PyLCSS

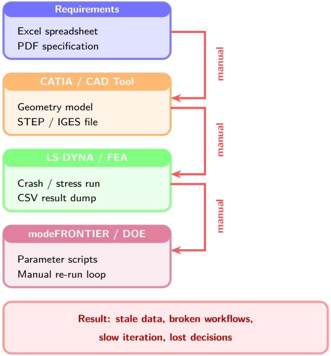
Integrated Product Development Environment

Kutay Demir

March 2026

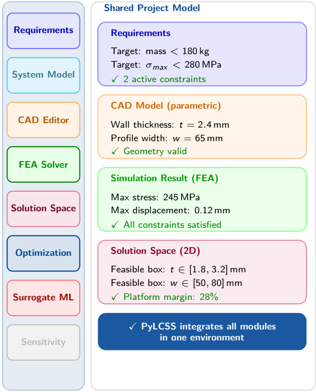
The Product Development Challenge

- Product teams use many separate tools for each step.
- Requirements, CAD, simulation, and optimization are disconnected.
- Manual handoffs slow iteration and degrade decision quality.
- *Example: a car body is designed in CATIA, crash-simulated in LS-DYNA, and optimized in modeFRONTIER — each with separate file exchanges.*
- **PyLCSS connects all these steps in one environment.**



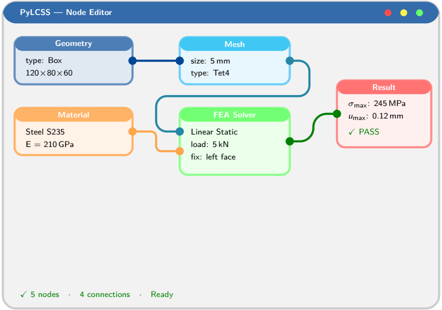
Core Idea of PyLCSS

- One integrated environment for product development.
- Connects requirements, models, CAD, simulation, and decisions.
- Combines robust design and optimization in one workflow.
- Low-code: advanced methods usable without programming skills.



Why Low-Code Matters

- Most engineers understand physics and products better than code.
- Visual node-based workflows lower the barrier to entry.
- Advanced methods become accessible to the whole team.
- Companies can involve more people in digital development.



PyLCSS Workflow

1. Define

Requirements & System Model

- System architecture
- Functional requirements
- Load cases & constraints
- Interface definitions

→

2. Parameterize

Variables & Objectives

- Design variables x_i
- Objectives $f(x)$
- Bounds & tolerances
- Sensitivity screening

→

3. Evaluate

Simulation / Black-box calls

- PyLCSS CAD + FEA
- External Python APIs
- Surrogate models (ML)
- DOE sampling

→

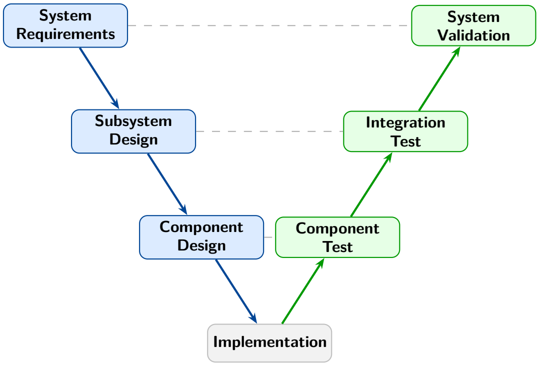
4. Decide

Solution Space or Optimum

- Feasible box search
- Pareto front / NSGA-II
- Robust design margin
- Platform intersection

V-Model for Product Development

- The **V-model** mirrors each development step with a verification step.
- PyLCSS supports every level: from system requirements to component tests.
- Simulation and solution spaces provide **early validation** before prototyping.
- Continuous feedback loops reduce late-stage rework.



PyLCSS covers all levels

What PyLCSS Covers

Analysis & Decision Methods

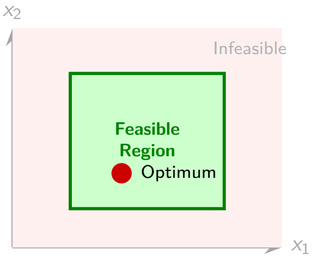
- Solution spaces and optimization
- Product families and cost reduction
- Surrogate models for expensive functions
- Sensitivity analysis and variable reduction

Engineering Environment

- CAD, FEA, topology optimization, crash simulation
- Black-box integration via Python API
- Voice assistance and accessibility

Solution Space and Optimization

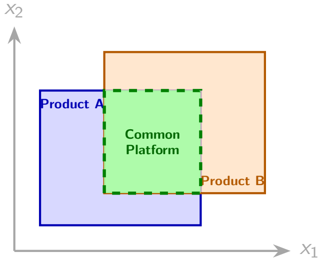
- PyLCSS finds a **feasible region**, not only one point.
- Any design inside the box satisfies the given requirements.
- Supports tolerance-robust and decoupled subsystem design.
- Optimization can find the best candidate inside the region.



- *Car: crash-box - wall thickness x_1 and profile width x_2 must jointly satisfy energy-absorption and mass constraints.*
- *Aircraft: wing-spar - flange thickness x_1 and web height x_2 bounded by fatigue life and weight limits.*

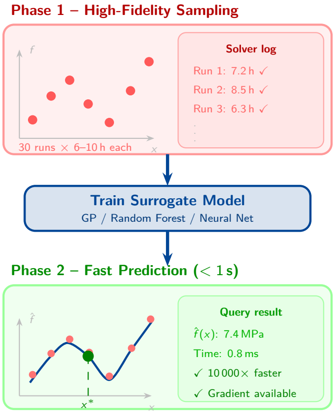
Product Families and Cost Reduction

- Compare solution spaces across product variants.
- The overlap reveals a **common platform region**.
- Drives modular design and part sharing.
- Reduces development, manufacturing, and supply-chain cost.
- *Car: VW MQB platform, one floor pan shared across Golf, Tiguan, and Octavia.*
- *Aircraft: Airbus A320 vs. A321, wing structure shared, fuselage length varies.*



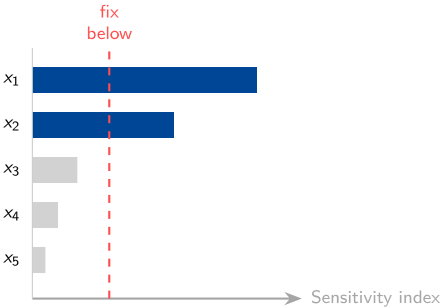
Surrogate Training for Expensive Models

- Some functions take minutes or hours per evaluation.
- *Car: a full-vehicle crash simulation in LS-DYNA takes 6–10 h per run.*
- *Aircraft: a CFD analysis of a wing profile at transonic speed takes hours on an HPC cluster.*
- Train an ML surrogate on sampled high-fidelity results.
- Replace the expensive call with a near-instant prediction.



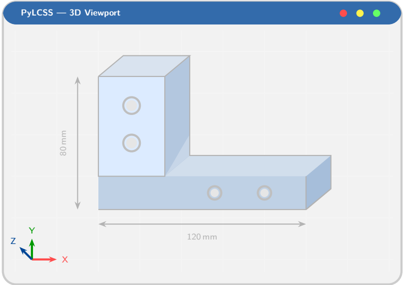
Sensitivity Analysis

- Identifies which variables most strongly influence the output.
- Variables below the threshold can be fixed.
- Reduces the active design space and improves solver efficiency.
- *Car: engine mount with 12 variables — only 4 dominate NVH response.*
- Supported methods: Sobol, Morris, FAST, Delta.



CAD and Simulation Environment

- **Parametric CAD:** 60+ node types — primitives, sketching, booleans, patterns, assembly
- **Import:** STEP and STL for FEA or topology optimization
- **FEA:** linear structural analysis — stress, displacement, compliance
- **Topology opt.:** most material-efficient structure for a load case
- **Crash simulation:** impact and energy-absorption studies
- **Export:** STEP, STL, OBJ for downstream tools



Black-Box Integration with External Tools

- Any tool with a Python API can be used as a black-box function.
- Examples: ANSYS, MATLAB, LS-DYNA, multibody solvers, in-house tools.
- PyLCSS passes design variables in and receives results back.
- It does not replace existing solvers — it **orchestrates** them.

black_box_wrapper.py

```
def evaluate(x):
  # call external tool via API
  result = FEASolver.run(x)
  # or:  matlab.feval(...)
  # or:  subprocess ANSYS
  return result["stress"],
      result["mass"]
```

Execution Log

- ▷ Run 1: $x = [2.4, 65]$ → calling FEASolver...
- ✓ Run 1: stress=245 MPa, mass=171 kg (0.8 s)
- ▷ Run 2: $x = [2.1, 72]$ → calling FEASolver...
- ✓ Run 2: stress=268 MPa, mass=164 kg (0.7 s)
- ▷ Run 3: $x = [3.0, 58]$ → calling MATLAB...
- ✓ Run 3: stress=201 MPa, mass=183 kg (1.1 s)

Results Summary (passed to Solution Space / Optimizer)

Run#	t [mm]	w [mm]	σ [MPa]	m [kg]
1	2.4	65	245	171
2	2.1	72	268	164
3	3.0	58	201	183

✓ 3 evaluations complete · Feeding optimizer...

Use Cases for Companies

- Requirement-driven concept development
- Product-family and platform design
- Lightweight and structural design
- Surrogate-supported engineering studies
- Safety and crash-oriented analysis
- Orchestrating existing tools via Python
- Reusable digital development workflows

Industry Examples

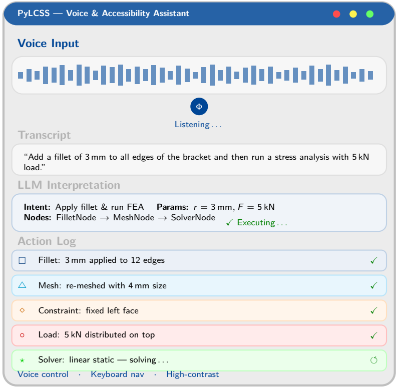
- *Car:* platform strategy (VW MQB), crash-box optimization, NVH surrogates, topology opt. for chassis parts
- *Aircraft:* wing-structure sizing, pylon-bracket topology opt., surrogate CFD, A320/A321 shared structures

Value for Companies

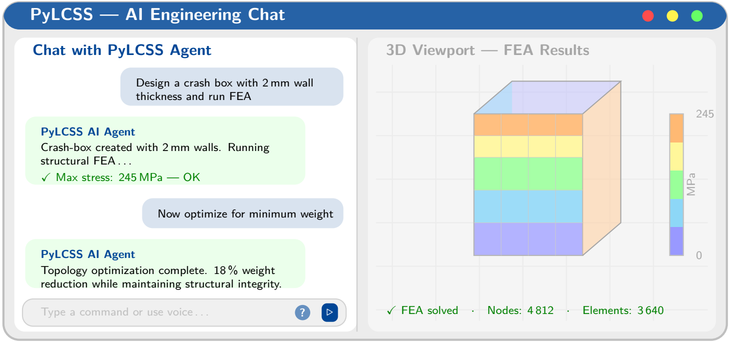
- One environment instead of disconnected tools
- Faster iteration from requirements to evaluated design
- Better decisions through robust-design thinking
- Lower cost through commonality and reduced effort
- Accessible to engineers without a coding background
- Source-available, modifiable, no license cost
- Integrates with existing company solvers

Accessibility and Voice Assistance

- Speech-to-text converts voice commands into engineering actions.
- An LLM interprets intent and controls workflows directly.
- Supports engineers with physical disabilities.
- Accessibility is a design principle of PyLCSS, not an add-on.



Future Vision: AI-Supported Engineering



AI agents plan and execute full development cycles, from product idea to validated concept.

PyLCSS

Integrated Product Development Environment

Faster decisions. More robust products. Accessible engineering.

Kutay Demir · March 2026