ChatGPT

Clean-Room Functional Bounds for Replicating the Described Engineering Software Locally

Scope, interpretation, and guardrails

The pasted text describes a set of commercial engineering tools spanning (a) multiphysics simulation and “simulation app” deployment, (b) semiconductor/MEMS process and device modeling, (c) IC/package/PCB design and analysis, (d) manufacturing process simulation, and (e) safety/reliability and embedded “infrastructure” software (RTOS/virtualization, embedded database, software verification). [1]

This report produces **systematic “bounds”** (what each process is responsible for vs. what is external), with enough detail to build a **local, clean-room implementation** that matches the publicly documented behavior and workflows—without copying proprietary internals or bypassing licensing/entitlements. (Anything that would amount to circumventing licensing or reverse engineering protected components is intentionally out of scope.) [2]

To make this digestible by automated reasoning systems, each process is expressed as a **contract-style decomposition**:

- **Purpose** (user-visible outcome)
- **In-scope responsibilities** (what your replica must implement)
- **Out-of-scope dependencies** (what is external and can be delegated)
- **Inputs / outputs** (data contracts + canonical formats)
- **Core computations** (numerics/algorithms implied by the marketed capability)
- **Local replica blueprint** (minimal components needed for a credible baseline)

Where possible, the bounds are anchored to vendor documentation describing the relevant capability. [3]

Cross-cutting reference architecture for a local replica

A practical way to “replicate locally” across such a diverse suite is to build a **shared platform substrate** with domain plug-ins, rather than many unrelated one-off applications. The pasted text repeatedly implies “platform + add-ons + interfaces + deployment,” which naturally maps to a modular architecture. [4]

Core substrate components

Model/asset registry (project layer)
Purpose: unify “models,” “apps,” “libraries,” and “runs” into versioned artifacts with provenance and reproducibility. This mirrors how platforms emphasize model management and traceability. [5]

Geometry + layout ingestion layer
Purpose: turn CAD/ECAD/layout into simulation-ready geometry/topology with stable IDs for selections/

regions/nets. This is explicitly called out for ECAD import (GDS-II/IPC-2581/ODB++ → 3D geometry) and for semiconductor process tools (layout-driven 3D build). [6]

Discretization layer (meshing/voxelization)
Purpose: provide multiple discretization backends—unstructured FEM meshes, structured FDTD grids, and voxel/level-set topography representations—under one API. The text spans all three paradigms: FEM-style multiphysics, FDTD for EMC/cables, and voxel-based process simulation. [7]

Solver orchestration layer
Purpose: schedule coupled solves, parameter sweeps, Monte Carlo/DOE, and optimization loops while capturing run metadata. This is essential for uncertainty quantification, robust design, process variation experiments, and high-sigma/yield analysis. [8]

Postprocessing + reporting layer
Purpose: consistent field visualization, cross sections, measurements, and export to reports/slide decks; multiple tools explicitly emphasize guided visualization and reporting/export. [9]

Proven open-source building blocks (as “reference implementations”)

A clean-room replica can accelerate by adopting established open components for each substrate role:

- FEM/multiphysics frameworks (examples): **MOOSE** (open-source multiphysics FEM framework) [10] and **FEniCS/FEniCSx** (open-source platform solving PDEs with FEM, Python/C++ interfaces). [11]
- Meshing: **Gmsh** (open-source 3D FEM mesh generator with built-in CAD engine and scripting/APIs). [12]
- CFD: **OpenFOAM** (open-source CFD toolbox; extensive continuum mechanics scope). [13]
- Visualization: **VTK** (open-source scientific visualization toolkit) [14] and **ParaView** (open-source visualization application built on VTK). [15]
- Circuit simulation: **ngspice** (open-source SPICE simulator) [16] and **Xyce** (open-source SPICE-compatible, high-performance parallel analog simulator). [17]
- FPGA toolchain primitives: **Yosys** (open-source Verilog synthesis) [18] , **nextpnr** (open-source timing-driven FPGA place & route) [19] , and **VTR/VPR** (open-source FPGA CAD flow for packing/placement/routing/timing analysis). [20]
- Embedded “infrastructure”: **SQLite** as an embedded SQL database engine pattern (serverless, cross-platform file format). [21]

These are not “drop-in replacements” for the commercial products; they are credible anchors for a modular replica baseline aligned with the processes described. [22]

Multiphysics platforms and simulation-application deployment bounds

Key vendor anchor: COMSOL [23] . [24]

Physics-model authoring platform

Purpose
Create single-physics or fully coupled multiphysics models; solve; visualize/evaluate results; and build simulation applications on top of the modeling core. [25]

In-scope responsibilities (replica target)
- A hierarchical “model tree” that binds: geometry → materials → physics interfaces → boundary/initial conditions → meshing/discretization → solver setup → postprocessing. [24]
- Strong multiphysics coupling: shared fields, shared meshes (or mapped meshes), and nonlinear coupled solves. (The marketing language “fully coupled multiphysics” implies robust coupled nonlinear solves.) [26]
- Study types: stationary, transient, eigenfrequency, parametric sweeps (implied across the suite by optimization/UQ/experiments). [27]

Out-of-scope dependencies (acceptable to delegate)
- A full CAD kernel: many platforms embed commercial geometry kernels; a replica can use OpenCASCADE/mesh-based geometry, provided import and repair are adequate for downstream discretization. (COMSOL’s CAD Import uses Parasolid for supported formats.) [28]

Inputs / outputs (minimum contracts)
- Inputs: CAD (STEP/IGES/Parasolid/ACIS, etc.) and/or procedurally defined primitives; material property tables; boundary-condition selections. [29]
- Outputs: field data (nodal/element), derived quantities, plots, and exportable meshes for downstream tools. [30]

Core computations
- FEM/FVM PDE assembly and nonlinear solution (Newton/Krylov + preconditioning is a common baseline in multiphysics FEM frameworks). [31]
- Consistent units, coupled constitutive models, and robust Jacobian handling for nonlinear multiphysics. [32]

Local replica blueprint
A minimal credible multiphysics replica can be built as: geometry ingestion → Gmsh meshing → FEM kernel (MOOSE or FEniCSx) → solver backend (PETSc via those frameworks) → VTK/ParaView output. [33]

Add-on module pattern bounds

The text describes specialized modules (electromagnetics, fluids/heat, structural/acoustics, chemical engineering) that “connect seamlessly” via the platform, keeping a consistent workflow. [25]

Purpose
Extend the platform with domain-specific PDEs, constitutive laws, boundary conditions, and postprocessing. [26]

In-scope responsibilities (replica target)
- A plug-in registry that adds: (a) PDE forms/kernels, (b) material models, (c) default study templates, and (d) domain postprocessing operators, without changing the core workflow. [32]

- A shared “selection model” for applying physics to parts of a geometry (domains/boundaries/edges), because both ECAD conversion and CAD import emphasize downstream selections. [34]

Core computations implied by named examples
- Particle tracing as “equations of motion in fields/fluids,” including user-defined forces and coupling patterns (one-way coupling is explicitly documented). [35]
- Optimization loops for parameter/shape/topology optimization and parameter estimation. [36]
- Uncertainty quantification via screening/sensitivity/propagation/reliability, with surrogate models such as sparse PCE and Gaussian Process, and adaptive sampling. [37]

Local replica blueprint
- Particle tracing: ODE integrators + field samplers + collision/boundary handling; keep coupling modes explicit (one-way vs two-way). [38]
- Optimization: wrap the solver in an objective-function API + gradient strategy (adjoint where feasible, finite-diff otherwise) + constraints. [39]
- UQ: centralize sampling, surrogate training, and QoI extraction; implement at least PCE + GP to match documented module behavior. [40]

Interfacing and ECAD/CAD import bounds

Purpose
Bring external design data (CAD and ECAD/layout) into the simulation environment and preserve semantic structure (layers, nets, assemblies) for downstream physics assignment. [41]

In-scope responsibilities (replica target)
- CAD ingestion/repair pipeline (tolerant geometry healing, defeaturing hooks, watertightness checks). [42]
- ECAD/layout ingestion: import GDS-II/IPC-2581/ODB++ and generate 3D geometry + selections for copper/dielectric regions and nets. [43]

Out-of-scope dependencies
- Exact fidelity to commercial importers (which may rely on expensive proprietary format parsers); a replica can target a strict subset of formats, as long as the geometry + selection semantics are preserved. [44]

Data contracts
- Layout import must produce: (a) a 3D solid model, (b) a net/layer map, and (c) stable region IDs for assignment. [45]
- If you support OASIS export for layouts, that becomes a separate contract from the simulation mesh export. [46]

Simulation-application deployment bounds

The text describes two deployment concepts: (1) compile apps to run “anywhere,” and (2) host apps for browser/thin-client access with admin controls. [47]

Process: “Standalone app deployment” (Compiler-class)
Purpose: package a simulation application into a distributable artifact that runs without the full authoring environment. [48]

Replica bounds: treat this as an **application bundler** + **runtime** (model executor + UI). Keep model execution deterministic and isolate licensing concerns by relying on your own runtime, not proprietary engines. (Do not attempt to reproduce vendor licensing mechanics.) [49]

Process: "Server-hosted apps" (Server-class)
Purpose: publish apps so users can run them via web browser or client, with administration and scaling. [50]
In-scope responsibilities: authentication, app catalog, job submission, resource scheduling, and audit logs. [51]
Scaling bound: the COMSOL Server manual describes a "Primary" instance handling incoming connections and "Secondary" servers executing workloads, implying a head-node + worker-node architecture with shared working directories. [52]

Semiconductor process modeling, MEMS platforms, and embedded programmable logic bounds

Key vendor anchors: Lam Research [53] and Coventor [54] . [55]

Virtual fabrication and 3D topography evolution

The SEMulator3D description is unusually explicit about internal *capability partitioning*, making it well-suited to a "bounds" specification: core voxel engine + surface-evolution engine + automation + electrical extraction + export modules. [56]

Process contract: Layout + process flow → evolving 3D structure
Purpose: predict the final 3D device structure from (a) design/layout data and (b) an integrated process flow, enabling early detection of process/design issues and "virtual experiments." [57]
In-scope responsibilities:
- **Layout ingestion** (GDS/OASIS compatible layout editor is explicitly listed) and mapping layout layers to materials/masks. [56]
- **Process step library** with calibration parameters (deposition conformality, etch anisotropy/selectivity, etc. are highlighted in the pasted text; the datasheet formalizes this as a "default step library" plus "custom Python library"). [58]
- **3D voxel modeling engine** and **3D model viewer**, plus "direct voxel-data export" for interfacing to other software. [58]
Out-of-scope dependencies: exact matching of any particular foundry's proprietary process recipes; a clean-room replica should define a parameter schema and calibration workflow rather than attempt to embed confidential recipes. [59]
Inputs: layout + process-flow DSL + material DB; Outputs: time-indexed 3D structures + measures + optional mesh export. [56]

Process contract: Advanced surface evolution (etch/deposition/epi)
Purpose: improve accuracy for plasma etch, selective epitaxy, and other advanced topography processes using a dedicated surface-evolution modeling engine (MultiEtch, Selective Epitaxy, pattern dependence, visibility-limited deposition/etch, etc.). [60]
In-scope responsibilities: implement at least one robust topography representation beyond naive voxels for

accuracy/performance—commonly a level-set method for surface evolution. An open-source reference in this specific niche is **ViennaPS**, described as an open-source process simulation framework focused on etching/deposition topography evolution, using a high-performance level-set method with hierarchical run-length encoding and supporting Monte Carlo ray tracing for feature-scale flux. [61]
Local replica blueprint:
- Geometry state: voxel grid for "always succeeds" robustness + optional level-set surface for high-fidelity steps. [62]
- Process-step API: `apply(step, state) -> state'` with explicit anisotropy/selectivity and visibility/flux submodels. [62]
- Flux models: deterministic analytic kernels plus optional Monte Carlo ray tracing when shadowing/visibility is central (as ViennaPS explicitly supports). [61]

Process contract: Dopant concentration handling
Purpose: support ion implant, thermal diffusion, doped epitaxy/deposition, and gradient visualization. [58]
In-scope responsibilities: add scalar fields on the same spatial representation (voxel or mesh) and couple diffusion/activation models to geometry evolution where needed. [58]

Automation, virtual experiments, and analytics

Process contract: Massively parallel process variation studies
Purpose: automate large numbers of experiments and use "virtual metrology" to measure critical geometry, with "structure search" inspecting build areas for violations. [63]
In-scope responsibilities:
- Parameter sweep engine (grid, Latin hypercube, DOE plans) with concurrency control (work stealing / queue). [64]
- Virtual metrology: measurement operators that act on 3D state (CDs, thickness, spacing, overlay proxies) and generate response tables. [65]
- Design-technology checking hooks ("Design – Technology Checking" appears explicitly), which implies geometric rule checks bound to the generated structure, not just the original layout. [66]
- "Profile export" for dopant/surface (i.e., export 1D/2D slices akin to metrology traces). [67]

Process contract: Statistical analytics module
Purpose: guide design/execution/analysis of large statistical experiments, including Monte Carlo process variation, and rank parameters by impact using multivariate regression. [68]
In-scope responsibilities: experiment design UI/workflow, run orchestration, and a stats engine implementing at least: Monte Carlo sampling, regression/feature importance, and sensitivity indices. [69]

Electrical extraction inside the process modeler

Process contract: R/C extraction and device characteristics
Purpose: compute conductor resistance and net-to-net capacitance and extract electrical characteristics (e.g., transistor characteristics) directly from the 3D structure. [70]
In-scope responsibilities:
- A net assignment model ("Port/Net Assignment" appears explicitly). [66]
- Resistance solver and capacitance solver operating on the 3D geometry (voxel → mesh → field solve is a common pipeline). [71]
Local replica blueprint:

- Convert voxel/level-set solids into a tetra/hex mesh for electrostatics/conduction solves (mesh export is listed as a module capability). [72]
- Use FEM (or finite-volume) to solve $\nabla\cdot(\sigma\nabla V)=0$ for resistance and $\nabla\cdot(\varepsilon\nabla\varphi)=0$ for capacitance; build R and C matrices from boundary conditions on terminals. (This is a standard formulation for such extraction; your implementation must validate against known structures.) [73]

MEMS design automation platform

CoventorMP is described as a unified environment combining multiple modeling abstractions—from "fully parametric design entry" to detailed solvers and reduced-order/compact models—so a viable local replica needs both a high-fidelity solver path and a model-order-reduction/compact-model path. [74]

Process contract: Single "golden master" design → multiple model forms
Purpose: maintain one parametric design source that can generate (a) detailed device-level simulations and (b) compact/system-level models, reducing manual transfer between tools. [75]
In-scope responsibilities:
- Parametric geometry/layout representation for MEMS (component- and layout-based entry). [76]
- Two downstream compilation targets:
- Detailed FEM/BEM solves (field solvers) for multiphysics MEMS, consistent with CoventorWare being positioned as a field-solver suite. [77]
- Compact/reduced-order models for system/circuit integration (explicitly described as a goal of MEMS+ and CoventorMP conceptually). [78]
- Meshing optimized for MEMS geometries (CoventorWare materials describe multiple optimized mesh generators for MEMS solid models, including brick/extrude/tet options). [79]

Local replica blueprint
- High-fidelity path: CAD/mesh → FEM solve (MOOSE/FEniCSx class) → sensitivity extraction. [80]
- Compact-model path: apply MOR techniques and generate SPICE-compatible macromodels for system simulation (model-order reduction is a standard approach in system-level MEMS modeling literature). [81]

Embedded programmable logic IP and programming toolchain

Key vendor anchor: Menta [82] . [83]

The eFPGA description provides a crisp decomposition into (a) hardware fabric IP and (b) software to compile RTL into a fabric configuration/bitstream. [83]

Process contract: eFPGA fabric definition and integration
Purpose: provide embedded programmable logic for SoC/ASIC integration, as hard macro or customizable architecture, integrated into standard ASIC flows. [83]
In-scope responsibilities: define fabric primitives (logic blocks, routing, configuration memory, IO bank abstraction) and verification collateral for integration (timing models, DFT hooks are implied by "manufacturing DFT with full testability"). [84]
Out-of-scope: foundry-specific physical implementation details unless you have the PDK and legal rights; a replica can target open PDKs for experimentation. [85]

Process contract: RTL → mapping → place/route → bitstream
Purpose: generate a bitstream targeting the fabric from IEEE RTL (Verilog/VHDL/SystemVerilog), with mapping and place-and-route steps encapsulated in the programming tool flow. [84]
Local replica blueprint: build around open FPGA CAD flow components (Yosys for synthesis; nextpnr or VTR/VPR for P&R + timing) and substitute your fabric's architecture model + bitstream format. [86]

IC/package/PCB co-design, layout automation, and circuit optimization bounds

Key vendor anchor: Cadence Design Systems [87] . [88]

Package and PCB design platforms

Process contract: Multi-board PCB design with integrated constraints and collaboration
Purpose: provide a scalable, integrated environment for multi-board design and concurrent collaboration, with constraint-driven workflows and integrated analysis. [88]
In-scope responsibilities:
- Constraint system spanning schematic → layout → signoff checks, with real-time DRC-like enforcement. [88]
- Collaboration primitives: project/design data management, concurrent editing guards, and review workflows (explicitly emphasized as "global teams" and "concurrently collaborate"). [89]
Out-of-scope: exact equivalence to vendor-specific internal databases; a replica can implement an open data model (e.g., KiCad-like project model) as long as constraints and checks are consistent. [90]

Process contract: SI/PI analysis integrated into design
Purpose: power-aware SI/PI analysis, PDN analysis, and interconnect modeling for PCB and IC package designs, integrated with major design platforms. [91]
In-scope responsibilities:
- Extraction contract: build RLGC models from interconnect geometry. [92]
- Analysis contract: time/frequency domain simulations for channels/PDN, plus "in-design" feedback loops for early issue detection. [93]
Local replica blueprint:
- Geometry → field solver extraction (FEM/FDTD) → reduced-order models for fast sweeps → integration into layout UI. [94]

Layout automation add-ons

SkillCad is described as a layout automation suite integrating with Cadence Virtuoso, with "correct by construction" routing/commands to reduce DRC violations and accelerate productivity. [95]

Process contract: "Correct-by-construction" layout command library
Purpose: provide scripted/user-guided layout operations (autorouting-like commands) constrained to design rules to reduce violations and speed tape-out. [95]
In-scope responsibilities:
- A rule-query interface to the host layout system (DRC deck abstraction, spacing/width/via constraints). [95]
- Geometric construction algorithms that guarantee constraints (routing, jog insertion, via placement,

pattern fills). [96]
Local replica blueprint: build a scripting layer over an open layout representation; encode rules as constraints and use pathfinding + constraint satisfaction for routing operations. (For learning/prototyping, start with open EDA geometry tooling; production Virtuoso integration is proprietary.) [97]

Circuit sizing/optimization and verification automation

Key vendor anchor: MunEDA [98]. The pasted text describes a suite providing fast simulation-based sizing/tuning, sensitivity/gradient optimization, yield/high-sigma optimization, worst-case analysis, and multi-simulator parallelization. [99]

Process contract: Spec-driven circuit sizing and tuning
Purpose: automatically tune circuit parameters so that performance specifications and constraints are met, including variation and reliability considerations, with capacity to handle very large circuits and parallel simulation using industry SPICE simulators. [99]
In-scope responsibilities:
- Parameterization: map schematic/netlist elements into a parameter vector (with bounds, discrete grids for FinFET/FD-SOI style discretization implied by the suite's positioning). [99]
- Optimization core: deterministic nominal optimization + global/stochastic optimization + yield optimization; the slide explicitly lists constraint/feasibility optimization, deterministic nominal, global nominal, discrete optimization, yield optimization, and degradation/reliability-aware sizing. [99]
- Simulator interface: multi-testbench orchestration + parallel execution. [100]
Local replica blueprint: wrap ngspice/Xyce with a Python orchestration layer; implement spec evaluation functions, sensitivity estimation (finite diff, adjoint if supported), and sampling-based yield loops. [101]

Process contract: High-sigma variation and worst-case analysis
Purpose: accelerate yield verification across low → ultra-high sigma regimes using advanced sampling and worst-case search, while remaining "simulator-true." [102]
In-scope responsibilities: implement a portfolio of sampling strategies (importance sampling, scaled sampling, quasi-MC) and deterministic worst-case search that calls the simulator as ground truth. [103]

Manufacturing process simulation and EMC cable-harness simulation bounds

Key vendor anchors: Moldex3D [104] and Kitware [105] (re visualization substrate). [106]

IC packaging molding and encapsulation simulation

The Moldex3D IC Packaging description decomposes into: process physics (flow/cure/thermal/stress/warp), defect modes (voids, wire sweep, warpage), and meshing + interfaces. [107]

Process contract: Encapsulation process simulation
Purpose: simulate chip encapsulation processes (e.g., transfer molding, underfill variants) across phases: filling → curing → cooling, and predict defects like air traps/voids, wire sweep, paddle shift, and package warpage. [108]
In-scope responsibilities:

- Multiphase/transient flow in complex molds with thermoset cure kinetics coupling viscosity↔temperature↔degree-of-cure. [108]
- Structural/thermal stress and warpage prediction from cure shrinkage and thermal history (the Moldex3D page emphasizes stress distribution/structural evaluation and post-mold cure phenomena). [108]
- Wire sweep modeling during flow and comparison vs. original geometry. [109]
Data contracts: geometry model of package/mold, material property models (rheology, DSC cure kinetics, PVTC, DMA, etc. are listed as material test support categories). [110]
Local replica blueprint: couple CFD (OpenFOAM-class) + user-defined cure kinetics + structural solver (FEM) + meshing pipeline (Gmsh) + ParaView postprocessing. [111]

EMC/high-intensity fields and lightning coupling to cable harnesses

The EMA3D Cable description is explicit: a full-wave FDTD solver + an integrated multi-conductor transmission line solver that co-simulate, targeting EMC/HIRF/lightning at system scale. [112]

Process contract: Co-simulated FDTD + multiconductor transmission lines
Purpose: predict radiated coupling/emissions, shield coupling, crosstalk, current return path issues, and lightning/HIRF coupling for complex harnesses embedded in complex platforms. [113]
In-scope responsibilities:
- FDTD field solve on a 3D grid covering the platform geometry and materials. [112]
- Cable harness representation as multiconductor transmission line (MTL) networks embedded in the FDTD domain, with bidirectional exchange (fields induce cable voltages/currents; cable currents radiate back into fields). [112]
- CAD preprocessing robust to imperfect geometry and streamlined harness definition (described as “advanced CAD preprocessor” and fast meshing that “never fails” in the pasted text; your replica target is *tolerant preprocessing* + *structured grid generation*). [113]
Local replica blueprint: implement (1) structured-grid meshing/voxelization for FDTD, (2) an MTL solver (e.g., frequency-dependent RLGC per segment), and (3) a coupling interface (field sampling along cable paths + impressed sources). [114]

Assurance, reliability/safety engineering, and embedded infrastructure bounds

Key vendor anchors: LDRA [115] , Isograph [116] , Raima [117] , TenAsys [118] , and Ansys [119] . [120]

Safety-critical software verification tool suite

Process contract: End-to-end verification workflow integration
Purpose: integrate requirements traceability, static analysis, dynamic analysis, unit/integration/system testing, coverage, and tool qualification artifacts into one platform. [121]
In-scope responsibilities:
- Static analysis: coding standards checking, defect detection, and code metrics (LDRA explicitly frames static analysis in these terms). [122]
- Dynamic analysis: instrumentation, runtime execution data collection, and mapping results back to source, as part of coverage/traceability. [123]
- Requirements traceability: a requirements tool (TBmanager is identified as a component) linking

requirements → code → verification artifacts. [124]
- Tool qualification support packs: packaging of artifacts/guidance to simplify qualification in project environments. [125]
Local replica blueprint:
- Use a standards-driven data model: `Requirement ↔ TestCase ↔ CodeUnit ↔ CoveragePoint ↔ EvidenceArtifact`. [126]
- Bring your own static analyzer/coverage backends (LLVM tooling, gcov-like instrumentation) but keep the *traceability evidence model* first-class. [127]

Model-based development and certified code generation

From the embedded software product list: SCADE Suite (model-based development for critical embedded software, with requirements linkage, verification, and certifiable code generation), plus Display (HMI), Test (testing/coverage), and Architect (architecture modeling). [128]

Process contract: Formal model → verified implementation
Purpose: capture system behavior in models, simulate/debug, generate code automatically, and maintain traceability to requirements and safety standards. [129]
In-scope responsibilities:
- A model semantics engine (state machines/dataflow) with deterministic simulation. [128]
- Code generation pipeline with qualification artifacts ("qualifiable/certified automatic code generation" and "certification kits" are explicit). [128]
- Test and coverage integration (SCADE Test explicitly includes model and code coverage analysis). [130]
Local replica blueprint: for a clean-room baseline, implement a restricted synchronous/dataflow language subset (with a provable translation to C) and attach a traceability schema aligned to certification evidence needs. [131]

Reliability, availability, and safety assessment modeling

Process contract: Reliability Workbench-class analyses
Purpose: reliability block diagrams, fault tree analysis (including common cause/importance), event trees, Markov analysis (including multi-phase), FMEA/FMECA, Weibull analysis, reliability growth/allocation, and standards support. [132]
In-scope responsibilities:
- Graphical model editors for RBD/FTA/ETA plus a calculation engine (Boolean reductions, cut sets, Markov chain solvers). [133]
- Requirements/hazard log linking (explicitly mentioned as linking hazard logs/requirements to verification models). [134]

Process contract: Availability Workbench-class asset simulation
Purpose: maintenance/spares optimization, availability studies, RCM, life cycle cost, accelerated life testing, with direct connectors to SAP/Maximo and Weibull-based failure model building. [135]
In-scope responsibilities: discrete-event simulation of asset operations + maintenance policies + spare parts inventory states + cost penalties; connectors that import/export work orders and maintenance plans. [136]

Process contract: AttackTree and Hazop+
- AttackTree: threat analysis/risk assessment aligned to standards (ISO/SAE 21434, SAE J3061, ISO 26262,

etc.), mitigation trees, and links to requirements tools. [137]
- Hazop+: manage HAZOP study records, risk ranking matrices, action tracking, reporting, and import/export with common office formats. [138]

Local replica blueprint: express all of these as transformations over a common "assurance graph" model (nodes = components/functions; edges = dependencies; annotations = failure/attack modes; solvers = probability and consequence propagation). [139]

Embedded database manager and edge-to-cloud replication

Process contract: Embedded, in-memory database kernel with multi-API surface
Purpose: provide an embedded database for IoT/edge with in-memory performance and multiple APIs (C/C++ cursor API, SQL, JDBC/Java, REST), plus SQL features like stored procedures/triggers. [140]
In-scope responsibilities:
- Storage engine (in-memory + persistence), query engine, and API bindings. [140]
- Indexing methods: B-Tree, Hash, R-/R+ Tree, AVL are explicitly marketed as supported. [141]
- Cross-platform file compatibility is positioned as a feature (portable file format across architectures is claimed in product materials); a replica should define an endian-stable on-disk format and a compatibility test matrix. [140]

Process contract: Replication edge ↔ cloud
Purpose: replicate edge data outward using SymmetricDS integration/dialects. [142]
In-scope responsibilities: replication adapters (table mapping, conflict strategy, batching, offline/asynchronous behavior). SymmetricDS itself is described as replicating relational tables between databases, asynchronously, using a lightweight web protocol. [143]
Local replica blueprint: if you want a straightforward baseline, adopt SQLite-like embedding semantics (serverless, file-based) plus a replication layer (e.g., SymmetricDS-style) rather than attempting to mirror every proprietary API. [144]

Real-time OS and partitioned mixed-criticality execution

Process contract: Mixed Windows + RT workloads via partitioning/virtualization
Purpose: run real-time and general-purpose OS workloads side-by-side on the same multicore hardware using explicit partitioning/AMP and virtualization. [145]
In-scope responsibilities:
- CPU core, memory space, and I/O partitioning so that RT workloads have predictable timing; the INtime datasheet explicitly describes explicit partitioning enabling INtime and Windows instances to occupy separate cores/memory/I/O. [146]
- IPC between partitions (the datasheet describes INtime communicating with Windows via a DLL-based IPC mechanism). [147]
Local replica blueprint: implement using a partitioning hypervisor pattern such as **Jailhouse** (partitioning hypervisor based on Linux, splitting hardware into isolated "cells"). [148]
For RTOS baselines, Zephyr and FreeRTOS provide open RTOS kernels (different footprints and target domains). [149]

---

[1] [4] [5] [24] [47] [55] [105] [118] https://www.lamresearch.com/product/semulator3d/
https://www.lamresearch.com/product/semulator3d/

[2] [120] [121] [122] [127] https://ldra.com/products/ldra-tool-suite/
https://ldra.com/products/ldra-tool-suite/

[3] [25] [26] https://www.comsol.com/comsol-multiphysics
https://www.comsol.com/comsol-multiphysics

[6] [43] [44] https://www.comsol.com/ecad-import-module
https://www.comsol.com/ecad-import-module

[7] https://www.ansys.com/products/electronics
https://www.ansys.com/products/electronics

[8] [37] https://www.comsol.com/uncertainty-quantification-module
https://www.comsol.com/uncertainty-quantification-module

[9] [53] [56] [57] [58] [59] [60] [62] [63] [65] [66] [67] [70] [71] [72] [116] [119] https://www.europractice.stfc.ac.uk/media/pdf/coventor/coventor_semulator_datasheet.pdf
https://www.europractice.stfc.ac.uk/media/pdf/coventor/coventor_semulator_datasheet.pdf

[10] https://mooseframework.inl.gov/
https://mooseframework.inl.gov/

[11] https://fenicsproject.org/
https://fenicsproject.org/

[12] [33] https://gmsh.info/
https://gmsh.info/

[13] [111] https://www.openfoam.com/
https://www.openfoam.com/

[14] https://vtk.org/overview/
https://vtk.org/overview/

[15] https://www.paraview.org/
https://www.paraview.org/

[16] [101] https://ngspice.sourceforge.io/
https://ngspice.sourceforge.io/

[17] https://xyce.sandia.gov/
https://xyce.sandia.gov/

[18] [86] https://yosyshq.net/yosys/
https://yosyshq.net/yosys/

[19] https://github.com/YosysHQ/nextpnr
https://github.com/YosysHQ/nextpnr

[20] https://github.com/verilog-to-routing/vtr-verilog-to-routing
https://github.com/verilog-to-routing/vtr-verilog-to-routing

[21] https://sqlite.org/
https://sqlite.org/

22 31 32 80 https://mooseframework.inl.gov/sqa/framework_srs.html
https://mooseframework.inl.gov/sqa/framework_srs.html

23 88 89 https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis/allegro-x-design-platform.html
https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis/allegro-x-design-platform.html

27 36 39 https://www.comsol.com/optimization-module
https://www.comsol.com/optimization-module

28 29 41 https://www.comsol.com/cad-import-module
https://www.comsol.com/cad-import-module

30 https://www.comsol.com/blogs/how-to-import-an-ecad-geometry-for-printed-circuit-board-design
https://www.comsol.com/blogs/how-to-import-an-ecad-geometry-for-printed-circuit-board-design

34 45 87 https://doc.comsol.com/6.3/doc/com.comsol.help.ecad/IntroductionToECADImportModule.pdf
https://doc.comsol.com/6.3/doc/com.comsol.help.ecad/IntroductionToECADImportModule.pdf

35 https://www.comsol.com/particle-tracing-module
https://www.comsol.com/particle-tracing-module

38 https://doc.comsol.com/6.3/doc/com.comsol.help.particle/IntroductionToParticleTracingModule.pdf
https://doc.comsol.com/6.3/doc/com.comsol.help.particle/IntroductionToParticleTracingModule.pdf

40 64 https://www.comsol.com/support/learning-center/course/introduction-to-uncertainty-quantification-251/fundamentals-of-uncertainty-quantification-93621
https://www.comsol.com/support/learning-center/course/introduction-to-uncertainty-quantification-251/fundamentals-of-uncertainty-quantification-93621

42 115 https://www.comsol.it/blogs/category/interfacing/cad-import-and-livelink-products-for-cad/page/4/?setlang=1
https://www.comsol.it/blogs/category/interfacing/cad-import-and-livelink-products-for-cad/page/4/?setlang=1

46 https://doc.comsol.com/6.3/doc/com.comsol.help.ecad/ECADImportModuleUsersGuide.pdf
https://doc.comsol.com/6.3/doc/com.comsol.help.ecad/ECADImportModuleUsersGuide.pdf

48 49 74 75 76 78 https://siliconsemiconductor.net/article/101428/Coventor_Launches_Unified_Environment_for_Designing_MEMS_for_IoT_Devices
https://siliconsemiconductor.net/article/101428/Coventor_Launches_Unified_Environment_for_Designing_MEMS_for_IoT_Devices

50 98 https://www.comsol.com/video/intro-to-deploying-simulation-apps-with-comsol-server-mar-28-2018
https://www.comsol.com/video/intro-to-deploying-simulation-apps-with-comsol-server-mar-28-2018

51 52 https://cdn.comsol.com/doc/5.2/COMSOL_ServerManual.pdf
https://cdn.comsol.com/doc/5.2/COMSOL_ServerManual.pdf

54 83 84 85 https://www.design-reuse-embedded.com/ATT_FILES/temp/EFPGA_14_Datasheet_product_brief_-_efpga_core.pdf
https://www.design-reuse-embedded.com/ATT_FILES/temp/EFPGA_14_Datasheet_product_brief_-_efpga_core.pdf

61 https://www.sciencedirect.com/science/article/pii/S2352711025004194
https://www.sciencedirect.com/science/article/pii/S2352711025004194

68 69 https://www.asminternational.org/results/-/journal_content/56/10192/27517777/NEWS/?srsltid=AfmBOorm0O8lv2dF5VYRuiPqxW1NwpUsfpzkXWV92He4rCKYRVY44B53
https://www.asminternational.org/results/-/journal_content/56/10192/27517777/NEWS/?srsltid=AfmBOorm0O8lv2dF5VYRuiPqxW1NwpUsfpzkXWV92He4rCKYRVY44B53

73 https://en.wikipedia.org/wiki/Parasitic_extraction
https://en.wikipedia.org/wiki/Parasitic_extraction

77 https://www.ednc.com/wp/wp-content/uploads/2012/10/CoventorWare2012.pdf
https://www.ednc.com/wp/wp-content/uploads/2012/10/CoventorWare2012.pdf

79 https://micro.fel.cvut.cz/wp-content/uploads/2019/03/CoventorWare_10_Overview.pdf
https://micro.fel.cvut.cz/wp-content/uploads/2019/03/CoventorWare_10_Overview.pdf

81 https://download.e-bookshelf.de/download/0000/7534/38/L-G-0000753438-0002366618.pdf
https://download.e-bookshelf.de/download/0000/7534/38/L-G-0000753438-0002366618.pdf

82 128 130 131 https://www.ansys.com/products/embedded-software
https://www.ansys.com/products/embedded-software

90 https://docs.kicad.org/8.0/en/getting_started_in_kicad/getting_started_in_kicad.html
https://docs.kicad.org/8.0/en/getting_started_in_kicad/getting_started_in_kicad.html

91 92 93 94 https://www.cadence.com/en_US/home/tools/system-analysis/signal-and-power-integrity.html
https://www.cadence.com/en_US/home/tools/system-analysis/signal-and-power-integrity.html

95 96 97 https://skillcad.com/
https://skillcad.com/

99 100 102 103 https://www.coseda-tech.com/files/Files/Dokumente/Optimization_of_Complex_Circuits_at_System_Level_COSEDA_UGM_2018.pdf
https://www.coseda-tech.com/files/Files/Dokumente/Optimization_of_Complex_Circuits_at_System_Level_COSEDA_UGM_2018.pdf

104 135 https://www.isograph.com/software/availability-workbench/
https://www.isograph.com/software/availability-workbench/

106 107 110 117 https://picture.iczhiku.com/resource/eetop/sHktdgGDyJaoibcX.pdf
https://picture.iczhiku.com/resource/eetop/sHktdgGDyJaoibcX.pdf

108 109 https://ftdsolutions.com/menta-efpga/
https://ftdsolutions.com/menta-efpga/

112 114 https://www.ema3d.com/wp-content/uploads/2020/07/Virtual-Electromagnetic-Test-Environment-EMA.pdf
https://www.ema3d.com/wp-content/uploads/2020/07/Virtual-Electromagnetic-Test-Environment-EMA.pdf

113 https://optics.ansys.com/hc/en-us/articles/10072384186899-Head-Up-Display-HUD
https://optics.ansys.com/hc/en-us/articles/10072384186899-Head-Up-Display-HUD

123 https://ldra.com/nail-software-security-dynamic-analysis/
https://ldra.com/nail-software-security-dynamic-analysis/

124 126 https://ldra.com/capabilities/requirements-traceability/
https://ldra.com/capabilities/requirements-traceability/

[125] https://ldra.com/products/tool-qualification-support-packs-tqsp/
https://ldra.com/products/tool-qualification-support-packs-tqsp/

[129] https://www.ansys.com/products/embedded-software/ansys-scade-suite
https://www.ansys.com/products/embedded-software/ansys-scade-suite

[132] [133] [134] [139] https://www.cadence.com/en_US/home/tools/ic-package-design-and-analysis.html
https://www.cadence.com/en_US/home/tools/ic-package-design-and-analysis.html

[136] https://www.isograph.com/software/availability-workbench/maximo/
https://www.isograph.com/software/availability-workbench/maximo/

[137] https://www.isograph.com/software/attacktree/
https://www.isograph.com/software/attacktree/

[138] https://www.isograph.com/software/hazop/
https://www.isograph.com/software/hazop/

[140] [141] https://raima.com/wp-content/uploads/Technical-Whitepaper.pdf
https://raima.com/wp-content/uploads/Technical-Whitepaper.pdf

[142] https://raima.com/replication/
https://raima.com/replication/

[143] https://symmetricds.org/docs/faq/
https://symmetricds.org/docs/faq/

[144] https://sqlite.org/about.html
https://sqlite.org/about.html

[145] https://tenasys.com/overview/
https://tenasys.com/overview/

[146] [147] https://tenasys.com/wp-content/uploads/2021/08/TenAsys-INtime-RTOS-datasheet-07022024.pdf
https://tenasys.com/wp-content/uploads/2021/08/TenAsys-INtime-RTOS-datasheet-07022024.pdf

[148] https://github.com/siemens/jailhouse
https://github.com/siemens/jailhouse

[149] https://www.zephyrproject.org/learn-about/
https://www.zephyrproject.org/learn-about/