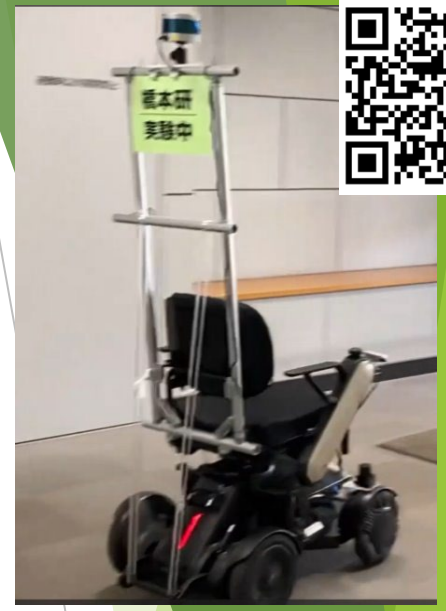
サイバー攻撃に対応した
安全・セキュリティ統合設計の推進

- 合同会社 Forehacks 代表社員
- 名古屋工業大学 ものづくり DX 研究所 外部研究員
- 佐々木泰斗

自己紹介　佐々木泰斗
合同会社 Forehacks 代表社員



(略歴)
2021 年　名古屋工業大学 工学部 卒業
2022 年　合同会社 Forehacks 代表社員　(名古屋工業大学発ベンチャー企業)
　　ミラノ工科大学 Computer Science and Engineering NECST Lab

2023 年 名古屋工業大学大学院 工学研究科 工学専攻 修士課程 修了
　　名古屋工業大学 ものづくりDX研究所 研究員
　　Windows エンジニア
　　(クラウド環境含め認証系全般、Active Directory、PKI、NW、パフォーマンス、WU 関連等)

(サイバー セキュリティ関連)
2019 年 ~ 2023 年 自動運転車いすシステム開発および、安全とセキュリティに関する研究
2022 年 ~ 2023 年 ミラノ工科大学にてサイバー セキュリティ教育および、脆弱性について研究
2022 年 ~　　サイバー セキュリティ教育教材の開発、ワークショップ等での講師
2024 年 ~　　IPA産業サイバーセキュリティセンター 補助員
2025 年　　中部DX推進人材育成プラットフォーム　サイバーセキュリティ講座での講師

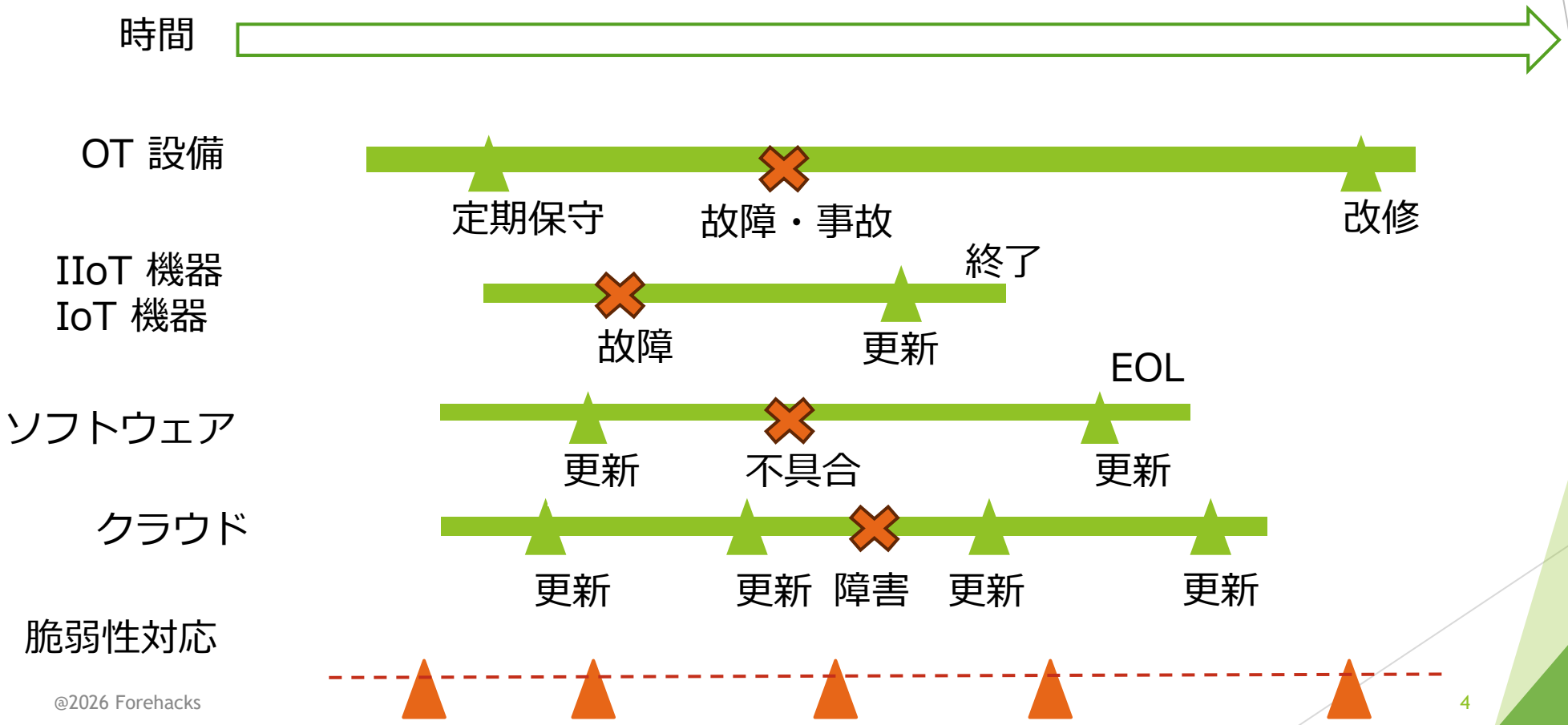
Linkedin
URL:www.linkedin.com/in/taito-sasaki-398a15223/

DFD（構造） × 生成AI（知能）

DFDで「構造」を整理し
Safety / Security / SBOM を
「同じ台帳」に束ね
人が根拠を持って判断する

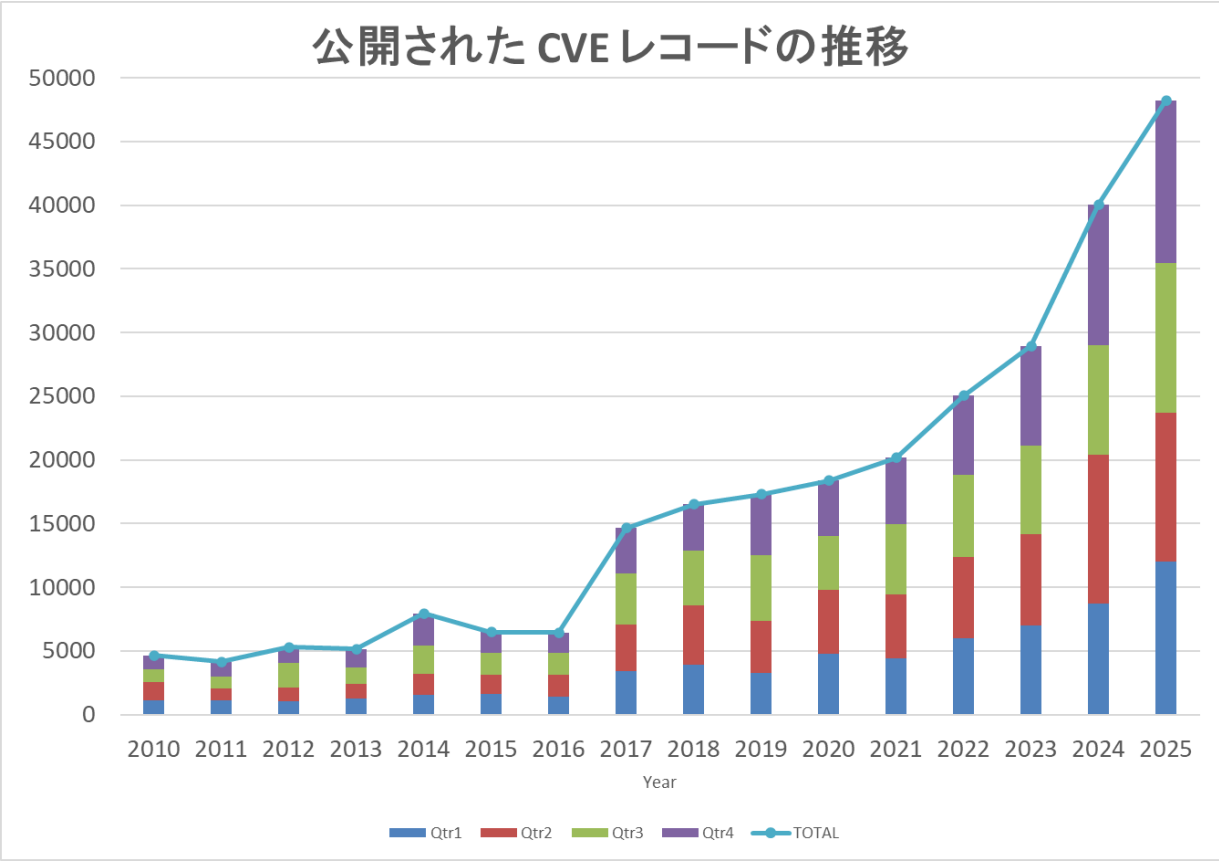
多様化する制御システム
― 予測不能なライフサイクル変動 ―

同一システム内で変化が常時・非同期に発生
再評価トリガーとなりうる



脆弱性登録件数の推移

サイバー攻撃者は、**脆弱性**を狙って、攻撃してくる。
攻撃者の発明品



https://www.cve.org/About/Metrics

CVE の脆弱性データベースの登録数は、年々増加し、2022 年 11 月の生成 AI の登場後ペースが加速。2025年では年間 48244件

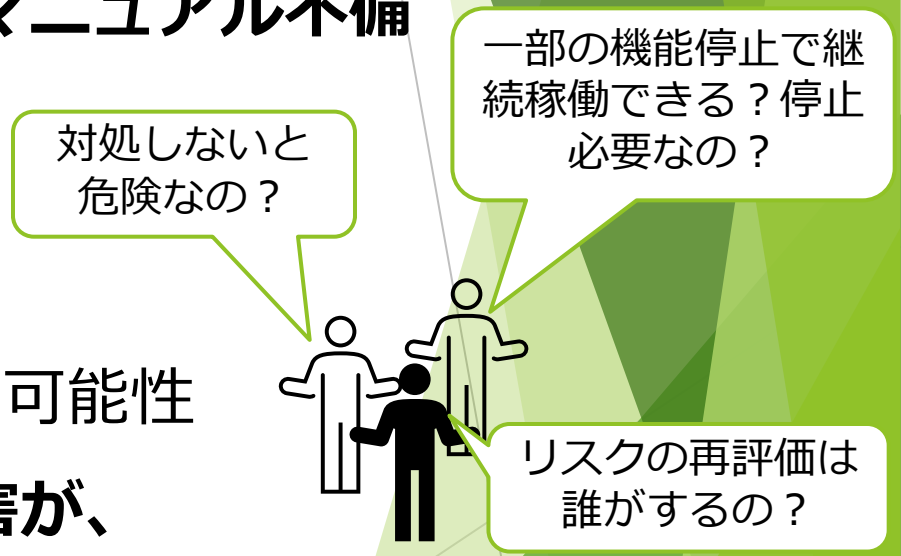
脆弱性管理の難しさ

セキュア開発はシステムライフサイクルにおいて管理が必要

管理対象

セーフティ → **故障、設備不全、マニュアル不備**

セキュリティ → **脆弱性**



膨大な脆弱性発生はセーフティの破綻をもたらす可能性

制御システムでは **サイバー攻撃による情報被害が、人命・環境への影響につながる**

セーフティとセキュリティは切り離せない

セーフティとセキュリティの衝突

システム開発において、セーフティとセキュリティを開発初期から検討した方がリリース後の脆弱性対応も考慮しやすくなる

対象とするシステムは同じでも、扱うモデルも解析手法・対策も異なる

緊急停止ボタン

セーフティのための緊急停止機能が、セキュリティ対策なしに実装されていれば、それは攻撃者にとっての格好の攻撃手段 (DoS)

フェールセーフ動作

異常を検知したらすぐ停止して安全確保！
でも悪意を持った人間が意図的にその異常を偽装するケースも

リリース後の致命的な混乱

システムを強制終了（セーフティ動作）させると、物理的な事故が起きる可能性
逆に動かし続ければ被害が拡大する。現場の判断基準は？？

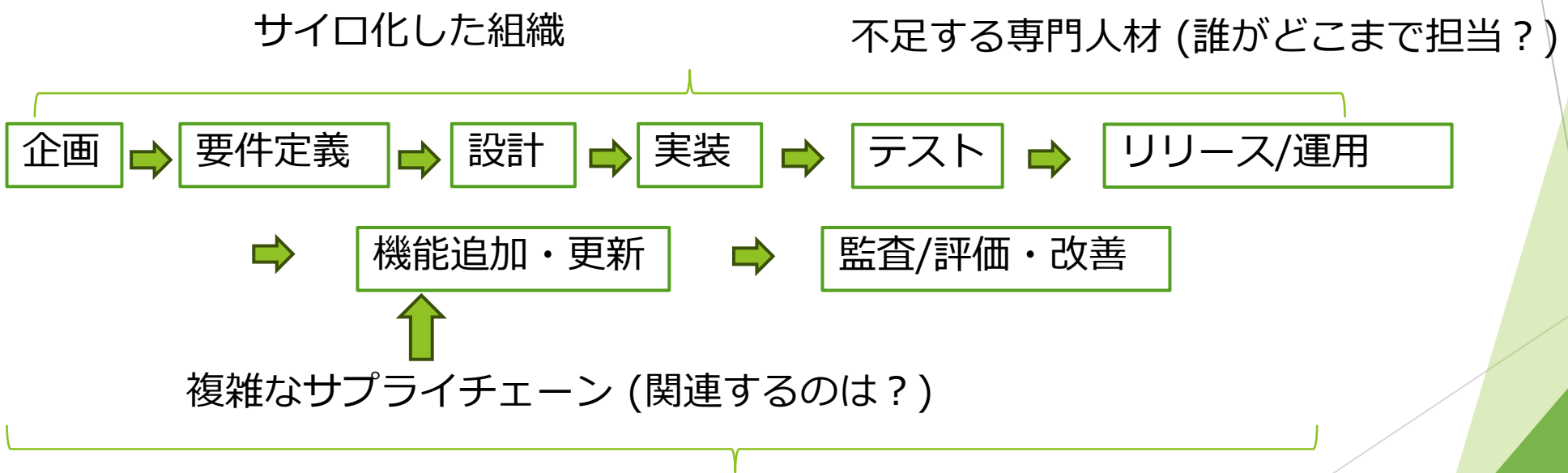
『同じシステムモデル』の上で、お互いの対策をぶつけ合わせる場が必要ではないですか？

複雑化するトレーサビリティ

システムライフサイクル全体で整合性を維持するのは困難

- 機能更新・仕様変更 → 過去設計や影響範囲の確認が大変
- 脆弱性対応 → 修正履歴や影響の把握が困難
- 運用改善 → 設定変更の安全性・セキュリティ評価が必須
- 監査対応 → 誰が何をしたかの証跡追跡

サイロ化した組織、複雑なサプライチェーン、不足する専門人材が負荷を増幅



EUサイバーレジリエンス法 (EU Cyber Resilience Act：CRA)

2022年提案 2024年施行　脆弱性・インシデント報告義務2026年9月11日からその他2027年12月11日から罰則適用

（対象）デバイスやネットワークに直接的/間接的に接続されるデジタル要素を備えたすべての製品

（DCS, ロボットなどは重要な対象で、ほとんどのソフトウェアが関係する）

ただし、医療関係の機器や民間航空機、自動車など他のEUセキュリティ規制の対象となっているものや安全保障関係などは対象外。

（要件）

1. **製品の計画、設計、開発、製造、配送、保守のすべてのフェーズ**において、リスクアセスメントを実施する。
   法が定めるセキュリティ特性要件を順守して設計、開発、製造を行う。
2. **適合性評価を行う。重要なものに対しては、第三者認証が必要：CEマーク**
3. すべてのサイバーセキュリティリスクについて文書化する。
4. **メーカー／開発者**は、悪用された脆弱性およびインシデントが発生した場合、**24時間以内に**関連する国のサイバーセキュリティ当局、欧州ネットワーク・情報セキュリティ機関（ENISA）に**報告**しなければならない。
5. 製品販売後、メーカー／開発者は、サポート期間中は脆弱性に対して効果的に対処する。
6. 製品の使用について明確かつわかりやすい説明書を用意する。
7. 製品が使われると想定される期間中（最低5年間）、ユーザーにセキュリティアップデートを提供する。

（罰則）

【重大な違反*】1,500万ユーロ(約23.5億円)またはグローバルの売り上げの2.5%の高い方を上限

【その他違反】1,000万ユーロまたはグローバルの売り上げの2%の高い方を上限**

【情報提供義務違反*】500万ユーロまたはグローバルの売り上げの1%の高い方を上限**

*（必須サイバーセキュリティ要件の不遵守など）　**（適合性評価の義務の不遵守など）　***（当局への情報提供拒否など）

制御システムにおける管理の破綻

1. 設計の断絶（Safety × Security）

合意形成の困難:

同じシステムなのに、解析モデルも対策もバラバラで、設計段階で合意が難しい。

トレードオフの露呈:

リリース直前や運用中に ”安全だが守れない(Security欠如)”、“守れるが危険(Safety欠如)”
という矛盾が発覚し、手戻りコストが爆発する。

2. 情報の断絶（時間の経過とトレーサビリティ）

設計意図の喪失::

長い製品ライフサイクルの中で「なぜこの設計にしたか」の意図が繋がっておらず、
変更時の整合性が維持できない。

影響予測の不能:

ドキュメントの欠落や複雑なサプライチェーンにより、
脆弱性パッチ一つ当てる際の影響範囲が誰にも予測できない。

3. 責任の断絶（組織のサイロ化）

リソースの枯渇:

専門人材が不足する中、開発・運用・セキュリティの組織が分断され、情報共有だけで手一杯

判断の停滞:

ベンダー連携や責任所在が不明確なまま、
未対応の脆弱性が「解決できない技術負債」として積み上がり続ける。

DFDを共通モデルとして「構造」を整理し
Safety / Security / SBOM を
「同じ台帳」に束ね
人が根拠を持って判断する

共通モデルとしてのDFD

機能ベースモデルはセーフティ、セキュリティ双方になじみがある

セキュリティ解析：
　DFDを用いた機能ごとの脅威分析

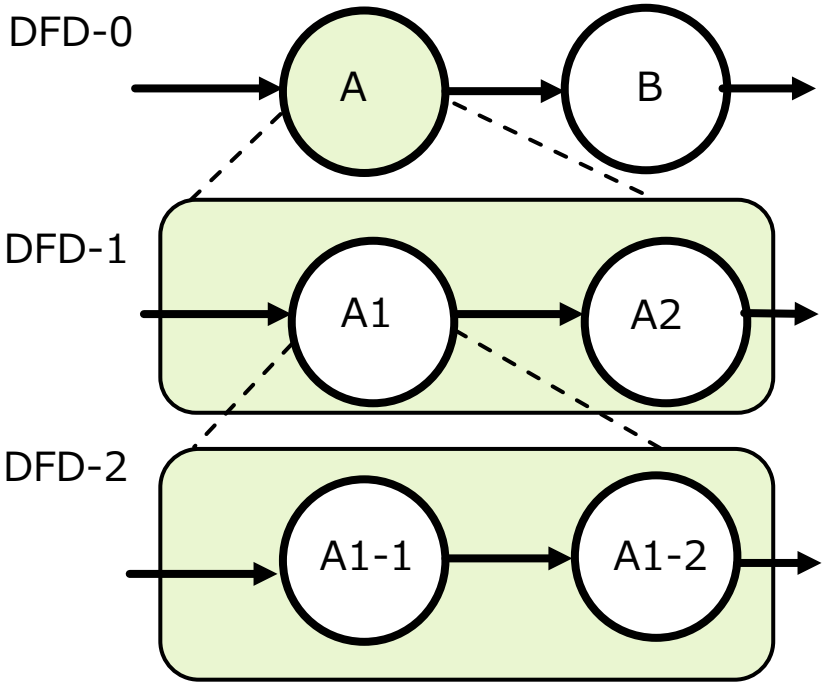
セーフティ解析：
　FMEAなど、機能ベースで解析

ライフサイクル全体で機能ベースモデル (DFD) を共通モデルとして扱い、
同時に検討していく方がお互いの考えを理解していくうえでハードルが低い。

セキュリティの破綻がセーフティの破綻につながるという観点で
セーフティとセキュリティを一緒に議論していく

システムの構造モデル DFD (Data Flow Diagram) (1)

ライフサイクル管理には、開発から保守まで、共通したモデルを扱いたい。

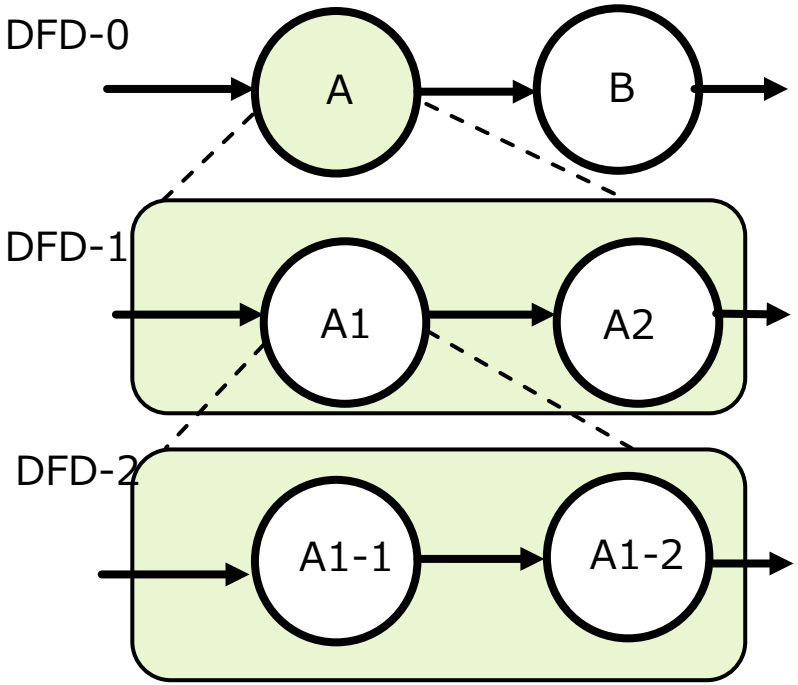


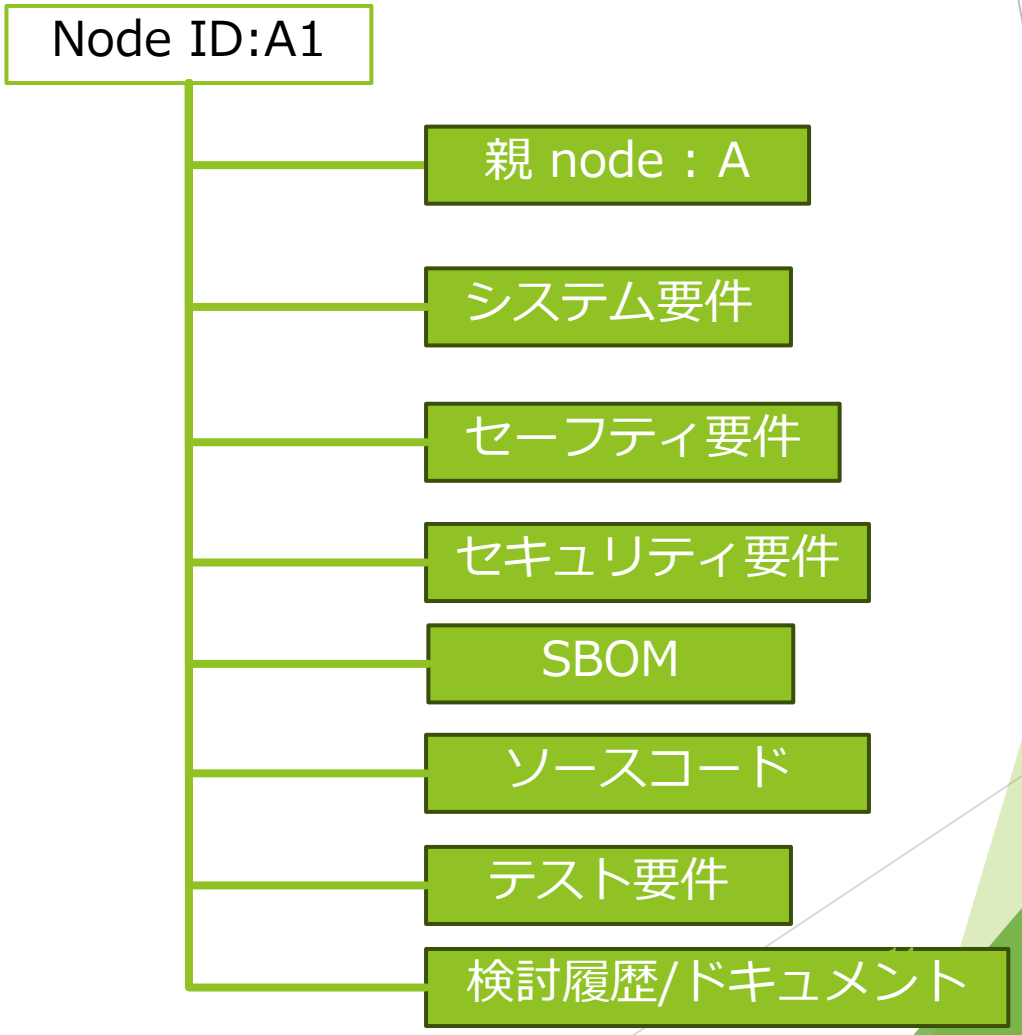
Symbol	Description	Symbol	Description
○	Process	◎	Multiplexed Function
→	Data Flow	⇒	Multiplexed Pass
▭	External Entity	▭	Multiplexed External Entity
⌭	Data Store	💬	Coments
▭	Module	◂····▸	Contact

セーフティ、セキュリティのどちらでも、
異常検知、隔離、代替、停止をモジュールの関係性から
両者が同じ目線で検討できる構造のモデルは有効なはず。

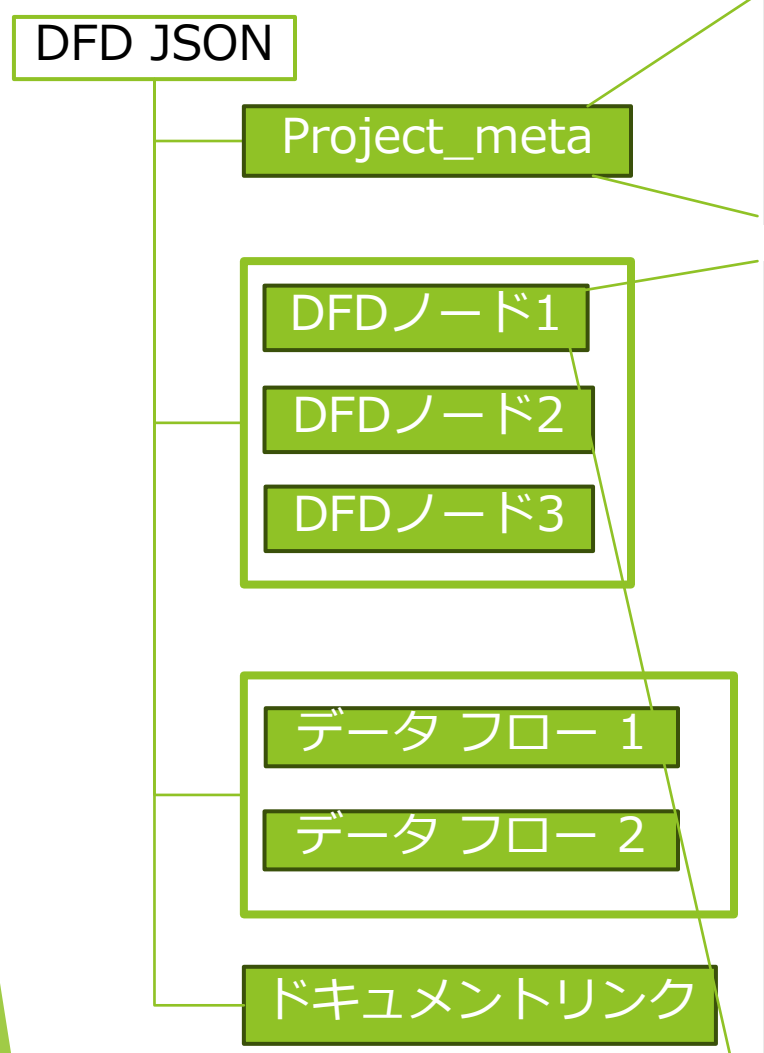
システムの構造モデル DFD (2)

DFD は単なる図ではなく、ライフサイクル全体の索引
Node ID ・ Data Flow ID ごとに属性を付与





DFD JSONで台帳化して管理

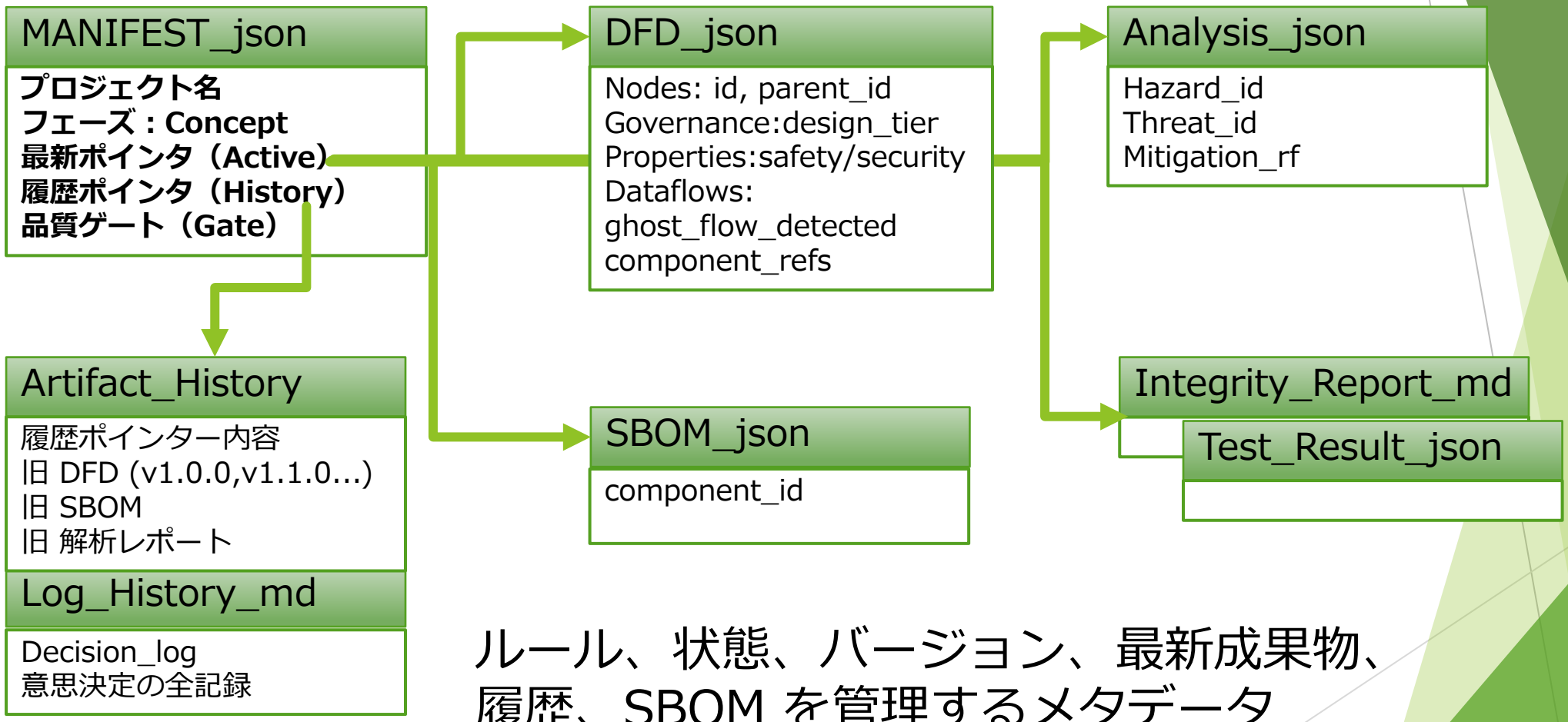



```
"project_meta": {
    "system_name": "SL-DSS_Autonomous_Wheelchair",
    "version": "1.0.0",
}
```

```
{
  "id": "DEV_WHEEL_BASE",      // 索引主キー：すべての成果物を紐付けるID
  “parent_id”: “SYS_WHEELCHAIR_SYSTEM”,    // 親ノードに紐付けるID
                                                    //属性の継承
  "governance": { ... },              //   責務モデルについて定義
                                                    //【組織】誰が・どこから・どの責任で
  "properties": {
    “safety”:   { ... },        //【安全】安全目標と具体的な安全機構
    “security”: { ... },        //【セキュリティ】セキュリティ目標と具体的な要件
    "system":   { ... }         //  【システム】CPU/電力/遅延の限界値
  },

  "links": {                    //  【追跡】マニュアル/決定理由/SBOMへの直リンク
    "decision_log": "LOG_HISTORY.md#DEC_WHEEL_BASE",
    "component_refs": ["SBOM_ID_001"]
  }
}
```

MANIFESTファイルによるプロジェクト管理

成果物間のデジタル・スレッド：MANIFEST を核とした相互参照



ルール、状態、バージョン、最新成果物、履歴、SBOM を管理するメタデータ

生成 AI を活用したエコシステム

DFDを「システムの索引」へ：情報の断絶を解消

システムを node_id ごとに分解し、全要素（要件・リスク・コード・部品）をIDに集約

マニフェストによる整合性:

プロジェクト全体の状況、規格準拠、意思決定の履歴をマニフェストに集約

Safety & Security 解析・対策立案支援：

DFD JSON に対し、指定した条件、規格（ISO 26262/21434等）をベースに解析
対策間のトレードオフを検知・可視化

論理的な差分検証：

実装（コード）から DFD を逆生成し、設計図と照合

エビデンス自動生成支援:

AI がDFD JSON に紐づくデータから、監査証跡や外注先へ仕様書、テスト仕様書等のドキュメント生成

SBOM 連携 & 脆弱性管理:

ID に SBOM 情報を紐づけ、最新の CVE 情報をもとにした脆弱性調査支援

生成 AI をライフサイクルに組み込み、
人間は「AIの出力結果」をベースに意思決定を下す。

自動運転車いすの開発例 –動画-

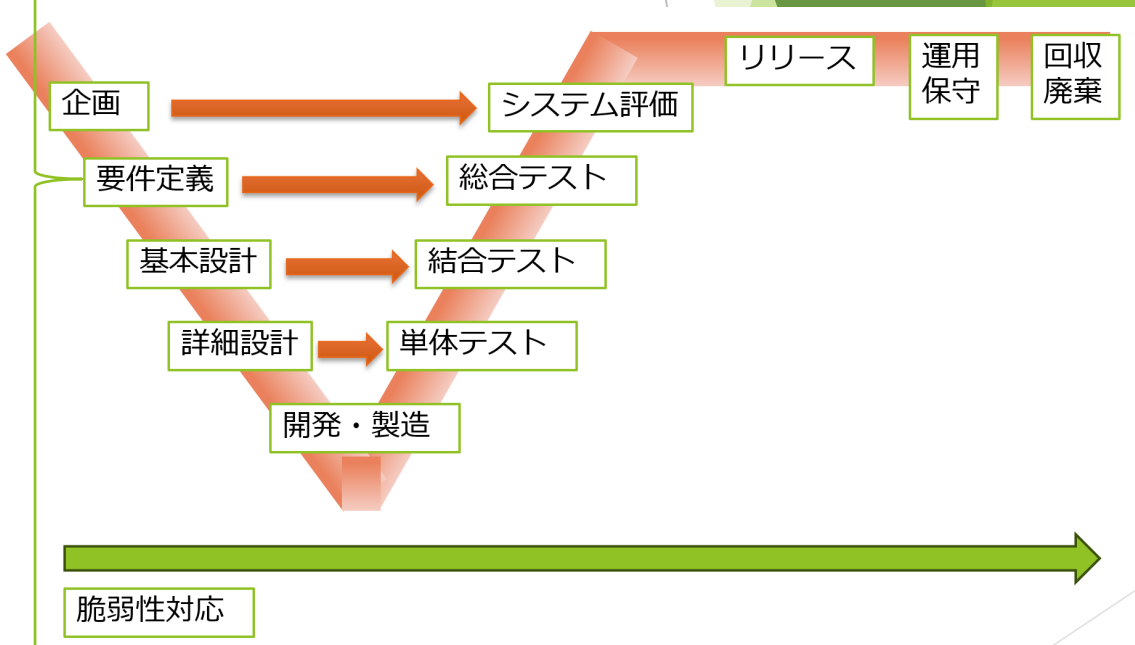
自動運転車いす開発におけるシステムライフサイクルでの管理

想定利用者：歩行に難がある方
行動範囲：**ROS を用いて自動運転・大学館外呼び出し・自動返却等を実現**
移動先指定：タブレット操作により指定する
制約：施設の玄関には段差があり、そこに車いすで落ちると大けがの恐れがある。人とぶつからないように、**避けて通るか、停止して、避けてもらうように依頼**すること。

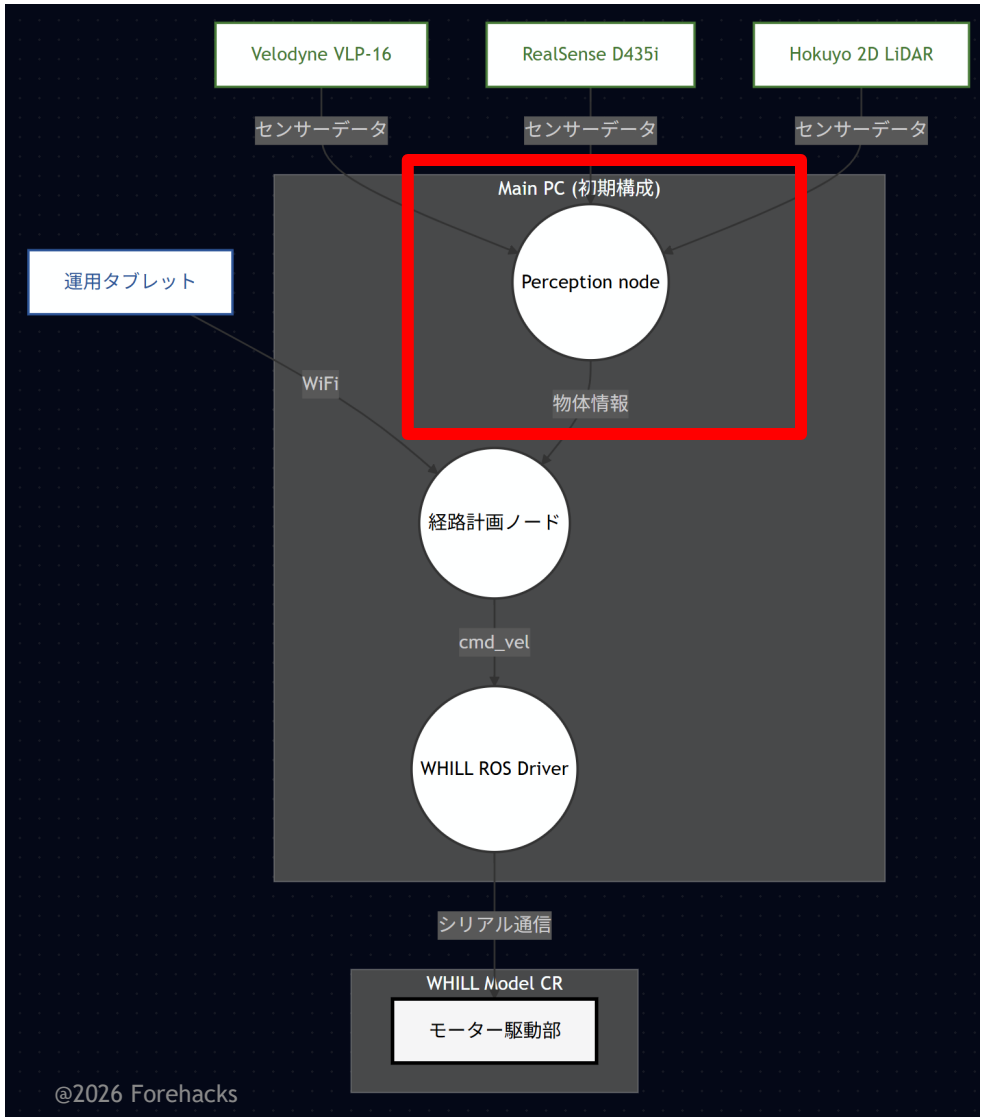
■ **外注解析依頼パッケージ (Phase 1.3, 1.4, 1.5)**
対象プロジェクト: SL-DSS_Autonomous_Wheelchair (v0.2.0)
依頼先: SCC, Lead Safety Expert, Lead Security Expert
1. 解析対象コンテキスト
システム境界: 規約 1.1 に基づく。
•**主要コンポーネント**:
- DEV_LIDAR_3D: 段差（玄関）および歩行者の検知。
- SW_NAV_STACK: ROSベースの自律走行制御。
- SW_SAFETY_GATE: 独立した衝突・落下防止制約ロジック。
- EXT_TABLET: ネットワーク経由の目的地指令。

•物理的制約: 大学玄関の段差（致命的落下リスク）、狭い廊下。

2. 専門家へのミッション {#SCOPE_1_5}
•**SCC (1.3)**: ROSパッケージおよびLiDARデバイス等、構想を実現するための暫定SBOMの妥当性確認。
•**Lead Safety Expert (1.4)**: HARA（ハザード解析）の実施。
•**Lead Security Expert (1.5)**: **TARA（脅威分析）**の実施。



設計段階でのセーフティ解析 & セキュリティ解析



1. セーフティ解析：故障への防衛 (Phase 2.4)
物理センサー（VLP/RS/Hokuyo）を選定した直後、それぞれの「壊れ方」を解析
•リスク（故障モード）：
- RealSense: 直射日光（西日）による白飛び、または夜間の光量不足で障害物を見失う。
- Velodyne: センサー表面への泥跳ねや雨滴による、偽の障害物検知。
- 共通: ケーブル断線やプロセスハングアップによる、データ更新の停止。

•対策 (Safety Mechanism):
- **SM_005 (多数決照合): 1つのセンサーが「異常なし」と言っても、他が「異常あり」と言えば安全側に倒す。**
- **SM_006 (生存信号): 各センサーの死活監視を100ms周期で行う。**

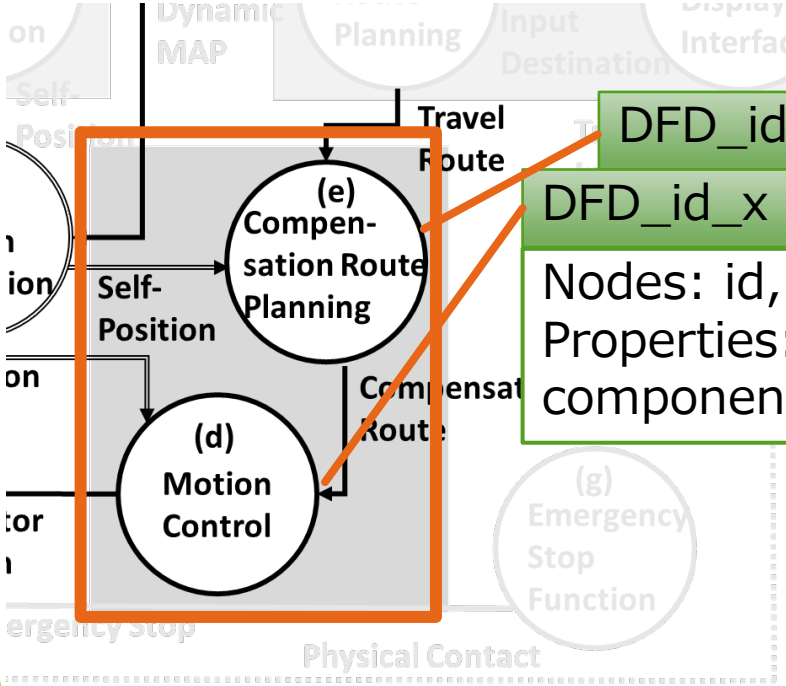
2. セキュリティ解析：悪意への防衛 (Phase 2.5)
同じセンサー構成に対し「悪意ある第三者」がどう介入するか
・リスク（脅威）：
センサー・スプーフィング: レーザー照射等により、存在しない壁を見せて急ブレーキを誘発する、あるいは本物の壁を「消去」して衝突させる。
物理ジャック: センサーの配線に偽造デバイスを割り込ませ、PCに偽のデータを送り込む。
・対策 (Security Requirement):
SR_007 (異種照合検問): 原理の異なるセンサー（光と画像）で同じ場所を監視し、片方だけに現れる「不自然な情報」を攻撃として弾く。
SR_005 (物理封印): 物理ジャックを防ぐため、コネクタ部に改ざん防止封印を施す。

要求の源泉	対策の名称	具体的役割
Safety	冗長性の確保	1系統が「故障」しても、残りの2系統で安全に停止・運行を継続する。
Security	妥当性の検証	1系統が「ハック」されても、物理原理の異なる他系統との不一致により攻撃を検知する。
Merge	3点フュージョン	VLP-16 × RealSense × Hokuyo の3系統を突き合わせ、信頼度を算出する。

DFD スライシングによる責任境界の明確化

DFD（JSON）モデルはすべての情報の Single Source of Truth (SSOT)

- 属性の継承（DNA）：
  parent_id を介して、最上位 DFD で決めた安全目標が階層化された DFD モデルの子ノードまで継承される。



DFD_id_y

DFD_id_x

Nodes: id, parent_id
Properties:safety/security
component_refs

Dataflow_id_z

Dataflow_id_y

Dataflow_id_x

source
Threat
Protocol
Pyhysical_connection
Properties:safety/security

Item_id

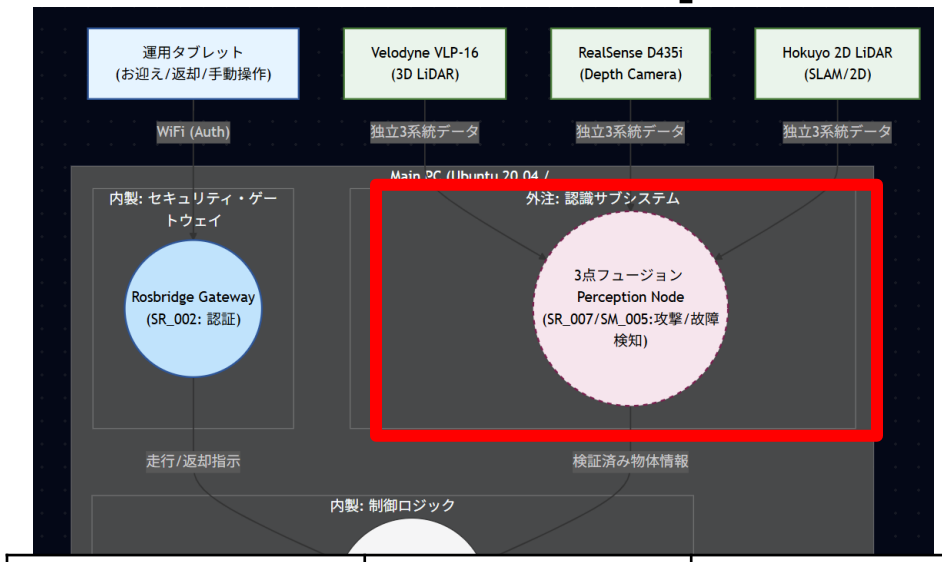
Safety/Security

SBOM_component_x

component_id

T.Sasaki, 17 June 2022

外部委託: Perception Node を DFD スライシング



Topic Name	Message Type	内容
/perception/obstacle_map	nav_msgs/OccupancyGrid	周囲の障害物コストマップ
/perception/emergency_stop	std_msgs/Bool	直近の衝突回避フラグ（高優先）
/perception/diag	diagnostic_msgs/DiagnosticArray	センサーおよびノードの健康状態

1. プロジェクト基本要件 {#SPEC_GENERAL}
開発環境: Ubuntu 20.04 LTS / ROS Noetic 厳守

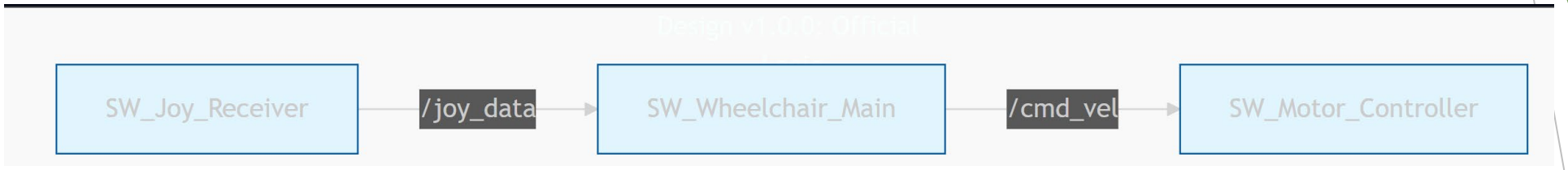
3. セーフティ要件 (Safety Goals: SM_005) {#SPEC_SAFETY}
Sensor Fusion Logic: Velodyne と RealSense のデータを空間的に同期させ、一方が障害物を検知し、他方が検知しない場合の「不一致アラート」を実装すること。

4. セキュリティ要件 (Security Requirements: SR_007)
{#SPEC_SECURITY}
Anti-Spoofing: 物理的にあり得ない速度での物体の出現・消失、またはLiDAR 点群の異常な高輝度反応を検知し、フィルタリングすること。
論理隔離: 外注ノード内から外部ネットワーク（インターネット）への一切の通信を禁止する。ROS 以外のポート開放は不許可。
監査対応(SBOM): 納品時にソースコードの脆弱性スキャン結果、およびSBOM（使用ライブラリ一覧）を提出すること。

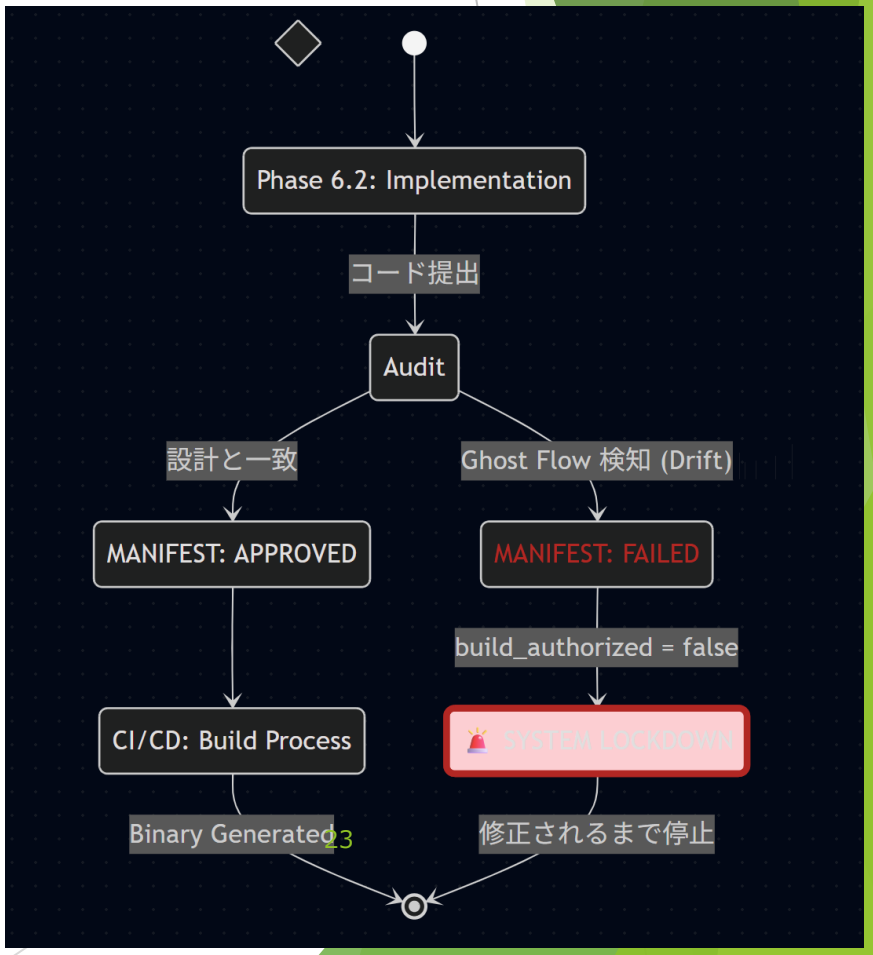
納品・監査要件
プログラム本体
SBOM (Software Bill of Materials) JSON形式
範囲: 全ての直接・間接依存ライブラリ、OSパッケージを含む。
脆弱性スキャン結果レポート
CVSS v3 Score 7.0 (High) 以上の脆弱性がゼロであること。
ソースコード脆弱性スキャン結果
　SASTツールによる静的解析結果の提出。

実装・テスト段階

外部接続した Joystick コントローラーで運転車いすを操縦する監査対象のソースコードを解析し、DFD (JSON) を抽出し、設計 JSON との差分を生成 AI によって評価。

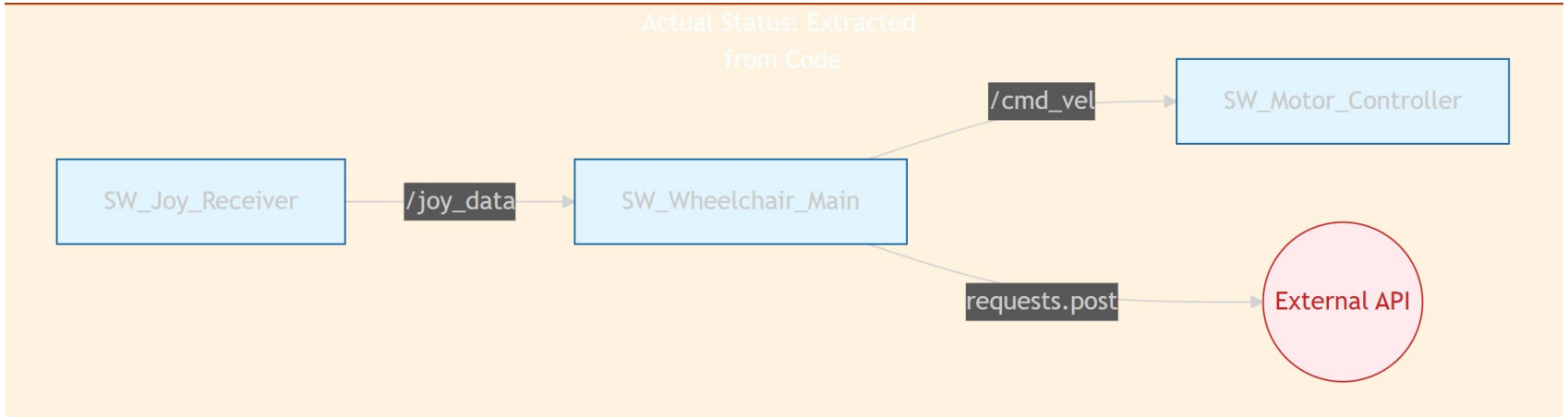



```
# SW_Wheelchair_Main.py
from geometry_msgs.msg import Twist
import requests # <--- !! 監査対象：設計にない外部ライブラリのインポート

class WheelchairNode(rclpy.node.Node):
    def __init__(self):
        super().__init__('wheelchair_main')
        self.subscription = self.create_subscription(Joy, '/joy_data', self.joy_callback, 10)
        self.publisher = self.create_publisher(Twist, '/cmd_vel', 10)

    def joy_callback(self, msg):
        twist = Twist()
        # 正常な設計パス：ジョイスティック入力から速度変換
        twist.linear.x = msg.axes[1]
        twist.angular.z = msg.axes[0]
        self.publisher.publish(twist)

        # GHOST FLOW：設計にないテレメトリ送信
        # 開発者が「デバッグ用」として、勝手に外部サーバへジョイスティック情報を送信
        try:
            requests.post("http://debug-telemetry.local/log", json={"pos": msg.axes})
        except:
            pass
```



実装・テスト段階

逆解析による監査コードからDFDを抽出し、設計JSONとの差分を評価。



監査レポート：GHOST_FLOW_DETECTED

判定: **REJECT** (インポート拒否)

検知内容:
SW_Wheelchair_Main ノードにおいて、設計図 v1.3.0 に**存在しない外部接続先 http://debug-telemetry.local/log** へのデータ流出パスを特定しました。

セキュリティリスク:
このパスは、悪意ある第三者によってジョイスティックの操作をリアルタイムに傍受（スニッフィング）される、あるいはシステムの応答遅延（DoS）を引き起こす脆弱性になり得ます。

是正命令:
requests ライブラリの使用を削除し、設計図 4.8 に戻って再ビルドせよ。

リリース後の SBOM を利用した脆弱性の確認

運用段階の脆弱性調査

・特定のノードに影響がある CVE が登録されているか？

・特定の CVE に影響するコンポーネントが
SBOM 上に存在するか？

コンポーネント	バージョン	出典
OS Kernel	5.4.0-x	Ubuntu 20.04
OpenCV	4.2.0	APT (libopencv-dev)
ROS	Noetic	OSRF
Auth-GW	1.0.2	内製 (Go)
librealsense2	2.50.0	Intel

脆弱性が存在する場合：
リスクを再評価し、
なぜその意思決定をしたのかを ID に紐付け
判断根拠を残す。

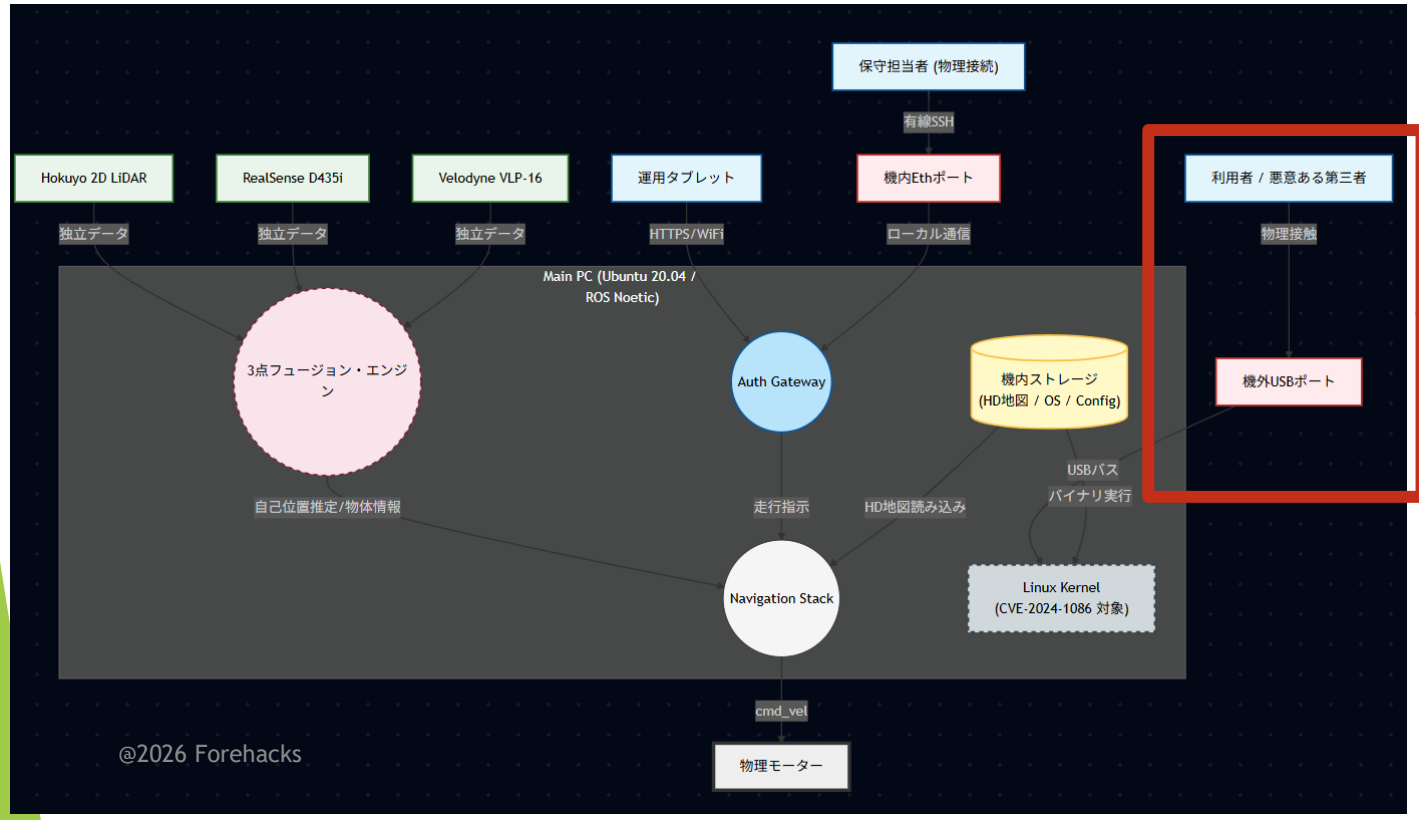
・誰が担当する責任があるのか？
・対処が必要か？
・リスクを受容できるのか？

Linux Kernel / Glibc
Ubuntu 20.04（LTS）
CVE-2024-1086:
Netfilter（ファイアウォール）サブシステムにおける
Use-after-free の脆弱性。
リスク:
ローカルの一般ユーザー権限（外注ノードの権限）から、
一気に Root権限を奪取される恐れがあります。
システム全体の支配権を奪われる致命的な脆弱性です

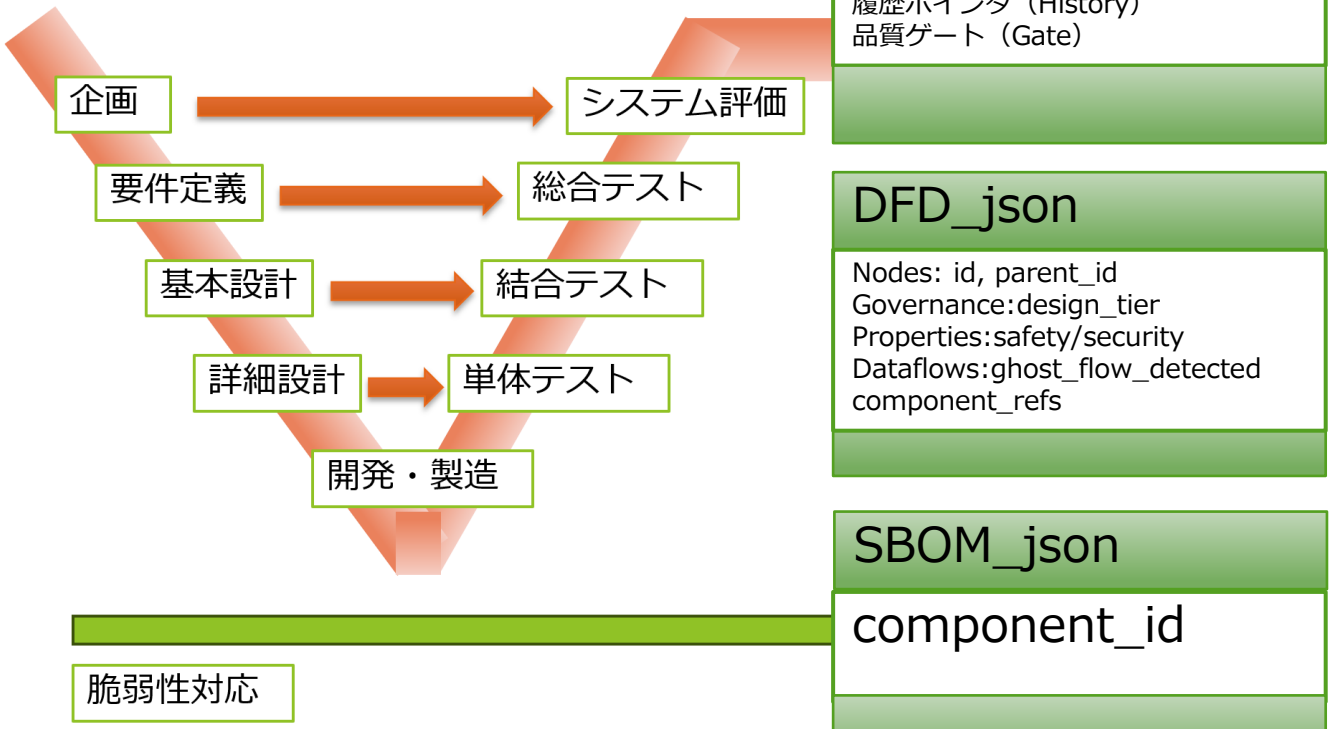
リスク評価による意思決定

本システム特有の問題
車いすは誰でも物理接触可能 USB/LAN 直挿し = ローカル侵入が容易
Kernel 脆弱性 → 即 root 権限奪取

モーター制御停止
センサー偽装暴走/誤走行 → 人身事故に直結（Safety破綻）



ライフサイクルにわたる「構造」の整理
-開発手法を選ばない-



MANIFEST_json

プロジェクト名
フェーズ：Concept
最新ポインタ（Active）
履歴ポインタ（History）
品質ゲート（Gate）

DFD_json

Nodes: id, parent_id
Governance:design_tier
Properties:safety/security
Dataflows:ghost_flow_detected
component_refs

SBOM_json

component_id

Images generated by Gemini

DFD を共通モデルとして「構造」を整理し
Safety / Security / SBOM を
「同じ台帳」に束ね
人が根拠を持って判断する