Облачная параметрическая CAD-система RECADA / Airholder — полное описание проекта (v30)

Идея · архитектура · файлы и форматы · рабочий процесс проектирования · конфигуратор (Unity) · прототипирование · испытания ISO 11154 · аксессуары · наём в разных странах и кадры · жизненный цикл · ISO 9001 · защита и VDI · инфраструктура и выбор серверов · инструмент реверса (recada.py) · редактор изделия · автоматизация · финансы · управление · риски · дорожная карта — Сентябрь 2026

Легенда: термины и сокращения

Короткий словарь для чтения документа. Каждый термин объяснён одной строкой — подробности в соответствующих разделах.

Термин	Что это простыми словами
JSON	Текстовый формат данных «поле: значение» — язык параметров системы (params.json, паспорта); правится блокнотом
YAML	«JSON для людей» — тот же смысл, но читается как список с отступами; у нас — язык инструкций по монтажу
spec.yaml / SDD	Спека задачи до геометрии: зачем деталь, с чем стыкуется, проверяемые условия приёмки, чего не делаем; CI строит из неё приёмочный прогон; основа сдельной оплаты (Spec-Driven Development)
Python / build123d	Язык программирования / библиотека, на которой написаны генераторы — код, строящий геометрию из параметров
Генератор (generator.py)	Скрипт, читающий params.json и паспорт и строящий деталь; читается как техпроцесс
Паспорт (vehicle/accessory passport)	JSON с точками крепления, углами и поверхностями машины или аксессуара — всё, что нужно знать о чужом объекте
Скелет / MasterParams	Каркас из точек и осей с главными параметрами (высота, углы) — мост между машиной и изделием
Git	Система контроля версий: полная история «кто, что, когда и зачем изменил»; основа всей работы
Gitea	Наш собственный веб-сервер для Git (аналог GitHub, но на нашем железе): репозитории, права, ревью, канбан
Репозиторий (repo)	Папка проекта под управлением Git со всей историей
Коммит (commit)	Одно сохранённое изменение с обязательной причиной («зачем») в сообщении
Дифф (diff)	Наглядная разница между двумя версиями файла: что добавлено, что убрано, строка за строкой; «читаемый дифф» = изменение видно глазами (height: 60 → 80), поэтому истина хранится текстом
Ветка (branch)	Параллельная линия работы для экспериментов; main — главная, защищённая
PR (Pull Request)	Запрос «принять мои изменения в main» — точка ревью и подписи; в main нельзя писать напрямую
Тег (git-tag) / Релиз	Именованная метка версии; релиз детали = подписанный тег, только из него идёт производство
LFS (Large File Storage)	Расширение Git для тяжёлых файлов: в истории — лёгкий указатель, тело — в хранилище рядом; для сканов, STEP, отчётов
zippey	Фильтр, позволяющий Git читать содержимое .FCStd (FreeCAD) и показывать осмысленные диффы

Термин	Что это простыми словами
CI (Continuous Integration)	Серверный конвейер: по коммиту/тегу сам собирает геометрию, гоняет проверки и выкладывает артефакты
Артефакт	Файл, построенный CI из истины (STEP, GLB, PDF, BOM): не редактируется — пересобирается
Валидатор / DFM	Автопроверка «производимо ли»: правила гибки Blechcon, зазоры, упаковка, крепёж; красный = стоп
B-Rep / STEP-топология	Способ хранения точной геометрии: тело → оболочки → грани → рёбра → вершины; из него и извлекаются размеры при обмере
AAG (граф смежности граней)	Граф «грань → соседняя грань»: отсечение торцевых граней расщепляет деталь на верхнюю и нижнюю оболочки — основа автоматического распознавания гибов
Скелет детали / Medial Axis	Срединное представление без толщины: пластины + линии гиба + толщина одним параметром; результат обмера и основа параметризации
FTG (граф переходов гибки)	Свёртка AAG: гиб = ребро графа, фланцы = узлы; объединяет соосные гибы, разорванные разгрузками, в один технологический переход
BA / BD (припуск и вычет)	Формулы длины развёртки через K-фактор и нейтральную ось (ANSI/DIN) — считаются аналитически, без CAD
Kerf (ширина реза)	Металл, съедаемый лазером/плазмой; закладывается в раскрой, иначе деталь выходит меньше номинала
Разгрузка (Entlastungsschnitt / relief)	Вырез у края гиба, чтобы металл не рвался при гибке; в системе вычисляется правилом по t/R, а не копируется из оригинала
Пружинение (springback)	Металл после гибки частично распрямляется; у нержавейки заметно — угол на прессе задают с запасом
FEM / CalculiX / Gmsh	Расчёт прочности и собственных частот (модальный); Gmsh строит сетку, CalculiX считает — бесплатные
CFD / OpenFOAM	Расчёт обтекания воздухом (аэродинамика); тяжёлый, применяется разово вне конвейера
STEP / DXF	Обменные форматы точной геометрии: STEP — 3D для Blechcon и от OEM, DXF — плоские развёртки и сечения
STL / GLB	Сетки из треугольников без точных размеров: STL — печать прототипов, GLB — показ в конфигураторе
.FCStd	Файл FreeCAD; у нас двойной статус: скелеты руками = истина, просмотровые сборки от CI = артефакт
BOM (Bill of Materials)	Спецификация: из чего состоит изделие, массы, цены — генерируется CI
catalog.json	Контракт между CI и конфигуратором: список деталей со статусами, ссылками на GLB и правилами
SKU / Rev (ревизия)	Артикул детали / версия конструкции внутри артикула (Rev A, B…)
FFF (Form-Fit-Function)	Правило: посадка и функция не изменились → новая ревизия того же SKU; изменились → новый SKU
VDI / Kasm	FreeCAD в браузере: программа работает на нашем сервере, человеку летят только пиксели; файлы не покидают сервер
Контейнер / Docker	Изолированная «коробка» с одним приложением; VDI-сессия = одноразовый контейнер
WireGuard / VPN	Шифрованный туннель в нашу сеть; ключ = допуск, отзыв ключа = мгновенное отключение человека
2FA / TOTP	Второй фактор входа: код из приложения на телефоне
Webhook	Автоматический сигнал одной системы другой («остаток = 0») — триггер для скриптов CI

Термин	Что это простыми словами
LLM / Claude / MCP	Языковая модель как помощник; MCP — протокол, через который Claude управляет FreeCAD и инструментами
Промпт	Задание модели обычным языком («поднять платформу на 40 мм»); меняет параметры, не геометрию напрямую
GDPR / GeschGehG / StVZO / ISO 11154	Регуляторика: защита данных ЕС / закон ФРГ о коммерческой тайне / правила допуска ТС / стандарт испытаний багажников
CM1 / CM2	Маржа изделия / маржа с учётом стоимости разработки — уровни финансовой модели
CPQ	Configure-Price-Quote: инструмент дилера «собрал конфигурацию → цена → КП → заказ в производство»

1. Идея и исходная ситуация

RECADA (Аугсбург) производит автомобильные багажники Airholder: платформа из алюминиевых экструдированных профилей (профили на складе, сечения нарисованы один раз) + гнутые ноги из нержавеющей стали 1.4301 (2–3 мм); резка и гибка — аутсорс (Blechcon.de). Узкое место — ноги под каждую машину и аксессуары. Цель: система, где на входе крыша (STL-скан с точками крепления или STEP от OEM), на выходе — производственные файлы, обновлённый конфигуратор и инструкции; дизайн управляется по промпту; работа командная с версионированием; знания накапливаются; исполнители — удалённые инженеры из разных стран, отбираемые по результатам тестовых работ.

2. Архитектурный принцип и слои

Источник истины — параметры и код, а не геометрия. Файлы FreeCAD/STEP — артефакты сборки. Это даёт промпт-управление (LLM меняет числа в JSON, не двигает грани), читаемые Git-диффы, командную работу без конфликтов бинарников, автоматическую пересборку производных.

Слой	Содержание
1. Паспорт автомобиля	JSON из STL/STEP (один раз): точки крепления с нормалями, углы, межосевые; для деталей, повторяющих крышу (windshield), — опорные поверхности как сплайн-аппроксимация (не сырая сетка); поле точности geometry_source. Сырой скан — в архив, роль: контрольный эталон для проверки зазоров.
2. Скелет + MasterParams	Точки, оси, плоскости, главные параметры (H, alpha). Всё «твёрдое» читает размеры отсюда. Скелет — мост между паспортом машины и паспортом изделия. Железное правило скелетного метода: детали никогда не ссылаются друг на друга — только на скелет (иначе связи рвутся непредсказуемо); направление зависимостей строго сверху вниз: MasterParams → Skeleton → детали.
3. Каталог профилей	Сечения один раз (импорт DXF); параметр — длина резки; «ширина от длины» — правило.
4. Генераторы (build123d)	Код строит детали от скелета; материал, R3, покрытие зафиксированы; библиотека «кубиков» (bent_flange, hole_array, bolt_joint) прячет сложность.
5. Валидатор DFM+	Правила Blechcon (отгибы, радиусы, развёртка, коллизии) + зазор до крыши ≥20 мм (статический запас вместо расчёта деформации кузова) + упаковка по лимитам канала доставки (текущий канал GLS: ≤40 кг/коробка — главный лимит, масса считается по коробкам комплекта с раскладкой в BOM; длина до ~2 м практически, пороги надбавок — из тарифа GLS в rules/) + крепёж (таблица болт/гайка, проверка зацепления). Красный — артефакты не собираются. Физическая калибровка правил гибки (FEA-in-the-loop): CalculiX симулирует сам процесс гиба и показывает, при каком радиусе напряжение на внешнем волокне грозит микротрещинами у 1.4301; расчёт делается один раз для сетки «толщина × радиус × марка» → таблица допустимых R в dfm_blechcon.json → валидатор смотрит в неё мгновенно. Так справочное «R ≥ 1×t» заменяется на подтверждённое расчётом и практикой поставщика. Проверка против реальной оснастки цеха (CAM-in-the-Loop в практичной форме): у Blechcon запрашивается список доступных пуансонов и матриц (радиусы, ширина V-паза, максимальная длина гиба) и заводится таблицей в dfm_blechcon.json — валидатор отвечает «R2.5 нет, ближайший R3, пересчитываю K-фактор» и проверяет минимальную длину фланца, чтобы деталь не проваливалась в матрицу, ещё до отправки заказа. Генерация G-кода для их пресса не входит в систему — станок программирует поставщик.
6. CI-сборка	STEP/DXF → Blechcon (перед заказом автоматический раскрой: SheetNest / nest2d / CAD-N, для больших партий u-nesting на Rust с Python-интерфейсом — отходы, проверка счёта, учёт ширины реза kerf из dfm_blechcon.json); GLB → конфигуратор; студийные рендеры (Blender headless) → Shopify; PDF/HTML → инструкции; BOM → цены и склад.

```
recada-cad/
├── vehicles/     паспорта машин          ├── products/     конфигурации (params.json)
├── catalog/      профили, крепёж         ├── accessories/  аксессуары и адаптеры
├── rules/        DFM, правила            └── ci/           пайплайны сборки
├── generators/   build123d + кубики
```

Версионирование: Gitea (self-hosted) + Git LFS; .FCStd — через zippey (читаемые диффы). Релиз = git-тег с персональной подписью ревьюера. Новая машина = паспорт + params.json — механизм масштабирования.

3. Файлы и форматы: как устроена истина

Никакого собственного формата: истина детали = два обычных файла. params.json — числа и слова (SKU, машина, материал, высота, отверстия), правится блокнотом/промптом. generator.py — Python/build123d, читается как техпроцесс: «лист 120×340×3 → скруглить углы → отверстия по точкам паспорта → фланец 90° R3 → нижний гиб под угол крыши из паспорта». Числа приходят из трёх источников (params / паспорт / каталог материалов), в коде их почти нет. FCStd — не источник, а результат (как .exe при исходниках): CI собирает его для просмотра. Исключение: скелеты и эталоны реверса, построенные руками, — полноценные FCStd-исходники в Git.

Дерево построения не исчезло — оно переехало в текст: порядок строк = порядок операций, но вместо хрупких ссылок на номера граней (Topological Naming Problem) — устойчивые правила (edge="top", точки по именам) и порождающие конструкции (циклы по точкам паспорта, условия). Плюс история с «почему»: каждый коммит несёт причину/промпт («поднять платформу 40 мм — бокс задевает антенну»), чего нет ни в одном GUI-дереве; это же — QMS-след и обучающие данные системы.

Сборки: тот же принцип уровнем выше

Сборочный JSON описывает состав и привязки, но не координаты: детали публикуют именованные интерфейсы («sole», «head» с плоскостью, отверстиями, нормалью), сборка говорит «посадить sole на mount_front_L». Вместо решателя констрейнов (Inventor) — детерминированное вычисление позиций из интерфейсов: ничего не «желтеет» при изменениях. Крепёж — свойство соединения: bolt_joint() сам ставит болт по оси, считает длину из пакета, кормит BOM. Промпт «поднять платформу на 40 мм» меняет один параметр скелета — всё пересобирается. Иерархия: деталь → сборка изделия → конфигурация клиента, один формат и конвейер. Цена подхода: нет свободной кинематики (шарниры — позой-параметром или в песочнице).

Роли форматов и «труба»

```
JSON+Python (истина) → STEP/FCStd (точная геометрия) → DXF (развёртка) → STL/GLB (сетки)
Вниз — всегда, автоматически. Вверх — только реверс-инжиниринг (дорого, один раз, на входе).
```

STEP — «PDF мира CAD»: точный, но мёртвый; выход к Blechcon и вход от OEM. STL/GLB — сетки без размеров: печать и показ, никогда для производства и измерений. У каждого производного файла есть родитель-истина (SKU+ревизия в имени); файл-сирота — нарушение процесса. При конфликте прав всегда верхний по трубе, нижние пересобираются. Разнородные исходники (скан машины + чужой STEP багажника) никогда не стыкуются напрямую — каждый переводится в паспорт-JSON, стыковка на языке паспортов; ноги — материализованный мост между двумя паспортами.

FFF-правило (Form-Fit-Function)

Посадка и функция не изменились → новая ревизия того же артикула (взаимозаменяемо, склад смешивает). Изменились → новый SKU с политикой замены. Защищает от зоопарка артикулов и от тихой несовместимости. Исполняется машиной: валидатор сравнивает именованные интерфейсы между ревизиями; «функция сохранена?» — единственный пункт, остающийся подписи человека.

Справочник типов файлов: что лежит в Git и как с этим работать

Тип файла	Что это	В Git?	Кто и как правит
params.json	Параметры детали/изделия: SKU, машина, материал, размеры, статусы; рядом — spec.yaml: намерение, интерфейсы, ограничения приёмки, non-goals (контракт задачи)	Да, обычный текст — главный источник истины	Инженер (редактор/веб-форма), LLM по промпту; каждое изменение — коммит с причиной; в main только через PR
generator.py и recada_lib	Python/build123d: код построения геометрии; библиотека кубиков (bent_flange, hole_array, bolt_joint)	Да, обычный текст — источник истины	Senior/владелец (+LLM с ревью); пробы в ветке convert/*, squash-merge при приёмке
Паспорта: vehicle_*.json, accessory_*.json	Точки крепления с нормалями, углы, опорные поверхности (сплайны), интерфейсы, масса/ЦТ, geometry_source с точностью	Да — источник истины	Создаются один раз при оцифровке (человек или ИИ-агент, подпись человека обязательна); уточняются при пилотных установках
Сборочные JSON	Состав и привязки по именованным интерфейсам (без координат)	Да — источник истины	Инженер/LLM; те же PR-правила
rules/: DFM, материалы, тарифы	Правила Blechcon, materials.json (1.4301, R3), лимиты GLS, таблица крепежа, пороги FEM	Да — источник истины	Владелец/lead; пополняются из рекламаций и ревью технолога Blechcon (цикл CAPA)
YAML инструкций	Шаги монтажа: деталь, крепёж, момент затяжки	Да	Инженер; CI рендерит PDF/HTML/Unity-сцены сам
.FCStd (FreeCAD)	Два разных случая! (а) Скелеты и эталоны реверса — построены руками; (б) просмотровые сборки — построены CI	(а) Да, через zippey (читаемые диффы) + LFS; (б) НЕТ — артефакт	(а) Владелец/senior в FreeCAD; (б) не редактируется никогда — пересобирается
STEP / DXF	Точная «мёртвая» геометрия: обмен с Blechcon и приём от OEM/исполнителей	Входящие — архив (LFS) как сырьё; исходящие — НЕТ, артефакты CI с SKU+ревизией в имени	Не редактируются; исходящие уходят Blechcon только из CI по тегу; входящие — через извлечение в паспорт
STL / GLB	Сетки без размеров: печать прототипов (STL с ID получателя), конфигуратор (GLB огрублённый)	НЕТ — артефакты CI	Не редактируются; для производства и измерений непригодны принципиально
Сканы крыш (STL), STEP от OEM	Сырьё оцифровки	Архив в LFS (один раз)	Не редактируются; роль — контрольный эталон зазоров и источник паспорта
BOM, catalog.json, рендеры, PDF-инструкции	Производные для склада, конфигуратора, клиентов	НЕТ — артефакты CI (хранилище артефактов; релизные — вечно)	Не редактируются; изменения — только через правку истины и пересборку
Отчёты испытаний и FEM	Протоколы физических испытаний, эталонные расчёты (статика, модальный)	Да (LFS) — прикрепляются к релизному тегу	Только добавляются; связь «ревизия ↔ протокол» = QMS-след

Три железных правила для всей команды: (1) редактируется только истина (JSON, .py, паспорта, rules, YAML, скелеты) — всё остальное пересобирается, файл-артефакт руками не трогается никогда, даже «быстро поправить»; (2) у файла без SKU и ревизии в имени нет права покидать сервер; (3) при расхождении версий прав всегда тот файл, что выше по трубе истины — нижние пересобираются, а не чинятся. Нарушение правила 1 (ручной STEP в производство мимо params) — главный риск смерти системы, поэтому отправка на Blechcon технически возможна только из

CI-артефактов релизного тега.

4. Как проектирует человек

Принцип «шаблон, не копия»: дерево/генератор строится из конструктивного замысла (база → полки → гибы → крепёж, каждый размер осмыслен и привязан к параметру), а не как копия всех артефактов конкретной штампованной детали — копия хрупка и ломается при больших изменениях, шаблон держит. Путь 1 — новый тип детали: творческий поиск руками в песочнице FreeCAD (мышкой, как привык) → готовая конструкция становится измеримым эталоном (объём, масса, точки поверхности) → перевод в код по шагам (контур → сверка → гибы → сверка → отверстия → сверка), каждая попытка автоматически сверяется с эталоном; пробы живут в отдельной ветке и сплющиваются в один коммит при приёмке; 3 неудачи LLM подряд → эскалация человеку. Путь 2 — повседневный: правка чисел в params (редактор/форма/промпт) → пересборка → просмотр; CAD превращается в место просмотра. Путь 3 — LLM-переводчик: человек описывает словами/эскизом, Claude пишет build123d по образцам библиотеки, CI собирает, метрология возвращает расхождение, LLM чинит.

Классы деталей по стоимости перевода

Класс	Перевод в систему	Трудозатраты
Профили (алюминий, прокат)	Импорт DXF-сечения + extrude; параметр — длина	час на каталог
Гнутый лист (ноги, адаптеры)	Реверс техпроцесса в код; быстрее с библиотекой кубиков	часы–дни
Литые/скульптурные	НЕ переводить: паспорт интерфейсов (20 мин) + STEP как мёртвый референс; офсеты от опорных поверхностей при необходимости; полный реверс — только если планируется семейство размеров	минуты
Покупное (крепёж)	Библиотеки стандартных деталей build123d	0

Правило границы внутри детали: форма — руками (один раз), интерфейсы и адаптация — кодом. Люди в системе остаются для четырёх ролей, где машина слаба: постановка задачи (знание реального мира), топология («как вообще решить»), суждение («валидно, но глупо» — подпись ревьюера), обучение системы (опыт → правила). Час «обвести контур» система обнуляет; час «понять, как обойти рейлинг» становится вечным активом.

Вход открыт для любого CAD: исполнитель рисует в SolidWorks/Inventor/Catia/FreeCAD — сдаёт STEP + чертёж с размерами; перевод в код не зависит от исходного CAD. Требовать build123d от конкурсантов = сузить пул вдесятеро. Лицензии: проверить чужую лицензию невозможно — декларация в договоре переносит ответственность, а VDI-сессия снимает проблему полностью (легальное рабочее место без единой лицензии у исполнителя) и очищает цепочку поставок для ISO/GeschGehG. Субподряд безопасен по построению: оплата за деталь, прошедшую валидатор; аккаунт персонален (сессии из двух стран видны в логах); на лестнице доверия субподрядчик не растёт.

Инженерный задел: что уже проверено практикой

Реверс-инжиниринг с метрологической приёмкой (point-sampling по контрольным точкам): Bracket_L — 100% совпадение; Bracket_R — зеркало, 99.12%; wind_side_v2 — 100.000% по 88 точкам; Part55 → новый Bracket_88_W30 с полным DFM-контролем; узел Defender 110 (LRDE-2BR1) — 99.9994%, SheetMetal-дерево от скелета. Материал везде 1.4301 2B, t=3.0, K-фактор из таблицы. Критический урок сессий: установка networkx включила SheetMetal Unfolder V2, который показал истинные углы гиба 99.389° и 135° вместо ошибочно измеренных V1 80.6° и 45° (дополнительные углы) — ошибка с прямыми производственными последствиями; правило: развёртки только через V2. Словарь конструкции зафиксирован: полка (Flansch) — боковые грани с отверстиями, стенка (Steg) — центральная грань с логотипом.

Выученные технические правила (в skills): SMBendWall падает на прерванных гибах → Revolution или реконструкция из face-призм; extend задавать до навешивания дочерних стенок (иначе TNP рвёт привязки); упавший конструктор оставляет null-объект — удалять немедленно; boolean cut ненадёжен для верификации — только point-sampling через distToShape; перед поиском кромок — полный touch()+recompute (кэш SheetMetal не сбрасывается сам); привязки только через SubShapeBinder (защита от TNP); для деталей с формовками — двухконфигурационное дерево (плоская модель для развёртки + финальная с формовкой). Модули к установке: SheetMetal 0.8.22 (Solid-to-SheetMetal, folding marks для пресса Blechcon, индикация битых ссылок), Defeaturing WB (срезать фаски в зоне гиба → развернуть → вернуть вырезы), Curves WB (чистка B-Spline-граней из STEP Inventor), networkx. Материалы — Material Definition Sheet (таблица R/T → K-фактор вместо жёсткого K=0.42): одна таблица питает и Claude, и параметрические модели. Unattended Unfold — headless-развёртка для CI. Точка входа репозитория — CLAUDE.md: описание изделия, материалы по умолчанию, какие skills когда применять.

Валидатор с автопочинкой: следующая ступень DFM

Правила оцифровываются в три слоя: числа → dfm_blechcon.json (R=1×t, мин. полка 15 мм для нержавейки, отступ отверстия от гиба, Entlastungsnut, зазоры ≥ t, лимиты листа, GLS-габариты); исправления → features.py (генераторы разгрузок: овал, Entlastungsnut, угловая разгрузка); методики → skills (markdown: «как думать»). Валидатор не только говорит «ошибка: отверстие близко к гибу», но предлагает исправление: «конфликт → предлагаю relief_obround, вот превью» — в RECADA-workbench это красная подсветка + кнопка «исправить», в CI тот же код headless блокирует выпуск неисправленного. Калибровка библиотеки — по реальному производству: принятые Blechcon детали с их правками разгрузок обмеряются (простой реверс), формулы «размер разгрузки от t/R/d» зашиваются в features.py; каждое письмо-исправление от Blechcon — строка в dfm_blechcon.json, и ошибка не повторяется ни у кого. Дизайн по правилам перестаёт зависеть от того, помнит ли инженер PDF поставщика.

Реверс без GUI: STEP → Python-код (метод и стек)

Дерево построения как таковое не нужно — нужен Python-код: он хранит ту же последовательность операций, но без хрупких ссылок на грани (TNP), с циклами и условиями, диффами и историей «зачем». Просмотровое дерево CI может собрать для глаз, но истина — код. Вся труба реверса работает headless (build123d/OCP — то же ядро OpenCASCADE, что у FreeCAD): без GUI, без MCP-сессий, повторяемо и пригодно для CI (десять STEP за ночь).

Алгоритм автоматического обмера (step_probe.py): обход B-Rep через TopExp_Explorer → классификация граней (плоские = полки/стенки, цилиндрические = гибы, узкие торцевые = толщина) → построение AAG (атрибутированный граф смежности граней, networkx): узлы — грани, рёбра — сопряжения → отсечение торцевых граней расщепляет граф на две изолированные эквидистантные оболочки (верх/низ), их сопоставление даёт нейтральную поверхность и толщину t → извлечение параметров гибов (радиус, угол, ось; numpy: нормали, PCA осей), отверстий (цилиндры с осями и D), внешнего контура. Выход — measurements.json.

Скелет детали — главный результат обмера и основа параметризации

Ключевое решение: обмерщик выдаёт не «список размеров», а СКЕЛЕТ детали — срединное представление без толщины: пластины (плоскости фланцев с контурами) + линии гиба (оси, углы, радиусы) + толщина как ОДИН параметр (t = 2× расстояние от срединной поверхности до

границы). Реверс превращается из «восстановления формы» в «восстановление скелета», и это даёт три эффекта. (1) Толщина перестаёт быть геометрией: смена 3 мм → 2 мм — правка одного числа, а не перестроение всех поверхностей (актуально: применяются обе толщины). (2) Скелет детали и общий Skeleton сборки говорят на одном языке — плоскости и линии, поэтому связка «изменил H платформы → пересчитались углы фланцев ноги» становится геометрически прямой: линия гиба лежит на пересечении плоскостей, заданных скелетом сборки. (3) Параметризация осмысляется почти сама: длины пластин, углы между ними и положения линий гиба — ровно те величины, что меняются от машины к машине, а артефакты производства (керн-метки, случайные скругления) в скелет не попадают вовсе — схлопываются вместе с толщиной. Medial Axis фильтрует шум за инженера.

Реализация — без «настоящего» Medial Axis Transform (в OCC штатной функции нет, устойчивое построение срединных поверхностей на реальных B-Rep нетривиально): скелет строится из уже существующего AAG — пары эквидистантных граней дают срединные плоскости, цилиндры гибов дают срединные оси и радиусы. Результат идентичен, а код — тот, что и так спроектирован. Скелет детали сохраняется в measurements.json как основная структура; дальше на него навешиваются параметры (spec.yaml) и правила (rules/).

Кривизна как устойчивый детектор гибов и формовок

Второй, независимый от описания поверхности критерий: для каждой грани считаются главные кривизны и производные величины — плоскость даёт гауссову K=0 и среднюю H=0; цилиндрический гиб — K=0 при H≠0, причём направление нулевой главной кривизны и есть ось гиба, а вторая кривизна даёт 1/R; формовки и выштамповки — K≠0 (неразвёртываемые участки). Практическая ценность прямая: критерий работает независимо от того, как STEP описал грань — при экспорте из Inventor плоскости и цилиндры часто приходят B-spline-аппроксимациями, где формального «цилиндра» нет и поиск по типу поверхности проваливается, а кривизна есть у любой поверхности. Бонус: K≠0 сразу помечает деталь как содержащую формовку, то есть требующую двухконфигурационного подхода, — ещё до попытки развернуть. Порядок в обмерщике: сначала опрос типа поверхности через OCC, при неудаче — замер кривизны.

Автопроверка корректности развёртки (Гаусс-Бонне): для правильной плоской развёртки сумма угловых дефектов по контурам даёт 2π; отклонение означает топологически неверную развёртку — складки или самонакладки. Одна формула в приёмке развёртки: дешёвая страховка, ловящая грубые провалы анфолдера автоматически, без человеческого взгляда на DXF (метрологию не заменяет).

Что автоматика НЕ делает: выбор, какое число — параметр (высота, меняется от машины), какое — правило (R3 всегда), какое — формула (ширина от длины), а какое — артефакт производства, который не воспроизводится. Это конструктивный замысел; его расставляет человек или Claude со skills, читая measurements.json как текст. Далее — spec.yaml + generator.py + params.json.

Детектор разгрузочных вырезов (Entlastungsschnitte / bend reliefs)

Разгрузки предотвращают разрыв и смятие металла при гибке, но усложняют топологию — именно они вызывают «прерванные гибы», на которых падал SMBendWall и путался Unfolder V1. Распознавание в три шага, встраивается в тот же обмерщик: (1) дешёвая эвристика по контурам и рёбрам грани гиба — канонический гиб имеет 1 контур и 4 ребра; больше одного контура = сквозной вырез внутри гиба, больше четырёх рёбер = вырезы по краям; (2) паттерн в AAG — идеальный гиб смежен ровно с двумя плоскими фланцами, разгрузка добавляет ему в соседи мелкие торцевые грани или разрывает цилиндр на несколько соосных (поиск подграфа в уже построенном графе, ничего нового не требуется); (3) UV-проекция границ гиба на цилиндр — отклонение от прямоугольника даёт точные размеры выреза (глубина, ширина, отступ); применяется только к найденным случаям.

Ключевое отличие от «просто копирования»: в системе разгрузка — это ПРАВИЛО, а не форма из оригинала. Генератор вычисляет её по толщине, радиусу и типу узла (features.py), а не переносит контур. Поэтому у детектора две роли: сигнал «этот узел технологически требует разгрузки» (иначе валидатор выдаст конфликт и предложит relief_obround) и источник

калибровки — обмер разгрузок на деталях, принятых Blechcon, даёт реальную зависимость «размер от t/R/d», которая зашивается в features.py навсегда. Третий выигрыш практический: знание «на этом гибе прерывание из-за разгрузки» позволяет автоматически выбрать проверенный обход развёртки (Revolution вместо SMBendWall либо Defeaturing: срезать вырезы → развернуть чистый гиб → вернуть вырезы плоскими) — то, что сегодня решается руками в каждой сессии, становится ветвью в коде. Инструменты те же: OCP (TopExp_Explorer, BRepAdaptor_Surface) + networkx; отдельный pythonocc-core не нужен — то же ядро уже доступно через OCP.

Гибрид «точные числа + зрение» и агентная петля коррекции

Ключевое дополнение к обмерщику: геометрия и семантика извлекаются разными средствами и соединяются. OCC/AAG/FTG дают точные числа (толщина, радиусы, углы, координаты отверстий) — в этом машина безупречна. Рендер детали в 4 ракурсах (проекции + изометрия из build123d) плюс мультимодальная модель дают семантику ролей: «это разгрузка у основания гиба», «это фальц для жёсткости края», «это П-образный кронштейн», «это артефакт штампа, воспроизводить не нужно». Именно семантики не хватало: AAG измеряет, но не знает, что параметр, что правило, а что артефакт. Работает сегодня без обучения моделей и без датасетов — рендер, взгляд, разметка measurements.json, генерация.

Агентная петля автокоррекции (штатный режим реверса): экстрактор → LLM пишет generator.py → CI собирает STEP → сверка с исходным → расхождения возвращаются модели числами («вырез смещён на 2 мм, строка 45») → правка → повтор до сходимости. Стоп-правило прежнее: 3 неудачные итерации подряд → эскалация человеку с дневником попыток (иначе модель крутится вечно). Судья — только числовая метрология point-sampling ≥ 99.9% и точные интерфейсы; визуальный дифф (наложение рендеров оригинала и сборки с подсветкой расхождений) используется как навигация по ошибке и объяснение человеку, но НЕ как критерий приёмки: деталь может «выглядеть одинаково» при расхождении в полмиллиметра на посадочном месте.

Эволюционный подбор параметров — для тех 10–20% случаев, где распознавание сдаётся (свободные радиусы, скульптурные переходы): шаблон пишет человек, числа подбирает оптимизатор (scipy / CMA-ES, без ML), минимизируя ту же метрологическую невязку point-sampling. Дёшево, детерминированно, встраивается в CI как ночной прогон.

Оригами-кинематика и модальная сигнатура

Кинематическая развёртка («жёсткое оригами»): деталь представляется как жёсткие плоские пластины (фланцы) и одномерные шарниры по осям гибов; последовательные повороты вокруг осей до нулевых углов дают плоскую заготовку. Один алгоритм даёт два выхода сразу: траектория решателя (последовательность поворотов) — это и есть порядок гибки, а конечное состояние — заготовка. Метод закрывает сразу две задачи: развёртку без анфолдера (углы шарниров = углы гибов, конечное состояние = контур заготовки) и автоматическую проверку самопересечений при складывании — ту самую Bending Feasibility, которую иначе считает регрессивная развёртка. Реализация — чистая геометрия на numpy (последовательные повороты + тест пересечений); физические движки (PyBullet/MuJoCo) для жёсткого оригами избыточны.

Модальная сигнатура как финальный валидатор реверса: для оригинала и для реверснутой модели считаются первые ~5 собственных частот (CalculiX). Совпадение спектров доказывает механическую эквивалентность, а не только геометрическую: частоты чувствительны к толщине, радиусам в зонах гиба и распределению массы — то, что точечная метрология может проглядеть. Применяется к ответственным деталям (ноги, несущие кронштейны), необязательно для мелочи; инфраструктура FEM всё равно строится под валидатор, так что чек почти бесплатный. Красивое замыкание: тот же модальный анализ, что уже выполнен компанией, служит и приёмкой реверса, и правилом «f1 ≥ порога по Струхалю» против гула на скорости.

Цеховая практика в коде: базы, прокат, допуски, следы инструмента

Базирование от технологических баз (datums), а не от нуля CAD: деталь описывается от тех плоскостей, от которых её меряет ОТК (базы на поверочной плите, щуп КИМ) и от которых её позиционирует оператор у заднего упора листогиба. Тогда переменные генератора называются STOP_DISTANCE_X1, а не offset_x, размеры в коде совпадают с контролируемыми на производстве, и цепочка размеров идёт от базы — меньше накопленной ошибки. В spec.yaml появляется секция datums (базовые плоскости и точки установки).

Направление проката: гиб вдоль волокна даёт микротрещины на внешнем радиусе у нержавейки — при радиусах, близких к минимальным, риск реальный. Правило связывает конструкцию и раскрой: если ось гиба параллельна направлению проката (оно задаётся ориентацией детали на листе при нестинге), валидатор либо требует увеличить минимальный радиус, либо переориентировать деталь в раскройной карте — и записывает предупреждение в комментарий генератора.

Толщина как распределение, а не число: реальный лист «3 мм» бывает 2.93. Обмерщик замеряет толщину в N точках и выдаёт номинал плюс разброс; в генератор идёт номинал, в спеку — допуск с assert-проверкой, K-фактор считается от фактической толщины. Обмерщик перестаёт спотыкаться на «плавающей» толщине.

Следы матрицы как источник технологии: при воздушной гибке на внешнем радиусе остаются две параллельные линии сопряжения цилиндра и плоскости — расстояние между ними выдаёт ширину V-канала, на котором деталь гнули. По чужой детали восстанавливается не только геометрия, но и технология, а радиус сразу привязывается к стандартному ряду матриц (V = 6, 8, 10, 12, 16 мм) — прямое дополнение к таблице оснастки поставщика. Проверка коллизий уточняется: не «согнётся ли вообще», а «согнётся ли на этом прессе в этом порядке» — деталь не должна задевать верхнюю траверсу на 2-м и 3-м гибе (габаритные модели инструмента запрашиваются у Blechcon; фирменные каталоги Amada/Trumpf не нужны). В задел без приоритета: распознавание замков «шип-паз» и зазоров под сварку — актуально, если появятся сварные узлы (сейчас соединения болтовые).

Рассмотрено и отклонено (чтобы не возвращаться по кругу): GNN-распознавание граней (BRepNet/CADNet/Sheet-metalNet) — нет открытых моделей под листовой металл, нужен размеченный корпус в тысячи деталей, точность вероятностная; ваш класс деталей распознаётся правилами детерминированно. ML-предиктор пружинения на FEA-датасетах (ANSYS/LS-DYNA — платные) — решает задачу, которую уже решает гибщик на своём прессе; наш путь — строка в dfm_blechcon.json из практики Blechcon. Специализированная VLM «картинка → код» (CAD-Coder) — research-модели слабее связки «точные числа OCC + рассуждение сильной модели». EvoCAD (популяция вариантов с VLM-жюри) — дорого по токенам и оптимизирует визуальное сходство вместо размерной точности. Gaussian Splatting с телефона — точность на бликующем металле далека от требований к интерфейсам; практичная версия (фотограмметрия оболочки + штангенциркуль по крепежу) уже принята; вернуться при оцифровке чужих аксессуаров «в поле». Analysis Situs, pythonocc-core отдельно, CadQuery — дублируют то, что уже в стеке. Инверсный CSG («воздушный слепок»: вычесть деталь из габаритного параллелепипеда и разбирать отрицательный объём на примитивы) — опирается на булевы операции, которые по нашему же выученному уроку на сложных телах возвращают бессмыслицу; радиусы и отверстия и так берутся прямо с цилиндрических граней. Синтетический лазерный скан STEP + нейросети облака точек (CAD-Recode / Point2CAD) — осознанная потеря точности ради устойчивости к дефектам STEP, которые лечатся healing-ом в две строки OCC. Дифференцируемый CAD (SDF + градиентный спуск) — технически рабочий, но решает ту же задачу, что принятый оптимизатор CMA-ES, требуя дифференцируемого рендера и подбора гиперпараметров; вернуться, если CMA-ES упрётся по скорости. Синтез программ через SMT-решатели (SyGuS/Z3) — «доказанная корректность» недостижима: решатели работают с логикой и арифметикой, а не с B-Rep; наша гарантия — измеренная (метрология + spec-прогон), а не доказанная. Латентная диффузия CAD-деревьев (GenCAD-3D) и спектр Лапласа-Бельтрами — обучение на миллионах программ / математика для поиска похожих форм; плоскости и цилиндры распознаются напрямую. PINN для динамического K-фактора — уравнения пластичности ради числа, которое берётся из Material Definition Sheet и калибруется по принятым Blechcon деталям. RL-агент для развёртки (обучение подбору последовательности

разгибания) — задача уже решена детерминированно дважды: регрессивная развёртка и оригами-кинематика дают порядок гибов и проверку самопересечений за секунды и воспроизводимо. Нейро-символический DSL с AST-валидацией через Z3 — тревога «LLM ошибается синтаксически» снята дешевле: код исполняется в CI (синтаксис падает мгновенно и возвращается модели), семантику ловят валидатор и метрология. INR/SIREN (нейросетевое поле заполненности) — потеря точности ради градиентов там, где точные плоскости лежат прямо в B-Rep. Квантово-вдохновлённый генетический алгоритм — обёртка над обычным эволюционным поиском; принятый CMA-ES делает ту же работу с проверяемой сходимостью. Персистентные гомологии (TDA) — из идеи взята только суть (инвариант целостности) в виде чек-листа сохранности элементов, без алгебраического аппарата. Оптические каустики (трассировка лучей для замера радиуса), голономия/параллельный перенос (двугранный угол получается скалярным произведением нормалей в одну строку), тепловой поток по уравнению Фурье для поиска вырезов (перечисление внутренних контуров дешевле на два порядка) — экзотика ради экзотики. Карта мартенситного превращения (γ→α′ при холодной деформации) — физика верная, но для расчёта нужна симуляция деформации, для которой уже надо знать, где гибы: круговая логика; реальная ценность темы у нас другая — наклёп в зонах гиба влияет на коррозионную стойкость и магнитность готовой детали, это вопрос качества продукта, а не реверса. Класс «фундаментально-физических» методов рассмотрен целиком и отклонён (фаза Берри, поток Риччи/Ямабе, калибровочная теория дефектов, полиномы узлов, обратная задача рассеяния, голографический принцип, симплектическая механика, топологические краевые состояния, функционал Онсагера, сложность Колмогорова): все они решают три задачи — углы и радиусы гибов, развёртка, поиск вырезов — которые уже закрыты кривизной, оригами-кинематикой и счётом контуров с машинной точностью за секунды. Точность системы сегодня упирается не в математику распознавания, а в физику и данные: K-фактор реального листа, компенсация пружинения на конкретном прессе, допуски исходного STEP, точность скана крыши — их эти методы не уменьшают. Читаемость кода (циклы вместо повторов) достигается инструкцией в промпте и ревью, а не вариационной задачей. Класс «геймдев-математических» подходов также рассмотрен и отклонён: IK-риг совпадает с уже принятой оригами-кинематикой (приятное подтверждение естественности решения); SDF-реймаршинг, поток Риччи, shape grammars и Wave Function Collapse, MCTS-поиск кода, воксельные октодеревья, клеточные автоматы — это ранее отклонённые классы в новой упаковке (дифференцируемый CAD, нейро-символический DSL, потеря точности ради устойчивости); краудсорсинговый Foldit требует аудитории игроков, которой нет. Из теории Морса взято только следствие, и так вытекающее из скелетного подхода: строить деталь в коде от базовой плоскости к периферии, а не в произвольном порядке.

Критерий «100% копия с параметрическими изменениями» (в rules/): реверс завершён, когда выполнены все четыре условия — (1) интерфейсы (отверстия, посадочные плоскости, углы) совпадают точно; (2) метрология point-sampling на номинальных параметрах ≥ 99.9% против исходного STEP; (3) прогон на ИЗМЕНЁННЫХ параметрах зелёный у валидатора и spec-теста (доказательство, что параметрика живая, а не декорация); (4) чек-лист сохранности элементов: отверстий было N — стало N, вырезов и разгрузок было M — стало M (считается напрямую: сквозные отверстия — парные цилиндрические грани, вырезы — внутренние контуры; ловит самую обидную ошибку «забыл дырку»); (5) список осознанных отличий от оригинала записан в спеку (керн-метки и случайные скругления не воспроизводятся; технологический язычок воспроизводится, но как правило, а не как копия). Оставшиеся доли процента — обычно K-фактор, точность сплайнов и допуски самого исходного STEP; гнаться за ними — воспроизводить чужие ошибки экспорта.

Стек (только то, что уже в системе): build123d/OCP — чтение STEP и B-Rep; networkx — AAG; numpy — векторная математика; метрология — distToShape point-sampling; развёртка — SheetMetalUnfolder V2 headless (FreeCAD здесь импортируемая библиотека, не среда), с миграцией на YouCanNotUnfold (быстрый анфолдер на networkx, drop-in замена) или встроенную развёртку build123d по мере зрелости; нестинг — SheetNest / nest2d / CAD-N / u-nesting (NFP-алгоритмы с учётом kerf). Отвергнуто как избыточное: pythonocc-core отдельно (то же ядро уже есть через OCP), CadQuery (дублирует build123d), Analysis Situs (тяжёлая C++-платформа ради того, что закрывают ~100 строк на networkx). Пружинение (springback): нержавейка 1.4301

сильно наклёпывается — измеренный на готовой детали угол уже включает отпружинивание; компенсация — забота гибщика, но её реальные значения от t/R стоит завести строкой в dfm_blechcon.json из практики Blechcon.

Spec-Driven Development: спека задачи как контракт

Система уже устроена по принципам SDD (истина — спецификация, геометрия — артефакт, валидатор — исполняемая спека ограничений, история «зачем» в коммитах). Дополняем тремя приёмами. (1) Спека до геометрии: для каждой новой детали до первой линии в песочнице пишется spec.yaml — намерение (intent), интерфейсы стыковки (из паспортов), проверяемые ограничения приёмки (масса, нагрузка, зазор, целевая себестоимость по BOM) и явные non-goals (чего НЕ делаем). Спека — файл истины в products/ рядом с params.json. (2) Спека → тест → реализация: CI генерирует приёмочный прогон из constraints автоматически ещё до начала работы; красный прогон на пустой детали — норма, работа исполнителя = сделать его зелёным; приёмка превращается из «проверить сданное» в «позеленить спеку», субъективность ревью сжимается до вопроса «валидно и не глупо?». Спека = формальное основание сдельной оплаты — контракт чище любого текстового ТЗ; спор «деталь не та» превращается в «спека была неполна», и чинится спека, обогащая шаблон. (3) LLM работает от спеки, не от чата: промпт сначала материализуется в spec.yaml (Claude сам оформляет), человек подтверждает за секунды, потом генерация — намерение зафиксировано в Git до реализации, «LLM понял не так» ловится на спеке, а не на геометрии; правило зашивается в rules/ и в системный промпт RECADA-ассистента. Внедрение: этап 2, рядом с валидатором (генератор приёмочного прогона — малый скрипт CI).

Сборка: проверки уровня изделия

Детали собираются в багажник, поэтому у сборки свой слой проверок — отдельно от геометрии деталей. Позиционирование уже описано (именованные интерфейсы вместо констрейнов, всё от скелета); добавляются четыре проверки, все — геометрия и арифметика, без физических движков и симуляции монтажника.

Проверка	Что делает и зачем
Доступ инструмента (assembly feasibility)	Для каждого болтового соединения проверяется свободный цилиндр доступа: прямой канал от оси болта наружу под головку ключа (диаметр, длина хода) и вылет для установки стержня. Ловит классическую ошибку «спроектировано, но не собирается»; критично, потому что монтаж идёт на крыше в неудобной позе. Заодно проверяется, что деталь заводится на место при уже смонтированных соседях.
Размерные цепи (tolerance stack-up)	Платформа стоит на четырёх ногах: допуск гибки (±0.5–1°), допуск отверстий лазера, разброс крыши — сумма даёт перекос или несведение отверстий на последней ноге. Система проходит цепочку «крыша → нога → профиль → следующая нога» по худшему случаю и статистически (RSS) и выдаёт минимальный необходимый овал отверстий. Снимает целый класс рекламаций «не встало».
Автопорядок монтажа	Последовательность выводится из графа связей (нога к крыше → профиль к ногам → аксессуар к профилю) с учётом проверки доступа, а не пишется руками. Инструкция генерируется автоматически и всегда актуальна; Unity-конфигуратор получает готовую последовательность для интерактивной 3D-инструкции. YAML остаётся для человеческих подсказок («не затягивать до установки задних ног»).
spec.yaml сборки	У изделия тоже есть контракт приёмки: суммарная масса, допустимая нагрузка, число уникальных SKU (метрика унификации — чем меньше разных деталей, тем дешевле склад), оценка времени монтажа по числу крепежа, совместимость аксессуаров. Приёмка сборки — такой же зелёный прогон, как у детали.

5. Конфигуратор (Unity + Shopify) и инструкции

Конфигуратор разрабатывается на Unity (WebGL для сайта; из того же проекта — нативные мобильные приложения с AR-примеркой через AR Foundation и десктоп для дилерского киоска/выставки). Контракт CI ↔ фронтенд не зависит от движка — catalog.json: SKU, ревизия, статус, GLB, точки стыковки, варианты, правило замены; Unity грузит GLB в рантайме через

glTFast, attach-точки читаются как named nodes. Возможность «техпроцесс в конфигураторе»: Unity-сцена визуализирует сборочные шаги из тех же YAML-инструкций — интерактивная 3D-инструкция вместо PDF. Риски Unity-WebGL: тяжёлый билд (15–50 МБ против 2–5 у three.js) — следить за временем первой загрузки на мобильном (~5 с порог); при провале — гибрид: лёгкая витрина-превью + полный Unity-конфигуратор. GLB сжимается (gltfpack+Draco: десятки КБ), ревизия в имени файла против кэша CDN. Клиенты видят только протегированные релизы. «Светофор нагрузки»: суммирование масс аксессуаров против допустимой нагрузки крыши — рост конверсии, все данные уже в каталоге. Инструкции из YAML (шаг, детали, крепёж, момент) рендерятся CI с актуальными изображениями; конфигурация клиента известна — инструкция генерируется под его заказ. Конфигуратор на базе FreeCAD для розницы отвергнут (секунды на пересборку, ферма контейнеров, мобильные): розница = pre-built GLB, мгновенно; VDI-FreeCAD с панелью RECADA = CPQ-инструмент для дилеров/нестандарта, выход — params.json сразу в производство.

6. Прототипирование: четыре ступени

Ступень	Что проверяет	Срок/цена ошибки
1. Цифровой	Коллизии, FEM (CalculiX+Gmsh в валидаторе), AR-примерка на машине (model-viewer, тот же GLB)	минуты / 0 €
2. Печатный	--target=proto → STL, FDM 1:1 (PETG): посадка, отверстия, помехи; ID получателя выдавлен на нелицевой стороне; принтер 300–500 € окупается первой итерацией	часы / ~2 €
3. Металл (DC01)	Blechcon как прототипная мастерская; material_override: DC01 вместо 1.4301 (2–3× дешевле); маркировка PROTO not for sale	дни / десятки €
4. Пилотный комплект	Боевой материал, установка, фотофиксация → правки в паспорт и DFM-правила: цикл обучения замкнут	недели / сотни €

7. Аксессуары: вторая ось

Боксы, маркизы, Starlink, LED-бары, солнечные панели. Ключ — стандартизованный интерфейс платформы (Т-паз, шаг): аксессуар привязан к интерфейсу, не к машине — спроектирован один раз, подходит всем. Паспорт аксессуара: крепление, объём + зоны исключения, масса и ЦТ (валидатор считает суммарную нагрузку конфигурации), совместимость. Для покупных изделий проектируется только адаптер — тот же конвейер. Оцифровка: STEP от производителя (дают дилерам) или скан — фотограмметрия для оболочки (±1–2 мм, глянец матировать) + штангенциркуль для крепежа (±0.05); поле точности в паспорте, неточность компенсируется конструктивно (овальные отверстия). Печатная примерка ловит ошибки оцифровки за 2 €. Библиотека паспортов популярных аксессуаров — накапливаемый актив.

8. Кадровая лестница: рост по результатам в логах

Ступень	Доступ	Переход
Новичок	Тестовые и простые задачи, только VDI, сдельно	вход свободный
Проверенный	Прямые заказы, ставка выше	3+ принятых работы
Senior	Сложные узлы, Git-доступ, проектная оплата	устойчивое качество работ + коммуникация
Lead/reviewer	Первичная приёмка, наставничество, ТЗ; решает bus factor	отдельный отбор по качеству мышления в комментариях (принцип Питера: лучший чертёжник ≠ лучший ревьюер)

Одна удачная работа — шум, паттерн на 4–6 задачах — сигнал. Система логирует: качество вопросов к ТЗ, итерации до зелёного, поведение при проигрыше, экономическое чутьё. Повышение вычисляется, а не назначается — прозрачность правил сильнее премий на рынках СНГ/Азии. Финальное доверие (доступ к ядру) — личное решение.

9. Наём в разных странах: рынки и ставки

Рынок	€/час	Роль
Индия	4–10	Основной пул: глубина, английский, бюро sheet metal (Пуна/Ченнай/Бангалор), UTC+5:30. Пинг 100–160 мс — фильтровать пробной сессией.
Узбекистан/Кыргызстан	5–12	Русский трек: ГОСТ-школа, лояльность; отсюда вырастут постоянные и lead. Пинг 60–90 мс — VDI комфортен.
Пакистан	3–7	Добавочные руки строго сдельно (свет/интернет — риск исполнителя; PayPal нет).
Вьетнам	5–12	Отложить (язык, UTC+7, пинг 180–250); стратегичен при азиатском векторе производства.
Россия	—	Напрямую нет (платежи, комплаенс); легально — релоканты (Армения/Грузия/Казахстан) или ИП в третьей стране; Kasm по HTTPS работает при любых блокировках.

Модель найма: удалённые инженеры-конструкторы на сдельной основе (оплата за деталь, прошедшую валидатор) с ростом до постоянного сотрудничества. Воронка: 100 приглашений → 30 откликов → 15 тестов → 8 сдали → 3–4 прошли → 1–2 на скамейку; цена постоянного ~300–500 €. Каналы: Upwork (активные приглашения на платный тест), hh.uz, GrabCAD, кафедры вузов, рефералы. Оффер-магниты: платный тест; не нужен софт и мощный ПК (браузерная сессия — заодно легальность и доступ пула без железа); автоматическая приёмка по опубликованным правилам; прозрачная лестница; постоянный поток. Ступенчатый отсев: фильтр-вопрос → платный тест с автоприёмкой (эталон Bracket_L) → лично смотрите 3–4 из ста. Тест включает 10-мин пробную VDI-сессию (меряет реальный канал кандидата).

10. Жизненный цикл: две оси

Конструкция (status)	Склад/продажа (sales_status)
Draft / Черновик — в работе, снаружи не видна In Review / На проверке — валидатор зелёный, PR открыт Released / Выпущена — подпись + тег + Rev; единственный производимый статус Superseded / Заменена — вышла новая ревизия (автоматически); хранится вечно Obsolete / Снята — негодна, преемника нет; производство блокирует CI On Hold / Заморожена — остановлена без решения	Not for Sale — прототипы (PROTO), пилоты Active — производится и продаётся Run-out — superseded, остаток >0: продаём, не дозаказываем; webhook Shopify при нуле включает преемника Discontinued — снята; карточка живёт вечно (гарантия, QR-инструкции) Service Only — по обращению как запчасть к проданному

Политика замены A→B выбирается при релизе: run-out / немедленная (дефект) / параллельно (FFF: та же посадка и функция = ревизия, не новый SKU). Ось 1 движет CI по git-событиям; ось 2 — webhook Shopify + два ручных решения.

11. Изменение деталей: где дорого

Создание дёшево; изменение дорого пропорционально зависимостям. Три тяжести: в существующих параметрах (бесплатно) / новый параметр (правка генератора — узкое место квалификации) / слом топологии (= новая деталь; правило «больше N строк — новый шаблон» против костылей). Каскад (развёртка, масса и нагрузка конфигураций, GLB, инструкция, BOM) CI пересобирает сам; управленческими остаются: неприкосновенность интерфейса (изменение = «версия 2 платформы»), явная политика замены на складе, запрет обходов мимо params (STEP на Blechcon — только из CI, технически). Плюс: геометрический дифф в PR (подсветка изменений старый/новый STEP), версионирование библиотечных блоков с явной миграцией, шлюз на изменение released (issue с обоснованием + go владельца).

12. ISO 9001

Система = исполняемый QMS: 8.3 (входы=паспорт+ТЗ, выходы=артефакты CI, верификация=валидатор, валидация=прототипы, изменения=PR+ревизии); 8.5.2 прослеживаемость (деталь → тег → автор → ревьюер → заказ); 7.5 документация = Git; 10.2 CAPA (рекламация → issue → правило валидатора навсегда). Фрилансеры = внешние поставщики (8.4): страница критериев допуска/оценки/отстранения (данные собирает Gitea), объём контроля привязан к лестнице (риск-ориентированность), компетентность (7.2) = конкурсная история, ревьюеру — чек-лист и персональная подпись. GDPR: обработка данных фрилансеров (аккаунты, логи) — пункт в договоре, запись в Verzeichnis, срок хранения логов. Сертифицироваться — когда спросит рынок (B2B/OEM): фундамент ISO-ready, формализация 2–3 мес, ~3–6 т.€. Правило: документируется исполняемое или используемое — иначе театр качества.

13. Физические испытания и ISO 11154 (новый трек)

Слепая зона, закрытая внешним аудитом: система проектирования не заменяет испытаний готового изделия. Даже если съёмный багажник юридически классифицируется как груз (Ladung, §22a StVZO — проверить у юриста, не принимать на веру) и не требует ABE/Teilegutachten, отсутствие документированных динамических испытаний по ISO 11154 (включая City Crash, ускорения до ~16g) при аварии превращает Produkthaftung в персональный риск руководства и закрывает вход в B2B-дилерские сети. Действия: (1) выяснить текущий статус испытаний продукта; (2) при отсутствии — отдельный испытательный трек: эталонные испытания базовой конструкции в аккредитованной лаборатории (бюджет — тысячи €, это цена входа в B2B); (3) расширить FFF-правило пунктом «изменение, влияющее на несущую способность, требует переиспытания» — валидатор помечает такие изменения автоматически; (4) прослеживаемость «какая ревизия испытана» система уже даёт бесплатно (git-тег + протокол испытаний как артефакт релиза).

Текущий статус и расчётная база

Компанией уже выполнены прочностной (статический) и модальный анализы конструкции; планируется аэродинамический. Разграничение: расчёт (FEM) и физическое испытание по ISO 11154 юридически не взаимозаменяемы — расчёт говорит «должна выдержать», протокол испытания документирует «выдержала»; для Produkthaftung и B2B нужен физический протокол, а расчёты — сильное дополнение, удешевляющее сертификацию (меньше итераций в лаборатории). Модальный анализ: собственные частоты против частот вихревого срыва на 100–130 км/ч (гул, резонанс, усталость) — превращается в правило валидатора «первая собственная частота ноги ≥ порога» (порог по формуле Струхаля из скорости и размера профиля; CalculiX умеет модальный анализ штатно). Аэродинамика прагматично: ветровые нагрузки как входные силы прочностного расчёта — нормативные формулы без CFD; частоты срыва — Струхаль; полный CFD (OpenFOAM) — разово для формы windshield и гула, вне конвейера. Существующие отчёты — эталоны калибровки CalculiX-пайплайна: прогнать ту же деталь, сверить частоты и напряжения — валидатор рождается проверенным (логика метрологического эталона Bracket_L, но для физики).

14. Открытая платформа сообщества

Печатаемые аксессуары сообщества (держатели, клипсы) делают платформу «липкой» и служат разведкой спроса (популярная модель → задача на металлическую версию). Публикуется открытая спецификация интерфейса (измерима штангенциркулем; опубликовать первым = стать стандартом; управление — у вас, модель USB); публичный репозиторий + STL; автопроверка посадки тем же конвейером; категория Community в конфигураторе. Защита (Produkthaftung): CC BY-NC-SA + disclaimer (вы хостинг, не производитель), автомодерация категорий (несущее на скорости — на свой риск/не публикуется), витринная граница «металл протестирован / community на свой страх». В правилах — право Airholder на металлическую версию по мотивам (с упоминанием автора). Сообщество = нулевая ступень кадровой лестницы. Веб-визуализация не раскрывает ИС: в браузер уходит огрублённый GLB («фотография часов, не чертёж»); децимация — параметр CI; точная геометрия покидает сервер единственным маршрутом CI → Blechcon. От обмера купленного изделия не защищает ничто — реальная защита: скорость конвейера, паспорта, библиотека решений, валидатор, скамейка, стандарт.

15. Роль FreeCAD и инженера: кто что делает

Позиция уточнена по итогам практики. Инженер из системы не уходит — он остаётся автором формы. Полноценный CAD мы не пишем: это не наш продукт и не наша экспертиза. Но новые детали делать надо, и делать их надо руками — в FreeCAD. Инструмент recada.py не заменяет конструктора, он снимает с него рутину: обмер, реверс, проверки, развёртки, пакет производства.

Разделение труда

Кто	Что делает	Чем
Инженер	придумывает и рисует новую деталь, принимает решения по форме, радиусам, разгрузкам	FreeCAD — руками, привычно
Генератор кода	превращает нарисованное или реверснутое в код с параметрами; код открывается во FreeCAD как обычная деталь	recada.py → Python для FreeCAD
Инструмент	обмер, реверс, растяжение, операции, три критерия приёмки, DXF, BOM, пакет для Blechcon	headless-ядро на OCCT, веб-интерфейс
Редактор изделия	багажник как параметры: высота, положение, длина; ноги перегенерируются	рабочее окно

Почему FreeCAD остаётся, а не уходит

Опыт показал: в цепочке автоматики FreeCAD не нужен и даже мешает — его OCCT 7.8 не выполняет операции, которые проходят на 7.9 в нашем ядре. Но там, где нужна рука конструктора — новая форма, нестандартный узел, эскиз под конкретную машину — FreeCAD незаменим и бесплатен. Отсюда гибрид: FreeCAD рисует, инструмент считает и проверяет,

генератор кода связывает их в обе стороны.

Идея workbench: как связать

Два пути, проверены обсуждением. Первый — полноценный верстак FreeCAD на его Part-API: требует переноса кода с OCP на другой API и упирается в отличия версий OCCT (offset у нас падал именно там). Второй — гибридный верстак: кнопки в FreeCAD сохраняют выделенное тело во временный STEP, вызывают внешний инструмент из его окружения и импортируют результат обратно. Ни переноса, ни конфликта версий; тяжёлая геометрия считается там, где она работает. Рекомендован второй путь. Делать его стоит после того, как устаканятся операции ядра — оборачивать движущуюся цель в интерфейс рано.

VDI (Kasm/Guacamole) как способ дать внешним исполнителям FreeCAD в браузере остаётся в резерве — для сценария с недоверенными подрядчиками из раздела о найме. Для доверенной команды достаточно WireGuard и локального FreeCAD с генератором кода.

15a. Инструмент реверса recada.py — построен и проверен

За два дня сентября собран рабочий инструмент, который снимает деталь с чужой геометрии и превращает её в проверенный производственный файл. Всё headless, без FreeCAD в цепочке: ядро OpenCASCADE подключено как библиотека (OCP), FreeCAD остаётся только смотрелкой. Один файл recada.py: командная строка и браузерный интерфейс.

Что умеет

Функция	Что делает	Проверено на
Обмер	толщина, гибы с двумя углами (поворот и двугранный), радиусы, отверстия с осями, разгрузки	Defender LRDE-2BR1: t=3.0, 7 гибов, 6 отверстий — повторяется до сотых
Расщепление на оболочки	виртуальный щуп: точка − нормаль × t попадает на поверхность → грань оболочечная; парование + двухцветная раскраска графа	41 грань покрывает 93 % площади; стороны 21/20
Реконструкция	кинематика на всех гранях тела, перестроение зоны гиба, заделка дыр	объём 78031.6 против эталонных 78031.6
Растяжение	разрез полупространством вдоль плоскости стенки, сдвиг, призма сечения; автоподбор места разреза по одиночному сечению	стенка +10 мм, +20 мм — замкнуто, годно
Операции	зеркало, сверление, заглушка, обрезка, скругления — с проверкой после каждой	все дают замкнутый солид
Развёртка	DXF с линиями гиба на отдельном слое	299 × 168 мм — совпало с порталом
Сборка	разбор STEP на детали с именами, группировка одинаковых, пакетный реверс	84.5 МБ, 628 объектов

Три критерия приёмки — главный вывод

Критериев оказалось не один и не два, а три, и третий не заменяется первыми. Это не теория: каждый найден через отказ.

Критерий	Чем проверяется	Что ловит	Как найден
Метрология	point-sampling двусторонний, расстояние до ПОВЕРХНОСТИ	точность геометрии	свой
Целостность	свободных рёбер = 0, солидов ≥ 1	дыры, незамкнутость	отказ портала Blechcon
Производство	загрузка на портал поставщика	распознаётся ли как листовая деталь	опыт

Три ошибки, найденные и исправленные

1. Метрология мерила расстояние до тела, а не до поверхности. distToShape до солида возвращает ноль для точек внутри материала — «толстая» деталь проходила приёмку. Проверка: сдвиг детали на 0.3 мм давал 74 % точек в допуске, после исправления 29.8 %.

2. Односторонняя выборка не видела лишний металл: кандидат длиннее эталона давал RMS 0 и 100 %. Невязка сделана двусторонней.

3. Проверки замкнутости не было вовсе. Деталь может быть геометрически точной и дырявой одновременно — метрология этого не ловит. Обнаружено отказом портала, теперь считается одной строкой и предсказывает отказ до отправки.

Что не получилось и почему

Наращивание оболочки целиком (offset всей сшитой оболочки) оставляет дыры и зависит от версии OCCT: на 7.9 работает, на 7.8 внутри FreeCAD падает. Заменено поэлементным наращиванием с булевым объединением — для одной грани offset корректен всегда.

Заплатки BRepFill в зоне гиба дают сплайновую грань — распознаватель листового металла на такой ломается. Отсюда правило: в зоне гиба строить только канонические поверхности (цилиндры и плоские секторы), профиль разгрузки генерировать по правилу, а не латать.

Параметрический поворот угла гиба даёт замкнутое тело, но пока не проходит распознавание у поставщика: торцевой профиль гиба с разгрузкой надо генерировать заново, а не копировать. Это единственная незакрытая задача цепочки — она понятна до деталей.

15б. Редактор изделия: багажник как параметр, а не как геометрия

Ключевая мысль, к которой пришли в работе: у багажника нет произвольных степеней свободы. У него есть продуктовые параметры, а положение и форма деталей — их следствия. Редактор поэтому не двигает детали, а перегенерирует их.

Сценарий, ради которого всё делалось

Багажник спроектирован, между крышей и низом поперечины 20 мм — руку не просунуть. Решение: приподнять. Если ноги уже реверснуты в код, достаточно одного числа: точки крепления на крыше зафиксированы, вертикальная стенка каждой ноги растягивается на 20 мм, развёртки пересчитываются, DXF обновляются, три критерия приёмки прогоняются заново. Полный реверс всей системы не нужен.

Тот же цикл обслуживает аэродинамику: CFD говорит «сдвинуть платформу вперёд на 30, поднять на 10» — это два числа в product.json, а не неделя ручной работы. Именно поэтому гнутые детали обязаны жить как код с параметрами.

Единое рабочее окно

Зона	Содержимое
Дерево слева	таблица как BOM: видимость, выбор, имя, количество экземпляров, статус (исходник / реверснута / параметрическая), роль (нога, кронштейн крыши, платформа, поперечина, крепёж, кузов). Поиск по имени, автоопределение ролей по геометрии
Сцена	свой просмотрщик на three.js: разлёт, прозрачность, сечение, измерение, виды, изоляция. Выбранные подсвечены, реверснутые зелёные
Действия	для исходника — «Полный реверс» (одинаковые детали считаются один раз); для реверснутой — растяжение, отверстия, зеркало, DXF
Изделие	точки крепления к крыше как якорь, «поднять платформу на N мм» — перегенерация всех ног
Журнал	project.json: каждая операция записана параметрами. Истина — журнал, геометрия — артефакт

Пакет производства

Одна кнопка собирает архив: step/ — уникальные детали, dxf/ — развёртки с линиями гиба, passport/ — обмер каждой в JSON, BOM.csv — спецификация с количествами, project.json — журнал операций. Это и есть то, что уходит в Blechcon.

Что это меняет в экономике проекта

Реверс кронштейна Defender занял минуты вместо дней. Четыре одинаковые ноги считаются один раз. Подъём платформы — одно число вместо перепроектирования. Инфраструктура: EX44 за 59 евро нетто плюс Storage Box — около 63 евро в месяц за систему, которой нет в коробочных CAD: автоматический реверс листа с проверкой производственной пригодности.

15в. Библиотека деталей и сервер: система с обратной связью

Сервер EX44 куплен, библиотека собственных STEP-файлов собирается. Вместе они превращают инструмент из демонстрации в систему, где каждая проверенная деталь делает следующую точнее.

Что даёт сервер

Разделение ядра и интерфейса: ядро — библиотека функций с командной строкой, интерфейс — тонкий слой над очередью задач. Реверс ставится в очередь, страница не блокируется, результаты сохраняются в постоянное хранилище по хешу файла. Без этого не заработают ни второй пользователь, ни библиотека.

Непрерывная регрессия: каждый коммит в инструмент прогоняется по библиотеке ночью на CI, утром — одна цифра: сколько деталей проходят три критерия. Ломать работающее незаметно больше нельзя.

Общий доступ: удалённые инженеры из раздела о найме работают через WireGuard с одним инструментом и одними правилами.

Что даёт библиотека

Применение	Как работает	Что закрывает
Калибровка K-фактора	наша аналитическая развёртка против ответа портала Blechcon по каждой детали (299 × 168 подтверждено)	третий критерий из ручной проверки становится источником данных
Каталог разгрузок	типы, глубины, привязка к кромке — снятые с реальных деталей	незакрытую задачу зоны гиба: генерировать по правилу, не латать
DFM-правила	статистика радиусов, углов, толщин, диаметров, отступов от кромки	dfm_blechcon.json из практики, а не из головы
Поиск похожего	подпись детали: t, число гибов, набор углов, рисунок отверстий	взять проверенную и растянуть вместо проектирования с нуля
Паспорта авто	точки крепления и профили крыш как данные библиотеки	редактор изделия без ручного ввода координат
Стандарт школы	новый конструктор видит принятые радиусы и решения	передачу опыта без устных преданий

Граница честности: библиотека советует и предупреждает, решает человек. Она не скажет, выдержит ли деталь нагрузку, и не заменит инженерное решение там, где новая машина требует того, чего в практике ещё не было.

Рынок: Zoo как ориентир

Zoo (zoo.dev, бывший KittyCAD) — ближайший по философии проект: их язык KCL — читаемый текст как источник истины, версионирование в Git, агент Zookeeper переписывает STEP в код. Два отличия делают нашу нишу свободной: геометрический движок Zoo облачный и американский — 3D-вид это видеопоток с их серверов, чертежи уходят к ним; и у них нет производственной приёмки листового металла — метрологии, целостности, распознавания порталом поставщика. Zoo подтверждает направление и не закрывает нашу задачу.

Как Zoo стримит и что из этого следует

Движок Zoo работает на их серверах. Клиент ничего не считает: команды моделирования уходят по WebSocket, обратно приходят видеокадры по WebRTC — как видеозвонок. Стримится только 3D-вьюпорт; интерфейс, дерево и редактор кода живут в клиенте. Это не Kasm: Kasm стримит целый рабочий стол или приложение через захват экрана, тяжело и с задержкой; Zoo встроили рендер и кодек прямо в собственный движок на Rust, поэтому клиент лёгкий. Три причины их выбора: не портировать движок под платформы, открывать любые модели на слабых машинах, и главное — геометрия никогда не покидает их инфраструктуру, клиент видит только пиксели.

Для нас это независимое подтверждение принципа из раздела о VDI — «пиксели, не геометрия». Разница в реализации: они встроили стриминг в свой движок, мы стримили бы готовое приложение. Обратная сторона нашей текущей схемы: просмотрщик на three.js отдаёт в браузер триангулированную сетку — для доверенной команды через WireGuard это нормально, для внешних подрядчиков это утечка, сетку можно сохранить. Поэтому для недоверенных остаётся стриминг пикселей.

Свой движок: ядро — нет, обвязка — да

«Движок» Zoo — три слоя: геометрическое ядро (B-Rep, булевы, скругления, развёртки), рендер и стриминг с API. Ядро они пишут годами командой специалистов по геометрическим ядрам с десятками миллионов инвестиций — и в марте 2026 всё ещё писали, что скругления рёбер остаются их болью. Это цена ядра. Нам его писать не нужно: OpenCASCADE — тридцать лет

работы, бесплатно и открыто, и на нём вчера реверснута деталь, которую принял портал. Ядро — решённая задача чужими руками.

Что собрать реально можно — два верхних слоя вокруг OCCT. Рендер на сервере: наш three.js-просмотрщик в headless-браузере, либо встроенная визуализация OCCT. Стриминг: картинка по WebRTC в клиент, команды камеры обратно — готовые библиотеки, не исследование, порядок работы недели. Это «стриминг вьюпорта вокруг OCCT», не «свой движок».

Когда это нужно: только для внешних исполнителей, которым нельзя отдавать геометрию. Для своей команды нынешняя схема проще и быстрее. Стриминг вьюпорта — в резерве рядом с VDI, делается при появлении первого недоверенного подрядчика. Zoo пришлось писать ядро, потому что они продают платформу. Мы продаём багажники — разная экономика, разные решения.

Ядро напрямую: OpenCASCADE без FreeCAD

Точная формулировка архитектуры: OpenCASCADE и есть ядро внутри FreeCAD. Мы вызываем его напрямую через обёртку OCP, минуя FreeCAD — двигатель на стенде вместо автомобиля. Это даёт headless-работу на сервере и в CI, пакетную обработку библиотеки и свою версию ядра (7.9, где наращивание работает; в FreeCAD 7.8 оно падает). FreeCAD остаётся кабиной для инженера — там, где нужна рука. Одна геометрия, два способа обращения: программный для автоматики, ручной для человека. Практика: держать FreeCAD той сборки, где версия OCCT совпадает с ядром.

Ядро создано с ИИ, проверено реальностью, работает без ИИ. ИИ писал код ядра, но не работает внутри него: каждая функция — обычный детерминированный Python, один вход даёт один выход. Три ошибки ядра нашёл не ИИ — их нашли перепроверка, негативный тест и отказ портала. Отсюда правило разработки: тест раньше кода (Spec-Driven Development, применённый к самому инструменту), библиотека как регрессия после. Обучать свою модель не нужно и не выйдет — десятки деталей не выборка; библиотека работает как справочник. Готовый ИИ подключается как переводчик намерений к операциям ядра, не как источник геометрии.

Просмотрщики на том же ядре (состояние на сентябрь 2026)

Инструмент	Что это	Где применить
OCP.wasm + Yet Another CAD Viewer	OpenCascade и обёртка OCP скомпилированы в WebAssembly через Pyodide; build123d работает целиком в браузере. Та же OCP, что в recada.py	инструмент у клиента в браузере без сервера — ничего не покидает машину; ответ на утечку сетки для внешних подрядчиков
OpenCascade.js (ocjs.org)	порт OCCT в JavaScript/WASM через Emscripten, почти нативная скорость, многопоточность, урезанные сборки; на нём CascadeStudio	браузерный просмотр и лёгкие операции без утечки B-Rep
OCCT WebGL viewer	собственная визуализация OCCT в WASM — открывает BREP в браузере, нужен WebGL 2.0	показать «ровно как в ядре» без FreeCAD
Mayo	настольный просмотрщик и конвертер на Qt + OCCT: STEP, IGES, BREP, экспорт glTF; обновляется	бесплатный десктопный STEP-вьювер на нашем ядре — проверить, что видит ядро
OCP CAD Viewer (VS Code)	показывает объекты OCP/build123d прямо из Python, без STEP	отладка ядра при разработке
Наш three.js-просмотрщик	сетка с сервера, выбор, разлёт, сечение, измерение	быстрый интерфейс для доверенной команды

Стратегически интереснее всего OCP.wasm: наша обёртка та же, значит recada.py может работать у клиента в браузере без сервера вообще — обратный подход Zoo, где всё улетает в облако. Для внешних исполнителей это третий путь наряду с VDI и стримингом вьюпорта: считают у себя, отправляют только результат приёмки.

Решение: каскадный вьювер на сервере, наружу только пиксели

Принято после появления EX44. Ядро и рендер OCCT работают на сервере; визуализация OCCT (V3d, AIS) доступна из Python через ту же OCP — без C++ и без WASM. Браузер получает кадры и отправляет клики; сервер сам определяет, на какую грань, ребро или вершину пришёлся клик — выбор точный по B-Rep, с привязкой к центрам отверстий и серединам рёбер. Геометрия не покидает сервер: ни сетка, ни STEP. Это модель Zoo на OCCT и раздел о VDI без Kasm.

Отвергнутая альтернатива — гибрид с локальным вращением по сетке: быстрее и дешевле для сервера, но сетка уходит в браузер и может быть сохранена. Для защиты данных неприемлемо.

Достаточно ли EX44

Видеокарты нет — рендер программный, через Mesa на процессоре. Сцена CAD проста: сотни тысяч треугольников, десятки кадров в секунду на ядро. Плюс кодирование кадра — ещё ядро. Итого два потока на активную сессию; активная — только пока человек вращает, при просмотре нагрузка нулевая. За вычетом обмера и инфраструктуры — пять-шесть одновременно вращающих. Для команды с запасом, для внешних подрядчиков — несколько человек одновременно.

Что расширяет запас без железа: во время вращения кадр пониженного разрешения, после остановки — один чёткий; лимит частоты кадров на сессию. Когда понадобится другое железо: при десяти и более одновременно активных — GEX-серверы Hetzner с GPU, цена в три-четыре раза выше; решение на потом.

Защита данных — не только стриминг

Мера	Что закрывает
Все библиотеки (three.js, pako и др.) на своём сервере	сейчас просмотрщик грузит их с внешних CDN — факт открытия детали виден третьей стороне по запросам
Доступ только через WireGuard, свой ключ у каждого	в логах видно, кто что открывал и когда
Без точек скачивания для внешних ролей	экспорт STEP и DXF — только доверенным
Водяной знак с именем сессии на кадре	снимок экрана запретить нельзя, атрибутировать — можно
Рендер по запросу, не поток	нагрузка на сервер и объём трафика

Граница, признанная заранее: от фотографии экрана не защищает ничто. Пиксели закрывают утечку системы — моделей целиком, параметров, библиотеки. Одну деталь перерисовать с экрана можно всегда: «перенабрать деталь можно, систему нельзя».

Этапность: сначала статичный рендер по запросу — кадр с заданной камерой и номер грани по клику; непрерывный поток — позже, если понадобится. Место в очереди: после разделения ядра и интерфейса и очереди задач; строить точный вьювер к инструменту, который меняется по три раза в день, рано.

16. Инфраструктура: решения по железу

Почему Германия — самые дешёвые серверы мира: промышленные энергоконтракты + free cooling, DE-CIX (почти бесплатный трафик → unlimited), ЦОДы в дешёвой провинции при толстой оптике, культура ценовой конкуренции (Hetzner/netcup — семейные компании без инвесторов), вертикальная интеграция с десктопным железом. AWS отвергнут: 6–8× цена инстанса, egress ~0,09 $/ГБ убивает VDI-потоки (у Hetzner unlimited), CLOUD Act против data sovereignty, сложность без пользы. Совместимость бесплатна (Docker переезжает за день), зависимость не нужна.

Роль	Решение	Статус/цена
Целевой сервер (ядро+VDI 8–10 сессий)	Hetzner EX44: i5-13500 (быстрый однопоток — ключ к отзывчивости), 64 ГБ, 2×512 NVMe RAID 1, unlimited, FSN1	€70,21, setup 0, почасово; Limited-волны — подписка на availability, правило 15 секунд на заказ
Параллельный канал	Server Auction (Robot → Server Market): запросы «i5-13», «i5-12», «i7-12»; NVMe, не HDD; ничего старше 12-го поколения	критерий: ≤ €75–80
План Б (через ~3 недели)	OVH RISE-S: Ryzen 9700X (лучший однопоток), 64 ГБ, Frankfurt/Strasbourg, месяц-к-месяцу	€78 + €78 setup
Отвергнуто	AX41 (Zen 2 2019 за те же €70), AX42 (€118 за 35-Вт чип), EX63 (€177 — переплата за ядра), i9-13900 аукцион (€165 — рано), Contabo (CPU steal)	—
Масштаб (этап 5, 10+ сессий)	OVH RISE-L (9950X, 128 ГБ, €179) — вытеснил AX102 (€308); i9 с аукциона	по факту роста
Уже есть	ThinkPad i7-8850H/32ГБ/Quadro P1000 — рабочее место + локальный старт этапа 1 (Git, build123d, FreeCAD; диск 54 ГБ свободно — Docker позже); Strato VPS 2/2/90 — Gitea+WireGuard и/или вторая копия бэкапа; Hetzner CPX — CI	0 € новых
Второй провайдер	netcup: RS G12 (выделенные EPYC-ядра, ECC, дёшево) — бэкап сейчас (~5 €), в будущем возможное отдельное ядро (Gitea+CI) при VDI на десктопном CPU; EPYC-однопоток слаб для VDI	~5 €/мес

Три входа Hetzner: accounts (паспорт) → Cloud Console (облачные CPX) и Robot (dedicated: заказ, rescue, Server Market). EX44 после поставки: rescue → installimage → Ubuntu 24.04 LTS + RAID 1 → SSH-ключ, ufw (наружу только SSH+WireGuard), unattended-upgrades, бэкап в Storage Box (~4 €) + вторая копия у netcup. Домен: поддомены git./work./cdn. существующего домена (или отдельный технический ~15 €/год) — Let's Encrypt, стабильность при переездах, разделение контуров; рабочие поддомены могут резолвиться только внутри WireGuard. Авторизация — три непересекающихся контура: рабочий (локальные аккаунты Gitea+Kasm + TOTP 2FA + WireGuard-периметр; SSO/Keycloak после 20 человек), клиентский (Shopify Customer Accounts / Google — конфигуратор наследует), сообщество (OAuth GitHub/Google в публичной Gitea). Бюджет полной системы при 100 зарегистрированных (пик 20–25 сессий): ~150–200 €/мес.

Стоимость владения: только open source

Категория	Инструменты (все бесплатны, без подписок)
CAD и генерация	FreeCAD + SheetMetal + аддоны (GPL/LGPL), build123d/CadQuery (Apache)
Версии и сервер	Git, Gitea, Git LFS, zippey; Ubuntu, Docker (self-hosted), WireGuard, ufw; Let's Encrypt; Cloudflare free (CDN)
Расчёты	CalculiX + Gmsh (статика, модальный), OpenFOAM (CFD разово)
Рендер и артефакты	Blender headless, SheetNest (раскрой), gltfpack/Draco (GLB)
VDI	Kasm Community Edition (до 5 одновременных сессий); без лимитов — Apache Guacamole / webtop-образы linuxserver.io

Категория	Инструменты (все бесплатны, без подписок)
Визуализация	Mermaid/Graphviz (граф зависимостей), React Flow/Rete.js (composep, этап 5), Sverchok (нодовое моделирование)
Конфигуратор	Unity Personal — бесплатен до $200k годовой выручки с продукта; единственный инструмент с порогом; бесплатная альтернатива при необходимости — three.js/Babylon.js

Платные строки проекта — только три, все приняты осознанно: серверы (~70–200 €/мес по росту), токены Claude/API (~20–50 €/мес на старте), физический мир (FDM-принтер разово 300–500 €; испытания ISO 11154 — разовые тысячи €). Правило проекта: инструмент входит в стек, только если он open source или бесплатен для нашего масштаба без подписки; платными могут быть только вычислительные ресурсы и физические вещи; кандидат на платный инструмент требует явного решения владельца с бесплатной альтернативой в кармане.

Claude как копилот — четыре слоя: (1) Claude Code на сервере — админ-диалог вместо ручного Linux; (2) MCP-FreeCAD headless — двигатель промпт-слоя; (3) в CI неинтерактивно (claude -p): упал валидатор → фикс коммитом в ветку → ревью человеком; перевод сданных STEP в черновики генераторов; (4) в VDI-образе подрядчика — отдельный API-ключ с лимитом и системный промпт RECADA (ядро невидимо). Правило: Claude ходит через те же двери, что люди (ветки, PR, валидатор), автор коммита «LLM via MCP» с промптом в сообщении. Бюджет 20–50 €/мес на старте.

17. Граф зависимостей конструкции и автоматизация

Граф зависимостей — визуальная карта конструкции: узлы — файлы истины (паспорта, скелеты, генераторы, детали, сборки, конфигурации), стрелки — «читает из». Зависимости в системе объявлены явно, поэтому карта строится автоматически скриптом CI (Mermaid/Graphviz, бесплатно) — рисовать руками ничего не надо, схема всегда актуальна. Главный режим — анализ изменения: перед merge CI отвечает «что пересоберётся, если принять этот PR» — список и подсветка на схеме (красное = затронуто, отдельно помечаются released-детали). Каскад «что за что тянет» становится видимым ДО нажатия кнопки; это же — инструмент онбординга (новый человек понимает систему за час по карте) и диагностики (где в цепочке красный валидатор). Делается на этапе 2 рядом с валидатором.

Автоматизация бизнес-событий — простыми скриптами и webhook в CI, без отдельного оркестратора: складской webhook Shopify (остаток А=0 → политика run-out → активировать Б → уведомление), онбординг (тест пройден → карточка исполнителя → письмо), мониторинг (релизный тег → сводка в канал; CI красный сутки → алерт), маркетинг (релиз → рендер → черновик карточки через LLM на утверждение), продажи как источник ТЗ (форма «нет крепления?» → issue + счётчик спроса → 3 запроса = приоритет в разработку). Граница неизменна: автоматизация не трогает геометрию и не пишет в main — всё, что меняет истину, идёт через PR и валидатор. Скрипт создаётся под событие, случившееся руками трижды. Нодовый composep поверх библиотеки кубиков (React Flow/Rete.js — открытые) — этап 5: интерфейс для не-программистов (дилер-CPQ, клиент с шаблоном); свой Grasshopper не делать (есть Sverchok/FreeCAD-ноды — бесплатные).

18. Финансовая модель — каркас

Поставщики и данные для честной экономики: структурированный архив счетов Blechcon и тарифа GLS с первого заказа + pre-flight nesting (знать справедливую цену резки до запроса) + периодический ручной бенчмарк 1–2 европейских альтернатив + квалификация второго поставщика как страховки от монополии. Полный API-аукцион нескольких сервисов отвергнут: правила валидатора = правила конкретного производителя, несколько поставщиков = кратная поддержка DFM-правил.

```
CM1 (маржа комплекта) = Цена − металл − Blechcon − профили − крепёж/покрытие − упаковка/доставка − комиссия канала
CM2 (с учётом дизайна) = CM1 × ожидаемые продажи модели − (турниры + часы владельца × ставка + прототипы + паспорт
Операционная = Σ CM2 − постоянные (инфраструктура, время, маркетинг, склад)
```

Ориентир отрасли (4x4-аксессуары): валовая 45–60% в рознице, 30–40% через дилеров; ниже ~35–40% от розницы — чинить до масштабирования. Ключевой вопрос системы: за сколько проданных комплектов окупается вывод новой машины. Для расчёта нужны 8 цифр: розничная цена комплекта и канал; счёт Blechcon за комплект ног (или масса); €/метр профиля и метраж; крепёж/уплотнители/покрытие; упаковка+доставка; текущие и плановые объёмы; внутренняя ставка часа владельца; текущая стоимость вывода машины по-старому. Метрика №2 включает доставку металла — юнит-экономика честная.

19. Управленческий слой: роли подробно

Роли — это не должности, а зоны ответственности с чёткими правами решений; в малой команде один человек совмещает несколько ролей, но границы решений остаются письменными. Принцип: у каждого решения — один владелец; «решили вместе» = не решил никто.

Роль	За что отвечает	Права решений	Доступы	Время
Owner (владелец)	Стратегия и деньги; приоритеты трубы (что делаем следующим); критерии приёмки; доверие людям; отношения с Blechcon и ключевыми партнёрами; развитие системы (ворота этапов)	Единолично: релизная кнопка (тег), бюджеты трёх конвертов, повышения по лестнице, изменение интерфейса платформы, изменение released-деталей (go на issue), допуск инструментов в стек (FOSS-правило)	Полный: Git-admin, сервер root, Robot/Cloud Hetzner, Shopify, банк	Цель — сжатие до управления трубой; всё, что затягивает внутрь (рисовать, чинить CI), — сигнал незакрытой роли
Lead Reviewer	Первичная приёмка всех PR (геометрия, посадка, собираемость, экономика по BOM, эстетика); ведение исполнителей (ответы на вопросы к ТЗ, разбор отказов валидатора); формулировка ТЗ вместе с Owner; наставничество новичков; предложение правил в rules/ из повторяющихся ошибок	Merge в main для деталей (кроме изменений интерфейса и released — эскалация Owner); отклонение работ; распределение задач по скамейке; эскалация повышений	Git: merge-права на products/accessories; чтение generators и rules; Gitea-канбан; без root сервера	Первая делегируемая роль — решает bus factor: минимум раз в месяц проводит приёмку полностью без Owner. Отбор — по качеству мышления в комментариях, не по скорости черчения (принцип Питера)
Senior Engineer	Сложные узлы и новые типы деталей (топология); перевод «песочница → код» (реверс в генераторы); развитие библиотеки кубиков; калибровка FEM по эталонным отчётам; сложные паспорта (windshield-поверхности)	Коммиты в generators/ через PR; создание веток convert/*; предложение новых кубиков; техрешения внутри ТЗ без согласования	Git-доступ (режим Б: локальная работа + push), выбранные репозитории; VDI по желанию; без прав merge в main	Вырастает из проверенных по логам (устойчивое качество + коммуникация); финальное доверие — личное решение Owner
Engineer (проверенный)	Прямые задачи по готовым шаблонам и ТЗ: адаптеры, вариации ног, паспорта аксессуаров по чек-листу; печатные примерки своих деталей	Работа в своей ветке; сдача через PR; вопросы к ТЗ приветствуются (логируются как сигнал роста)	VDI-сессия с расширенным набором репозиториев; либо локально + Git по решению Owner	3+ принятых работы; сдельная оплата выше новичка; очередь на задачи
Engineer (новичок)	Тестовые и простые задачи строго по ТЗ; работа только в изолированной VDI-сессии	Только своя ветка своей задачи; ничего кроме	VDI-сессия: один репозиторий задачи, egress только Gitea, водяной знак, без скачивания	Вход свободный, платный тест с автоприёмкой; риск компании — сотни евро

Роль	За что отвечает	Права решений	Доступы	Время
Ops (инфраструктура)	Сервер, Docker, Gitea, Kasm, WireGuard, бэкапы (+ квартальный тест восстановления!), обновления с фиксацией версий, мониторинг, онбординг/оффбординг аккаунтов (одна процедура: ключи, сессии, права)	Плановые обновления и правки конфигурации; экстренные действия при инцидентах с пост-отчётом Owner; НЕ решает про доступы людей (это Owner)	Root сервера, admin Gitea/Kasm; без прав в конструкторские репозитории по содержанию	10–20% времени одного человека; на старте совмещает Owner (+Claude Code как копилот), позже — первый кандидат на аутсорс/частичную занятость
Технолог Blechcon (внешняя)	Производственное ревью выпускаемых деталей («если гнуть иначе — на 15% дешевле»); консультации по граничным случаям DFM	Рекомендательные; замечания конвертируются в правила rules/ через PR (цикл CAPA)	Получает только конкретные DXF/STEP по заказу; доступа в систему не имеет	По договорённости, возможно платно; дешёвый канал чужого производственного опыта

Правила совмещения и роста

На старте: Owner совмещает Lead Reviewer и Ops (с Claude как копилотом обеих ролей), Senior — по мере появления. Порядок делегирования: сначала Lead Reviewer (снимает приёмку — главный пожиратель времени Owner и единственное лекарство от bus factor), затем Ops, затем постановка ТЗ. Несовместимость: автор детали не может быть её единственным ревьюером — подпись на released всегда чужая; Ops не решает про доверие и доступы людей. Каждое повышение = расширение доступов по гибридной модели (VDI-only → +репозитории → Git локально → merge-права) и фиксируется в карточке исполнителя в Gitea. Оффбординг любой роли — одна процедура: отзыв WireGuard-ключа, закрытие сессий, ревизия прав, чек передачи незавершённого.

Ритм и бюджет (без изменений)

Релизный такт 2 недели; еженедельный 30-мин обзор трубы (канбан); месячный бизнес-срез (юнит-экономика, метрики); квартальные ворота со стоп-лоссами. Бюджет — три конверта: Infra (~50–150 €/мес фикс), Design (N деталей × целевая стоимость), System-dev (только под ворота); часы Owner — всегда отдельной строкой по внутренней ставке.

№	Метрика	Смысл
1	Время «задача → released»	скорость трубы
2	Полная стоимость released-детали (исполнители + часы владельца + прототипы + доставка металла)	честная цена дизайна
3	Доля прохода валидатора с 1-го раза	качество пула и ТЗ
4	Время добавления новой машины	главная метрика системы
5	Доля выручки от деталей без участия владельца в рисовании	освобождённость владельца

20. Риски и узкие места

Риск	Суть и защита
Обратный поток правок (№1)	Ручная правка в GUI не возвращается в params → два источника истины → смерть системы. Ручное — только песочница; в систему — через параметры/генератор; STEP на Blechcon — только из CI технически.
FreeCAD/SheetMetal	Community-аддон с пределами (SMBendWall, boolean OCC, 7-й фланец — предвестник: реальные крыши грязные, паспорт может расщепиться по комплектациям). Правило 80/20: экзотика руками вне конвейера — это нормально.

Риск	Суть и защита
LLM-слой	Валидатор не ловит «корректную глупость» — человеческое ревью остаётся; время владельца — узкое место.
Bus factor = 1	Lead reviewer реально проводит приёмку без владельца минимум раз в месяц.
Scope creep	Система строится только вслед за реальным заказом; ворота со стоп-лоссами; чужие ревью концепта фильтровать вопросом «что убрать?» — оба присланных списка только добавляли.
Вы — IT-отдел	Gitea/CI/Kasm/VPN — скрытые 10–20% времени, учитывать в бюджете.
Заморозка интерфейса	Максимум внимания при минимуме информации; после продаж менять нельзя (legacy или слом совместимости).
Внешние зависимости	Shopify API, three.js, аддоны — версии фиксируются; восстановление из бэкапа тестируется ежеквартально.
Дешёвый аутсорс	Не бесплатен: ТЗ и ревью — часы владельца; смягчение — валидатор и сдельность.

Риски, добавленные по итогам практики (сентябрь 2026)

Инструмент — один файл без автоматических тестов; библиотека как регрессия — первое лекарство. Зависимость от версии OCCT: 7.8 в FreeCAD и 7.9 в ядре ведут себя по-разному — стоило полдня; фиксировать версию ядра и не смешивать. Третий критерий проверяется руками через чужой портал, который может измениться без предупреждения — вести журнал ответов портала как данные, чтобы изменения были видны.

21. Дорожная карта

Ближайший порядок работ — конкретно

Шаг	Что	Зачем
1	установка сервера: ОС, ключи, ufw, WireGuard, restic на Storage Box (Хельсинки)	база и бэкап в другом ДЦ
2	разделить ядро и интерфейс, очередь задач, постоянное хранилище	многопользовательность
3	сканер библиотеки: пакетный обмер, таблица, статистика	данные вместо догадок
4	регрессия на CI по библиотеке: процент прохождения трёх критериев	не ломать работающее
5	калибровка K-фактора по ответам портала	развёртки точнее
6	каталог разгрузок → генерация зоны гиба по правилу	параметрика углов проходит производство
7	гибридный workbench FreeCAD (вызов внешнего инструмента)	инженер рисует руками, считает ядро
8	паспорта автомобилей в библиотеке → редактор изделия на данных	багажник как параметры

Этап	Содержание	Статус
0. Фундамент	Реверс Bracket_L/R (метрология 100%), wind_side_v2, Skeleton_Ranger + MasterParams, паспорт Ranger, DFM-правила как skills; решения по железу зафиксированы (EX44-критерий)	Сделано
1. Хребет	Локальный старт на ThinkPad: git init + структура + build123d + первый генератор L-кронштейна с автосверкой (вечер №1); Gitea+WireGuard (Strato или сразу EX44 по прибытии); первый CI «params → STEP+GLB»; step_probe.py (обмерщик STEP на AAG) как первый модуль библиотеки; zippey; 7-й фланец (BSpline)	Первый шаг

Этап	Содержание	Статус
2. Валидатор и каталог	Валидатор DFM+ (cad-khana + зазор 20/упаковка GLS (40 кг/коробка)/крепёж) + геометрический дифф в PR; spec.yaml + автогенерация приёмочного прогона из constraints; catalog.json; статусы двух осей + webhook Shopify; SheetNest; рендер-конвейер; автосхема конвейера (Mermaid); архив счетов Blechcon и тарифа GLS (пороги надбавок — в rules/, честная доставка в CM1); параллельный трек испытаний: собрать существующие отчёты (статика, модальный) как эталоны калибровки; запланировать физические испытания ISO 11154 с расчётным обоснованием	
3. Рабочие места	Kasm CE на EX44 (пентест лично, вкл. Python-консоль); egress-файрвол; RECADA-workbench; Claude в VDI-образе с лимитированным ключом	
4. Люди	Onboarding-пакет EN; воронка Upwork+hh.uz с пробой канала; автотест с автоприёмкой; GDPR-оформление; такт 2 недели; YAML-инструкции; светофор нагрузки; скрипты CI: складской webhook и онбординг	
5. Масштаб	Вторая машина по конвейеру; lead reviewer; библиотека узлов → нодовый composep (CPQ с PDF-КП для дилеров; быстрый внутренний прототип панели параметров — Streamlit, выход строго в params.json/Git, чтобы не плодить второй источник истины); FEM в валидаторе (CalculiX+Gmsh: статика с нормативной аэронагрузкой, Мизес с запасом 1.5, прогиб подтверждает зазор 20 мм, модальная проверка f1 ≥ порога по Струхалю, автоотклонение PR с картой напряжений; калибровка по существующим отчётам); ИИ-агент паспортов (STL → распознавание отверстий/нормалей → черновик vehicle_passport + PR; подпись человека обязательна); второй поставщик металла; публикация спецификации интерфейса; сообщество; узел Сингапур/Мумбаи при 40+ ч/мес; RISE-L/i9 при 10+ сессиях; контент-маркетинг; ISO-формализация по запросу рынка	

Формула проекта: мозг и знания — в Германии (структурно самой дешёвой серверной стране), руки — где угодно (пиксели вместо файлов, любой CAD на входе, STEP на выходе), приёмка — автоматическая, рост людей — по логам, а не по резюме. Деталь описывается один раз параметрами; производство, конфигуратор, инструкции и склад — производные одного конвейера. Владелец управляет трубой: приоритеты на входе, критерии приёмки, кнопка релиза.
