

SW First の本質を考える

基調講演 @ Osaka NDS Embedded Cross Online Forum #12

宗像尚郎

Automotive Grade Linux Advisory board member
Linux Foundation board member

2021-2-12

自己紹介

ルネサスエレクトロニクス という半導体メーカーで働いています

■ ルネサスの概要と私の担当業務

- ルネサスエレクトロニクス = (日立+三菱) + NEC の半導体事業部門
- 車載向け SOC (System on Chip) のグローバルリーディングサプライヤーです
- R-CarH3 = 64bit ARM Octa-core + GPU + LPDDR4 8G、1,384pin BGA
- 産業コンソーシアム活動、コミュニティ活動でも指導的な役割をもっています
- Linux Foundation のボードメンバー です
- Automotive Grade Linux、yocto project の理事も務めています
- 社内オープンソース開発チームを運営、コミュニティ開発でも顕著な実績
- Xen FUSA SIG メンバーとして、Xen の車載分野への適用 を検討しています

ルネサスはオープンソース開発活動への貢献実績において日本を代表する企業です

ソフトウェアファーストとは何か

『ソフトウェアファースト』 by 及川卓也



ソフトウェアファースト (=SF) の命名者

- 及川氏 = IT 技術の急激な進化の現場にいた当事者
 - Dec → Microsoft → Google
 - Increments (Qiita 運営母体)
 - 2017 から個人としてコンサルタント活動をスタート
 - Denso とも技術顧問契約を締結
- 及川氏がこの本で問題提起しているのは
 - 日本企業は IT 技術を軽視 (= 社外に丸投げ) してきた
 - 各種 IT サービスのクラウド化への対応遅れ が顕著な例
 - その結果、日本の企業は GAFA などの先端 IT 企業から遅れをとってしまっているようにみえる
 - 先端 IT 技術はデジタルトランスフォーメーションの担い手 なので早急なキャッチアップが必要である

トヨタが組織再編、ソフトウェアファーストを具現化する体制に

20年3月、トヨタ社長の豊田章男氏はNTTとの提携発表の場で、「ソフトウェアファースト」の開発体制に移行することを宣言した。ソフトとデータを活用し、自動車の機能向上を実現する構想だ。「ソフトを先行して実装し、自動車の走行時にデータを収集する。AI(人工知能)をレベルアップさせて、ある段階でソフトを更新して機能を追加できるようにする」(豊田氏)。組織再編は、ソフト第一を実現する手段の1つになる。

従来の車両開発は、ハードとソフトの一体開発が基本だった。車両の全面改良に併せて、電子制御ユニット(ECU)とソフトをセットで開発するものだった。ECUの能力に見合うムダの少ないソフトを開発しやすい一方で、進化の遅いハードにソフト開発がしばられる課題があった。

ソフト第一の開発体制への移行で鍵を握るのが、18年に設立した自動運転ソフト子会社トヨタ・リサーチ・インスティテュート・アドバンスト・デベロップメント(TRI-AD)である(図2)。「Arene(アリーネ)」と呼ぶソフト基盤を開発する。同基盤は、安全で信頼性の高い車載ソフトの統合開発環境(IDE: Integrated Development Environment)と言えそうなもの。この自動車版IDEの出来栄が、トヨタ全体の今後のソフト開発効率を左右するだろう。

引用元 = <https://xtech.nikkei.com/atcl/nxt/mag/at/18/00006/00303/> (日経 XTECH 2020.11.30 記事)

氏名	新担当	現担当
* 白水 浩一	クルマ開発センター <u>ソフトウェアファースト担当</u> (Chief Project Leader)	クルマ開発センター 制御電子プラットフォーム領域(統括部長)

2021年1月
Toyota 人事

<https://global.toyota/jp/newsroom/corporate/34423757.html?>

よくある誤解 (私の身の回りだけ？)

組み込み系の SW の開発現場は、これまでずっと **ソフトウェアラスト** だった？

- これまで組み込みソフトウェアの開発現場では
 - 組み込み機器開発では **常にプログラムを実行するターゲット** が先に決まっている
 - プログラマーの仕事は **ハードウェア性能を最大限に引き出す制御手順** を考えること
 - 時には **ハードウェアの制約 (不具合) 回避** もソフトウェアの役割とされた
 - 開発予算配分ではソフトウェア開発費が削減対象コストになることも多い
- ソフトウェアファーストになると (以下は全部ウソです)
 - **どんなハードウェアでも動く (= 移植可能な) ソフトウェア** が求められる
 - 機器の優位性はソフトウェアによって生み出されるので **ハードは何でもよくなる**
 - 特殊なコーディングが必要な **HW アクセラレータ活用よりも全て CPU 処理** としたい
 - ハードウェア開発よりも **ソフトウェア開発費用に多くの費用を使うのは当然**

「ソフトウェアがハードウェアに優先する」などとは誰も言っていない点に注意

PC/サーバー向けソフトウェアにおける先端 IT 技術活用（１）

SaaS、IaaS 活用による情シス体制の抜本的な見直し

- オンプレミス（＝ 社内の情報システム部門で運用）
 - インフラ投資（固定資産購入）は大きな財務的な負担
 - 原価償却期間（５年）の継続使用が前提
 - 社内に 運用要員 の確保が必要
- クラウド（プライベート／パブリック）
 - 投資 → 変動費用化（使った分だけ支払う）
 - デマンド変化に応じて 設備規模も逐次変動 できる
 - 最先端の情報技術 をいち早くビジネスに活用可能
 - 社内の運用体制が不要となり 組織のスリム化 を達成

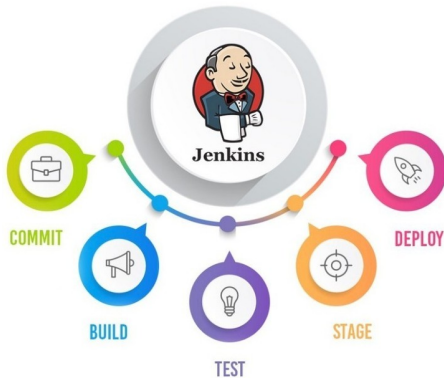


原価償却期間に縛られない弾力的な投資によって先端情報技術の即戦力化が可能に

PC/サーバー向けソフトウェア における先端 IT 技術活用（２）

最新情報技術を活用した SW 開発効率アップ

- ソフトウェア 開発プロセス連系の自動化
 - 自動コードレビュー（フォーマット確認）
 - 累積リグレーションシナリオ の自動確認
 - バージョン管理システムとの自動連係
 - リリース管理システムとの自動連係
- 開発マネジメント の品質向上
 - 分散／大規模開発 に適用可能
 - 開発者の スキルレベルの自動スコア判定
 - 各開発者の 作業負荷の平準化



CI/CD パイプラインの自動化により SW 生産性の大幅向上と品質改善を両立

PC/サーバー向けソフトウェア における先端 IT 技術活用（3）

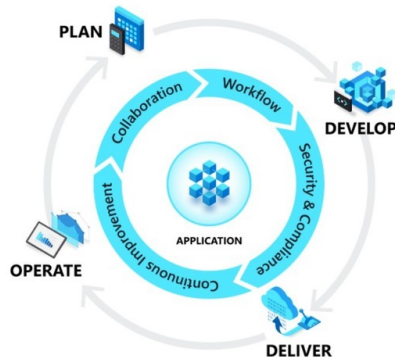
DevOps を実現するアジャイル開発スキーム

■ DevOps とは

- 開発（Dev）と運用（Ops）を有機連携
- コードを小さく生んで継続的に育てる
- Release Early, Release Often の原則
- 開発着手時に 完全な要件定義は必要ない

■ ニーズ変化へ最短数時間で対応

- Ops → Dev のフィードバックを強化
- ニーズ変化への逐次対応力 がキー
- 高頻度更新が可能なデリバリーメカニズム



ユーザーニーズ変化に迅速追従できる **SW** の開発／提供スキームが企業の戦略要因に

組み込みソフトウェア開発における先端 IT 技術活用

組み込みソフトウェアの開発現場も 近年の技術革新の恩恵 にはあずかっているが

- コンパイル時間の大幅な短縮（CPU 能力、メモリーサイズ、ストレージ性能等）
- 分散開発に対応した バージョン管理ツール の充実（git など）
- Window ベースのグラフィカルな 統合開発環境（VSCode, Android Studio など）
- 個人でも購入可能な 廉価な開発用ボード の充実（StarterKit, RaspberryPie など）
- 一方で 開発規模（= コーディング行数、参加プログラマー数）の爆発的拡大
- サイバーセキュリティ対応、機能追加のための ソフトウェアリリース頻度拡大
- ネットワーク経由で接続された サーバー、クラウド連携への新規対応

IT ハードウェア活用は進んだが、先端ソフトウェアやサービスの活用は進んでいない

SF の神髄は 先端 IT 技術を活用したアーキテクチャ構想力 なのだろう

第三次産業革命 (=ICT 革命)

- レディメイドな情報通信技術が存在
 - データベース、LAN/WAN 等
- ビジネス適用は比較的シンプル
- 故に日本では外注化が進行した
- 技術課題の本質は「実装」
- 日本は「実装」が上手く成果を出せた
- 結果として各社独自の実装が乱立しガラパゴス化が進行

第四次産業革命 (=DX 革命)

- 新しい情報技術メソッド が次々に登場
 - AI、ブロックチェーン、Web 技術
- 技術の組み合わせには目利き が必要
- 先行企業は社内人材の育成に注力
- オープンな技術の組み合わせによって新たな価値を創造していく構想力
- 差異化の源泉は「方式設計力」
- 日本の課題は「方式設計力」の醸成

方式設計力 (=アーキテクト力) を強化しなければ **DX** 時代には勝ち残れない

組み込みソフトウェア開発における 先端 IT 技術活用の可能性

(参考) トヨタの 自動運転車開発現場での AI、モダン IT 技術 の適用

トヨタ自動車でAI（人工知能）を研究するTRI（Toyota Research Institute）は1月14日（現地時間）、ドリフト中の後輪駆動車を制御できるアーキテクチャを「GRスープラ」に適用して、AIを活用してプロドライバーが行なう「ドリフト」を再現する様子を公開した。

この取り組みは、トヨタリサーチインスティテュートとスタンフォード大学のDynamic DesignLabが協力して、プロドライバーの能力と自動運転技術をどのように組み合わせるのかについて研究を行っているもの。



<https://car.watch.impress.co.jp/docs/news/1300306.html>



人間と機械の最終決戦において、Google DeepMindのデミス・ハサビス（写真）が開発した「アルファ碁」は2017年、人類最強の棋士と呼ばれた中国の柯潔に勝利した。（写真：AP/アフロ）
実はこれと同じことが、自動運転の世界でも可能です。

はじめは人間ドライバーの経験とスキルから学び、多くの走行データを集めていくことで、経験を蓄積する。それをロジックと組み合わせれば、高い運転スキルを持ったシステムを作れるでしょう。

その後は、膨大な走行データを集めてゆく工夫が必要です。

例えば、ある走行スキルを身につけるのに、1万時間の走行経験が必要だとします。1つのロボットを1万時間走らせることもできますが、クラウド・ロボティクスの概念を応用すれば、100台のロボットに100時間ずつ走らせればいい。

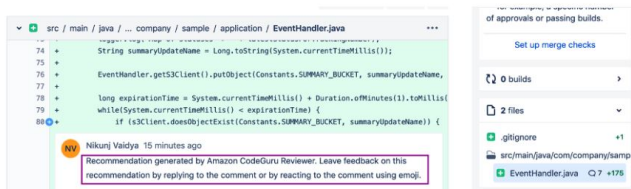


<https://newspicks.com/news/3733474/body/>

(参考) Amazon CodeGuru (Reviewer / Profiler)

CodeGuru Reviewer は、ソースコードのプルリクエストを自動的に分析して重要な問題を見つけ、そのプルリクエスト内でコードの欠陥を直接解決するためのインテリジェントな推奨事項を提供します。CodeGuru Reviewer では、コードの品質に関する問題を、次のように大きく 9 つのカテゴリを特定します。

- AWS のベストプラクティス: AWS API (ポーリング、ページ区切りなど) の使用方法を修正します。
- Java と Python (プレビューで利用可能) のベストプラクティス: 一般的な Java と Python の言語とライブラリ機能の正しい使用
- 同時実行: 機能上の障害を起こしている同期不良、あるいはパフォーマンスを低下させている過剰な同期を検出します。
- デッドロック: 同時実行するスレッド間の割り当てをチェックする
- リソースリーク: リソースの処理 (データベース接続の解放など) を修正する
- 機密情報のリーク: 個人識別情報 (ログインしているクレジットカードの詳細など) の漏洩を検出する
- 一般的なコード上のバグ: Lambda 関数の呼び出し時にクライアントを作成していないなどの、発見しづらい問題を検出する
- クローンコード: 統合することでコードの保守性を高められる可能性のある、重複コードを特定する
- 入力検証: 信頼されないソースから送られる、不適当な形態の、もしくは悪意のあるデータをチェックする



https://aws.amazon.com/jp/codeguru/features/#Amazon_CodeGuru_Reviewer

(参考) arm Simulation Models

Cycle Models

- 100% cycle accurate models of Arm IP, compiled directly from RTL
- Prove designs before manufacture
- Develop bare metal software early
- Accelerate system-level debugging

[Learn more](#)

Fast Models

- Fast and functionally accurate programmers view models of Arm IP
- Pre-hardware software development
- Integrates with 3rd party IP models

[Learn more](#)

Fixed Virtual Platforms

- Ready-to-use, fast and functionally accurate simulation models
- Pre-configured virtual prototypes
- Easy to deploy and scale
- Fully debuggable and automatable

[Learn more](#)

AMBA TLM Library

- AMBA Transaction-Level Modeling library
- Model approximately-timed and cycle accurate ports
- C++ pre-compiled binary library

[Learn more](#)

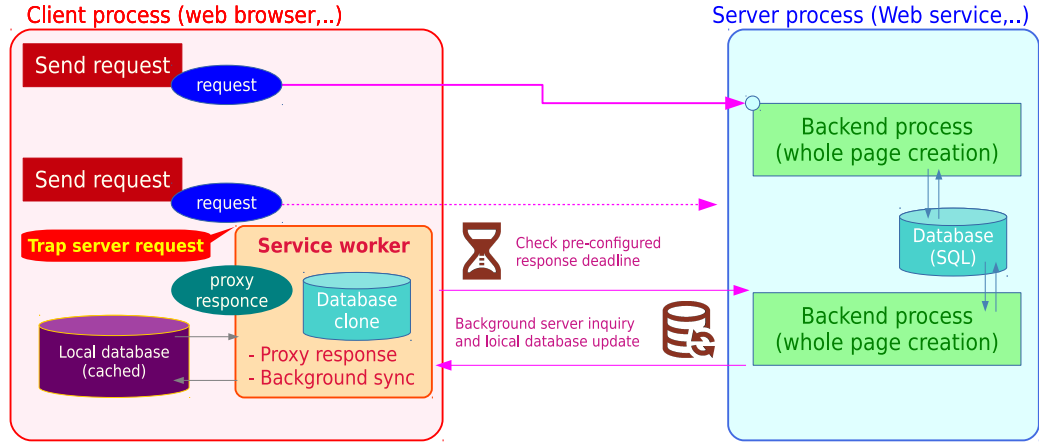
Arm Flexible Access Models

- Select the optimum Arm IP and IP configuration
- Evaluate and design solutions before committing to production

[Learn more](#)

<https://developer.arm.com/tools-and-software/simulation-models>

(参考) Cloud Native Application (service worker)



モダン IT 技術の適用可能範囲は限定される?

ソフトウェアファーストを支える モダンな IT 技術

IT/Cloud → 組み込み開発にも波及中

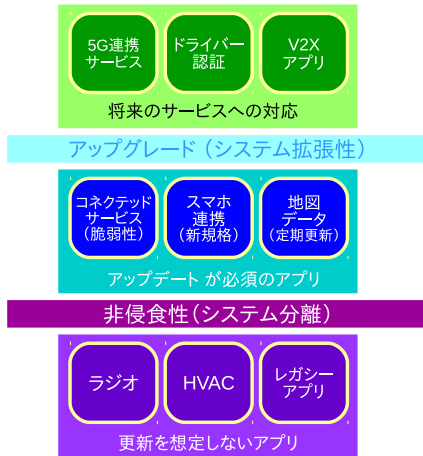
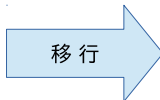
- git (分散開発、履歴管理)
- CI/CD (開発ステージのパイプライン化)
- Test Automation
- DevOps (ニーズの取り込み)
- 開発進捗ステータスの可視化
- クラウド活用
 - 計算資源の動的割付け
 - AI によるコードインスペクション

これからの組み込み開発での活用課題

- VPF (仮想開発環境)
 - エミュレータ (バイナリー互換)
 - シミュレータ (アプリ API 互換)
 - 仮想実行 (デジタルツイン)
- エッジ／クラウド連携
 - IoT 応用 (イベント／データ収集)
 - ワークロード分散 (解析、学習)
 - アプリ協調 (クラウドネイティブ)
 - リモート制御

今後は組み込み機器向けソフト開発にも先進 IT 技術の適用が進むと考えられる

システム内に 属性の異なるソフトウェアが同居 する時代に



今後のシステムでは

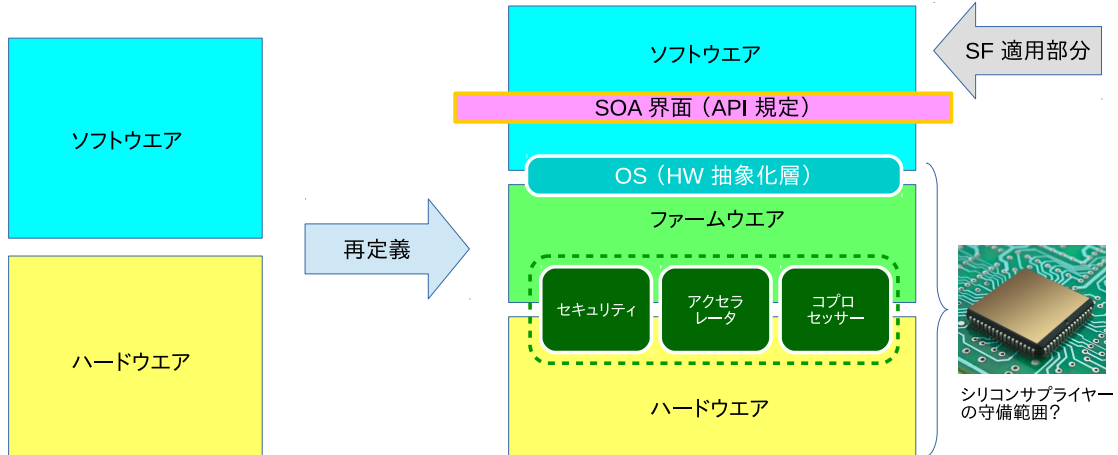
- 1) アップデート
- 2) アップグレード
(= インストール)

の両方への対応が必要

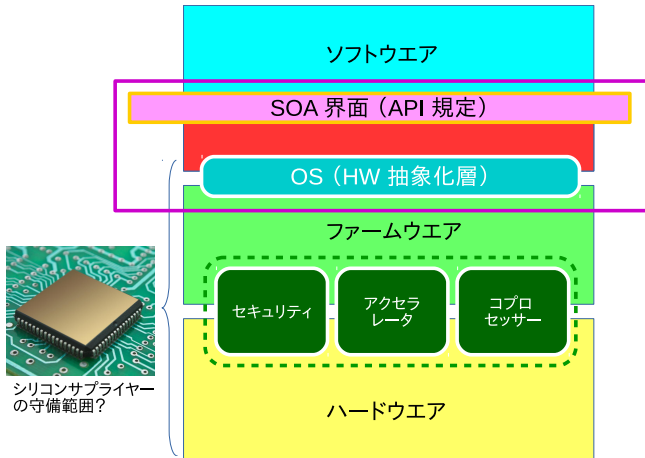
また、属性が異なる SW
カテゴリー間には **非侵食性
(FFI) 担保**が求められる

モダンIT 技術の導入には
システムパーティショニング
をペアで考える必要あり

2 層階層 → 3 層階層 でモダン IT 技術適用範囲を明確に



組み込み製品で SF を具現化するには **新しい SW 基盤** の整備が必要か？



認証

PKG
管理

監査

仮想化

SF を実現する **新しい SW 基盤** が必要に

<https://japan.zdnet.com/article/35164607/>



SolarWinds製品のハッキングはどれほど深刻か

Steven J. Vaughan-Nichols (Special to ZDNet.com) 翻訳校正: 編集部 2021-01-05 13:16

その理由は、Orionの脆弱性を悪用して、マルウェアの「SUPERNOVA」と「COSMICGALE」がインストールされたためだ。セキュリティホールの「2020-10148」により、攻撃者はOrion APIの認証を迂回して、Orion上でリモートコードを実行できるという。

上院情報委員会の幹部であるMark Warner上院議員は、ハッキングの被害は最初の予想よりも「はるかに、はるかに悪いようだ」とNew York Times紙に語っている。そして「その規模は拡大し続けている」という。