SWEST20@ 水明館

# ソフトウェア開発上流工程における〝抽象化〟の重要性

アイシン精機株式会社　　浅野　雅樹

asano_m@elec.aisin.co.jp

# 概要

## ■ 従来のアイシン精機でのソフトウェア開発

- ✓ ターゲット仕様に最も近い開発済み車種を、部分的に改変する派生開発
- ✓ 簡単な仕様変更でも、変更影響の把握等で開発工数が増大

## ■ アイシン精機での課題解決のアプローチ

- ✓ 過去製品の仕様をフィーチャー分析で整理し、プロダクトライン開発へ移行
  →抽象化/関心事の分離により整理

## ■ 本セッションの狙い

- ✓ 上流開発工程における“抽象化”について、アイシン精機の事例をご紹介
- ✓ 例題を用いて“抽象化”に取り組み、手法を持ち帰っていただく
- ✓ 討論形式により、参加者の組織での上流工程の課題・対応策を共有する

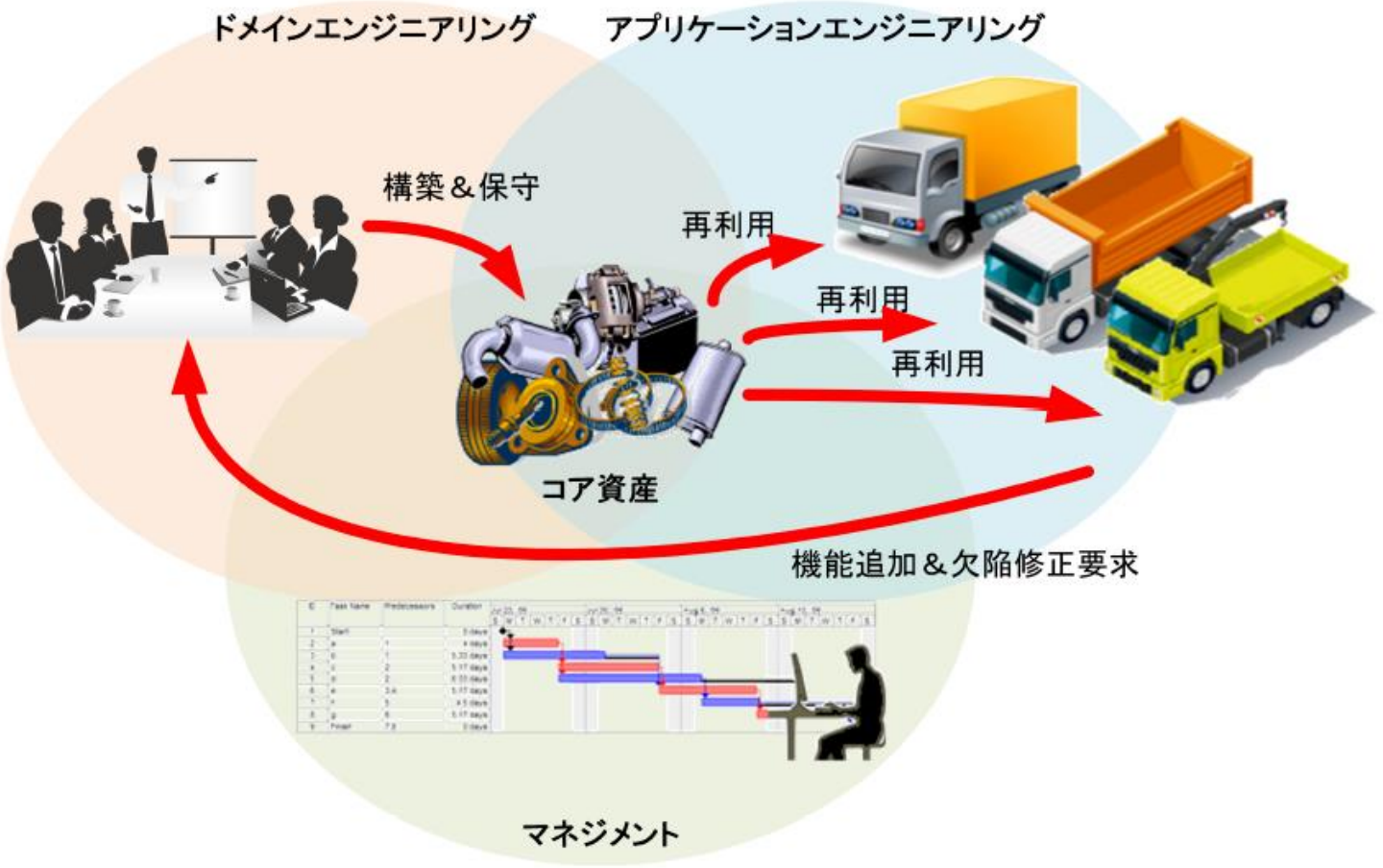
# タイムテーブル

- ✓ アイシン精機の事例をご紹介　：　15分
- ✓ 例題を用いた、グループワーク　：　20分
- ✓ グループワークの成果発表　：　15分

計50分

# プロダクトライン開発とは

**製品群での共通性と相違性を捉え、共有する部分を再利用し、
個別製品を構築する開発思想。**

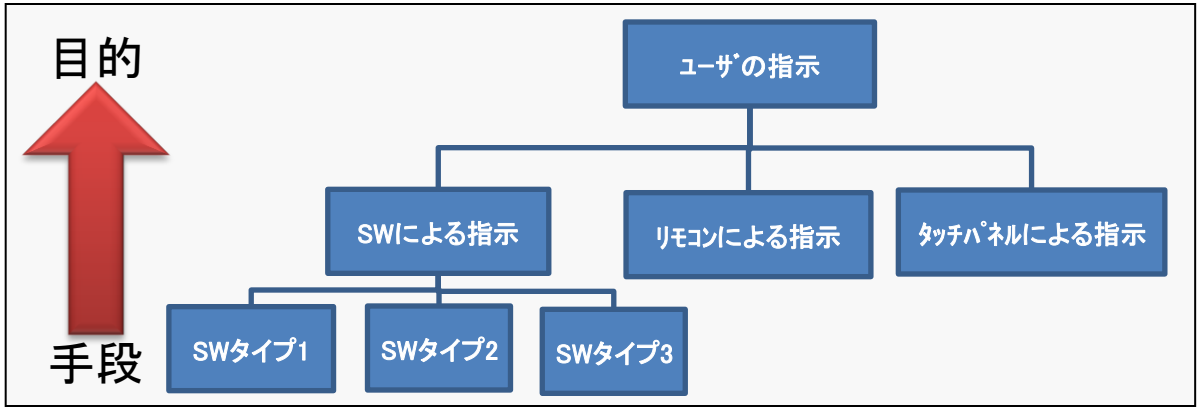



**本日は、アイシン精機での取り組み事例をご紹介させて頂きます。**

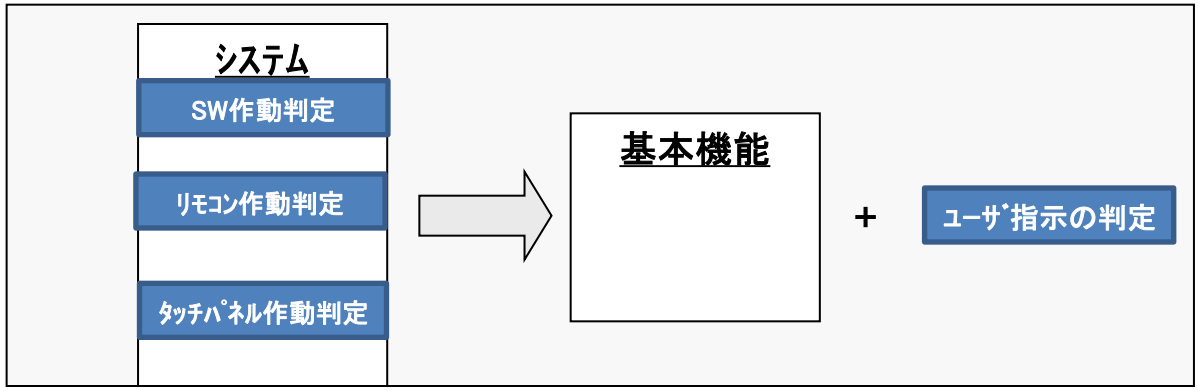
# 抽象化と関心事の分離とは

## プロダクトライン開発導入で重要な技術

- **抽象化**
  - 実現手段の違いではなく、目的にさかのぼって纏める技術




- **関心事の分離**
  - システム仕様の中で横断的に出現する共通的な機能[関心事]を、抽出して独立させる技術



# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

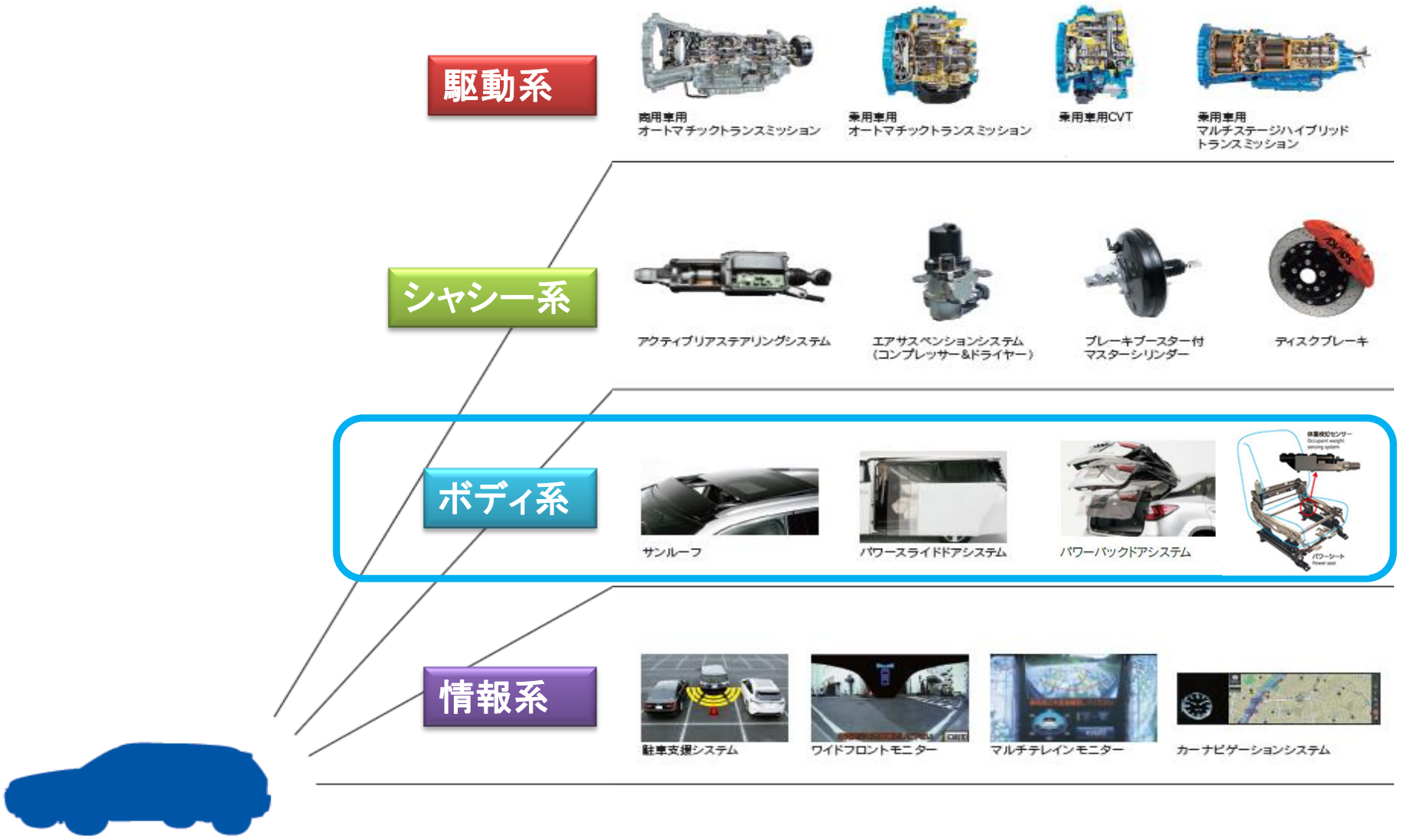
6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

# アイシン精機の主要製品と活動の対象範囲

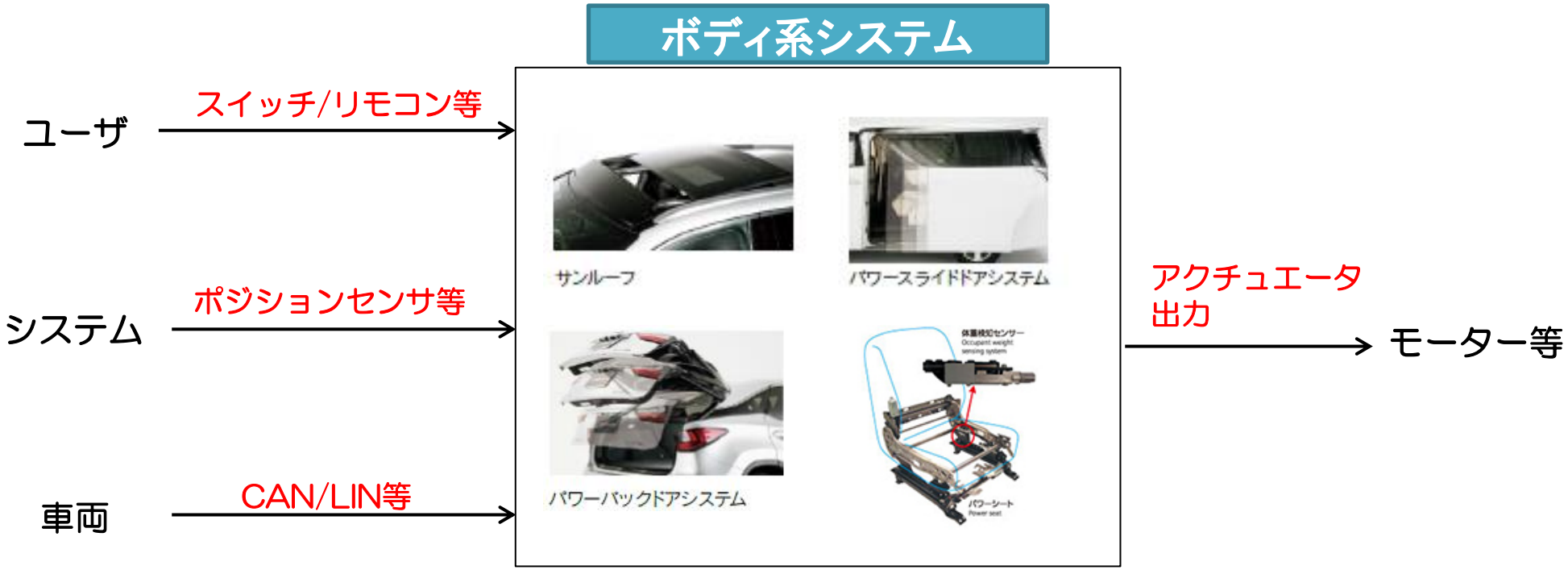


# 自動車ボディ系システムの特徴

①ユーザから直接、作動の要求がある

②車両の他システムと、車載通信による連携が必要

③駆動させるアクチュエータは1、2個（3個以上の場合も有）



# 自動車ボディ系システムに対する仕様変更の分類

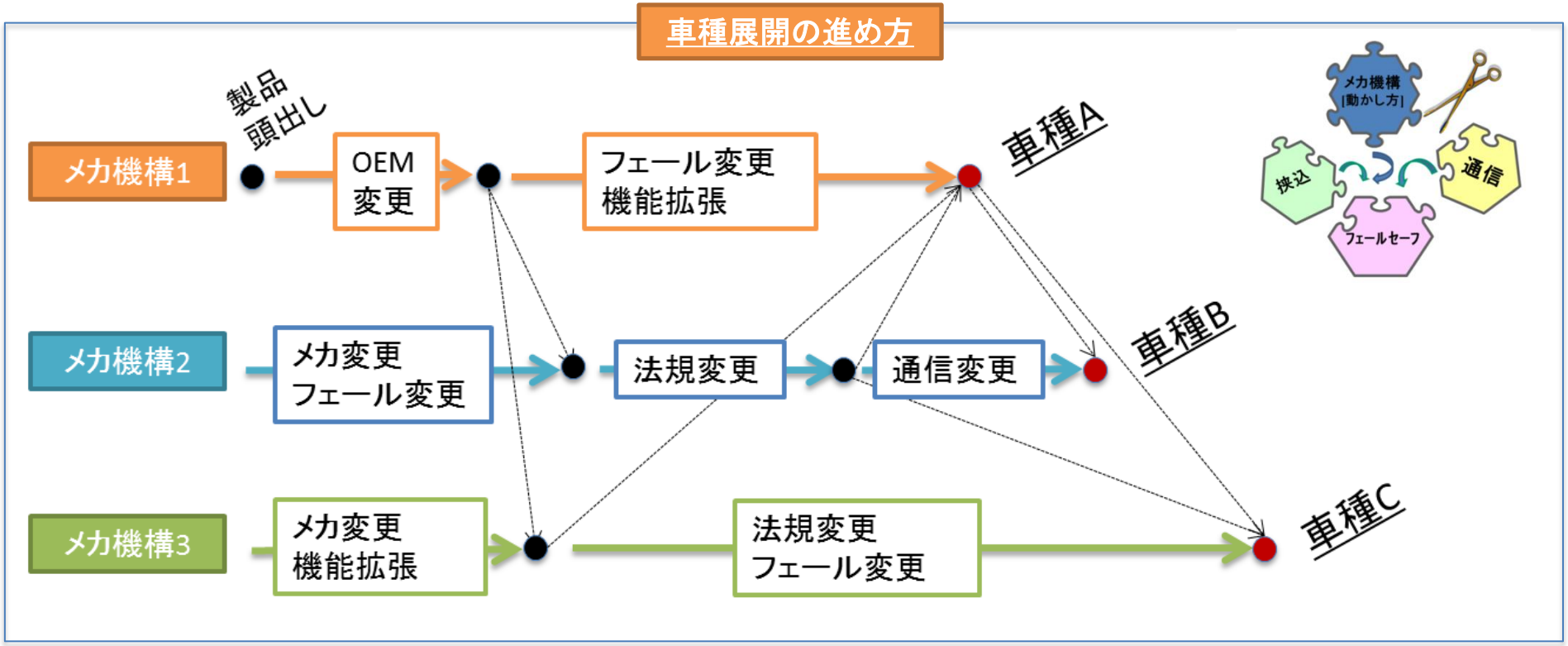
## 複数の仕様変更に、同時対応する開発を実施

| 分類 | | 例 |
|---|---|---|
| 社外的要因<br>(お客様要望) | 納入先OEM | 国内:6社、 国外:5社 |
| | 法規対応 | 日本、北米、中国、欧州 |
| | ユーザI/F | スイッチ、リモコン、カーナビ、スマホ |
| 社内的要因<br>(商品性向上やコストダウン) | メカ機構 | メカ機構1、メカ機構2、メカ機構3 |
| | ECU構成部品 | マイコン、駆動回路、センサ |

# 自動車ボディ系システムにおける開発の進め方

・開発ターゲットに一番近い仕様を、流用元として選定

・過去の類似仕様を流用して、ターゲット車種を開発




典型的な派生開発のスタイルで開発している

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

# ソフトウェア開発プロセスの定着状況

**アイシン精機では、ソフトウェア開発プロセスの定着を図ってきた**

■ **CMMI Level3**

- 2007年3月に達成

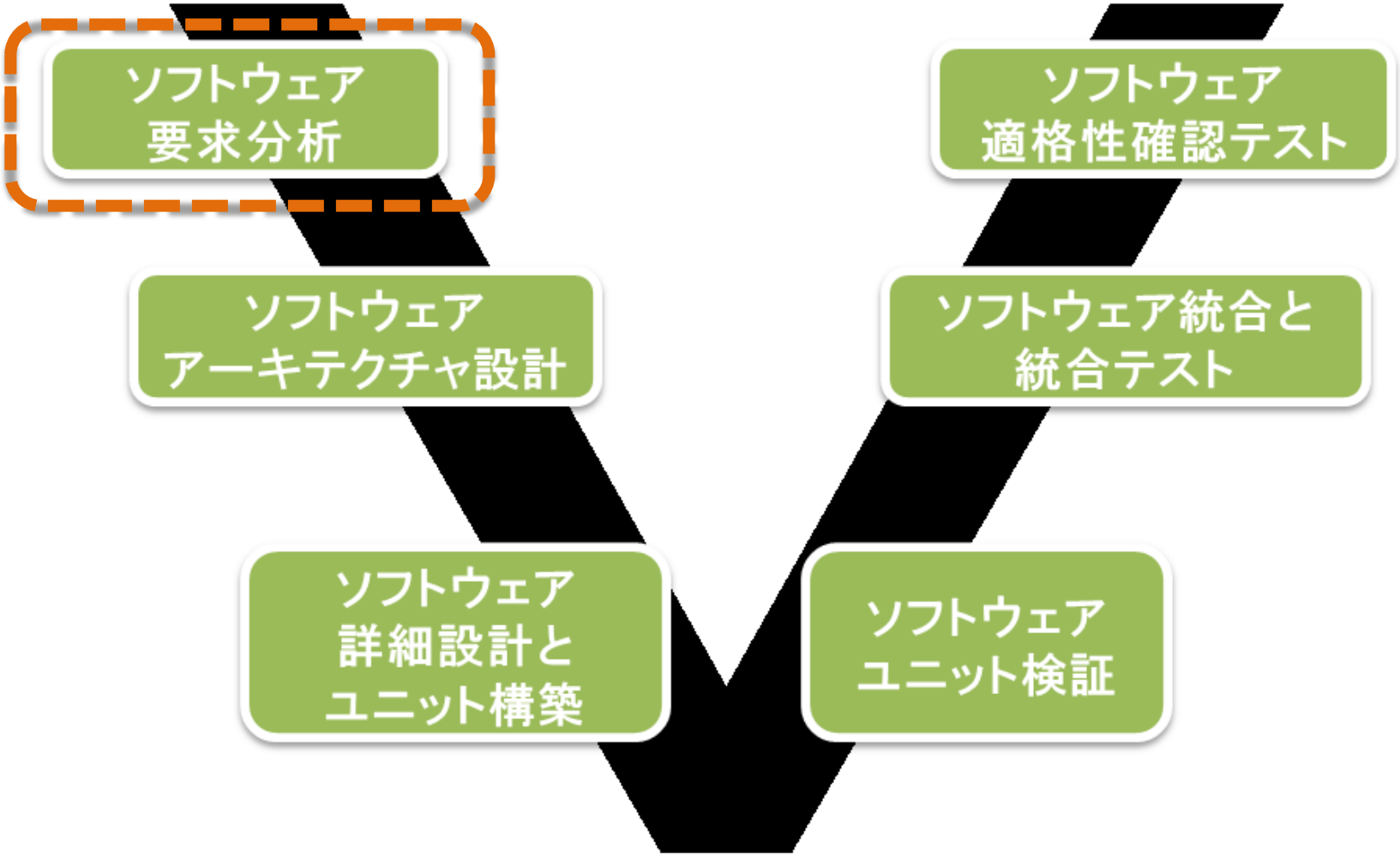
■ **ISO26262　TÜV プロセス認証**

- 2012年12月に取得

■ **社内監査部署による各開発工程でのゲートチェック実施**

- 2000年ごろから現在まで実施

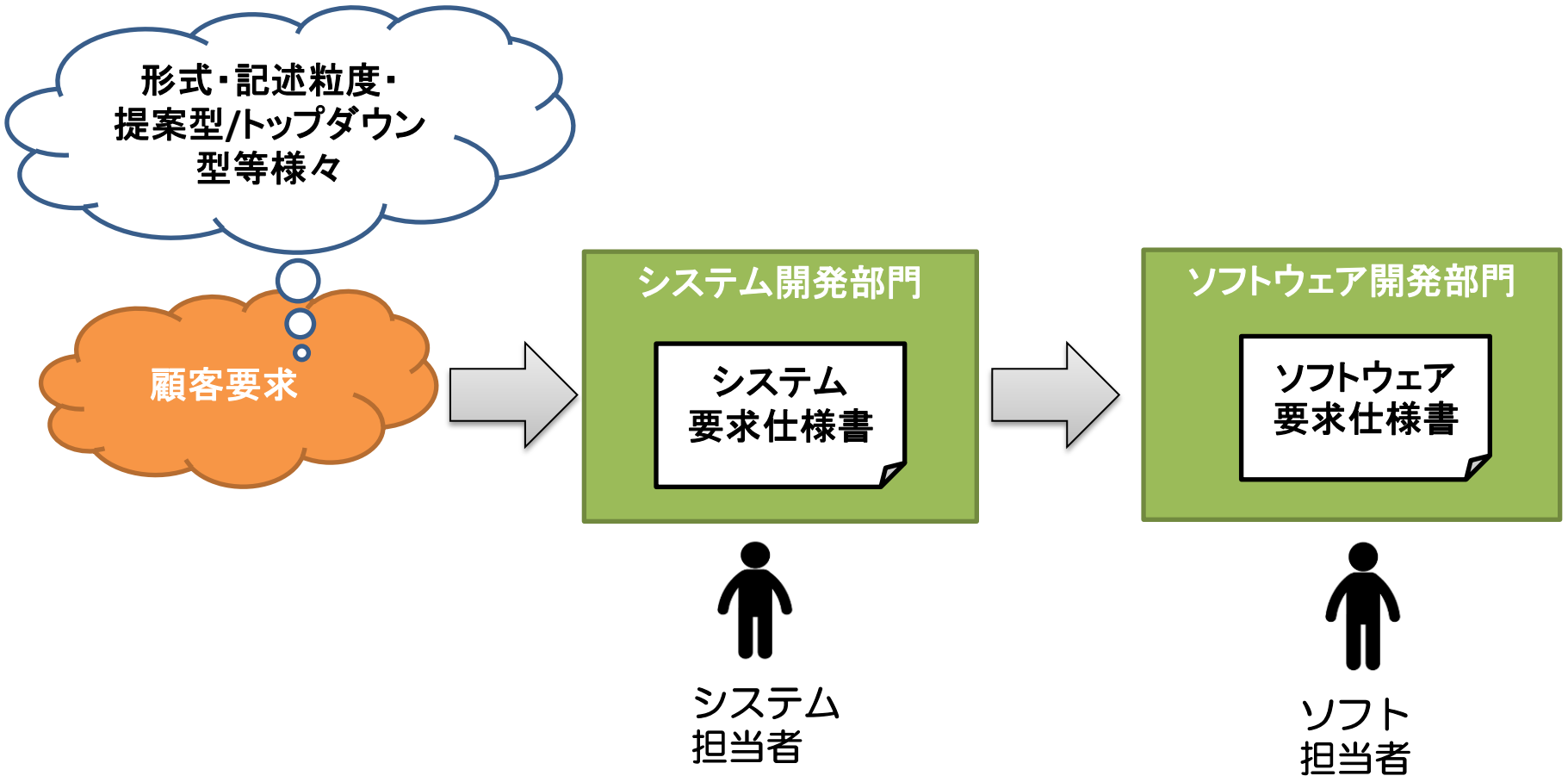
# 報告対象のソフトウェア開発工程



# ソフトウェア要求分析までのプロセスと役割分担

様々な形式・記述粒度の顧客要求を分析し、システム要求仕様書を作成

その後、ソフトウェア要求仕様を作成



# ソフトウェア開発状況

## 定義した社内プロセスにもとづき、ソフトウェア開発を実施

発表時のみ公開

## 工程の中でも、上流工程に開発工数をかけている

# 増大するソフトウェア開発工数

## 複雑化したソフトウェアから派生開発を行う為、変更箇所の把握や設計工数が増大

発表時のみ公開

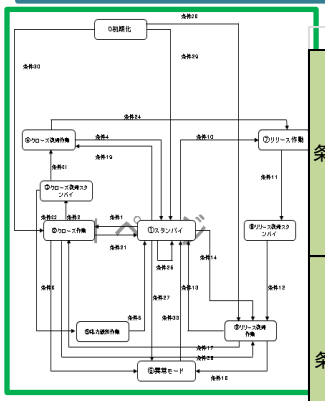
## 効率的な派生開発を行う為、ソフトウェア開発における改善活動が必要

# 発表内容

# ソフトウェア開発課題①-1：実現手段を記載したシステム要求仕様書

## システム要求仕様書に、”仕様の目的”ではなく、”実現手段”を記載している

**システム要求仕様書**



| | | 条件 | | | | |
|---|---|---|---|---|---|---|
| 条件1 | AND | 異常状態でない | | | | |
| | | OR | | AND | | |
| | | | | | OR | ○○トリガ：ON |
| | | | | | | ××SW：ON |
| | | | | | | ××SW：＊＊ |
| | | | | | | △△SW：ON |
| | | | | | | ＊＊SW：ON |
| | | | | 通信信号A：ON | | |
| | | 通信信号B：OFF | | | | |
| | | センサ情報：取得 | | | | |
| 条件2 | AND | 異常状態 | | | | |
| | | OR | | AND | | |
| | | | | | OR | ○○トリガ：OFF |
| | | | | | | ××SW：ON |
| | | | | | | ××SW：＊＊ |
| | | | | | | △△SW：ON |
| | | | | | | ＊＊SW：OFF |
| | | | | 通信信号A：OFF | | |
| | | 通信信号B：OFF | | | | |
| | | センサ情報：未取得 | | | | |

**ソフトウェア要求仕様書**

| | | 条件 | | | | |
|---|---|---|---|---|---|---|
| 条件1 | AND | 異常状態でない | | | | |
| | | OR | | AND | | |
| | | | | | OR | ○○データ：ON |
| | | | | | | ××SWデータ：ON |
| | | | | | | ××SWデータ：＊＊ |
| | | | | | | △△SWデータ：ON |
| | | | | | | ＊＊SWデータ：ON |
| | | | | 通信信号A：ON | | |
| | | 通信信号B：OFF | | | | |
| | | センサ情報：取得 | | | | |
| 条件2 | AND | 異常状態 | | | | |
| | | OR | | AND | | |
| | | | | | OR | ○○データ：OFF |
| | | | | | | ××SWデータ：ON |
| | | | | | | ××SWデータ：＊＊ |
| | | | | | | △△SWデータ：ON |
| | | | | | | ＊＊SWデータ：OFF |
| | | | | 通信信号A：OFF | | |
| | | 通信信号B：OFF | | | | |
| | | センサ情報：未取得 | | | | |

**ソフトウェア アーキテクチャ**

**複雑なソフト構造**

”実現手段”の遵守

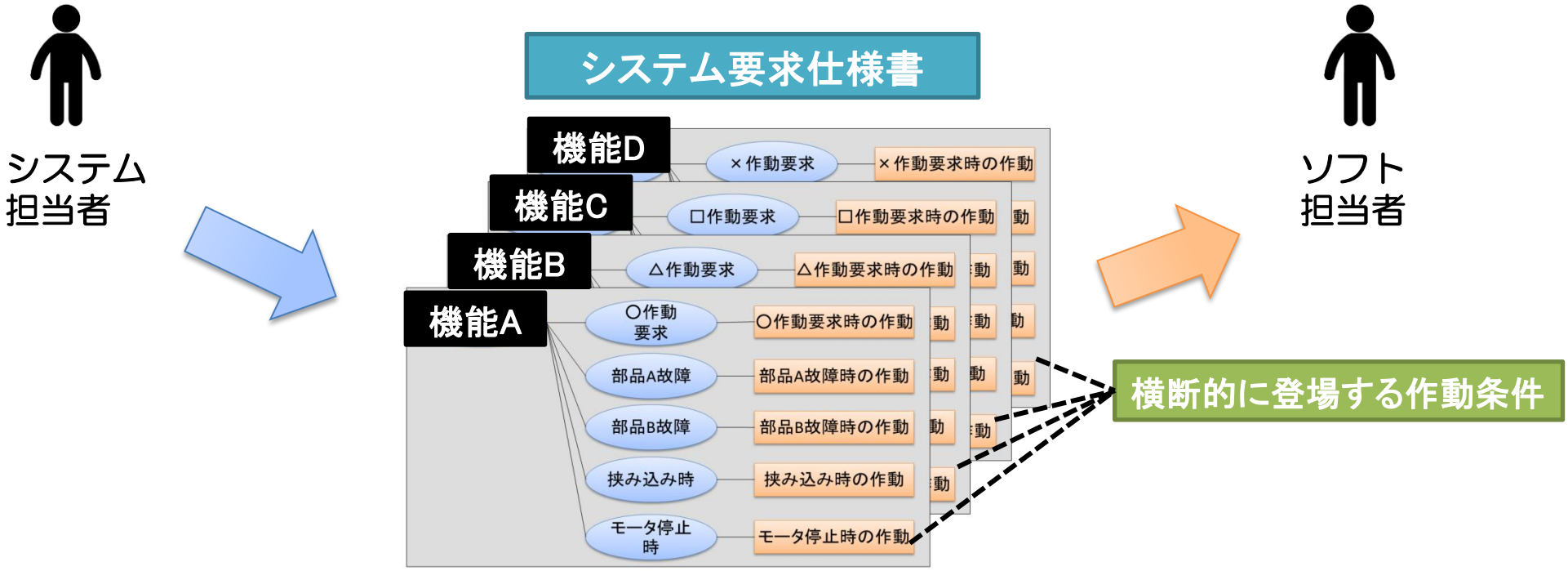
システム担当者

ソフト担当者

## ソフト担当者は、実現手段を遵守した設計を行い、ソフトウェアが複雑化

# ソフトウェア開発課題①-2：複雑なシステム要求仕様書

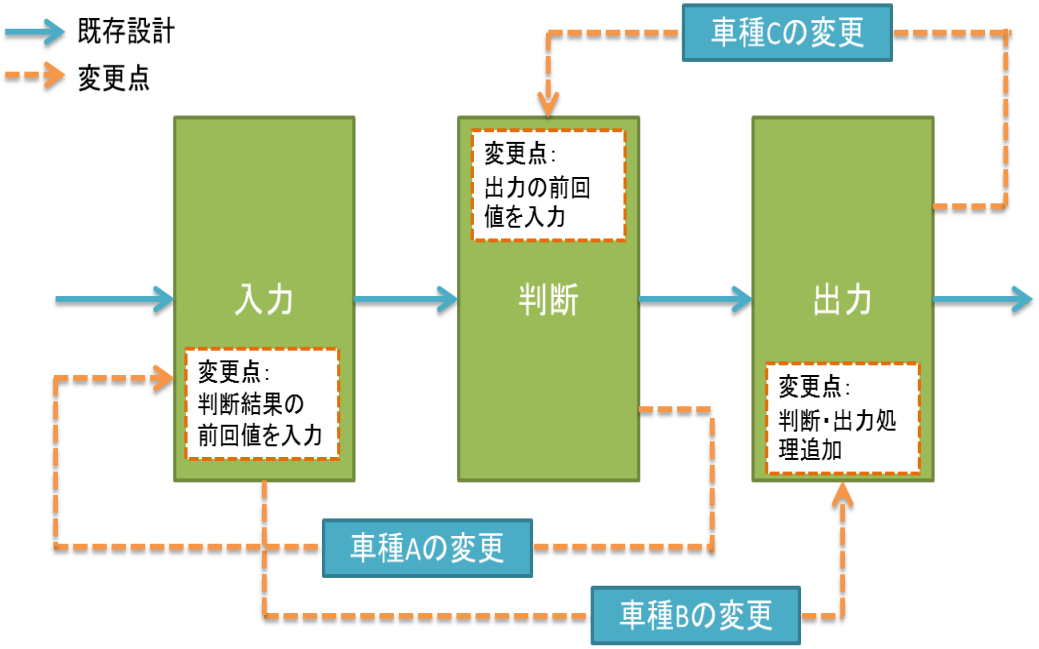
## 類似の作動条件が、複数機能に横断的に記載している




## 1つの作動条件を変更しても、複数機能にまたがっての修正が必要

# ソフトウェア開発課題②：車種展開毎に劣化するソフトウェアアーキテクチャ

## プロジェクト毎に、個別最適（コード変更量を最小化）な設計を実施




**車種毎に悪化する影響率**

※影響率：
1個のコンポーネントを変更した際、
影響を受ける、コンポーネントの割合

## 製品全体では、変更による影響箇所が多いソフトウェアとなっている

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

**4．課題解決活動(プロダクトライン開発導入)**

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

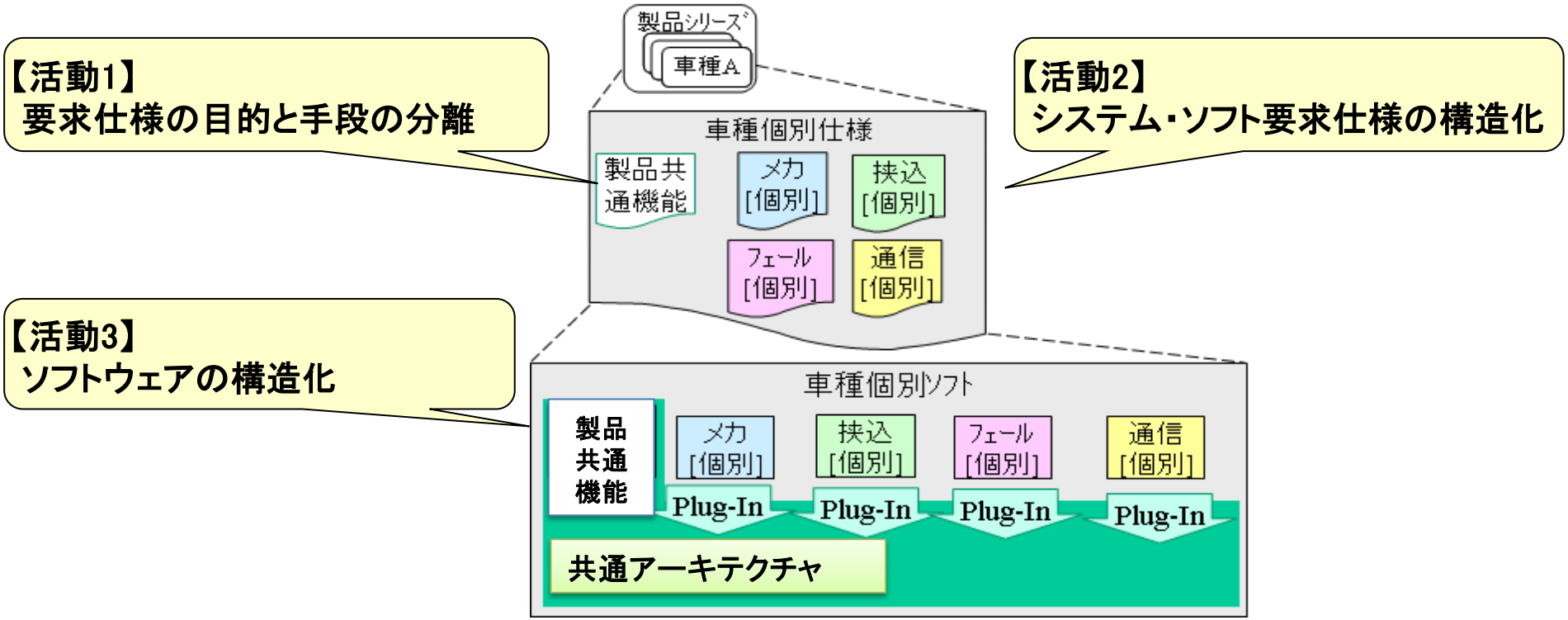
7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

# 課題解決活動：プロダクトライン開発導入

【活動1】
・要求仕様の[目的]を抽出し、[実現手段]と分離する
【活動2】
・変更頻度の高い機能[個別部]と、変わらない機能[共通部]を、要求仕様レベルで層別する
【活動3】
・個別部の変更影響を局所化した、ソフトウェアアーキテクチャを設計する



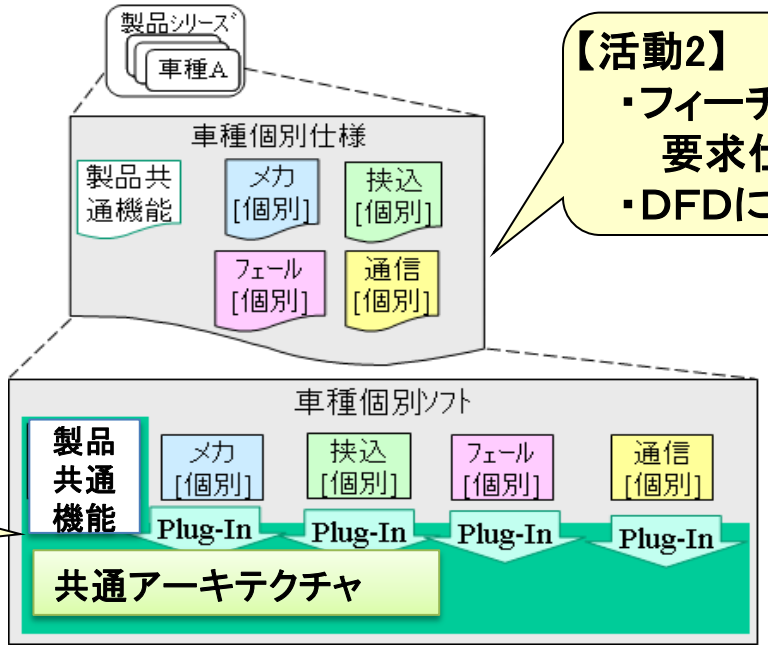
# 課題解決活動の手法

【手法1】
・フィーチャーモデルによる、要求仕様の分析[目的の抽出・手段の分離]を行う
【手法2】
・フィーチャーモデルによる、共通部と個別部の層別を行う
・DFDによる構造化分析を行い、共通部と個別部のプロセス・データを層別する
【手法3】
・DFDにもとづき、ソフトウェアアーキテクチャを設計する

【活動1】
フィーチャーモデルによる
要求仕様分析

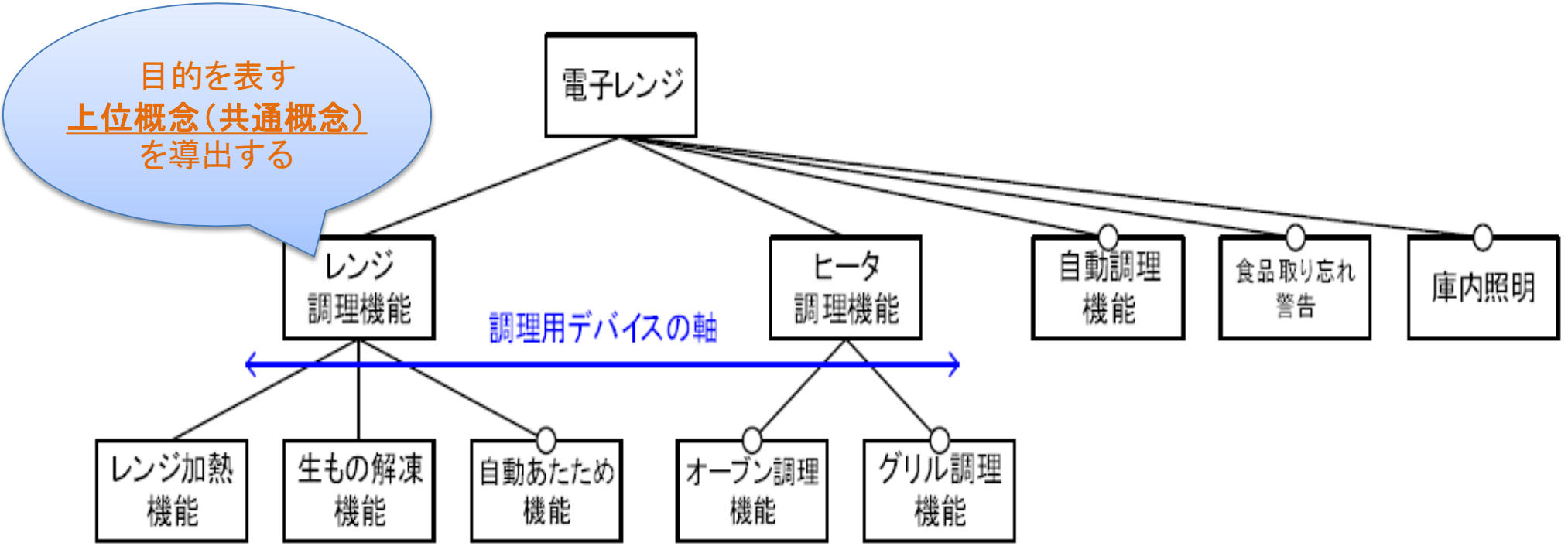
【活動2】
・フィーチャーモデルによる
要求仕様分析
・DFDによる構造化分析

【活動3】
DFDによるソフトウェア
アーキテクチャ設計



# 課題解決手段のフィーチャーモデルとは

**プロダクトライン中の製品間の共通性、可変性を各製品の備えるフィーチャとして捉えそれらの関連を記述した樹形図状のモデル（Kang, 1990; Kang 1998）**




**【留意点】**

・「目的」を問い続け、上位概念(共通概念)のフィーチャを見つける

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

**5．課題解決を取り組む際の工夫点**

6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

# フィーチャーモデルを改善する工夫

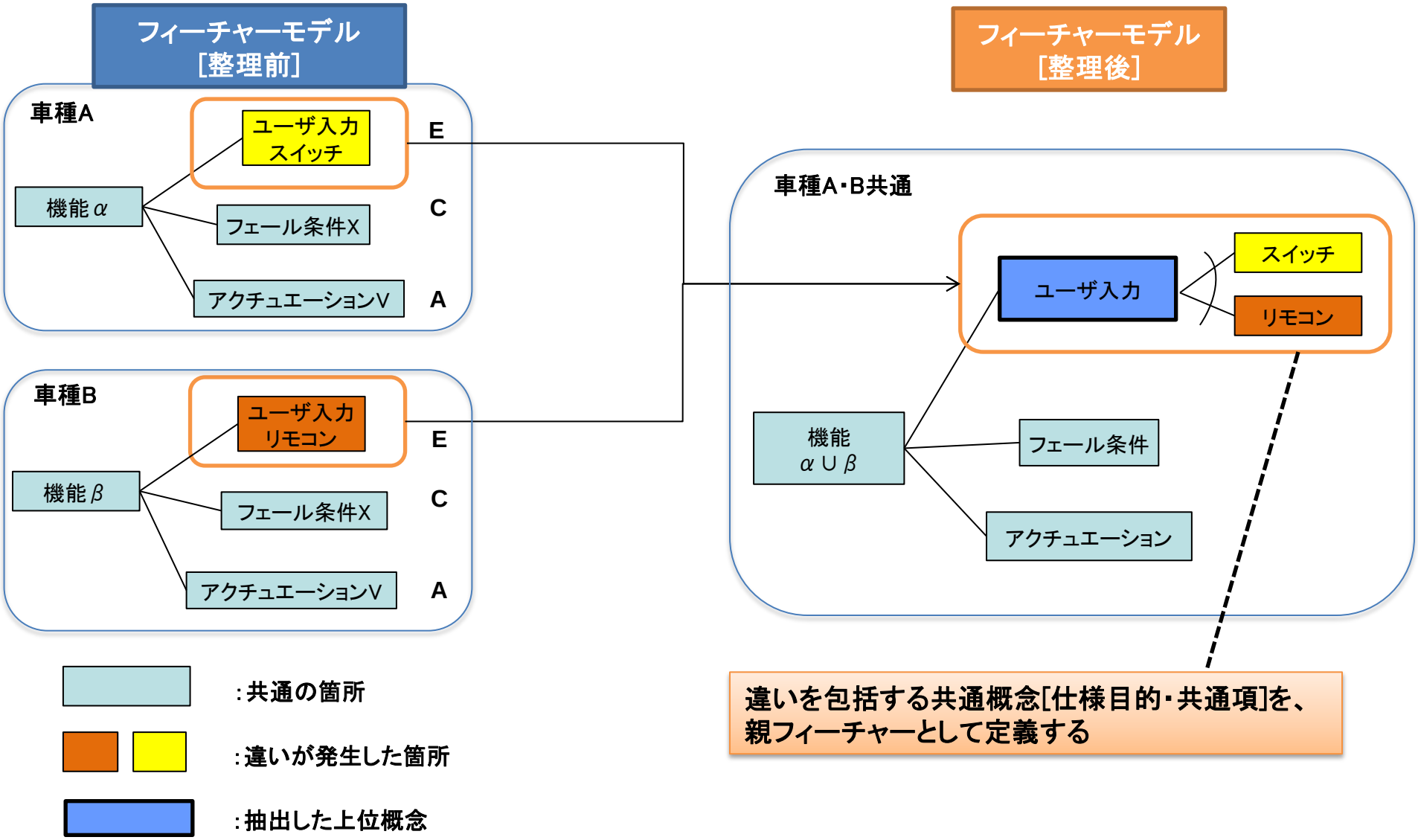
**共通概念を抽出する為、システム共通の観点でフィーチャーモデルを作成**

## 【ボディ系システム共通の観点】

**ボディ系システムは、イベント、制御条件から、アクションを決定する、“イベントドリブンシステム”**


**ECAの観点で記載内容を統一した後、フィーチャーモデルを整理する手法を考案**

# フィーチャーモデルを整理する際の考え方




**ECA観点から共通概念[仕様目的・共通項]を導出し、共通フィーチャーを作成**

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

**6．具体的な取り組み内容**

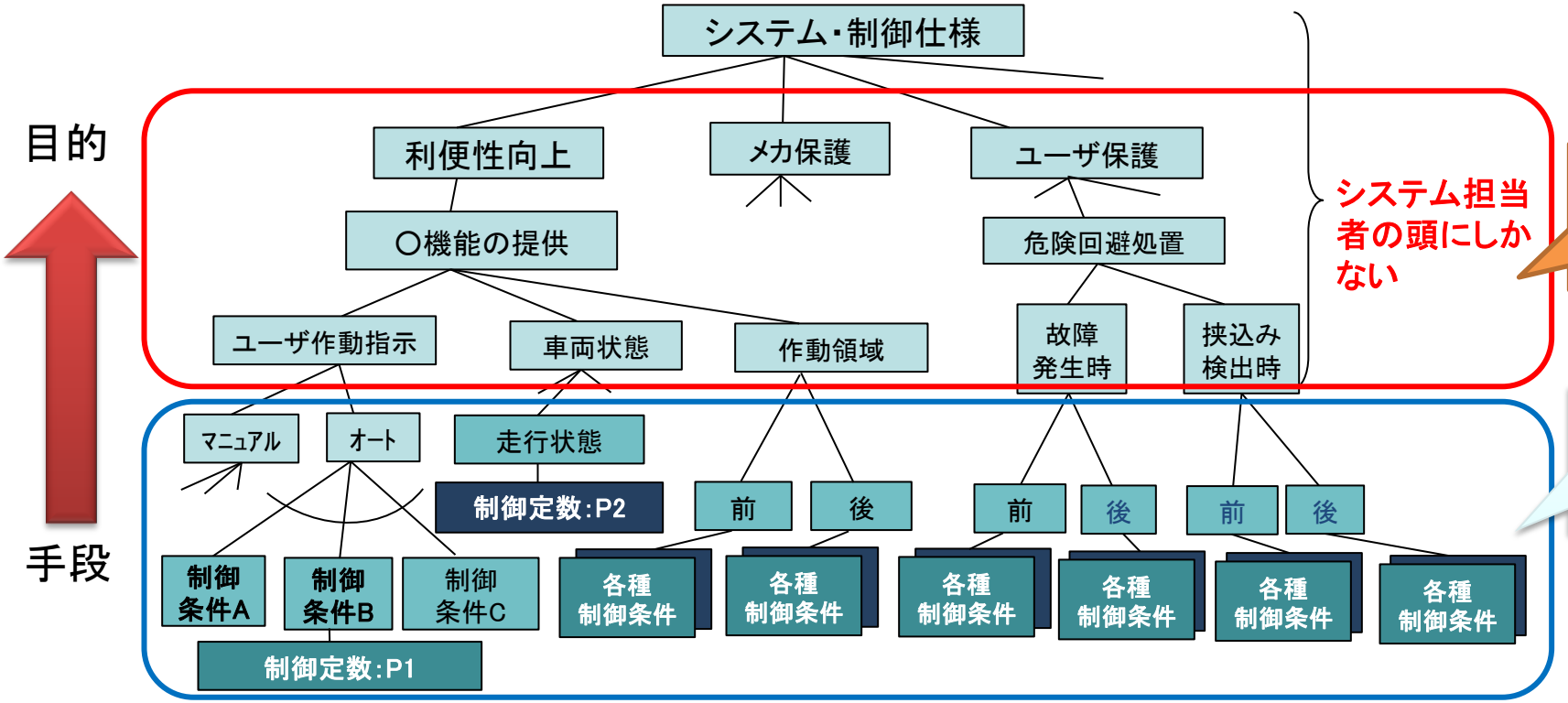
7．プロダクトライン開発導入の効果

8．まとめ

9．今後の改善活動

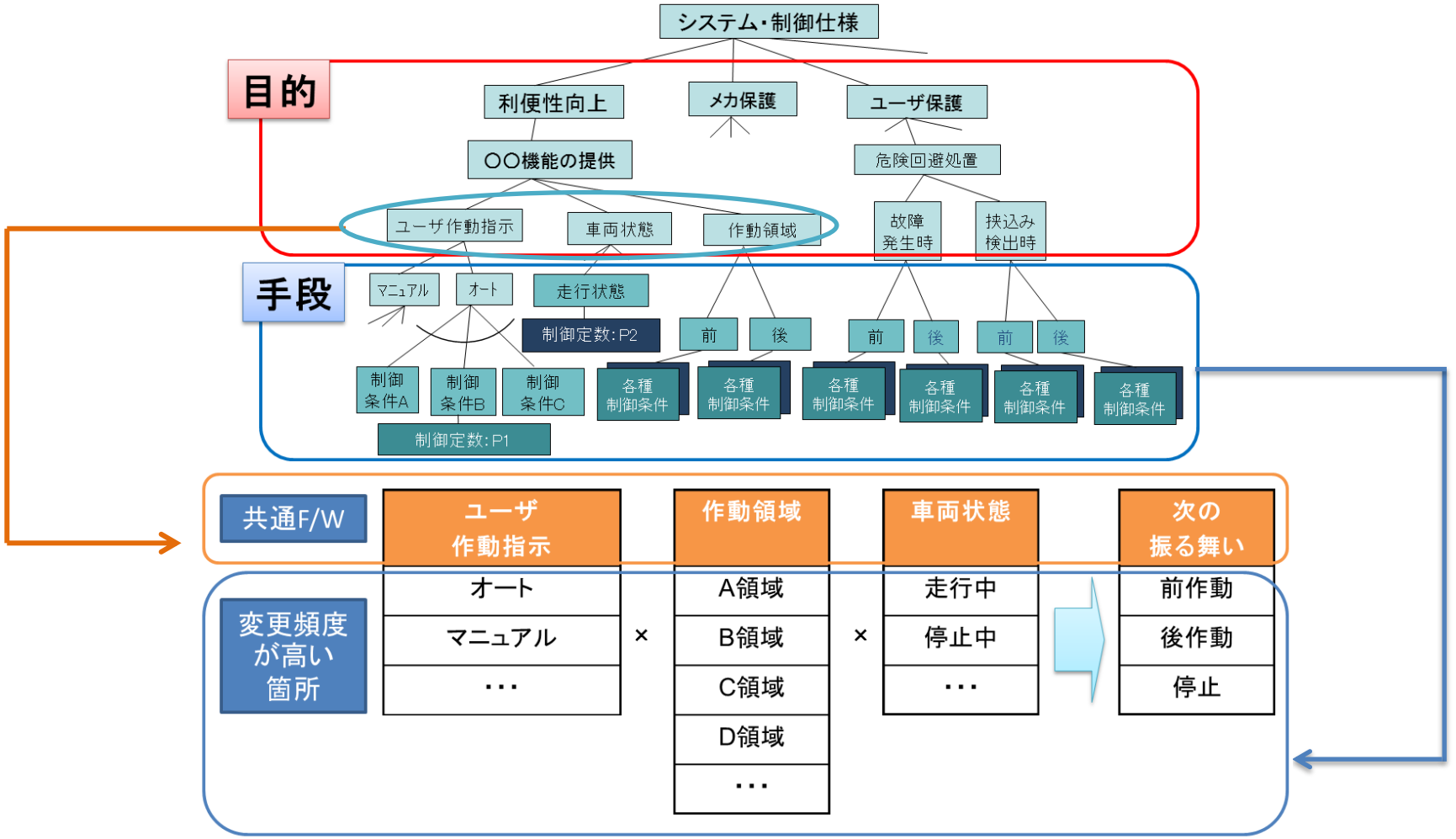
# 具体的な取り組み内容：活動1　（フィーチャーモデルによる、要求仕様の分析）

## システム担当者と協力して、要求仕様書に記載されていない”仕様目的”を抽出



# 具体的な取り組み内容：活動2　（フィーチャーモデルによる、共通部と個別部の層別）

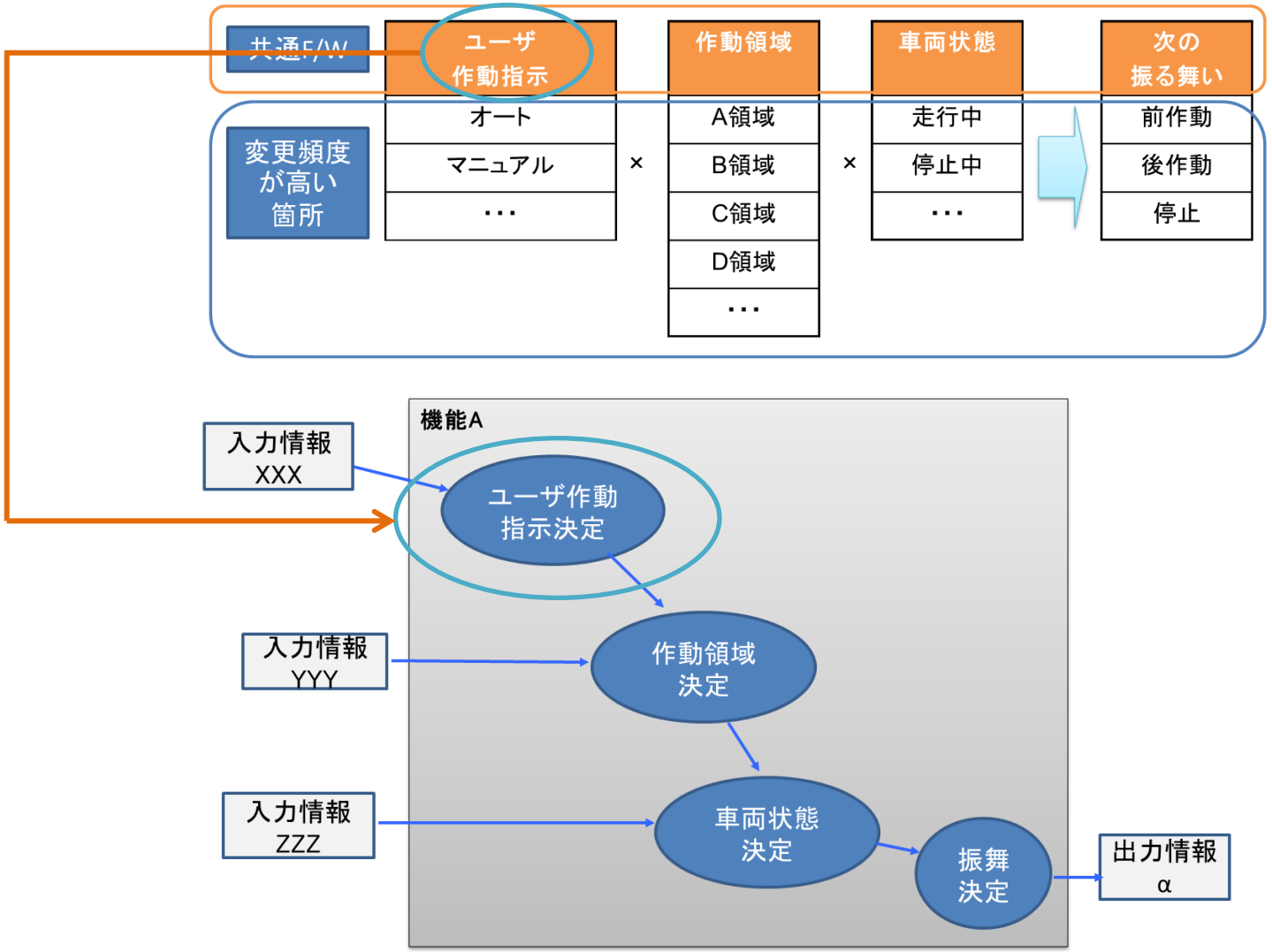
## 変更頻度が高いフィーチャーと、変更が無いフィーチャーを分類




## 変更が無いフィーチャーから、共通のF/W(フレームワーク)を抽出

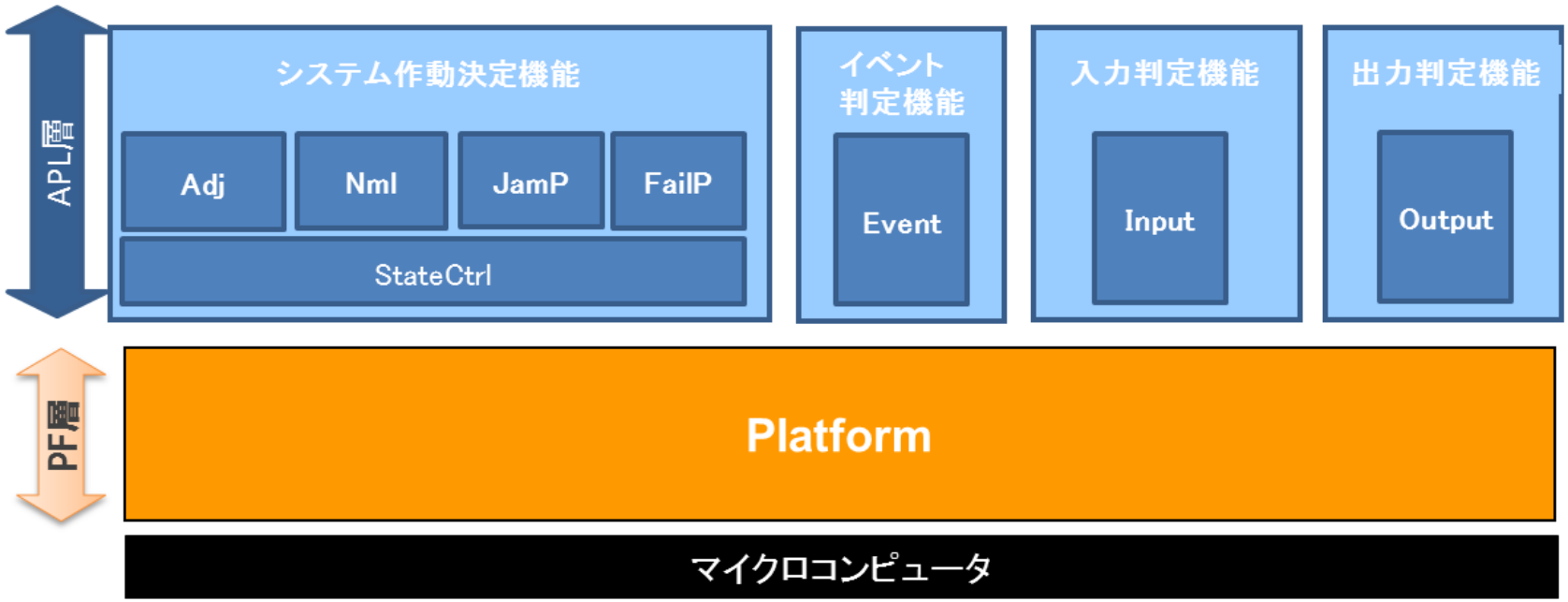
# 具体的な取り組み内容：活動2　（DFDによる構造化分析）

## 共通のF/Wから、上位階層のDFDを作成



# 具体的な取り組み内容：活動3　（ソフトウェアアーキテクチャ設計）

**特別な手法はなく、機能毎に構造化したDFDを、各ソフトウェアコンポーネントへ反映**




**APL層の変更頻度が高い部分は、全てパラメータ(2元表)変更で実現できた**

# 取り組み内容のまとめ

**取り組み内容において、特に必要不可欠な実施内容は下記の2点**

- ✓ ソフトウェアアーキテクチャの構造化設計も重要だが、
  要求仕様の整理(目的の抽出・手段との分離)がより重要

- ✓ 要求仕様の整理を行うには、
  システム担当者とソフトウェア担当者の協力が必要不可欠
  (フィーチャーモデルは、両者が協力する為の有益なツール)

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

**7．プロダクトライン開発導入の効果**

8．まとめ

9．今後の改善活動

# 効果：複雑な要求仕様記述の改善

## システム・ソフトウェア要求仕様書に対応するフィーチャーノードの数を削減できた

発表時のみ公開

## 仕様変更時に、変更箇所を特定する工数が低減可能となった

# 効果：複雑なソフトウェア構造の改善

## 簡素化した依存関係のソフトウェアアーキテクチャを実現できた

発表時のみ公開

## 変更箇所が局所化できた為、ソフトウェア構造設計工数が低減可能となった

# 効果：ソフトウェア構造指標の改善

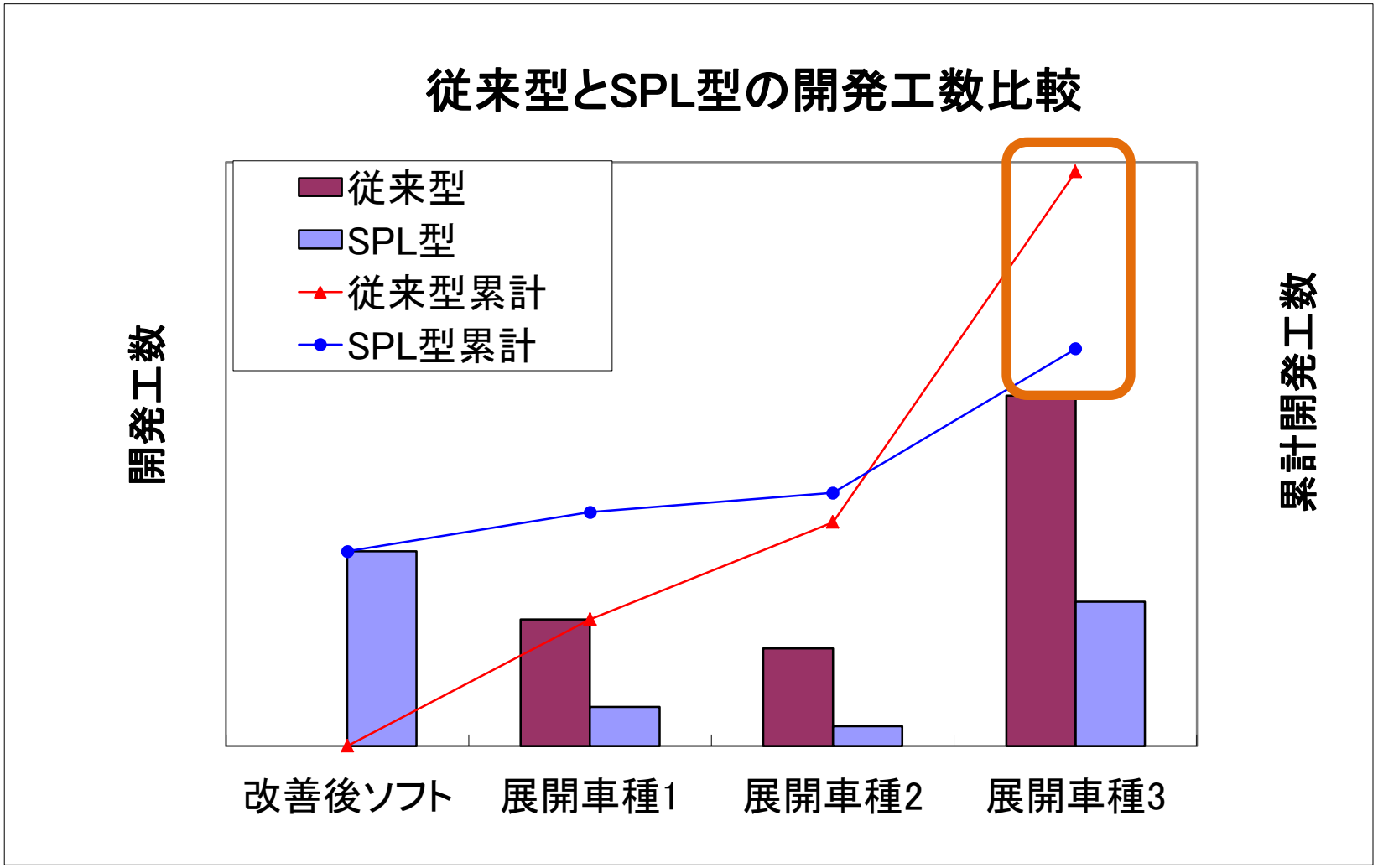
## 1個のソフトウェアコンポーネントの変更に対する影響範囲(影響率)を低減できた

発表時のみ公開

## 影響率は1/8に改善し、車種展開を進めても、効果を維持することができた

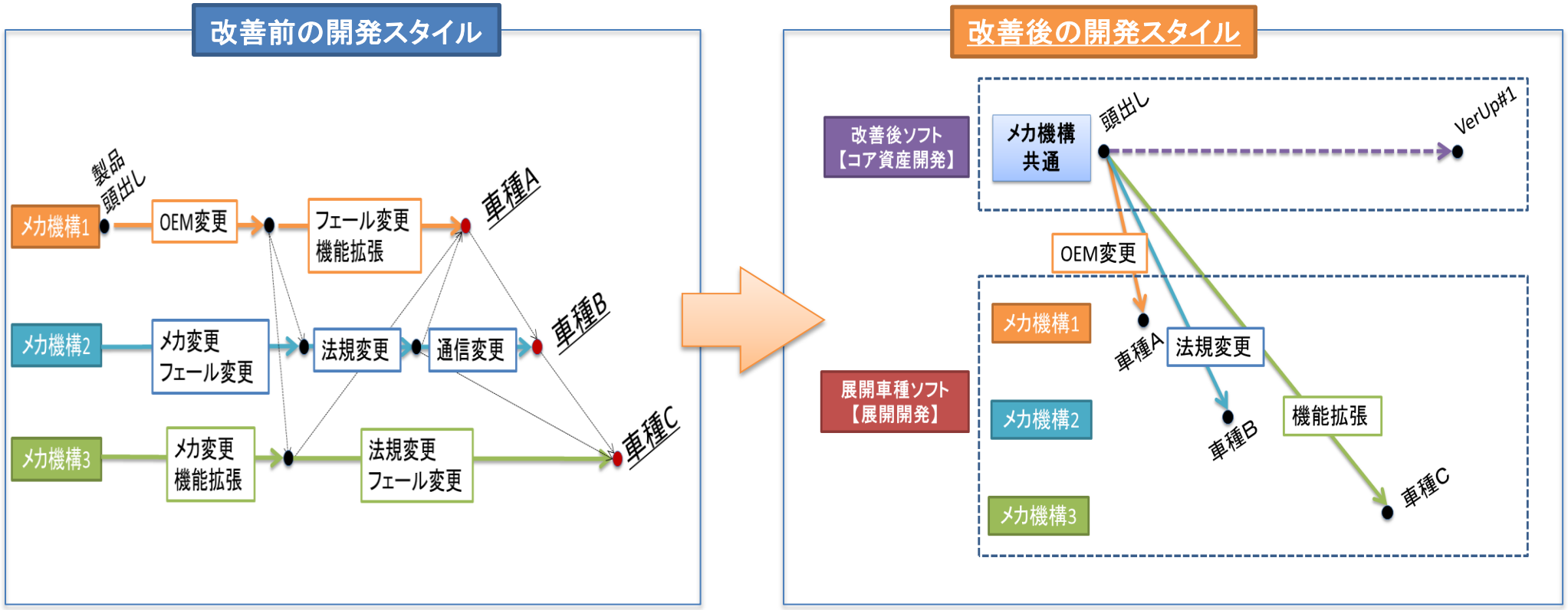
# 効果：開発コストの低減

**3車種目の展開開発にて、改善後ソフト開発分の初期投資を回収できた**



# 開発の進め方の変化

## 派生元は必ず改善後ソフト[コア資産]をベースとする開発へ変化




## プロダクトライン型開発への移行を目途づけできた

# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

**8．まとめ**

9．今後の改善活動

## ■ 改善効果を得られた要因

- ソフトウェアアーキテクチャ設計ではなく、
  要求仕様の整理(目的の抽出・手段との分離)に重点を置いた活動をしたこと
  - ✓ 活動において、システム担当者の協力を得られたこと
  - ✓ フィーチャーモデルを用いることで、
    システム担当者とソフトウェア担当者の理解を共有することができたこと

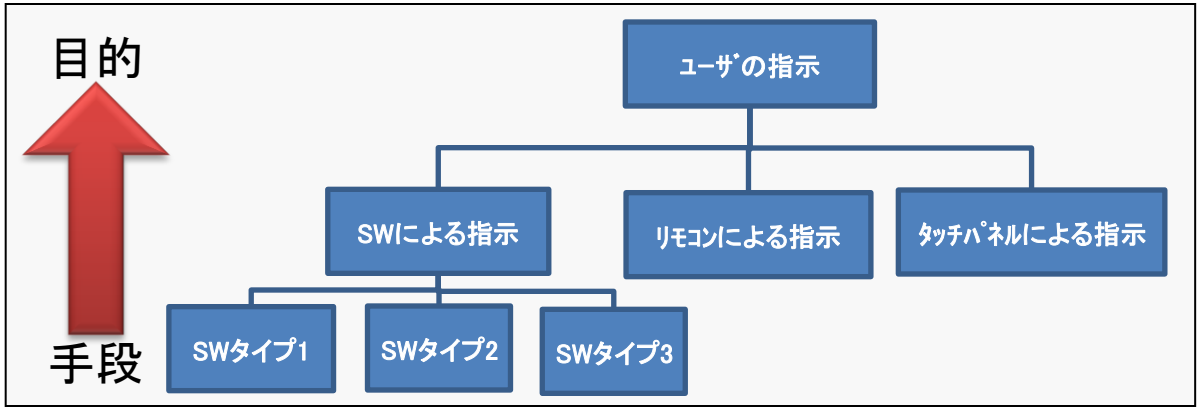
## ■ プロダクトライン開発で効果的な手法を獲得できた

- 特に以下の手法が効果的であった
  - ✓ フィーチャーモデルによる、”抽象化”と”関心事の分離”

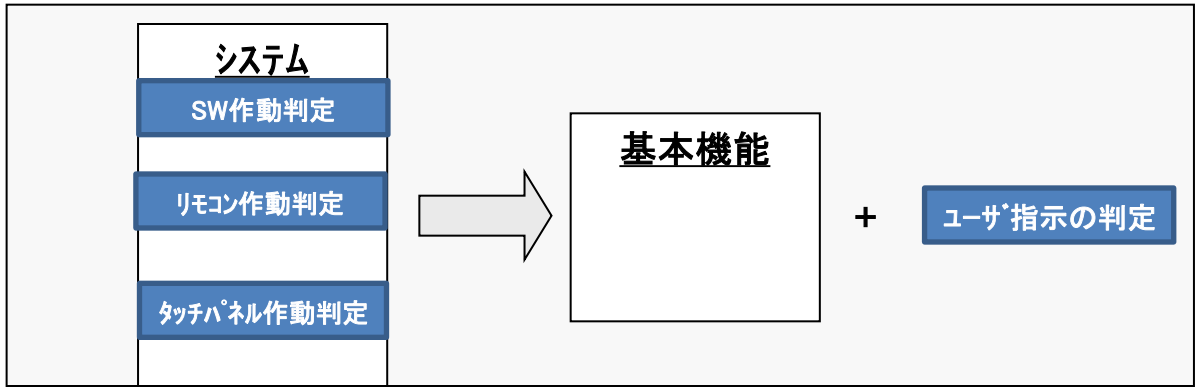
# まとめ

## プロダクトライン開発導入で重要な技術

- **抽象化**
  - 実現手段の違いではなく、目的にさかのぼって纏める技術




- **関心事の分離**
  - システム仕様の中で横断的に出現する共通的な機能[関心事]を、抽出して独立させる技術



# 発表内容

1．はじめに

2．ソフトウェア開発状況

3．ソフトウェア開発における課題

4．課題解決活動(プロダクトライン開発導入)

5．課題解決を取り組む際の工夫点

6．具体的な取り組み内容

7．プロダクトライン開発導入の効果

8．まとめ

**9．今後の改善活動**

# 更なる改善項目

## ■ 改善した要求仕様書の記載内容について

- ✓ システム要求仕様書に”実現手段”を書いてしまう
  - ➢ システム担当者は、”実現手段”がシステム動作のイメージに繋がり易い
    - – 担当分野ごとに、理解しやすい表現の違いに起因
      （システム担当：ユースケースを意識した表現　⇔　ソフト担当：ソフト構造を意識した表現）
- ✓ 目的を記載しているつもりでも、日本語の文章にすると、手段を記載してしまう

## ■ フィーチャーモデルの作成手法について

- ✓ フィーチャーモデルのまとめ方が、担当者によってばらついている
  - – 共通概念を抽出できる人と、できない人がいる
    （まとめる観点の有無や、開発経験の差　等）

# 今後の改善活動

## ■ 要求仕様書の記載内容改善

- ✓ システム、ソフトウェア担当者が、相互理解しやすい表現を確立する
- ✓ 日本語表現における、目的・手段の記述内容を改善する

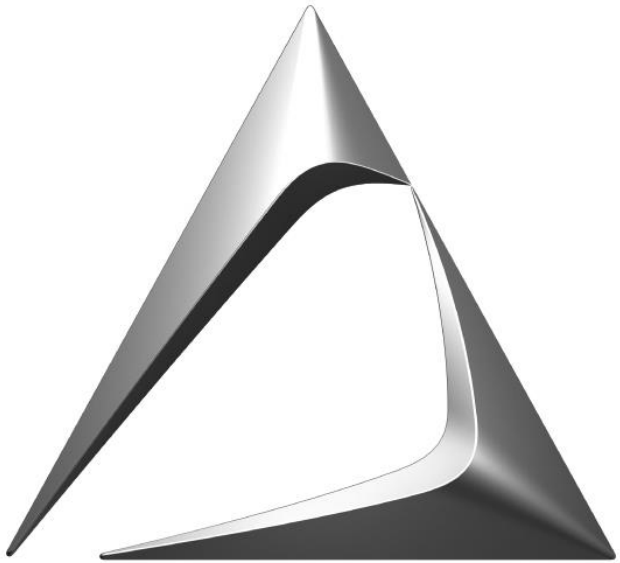
## ■ フィーチャーモデルの作成手法改善

- ✓ ECAの観点から共通概念の抽出ノウハウを、一般化する

## ■コア資産の効果を維持する為のプロセス定義

- ✓ コア資産の機能拡張や、派生開発からのフィードバックプロセスを定義する

**これらの改善活動を継続し、さらなる開発効率の向上をめざしていきたい**



For a Better Tomorrow

AISIN GROUP

# ご清聴ありがとうございました

# 参考文献


[1] P. Clements and L. Northrop, *Software Product Lines: Practice and Patterns*, Addison-Wesley, 2001.（邦訳: 前田 卓雄（翻訳）,『ソフトウェアプロダクトライン: ユビキタスネットワーク時代のソフトウェアビジネス戦略と実践』, 日刊工業新聞社, 2003.）

[2] K. Pohl, G. Böckle, and F. v. d. Linden, Software Product Line Engineering: Foundation, Principles, and Techniques, Springer, 2005.（邦訳: 林 好一, 吉村 健太郎, 今関 剛（訳）,『ソフトウェアプロダクトラインエンジニアリング: ソフトウェア製品系列開発の基礎と概念から技法まで』, SiBアクセス, 2009.）

[3] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, SEI/CMU, Nov. 1990.

[4] K. C. Kang, S. Kim, J. Lee, K. Kim, G. J. Kim, and E. Shin, "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architecture," Annals of Software Engineering, Vol.5, pp.143-168, 1998.

[5] F. v. d. Linden, K. Schmid, and E. Rommes, *Software Product Lines in Action*, Springer, 2007.

[6] M. Steger, C. Tischer, B. Boss, A. Müller, O. Pertler, W. Stotz, and S. Ferber, "Introducing PLA at Bosch Gasoline Systems: Experiences and Practices," Proc. Software Prodcut Line Conf. 2004, pp. 34-50, Aug. 2004.

[7] Y. Takebe, N. Fukaya, M. Chikahisa, T. Hanawa, and O. Shirai, "Experiences with Software Product Line Engineering in Product Development Oriented Organization," Proc. Software Product Line Conf. 2009, pp. 275-283, Sep. 2009.

[8] T. Iwasaki, M. Uchiba, J. Ohtsuka, K. Hachiya, T. Nakanishi, K. Hisazumi, and A. Fukuda, "An Experience Report of Introducing Product Line Engineering across the Board," Proc. 14th Int. Software Product Line Conf. (SPLC) 2010, Vol.2, pp.255-258, Sep. 2010.

[9] 岩崎 孝司, 内場 誠, 大塚 潤, 八谷 浩二, 中西 恒夫, 久住 憲嗣, 福田 晃,「プロダクトライン開発手法の全社的導入に関する一事例報告」, 組込みシステムシンポジウム2010（ESS2010）予稿集, 2010年10月.

[10] 大塚 潤, 河原畑 光一, 岩崎 孝司, 内場 誠, 中西 恒夫, 久住 憲嗣, 福田 晃,「大規模移動体ネットワーク機器ファームウェア開発へのソフトウェアプロダクトライン適用事例」, 組込みシステムシンポジウム2011（ESS2011）予稿集, 2011年10月.