA, and Kinova platforms; insertion and motion-retracing tests on a Franka examine consequences beyond calibration residuals. This reduces dependence on external metrology, not the human role in choosing and collecting configurations. Its physical fixture and parameter optimizer could support an active procedure, but that combination should not be credited to the reported implementation.

Together these cases show why calibration autonomy has several axes: selection of configurations, execution of measurements, parameter/model estimation, and verification of the corrected recipient. Better experimental design, cheaper fixtures, and a better estimator are complementary interventions, not interchangeable autonomy levels.

Commission the configuration, not only the estimator. A useful calibration file must be associated with the actual sensors, joints, mounting, firmware, and payload. Commissioning additionally checks timing, limits, interfaces, and retained capabilities. A model update can pass held-out geometric tests while failing an integrated motion task; conversely, a compliant controller may mask a geometric error. The proposed acceptance record therefore distinguishes parameter fit, independent measurement, and task-level validation.

Reset and recovery are also experimental conditions. MEDAL++ learns undo behaviours for non-episodic practice [5]; FAST uses a structured gantry, object array, and tool-changing setup for repeatable component testing [49]. These are different ways of restoring a usable trial state. A comparison of calibration or design algorithms should account for that restoration work rather than treating each physical measurement as an equally priced query.

Table 13 collects the proposed hand-off requirements. It is an engineering contract, not an empirical assertion that every cited system implements every row. Study-specific realization and human boundaries are compared in Table 14.

Table 13: Minimum hand-off contracts for autonomous system-level robot engineering.

Hand-off	Required artifact	Acceptance evidence	Frequent hidden dependency
Requirement → subsystem	Capability, environment, hazard, resource budget, interface and testable criterion	Traceability from system objective to allocated constraint	Human resolves ambiguous or conflicting values
Design → fabrication/build	Typed CAD/code, bill of materials/dependencies, tolerances, process and version	Feasibility checks and reproducible build record	Manual repair, part selection, credentials, proprietary tooling
Build → commissioning	Physical/software identity, measured deviations, firmware and calibration state	Interface, timing, calibration and basic safety checks	Cable routing, mounting, charging, cleaning and datum setup
Commissioning → experiment	Valid initial state, protocol, reset/recovery, instrumentation and stop rules	Pre-flight checks and measurement integrity	Human resets, scene preparation and rare-fault recovery
Experiment → acceptance	Raw data, uncertainty, failures, resources, baseline and regression results	Independent criteria on target gain, invariants and retention	Selective reporting or evaluator coupled to generator
Acceptance → deployment	Signed release, compatible recipient, rollout, monitoring and rollback plan	Canary/shadow evidence and post-update health	Manual approval, access control and undocumented configuration
Deployment → later R&D	Versioned traces, intervention and maintenance history, negative evidence	Evidence is attributable to a known robot/environment version	Fleet heterogeneity and policy-dependent sampling bias

6.5 Deployment, maintenance, and fleet updates

Lifecycle development includes learning to compensate for damage, selecting a different available configuration, physically replacing a module, and sharing an update across robots. These interventions change different artifacts and should not all be called self-repair.

Learn around damage without restoring the body. Reset-free Trial-and-Error combines a simulation-built behaviour repertoire, Gaussian-process outcome learning, and Monte Carlo tree search [119]. Unlike episodic recovery, trials continue from the current state while the robot navigates around obstacles. Physical tests use a hexapod with one removed leg, external motion capture, and five repetitions of each arena condition. In the first reported physical condition, median episodes per target are 13 for RTE (interquartile range 12–14) and 28 for planning without the learned correction (24–31). These are recovery trials, not replaced hardware or autonomous field servicing. Precomputing the repertoire and providing localization remain part of the resource boundary.

This case illustrates how deployment and experimentation can coincide: the action both advances a task and informs the damaged-robot model. Its benefit concerns operational recovery. A later research-efficiency claim would require a separate comparison, rather than interpreting reduced episodes during the same adaptation campaign as unrestricted recursive improvement.

Reconfigure from supplied design knowledge. The UX-1 metacontrol system uses a functional knowledge model and diagnostic events to select a thruster-allocation configuration, then sends the resulting matrix to the low-level controller through ROS [120]. The reported fault experiments are software-in-the-loop tests in Gazebo for surge and heave, not physical flooded-mine trials. Available function designs are specified in advance. We therefore treat this implementation as enabling runtime infrastructure: it links diagnosis to a configuration change, but does not autonomously invent or test new alternatives. This boundary is useful precisely because a future researcher could propose alternatives for such a controlled adoption interface.

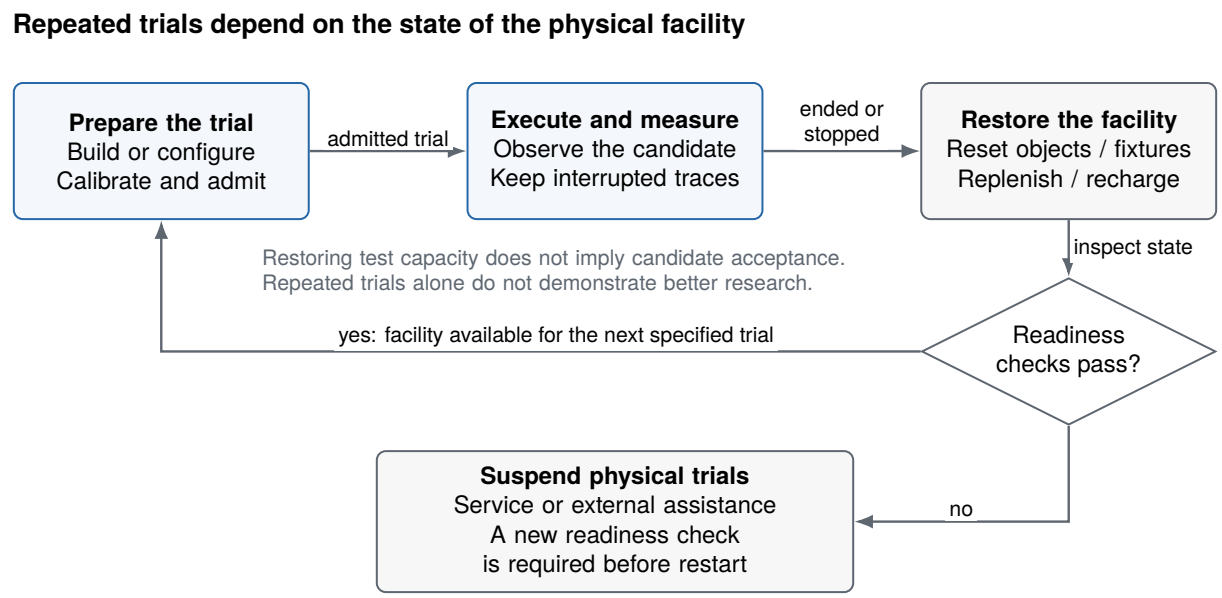

Figure 8: Physical experiment continuity is a state-management problem. This proposed facility workflow separates an admitted trial from restoration, readiness checking, and suspension when the facility cannot continue. The return arrow denotes renewed experimental availability, not recursive research improvement or candidate acceptance. Readiness is specific to the next experiment and does not replace its preparation and admission checks. The suspension branch has no automatic restart; service must be followed by another readiness check. The recording strip applies to every stage, including failed or interrupted attempts, and is not another stage in a serial pipeline. Deployment and version lineage are treated separately in Figures 4 and 11. This is a conceptual workflow, not a claim that a cited facility implements every operation autonomously.

Physical module replacement is a third intervention. Robot Metabolism demonstrates that suitable connectors and morphology permit this operation, but its operator-controlled experiments are not autonomous maintenance planning [113]. Behavioural compensation, rule-based reconfiguration, and material replacement can be combined; their individual reports do not establish that combined system.

Learn collectively while accounting for version lag. Yahya et al. distribute guided policy search across local robot workers and a shared parameter server [121]. Their four-robot door-opening study evaluates one learned policy on an unseen door; local policies begin from kinesthetic demonstrations. Worker-count and asynchronous-speed comparisons are conducted in simulation. Concurrent collection and optimization improve utilization while introducing stale policy/data versions. This is a concrete mechanism for collective policy development, not a validated fleet-wide hardware/software rollout process.

The distinction matters for interpreting scale. More robots can reduce elapsed training time while increasing total robot-hours, apparatus cost, or support effort. A shared model can also inherit site-specific bias from uneven contributions. Fleet-level comparisons need both a recipient outcome and an account of which versions collected the data, supplied the gradients, and received the update.

From recovery episodes to lifecycle evidence. The selected literature supports bounded calibration, recovery, reconfiguration, and collective learning. It does not yet substantiate every deployment practice proposed for RobotRSI. Staged releases, compatible configuration manifests, monitored exposure, and recoverable updates are requirements of the perspective rather than capabilities inferred from a policy-learning experiment. Longitudinal evidence should connect repeated faults, maintenance actions, wear, and intervention costs to design changes. Shorter downtime alone cannot distinguish a durable improvement from deferred maintenance.

6.6 Comparative system-level cases

Figure 8 isolates a facility-level condition that a software research loop can overlook: completing a measurement does not make the apparatus ready for another experiment. Object positions, charge, fixtures, calibration, and wear can change during a trial. Restoration must therefore be followed by a readiness check, with suspension and service when it fails. This feedback maintains experimental capacity; it is not evidence that the research procedure has improved. Candidate acceptance and release are separate decisions.

The facility boundary may include cloud compute, external workshops, and people; physical co-location is not required. Its records should connect the trial and candidate identities to both measured outcomes and the work needed to obtain them. Keeping only successful, completed trials can hide failed builds, interrupted tests, and manual recovery. In the proposed workflow, recording spans preparation, execution, and restoration rather than occurring only after a release. This makes experimental throughput and reproducibility assessable without assuming that every operation is autonomous.

Table 14 compares selected system-engineering pathways. Construction, active calibration, damage recovery, runtime reconfiguration, and collective learning operate on different experimental units. The table distinguishes implemented development from enabling physical or software infrastructure, and reports realization independently of that distinction. This prevents a manually directed physical transition from outranking a computational research method simply because only the former involves hardware.

Three integration patterns emerge. First, *same-platform adaptation* updates software or calibration on the deployed robot and is comparatively reversible. Second, *facility-mediated succession* produces a component or child robot using external manufacturing and test infrastructure. Third, *fleet-mediated development* aggregates experience, trains centrally, and releases to multiple recipients. A complete RobotRSI architecture may combine all three, but its identity and recursive claims must specify which actor researches, which artifact changes, and which version benefits.

6.7 Section synthesis

Autonomous system-level robot engineering is the discipline of closing hand-offs. Requirements must be allocated across shared budgets; hardware and software changes must be co-designed or their assumptions made explicit; fabrication, calibration, reset, and recovery must be part of the experimental boundary; and acceptance must produce a versioned, monitorable, reversible update. The comparison identifies different bottlenecks: evolutionary construction restricts bodies to what a facility can build; active calibration economizes measurement selection; contact fixtures reduce metrology requirements; online recovery learns around damage; metacontrol connects diagnosis to supplied configurations; and asynchronous learning shares experience across recipients. Their open interfaces concern the cost and validity of moving from one such capability to another, not simply the absence of a more powerful optimizer.

The feasible chain depends strongly on robot type and operating context. A factory manipulator can rely on fixtures and external infrastructure; a soft microrobot, surgical platform, underwater vehicle, planetary rover, or distributed fleet cannot share the same modification, reset, and acceptance assumptions. Section 7 therefore treats robot context as an axis that conditions every loop rather than as a list of application examples.

7 RobotRSI Across Robot Types and Operating Contexts

The same research mechanism can have different engineering meanings on different robots. A Bayesian optimizer may choose a flight experiment, a simulated catheter design, or a controller for a swarm. A learned representation may be updated on a field robot or on a workstation receiving microscopic images. Comparing these systems requires more than an application label: the experimental boundary determines what is changed, where evidence is obtained, and who can use the result.

We describe context using

$$c = (b, e, p, s, m, w, o, i), \tag{3}$$

Table 14: Selected system-engineering methods and realization boundaries. A denotes an implemented robot-development operation; C denotes enabling physical or runtime infrastructure without a demonstrated autonomous research decision in the reported case. These use roles are not quality or maturity ranks. Physical execution, research decisions, and later R&D benefit are separate claims.

Study / role	Changed object	Decision or transformation	Evaluated realization	Boundary for interpretation
Mother Robot [2] (A)	Body / controller population	Generational selection and robotic construction	Physical child robots in a bounded grammar	Supplied modules and producing apparatus
ARE/RoboFab [22, 112] (A/C)	Evolved bodies and production process	Evolution framework and robotic assembly infrastructure	Reported facility components and practical designs	Complete continuous orchestration not established by the framework
Bosio and Mueller [109] (A)	Multi-UAV attachment layout and controller	Joint optimization for disturbance rejection	Optimized/suboptimal layouts flown with three/four quadcopters	Human realization; no autonomous assembly or later-R&D comparison
AutoSoC [76] (A)	Obstacle-avoidance policy and accelerator layout	Training/validation plus digital implementation flow	Placed-and-routed accelerator candidates	No silicon, aircraft installation, or integrated mission test
RobotSmith [66] (A)	Tool geometry and manipulation trajectory	VLM generation plus simulated joint refinement	Selected tools printed and used on an XArm7	Production and mounting external; base robot fixed
iRonCub CAD co-design [110] (A)	Link geometry and controller-facing model	Evolutionary search with automated structural FEM filter	Digital CAD/URDF candidates for jet-powered humanoid	No autonomous fabrication or commissioning established
Robot Metabolism [113] (C)	Module topology and replacement	Robot-executed physical transitions	Growth, recovery, and assisted assembly experiments	Operator commands/scripts; designed terrain; not autonomous design research
Daş and Burdick [114] (A)	Kinematic residual model	GP-UCB measurement selection	Three simulated arm geometries	No physical calibration campaign in the inspected study
Selingue et al. [116] (A)	Kinematic parameters and residual GP	Active measurement plus supplied stopping rule	Physical calibration on two manipulators	Method design tuned on an existing dataset; no method self-revision
Next-best-view [115] (A)	Hand–eye parameters	Information-based view selection	Synthetic and real image/pose datasets	Motion-planning cost excluded; supplied board and intrinsic calibration
MUKCa [118] (A)	Kinematic parameters	Constrained-fixture estimation	Cobot calibration; Franka task validation	Human-guided configurations and capture
Reset-free Trial-and-Error [119] (A)	Damaged-body outcome model / behaviour	Repertoire, online learning, and planning	Simulated robots; damaged physical hexapod	Precomputed repertoire and external tracking; compensation, not body repair
UX-1 metacontrol [120] (C)	Thruster-allocation configuration	Functional reasoning selects a supplied alternative	Gazebo software-in-the-loop fault tests	Runtime reconfiguration; new design generation not implemented
Distributed guided policy search [121] (A)	Shared policy / local policies	Asynchronous collection and parameter updates	Four-robot door task; scaling tests in simulation	Human demonstrations; training architecture, not fleet-release validation

where b is intended task breadth, e embodiment family, p physical realization, s scale, m mission, w operating world, o organizational scale, and i available infrastructure. These coordinates are multi-label, not disjoint robot classes. A soft medical device can be millimetre-scale, externally actuated, and optimized in a laboratory; a general-purpose mobile manipulator can still be evaluated in a narrowly specified office setting. We use this model to compare research arrangements, not to infer autonomy from morphology.

7.1 Task breadth and specialization

Intended generality versus tested breadth. General-purpose development concerns a distribution of tasks and conditions, not simply a long list of instructions. The experience systems in Section 5 illustrate different dimensions of breadth. SOAR spans language-conditioned tabletop activities; AutoRT coordinates mobile-manipulator experience with mixed execution modes; RoboCat studies generalist training and subsequent adaptation; and Self-Improving Embodied Foundation Models examines autonomous practice from a broadly pretrained policy [6, 93, 104, 105]. These are complementary experiments, not successive points on one generality scale. Task proposals, feasible executions, training examples, and validated recipient capabilities remain different sets.

Specialized engineering can expose a comparatively clear experimental question. FAST searches soft-gripper designs within an instrumented component workflow [49]; safe Bayesian locomotion tuning tests a bounded controller interface [86]. Their narrow scope makes some comparisons tractable, while leaving transfer to other objects, terrain, or robot versions unresolved. Conversely, broad task descriptions expand neither a platform's physical action space nor the permitted research changes by themselves.

For a general-purpose RobotRSI system, choosing the intervention level becomes part of research. A failed household task may call for additional data, a different gripper, new sensing, or a change in planning. A study that optimizes one of these objects should identify that scope rather than inherit the breadth of the overall mission. Retention and held-out task families then test whether the update broadens capability or merely moves performance between tasks.

7.2 Embodiment, physical realization, and scale

Embodiment constrains the search representation. On articulated platforms, changes to body geometry, sensing, and control can sometimes be represented as separate variables, although their effects remain coupled. Soft devices put material and geometry assumptions directly inside the motion model. Ghoreishi et al. optimize a soft catheter's design in an outer Bayesian loop while an inner gradient-based optimization selects actuator moments [122]. The objective concerns alignment with specified vessel shapes under a planar deformation model. The reported evaluation is numerical, before manufacturing, rather than autonomous catheter redesign during a procedure. This is a concrete medical-robot co-design method with a computational realization boundary.

Soft robotics also illustrates why a robot category is not a method category. The autonomous soft-design literature spans parametric search, simulated morphology–control co-evolution, learned models, and aspirations for high-throughput manufacturing [59]. Zardini et al.'s tensegrity study occupies the simulated co-evolution part of that space [61], whereas FAST uses adaptive physical testing of printed grippers [49]. Their shared material compliance does not make their feedback, facility, or realization boundaries equivalent.

In aerial engineering, the recipient's payload and energy budgets can change which algorithms are feasible. AutoPilot explores combinations of autonomy computation and UAV design assumptions across vehicle sizes [123]; AutoSoC couples an obstacle-avoidance policy to an accelerator layout but stops before silicon and vehicle integration [76]. Bosio and Mueller instead alter a cooperative vehicle–payload layout and controller and evaluate the resulting arrangement in flight [109]. These cases respectively end at system design-space exploration, placed-and-routed compute candidates, and a physically assembled multi-UAV recipient. The object and endpoint must therefore be recorded independently of the shared aerial context.

Microscopic robots need not contain the learner. Muiños-Landin et al. combine real artificial microswimmers with externally controlled reinforcement learning [124]. Their navigation experiment includes Brownian motion and supports collective learning through external control. The relevant learning system consequently includes the observation and control apparatus, not just the particle. This demonstrates physical learning at small scale without establishing onboard computation, autonomous fabrication, or biomedical deployment.

Xu and Zhu provide a contrasting LLM-based route [125]. Their swimmer models interact with a prompted LLM through action and displacement histories in a computed hydrodynamic environment. The published method also supplies history clearing when progress stalls and numerical transformations for the interface. It demonstrates gait acquisition for model swimmers, not experiments on fabricated microrobots. Thus, physical embodiment and agentic decision-making are independent descriptors: a simulated LLM study is not more physically closed than an externally mediated particle experiment.

Scale also changes the appropriate recipient. A fabrication recipe can improve a batch of microscopic devices even when individual devices cannot be rebuilt. A large field robot may instead retain its body while changing software or calibration. The full-stack vision includes both arrangements, but their lineage and evaluation units differ: design, specimen, individual robot, and population should not be silently interchanged.

7.3 Missions, environments, and experimental conditions

Experiment location is a methodological choice. Ryou et al. optimize quadrotor trajectory timing using feasibility evidence at different fidelities [126]. The physical campaign combines simulation with motion-capture flight experiments; it does not run all three proposed fidelity levels in every campaign. Flight data influence trajectory selection, so hardware participates in the search rather than serving only as a final demonstration. The platform, controller, waypoints, and feasibility limits are supplied. This illustrates how a demanding mission can motivate selective physical experimentation.

RADIUM presents the complementary order: it searches and repairs against simulated failure distributions and then subjects selected policies to physical race-car and Robotarium tests [89]. In Ryou et al., flight observations enter the candidate-selection loop; in RADIUM, hardware evaluates transfer after simulator-driven repair. Both contain physical trials, but only the former uses those trials as within-campaign research feedback.

Underwater fault recovery can motivate the opposite allocation. Ahmadzadeh et al. search control policies on a Girona500 dynamics model after a specified thruster failure [127]. Their comparison of derivative-free search methods and policy representations is evaluated in simulation. The approach is intended to avoid costly learning on a damaged vehicle, but does not demonstrate sea trials or a fully validated onboard fault-identification loop. The methodological point is still relevant: when physical trials are scarce, deciding which uncertainty can be resolved computationally becomes central.

Field experience supplies changing supervision. COTRATE learns traversability assessments from proprioceptive and inertial signals and uses them to train a visual model with feature replay [128]. Its preprint evaluates data from Spot and Husky across eleven terrains. A manually selected reference terrain anchors the scores. Its path-effort metric uses terrain-specific motor-current estimates and path distance; it is not a direct measurement of total energy consumed by an autonomous research campaign. The case connects changing field experience to perception updates while exposing assumptions behind the feedback signal.

These examples distinguish three placements of learning: physical experiments selected during a campaign, model-based search intended for constrained operation, and continual updates from encountered experience. Their budgets cannot be compared using iteration counts alone. One iteration may consume flight time and recovery labour, another simulator evaluations, and another a batch of logged sensor data.

Reset and irreversibility. A scene is not necessarily restored when a trial ends. Terrain is disturbed, batteries discharge, fixtures drift, and specimens may be consumed. MEDAL++ shows that undo behaviour can be learned as part of physical practice [5]; Figure 8 separates restoration from readiness and candidate acceptance. For experiments that cannot be repeated on the same specimen, comparisons require matched

Table 15: Conditional implications of operating context for autonomous R&D. These are design questions, not universal claims or an ordinal difficulty scale.

Context condition	Likely bottleneck	Required evidence practice	Enabling design response
Fixture-rich laboratory	Overfitting to instrumentation, layout, or reset	Report fixture dependence; test held-out layouts and degraded sensing	Versioned fixtures, randomized scenes, automatic exception logs
Production factory / logistics site	Downtime and non-stationary workload	Exposure-normalized comparison; staged release and rollback	Shadow trials, canary cells/robots, production telemetry
Home or public human space	Safety, privacy, diverse objects and users	Consent-aware data lineage; near-miss and intervention accounting	Conservative authority, privacy filters, remote recovery
Remote field, underwater, or space	Delayed evidence, weak communication, costly retrieval	Predeclared abort limits; uncertainty and missing-data treatment	On-board diagnosis, fault containment, digital rehearsal, sparse uplink
Medical or safety-critical intervention	Irreversibility and tightly constrained authority	Traceable specimens, formal acceptance, independent oversight	Simulation/phantoms, bounded adaptive parameters, certified fallback
Soft, micro, or batch-fabricated systems	Specimen variation, yield, metrology, simulator discrepancy	Batch statistics; process and specimen identity; destructive-test accounting	Automated fabrication/metrology, multi-fidelity surrogate calibration
Fleet deployment	Confounding by site, age, maintenance, and exposure	Randomized or defensible assignment; version and exposure logs	Cohort registry, federated evidence, staged fleet release

specimens or a design that accounts for changing state. Such controls belong to the proposed evaluation protocol, not to every cited implementation.

Table 15 retains the broader implications for medical, production, remote, and batch-fabricated settings. These are engineering questions rather than claims of autonomous R&D already demonstrated in every domain. In particular, a simulated underwater study, an outdoor navigation dataset, and a medical design model cannot collectively stand in for evidence about space, nuclear, agricultural-production, or clinical systems. Those contexts require their own source-level assessments.

7.4 Individual, collective, and distributed systems

A collective can receive automatically designed software. AutoMoDe generates swarm control software by composing supplied behavioural modules into probabilistic finite-state machines [129]. Topology, conditions, and parameters are searched against a mission objective, with aggregation and foraging evaluated on e-pucks. The researcher is the design system, while decentralized robots execute the resulting controller. Distributed operation therefore need not entail distributed research.

This differs from fleet-mediated experience collection such as AutoRT. A swarm-design objective concerns the joint behaviour produced by interacting controllers; a data fleet can instead aggregate episodes for a shared learner. For the first, robot count, density, communication, and arena geometry affect the collective outcome. For the second, robot/site exposure and the composition of the training data affect the recipient model. Neither arrangement guarantees that every participating robot benefits equally.

Self-Improving Embodied Foundation Models makes this collective-learning boundary measurable: three or four LanguageTable stations contribute autonomous practice to a common update while one person performs monitoring and resets [105]. The fleet is the experimental executor and data source; the shared policy is the changed recipient. This differs from a swarm whose inter-robot interaction is itself the task behaviour and from a population that inherits distinct body designs.

Infrastructure is part of the experimental system. The division between executor, target, and recipient introduced in Section 2 is especially useful here. Microscopic learning can use external observation and actuation; flight optimization can use motion capture; evolutionary construction can use printers, component banks, and assembly robots [112, 124, 126]. These dependencies make particular experiments possible. The

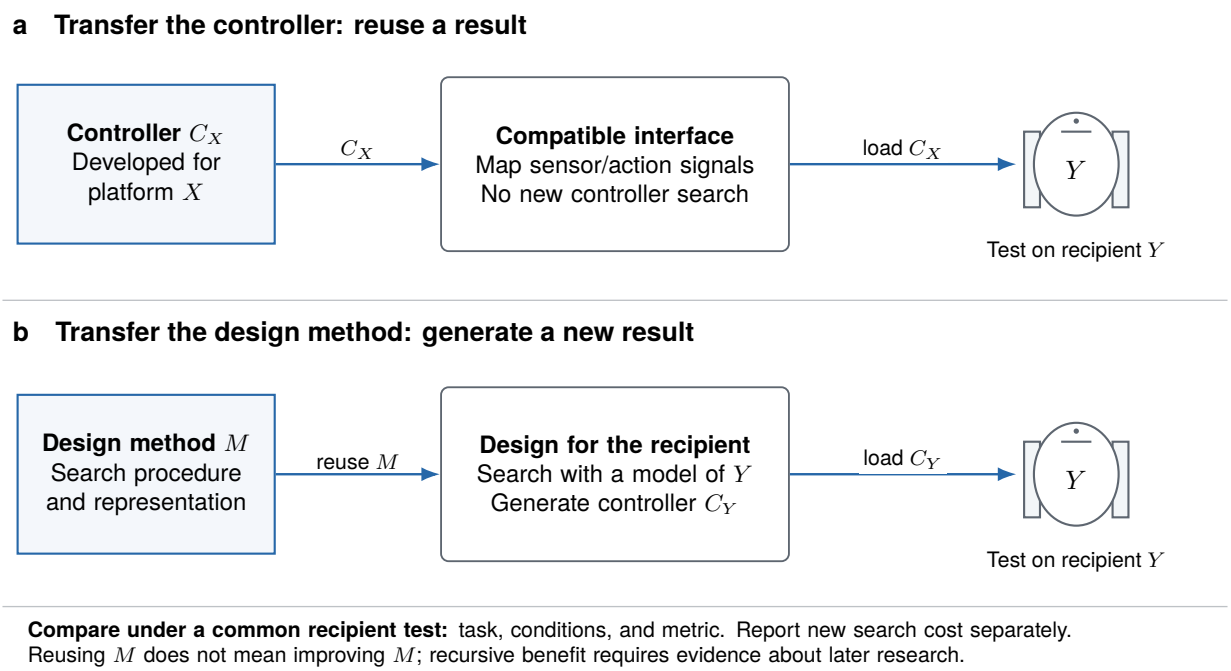

Figure 9: Transferring a result is different from transferring its design method. This conceptual comparison is informed by the swarm experiments of Kegeleirs et al. [130], not a reproduction of their setup or results. In (a), an existing controller crosses a compatible interface; in (b), a reused method searches with the recipient's model. The blue arrows denote the labelled artifact hand-offs, not performance gains. The two generic robot glyphs denote separate tests on the same recipient platform type Y, not robot construction or succession. Compatibility is an assumption to verify in both routes. The common-test and cost-accounting strip states this survey's comparison requirements; method reuse alone establishes no recursive improvement.

relevant question is which capabilities the declared system possesses, not whether all computation is physically inside the recipient.

Operational distribution and authority distribution should also be distinguished. Remote computation says where an algorithm runs; it does not say who selects the objective, permits a trial, changes an instrument, or accepts an update. These roles may remain human, automated, or mixed. Table 16 compares selected arrangements using the actual research object and realization boundary rather than a scalar autonomy score.

7.5 Transfer and inheritance

Transfer the result or transfer the method. A useful distinction is between reusing a controller developed elsewhere and rerunning a transferable design procedure for the recipient. Kegeleirs et al. test both on functionally comparable e-puck and Mercator platforms [130]. Their physical experiment uses three robots per swarm, whereas a larger experiment uses pseudo-reality. The comparisons generally favour designing for the recipient over importing a controller from the other platform. This supports method reuse under the tested correspondence, not a claim that the design method improved itself.

Figure 9 abstracts the two comparisons. Method transfer may preserve a useful search representation while adapting the resulting artifact to the new body. Artifact transfer avoids that new search cost but can inherit mismatched assumptions. A fair comparison should report the cost of recipient-specific design alongside deployment performance; a stronger controller is not automatically the cheaper transfer strategy.

An artifact's role depends on its later use. Datasets, policies, calibration priors, surrogate models, test procedures, and experiment generators can all be inherited. They are not permanently divided into operational artifacts that cannot support recursion and research artifacts that necessarily do. A policy used to perform the mission is an operational capability; the same policy used to reset experiments can become a research instrument. A dynamics model can likewise support control or the selection of informative tests. The recursive

Table 16: Selected context-specific development loops. Physical evidence, computational design, and mixed execution are distinguished explicitly; the rows are not an exhaustive coverage map or an autonomy ranking. The last column records a condition limiting interpretation.

Study / context	Changed object	Research and feedback	Realization / recipient	Condition or boundary
AutoRT [93]	Task proposals and experience	Foundation models coordinate robot data collection	Mobile-manipulator fleet; collected episodes support learning	Mixed autonomous and human execution
Self-improving EFM [105]	Policy, reward labels, and replay data	Three/four stations practice and update a shared policy	Real LanguageTable fleet; simulated ALOHA	One human monitors/resets; real ALOHA update incomplete
FAST [49]	Soft-gripper geometry and model	Adaptive test selection; automated grasp trials	Printed specimens; selected component design	External printing, post-processing, and mounting
AutoPilot [123]	Autonomy and UAV design	Model-based system co-design	Computational candidates at several vehicle sizes	Candidate specification is not a realized flight system
AutoSoC [76]	Policy and accelerator layout	Training plus synthesis/place-and-route	Digital accelerator candidates for aerial obstacle avoidance	No chip fabrication or aircraft integration
Bosio and Mueller [109]	Multi-UAV layout and controller	Joint optimization for robustness	Three/four quadcopters carry physical payloads	Human realization; no autonomous assembly
Multi-fidelity flight [126]	Trajectory timing	Simulation and physical feasibility feedback	Motion-capture flight participates in search	Supplied vehicle, controller, waypoints, and limits
Soft catheter [122]	Geometry, material, actuation	Outer Bayesian and inner gradient-based optimization	Numerical shape matching; pre-manufacture design	Planar model; no physical or clinical validation here
Artificial microswimmers [124]	Navigation policy	Externally mediated reinforcement learning	Real particles; external control apparatus	Physical learning does not imply onboard learning
LLM swimmer [125]	Swimming gait	Prompted action selection with computed feedback	Hydrodynamic models; no fabricated swimmer trial	Supplied history management and numerical interface
Thruster recovery [127]	Recovery controller	Derivative-free policy search	Simulated damaged underwater vehicle	Specified fault; no demonstrated sea-trial recovery
COTRATE [128]	Traversability models	Self-supervised terrain scores and feature replay	Spot/Husky field data and path evaluation	Manual reference terrain; path-effort proxy
Swarm transfer [130]	Collective control software	Offline search; controller versus method transfer	Physical three-robot swarms; larger pseudo-reality study	Functionally matched platforms, not arbitrary bodies
ARE [112]	Body/controller population	Facility-mediated evolutionary development	Modular designs and physical construction infrastructure	Parts, assembly, and test-facility dependencies

Table 17: Selected inheritance mechanisms and evaluated endpoints. Artifact type does not determine whether it serves operation or later research. A downstream benefit must be established at the claimed recipient and use; these studies do not share a single recursive-improvement metric.

Study and inherited object	Recipient comparison	Evaluated endpoint	Boundary
Open X-Embodiment [106] Multi-robot experience and policies	Cross-embodiment training and robot-task evaluation	Policy performance with pooled experience	No automatic research-selection claim follows
COTRATE [128] Visual terrain representation	Cross-platform tests on Spot/Husky	Traversability estimates and proxy-based path effort	Compatible terrain responses; not total measured energy
Swarm transfer [130] Controller or design method	Imported controller versus recipient-specific search	Collective mission performance on recipient platforms	Method portability, not method self-improvement
RoboCat [104] Generalist model and accumulated experience	Later adaptation of successive model versions	Bounded new-task adaptation performance	Data ablation and adaptation tests do not isolate one total-budget effect
GenSim [98] Task library and trained generator	Generator tests on held-out task specifications	Task-code generation metrics; separate robot-policy experiments	Code metrics omit semantic errors; physical adaptation is a distinct test

question concerns the downstream use and effect, not the filename or whether the outer algorithm changed.

Table 17 separates demonstrated transfer endpoints from later-research claims. RoboCat and GenSim provide bounded comparisons involving later adaptation or generator performance, as discussed in Sections 5 and 8. Their evidence should be preserved without merging all changes into an undifferentiated claim of improving research efficiency.

Inheritance also has a compatibility cost. A visual model can be transferred between robots with different terrain responses; an evolved controller can be deployed at a different density; a calibration prior can be attached to changed hardware. These transfers call for tests of the relevant correspondence, including negative transfer and retained capabilities. Cross-platform reuse demonstrates neither unlimited generality nor inevitable progress over successive versions.

7.6 Comparative synthesis

Three contextual distinctions cut across the engineering objects. First, *where experiments occur* determines the information and cost available to research: a simulator, an instrumented component bench, a flying vehicle, and logged field experience expose different discrepancies. Second, *what physically changes* distinguishes policy updates, new device designs, specimen batches, and constructed descendants. Third, *what is inherited* distinguishes an existing artifact from a procedure that can generate a recipient-specific artifact.

These distinctions allow specialized and general-purpose systems to be compared without ranking whole domains. They also prevent a collection of partial loops from being presented as one universal implementation. The selected studies establish several useful combinations of context and autonomous development, but uneven coverage remains a limitation of the present evidence base. Section 8 examines the additional evidence needed when a transferred or updated capability is claimed to improve subsequent R&D itself.

8 Closing the Loop: Recursive Robot Self-Improvement

Robot self-improvement is often illustrated as a single circular arrow. That picture suppresses the hardest question: *what returns, to which version, and with what demonstrated consequence?* Repeating a fixed optimizer can produce increasingly capable controllers. Updating a robot can improve its task performance. Improving

a laboratory can increase experimental throughput. None alone establishes that the system became better at conducting subsequent robot R&D.

We distinguish *recursive use* from *demonstrated recursive benefit*. Recursive use occurs when an output of research episode k changes a capability used in a later research episode. Demonstrating benefit additionally requires evaluating the downstream effect. Neither requires self-containment, unrestricted self-modification, monotonic gains, or exponential growth. A distributed facility with declared human authority can instantiate this feedback. Conversely, on-board learning does not establish it merely by improving task performance. A fixed outer algorithm may still update a learned research model or memory; the relevant question is what capability changes and how it affects later research.

8.1 From local improvement to system updates

Let S_k be the recipient robot system, R_k the research process, P_k the problem distribution, B_k the resource budget, and E a protected evaluation protocol. A local research episode produces an artifact and evidence

$$(z_k, D_k) = R_k(S_k, P_k, B_k), \tag{4}$$

followed by an acceptance decision a_k. If accepted, integration creates

$$S_{k+1} = U(S_k, z_k, D_k, a_k), \tag{5}$$

where U includes configuration, calibration, regression tests, release, and rollback information. This is a *system update*. It supports self-improvement only when the intended recipient improves under a defensible comparison, rather than when a candidate merely scores well inside its search loop.

Task benefit and research benefit are different outcomes. Define $V_E(S)$ as capability under the protected evaluation and $J_E(R; S, P, B)$ as the value of the validated artifacts produced by a research process under a fixed problem distribution and budget. Then

$$\Delta_{\text{task}} = V_E(S_{k+1}) - V_E(S_k), \tag{6}$$

$$\Delta_{\text{R\&D}} = J_E(R_{k+1}; S, P, B) - J_E(R_k; S, P, B). \tag{7}$$

The quantities may be vector-valued and include safety, retention, uncertainty, human labour, physical wear, and compute. A positive Δ_{task} does not entail positive $\Delta_{\text{R\&D}}$. Conversely, a better calibration rig may improve later evidence without immediately increasing task performance.

Identity matters in Equation (5). A software update on the same serial-numbered robot, a newly manufactured body controlled by inherited software, and a child platform assembled by a parent facility are different succession relations. The artifact ledger should state which hardware, software, data, calibration, research tool, and evaluation versions constitute S_k and R_k. Otherwise, claims of "the robot improved itself" can silently switch recipients between rounds.

8.2 Recursive pathways

Figure 10 separates four claims that circular diagrams often collapse: task use, task benefit, recursive use, and recursive benefit. It then identifies three carriers through which a versioned capability can enter a later research process. The figure is a claim map, not a temporal loop or maturity ladder.

Task improvement and repeated optimization. Evolutionary manufacture [1, 2], safe controller tuning [86], and autonomous practice [5] can produce better robot artifacts. This task-level result is valuable regardless of whether a later research effect is measured. Repeating the procedure does not by itself identify that effect, but describing an optimizer as fixed is also insufficient to rule it out: its surrogate, archive, or experimental policy may change with experience. The task row in Figure 10 additionally distinguishes installing an accepted update from demonstrating its benefit under a protected comparison.

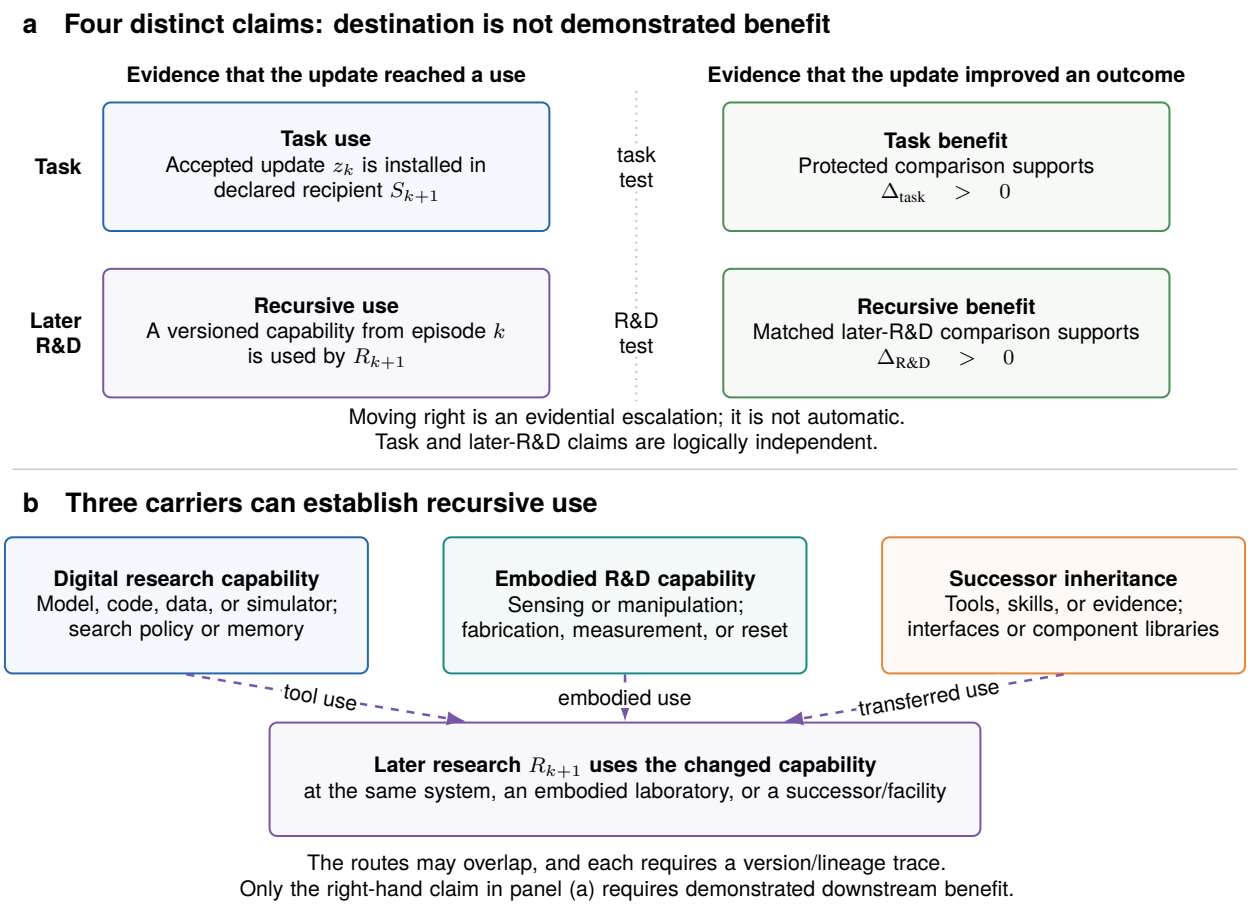

Figure 10: Claim semantics and carriers of recursive robot research. Panel (a) distinguishes two destinations and two evidence levels. Installing an update for task use is not a task-benefit result, and using an inherited capability in later research is not evidence that later research improved. Moreover, task benefit and research benefit need not coincide. Panel (b) groups three non-exclusive ways in which a changed capability can enter R_{k+1}. Dashed purple arrows denote conceptual later-research use; they do not denote a reported implementation or a measured gain. The version/lineage record establishes recursive use, whereas the matched comparison in Figure 12 tests recursive benefit.

Research-method and tool feedback. Evidence updates a surrogate, simulator, evaluator, acquisition strategy, reset policy, or code-generating agent used in later episodes. FAST, for example, couples physical results to model calibration and later design selection [49]. This is a direct route to research feedback. Yet model updating alone is not proof of benefit: a calibrated model can become overconfident or narrow the search incorrectly. The later campaign must be compared against the prior research process under matched conditions.

Embodied capability feedback. An improved robot may collect higher-quality data, manipulate a broader set of specimens, repair its test scene, carry better instruments, or fabricate more complex candidates. SOAR links autonomous experience collection to an improved instruction-following policy [6]; an additional recursive edge would be established if that recipient policy demonstrably improved the efficiency or reach of a later data-and-research campaign. Similarly, damage adaptation restores behaviour [3, 39], but it becomes research-capability feedback only if the restored or enhanced platform subsequently performs better diagnosis or experimentation.

Successor and population inheritance. A facility or parent system can transmit genomes, controllers, component libraries, failure records, or experimental procedures to physical descendants. Mother Robot makes physical succession tangible within a bounded construction grammar [2]; ARE proposes longer-lived software–hardware integration and population management [22, 112]. A descendant with higher locomotion score is an improved artifact. A recursive-R&D claim additionally requires that inherited research capabilities

enable that descendant or facility to generate stronger subsequent descendants or evidence.

Computational self-improvement provides a useful comparison. The Darwin Gödel Machine evolves coding agents that modify their own implementation, maintaining an archive and evaluating changes on coding tasks [41]. Hyperagents makes both the task agent and its modification procedure editable and reports transfer of meta-level improvements across computational domains [42]. These are relevant evidence for research-process modification; they are not physical robot experiments. Their transfer to robotics requires testing changes under fabrication, reset, measurement, integration, and recipient constraints that coding benchmarks do not supply.

The three carriers in Figure 10 can coexist; they are not a maturity ladder. Direct method feedback may occur without hardware succession, while a successor may inherit both digital tools and embodied experimental skills. The unit of analysis is the particular inherited change and its downstream effect, not the visual presence of multiple generations. Here R_k includes persistent learned state as well as program code. Rewriting the outer algorithm is consequently neither necessary nor sufficient for recursive benefit.

8.3 Persistent knowledge, versions, and successor systems

Multi-round research requires memory that survives beyond a chat transcript or model checkpoint. At minimum, the ledger links requirements, hypotheses, candidates, code/CAD/data, tool and simulator versions, physical specimens, calibrations, trial conditions, raw observations, analysis, human interventions, decisions, and recipient releases. Negative results and invalid trials are first-class records: without them, a later system repeats unsafe experiments or learns a biased map of the design space.

Research memory has four distinct roles. *Reproduction memory* recreates a result. *Decision memory* explains why a candidate was selected or rejected. *Transfer memory* states where an artifact is compatible. *learning memory* supplies data for improving later research choices. A repository can satisfy the first role while failing the other three. Natural-language rationales should point to machine-checkable artifacts and raw evidence rather than replace them.

Successor identity should use explicit parentage. Let

$$\Pi_{k\to k+1} = \{z, \text{source}(z), \text{recipient}(z), \text{compat}(z), \text{test}(z)\} \tag{8}$$

be the inheritance manifest. It records copied and transformed artifacts, compatibility assumptions, and acceptance tests. This prevents a software policy from being attributed to a body on which it was never evaluated, and prevents an updated simulator from silently changing the benchmark used to declare progress.

Figure 11 is a reporting template, not an empirical history. Its manifest matrix exposes asynchronous changes: a hardware revision may persist through several controller updates, while research tools or policy change less frequently. The release lineage distinguishes configurations actually adopted from candidates only tested and rejected. Both kinds of event retain their evidence, human actions, and costs. Repeated evaluator IDs make the protected comparison explicit; if the protocol changes, a new comparison or a justified bridge study is needed. A version ledger establishes identity and provenance, not recursive benefit by itself.

Fleet and population settings need lineage at two levels. The *individual lineage* records the exact hardware and software state producing each observation. The *research lineage* records which observations and tools produced an update. A centrally trained release may have hundreds of robot ancestors and thousands of recipients. This is compatible with RobotRSI, but it makes phrases such as "the robot learned from itself" too imprecise for evidence.

8.4 Cross-stack research direction and budgeting

Once several layers are mutable, the system must allocate research rather than only optimize within a chosen layer. A navigation failure might arise from camera placement, vibration, synchronization, representation, localization, planning, tyre wear, or an evaluator that rewards unsafe shortcuts. Research direction is the decision to spend budget on discriminating among such explanations and then select an intervention scope.

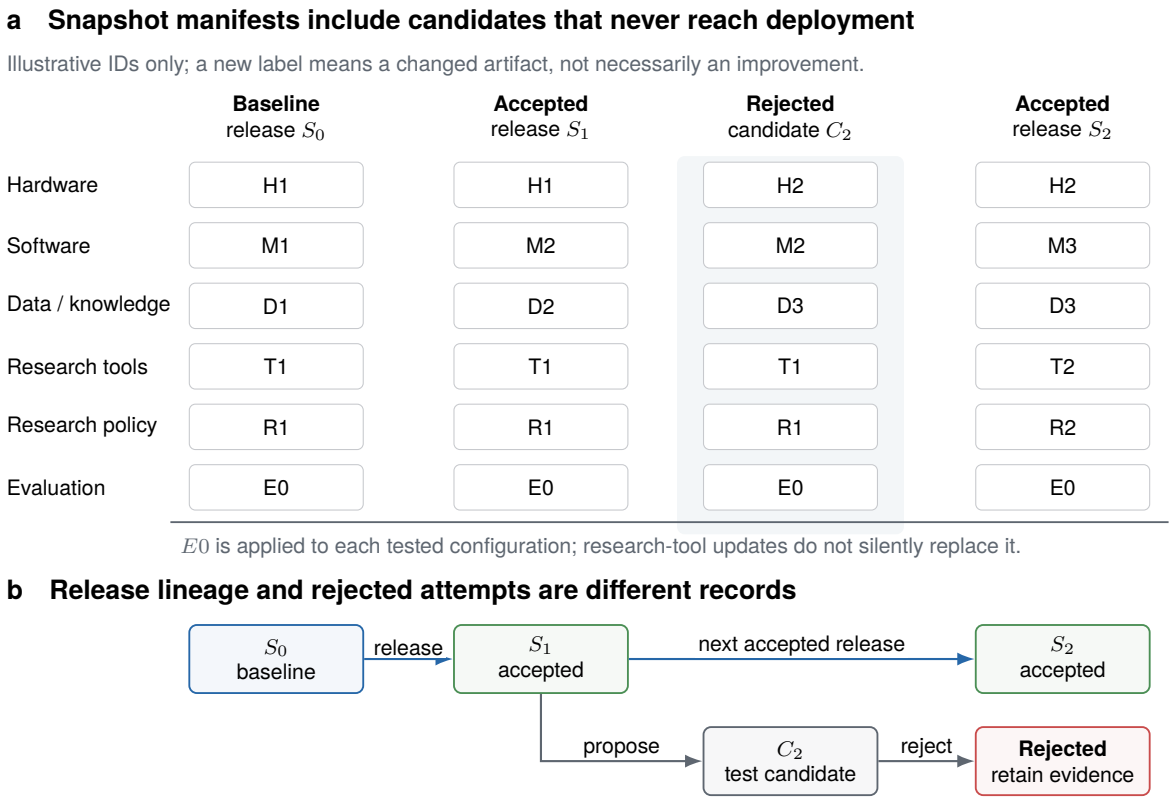

Figure 11: Version manifests and release lineage for multi-round RobotRSI. The upper panel records asynchronous artifact versions, including the rejected candidate C_2. Identical IDs mean unchanged artifacts; different IDs do not establish causal independence or improvement. In this illustrative history, C_2 tests new hardware with the earlier software and is rejected; a later compatible combination becomes S_2. The lower panel distinguishes accepted release succession from a rejected attempt. It is not a complete research dependency graph: knowledge from rejection can inform later proposals without deploying the rejected configuration. Evaluation version $E0$ remains fixed across this comparison. All IDs and outcomes are hypothetical.

Let candidate research programs $q \in \mathcal{Q}$ each contain a target, method, fidelity sequence, stopping rule, and expected cost. A bounded selector can use

$$q^* = \arg\max_{q \in \mathcal{Q}} \left[\mathbb{E}\left[\Delta V \mid q, H_k\right] + \lambda \mathbb{E}\left[\Delta I \mid q, H_k\right] - \boldsymbol{\mu}^\top \mathbb{E}\left[C(q)\right] \right], \tag{9}$$

subject to hard safety, authority, and compatibility constraints. ΔV is recipient capability gain, ΔI is durable information or future R&D value, C is a vector of compute, robot hours, materials, human time, delay, and risk, and H_k is research memory. The expression is a design abstraction, not a claim that these quantities are already calibrated for general robotics.

Information value prevents exclusive focus on immediately deployable performance. A diagnostic experiment, new sensor test point, or negative result may be valuable because it reduces uncertainty for later choices. At the same time, speculative future value should not override safety or become an unfalsifiable justification for resource use. Prospective predictions should be logged and scored against realized downstream benefit.

Cross-stack proposals complicate attribution. If a lighter body, new camera, larger model, controller, and test protocol change together, later gains could arise from any component, their interaction, or a relaxed evaluation. Matched-budget ablations, staged integration, factorial designs where feasible, and frozen external evaluation are required. The unit of acceptance may still be the joint change set, but the evidence should not allocate causal credit unsupported by the experiment.

Research direction can itself become a recursive target. A later selector might use accumulated failure data to choose more diagnostic experiments or more productive engineering layers. To demonstrate this, it must

face new, predeclared problems and compete with the earlier selector or appropriate heuristics. Retrospective stories about why the chosen direction was sensible are not sufficient.

8.5 Evidence of loop closure

Positive bounded evidence must be retained alongside missing tests. RoboCat evaluates later adaptation, while GenSim evaluates a trained task generator [98, 104]. Neither result should be discarded because it falls short of the full-stack vision. Table 18 distinguishes these observed endpoints from the additional causal and resource controls needed for the broader claim.

A loop-closure claim consists of edges. For episode k, the record should establish: problem $\rightarrow$ candidate; candidate $\rightarrow$ executable artifact; artifact $\rightarrow$ observation; observation $\rightarrow$ analysis; analysis $\rightarrow$ accepted update; update $\rightarrow$ recipient; and, for recursion, update $\rightarrow$ changed later research followed by changed later research outcome. Each edge needs a producing actor, artifact identifier, recipient, and evidence location.

The representative systems in Table 18 have different recipients and endpoints. MEDAL++ links physical practice to policy improvement; FAST uses component measurements in design/model decisions; SOAR uses autonomously collected real-robot experience to improve a policy; Mother Robot realizes physical descendants; and egocentric self-modelling supports damage compensation [2, 5, 6, 39, 49]. These distinctions matter more than whether each workflow can be drawn as a circle. An artifact update can be established without a prospective comparison of its effect on later research.

Three evidence grades are useful. A *reported edge* is explicitly implemented and evaluated in one defined system version. An *inferred edge* is plausible from reported artifacts but lacks direct tracing or comparison. A *proposed edge* is part of an architecture or research agenda. Inferences and proposals are scientifically useful when labelled; they become misleading only when drawn with the same status as observed hand-offs.

The table therefore keeps reported downstream outcomes separate from proposed additional tests. RoboCat and GenSim supply bounded positive outcomes relevant to later adaptation and generator development; other rows establish different forms of robot improvement. An unreported comparison is not a failed experiment. These selected cases support a comparison of claim boundaries, not a prevalence estimate for the field; corpus-level coverage requires the screening and coding described in Section 2.

Table 18 pairs each cited system with our interpretation of its supported claim. It also prevents an invalid construction: taking design generation from one paper, automated manufacture from a second, and autonomous evaluation from a third and presenting the composite as an implemented RobotRSI system. Cross-paper composition belongs in the reference architecture and research agenda, where its missing interfaces remain visible.

8.6 Limits of sustained improvement

Sustained improvement faces limits that software-only accounts often hide. Physical experiments consume energy, specimens, robot life, and calendar time; manufacturing introduces drift and yield loss; sensors and objectives are imperfect; maintenance changes the platform; and deployment distributions move. As capability rises, evaluation may become harder because failures are rarer, interactions longer, and the system can exploit weaknesses in its own evaluator.

Monotonicity should not be assumed. A change can improve average performance while increasing tail risk, help new tasks while causing forgetting, or reduce present cost by creating maintenance debt. Version acceptance therefore needs a vector of improvement, safety, retention, and lifecycle criteria, plus a rule for incomparable trade-offs. Archives and rollback protect software, but physical alteration or wear may be only partly reversible.

Resource growth is another limit. If each round requires a larger model, richer simulator, more elaborate facility, or more human verification, task performance may rise while autonomous research efficiency falls. External compute and people are legitimate resources; hiding them is not. Net recursive benefit should include transferred initialization cost, fixed facility cost, marginal trial cost, and the burden of validating a more complex research system.

Table 18: Audit of self-improvement and recursive-feedback claims in representative systems. The normalized claim is our interpretation of the reported system boundary; "not evaluated" is not evidence of absence or a criticism of the system's stated objective.

System	Normalized supported claim	Executor / recipient	Returned artifact	Downstream outcome reported	Additional recursive test	Primary limitation for RSI interpretation
RoboCat [104]	Self-generated data improve a generalist; broader training supports later adaptation	Training/collection system → manipulation policy	Trajectories and generalist weights	Task-data ablation and later adaptation comparison	Separate data-source effects and total preparation costs on new tasks	One demonstrated iteration; adaptation comparison includes broader data changes
GenSim [98]	Generated task libraries support policy and task-generator training	LLM/simulation pipeline → policy and generator	Task code, demonstrations, generator weights	Held-out coding gains; physical policy tests after adaptation	Compare subsequent task creation at matched total budget with semantic validation	Coding pass rates miss instruction mismatch; human checks remain
Mother Robot [2]	Automated generational design, physical construction, and evaluation in a bounded morphology grammar	Parent construction/test apparatus → child population	Fitness and selected design lineage	Physical descendants with evolved morphology/behaviour	Compare research or production capability of successor-enabled versus original process on new goals	Producing apparatus and evolutionary mechanism are not themselves improved recipients
MEDAL++ [5]	Self-supervised real-robot practice improves task policies with learned rewards and undo behaviour	Learning/robot system → same manipulator	Interaction data, reward, policy	Improved performance on demonstrated manipulation tasks	Test whether the updated robot or reset policy reduces cost or improves discovery on new research tasks	Research question, learner, and acceptance process remain externally fixed
FAST [49]	Automated multi-fidelity component design uses physical tests to update design/model decisions	Laboratory platform → selected soft component/design process	Measurements and calibrated surrogate	High-throughput evaluated gripper designs	Freeze a new campaign and compare old versus updated research process at matched budget	Component-level loop; complete fabrication/integration and recursive research gain are not jointly tested
SOAR [6]	Autonomous real-robot experience collection and evaluation improves an instruction policy	Collection/training system → robot policy	Curated trajectories and trained policy	Improved instruction-following performance in reported settings	Compare old versus updated collector on held-out tasks, sites, and equal robot-hour budget	Policy benefit is reported; later research-collection benefit is not isolated
Egocentric self-model [39]	Physical observations update a robot self-model and support damage compensation	Robot/model pipeline → damaged robot	Predictive self-model and adapted behaviour	Recovery/compensation in demonstrated physical conditions	Test whether the updated model/behaviour improves later autonomous diagnosis under new faults	Bounded platform/conditions and no prospective research-capability comparison
AI Scientist [15]	An agentic system automates portions of computational ML research and paper production	Software research agents → ML artifacts, not a robot system	Code, experiment records, manuscript/review artifacts	End-to-end computational research demonstrations	Embed with robot tools and compare updated versus original research process on new physical engineering problems	No robot embodiment, physical experiment chain, or robot recipient in the reported system

Goodhart effects can occur at both loops. The inner loop exploits a task metric; the outer loop can exploit a research-productivity metric by producing many trivial candidates, avoiding hard problems, or redefining success. Protected evaluators, independent acceptance, surprise tasks, negative-result audits, and periodic objective review mitigate but do not eliminate this risk. The authority to modify evaluators and safety constraints should be separated from routine optimization.

Finally, recursion and acceleration are distinct. A system can improve its research procedure once with a small, bounded benefit; this is recursive feedback. Accelerating improvement would require evidence that the rate or efficiency of valid R&D itself increases across multiple comparable rounds. No such conclusion follows from a rising task-performance curve alone. Section 9 develops evaluation protocols capable of testing these claims.

9 Evaluation and Benchmarking

Evaluation is the mechanism that turns a circular architecture into a falsifiable claim. RobotRSI requires at least three separable questions: did the recipient robot improve, did the autonomous R&D process produce valid improvements efficiently, and did an update to that process cause better subsequent R&D? A single task-success number answers none of them in isolation. This section defines units, metrics, controls, and benchmark families for comparing heterogeneous systems without forcing them into an unjustified total score.

9.1 Claims, units, and reporting

The evaluated unit must match the claim. A *paper* may report several system versions; a *system* may contain multiple agents and robots; a *research episode* may include many trials; a *candidate* may be tested at several fidelities; and a *release* may update many recipients. Counting papers as successful R&D episodes or treating individual trials as independent systems inflates evidence.

We recommend a claim tuple

$$\Gamma = (X, A, T, R, C, B, H, E, K), \tag{10}$$

where X is executor, A autonomous activity, T research target, R recipient, C context, B resource budget, H human/external contribution, E evaluation protocol, and K number and lineage of rounds. Every quantitative result should be traceable to this tuple and a version manifest. "Autonomous" is not a property to attach once to the whole paper: problem definition may be human, candidate selection automatic, reset manual, and release jointly authorized.

Missing and negative states need distinct codes. *Not reported*, *not tested*, *not applicable*, *explicitly manual*, *failed*, and *outside the system boundary* are not zeros. Likewise, unsuccessful candidates are observations, not missing data. They affect yield, cost, safety, and the evidence available to future rounds.

Claims should be evaluated at their asserted scope. A component force test supports a component claim; a full robot task supports an integrated-capability claim under the tested conditions; a field release supports operation under its exposure distribution. Generalization requires declared held-out dimensions. Recursion requires later research tasks and a causal comparison, not merely several points on one training curve.

9.2 Capability and research metrics

Robot capability should be a vector, for example

$$\boldsymbol{v} = \big(v_{\text{task}}, v_{\text{safety}}, v_{\text{robust}}, v_{\text{retention}}, v_{\text{lifecycle}}\big), \tag{11}$$

reported with task distribution, experimental unit, repeats, uncertainty, and aggregation rule. Average success can conceal catastrophic failures; peak performance can conceal instability; performance on the modified task can conceal forgetting. Safety constraint violations and near misses should be reported separately from reward, not absorbed by an undocumented penalty.

Research effectiveness concerns the artifacts produced, not the fluency or activity of an agent. Useful endpoints include the probability that a proposed candidate passes independent validation; best validated improvement at a fixed budget; cost or time to a predeclared acceptance threshold; calibrated prediction error; and the number of distinct problems solved without changing the external protocol. A basic yield is

$$Y_{\text{valid}}(B) = \frac{N_{\text{independently accepted}}(B)}{N_{\text{executed candidates}}(B)}, \tag{12}$$

but numerator and denominator must define whether simulation screens, repeated trials, invalid builds, and safety aborts count. When no candidate succeeds, time-to-threshold is censored rather than assigned an arbitrary maximum.

Efficiency remains a resource vector. Compute-hours, energy, robot-hours, number of resets, specimens, material mass, machine occupancy, wall-clock delay, human time by role, and physical damage are not generally convertible to one currency. Pareto comparisons are appropriate only when protocols and resource definitions match. Facility initialization should be separated from marginal episode cost, while amortization assumptions remain visible.

Human independence should also remain disaggregated. Hours of expert problem formulation, teleoperation, fixture preparation, exception recovery, labeling, design repair, safety review, and release approval have different implications. A short high-skill intervention can determine the entire research direction; thousands of low-skill minutes can dominate scalability. Both frequency and time, plus trigger conditions, should be reported.

Recursive benefit is measured prospectively. For research processes R_0 and R_1, the primary estimand on held-out research problems is

$$\delta_{\text{rec}} = \mathbb{E}_{p \sim P_{\text{test}}} \left[J_E(R_1; p, B) - J_E(R_0; p, B) \right], \tag{13}$$

with identical budget and protected evaluator E. Confidence intervals should reflect variation across problems and independent system runs, not only repeated task trials for one selected artifact. Reporting results per problem prevents a few easy tasks from dominating.

9.3 Existing benchmarks and platforms

Current benchmarks cover useful slices of RobotRSI but not the entire claim. Meta-World contains 50 simulated manipulation tasks and explicitly separates multi-task training and held-out meta-learning tasks, making it relevant to fast skill acquisition and generalization [131]. ManiSkill2 spans 20 manipulation task families, more than 2,000 object models, multiple robot configurations, and rigid/soft-body simulated manipulation [132]. BEHAVIOR-1K defines 1,000 everyday activities over diverse simulated scenes and objects and includes an initial mobile-manipulator sim-to-real study [133]. These benchmarks evaluate behavior learning under defined simulators; they do not by themselves test autonomous selection of engineering problems, physical redesign, manufacturing, release, or improvement of the research method.

Domain research systems provide complementary platforms rather than standardized leaderboards. Safe Bayesian optimization on a quadruped exposes constrained physical tuning [86]. MEDAL++ exposes autonomous practice and reset burden [5]. FAST exposes multi-fidelity physical component design and laboratory throughput [49]. Mother Robot and ARE/RoboFab expose constructible morphology and facility-mediated succession [2, 112]. SOAR exposes large-scale autonomous real-robot experience collection [6]. Their metrics and boundaries differ, so cross-paper success rates should not be ranked.

Agentic engineering benchmarks are informative for the research-process layer. ControlAgent reports automated control-system design and an evaluation suite for controller reasoning/code [43]; AutoML-Agent evaluates full-pipeline machine-learning automation [83]; AI Scientist evaluates computational idea, code, experiment, manuscript, and review workflows [15]. They test parts of research work but omit the stateful physical robot experiment and integration boundary. RobotRSI benchmarking should connect such research tasks to physical evidence without erasing the difference between generated artifacts and deployed robots.

Table 19: Metric families and minimum reporting protocol for RobotRSI claims. Metrics are retained as a dashboard unless an application supplies a justified utility function.

Claim family	Suggested endpoint	Statistical unit and comparison	Required uncertainty	Major confounder
Recipient improvement	Task, safety, robustness, retention vector	Robot or independently built specimen across held-out conditions; predecessor baseline	Across instances, seeds, sites, and task conditions	Changed evaluator, easier exposure, cherry-picked best artifact
Research validity	Independently accepted candidate yield; calibrated predicted gain	Candidate nested in episode; external acceptance compared with stated prediction	Binomial or hierarchical interval; calibration curve	Repeated tests counted as new discoveries
Research efficiency	Best valid gain at fixed budget; censored time-to-threshold	Independent research problem under identical budget rules	Across problems and full pipeline repetitions	Hidden initialization, parallelism, or failed builds
Human dependence	Time and frequency by role and trigger	Intervention event and episode; compare matched authority boundaries	Distribution and tail recovery burden	Excluding setup, reset, expert direction, or review
Transfer/retention	Held-out recipient gain and old-capability change	Task/body/site/version boundary; from-scratch and no-transfer baselines	Across transfer targets, including negative transfer	Shared test data or incompatible hardware versions
Recursive benefit	δ_{rec} on new research problems	Research problem and independent process run; R_1 versus frozen R_0	Interval across problems and runs; failure probability	More budget, evaluator drift, leakage, self-selected easy tasks
Acceleration	Change in valid R&D rate across comparable rounds	Round within calibrated problem families; predeclared stopping	Trend uncertainty and sensitivity to difficulty model	Increasing resources or decreasing task difficulty

Evolution Gym adds a complementary benchmark in which the robot body itself can change [50]. It therefore tests a different engineering variable from policy-only manipulation suites. A RobotRSI benchmark can draw on both, but should also expose the research decisions and physical adoption steps required by its claim.

Table 20 is deliberately a capability map. A benchmark designed for manipulation performance should not be criticized for omitting fabrication; its role is to show which additional modules and controls a RobotRSI evaluation must supply. A complete benchmark can be compositional, provided version identities and information flow across modules are controlled.

9.4 Protocol for recursive improvement

Figure 12 specifies a prospective experiment. Start with a base research process R_0. During a development phase it may improve a robot, collect memory, and produce a candidate update u to its own model, tool, facility procedure, embodied capability, or research policy. Freeze both R_0 and the updated process R_1. Then assign held-out research problems drawn from a distribution not used to construct u, using matched starting robots, information, tools, wall-clock rules, and resource vectors. A protected evaluator judges the resulting artifacts.

At least four baselines are needed. The *no-update* baseline reuses R_0. The *more-budget* baseline gives R_0 the additional resources spent producing u, distinguishing recursion from scale. The *random or heuristic update* baseline tests whether any perturbation or ordinary engineering rule would help. The *human-designed update* baseline locates the autonomous system relative to available expert practice; it is not required to prove superhuman performance.

Ablations isolate the returned artifact. If R_1 changes a simulator, data memory, robot body, experiment tool, and research policy, construct feasible partial variants or staged introductions. When interactions make independent ablations invalid, report the joint change and test predicted interaction signatures. The evaluator and test distribution must not be revised by the process being scored. If evaluator improvement is itself the target, a higher-level protected audit is required.

Information leakage is especially dangerous. Held-out research problems should be generated or se-

Table 20: Existing benchmarks and research platforms relevant to parts of RobotRSI. Benchmark rows provide repeatable task suites; platform rows demonstrate a research loop but are not necessarily standardized comparative benchmarks.

Benchmark/platform	Primary evaluated unit	Task and embodiment scope	Mutable artifact / feedback	Useful RobotRSI evidence	Not established by this resource alone
Evolution Gym [50]	Body–controller search method	Simulated voxel robots in locomotion and manipulation tasks	Body and controller; task-return feedback	Joint design and control evaluation with common tasks	Physical manufacture, multi-site transfer, or a recursively improved research process
Meta-World [131]	Learning algorithm / policy	50 simulated manipulation tasks; multi-task and held-out meta-learning splits	Policy and learning method; simulator feedback	Cross-task acquisition and generalization under a shared suite	Physical experimentation, engineering integration, or recursive research update
ManiSkill2 [132]	Manipulation policy	20 task families, 2,000+ objects, several robot configurations, rigid/soft simulation	Policy/model; demonstrations and simulator	Object/task generalization across diverse manipulation settings	Autonomous problem selection, fabrication, deployment, or later R&D benefit
BEHAVIOR-1K [133]	Embodied-agent policy	1,000 simulated everyday activities, 50 scenes, 5,000+ objects; initial real transfer study	Policy/planner; rich simulated states	Long-horizon household activity coverage and simulator–reality study	General real-robot R&D, hardware change, or recursive feedback
ControlAgent / ControlEval [43]	Control-design agent and generated solution	Computational control-system design tasks	Controller/code through tools and evaluator feedback	Agentic controller engineering and executable artifact evaluation	Stateful robot experiments, integration, or robot recipient
AutoML-Agent [83]	Full-pipeline ML agent	Computational AutoML tasks	Data/model/code pipeline	Multi-agent orchestration and held-out ML task performance	Embodied/physical robot research chain
FAST [49]	Design campaign / physical component	Soft-gripper geometry with simulation and automated physical testing	Geometry, surrogate, calibration; force/grasp measurements	Physical design throughput, multi-fidelity evidence, model feedback	Standardized multi-site benchmark or deployed whole-robot recursion
ARE/RoboFab [22, 112]	Evolutionary facility / physical population	Modular constructible robots in structured facility	Morphology, control, population; manufacture/test feedback	Software–hardware orchestration and physical succession infrastructure	Unrestricted design, complete unattended lifecycle, or proven recursive R&D uplift
SOAR [6]	Data-collection and policy-learning system	Real tabletop manipulation across five environments	Experience dataset and instruction policy	Autonomous real-robot collection-to-policy update at scale	Research-method uplift on later held-out campaigns

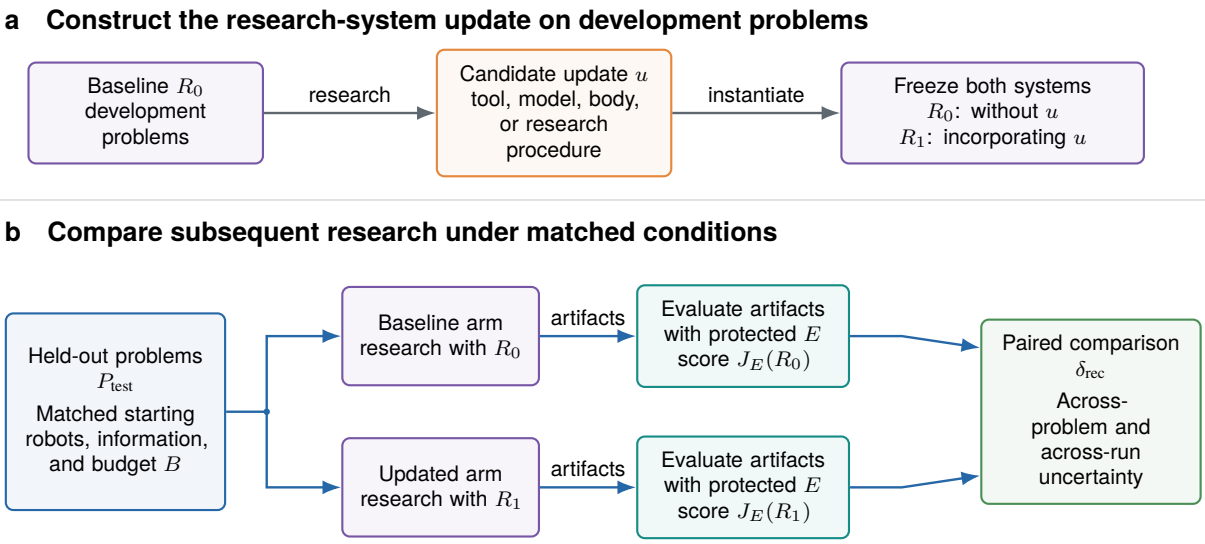

Figure 12: Proposed evaluation of recursive research benefit. Development produces an update; the evaluation phase compares research with and without that update. Both arms receive matched research problems and starting conditions. The same protected evaluator judges the artifacts produced by each arm, and the paired difference estimates later research benefit. Comparisons and uncertainty are computed over research problems and independent runs. This protocol is a proposed experiment, not a reported RobotRSI result.

questered before the recursive update; benchmark solutions, evaluator code, and acceptance thresholds should be access-controlled where appropriate. Foundation-model pretraining and web retrieval may contain task information and should be declared, not treated as automatically disqualifying. The comparison asks what information and tools each condition had, not whether it learned in an artificial vacuum.

Repeated rounds require stopping rules. Predeclare maximum rounds, total and per-round budgets, failure and safety termination, and whether unused resources carry forward. Report every release and rollback. A claim of acceleration requires comparable problems or a calibrated difficulty model across rounds; progressively easier self-selected problems cannot support it.

9.5 Resources and reproducibility

Reproduction of autonomous R&D has two levels. *Outcome reproduction* recreates the accepted robot artifact and its evaluation. *process reproduction* reruns the research system and characterizes the distribution of proposed and accepted artifacts. Stochastic search can reproduce an outcome distribution without producing the same design. Both are scientifically meaningful when the claim and seed policy are explicit.

The release package should include source code and prompts, model and dependency identifiers, container or environment specification, CAD and bills of materials, fabrication settings, firmware, calibration, raw and processed data, invalid/negative trials, experiment orchestration, safety and reset procedures, evaluator, random seeds, and the version graph. Proprietary components can be described by interfaces and hashes when redistribution is impossible, but the limitation should remain visible.

Physical reproducibility requires tolerances. A CAD file does not encode material batch, printer state, assembly force, wiring, wear, temperature, or calibration. Report measured as-built properties and specimen identity. For fleet data, report hardware revisions, exposure, site, and maintenance. For human involvement, provide role, qualification, time, decision, and trigger rather than the phrase “minor assistance.”

Statistical analysis should follow the hierarchy of the experiment. Trials are nested within candidates, candidates within research episodes, and episodes within problems or system instances. Selecting the best

Table 21: Resource and reproducibility ledger. Raw units and missing values should be published; monetary conversions and amortized summaries are optional derived views.

Resource/artifact	Minimum quantity or identifier	Boundary detail	Reproduction evidence
Compute and services	Device/model/API version; compute-hours; energy where available	Training, search, simulation, inference, failed jobs; concurrency	Environment lockfile/container, seeds, calls, code and logs
Robot and facility	Robot-hours, trials, resets, machine occupancy, charging	Setup versus active trial; remote services; operator presence	Hardware revision, orchestration, fixture and reset protocol
Manufacturing	Build count, material, machine time, yield, rework, discarded specimens	Automated versus manual fabrication/assembly/inspection	CAD/BOM, process settings, tolerances, as-built measurements
Human contribution	Minutes/hours by role; count and trigger of interventions	Problem definition, preparation, recovery, labeling, analysis, acceptance	Decision log, qualification, information visible to human
Failures and risk	Invalid trials/builds, aborts, damage, near misses, recovery time	Inclusion rule and severity; whether cost is censored	Raw event/telemetry links, incident and rollback records
Data and knowledge	Dataset/log/checkpoint hashes, provenance, licenses	Pretraining, retrieval, fleet/site exposure, negative results	Raw/processed mapping, schema, lineage and access date
System release	Code, model, firmware, CAD, calibration and evaluator versions	Exact executor, target, recipient, dependencies, authority	Signed manifest, acceptance suite, release/rollback trace

candidate and then computing error bars only over its task trials ignores research-selection variance. Bootstrap or hierarchical models may be appropriate, but the sampling unit and selection procedure matter more than a particular statistical technique.

9.6 Benchmark proposals

We propose a tiered benchmark family rather than one monolithic arena.

B1: bounded artifact research. The system receives a new requirement for one mutable object—controller, perception module, sensor layout, gripper geometry, or energy schedule—plus fixed tools and a budget. It must formulate candidates, execute permitted digital and/or physical tests, and return a versioned artifact with evidence. Hidden test conditions measure validated improvement, safety, retention, and resource use.

B2: diagnosis and layer selection. The robot exhibits a held-out failure whose cause may lie in hardware, software, calibration, or environment. The system chooses diagnostic experiments and an intervention layer. Matched fault families and protected ground truth evaluate diagnostic information gain, unnecessary modifications, and accepted system recovery. This tests research direction rather than search inside a disclosed parameter space.

B3: cross-stack change set. The task requires a coupled update such as sensor–model–compute, body–controller, or tool–skill–calibration. Candidates pass simulation, component tests, integration, and retention gates. Interfaces and budgets are standardized while designs remain open. The main endpoints are validated system Pareto improvement and attribution quality.

B4: facility-mediated physical succession. A bounded modular or soft-robot grammar supports automated manufacture/assembly, commissioning, and testing. Invalid builds, reset, tool failure, and human recovery count. The recipient is a newly realized component or robot, and the benchmark evaluates both artifact quality and closed hand-offs.

B5: recursive R&D transfer. After completing development problems, the system may return exactly one research-capability update. Frozen R_0 and R_1 then address sequestered problems under equal budgets following Figure 12. Success requires positive recursive uplift with uncertainty, no unacceptable retention/safety loss, and a causal ablation of the returned update.

Each tier can be instantiated for manipulation, locomotion, aerial systems, soft robotics, or field inspection without assuming identical physics or costs. A common claim tuple and resource ledger allow structural comparison, while domain metrics remain local. Multi-site replication should be introduced after within-site repeatability, since transport of fixtures, materials, and tacit procedures is itself an important test.

Tables 19 and 21 specify the reporting protocol: recipient capability, research effectiveness, resource use, and across-round evidence remain separate endpoints. Benchmark releases should publish task-level results, sampling units, uncertainty, failed and invalid trials, human and external contributions, and versioned artifacts. These records support appropriate Pareto comparisons without concealing incomparable resources in an undocumented total score.

Evaluation should make ambitious RobotRSI claims harder to state casually and easier to establish incrementally. A system can earn strong evidence for bounded physical research, cross-stack integration, or research-method transfer without claiming the terminal vision. The next section turns these measurable gaps into a research agenda.

10 Perspectives and Research Agenda

The central opportunity in RobotRSI is to make robot development cumulative across engineering layers and across generations. The studies reviewed here supply concrete starting points: physical morphology search, model-based damage adaptation, autonomous reward and policy learning, multi-fidelity component experiments, software–hardware co-design, and agentic computational research. Their combination requires research on interfaces and evidence as much as on stronger candidate generators. This section develops a testable agenda from those connections. The proposals are hypotheses motivated by the surveyed systems; they are not claims that the suggested experiments have already succeeded or that no related work exists.

10.1 Research direction and cross-domain reasoning

A robot that can optimize a specified controller is already useful. A robot that can determine whether its performance problem is caused by control, sensing, embodiment, or calibration addresses a different and potentially more consequential capability. The next research problem should therefore be selected from a model of system-level bottlenecks and uncertainty, grounded in operating failures and engineering requirements.

One tractable starting point is a library of controlled faults spanning several layers: shifted camera extrinsics, delayed observations, actuator friction, payload mismatch, contact-model error, and task-distribution shift. The autonomous researcher receives telemetry and tools but not the fault identity. It chooses interventions that discriminate among explanations, then proposes a bounded change set. Evaluation measures diagnostic accuracy, information gained per experiment, unnecessary modifications, and accepted recovery. A layer-blind optimizer, fixed diagnostic tree, and expert engineer provide complementary baselines.

The main hypothesis is that a structured causal representation of requirements, interfaces, and failure mechanisms can reduce the research budget relative to indiscriminate search. The hypothesis is refuted if diagnostic overhead exceeds the saved experiments, if performance depends on leaked fault labels, or if the method selects useful questions only inside its development library. CODEI and body–controller co-design illustrate relevant coupling structures, while RADIUM demonstrates that simulator-driven failure search can feed bounded policy or design repair [60, 73, 89]. None establishes autonomous diagnosis across unfamiliar engineering layers; RADIUM begins with a specified system, simulator, parameterization, and failure criterion.

Research direction should also represent uncertainty about the objective. Operators may ask for longer endurance, smoother motion, better contact quality, or lower maintenance burden. These requests can conflict and can change when a candidate is physically realized. A productive system exposes the trade-off and requests a decision at the appropriate authority boundary, while continuing experiments whose value is robust across the unresolved preference. Learning operator preferences is a possible research object, but the system's own metric cannot become the sole justification for redefining success.

10.2 Robots and infrastructure designed for modification

Physical self-improvement will initially be constrained by what can be accessed, measured, replaced, assembled, and validated. Evolvable hardware and RoboFab demonstrate the importance of component banks, interfaces, power, and fabrication rules [22, 112]. This suggests a design objective for *future modification cost*, alongside present task performance.

A minimal experiment compares otherwise matched robot modules with different service and measurement affordances: keyed connectors, machine-readable identity, accessible calibration targets, self-test motions, standard tool interfaces, or replaceable sensor mounts. Over a sequence of predeclared changes, measure change time, assembly success, calibration error, human recovery, and retained task capability. The result may favor a less modular design if added mass, compliance, or interface failures outweigh the R&D savings. Modularity is a testable trade-off, not a universal optimum.

Engineering interfaces should carry semantics as well as geometry or data types. A sensor driver needs units, timing, frames, calibration dependencies, and failure behaviour; a CAD artifact needs material, tolerances, assembly and inspection assumptions; a model needs operating domain, training lineage, and uncertainty. Machine-readable contracts can let conventional optimizers, language-model agents, manufacturing tools, and validators cooperate. A useful near-term deliverable is a small interoperable tool suite supporting one complete change, including rejected candidates and rollback.

The system boundary can grow incrementally. A robot may first research software using an external test cell, later learn to reset that cell, and eventually select or fabricate its own tools. Each addition should have a measurable purpose and a declared resource cost. A large automated facility is justified by the research it enables, rather than by the number of operations it can perform unattended.

10.3 Autonomous physical research

Physical research autonomy depends on trial preparation, execution, measurement, reset, and exception handling. These activities determine both throughput and bias: an agent that experiments only on easy-to-reset objects may produce an apparently strong policy with narrow coverage. MEDAL++ shows the value of learned reversibility, while FAST shows how an engineered apparatus can support repeated component measurements [5, 49]. Combining these lessons motivates experiments on the research value of improved reset and instrumentation.

Consider a robot gripper-design campaign with fixed geometry bounds and physical test budget. One condition uses a manually maintained reset protocol, another a fixed automatic fixture, and another a learned reset procedure. All candidates are validated on a common independent rig. The primary endpoint is accepted-design gain per total facility-hour, including failed reset, cleaning, reassembly, and human intervention. A downstream campaign with new objects tests whether the improved procedure transfers. This directly examines a recursive pathway from robot skill to better experimentation.

Measurement quality deserves equal attention. An autonomous campaign can optimize the artefacts of a sensor, fixture, or analysis script. Repeated standards, blind reference specimens, uncertainty calibration, and disagreement between independent instruments can detect this. Instrument and evaluator improvements should be validated outside the same optimization loop that selected them. Research on active metrology may be especially valuable where one additional measurement can reject an entire family of invalid designs.

Physical exception recovery is also an engineering target. Useful benchmarks include a misplaced specimen, failed connector, sensor dropout, charging failure, clogged printer, or non-reversible object interaction. Recovery quality should include correct refusal or escalation when the available evidence cannot establish a safe state. Long unattended duration alone is a weak objective if the system silently stops collecting informative data or excludes difficult experiments.

10.4 Knowledge accumulation and cross-robot development

A fleet can accumulate more observations than a single robot, but observations do not automatically become reusable engineering knowledge. Open X-Embodiment shows the scale of cross-robot data sharing, while

Table 22: Failure mechanisms and proposed discriminating tests. Examples are analytical risks motivated by the indicated research settings, not measured failure frequencies.

Location	Observable symptom	Competing explanation	Proposed distinguishing test and context
Objective/reward	Search reward rises; protected task score falls	Reward shortcut versus invalid external test	Evaluate unseen trajectories with independent task criteria; reward-design setting [95]
Candidate/realization	High simulated score; repeated build failure	Infeasible geometry versus process variation	Build reference geometry; inspect as-built tolerances; evolutionary fabrication [22]
Simulator/model	Ranking reverses on physical tests	Reality gap versus measurement drift	Test blind reference specimens and model residuals; component design [49]
Experiment/reset	Fewer useful trials despite unattended runtime	Scene drift versus policy degradation	Alternate reference policy and fixed reset; autonomous practice [5]
Evidence/selection	Reported gain fails independent replication	Selection bias, evaluator noise, or distribution shift	Rerun entire search across held-out problems and seeds; agentic research [15]
Integration	Component improves; robot degrades	Interface, timing, calibration, or interaction error	Staged substitution and end-to-end traces; co-design [73]
Inheritance	New task improves; older tasks regress	Negative transfer versus changed hardware/site	Version-controlled retention and transfer ablations; shared data [106]
Outer research loop	Apparent productivity rises each round	Better R&D versus easier tasks or more resources	Sequestered problems and frozen-process comparison; self-modifying agents [42]

autonomous collection systems such as SOAR supply routes for extending experience with less repeated human demonstration [6, 106]. The next step is to preserve why an experiment was run, what it ruled out, and which versions can use its conclusion.

Three transferable knowledge units are promising: failure explanations with discriminating tests; design surrogates with declared validity regions; and research procedures with measured cost and reliability. A cross-robot experiment should compare raw shared data, unstructured textual memory, structured evidence records, and no transfer. Held-out embodiment and site combinations reveal whether the benefit reflects reusable engineering relationships or a shared visual/task distribution.

Negative evidence is particularly valuable but difficult to transfer. A controller failure can arise from a bad design, a faulty specimen, poor calibration, insufficient inner-loop training, or a broken evaluator. The memory must record the uncertainty of the explanation. Treating every failure as a universal prohibition can prune viable designs; treating it as uninformative wastes resources and can repeat damage. A useful research objective is calibrated reuse of negative results under changing compatibility assumptions.

Knowledge accumulation also needs a mechanism for correction. An inherited result may be invalidated by a later instrument calibration, software bug, or new body revision. A dependency graph should identify affected claims, recipients, and experiments for reassessment. Versioned retraction and revalidation are as important as knowledge addition for sustained RobotRSI.

10.5 Reliable evidence and long-horizon evaluation

As a research system becomes more capable, the evaluation boundary must remain informative. A planner may exploit reward shortcuts; a design optimizer may exploit simulator error; an agent may select easier questions; and a fleet update may change its own exposure distribution. These are possible mechanisms, not a causal sequence. Figure 13 develops one concrete diagnostic example: the same two candidates reverse rank between simulation and physical tests. The crossing identity lines visualize the mismatch without assigning a cause. Revising the simulator is only one possible response. Reference specimens can expose instrument drift, as-built measurements can expose realization errors, and blinded repeats can test whether the ranking is stable. The appropriate intervention depends on evidence that discriminates among these explanations.

Diagnostic tests need not identify a unique cause. Model error and fabrication variation can coexist, and a stable reference specimen does not validate every measurement channel. The proposed loop therefore retains unresolved alternatives and selects further informative tests when justified by cost and risk. It does not turn a plausible explanation directly into an authorized repair. Figure 13 shows identity, alternatives, and sequential evidence use within one case; Table 22 retains the detailed comparisons across the broader engineering stack instead of repeating them inside the figure.

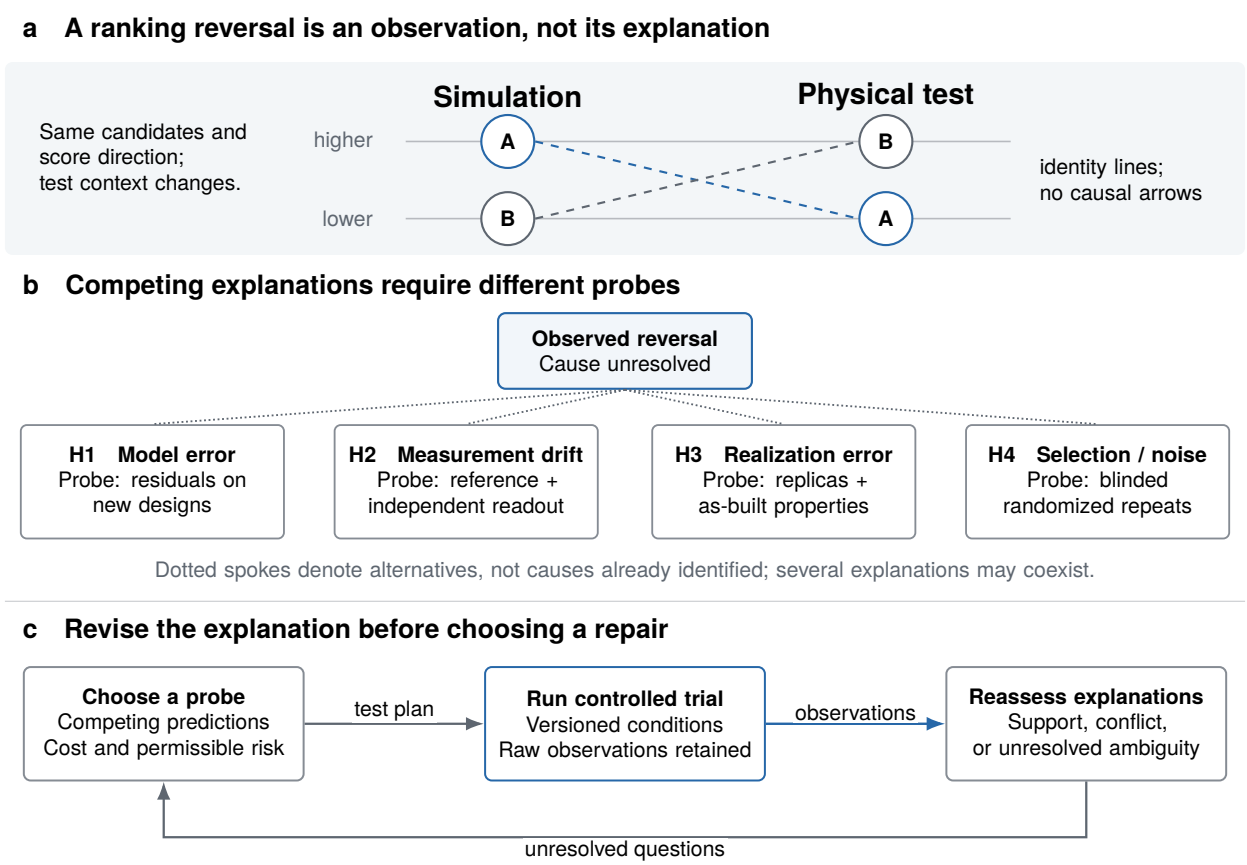

Figure 13: Diagnosing a sim-to-real ranking reversal without assuming its cause. Panel (a) is a hypothetical ordinal example; dashed lines preserve candidate identity across contexts and have no arrowheads. Panel (b) associates the same observation with four competing explanations and one discriminating probe for each. The dotted spokes are non-causal relations, and the list is not exhaustive. Panel (c) is the only process graph: a controlled test returns observations and, when ambiguity remains, unresolved questions motivate another test. None of these observations directly authorizes a component repair. Table 22 extends the hypothesis–test pairing across the engineering stack.

Long-horizon evaluation should follow a sequence of accepted and rejected versions with periodic protected challenges. One useful protocol interleaves ordinary research tasks, deliberately difficult tasks, instrument checks, and retention probes. Researchers should record both realized utility and the system's predicted gain before each experiment. This exposes overconfidence, easy-task selection, and degradation of research memory even when average task reward rises.

The protected evaluator must have a defined lifecycle. Keeping it fixed supports comparison, but eventually reveals only performance against a stale distribution. Updating it supports relevance, but changes the meaning of progress. A practical solution is to retain an immutable comparison suite and introduce separately versioned challenge suites, reporting both. Evaluator changes require independent review and cannot be retroactively applied only to favorable rounds.

Computational systems that edit their own agents provide relevant experimental ideas. DGM and Hyperagents investigate self-modification and meta-level improvement [41, 42]. In robots, the same questions must be tested with material cost, instrument error, irreversible operations, and system commissioning included. A successful result in that setting would establish a more specific and useful contribution than a broad claim of autonomous scientific intelligence.

10.6 A testable research roadmap

The agenda is organized around testable research questions rather than a single development sequence. Callable tools, physical experimentation, integration, evidence reuse, and research-policy revision can be

investigated in parallel. Their dependencies must be specified for the particular robot and experiment: a software-only study need not await autonomous fabrication, and an accepted robot update need not improve subsequent research. Table 23 makes the proposed intervention, comparison, resource boundary, and possible disconfirmation explicit for each question.

Table 23 gives six minimal experiments. Each declares a mutable artifact, comparison, budget boundary, success criterion, and a result that would weaken the hypothesis. They are intended as reusable patterns for different robot types, not a proposed universal benchmark ranking. The scale should be small enough to repeat across laboratories: a bounded gripper family, a modular sensor mount, a quadruped controller interface, or a controlled mobile-robot fault set can supply strong evidence before complete autonomous factories are available.

The progression should be judged by closed, validated hand-offs. An initial milestone is one accepted robot update with all human actions counted. A second is a coupled hardware–software update with compatible calibration and retained capabilities. A third is a reusable research artifact that improves a new campaign under matched budget. A fourth is replication on a different robot or site. These are evidence milestones rather than an obligatory order for every project.

Community infrastructure can make such work easier to compare. Shared artifact schemas, reference fixtures, open negative-result logs, standardized reset events, and versioned evaluation tasks lower the burden of process reproduction. Their utility should itself be measured: a schema that researchers cannot populate reliably or a fixture that excludes most relevant bodies can undermine the intended comparison. Initial standards should therefore emerge from concrete multi-system trials.

10.7 Trustworthy and controllable development

The freedom to study an engineering object should be separated from the authority to change a deployed system. RobotRSI benefits from a declared modification envelope: accessible artifacts, permissible experiments, resource limits, acceptance authority, and recovery options. These bounds make the research objective concrete and the resulting evidence interpretable. They need not prevent useful autonomous decisions within the envelope.

The research process can propose changes to safety monitors, evaluators, or release procedures, but such proposals require a separate validation route. A generator cannot establish its own permission by writing a persuasive justification, and improved task reward cannot justify bypassing an independent acceptance gate. The appropriate division of responsibility depends on the context: a simulated controller study and a physical intervention around people have different consequences and test requirements.

Security is part of the engineering interface. Research agents ingest papers, repositories, logs, sensor streams, and generated code; these inputs may be incorrect or adversarial. Tool permissions, executable isolation, provenance, integrity checks, and bounded physical commands help prevent an external artifact from becoming an unauthorized system action. A useful test measures whether the researcher identifies a corrupted measurement or malicious instruction without abandoning valid neighboring evidence.

Controllability also requires resource and state observability. Operators need to know which robot version is active, what experiment is running, what changes are pending, and how to stop or recover it. A stopping mechanism must remain effective after software updates and physical reconfiguration. Rollback is meaningful only if the predecessor state is reproducible; irreversible wear or manufacturing changes should have a recovery plan expressed in physical terms.

Finally, data, designs, model weights, and contributed research knowledge carry provenance and licensing constraints. Recording origin and permitted reuse is part of inheritance. These requirements can be implemented in the same artifact manifest used for technical compatibility, helping the system distinguish a valid engineering possibility from a permitted deployment action.

The resulting perspective is deliberately ambitious about scope and precise about evidence. Every layer of robot engineering can be a research target, but a convincing contribution need not modify every layer. It should demonstrate a consequential autonomous decision, a traceable artifact, a validated recipient effect, and, when recursion is claimed, a measured improvement in later research. Table 24 ties the main synthesis

Table 23: Proposed minimal experiments for RobotRSI. Budgets are experimental controls to specify before execution; no numerical outcome is implied.

Question / starting evidence	Mutable artifact and minimal system	Comparison	Budget and withheld condition	Success endpoint	Disconfirming outcome
Does diagnosis improve layer selection? Co-design [73]	Research selector on a mobile robot with controlled hardware/software faults	Causal selector, fixed diagnostic tree, layer-blind search	Equal trials, compute, human support; hidden fault combinations	More accepted recovery per total budget with correct fault attribution	No advantage after diagnosis cost or on unseen combinations
Does modifiability repay its cost? Evolvable hardware [22]	Sensor/tool interfaces on matched modular robot variants	Serviceable interface versus conventional integration	Same sequence of changes; include mass and task penalties	Lower lifecycle change cost while retaining task capability	Assembly overhead, added mass, or new failure modes erase savings
Does a better reset skill improve later R&D? MEDAL++ [5]	Reset procedure in a gripper-test or manipulation cell	Manual, fixed automatic, and learned reset	Equal total facility-hours including recovery; new objects	Better validated design/policy gain in subsequent campaign	Higher unattended time without better valid research yield
Does failure knowledge transfer? Shared robot data [106]	Evidence memory shared across two robot/site families	Structured failure evidence, raw data, text memory, no transfer	Equal prior access and budget; held-out body/site pair	Lower research cost with calibrated negative-transfer risk	Gains require identical embodiments or concealed extra data
Does physical feedback improve a research tool? FAST [49]	Simulator/surrogate updated from a first component campaign	Updated tool versus frozen tool and additional-sampling baseline	Equal later physical tests; new design subregion	Higher independently validated later-campaign gain	Prediction error falls but downstream design quality does not improve
Does a research-policy update generalize? Hyperagents [42]	Agent policy for bounded robot engineering tools	Frozen predecessor, updated policy, heuristic/expert update	Equal resources; sequestered requirements and protected evaluator	Positive recursive uplift, retained capabilities, attributable artifact effect	Gains vanish under matched budgets, leakage controls, or independent evaluation

Table 24: Main synthesis statements and their evidence boundaries. Conclusions concern the cited mechanisms and do not estimate prevalence across an exhaustive corpus.

Synthesis	Representative support	Supported scope	Remaining evidence obligation
Robot improvement has many mutable targets	Morphology, models, policy, rewards, and grippers [2, 39, 49, 95]	Existence of autonomous decisions across several engineering objects	Exhaustive coverage and measured coupling across the full stack
Physical experimentation can enter an automated loop	Construction, test rigs, and task/undo practice [2, 5, 49]	Bounded physical execution with declared facilities	Full preparation, recovery, manufacturing, and integration autonomy
Cross-stack interfaces determine recipient validity	Co-design and evolvable production [73, 112]	Concrete coupled constraints and facility architecture	Sustained, version-controlled whole-system releases and retention
Knowledge can improve capabilities across tasks/robots	Shared datasets and autonomous experience [6, 106]	Operational data/model learning and transfer	Prospective transfer of engineering and research capability
Recursive process modification has computational precedents	Self-modifying coding agents and editable meta procedures [41, 42]	Computational agent improvement in reported domains	Equivalent causal evidence through stateful robot experiments and deployment
General-purpose and specialized robots require context-specific loops	Manipulation, locomotion, soft components, and UAV co-design [49, 86, 123]	Different feasible update, test, and infrastructure boundaries	Cross-context replication under comparable resource accounting

statements to their support and remaining limits.

11 Conclusion

RobotRSI frames the long-term development of robots as autonomous research across the complete engineering stack, with validated results returning to a robot, a collective, a research facility, or a successor version. Its scope includes the physical body, intelligence and software, system integration, experimental infrastructure, manufacturing, maintenance, and the research process. Full-stack scope means that any of these can be a target; it does not require every iteration to redesign them all.

The survey's seven questions lead to a connected account. Adaptation, autonomous R&D, system improvement, and recursive benefit require distinct evidence (RQ1). Autonomous decisions already appear in morphology search, component experimentation, perception and self-model updates, reward and controller design, data collection, and computational co-design (RQ2). AI scientists, robotic instruments, conventional optimizers, and human expertise can share the research workflow, so decision and execution roles must be reported separately (RQ3). Task breadth, embodiment, scale, environment, organization, and infrastructure determine the feasible experiment and update boundary (RQ4).

The remaining connections concern integration and accumulation. A candidate becomes a system improvement through an identified recipient, compatible configuration, calibration, acceptance, and post-update evaluation (RQ5). Evaluation must report task capability, research validity, resources, human contribution, retention, and prospective recursive uplift under matched conditions (RQ6). The most actionable research opportunities connect diagnosis to layer selection, design to autonomous physical realization, experience to reusable engineering knowledge, and an accepted update to stronger subsequent research (RQ7).

Representative systems demonstrate meaningful local and physical loops. Their results should retain their own boundaries: a better policy, a calibrated simulator, a fabricated morphology, and a self-modifying computational agent are different achievements. The evidence considered here does not establish one continuously operating system that autonomously researches, realizes, validates, and inherits improvements across the full robot stack. Nor does it justify predicting inevitable acceleration. It does support a concrete experimental programme in which these connections can be built and tested individually and in combination.

The defining milestone for recursive RobotRSI is therefore an attributable change in later research capability. A robot that resets experiments more reliably, a facility that fabricates a broader valid design family, or a research agent that chooses better diagnostic experiments can all contribute. Their value is established when a subsequent, independently evaluated campaign produces stronger validated outcomes at a compara-

ble resource and human-support budget. That criterion turns the vision of continually evolving robots into an engineering and scientific question that can be answered with artifacts, experiments, and reproducible evidence.

References

[1] Hod Lipson and Jordan B. Pollack. "Automatic Design and Manufacture of Robotic Lifeforms". In: *Nature* 406 (2000), pp. 974–978. DOI: 10.1038/35023115.

[2] Luzius Brodbeck, Simon Hauser, and Fumiya Iida. "Morphological Evolution of Physical Robots through Model-Free Phenotype Development". In: *PLOS ONE* 10.6 (2015), e0128444. DOI: 10.1371/journal.pone.0128444.

[3] Josh Bongard, Victor Zykov, and Hod Lipson. "Resilient Machines Through Continuous Self-Modeling". In: *Science* 314.5802 (2006), pp. 1118–1121. DOI: 10.1126/science.1133687.

[4] David Matthews, Andrew Spielberg, Daniela Rus, Sam Kriegman, and Josh Bongard. "Efficient Automatic Design of Robots". In: *Proceedings of the National Academy of Sciences* 120.41 (2023), e2305180120. DOI: 10.1073/pnas.2305180120.

[5] Archit Sharma, Ahmed M. Ahmed, Rehaan Ahmad, and Chelsea Finn. "Self-Improving Robots: End-to-End Autonomous Visuomotor Reinforcement Learning". In: *Proceedings of the 7th Conference on Robot Learning*. Vol. 229. Proceedings of Machine Learning Research. 2023, pp. 3292–3308. URL: https://proceedings.mlr.press/v229/sharma23b.html.

[6] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Rich Walke, Oier Mees, and Sergey Levine. "Autonomous Improvement of Instruction Following Skills via Foundation Models". In: *Proceedings of the 8th Conference on Robot Learning*. Vol. 270. Proceedings of Machine Learning Research. 2025, pp. 4805–4825. URL: https://proceedings.mlr.press/v270/zhou25b.html.

[7] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Philip G. K. Reiser, Christopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. "Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist". In: *Nature* 427 (2004), pp. 247–252. DOI: 10.1038/nature02236.

[8] Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, Maria Liakata, Magdalena Markham, Pinar Pir, Larisa N. Soldatova, Andrew Sparkes, Kenneth E. Whelan, and Amanda Clare. "The Automation of Science". In: *Science* 324.5923 (2009), pp. 85–89. DOI: 10.1126/science.1165620.

[9] Andrew Sparkes et al. "Towards Robot Scientists for Autonomous Scientific Discovery". In: *Automated Experimentation* 2 (2010), p. 1. DOI: 10.1186/1759-4499-2-1.

[10] Kevin Williams, Elizabeth Bilsland, Andrew Sparkes, Wayne Aubrey, Michael Young, Larisa N. Soldatova, Kurt De Grave, Jan Ramon, Michaela de Clare, Worachart Sirawaraporn, Stephen G. Oliver, and Ross D. King. "Cheaper Faster Drug Development Validated by the Repositioning of Drugs against Neglected Tropical Diseases". In: *Journal of the Royal Society Interface* 12.104, 20141289 (2015). DOI: 10.1098/rsif.2014.1289.

[11] Benjamin P. MacLeod, Fraser G. L. Parlane, Thomas D. Morrissey, Florian Häse, Loïc M. Roch, Kevan E. Dettelbach, Raphaell Moreira, Lars P. E. Yunker, Michael B. Rooney, Joseph R. Deeth, Veronica Lai, Gordon J. Ng, Henry Situ, Ray H. Zhang, Michael S. Elliott, Ted H. Haley, David J. Dvorak, Alán Aspuru-Guzik, Jason E. Hein, and Curtis P. Berlinguette. "Self-Driving Laboratory for Accelerated Discovery of Thin-Film Materials". In: *Science Advances* 6.20, eaaz8867 (2020). DOI: 10.1126/sciadv.aaz8867.

[12] Benjamin Burger, Phillip M. Maffettone, Vladimir V. Gusev, Catherine M. Aitchison, Yang Bai, Xiaoyan Wang, Xiaobo Li, Ben M. Alston, Buyi Li, Rob Clowes, Nicola Rankin, Brandon Harris, Reiner Sebastian Sprick, and Andrew I. Cooper. "A Mobile Robotic Chemist". In: *Nature* 583 (2020), pp. 237–241. DOI: 10.1038/s41586-020-2442-2.

[13] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. "Autonomous Chemical Research with Large Language Models". In: *Nature* 624 (2023), pp. 570–578. DOI: 10.1038/s41586-023-06792-0.

[14] Nathan J. Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E. Kumar, Tanjin He, David Milsted, Matthew J. McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, Haegyeom Kim, Anubhav Jain, Christopher J. Bartel, Kristin A. Persson, Yan Zeng, and Gerbrand Ceder. "An Autonomous Laboratory for the Accelerated Synthesis of Inorganic Materials". In: *Nature* 624 (2023), pp. 86–91. DOI: 10.1038/s41586-023-06734-w.

[15] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. *The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery*. 2024. arXiv: 2408.06292.

[16] Chris Lu, Cong Lu, Robert Tjarko Lange, et al. "Towards End-to-End Automation of AI Research". In: *Nature* 651 (2026), pp. 914–919. DOI: 10.1038/s41586-026-10265-5.

[17] Mourad Gridach, Jay Nanavati, Khaldoun Zine El Abidine, Lenon Mendes, and Christina Mack. "Agentic AI for Scientific Discovery: A Survey of Progress, Challenges, and Future Directions". In: *ICLR 2025 Workshop on Agentic AI*. 2025. arXiv: 2503.08979.

[18] Shuo Ren, Can Xie, Pu Jian, Zhenjiang Ren, Chunlin Leng, and Jiajun Zhang. *Towards Scientific Intelligence: A Survey of LLM-based Scientific Agents*. 2025. arXiv: 2503.24047 [cs.AI].

[19] Tianshi Zheng, Zheye Deng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Zihao Wang, and Yangqiu Song. *From Automation to Autonomy: A Survey on Large Language Models in Scientific Discovery*. 2025. arXiv: 2505.13259 [cs.AI].

[20] Qiujie Xie, Yixuan Weng, Minjun Zhu, Fuchen Shen, Shulin Huang, Zhen Lin, Jiahui Zhou, Zilan Mao, Zijie Yang, Linyi Yang, Jian Wu, and Yue Zhang. *How Far Are AI Scientists from Changing the World?* 2025. arXiv: 2507.23276 [cs.AI].

[21] Jiaqi Wei, Yuejin Yang, Xiang Zhang, Yuhan Chen, Xiang Zhuang, Zhangyang Gao, Dongzhan Zhou, Guangshuai Wang, Zhiqiang Gao, Juntai Cao, Zijie Qiu, Ming Hu, Chenglong Ma, Shixiang Tang, Junjun He, Chunfeng Song, Xuming He, Qiang Zhang, Chenyu You, Shuangjia Zheng, Ning Ding, Wanli Ouyang, Nanqing Dong, Yu Cheng, Siqi Sun, Lei Bai, and Bowen Zhou. *From AI for Science to Agentic Science: A Survey on Autonomous Scientific Discovery*. 2025. arXiv: 2508.14111 [cs.AI].

[22] Mike Bilby, Edgar Buchanan, Léni K. Le Goff, Emma Hart, Agoston E. Eiben, Matteo De Carlo, Alan F. Winfield, Matthew F. Hale, Robert Woolley, Jon Timmis, and Andy M. Tyrrell. "Practical Hardware for Evolvable Robots". In: *Frontiers in Robotics and AI* 10 (2023), p. 1206055. DOI: 10.3389/frobt.2023.1206055.

[23] David Howard, Agoston E. Eiben, Danielle Frances Kennedy, Jean-Baptiste Mouret, Philip Valencia, and Dave Winkler. "Evolving Embodied Intelligence from Materials to Machines". In: *Nature Machine Intelligence* 1 (2019), pp. 12–19. DOI: 10.1038/s42256-018-0009-9.

[24] Robert Baines, Frank Fish, Josh Bongard, and Rebecca Kramer-Bottiglio. "Robots that Evolve on Demand". In: *Nature Reviews Materials* 9 (2024), pp. 822–835. DOI: 10.1038/s41578-024-00711-z.

[25] Wenji Li, Zhaojun Wang, Ruitao Mai, Pengxiang Ren, Qinchang Zhang, Yutao Zhou, Ning Xu, Jiafan Zhuang, Bin Xin, Liang Gao, Zhifeng Hao, and Zhun Fan. "Modular Design Automation of the Morphologies, Controllers, and Vision Systems for Intelligent Robots: A Survey". In: *Visual Intelligence* 1 (2023), p. 2. DOI: 10.1007/s44267-023-00006-x.

[26] Toby Howison, Simon Hauser, Josie Hughes, and Fumiya Iida. "Reality-Assisted Evolution of Soft Robots through Large-Scale Physical Experimentation: A Review". In: *Artificial Life* 26.4 (2020), pp. 484–506. DOI: 10.1162/artl_a_00330.

[27] Elvin Alberts, Ilias Gerostathopoulos, Ivano Malavolta, Carlos Hernández Corbato, and Patricia Lago. "Software Architecture-Based Self-Adaptation in Robotics". In: *Journal of Systems and Software* 219, 112258 (2025). DOI: 10.1016/j.jss.2024.112258.

[28] Martin Seifrid, Robert Pollice, Andrés Aguilar-Granda, Zamyla Morgan Chan, Kazuhiro Hotta, Cher Tian Ser, Jenya Vestfrid, Tony C. Wu, and Alán Aspuru-Guzik. "Autonomous Chemical Experiments: Challenges and Perspectives on Establishing a Self-Driving Lab". In: *Accounts of Chemical Research* 55.17 (2022), pp. 2454–2466. DOI: 10.1021/acs.accounts.2c00220.

[29] Joseph H. Montoya, Muratahan Aykol, Abraham Anapolsky, Chirranjeevi B. Gopal, Patrick K. Herring, Jens S. Hummelshøj, Linda Hung, Ha-Kyung Kwon, Daniel Schweigert, Shijing Sun, Santosh K. Suram, Steven B. Torrisi, Amalie Trewartha, and Brian D. Storey. "Toward Autonomous Materials Research: Recent Progress and Future Challenges". In: *Applied Physics Reviews* 9.1, 011405 (2022). DOI: 10.1063/5.0076324.

[30] Milad Abolhasani and Eugenia Kumacheva. "The Rise of Self-Driving Labs in Chemical and Materials Sciences". In: *Nature Synthesis* 2 (2023), pp. 483–492. DOI: 10.1038/s44160-022-00231-0.

[31] Gary Tom, Stefan P. Schmid, Sterling G. Baird, Yang Cao, Kourosh Darvish, Han Hao, Stanley Lo, Sergio Pablo-García, Ella M. Rajaonson, Marta Skreta, Naruki Yoshikawa, Samantha Corapi, Gün Deniz Akkoc, Felix Strieth-Kalthoff, Martin Seifrid, and Alán Aspuru-Guzik. "Self-Driving Laboratories for Chemistry and Materials Science". In: *Chemical Reviews* 124.16 (2024), pp. 9633–9732. DOI: 10.1021/acs.chemrev.4c00055.

[32] Amanda A. Volk and Milad Abolhasani. "Performance Metrics to Unleash the Power of Self-Driving Labs in Chemistry and Materials Science". In: *Nature Communications* 15, 1378 (2024). DOI: 10.1038/s41467-024-45569-5.

[33] Shi Xuan Leong, Caleb E. Griesbach, Rui Zhang, Kourosh Darvish, Yuchi Zhao, Abhijoy Mandal, Yunheng Zou, Han Hao, Varinia Bernales, and Alán Aspuru-Guzik. "Steering towards Safe Self-Driving Laboratories". In: *Nature Reviews Chemistry* 9 (2025), pp. 707–722. DOI: 10.1038/s41570-025-00747-x.

[34] Richard B. Canty and Milad Abolhasani. "The Past, Present and Future of Self-Driving Laboratories". In: *Nature Reviews Chemistry* 10 (2026), pp. 523–537. DOI: 10.1038/s41570-026-00847-2.

[35] Mingguang Chen, Licheng Wang, and Bo Qu. *Recursive Self-Improvement in AI: From Bounded Self-Refinement to Autonomous Research Loops*. 2026. arXiv: 2607.07663 [cs.AI].

[36] Hermann Blum, Francesco Milano, René Zurbrügg, Roland Siegwart, Cesar Cadena, and Abel Gawel. "Self-Improving Semantic Perception for Indoor Localisation". In: *Proceedings of the 5th Conference on Robot Learning*. Vol. 164. Proceedings of Machine Learning Research. 2022, pp. 1211–1222. URL: https://proceedings.mlr.press/v164/blum22a.html.

[37] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. "Robots that Can Adapt Like Animals". In: *Nature* 521 (2015), pp. 503–507. DOI: 10.1038/nature14422.

[38] Jarrett Holtz, Arjun Guha, and Joydeep Biswas. "Robot Action Selection Learning via Layered Dimension Informed Program Synthesis". In: *Proceedings of the Conference on Robot Learning*. Vol. 155. Proceedings of Machine Learning Research. 2021, pp. 1471–1480. URL: https://proceedings.mlr.press/v155/holtz21a.html.

[39] Yuhang Hu, Boyuan Chen, and Hod Lipson. "Egocentric Visual Self-Modeling for Autonomous Robot Dynamics Prediction and Adaptation". In: *npj Robotics* 3 (2025), p. 14. DOI: 10.1038/s44182-025-00031-6.

[40] Ross D. King. *The Past and Future of AI Scientists*. 2026. arXiv: 2608.14407 [cs.AI].

[41] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. *Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents*. Version 3, 12 March 2026. 2025. arXiv: 2505.22954.

[42] Jenny Zhang, Bingchen Zhao, Wannan Yang, Jakob Foerster, Jeff Clune, Minqi Jiang, Sam Devlin, and Tatiana Shavrina. *Hyperagents*. 2026. arXiv: 2603.19461.

[43] Xingang Guo, Darioush Keivan, Usman Syed, Lianhui Qin, Huan Zhang, Geir Dullerud, Peter Seiler, and Bin Hu. *ControlAgent: Automating Control System Design via Novel Integration of LLM Agents and Domain Expertise*. 2024. arXiv: 2410.19811.

[44] Zeyu Wang, Frank P.-W. Lo, Qian Chen, Yongqi Zhang, Chen Lin, Xu Chen, Zhenhua Yu, Alexander J. Thompson, Eric M. Yeatman, and Benny P. L. Lo. "An LLM-enabled Multi-Agent Autonomous Mechatronics Design Framework". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2025. DOI: 10.1109/CVPRW67362.2025.00404.

[45] Allan Zhao, Jie Xu, Mina Konaković Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. "RoboGrammar: Graph Grammar for Terrain-Optimized Robot Design". In: *ACM Transactions on Graphics* 39.6 (2020), 188:1–188:16. DOI: 10.1145/3414685.3417831.

[46] Pingchuan Ma, Tao Du, John Z. Zhang, Kui Wu, Andrew Spielberg, Robert K. Katzschmann, and Wojciech Matusik. "DiffAqua: A Differentiable Computational Design Pipeline for Soft Underwater Swimmers with Shape Interpolation". In: *ACM Transactions on Graphics* 40.4 (2021), 132:1–132:14. DOI: 10.1145/3450626.3459832.

[47] Etor Arza, Frank Veenstra, Tønnes F. Nygaard, and Kyrre Glette. *An Empirical Study on the Computation Budget of Co-Optimization of Robot Design and Control in Simulation*. Version 2; first preprint released in 2024. 2025. arXiv: 2409.08621.

[48] Aldair E. Gongora, Bowen Xu, Wyatt Perry, Chika Okoye, Patrick Riley, Kristofer G. Reyes, Elise F. Morgan, and Keith A. Brown. "A Bayesian Experimental Autonomous Researcher for Mechanical Design". In: *Science Advances* 6.15 (2020), eaaz1708. DOI: 10.1126/sciadv.aaz1708.

[49] Rafael Oliveira, Josh Pinskier, Xing Wang, Lois Liow, Sarah Baldwin, James Brett, Vinoth Viswanathan, Richard Scalzo, and David Howard. "High-Throughput Soft Robot Design via an Adaptive Experimental Platform". In: *npj Robotics* 4 (2026), p. 31. DOI: 10.1038/s44182-026-00092-1.

[50] Jagdeep Bhatia, Holly Jackson, Yunsheng Tian, Jie Xu, and Wojciech Matusik. "Evolution Gym: A Large-Scale Benchmark for Evolving Soft Robots". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/118921efba23fc329e6560b27861f0c2-Abstract.html.

[51] Andrew Spielberg, Allan Zhao, Yuanming Hu, Tao Du, Wojciech Matusik, and Daniela Rus. "Learning-in-the-Loop Optimization: End-to-End Control and Co-Design of Soft Robots through Learned Deep Latent Representations". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/438124b4c06f3a5caffab2c07863b617-Abstract.html.

[52] Yecheng Jason Ma, William Liang, Hung-Ju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. "DrEureka: Language Model Guided Sim-to-Real Transfer". In: *Robotics: Science and Systems*. 2024. arXiv: 2406.01967.

[53] Michael Jae-Yoon Chung and Maya Cakmak. "Iterative Repair of Social Robot Programs from Implicit User Feedback via Bayesian Inference". In: *Proceedings of Robotics: Science and Systems*. 2020. DOI: 10.15607/RSS.2020.XVI.028.

[54] D. M. Lyons and S. Zahra. *Using Taint Analysis and Reinforcement Learning (TARL) to Repair Autonomous Robot Software*. Workshop paper; inspected version 1. 2020. arXiv: 2005.03813.

[55] Edgar Buchanan, Léni K. Le Goff, Wei Li, Emma Hart, Agoston E. Eiben, Matteo De Carlo, Alan F. Winfield, Matthew F. Hale, Robert Woolley, Mike Angus, Jon Timmis, and Andy M. Tyrrell. "Bootstrapping Artificial Evolution to Design Robots for Autonomous Fabrication". In: *Robotics* 9.4 (2020), p. 106. DOI: 10.3390/robotics9040106.

[56] Pavel Nikolaev et al. "Autonomy in Materials Research: A Case Study in Carbon Nanotube Growth". In: *npj Computational Materials* 2 (2016), p. 16031. DOI: 10.1038/npjcompumats.2016.31.

[57] A. Gilad Kusne et al. “On-the-Fly Closed-Loop Materials Discovery via Bayesian Active Learning”. In: *Nature Communications* 11 (2020), p. 5966. DOI: 10.1038/s41467-020-19597-w.

[58] Alireza Ghafarollahi and Markus J. Buehler. “Automating Alloy Design and Discovery with Physics-Aware Multimodal Multiagent AI”. In: *Proceedings of the National Academy of Sciences* 122.4 (2025), e2414074122. DOI: 10.1073/pnas.2414074122.

[59] Joshua Pinskier and David Howard. “From Bioinspiration to Computer Generation: Developments in Autonomous Soft Robot Design”. In: *Advanced Intelligent Systems* 4.1, 2100086 (2022). DOI: 10.1002/aisy.202100086.

[60] André Rosendo, Marco von Atzigen, and Fumiya Iida. “The Trade-Off between Morphology and Control in the Co-Optimized Design of Robots”. In: *PLOS ONE* 12.10 (2017), e0186107. DOI: 10.1371/journal.pone.0186107.

[61] Enrico Zardini, Davide Zappetti, Davide Zambrano, Giovanni Iacca, and Dario Floreano. “Seeking Quality Diversity in Evolutionary Co-Design of Morphology and Control of Soft Tensegrity Modular Robots”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2021, pp. 189–197. DOI: 10.1145/3449639.3459311.

[62] Jiawei Fang, Yuxuan Sun, Chengtian Ma, Qiuyu Lu, and Lining Yao. *RoboMoRe: LLM-based Robot Co-design via Joint Optimization of Morphology and Reward*. 2025. arXiv: 2506.00276.

[63] Yayati Jadhav and Amir Barati Farimani. “Large Language Model Agent as a Mechanical Designer”. In: *Journal of Engineering Design* (2026). Published online 11 February 2026. DOI: 10.1080/09544828.2026.2624356.

[64] Haoju Lin, Wenchang Zhang, Weipeng Xu, Xiang Li, Tian Xu, and Tianju Xue. *TO-Master: An LLM-Agent Framework for Automated Topology Optimization*. 2026. arXiv: 2607.01812.

[65] Robert Pastor, Milan Mihola, Zdeněk Zeman, and Adam Boleslavský. “Knowledge-Based Automated Mechanical Design of a Robot Manipulator”. In: *Applied Sciences* 12.12, 5897 (2022). DOI: 10.3390/app12125897.

[66] Chunru Lin, Haotian Yuan, Yian Wang, Xiaowen Qiu, Tsun-Hsuan Johnson Wang, Minghao Guo, Bohan Wang, Yashraj Narang, Dieter Fox, and Chuang Gan. “RobotSmith: Generative Robotic Tool Design for Acquisition of Complex Manipulation Skills”. In: *Advances in Neural Information Processing Systems*. Vol. 38. 2025, pp. 122275–122304. DOI: 10.52202/085713-3684.

[67] Mahsa Raeisinezhad, Nicholas Pagliocca, Behrad Koohbor, and Mitja Trkov. “Design Optimization of a Pneumatic Soft Robotic Actuator Using Model-Based Optimization and Deep Reinforcement Learning”. In: *Frontiers in Robotics and AI* 8 (2021), p. 639102. DOI: 10.3389/frobt.2021.639102.

[68] Bartosz Kaczmarski, Derek E. Moulton, Alain Goriely, and Ellen Kuhl. “Bayesian Design Optimization of Biomimetic Soft Actuators”. In: *Computer Methods in Applied Mechanics and Engineering* 408 (2023), p. 115939. DOI: 10.1016/j.cma.2023.115939.

[69] Junjun Liu, Zeyu Wang, Letian Qian, Rong Luo, and Xin Luo. “Task-Oriented Systematic Design of a Heavy-Duty Electrically Actuated Quadruped Robot with High Performance”. In: *Sensors* 23.15 (2023), p. 6696. DOI: 10.3390/s23156696.

[70] Jinseong Han, Sunwoong Yang, and Namwoo Kang. *A Multi-Agent System for Motor Design Optimization via an FEA–AI Hybrid Approach*. 2026. arXiv: 2606.09037.

[71] Samuel Smocot, James R. Forbes, and Audrey Sedal. “Observability-Informed Optimal Sensor Placement for Soft Robots”. In: *2026 IEEE 9th International Conference on Soft Robotics (RoboSoft)*. 2026, pp. 430–437. DOI: 10.1109/ROBOSOFT67810.2026.11522828.

[72] Andrew Spielberg, Alexander Amini, Lillian Chin, Wojciech Matusik, and Daniela Rus. “Co-Learning of Task and Sensor Placement for Soft Robotics”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1208–1215. DOI: 10.1109/LRA.2021.3056369.

[73] Dejan Milojevic, Gioele Zardini, Miriam Elser, Andrea Censi, and Emilio Frazzoli. "CODEI: Resource-Efficient Task-Driven Co-Design of Perception and Decision Making for Mobile Robots Applied to Autonomous Vehicles". In: *IEEE Transactions on Robotics* 41 (2025), pp. 2727–2748. DOI: 10.1109/TRO.2025.3552347.

[74] Sabrina M. Neuman, Brian Plancher, Thomas Bourgeat, Thierry Tambe, Srinivas Devadas, and Vijay Janapa Reddi. "Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology". In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 2021, pp. 674–686. DOI: 10.1145/3445814.3446746.

[75] Sabrina M. Neuman, Radhika Ghosal, Thomas Bourgeat, Brian Plancher, and Vijay Janapa Reddi. "RoboShape: Using Topology Patterns to Scalably and Flexibly Deploy Accelerators Across Robots". In: *Proceedings of the 50th Annual International Symposium on Computer Architecture*. 2023, 69. DOI: 10.1145/3579371.3589104.

[76] Srivatsan Krishnan, Thierry Tambe, Zishen Wan, and Vijay Janapa Reddi. *AutoSoC: Automating Algorithm–SoC Co-Design for Aerial Robots*. 2021. arXiv: 2109.05683 [cs.AR].

[77] Zhuolun He, Haoyuan Wu, Xinyun Zhang, Xufeng Yao, Su Zheng, Haisheng Zheng, and Bei Yu. "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA". In: *2023 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. 2023, pp. 1–6. arXiv: 2308.10204.

[78] Peter M. Attia, Aditya Grover, Norman Jin, Kristen A. Severson, Todor M. Markov, Yang-Hung Liao, Michael H. Chen, Bryan Cheong, Nicholas Perkins, Zi Yang, Patrick K. Herring, Muratahan Aykol, Stephen J. Harris, Richard D. Braatz, Stefano Ermon, and William C. Chueh. "Closed-Loop Optimization of Fast-Charging Protocols for Batteries with Machine Learning". In: *Nature* 578 (2020), pp. 397–402. DOI: 10.1038/s41586-020-1994-5.

[79] Adarsh Dave, Jared Mitchell, Sven Burke, Hongyi Lin, Jay Whitacre, and Venkatasubramanian Viswanathan. "Autonomous Optimization of Non-Aqueous Li-Ion Battery Electrolytes via Robotic Experimentation and Machine Learning Coupling". In: *Nature Communications* 13 (2022), p. 5454. DOI: 10.1038/s41467-022-32938-1.

[80] Yu Chen, Haoran Chen, Hao Zeng, Jianjun Zhu, Kai Chen, Zhenyu Cui, and Jianli Wang. "Structural Optimization Design of Sinusoidal Wavy Plate Fin Heat Sink with Crosscut by Bayesian Optimization". In: *Applied Thermal Engineering* 213 (2022), p. 118755. DOI: 10.1016/j.applthermaleng.2022.118755.

[81] James R. Deneault, Jorge Chang, Jay Myung, Daylond Hooper, Andrew Armstrong, Mark Pitt, and Benji Maruyama. "Toward Autonomous Additive Manufacturing: Bayesian Optimization on a 3D Printer". In: *MRS Bulletin* 46 (2021), pp. 566–575. DOI: 10.1557/s43577-021-00051-1.

[82] Debargha Ganguly, Sumit Kumar, Ishwar Balappanawar, Weicong Chen, Shashank Kambhatla, Srinivasan Iyengar, Shivkumar Kalyanaraman, Ponnurangam Kumaraguru, and Vipin Chaudhary. *LABELING COPILOT: A Deep Research Agent for Automated Data Curation in Computer Vision*. 2025. arXiv: 2509.22631.

[83] Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. "AutoML-Agent: A Multi-Agent LLM Framework for Full-Pipeline AutoML". In: *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. Proceedings of Machine Learning Research. 2025, pp. 60099–60146. URL: https://proceedings.mlr.press/v267/trirat25a.html.

[84] Marc Peter Deisenroth and Carl Edward Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Proceedings of the 28th International Conference on Machine Learning*. 2011. URL: https://icml.cc/2011/papers/323_icmlpaper.pdf.

[85] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. "Automatic LQR Tuning Based on Gaussian Process Global Optimization". In: *IEEE International Conference on Robotics and Automation*. 2016, pp. 270–277. DOI: 10.1109/ICRA.2016.7487144.

[86] Daniel Widmer, Dongho Kang, Bhavya Sukhija, Jonas Hübotter, Andreas Krause, and Stelian Coros. "Tuning Legged Locomotion Controllers via Safe Bayesian Optimization". In: *Proceedings of the 7th Conference on Robot Learning*. Vol. 229. Proceedings of Machine Learning Research. 2023, pp. 2444–2464. URL: https://proceedings.mlr.press/v229/widmer23a.html.

[87] Akshara Rai, Rika Antonova, Franziska Meier, and Christopher G. Atkeson. "Using Simulation to Improve Sample-Efficiency of Bayesian Optimization for Bipedal Robots". In: *Journal of Machine Learning Research* 20.49 (2019), pp. 1–24. URL: https://www.jmlr.org/papers/v20/18-196.html.

[88] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. "Code as Policies: Language Model Programs for Embodied Control". In: *IEEE International Conference on Robotics and Automation*. 2023, pp. 9493–9500. DOI: 10.1109/ICRA48891.2023.10160591.

[89] Charles Dawson, Anjali Parashar, and Chuchu Fan. *RADIUM: Predicting and Repairing End-to-End Robot Failures Using Gradient-Accelerated Sampling*. 2024. arXiv: 2404.03412 [cs.RO].

[90] Guofeng Cui, Yuning Wang, Wensen Mao, Yuanlin Duan, and He Zhu. "Abstraction Refinement-Guided Program Synthesis for Robot Learning from Demonstrations". In: *Proceedings of the ACM on Programming Languages* 9.OOPSLA2, 292 (2025), pp. 555–583. DOI: 10.1145/3763070.

[91] Adrien Baranes and Pierre-Yves Oudeyer. "Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots". In: *Robotics and Autonomous Systems* 61.1 (2013), pp. 49–73. DOI: 10.1016/j.robot.2012.05.008.

[92] Simon Hangl, Vedran Dunjko, Hans J. Briegel, and Justus Piater. "Skill Learning by Autonomous Robotic Playing Using Active Learning and Exploratory Behavior Composition". In: *Frontiers in Robotics and AI* 7, 42 (2020). DOI: 10.3389/frobt.2020.00042.

[93] Michael Ahn, Debidatta Dwibedi, Chelsea Finn, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Karol Hausman, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Sean Kirmani, Isabel Leal, Edward Lee, Sergey Levine, Yao Lu, Sharath Maddineni, Kanishka Rao, Dorsa Sadigh, Pannag Sanketi, Pierre Sermanet, Quan Vuong, Stefan Welker, Fei Xia, Ted Xiao, Peng Xu, Steve Xu, and Zhuo Xu. *AutoRT: Embodied Foundation Models for Large Scale Orchestration of Robotic Agents*. 2024. arXiv: 2401.12963 [cs.RO].

[94] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. "Learning Navigation Behaviors End-to-End with AutoRL". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 2007–2014. DOI: 10.1109/LRA.2019.2899918.

[95] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. "Eureka: Human-Level Reward Design via Coding Large Language Models". In: *International Conference on Learning Representations*. 2024. arXiv: 2310.12931.

[96] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. "Text2Reward: Reward Shaping with Language Models for Reinforcement Learning". In: *International Conference on Learning Representations*. 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/hash/9941833e8327910ef25daeb9005e4748-Abstract-Conference.html.

[97] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. "Automatic Goal Generation for Reinforcement Learning Agents". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. 2018, pp. 1515–1528. URL: https://proceedings.mlr.press/v80/florensa18a.html.

[98] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. "GenSim: Generating Robotic Simulation Tasks via Large Language Models". In: *International Conference on Learning Representations*. 2024. URL: https://proceedings.iclr.cc/paper_files/paper/2024/hash/143ea4a156ef64f32d4d905206cf32e1-Abstract-Conference.html.

[99] Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. "GenSim2: Scaling Robot Data Generation with Multi-modal and Reasoning LLMs". In: *Proceedings of the 8th Conference on Robot Learning*. Vol. 270. Proceedings of Machine Learning Research. 2025, pp. 5030–5066. URL: https://proceedings.mlr.press/v270/hua25a.html.

[100] Kanghyun Ryu, Qiayuan Liao, Zhongyu Li, Payam Delgosha, Koushil Sreenath, and Negar Mehr. "CurricuLLM: Automatic Task Curricula Design for Learning Complex Robot Skills Using Large Language Models". In: *IEEE International Conference on Robotics and Automation*. 2025. arXiv: 2409.18382 [cs.RO].

[101] Ryan Julian, Benjamin Swanson, Gaurav Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. "Never Stop Learning: The Effectiveness of Fine-Tuning in Robotic Reinforcement Learning". In: *Proceedings of the 2020 Conference on Robot Learning*. Vol. 155. Proceedings of Machine Learning Research. 2021, pp. 2120–2136. URL: https://proceedings.mlr.press/v155/julian21a.html.

[102] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. *Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges*. 2019. arXiv: 1907.00182 [cs.RO].

[103] Yifei Yan and Linqi Ye. *Tree Learning: A Multi-Skill Continual Learning Framework for Humanoid Robots*. 2026. arXiv: 2604.12909 [cs.RO].

[104] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X. Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo F. Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Żołna, Scott Reed, Sergio Gómez Colmenarejo, Jon Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Tom Rothörl, José Enrique Chen, Yusuf Aytar, Dave Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. "RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation". In: *Transactions on Machine Learning Research* (2023). arXiv: 2306.11706. URL: https://openreview.net/forum?id=vsCpILiWHu.

[105] Seyed Kamyar Seyed Ghasemipour, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, and Igor Mordatch. "Self-Improving Embodied Foundation Models". In: *Advances in Neural Information Processing Systems*. Vol. 38. 2025. arXiv: 2509.15155.

[106] Open X-Embodiment Collaboration. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. Version 9, 14 May 2025. 2023. arXiv: 2310.08864.

[107] Seulbae Kim and Taesoo Kim. "RoboFuzz: Fuzzing Robotic Systems over Robot Operating System (ROS) for Finding Correctness Bugs". In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2022, pp. 447–458. DOI: 10.1145/3540250.3549164.

[108] Gustavo Rezende Silva, Juliane Päßler, S. Lizeth Tapia Tarifa, Einar Broch Johnsen, and Carlos Hernández Corbato. "ROSA: A Knowledge-Based Solution for Robot Self-Adaptation". In: *Frontiers in Robotics and AI* 12, 1531743 (2025). DOI: 10.3389/frobt.2025.1531743.

[109] Carlo Bosio and Mark W. Mueller. "Automated Layout and Control Co-Design of Robust Multi-UAV Transportation Systems". In: *IEEE Robotics and Automation Letters* 10.4 (2025), pp. 3956–3963. DOI: 10.1109/LRA.2025.3547307.

[110] Punith Reddy Vanteddu, Gabriele Nava, Fabio Bergonti, Giuseppe L'Erario, Antonello Paolino, and Daniele Pucci. *From CAD to URDF: Co-Design of a Jet-Powered Humanoid Robot Including CAD Geometry*. 2024. arXiv: 2410.07963 [cs.RO].

[111] Andrew Wilhelm and Josie Hughes. *SwarmCoDe: A Scalable Co-Design Framework for Heterogeneous Robot Swarms via Dynamic Speciation*. 2026. arXiv: 2603.26240 [cs.RO].

[112] Edgar Buchanan, Léni K. Le Goff, Matthew F. Hale, Emma Hart, Agoston E. Eiben, Matteo De Carlo, Mike Angus, Robert Woolley, Jon Timmis, Alan F. Winfield, and Andy M. Tyrrell. "Towards a Unified Framework for Software–Hardware Integration in Evolutionary Robotics". In: *Robotics* 13.11 (2024), p. 157. DOI: 10.3390/robotics13110157.

[113] Philippe Martin Wyder, Riyaan Bakhda, Meiqi Zhao, Quinn A. Booth, Matthew E. Modi, Andrew Song, Simon Kang, Jiahao Wu, Priya Patel, Robert T. Kasumi, David Yi, Nihar Niraj Garg, Pranav Jhunjhunwala, Siddharth Bhutoria, Evan H. Tong, Yuhang Hu, Judah Goldfeder, Omer Mustel, Donghan Kim, and Hod Lipson. "Robot Metabolism: Toward Machines That Can Grow by Consuming Other Machines". In: *Science Advances* 11.29, eadu6897 (2025). DOI: 10.1126/sciadv.adu6897.

[114] Ersin Daş and Joel W. Burdick. "An Active Learning Based Robot Kinematic Calibration Framework Using Gaussian Processes". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 11495–11501. DOI: 10.1109/ICRA48891.2023.10161070.

[115] Jun Yang, Jason Rebello, and Steven L. Waslander. "Next-Best-View Selection for Robot Eye-in-Hand Calibration". In: *2023 20th Conference on Robots and Vision (CRV)*. 2023, pp. 161–168. DOI: 10.1109/CRV60082.2023.00028.

[116] Maxime Selingue, Adel Olabi, Stéphane Thiery, and Richard Béarée. "Data-Efficient, Fast and Autonomous Hybrid Calibration for Robot Positioning Accuracy Enhancement Using Active Learning-Based Method". In: *Journal of Intelligent & Robotic Systems* 112, 26 (2026). DOI: 10.1007/s10846-026-02356-2.

[117] Marcin Kolakowski. "Automated Calibration of RSS Fingerprinting Based Systems Using a Mobile Robot and Machine Learning". In: *Sensors* 21.18, 6270 (2021). DOI: 10.3390/s21186270.

[118] Giovanni Franzese, Max Spahn, Jens Kober, Javier Alonso-Mora, and Cosimo Della Santina. "Accurate and Affordable Cobot Calibration without External Measurement Devices". In: *Communications Engineering* 5, 138 (2026). DOI: 10.1038/s44172-026-00633-4.

[119] Konstantinos Chatzilygeroudis, Vassilis Vassiliades, and Jean-Baptiste Mouret. "Reset-free Trial-and-Error Learning for Robot Damage Recovery". In: *Robotics and Autonomous Systems* 100 (2018), pp. 236–250. DOI: 10.1016/j.robot.2017.11.010.

[120] Esther Aguado, Zorana Milosevic, Carlos Hernández, Ricardo Sanz, Mario Garzon, Darko Bozhinoski, and Claudio Rossi. "Functional Self-Awareness and Metacontrol for Underwater Robot Autonomy". In: *Sensors* 21.4, 1210 (2021). DOI: 10.3390/s21041210.

[121] Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. "Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 79–86. DOI: 10.1109/IROS.2017.8202141.

[122] Seyede Fatemeh Ghoreishi, Ryan D. Sochol, Dheeraj Gandhi, Axel Krieger, and Mark Fuge. "Bayesian Optimization for Design of Multi-Actuator Soft Catheter Robots". In: *IEEE Transactions on Medical Robotics and Bionics* 3.3 (2021), pp. 725–737. DOI: 10.1109/TMRB.2021.3098119.

[123] Srivatsan Krishnan, Zishen Wan, Kshitij Bhardwaj, Paul Whatmough, Aleksandra Faust, Sabrina Neuman, Gu-Yeon Wei, David Brooks, and Vijay Janapa Reddi. *AutoPilot: Automating SoC Design Space Exploration for SWaP Constrained Autonomous UAVs*. 2021. arXiv: 2102.02988 [cs.RO].

[124] Santiago Muiños-Landin, Alexander Fischer, Viktor Holubec, and Frank Cichos. "Reinforcement Learning with Artificial Microswimmers". In: *Science Robotics* 6.52 (2021), eabd9285. DOI: 10.1126/scirobotics.abd9285.

[125] Zhuoqun Xu and Lailai Zhu. "Training Microrobots to Swim by a Large Language Model". In: *Physical Review Applied* 23.4 (2025), p. 044058. DOI: 10.1103/PhysRevApplied.23.044058.

[126] Gilhyun Ryou, Ezra Tal, and Sertac Karaman. "Multi-Fidelity Black-Box Optimization for Time-Optimal Quadrotor Maneuvers". In: *Proceedings of Robotics: Science and Systems*. 2020. DOI: 10.15607/RSS.2020.XVI.032.

[127] Seyed Reza Ahmadzadeh, Matteo Leonetti, and Petar Kormushev. "Online Direct Policy Search for Thruster Failure Recovery in Autonomous Underwater Vehicles". In: *6th International Workshop on Evolutionary and Reinforcement Learning for Autonomous Robot Systems*. 2013. URL: https://kormushev.com/papers/Ahmadzadeh_ERLARS-2013.pdf.

[128] Julia Hindel, Simon Bultmann, Houman Masnavi, Daniele Cattaneo, and Abhinav Valada. *Self-Supervised Online Robot-Agnostic Traversability Estimation for Open-World Environments*. 2026. arXiv: 2605.28442 [cs.RO].

[129] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. "AutoMoDe: A Novel Approach to the Automatic Design of Control Software for Robot Swarms". In: *Swarm Intelligence* 8 (2014), pp. 89–112. DOI: 10.1007/s11721-014-0092-4.

[130] Miquel Kegeleirs, David Garzón Ramos, Ken Hasselmann, Lorenzo Garattoni, Gianpiero Francesca, and Mauro Birattari. "Transferability in the Automatic Off-Line Design of Robot Swarms: From Sim-to-Real to Embodiment and Design-Method Transfer Across Different Platforms". In: *IEEE Robotics and Automation Letters* 9.3 (2024), pp. 2758–2765. DOI: 10.1109/LRA.2024.3360013.

[131] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *Proceedings of the Conference on Robot Learning*. Vol. 100. Proceedings of Machine Learning Research. 2020, pp. 1094–1100. URL: https://proceedings.mlr.press/v100/yu20a.html.

[132] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhao Huang, Rui Chen, and Hao Su. "ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills". In: *International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=b_CQDy9vrD1.

[133] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. "BEHAVIOR-1K: A Benchmark for Embodied AI with 1,000 Everyday Activities and Realistic Simulation". In: *Proceedings of the 6th Conference on Robot Learning*. Vol. 205. Proceedings of Machine Learning Research. 2023, pp. 80–93. URL: https://proceedings.mlr.press/v205/li23a.html.