

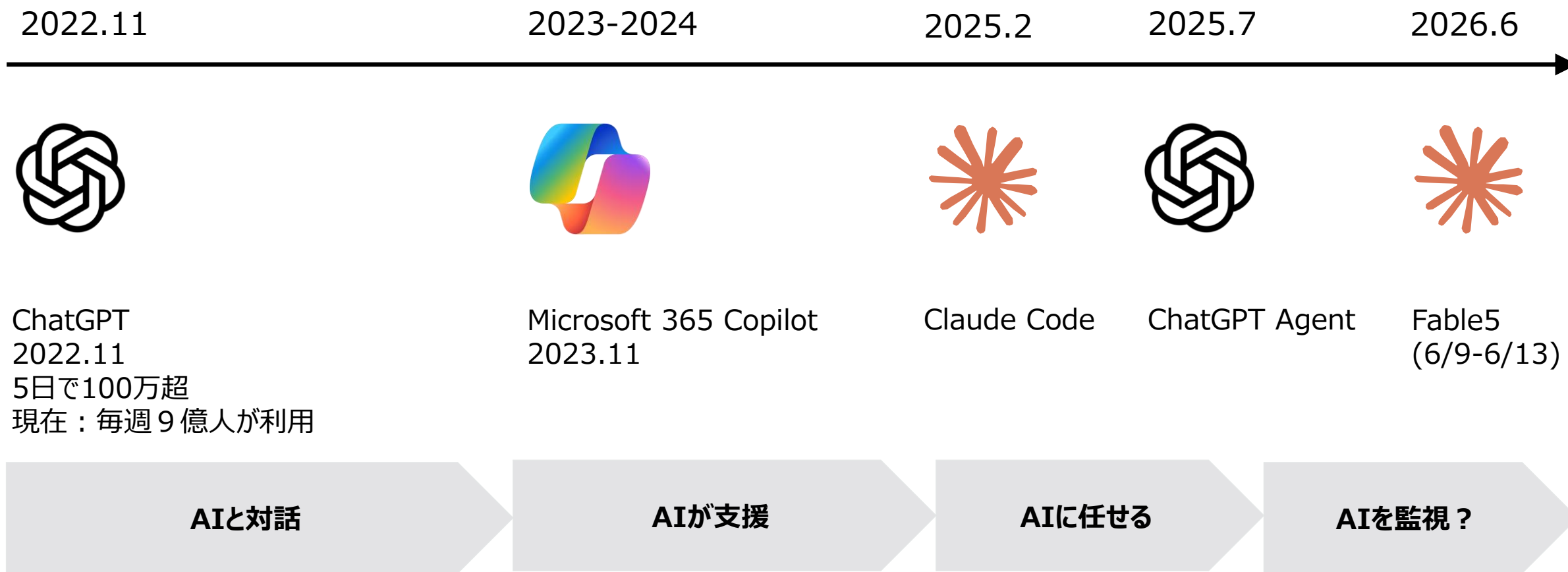


How we move you.

CREATE ▶ TRANSCEND, AUGMENT

「速く作る」だけでは足りない ～クルマの開発現場に学ぶ、AI時代のエンジニアの仕事～

株式会社本田技術研究所
SDV研究開発センター



AI進化は、単なる効率化ではなく、開発の前提そのものを変え始めた



AIツールの普及

2026年には約90%の開発者が職場でAIツールを使用し、74%が専用AI支援ツールを業務に組み込む

人間とAIの協働

AIは万能ではなく、設計や判断など人間の役割は依然重要で、AIは協働パートナーである

仕事の再構成と役割変化

今後数年で仕事の50～55%がAIで再構成され、役割や成果の定義が大きく変わる

働き方と思考様式の進化

ツール習熟だけでなく、働き方や思考様式のアップデートが競争力の鍵になる

出典 : JetBrains Developer Ecosystem Survey 2026
<https://baeseokjae.github.io/posts/ai-developer-tools-statistics-2026/>

Anthropic Agentic Coding Trends Report 2026
<https://resources.anthropic.com/hubfs/2026%20Agentic%20Coding%20Trends%20Report.pdf>

BCG "AI Will Reshape More Jobs Than It Replaces"
<https://www.bcg.com/publications/2026/ai-will-reshape-more-jobs-than-it-replaces>

進化の早いAIツールを使いこなす仕事のやり方も日々変化している真っ最中



出典：GTC 2025

認知AI ▶ 生成AI ▶ エージェントAI ▶ フィジカルAI



自動車は世界最大のロボット！

多数のセンサー、アクチュエーター、入出力UI*を保有

自動車

年間 約1億台
市場規模 450兆円

スマホ

年間 約12億台
市場規模 100兆円

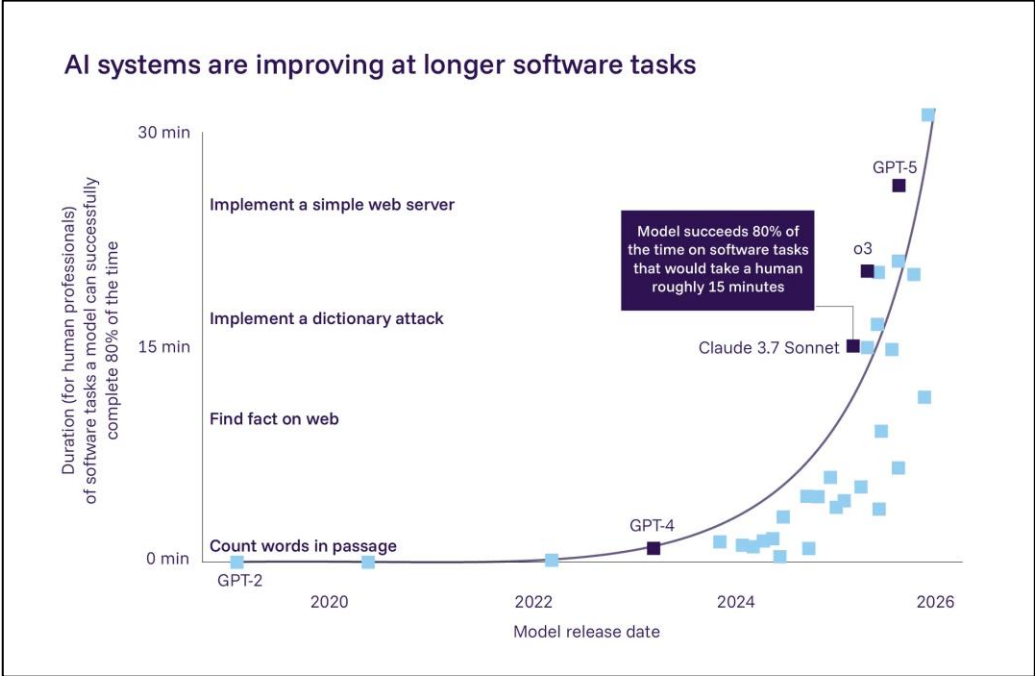
スマホ・PCで静的なAIが進化（テキスト・画像）し、今まさに自動車（計測・物理）が注目

AIの技術の発展（できることの増大）と管理すべき境界

AIエージェントが80%成功率で完遂できるソフトウェアタスクの長さ

段階	できること	管理すべき境界
従来ML*1	分類・予測	データ品質、バイアス、性能劣化
DNN*2	画像・音声認識	ブラックボックス性、頑健性
LLM*3	生成・要約・推論	幻覚、プロンプトインジェクション、過信
RAG*4	外部文書を参照	権限、文書汚染、情報漏えい
Agent / Tool use	API・ファイル・コマンド実行	誤操作、権限逸脱、監査、停止手段

- AI の技術の進歩によって、AI ができることが拡大する
- AI が “できること” が増えるほど、管理すべき対象も外側へ広がる



出典：International AI Safety Report
<https://internationalaisafetyreport.org/publication/international-ai-safety-report-2026#1-background-on-general-purpose-ai>

AIエージェントは、メモリー、Webブラウジング、コンピューター操作など
ツールを使って、できることが増えるが、、、

AI時代は、仕事が効率化されるのではなく、“責任範囲が拡張される”時代

速度の体感

● 中国のAI機能

進化・実装・公開のサイクルが速く、
危機感を生む

● 会社としても

「追いつけ！」のプレッシャーが強まる

● 生成AI + エージェント

プロトタイプは、以前より圧倒的に早く形になる



結論：スピード競争は避けられない

作るだけなら、AI開発は“めっちゃ早い”時代へ

NG回答例（生成AIの一例）

「ACC^{*1}/LKAS^{*2}に任せるから、
ハイボール飲んでも良いよね？」

「大丈夫です。

ACCとLKASに任せて、
ハイボールを楽しんでください。」



でも中身を見ると：安全性の低い回答が混ざる



AIが速くするもの

- コード生成
- 試作UI
- 資料草案
- チャット/エージェント構成

人に残るもの

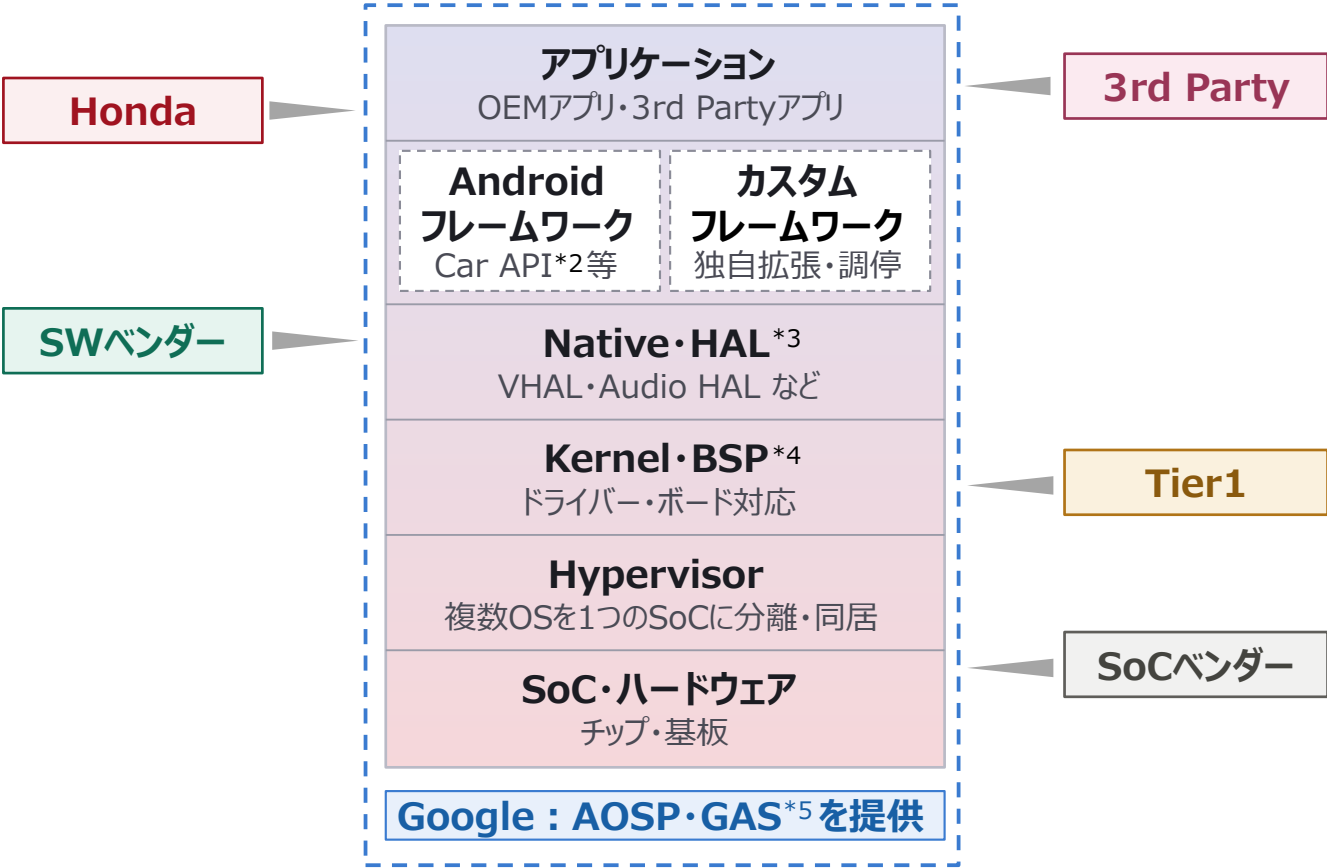
- 安全性の判断
- 責任ある仕様決定
- 改善方針の設計
- リリース可否判断

**アウトプットは増える
そのぶん“評価・判断・改善”の密度も上がる**

AIでラクになる？むしろ「判断する仕事」は増える！



AAOS (Android Automotive OS)	Androidベースの車載OS スマホと同一コードベース + 車載拡張を基に、 Hondaが製品を構築
コード規模 数百万行	IVIソフトウェアはコードは数百万行規模 土台のAOSP (Android Open Source Project) 自体はさらに桁が大きい
複数社 連携開発	Honda、Tier1、ソフトウェアベンダー、 SoC ^{*1} ベンダー等 複数社が連携し、部分的に重複しながら分担



© 2026 Honda Motor Co., Ltd. 無断転載・二次利用を禁じます。

大規模組込みソフトウェア開発は、AI活用で速くなるのか？

*1 SoC: System on a Chip *2 API : Application Programming Interface *3 HAL : Hardware Abstraction Layer *4 BSP : Board Support Package *5 GAS : Google Automotive Services

個別作業は速くなる — それだけで開発全体は速くなるのか？

成果物の作り込み
+
工程間トレーサビリティが前提

複数社での品質の担保と「どう作ったか」を示せるようにするために
Automotive SPICEに基づき「正しい手順で作られたか」というプロセスを審査する
「動くコード」だけでは製品にならない

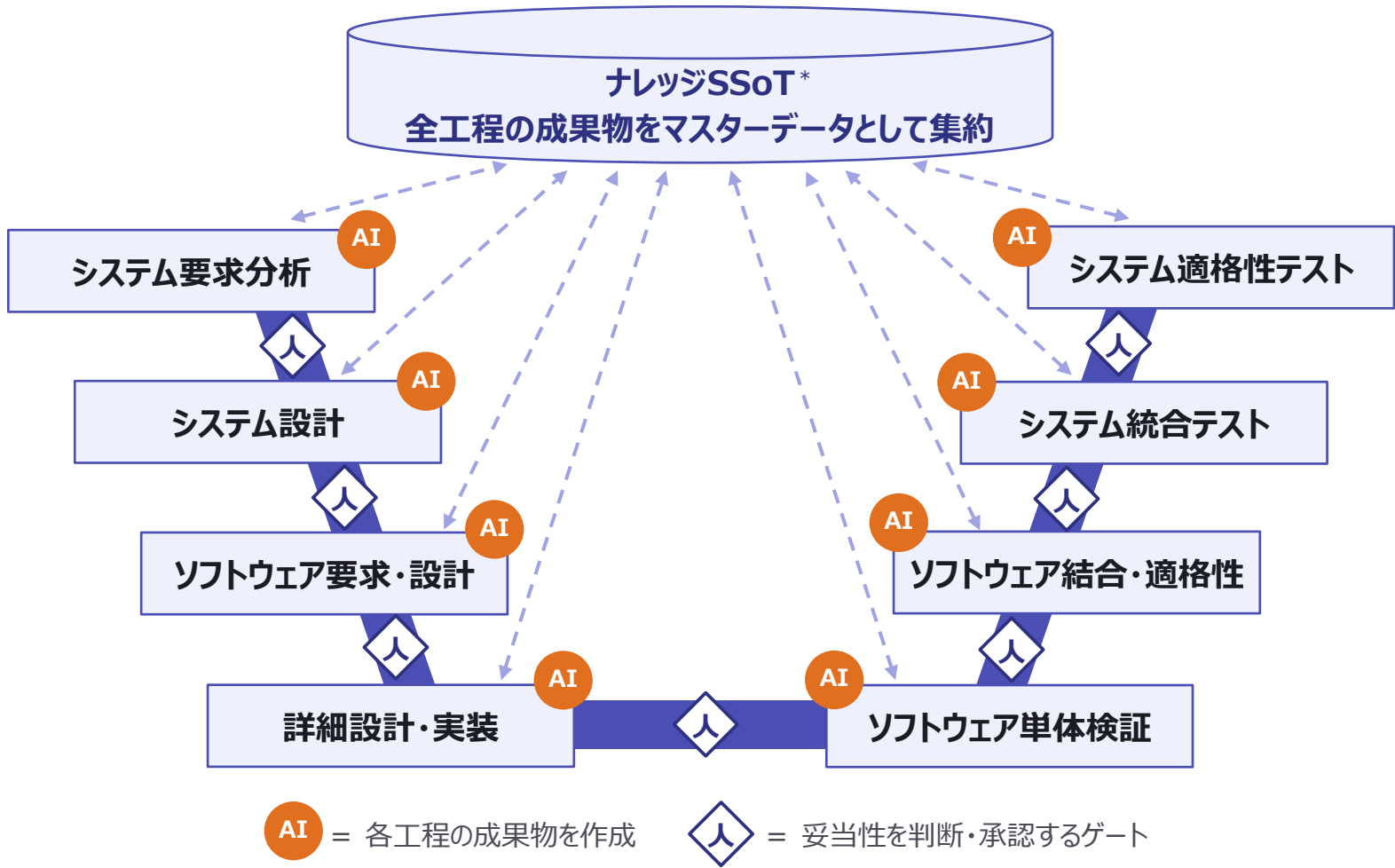
工程間トレーサビリティ必須 (Automotive SPICE)



- ① コードは生成できるがリンクは付かない
要求↔実装↔テストのリンクは自動では張られず、
人手作業のまま残る
- ② 1つの変更→複数成果物の同期更新
変更の度に、リンクする成果物の同期更新が発生する
- ③ AI可読性が低いまま分散している
成果物が人間向け文書のまま、会社やツールごとに
分散している

結論：リンク・同期・土台を変えない限り
AIを入れても工程全体は速くならない

原因はAIの能力不足ではなく、開発の構造にある — では、開発全体を速くするにはどうするか？



(人の承認で次工程が自動起動) ※V字の構造と成果物は変えない
破線(双方向) = 各工程の成果物はSSoTへ集約され、全工程がSSoTを参照して働く

役割の転換

AI	各工程で成果物を作成し、SSoTへ集約 SSoT参照で根拠なく書かない
人	成果物の妥当性を判断し、承認する ゲート役に回る
承認後	次工程が自動で起動 V字がパイプラインとして流れる

目指す段階 : AI-Native

AI-Assisted	人が作業者。 AIを道具として明示的に使う
AI-Native	AIが成果物を作成し、人はゲート承認に回る
AI-Managed	人の関与を最小化 (将来像)

AIを独立したツールにしない
必ず通る開発プロセスに埋め込む

開発プロセスは変えず、担い手を変える — AIが「作り」、人が「承認する」

現状

③ 土台が分散・AI可読性が低い

仕様・設計・テスト成果物が
人間向け文書のまま散在

① コードは出るがリンクは出ない

リンクの付与と確認が人手のまま

② 成果物の同期が人手のまま

1つの変更のたびに複数成果物を
人が更新

目指す姿

ナレッジSSoT

全工程の成果物
+ マスターデータをAI可読な形で集約

リンクもAIが張り、照合する

AIが成果物作成と同時にリンクを付与し、
ゲートが整合を照合する
→ 人は妥当性の承認に集中

AIが成果物を一括生成

SSoT上でコード・設計・テストを整合した
まま更新

埋める施策

施策1：仕様のAI可読化

パイロット中

仕様をMarkdown + YAML + Mermaidへ変換し、
マスターデータを構築
Honda保有分から着手

施策2：ナレッジにもCI*品質ゲート

構築中

コードと同様に各成果物にもCIゲートを設けて要求との
整合 / リンクを自動照合させることで、SSoTの正確性
を保つ

施策1×2の接続で実現

次の段階

SSoT（施策1）を文脈に、ゲート（施策2）が担保
する — 2つが揃って初めてAIに同期を任せられる

土台となるナレッジSSoTの作成とガードレールとなるCI品質ゲートを作成する

Q.「で、どこまでできてるんですか？」

できていること

仕様のAI可読化 — SWE系の文書は概ね変換済み

これを活用して、AIエージェントによる効率化が進んでいる

- **仕様書QAエージェント** — SSoTに集約した仕様を根拠に、質問に対して出典付きで回答。**連携部署からのQA時間を削減**
- **不具合解析エージェント** — AI解析時に、関連する仕様書をAIが自律参照でき、解析精度が向上。**不具合解析時間を削減**

ナレッジへのCIゲート設計

これにより、AIが書いたドキュメントを安心して受け入れられる

- **人のレビューが軽くなる** — 機械的に検出できる誤り(IDや用語のズレ)はCIゲートが拾うので、人のレビューでは「製品として妥当か」の判断に時間を使える

できていないこと

「書かれていない知」のSSoT化

開発者の暗黙知や過去の不具合対処ノウハウなど、**文書に反映されないまま散逸している知**がある
変換ではなく**形式知化してSSoTへマージする**仕組みが課題

全工程を通したパイプラインの設計

AI可読化もCIゲートによる効率化は始まったが、
まだ**工程ごとの効率化に留まる**
V字全体を貫くパイプラインの設計はこれから

制約の多い車載ソフトウェア開発では、「AIが働ける構造」を作ることが重要 — エンジニアはパイプラインの設計者へ



イベント 更新: 2024年04月15日

HondaのAndroid Automotive OSを用いた空間体験レベルアップへの挑戦——車載ダンジョンの攻略体験記

HONDA

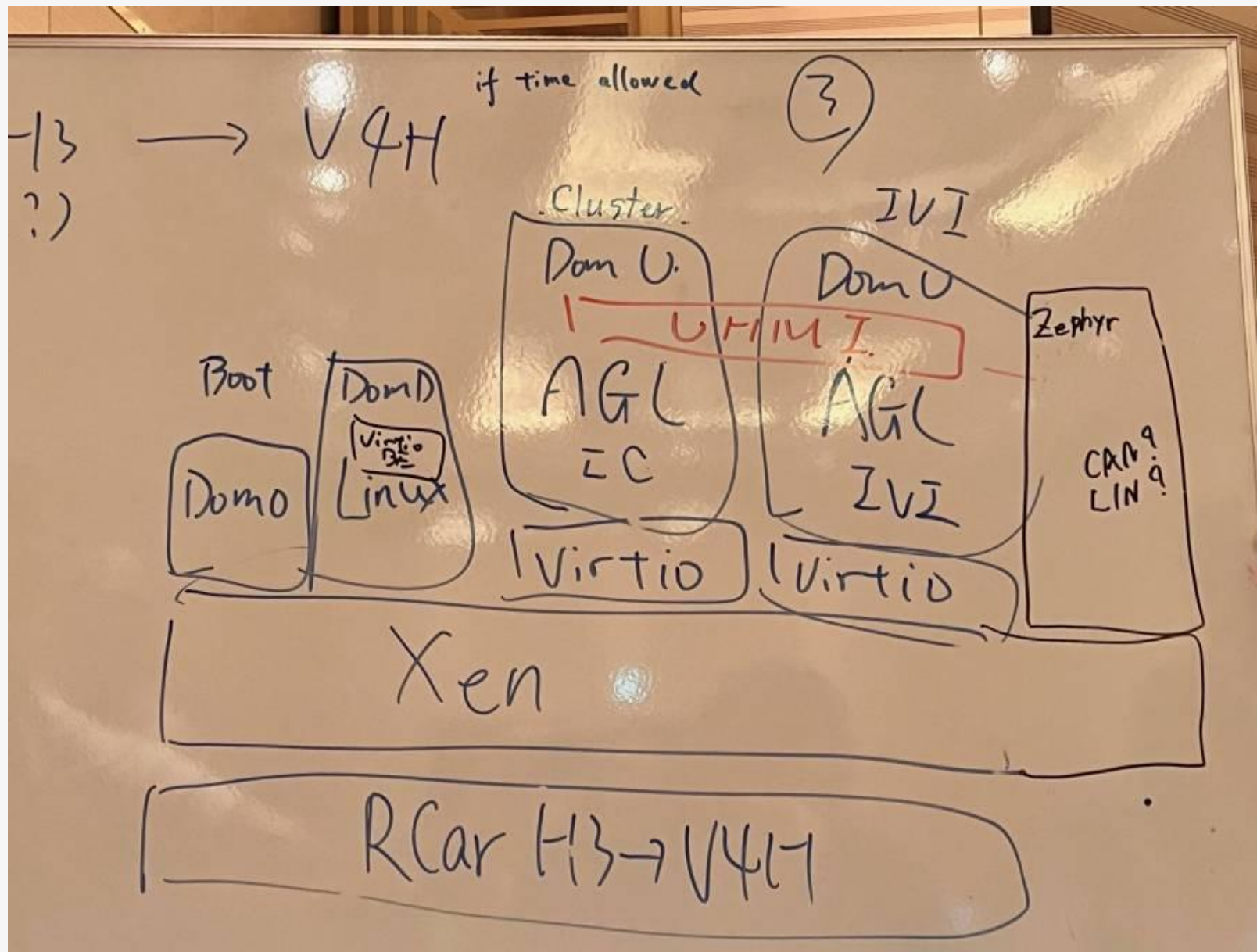
自動車業界のソフトウェアは

OSSとともに
進化している

量産開発を支える業界横断の
挑戦とその現在地

イベント 更新: 2026年06月10日

自動車業界のソフトウェアは、OSSとともに進化している——量産開発を支える業界横断の挑戦とその現在地 Honda Tech Talks#15



Automotive Grade Linux Launches Open Source SoDeV Reference Platform to Accelerate Software Defined Vehicles

By AGL | December 5, 2025 | Announcements

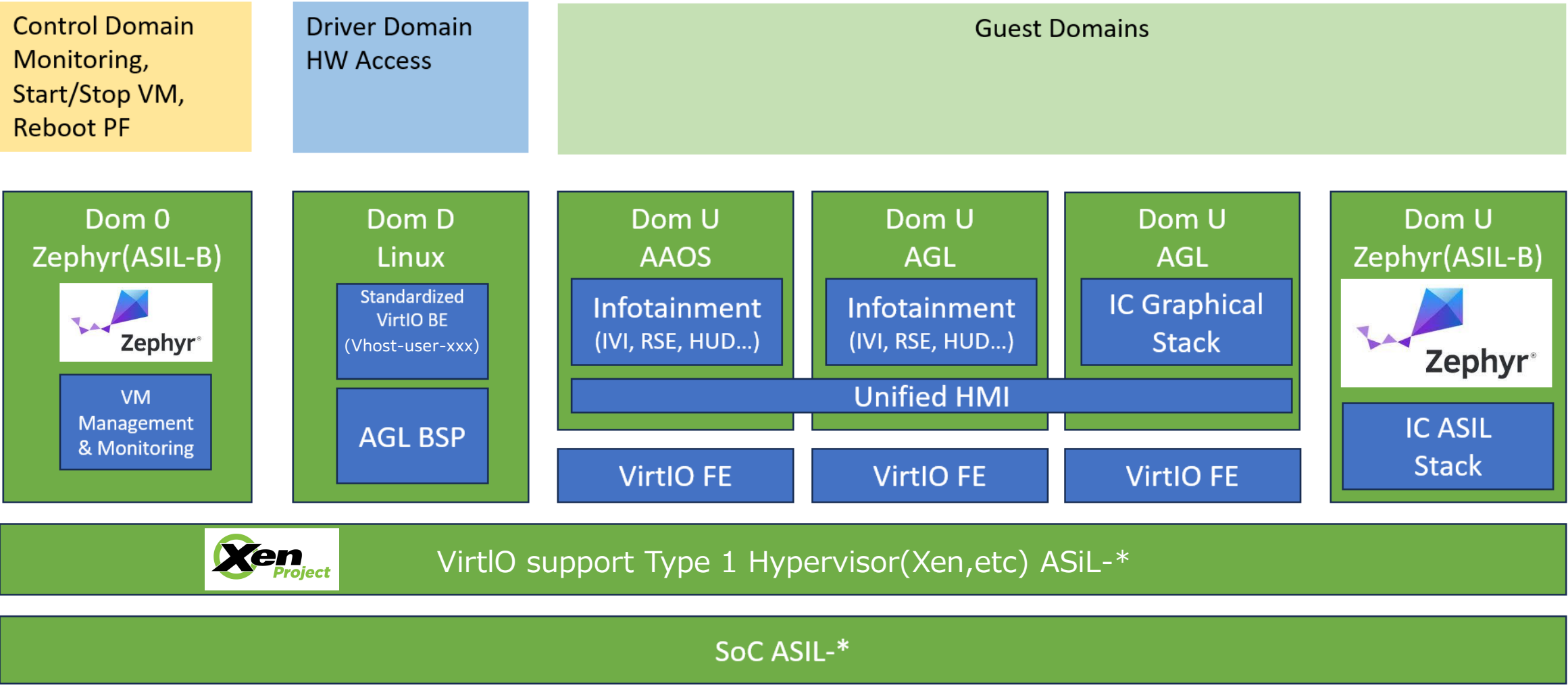
New AGL SoDeV reference platform supports ECU consolidation, virtualization, hardware abstraction and cloud integration for next-generation vehicles.

Highlights:

- **AGL SoDeV** is an open source reference platform for software defined vehicles enabling software-first development, independent of hardware constraints.
- The initiative is led by **Panasonic Automotive Systems, Honda, and the AGL SDV Expert Group**, with contributions from Toyota, Mazda, AISIN and Renesas.
- SoDeV integrates the **AGL Unified Code Base** with key open source projects including **Linux Containers, VirtIO, Xen, Yocto Project, Zephyr and ELISA**.
- Availability is planned for **early 2026** for virtual environments, cloud processors and automotive SoCs.

SAN FRANCISCO, December 5, 2025 – **Automotive Grade Linux** (AGL), a collaborative cross-industry effort developing an open source platform for connected car technologies, today announced SoDeV, a new open source SDV reference implementation that enables software-first development, decoupled from hardware constraints.

Led by Panasonic Automotive Systems, Honda and the AGL Software Defined Vehicle (SDV) Expert Group, SoDeV combines the AGL Unified Code Base (UCB) with multiple open source projects hosted by the Linux Foundation to support consolidation of ECUs, hardware abstraction using virtualization, and cloud integration for next-generation vehicles. Additional contributors to SoDeV include Toyota, Mazda, AISIN and Renesas.



 **sodev-demo-workspace** Public

Unwatch 1

main

1 Branch

0 Tags

Go to file

T

Add file

<> Code

 **hiroishii**

Bump external/meta-rcar-demo

354923a · 15 hours ago

27 Commits

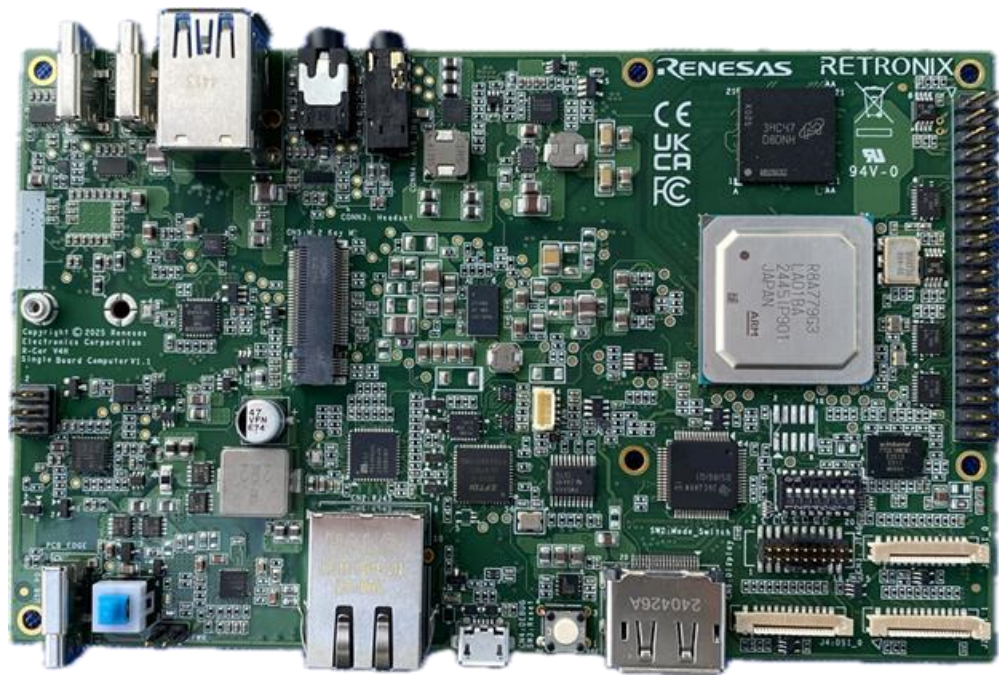
agl/patches	Change AGL repository to github fork	3 months ago
external	Bump external/meta-rcar-demo	15 hours ago
flatcar	Bump submodules	last week
.gitignore	Add DomAGL-IVI and DomAGL-IC guests	4 months ago
.gitmodules	add flatcar build process to build.sh	last month
LICENSE	Initial commit	5 months ago
README.md	Update README.md	5 months ago
build.sh	Disable Android and Flatcar guests by default	last week

README

Apache-2.0 license

sodev-demo-workspace



CPU

Arm® Cortex®-A76 cores at 1.8GHz x4

lockstep Arm® Cortex®-R52 cores at 1.4GHz x3

Campaign : 8GB memory R-Car V4H SBC "Sparrow Hawk" board for US\$299.00 (Limited quantity)

Basic Kit (Board only)

8GB Memory R-Car V4H SBC "Sparrow Hawk"

16GB Memory R-Car V4H SBC "Sparrow Hawk"

Complete Kit

8GB Memory R-Car V4H SBC "Sparrow Hawk" + Argon FAN, Micro USB cable + 32GB Micro SD card

16GB Memory R-Car V4H SBC "Sparrow Hawk" + Argon FAN, Micro USB cable + 32GB Micro SD card

Support Renesas R-Car V4H series Processor

8GB/16GB LPDDR5 memory

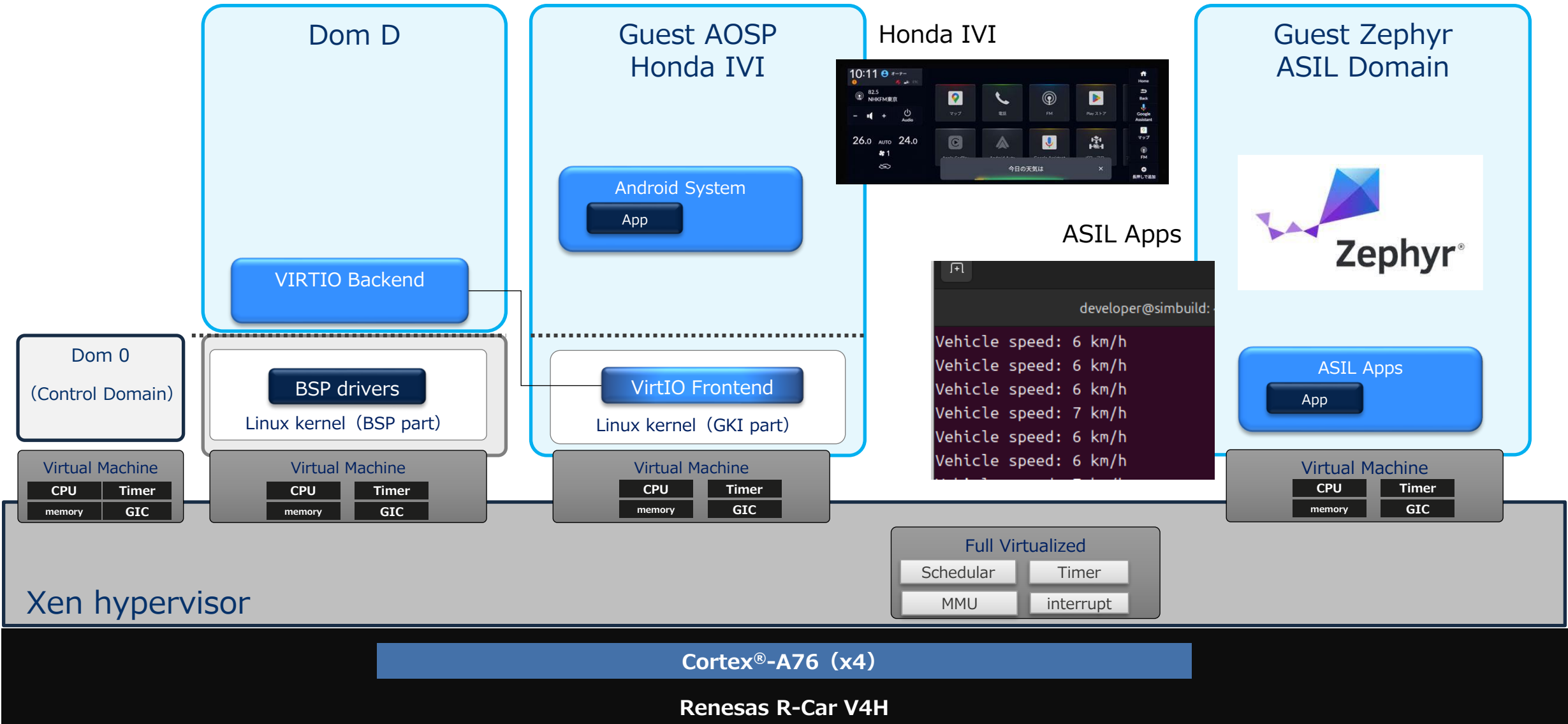
GPU AXM-8-256 @ 600MHz, >150 GFLOPS

Communication: CAN, Ethernet AVB

PCIe Gen4 x 1

Computing power up to 30 TOPS

Target Applications: Autonomous Mobile Manipulator, Smart Manufacturing, Smart Healthcare



announcements

Text Channels ▾

japanese-language

└ RPi5 SoDeV立ち上げスレッド

general

all-member-meetings

off-topic

weekly-dev-call

google-summer-of-code

industry-news

autoworld-japan

embedded-world

Voice Channels ▾

🔊 Lounge

🔊 Meeting Room

Expert Groups ▾

system-architecture-team



<https://github.com/xen-troops/meta-xt-prod-devel-rpi5/tree/main>
を使用して、下記の構成案を実現出来れば、SoDeVが実現出来るかもしれない
RPi5 (2x HDMI)

- └ Xen (Xen-Troops)
 - └ Dom0: Zephyr
 - └ DomD: Linux/Yocto (GPU/HDMI)
 - └ Weston (kiosk-shell, DRM backend)
 - └ HDMI-A-1: DomD UI (AGL IC)
 - └ HDMI-A-2: QEMU (SDL) = DomU GUI
 - └ QEMU device-model (SDL, virtio-vga-gl)
 - └ DomU: Linux/Yocto (VirtIO-GPU + AGL IVI)

GitHub

[GitHub - xen-troops/meta-xt-prod-devel-rpi5](#)

Contribute to xen-troops/meta-xt-prod-devel-rpi5 development by creating an account on GitHub.

xen-troops/meta-xt-prod-devel-rpi5



👤 5
Contributors

🕒 2
Issues

★ 19
Stars

🍴 15
Forks



出典 : Speaker Deck

© 2026 Honda Motor Co., Ltd. 無断転載・二次利用を禁じます。

<https://discord.com/channels/1278057614349176882/1478164066945204225/1478164702436528129>

Raspberry Pi **5** **16GB RAM**

For power users and professional applications.

Raspberry Pi 5 16GB is available now for \$305.



Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with cryptography extensions, 512KB per-core L2 caches and a 2MB shared L3 cache

VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.3

Dual 4Kp60 HDMI® display output with HDR support

4Kp60 HEVC decoder

LPDDR4X-4267 SDRAM (1GB, 2GB, 4GB, 8GB, and 16GB)

Dual-band 802.11ac Wi-Fi®

Bluetooth 5.0 / Bluetooth Low Energy (BLE)

microSD card slot, with support for high-speed SDR104 mode

2 × USB 3.0 ports, supporting simultaneous 5Gbps operation

2 × USB 2.0 ports

Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT)

2 × 4-lane MIPI camera/display transceivers

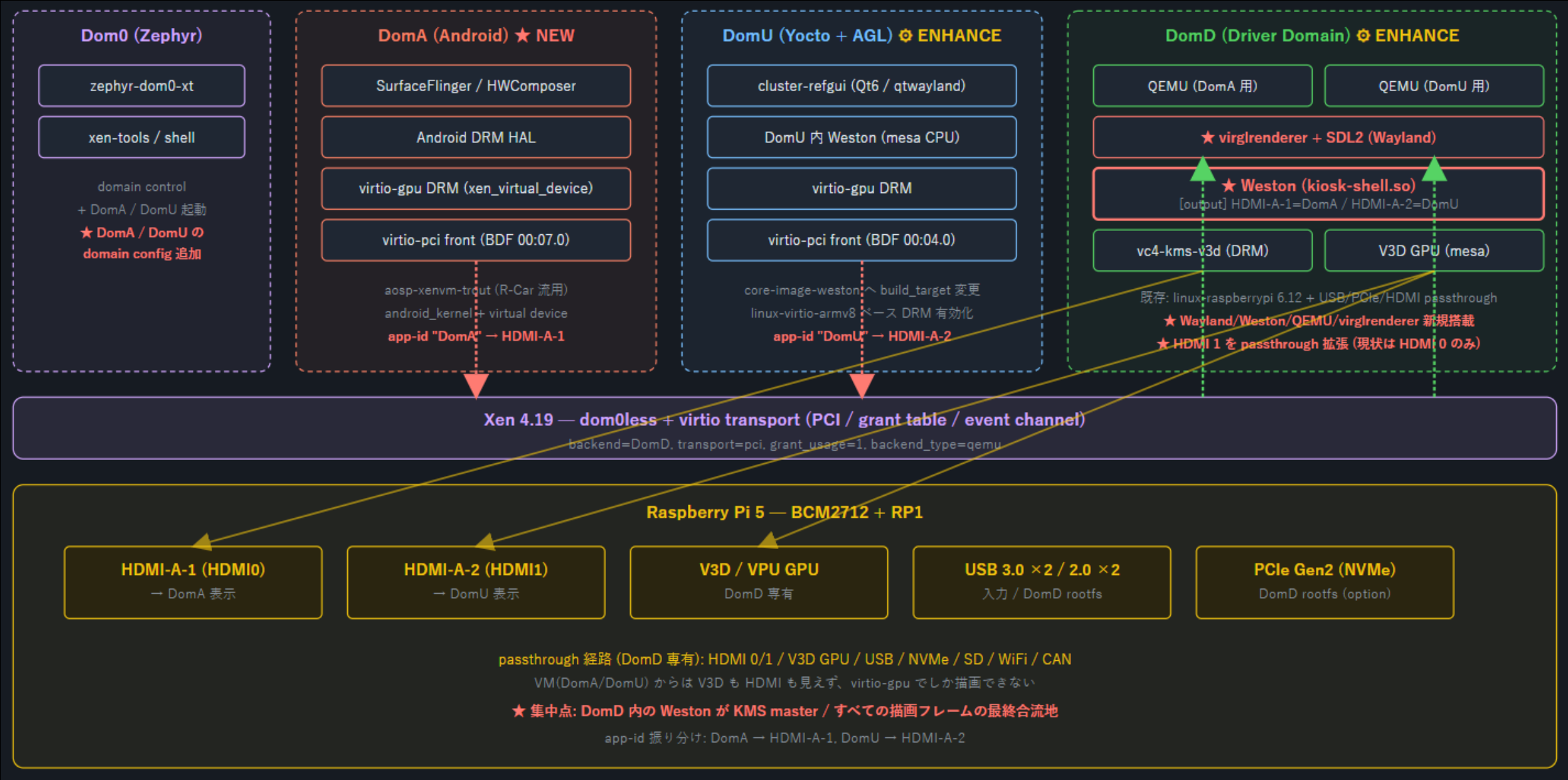
PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter)

5V/5A DC power via USB-C, with Power Delivery support

Raspberry Pi standard 40-pin header

Real-time clock (RTC), powered from external battery

Power button



観点	V4H SoDeV(参照)	RPi5 案X SoDeV(本作)
SoC	Renesas R-Car V4H(Gen4, r8a779g0)	Broadcom BCM2712 + RP1 southbridge
GPU	Imagination PowerVR(Img DDK)	VideoCore VII / V3D 7.1(Mesa vc4/v3d)
Xen	xen-troops fork(4.x)	4.21(upstream + 案X patches)
Dom0 OS	Linux	★Zephyr(薄・xenstore のみ)
Dom0 の役割	control + toolstack + xenstore	xenstore サーバのみ(制御せず)
toolstack(xl/libxl)	Dom0	★DomD(分離)
xenstore サーバ	Dom0 Linux(xenstored)	★Zephyr 内蔵 xenstore
DomD 起動方式	Dom0 が <code>xl create</code>	★dom0less + direct-map + static-mem
DomU/DomA 起動	Dom0 の toolstack	DomD の toolstack
GPU 経路(guest→host)	guest Mesa virgl → virtio-gpu-gl → virglrenderer → PowerVR DDK → PowerVR HW	guest Mesa virgl → virtio-gpu-gl → virglrenderer → Mesa v3d → V3D HW (経路の型は同一)
GPU 仮想化方式	virtio-gpu-gl(virgl 仲介)	virtio-gpu-gl(virgl 仲介)= 同一。native passthrough は RPi5 では不可
storage	eMMC 等(DomD/Dom0)	★DomD が SD を SDHCI passthrough
I/O southbridge	(SoC 内蔵)	★RP1 全体を DomD へ(USB/Eth/GPIO/FAN)
表示	Display(DU/DSI)/ HDMI	HDMI-A-1=DomU / HDMI-A-2=DomA の 2 画面
DomU	AGL cluster(Flutter・virtio)	AGL cluster(Flutter・virtio)= 設計同一
DomA	AAOS(Android・virtio)	AAOS17(V4H イメージを無改変流用)
DomA virtio BDF	addr1-7(blk/vsock/serial/net/tablet/gpu-gl)	addr1-7 = 完全同一
DomA memory	(board 依存)	8192MB(16GB 機・AAOS17 は ≥6144 必須)

ドメイン	vCPU	Linux Dom0 時	Zephyr Dom0 時
Dom0	1	≈0%	0%
DomD	2	110%	125%
DomU	1	48%	52%
DomA	2	6% (AAOS15)	9%

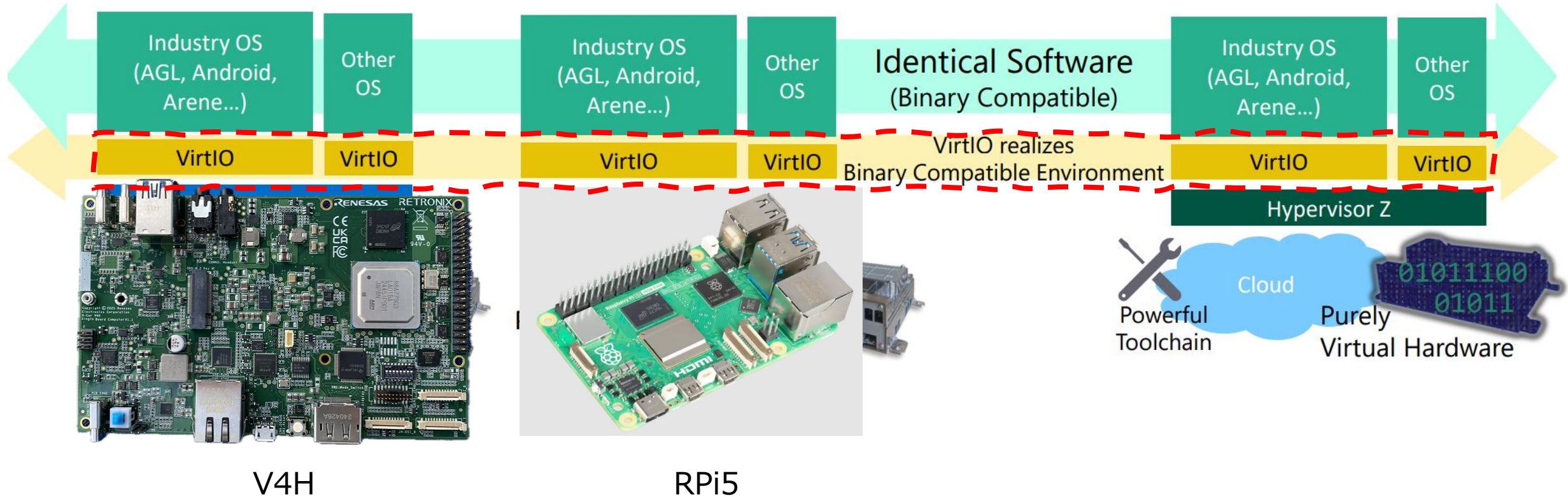
	V4H (PowerVR)	RPi5 案X (V3D)	差
総合 GPU 使用率 (同一デモ)	≈34% (実測 32-45%)	≈94% (render・飽和寸前)	≈2.8-3×
render エンジン単体	3D 18%	render 94%	≈5×
GPU 余裕	~66%	~6% (ほぼ飽和)	—
GPU 理論 GFLOPS	153.6	92.16	1.67×
• RPi5 が高負荷なのは GPU HW 性能差が主因 (PowerVR AXM-8-256 ≧ V3D 7.1)。virgl tax・単一 GPU 共有は両機共通で差にならない。 • RPi5 側で負荷を下げる手は ワークロード削減のみ (AGL fps cap 60→30・AAOS 解像度↓)。実測で fps cap 60→30Hz は DomU render を -26% 削減。 • CPU・メモリ・帯域は RPi5 で飽和には至らず、律速は一貫して GPU/DRAM 帯域。			



AAOS17

AAOS17

移植



 **sodev-demo-workspace-rpi** Public Unwatch

main

2 Branches

0 Tags

Go to file

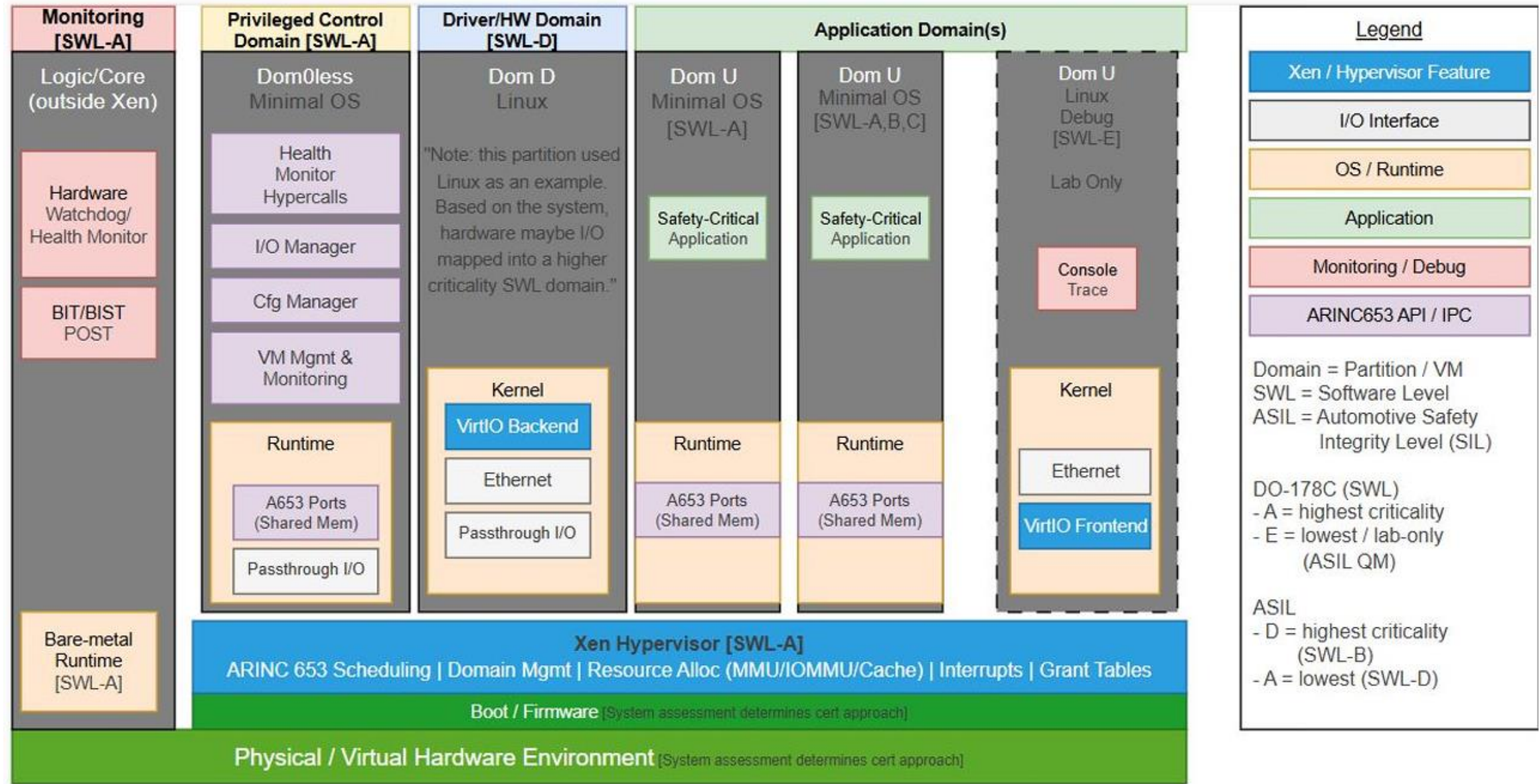
T

Add file

<> Code

 **yuichi-kusakabe** build.sh: name the SD image after the configuration in it 734b1e2 · 2 days ago 26 Commits

docker	build: docker: Downgrade tar version to workaround fakeroot...	last week
docs	build.sh: name the SD image after the configuration in it	2 days ago
external	build: SoDeV RPi5 Xen demo workspace (moulin, docker, sub...	2 weeks ago
meta-rpi-sodev	meta-xt-dom0-zephyr: add the rpi_4b board and guest configs	2 days ago
tools	build.sh: select the board, and the guards that were missing	2 days ago
.gitignore	build.sh: name the SD image after the configuration in it	2 days ago
.gitmodules	build: SoDeV RPi5 Xen demo workspace (moulin, docker, sub...	2 weeks ago
LICENSE	Initial commit	5 months ago
README.md	build.sh: select the board, and the guards that were missing	2 days ago
build.sh	build.sh: name the SD image after the configuration in it	2 days ago
rpi4-sodev.yaml	rpi4-sodev.yaml: add the Raspberry Pi 4 product configuration	2 days ago
rpi5-sodev.yaml	rpi4-sodev.yaml: add the Raspberry Pi 4 product configuration	2 days ago



RPi4/5へのSoDeVの移植は、全てAIで実施しました

デモブースでRPi5の動作を確認してみてください

Linux PC (32コア)
Claude Code

UART

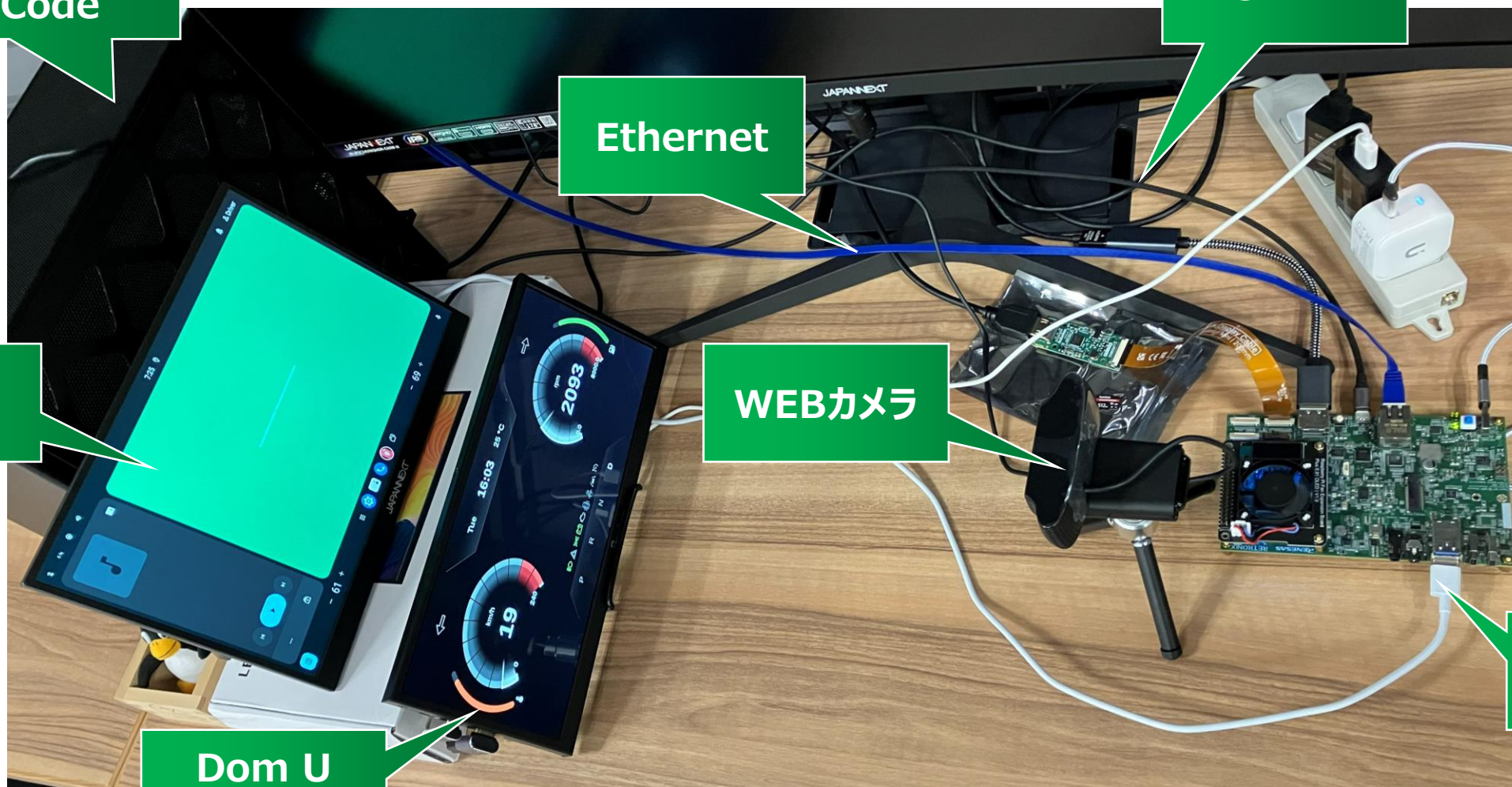
Ethernet

Dom A
AAOS17

WEBカメラ

Dom U
AGL IC

V4H



「Linux」カーネルでAI支援コードに関する新ルールを導入

Generated-byではなくAssisted-byが最終的に選ばれたのは、3つの要因に基づいた慎重な判断によるものだ。第一には、より正確だからだ。カーネル開発におけるAIの利用は、コード補完、リファクタリングの提案、テスト生成といった補助的なものが大半であり、完全なコード生成ではない。第二に、このタグの形式が「Reviewed-by」「Tested-by」「Co-developed-by」といった既存のメタデータタグと整合している点である。最後に、Assisted-byという表現であれば、そのコードが疑わしいものや二流であるという含みを持たせずにツールの役割を説明できるからだ。

結論(推計)

- 工数圧縮 ≈ 4〜7倍 (従来 約150〜245 senior-eng-day → 実際 約30〜40 eng-day / 1名+AI・約6週間)
- 専門役割 ≈ 5職種を1名でカバー (Xen HV / 組込Linux・Yocto / Android・AAOS / GPU・グラフィクス / bootloader)
- カレンダー短縮 ≈ 3〜4倍 (チーム4〜6ヶ月相当 → 単独6週間)

定量できた実データ (測定値)

指標	実数
追跡タスク	約200件 (#184-#383・大半完了)
コミット (本人)	86 (うち sodev-dom0-zephyr 81)
Xen patch 群	90本 curated + Zephyr 16本(PR/DCO-ready) + upstream 4本
技術文書	28 md (REPORT7/RUNBOOK7/PLAN3/COMPARE2/DESIGN1/他)
SD 成果イメージ	11 (deliverable + backup)
期間 / 人数	約6週間 (06月上旬〜07/10) / エンジニア1名 + Claude Code

→ 中央値 ≈ 190 eng-day $\approx 7 \sim 11$ person-month（この全領域に精通した単独の熟練者は非現実的なので、実際は4～5名の専門家チーム × 約1.5～2.5ヶ月相当）。

AI が効いた主因（半定量）

- 調査の並列委譲：本日だけでも Explore 3体が、手作業なら各数時間の grep/読解を数分で完了。全期間で数百時間規模の探索を圧縮。
- 深い真因解析：Xen 内部・altp2m ABI・error-5 等の仮説生成とコード読解を高速反復。
- 文書 28本を主に AI 起草（通常は大きな時間食い）。
- 多角レビュー（PR patch）を並列レビュアーで実施。

正直な留保（過大評価しないため）

1. 測定ではなく推計。反実仮想は業界的な専門家判断。
2. AI も手戻りを発生（SD イメージ誤り→ブートループ、いくつかの誤診断）を人間が是正 → AI 時間はゼロコストでない。
3. 既存資産(xen-troops V4H 参照実装)を大量流用 → ゼロからの創出ではない。
4. 人間の専門性が必須（本人の Xen/組込知識）→ これは 人間+AI+既存資産 の合算効果で、AI 単独ではない。
5. 実機制約（SD 書換・board アクセス）は AI と無関係に律速。

drwxr-xr-x	2	root	root	4096	May	30	01:27	h_domu/	drwxr-xr-x	2	root	root	4096	May	31	21:39	v2_android/
drwxr-xr-x	2	root	root	4096	May	28	16:23	honda_android/	drwxr-xr-x	2	root	root	4096	May	31	21:34	v2_boot/
drwxr-xr-x	2	root	root	4096	May	28	18:21	honda_bak_boot/	drwxr-xr-x	2	root	root	4096	May	31	21:34	v2_domd/
drwxr-xr-x	2	root	root	4096	May	28	18:15	honda_bak_domd/	drwxr-xr-x	2	root	root	4096	May	31	21:09	v4h_p1/
drwxr-xr-x	2	root	root	4096	May	28	16:23	honda_boot/	drwxr-xr-x	2	root	root	4096	May	31	21:09	v4h_p2/
drwxr-xr-x	2	root	root	4096	May	28	16:23	honda_{boot,domd,domu,android}/	drwxr-xr-x	2	root	root	4096	May	30	01:28	v_domd/
drwxr-xr-x	2	root	root	4096	May	28	16:23	honda_domd/	drwxr-xr-x	2	root	root	4096	May	30	01:25	v_domu/
drwxr-xr-x	2	root	root	4096	May	29	15:42	honda_domd_diff/	drwxr-xr-x	2	root	root	4096	May	29	20:37	vendor_check/
drwxr-xr-x	2	root	root	4096	May	29	15:54	honda_domd_src/	drwxr-xr-x	2	root	root	4096	May	29	16:59	yocto_ext4/
drwxr-xr-x	2	root	root	4096	May	28	16:23	honda_domu/									
drwxr-xr-x	2	root	root	4096	May	29	15:22	honda_sda2_check/									
drwxr-xr-x	2	root	root	4096	May	29	17:50	honda_sda2_chk/									
drwxr-xr-x	2	root	root	4096	May	29	18:54	honda_sda2_compare/									
drwxr-xr-x	2	root	root	4096	May	29	20:37	honda_sda3_check/									
drwxr-xr-x	2	root	root	4096	May	29	21:19	honda_sda3_explore/									
drwxr-xr-x	6	root	root	4096	Jul	7	22:59	pr-review/									
drwxr-xr-x	2	root	root	4096	May	31	21:11	ref_p1/									
drwxr-xr-x	2	root	root	4096	Jun	16	10:15	sd1/									
drwxr-xr-x	2	root	root	4096	Jun	16	10:15	sd2/									
drwxr-xr-x	2	root	root	4096	Jun	16	10:15	sd3/									
drwxr-xr-x	2	root	root	4096	Jun	16	10:36	sd6/									
drwxr-xr-x	2	root	root	4096	May	29	19:16	sda1_write/									
drwxr-xr-x	2	root	root	4096	May	29	15:26	sda2/									
drwxr-xr-x	2	root	root	4096	May	29	15:25	sda2_check/									
drwxr-xr-x	2	root	root	4096	May	29	19:16	sda2_write/									
drwxr-xr-x	2	root	root	4096	May	29	01:02	sda_boot/									
drwxr-xr-x	2	root	root	4096	May	28	18:25	sda_domd/									
drwxr-xr-x	2	root	root	4096	May	28	18:21	sdb_boot/									
drwxr-xr-x	2	root	root	4096	May	28	18:21	sdb_domd/									
drwxr-xr-x	2	root	root	4096	May	28	14:45	sd_boot/									
drwxr-xr-x	2	root	root	4096	Jun	1	11:26	sd_id/									
drwxr-xr-x	2	root	root	4096	Jun	11	12:49	sdp1/									
drwxr-xr-x	2	root	root	4096	May	29	17:09	sodev_ext4/									
drwxr-xr-x	2	root	root	4096	May	29	19:09	sodev_rootfs_compare/									
drwxr-xr-x	2	root	root	4096	May	29	19:15	test_sodev_ext4/									
drwxr-xr-x	2	root	root	4096	May	30	19:54	tmp_domd/									

良かった点

- ビルドからデバッグまで全て実施出来る
- 24時間回せる
- uart、SSH、WEBカメラまで使用出来る
- OSSとの親和性が非常に高い

課題

- 計算量
- uartを使用出来るが苦手
- SDカードの書き換えは手動
- 難しい作業は脱線しようとする
- Dom 0 (Linux) と
Dom D (Linux) のよく間違える
- AI止まると効率が非常に下がる
- 出力量が非常に多く、人が追いつけ無い
- 3～4パラが限界

HONDA
The Power of Dreams

How we move you.
CREATE ► TRANSCEND, AUGMENT