制動系製品の
リバースフィーチャモデリングの試み

古一 克也[1]　中西 恒夫[2]　大串 和弘[1]　山田 智司[1]

[1]株式会社アドヴィックス
[2]福岡大学工学部電子情報工学科

2024年5月24日

目次

0．会社紹介

社名	株式会社アドヴィックス
設立	2001年7月3日
事業内容	自動車用ブレーキシステムおよびそのシステムを構成する部品の開発・生産・販売
資本金	122億円（2023年3月31日現在）
売上高	連結　7,518億円 単独　4,001億円（いずれも2022年4月1日～2023年3月31日）
社長	秋山 晃
従業員	連結 13,293名 単独 4,613名（いずれも2023年3月31日現在）
主要	EPB、ESC、ABS、ブレーキブースタ、マスターシリンダ、ディスクブレーキ、ドラムブレーキ、その他関連部品

ブレーキシステム全体を手掛ける 世界で唯一のサプライヤーとして誕生

目次

1．プロダクトライン導入の背景　と　現場の問題

背景：組み込みソフトウェア開発は、ベースからの差分開発が主流である。
　　展開や新規機能の増加に伴い、ソフトウェア仕様のバリエーションが増加し、ソフト構造が複雑化している。

問題：現状のやり方だと、仕様・ソフトウェアの変更や管理が容易ではなく、開発工数が増加してしまう。

対策：上記解決に向け、
　　“共通性と可変性を分離し，ソフトウェア資産の再利用性向上を図る“プロダクトライン開発の導入を検討

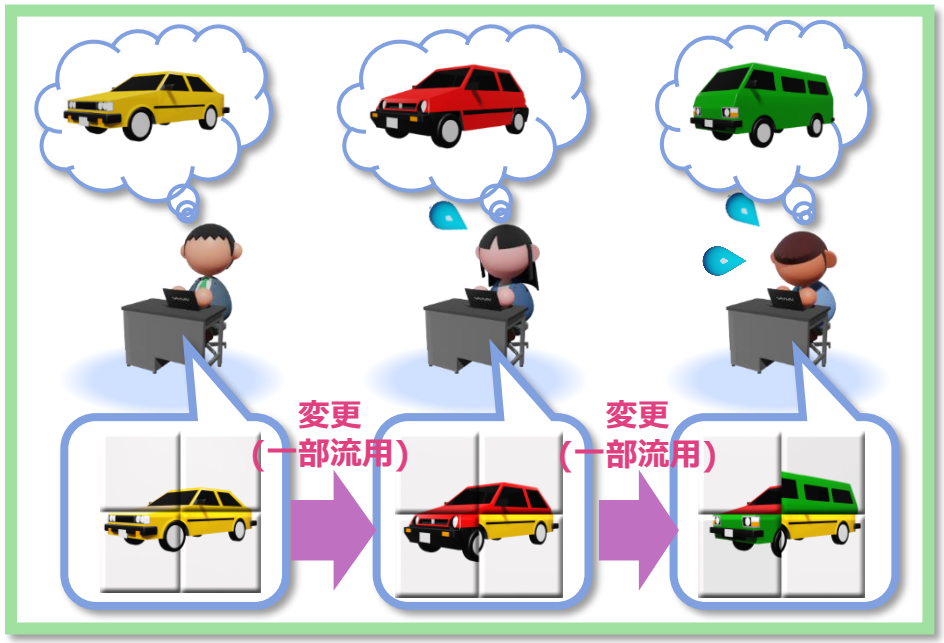


図1.差分開発のイメージ

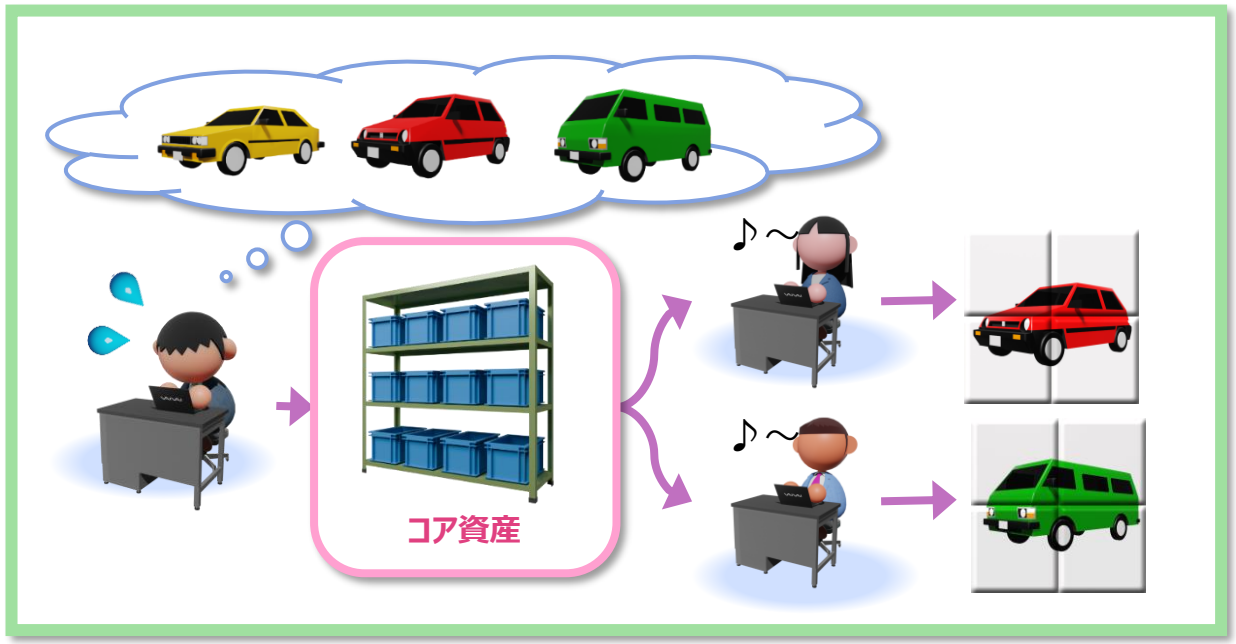
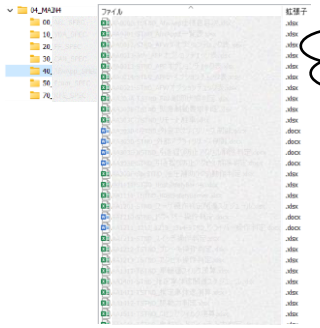


図2.プロダクトライン開発のイメージ

1．プロダクトライン導入の背景　と　現場の問題

本質的な問題点

製品仕様の全体像が把握できない



仕様変更

機能配置が適切にできない
（どこを変えるのが良いかを判断できない）

レビュー

指摘が多く、手戻り発生

フィーチャモデリングによって

製品仕様の全体像が把握できる

仕様変更

機能配置の目途付けができる
（どこを変えれば良いかが分かる）

レビュー

関係者間のコミュニケーションが
円滑になる

フィーチャモデリングによって、現場の『本質的な問題点』も併せて解決したい

目次

2．フィーチャモデリングの目的　と　アンチパターン

フィーチャモデリングの目的

- 製品の共通部と可変部の分離を見える化（記述）する。
- フィーチャは、製品の特徴を特定できる構成要素として捉える。
- アーキテクチャを予備的かつ探索的に構築する。

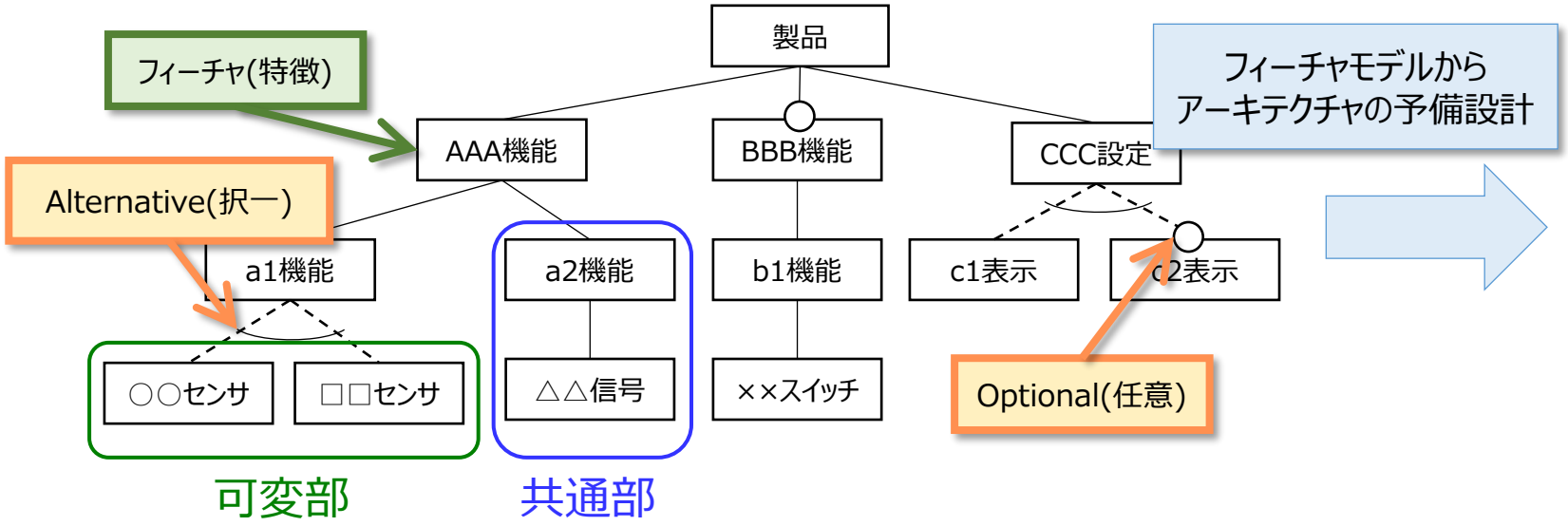


図3.フィーチャモデル例

フィーチャモデルから
アーキテクチャの予備設計

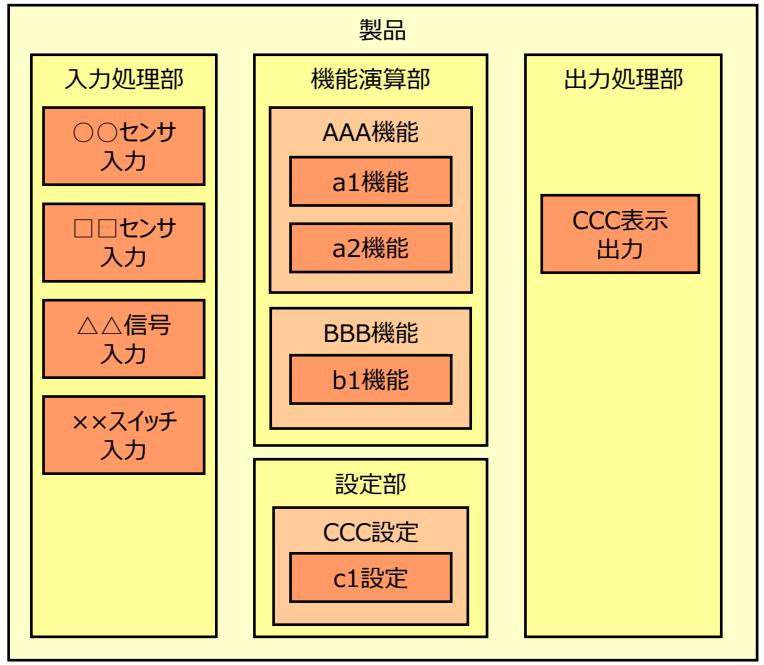


図4.アーキテクチャ例

プロダクトライン開発において、フィーチャモデリングが重要！！
そのためには、既存の開発資産からのリバースフィーチャモデリングが必要である。

2．フィーチャモデリングの目的　と　アンチパターン

フィーチャモデリングのアンチパターン

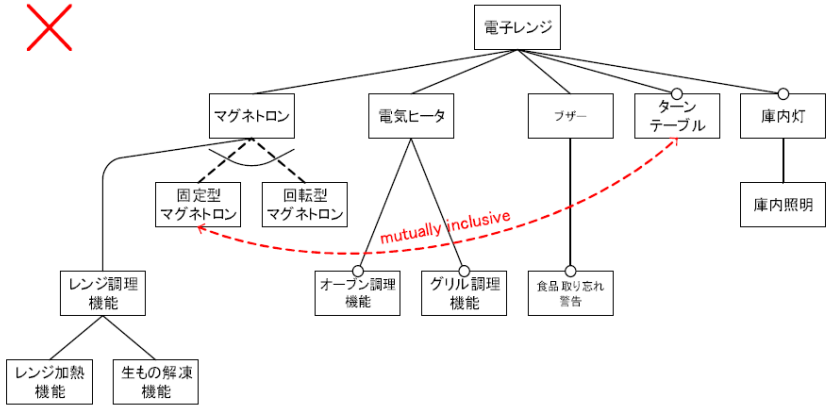
フィーチャモデリングのアンチパターン

アーキテクチャの計画をする目的でフィーチャモデリングをする場合のアンチパターン

- 構造上の相違のみを表現する。
- 目的と手段を逆転する。（抽象度の逆転）
- ちがいのちがいを表現しない。（抽象化の不足）
- ちがいの分解が足りない。（関心事分離の不足）
- 製品のちがいと実行時の振舞いのちがいを混同する。
- 概念の上下関係がマッチしていない。
- 関数呼出構造を表現する。

目的と手段を逆転する

サービスと実現手段のフィーチャの上下が逆転している例



- サービス，すなわち機能要求のほうが実現手段よりも抽象度が高い。
- 各フィーチャが何のためにあるのかを考え，上に来るべき，より抽象度の高いサービスフィーチャを見つけるとよい。
- 各フィーチャをどうやって実現するのかを考え，下に来るべき，より抽象度の低いサービスフィーチャを見つけるとよい。

アンチパターンに陥っていないか確認し、フィーチャモデリングすることが重要

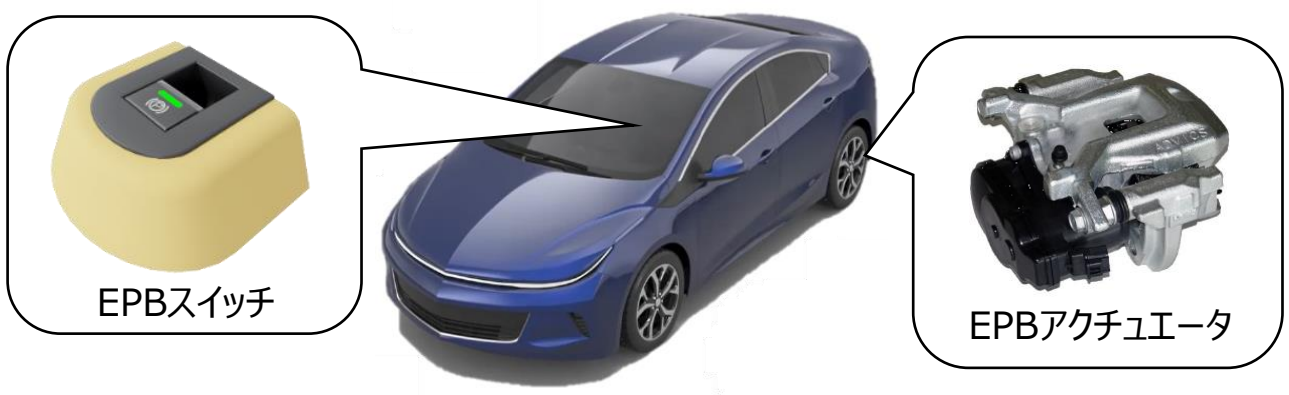
参考文献：中西，久住，福田，「ソフトウェアアーキテクチャ事前設計を目的とするフィーチャモデリングのアンチパターン」，組込みシステムシンポジウム論文集，2009年10月．

目次

3．ドキュメントとコードからマインドマップ生成

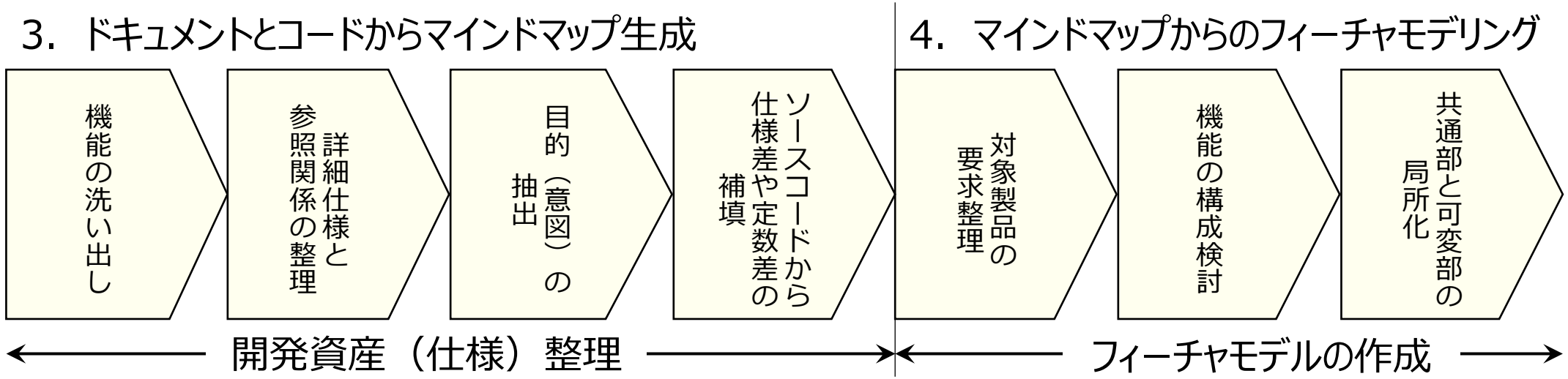
制動系製品のリバースフィーチャモデリングの試み

対象製品：電動パーキングブレーキ（**EPB**：**E**lectromechanical **P**arking **B**rake）



EPBとは、スイッチ等の操作により
電気モーター(EPBアクチュエータ)で
パーキングブレーキをかける装置のこと。

次項より、既存開発資産からフィーチャモデル構築（リバースフィーチャモデリング）の手順を紹介



3．ドキュメントとコードからマインドマップ生成

1）機能の抽出

- 製品が持つ機能を抽出する。

例）電動パーキングブレーキの場合、

「車両停止時にパーキングブレーキを掛ける」、「車両発進時にパーキングブレーキを解除する」 という機能

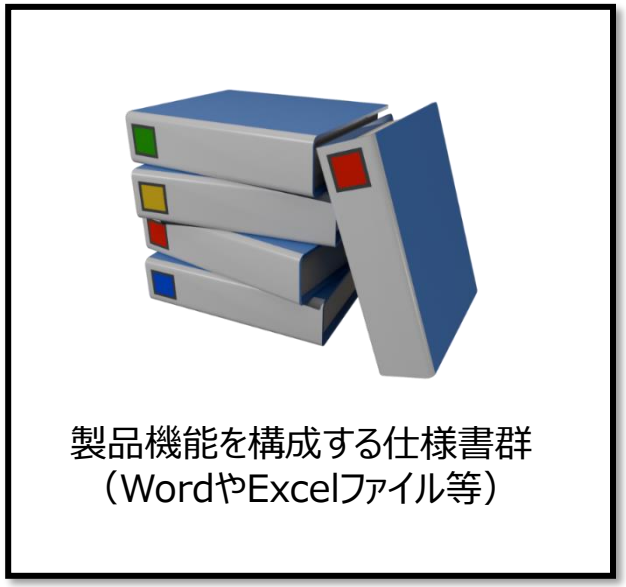


図5.既存の開発資産

抽出

抽出

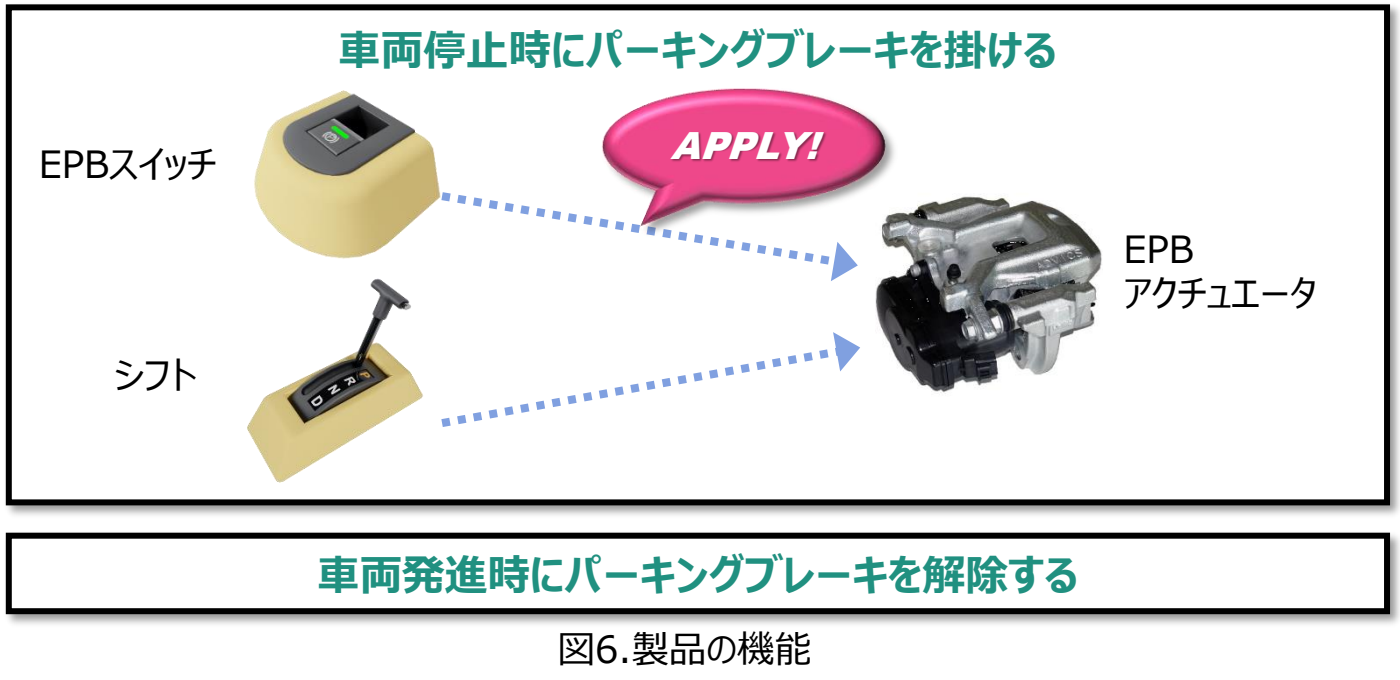


図6.製品の機能

リバースフィーチャモデリング対象のシステムが持つ機能を全て抽出

3．ドキュメントとコードからマインドマップ生成

2）詳細仕様と参照関係の整理

- 機能を構築する仕様を末端まで全て書き記す。
- 複数の機能で同じ仕様や条件が使われている場合、それに目印をつける。

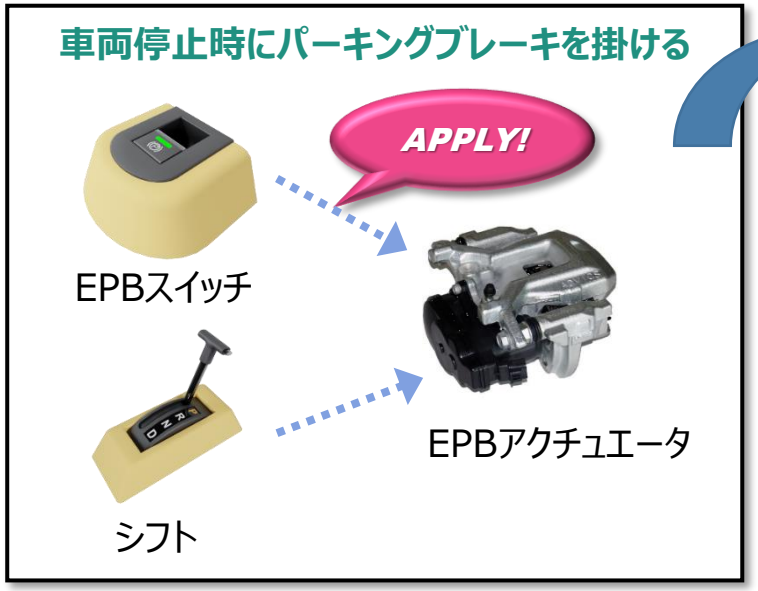


図7.製品の機能_車両停止時にパーキングブレーキを掛ける

マインドマップ化（ツール：FreeMind）

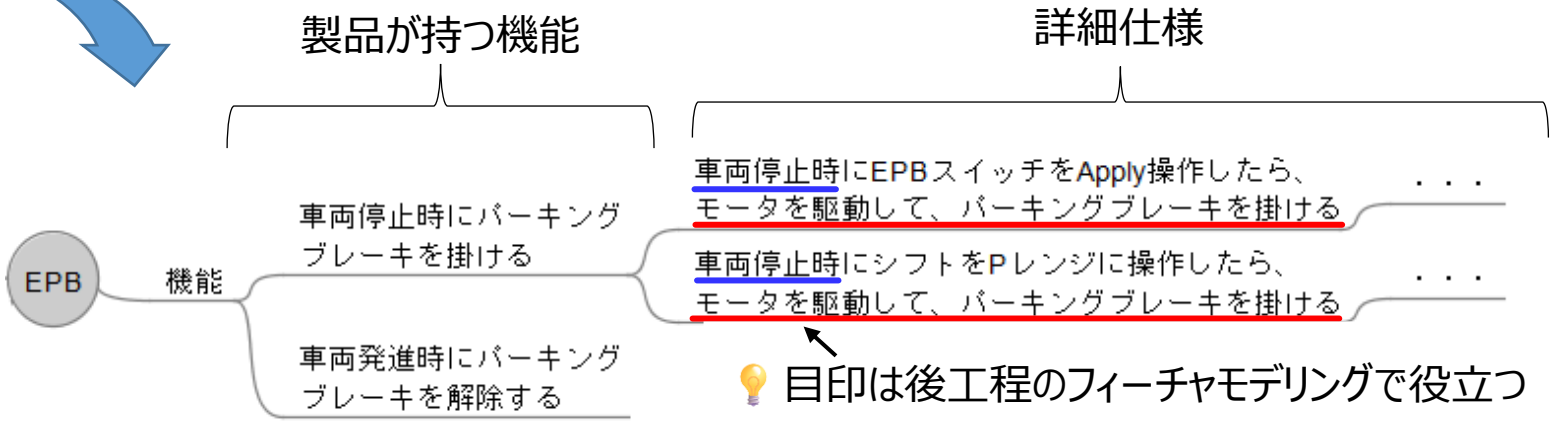


図8.製品機能のマインドマップ例

詳細仕様をマインドマップに起こすのは、地道な作業です。
しかし、時間をかけて丁寧に書き記すことが後々効果がでるため、重要！
（USDM化を見越して「～をして、～をして、～する」の形式で記載。）

システムが持つ機能と構成している仕様や情報を１つ１つ確実に書き記す

3．ドキュメントとコードからマインドマップ生成

3）目的の抽出

- 各機能や仕様の背後にある目的（何のため？）を明確にする。

例えば、「車両停止時にEPBスイッチをApply操作したら、モータを駆動して、パーキングブレーキを掛ける」機能の目的は、非力な方でも確実にパーキングブレーキを掛ける事ができることである。

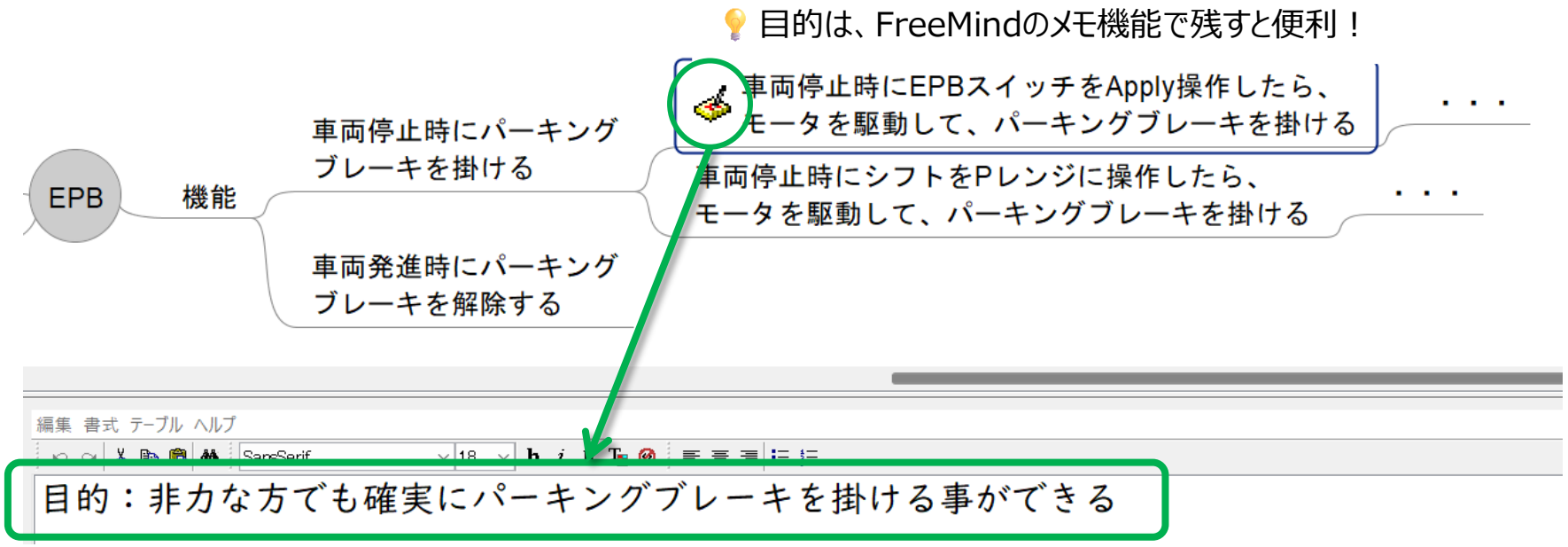


図8.製品機能のマインドマップ例_目的の明確化

各仕様の目的を明確にし、残しておく
⇒フィーチャモデリングでの類似仕様のグルーピング（中間概念の導出 P.17）に役立つ

3．ドキュメントとコードからマインドマップ生成

4）ソースコードから仕様差や定数差の補填

- 仕様は、メーカや車種などの違いによって異なることがある。

  ⇒コーディングツール（テキストエディタ）の検索機能を使って、違い＝可変部を見つけ出す。

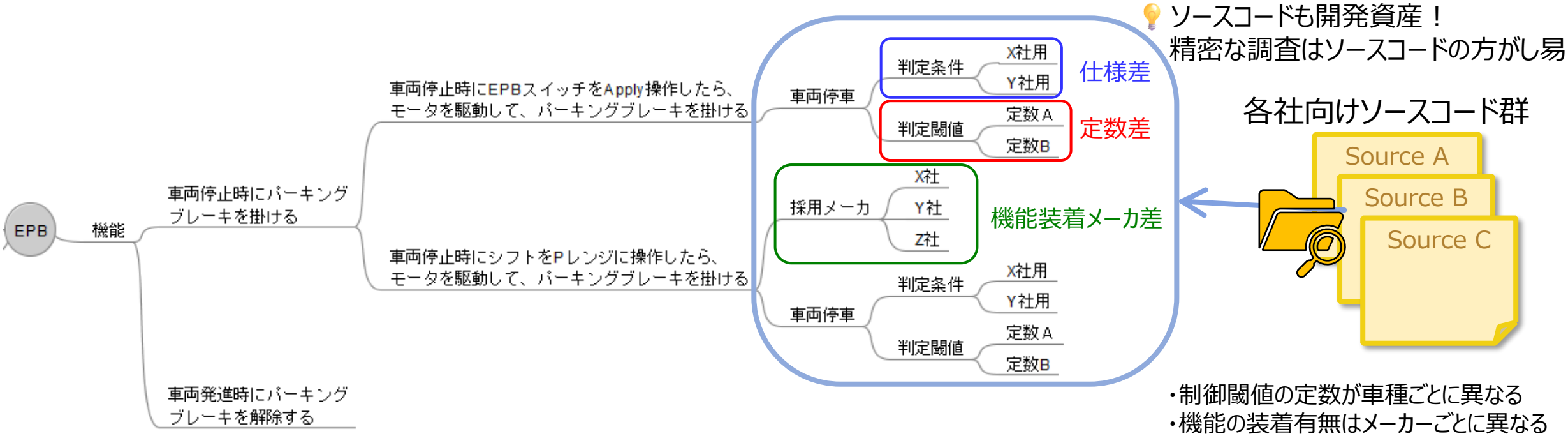


図9.製品機能のマインドマップ例_違いの抽出

確かな情報を記載するため、ドキュメントだけでなく、ソースコードも使って違いを抽出する

４．マインドマップからのフィーチャモデリング

１）対象製品の要求整理

- 製品機能の再定義 ：ユーザーが期待する機能の**最上位要求**をユースケース分析で定義する。
- 機能のシナリオ定義 ：ユーザーが行う具体的な操作やシナリオを検討する。

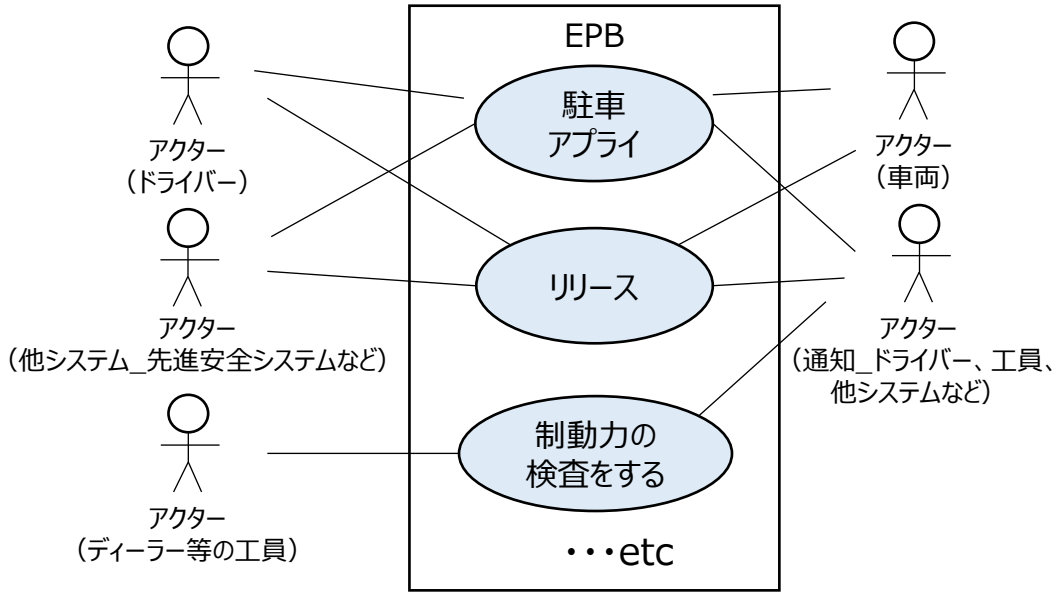


図10.EPBのユースケース図

ID	REL01		
名称	リリース		
目的	ID	PA01	
	名称	駐車アプライ	
アクタ	目的	駐車/停車中にEPBを掛けたい要求により、アクチュエータを作動させ、駐車/停車保持する。また、ユーザーにEPB作動状態を通知する。	
事前条件	アクタ	ドライバー	
事後条件	事前条件	車両が停車している	
イベント	事後条件	・EPBが作動し、駐車/停車保持していること ・EPBの作動完了を通知していること	
フロー	イベント	・ユーザーによるEPB作動操作 ・他システムによるEPB作動要求	
	フロー	ステップ	アクション
		1	モータをアプライ側に駆動する
		2	パーキングブレーキのアプライ完了を検知する
		3	モータの駆動を停止する
		4	パーキングブレーキのアプライ完了を通知する

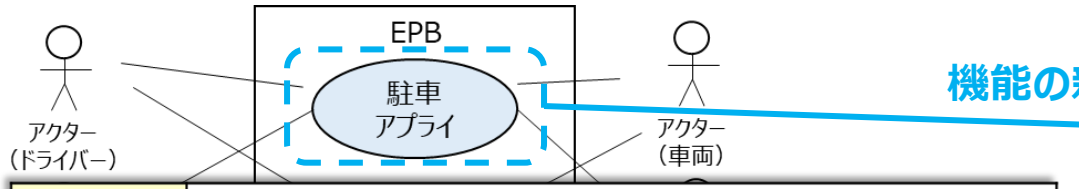
💡 再定義した機能（最上位要求）が、
既存機能を適切に配置できるかの確認もすること

原点に立ち返り、製品が求められているコト（要求）を定義する

4．マインドマップからのフィーチャモデリング

2）機能の構成検討

ユースケース分析結果から、機能フィーチャモデルの基本構成を決める



ID	PA01
名称	駐車アプライ
目的	駐車/停車中にEPBを掛けたい要求により、アクチュエータを作動させ、駐車/停車保持する。また、ユーザーにEPB作動状態を通知する。
アクタ	ドライバー
事前条件	車両が停車している
事後条件	・EPBが作動し、駐車/停車保持していること ・EPBの作動完了を通知していること
イベント	・ユーザーによるEPB作動操作 ・他システムによるEPB作動要求
フロー	ステップ / アクション
	1 / モータをアプライ側に駆動する
	2 / パーキングブレーキのアプライ完了を検知する
	3 / モータの駆動を停止する
	4 / パーキングブレーキのアプライ完了を通知する

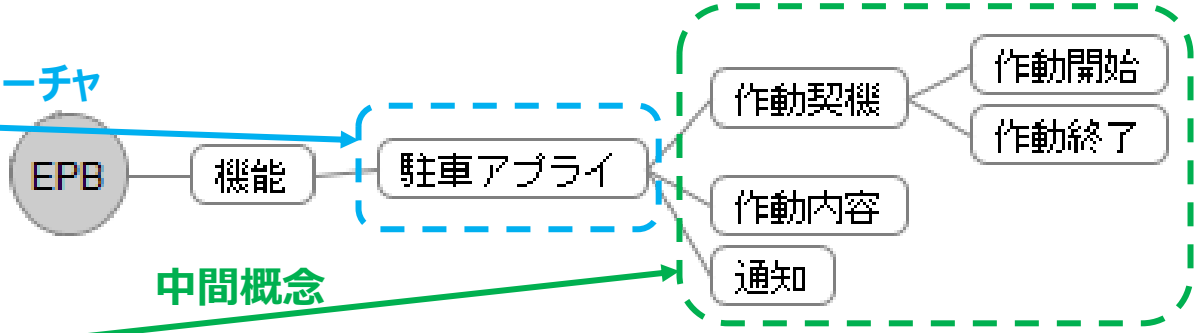


図11.EPBの機能フィーチャモデルの例

💡機能の振舞いに着目して中間概念を導出した。

例）駐車アプライ機能は、EPBスイッチ操作があった時 に、
作動契機_作動開始
EPBモータをアプライ側に駆動 して、
作動内容
パーキングブレーキ作動のアプライ完了を検知した 場合、
作動契機_作動終了
EPBモータを停止 する。また、パーキングブレーキのアプライ完了を通知 する。
作動内容 **通知**

機能のフィーチャ直下は「作動契機」「作動内容」「通知」の構成とする

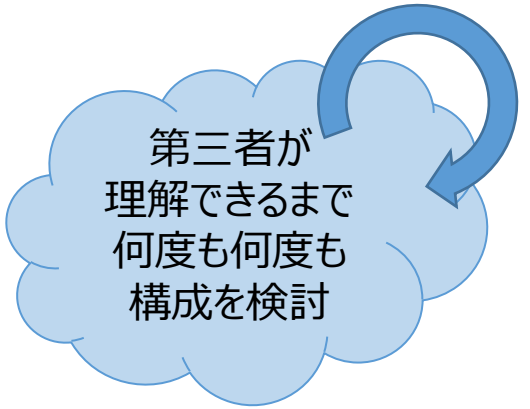
４．マインドマップからのフィーチャモデリング

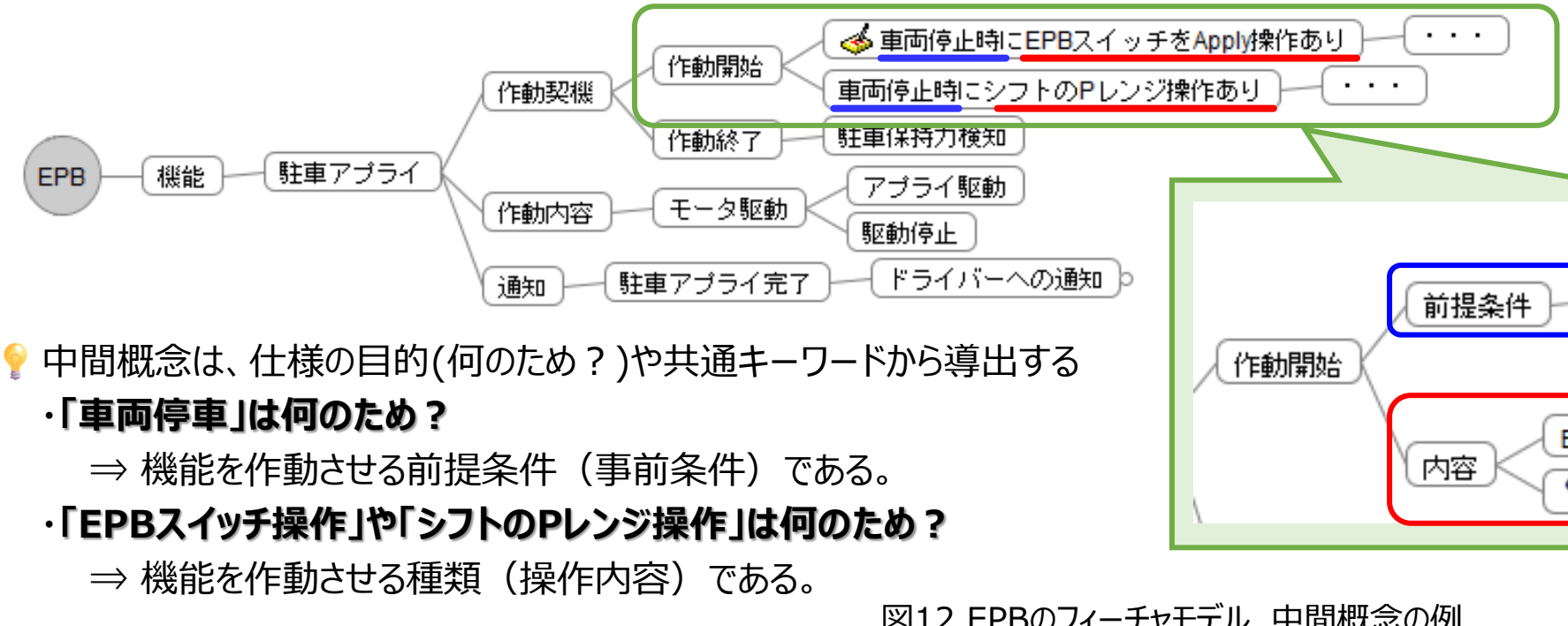
２）機能の構成検討

マインドマップからの配置：マインドマップを使って機能を配置する。

キーワード化　　：各フィーチャを要素毎にキーワード（名称）化する。

共通部分のまとめ　　：類似した目的や共通する要素を同じ中間概念でまとめる。





💡 中間概念は、仕様の目的(何のため？)や共通キーワードから導出する

- **「車両停車」は何のため？**
  ⇒ 機能を作動させる前提条件（事前条件）である。
- **「EPBスイッチ操作」や「シフトのPレンジ操作」は何のため？**
  ⇒ 機能を作動させる種類（操作内容）である。

図12.EPBのフィーチャモデル_中間概念の例

マインドマップの結果を整理し、フィーチャーモデルの基本構成を決定する

4．マインドマップからのフィーチャモデリング

3）共通部と可変部の局所化

- 差分有り仕様の局所化：可変点が多いフィーチャをまとめ、管理し易くする。

■マインドマップから配置したフィーチャモデル

機能の仕様は、
どのメーカ向けも差分は余り無い

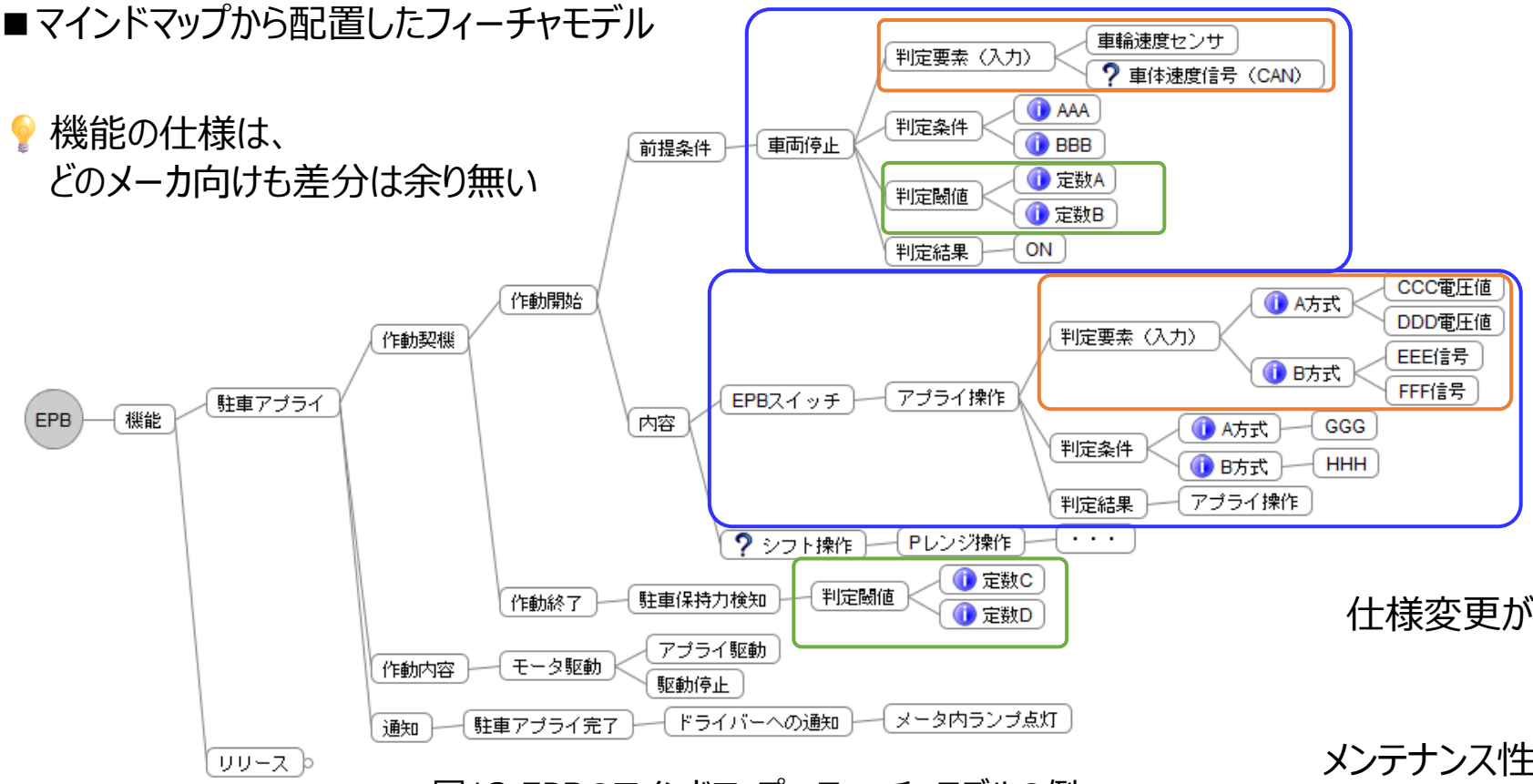


図13.EPBのマインドマップ⇒フィーチャモデルの例

可変点が多く出るのは、以下３点が主
- **入出力信号**
- **車両の状態を判定する条件**
- **判定閾値（定数）**

入出力信号
車両の状態を判定する条件 は、
駐車アプライ機能以外にも参照箇所多数あり。

仕様変更が入った際、**参照箇所全ての更新**が必要

メンテナンス性が悪くなり、**更新モレの懸念**が多いに有り

フィーチャモデルのメンテナンス性悪化の懸念課題あり、改善の検討が必要

4．マインドマップからのフィーチャモデリング

3）共通部と可変部の局所化

- 差分有り仕様の局所化：可変点が多いフィーチャをまとめ、管理し易くする。

入出力信号、**車両の状態を判定する条件**の**メンテンス性の課題**に、以下対策で解決。
対策1）専用の親フィーチャで集中管理する。（EPBの左側フィーチャ）
対策2）機能フィーチャツリーでは対策１の結果を参照とする。

入力では、
情報の種類でまとめる。
⇒ 使用目的ではまとめられない。。。

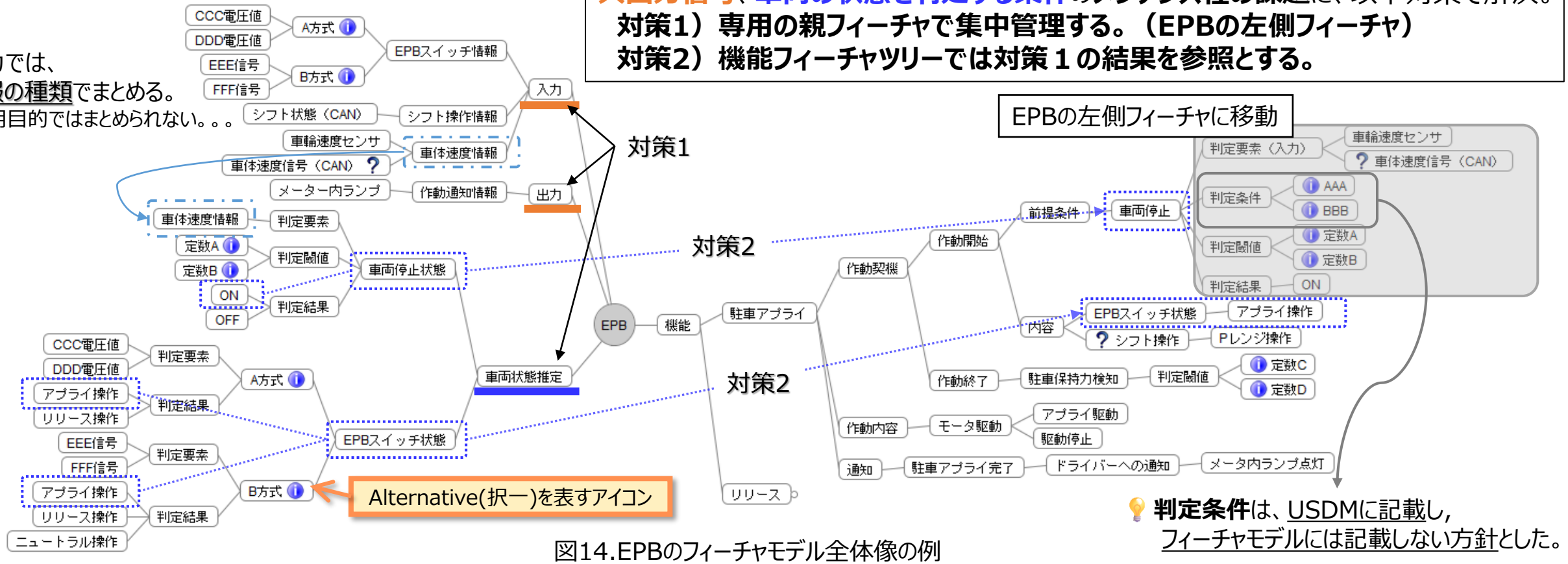


図14.EPBのフィーチャモデル全体像の例

判定条件は、USDMに記載し，
フィーチャモデルには記載しない方針とした。

自分たちが管理・分かり易い構成を考え、フィーチャーモデルを構築していく

目次

5．まとめ と 所感

まとめ　**制動系製品のリバースフィーチャモデリングの試みは成功した。**
成果として、以下のプロセスを構築できた。

アンチパターンに
陥っていない事も確認 👍

① 詳細仕様と参照関係を整理してマインドマップ生成 ･･･ P.12
製品が持つ機能と構成している仕様や情報を**１つ１つ**確実に書き記す。

② 目的の明確化と違いの抽出 ･･･ P.13、14
マインドマップを通じて、**各仕様の目的**（何のため）**の明確化** と **違い**（可変部）**の抽出**。

③ ユースケース分析による製品要求・振舞いの定義 ･･･ P.15
製品本来の**最上位要求** と **機能の振舞い** を定義する

④ 中間概念でのまとめ ･･･ P.16～19
中間概念を、**機能の振舞い** や **仕様の目的**、**類似仕様のまとめ** 等で導出し、管理・理解し易くする。

所感　形にするまではかなり苦労するが、その後が楽になる目途がたった。

・リバースフィーチャモデリングによって製品仕様のバリエーションを見える化できたことで、
仕様構成（全体像）の把握 や 仕様変更の目途付け が容易となった。

ご清聴ありがとうございました

（参考）　使用ツール FreeMind

フィーチャモデリングは、マインドマップ描画ツール（FreeMind）を使用した。

1) 操作性が高い

キーボードやマウスを使ったシンプルな操作で、フィーチャモデルの木構造を簡単に構築できる。

2) 記述や表現の柔軟性が高い

フィーチャモデルの文法的制約を受けず、アイコンや背景色などでユーザーが自由に属性を表現できる。

フィーチャモデリング中に発見した事実や未解決課題などのメモを該当箇所に配置できる。

3) 別アプリケーションでの汎用性が高い

マインドマップをXML形式で保存可能。

スクリプトによるインポート／エクスポート／機械的処理が容易。

4) 全体を俯瞰できるため、可読性が高い

限られたPC画面だったとしても大きなマインドマップを記述でき、

全体を把握しやすくできる。

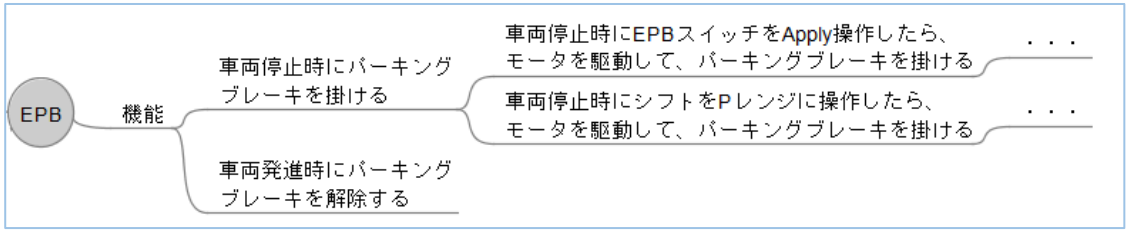


図14.製品機能のマインドマップ例

我々のチームでは、何度も作り直しが発生すると予想されたため、
操作性と柔軟性があるFreeMindでの作成を選択

参考文献：中西，山田，大串，古一，「制動系製品群のプロダクトライン開発のためのアプリケーションエンジニアリング自動化」，情処研報，Vol.2023-SE-214，No.9，2023年7月.