組込み系ソフトウェア開発の課題分析と提言

～大規模化，短納期化，多機種開発にどう立ち向かうか～

（JEITA活動報告）

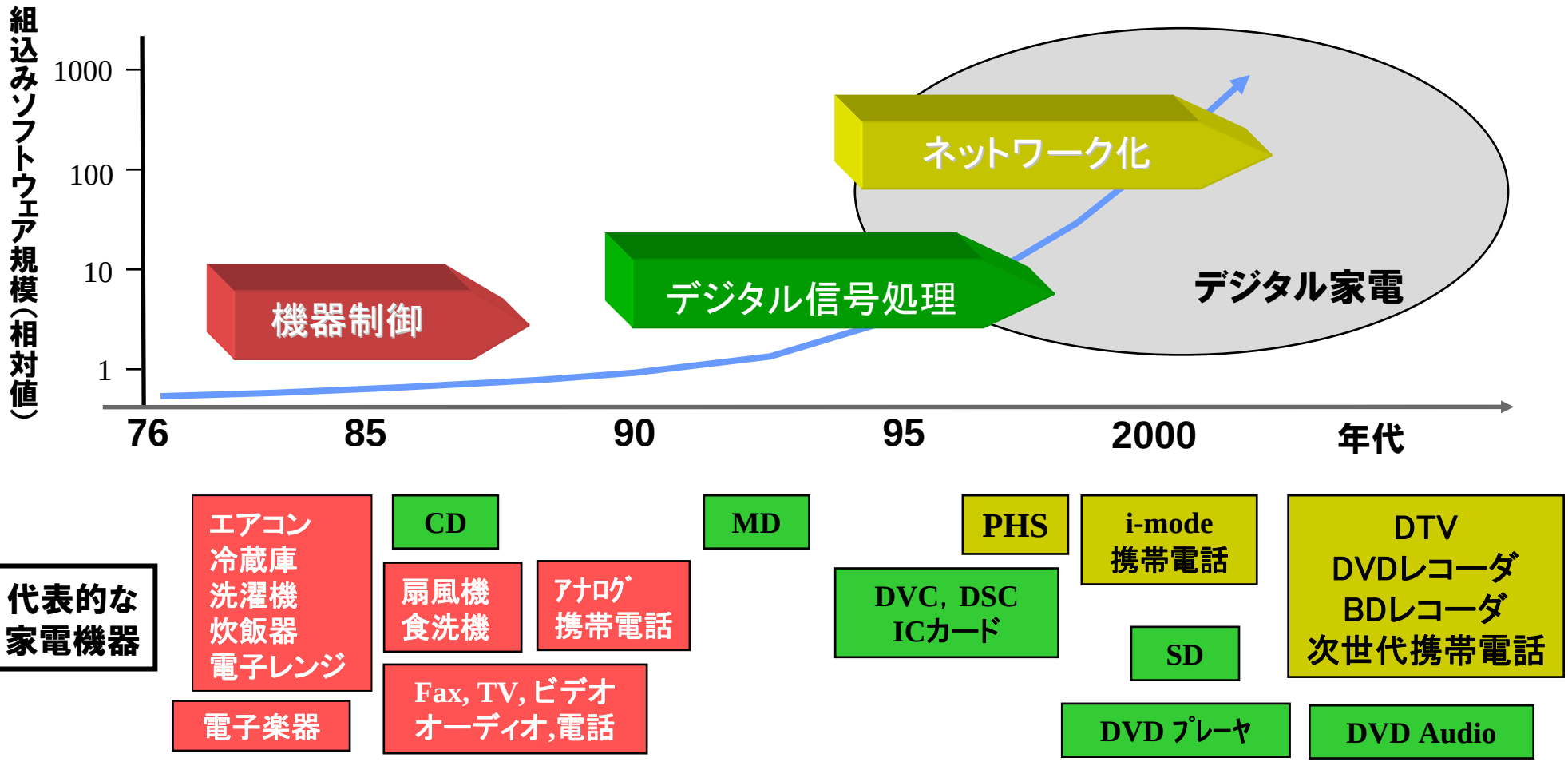
2007年10月2日

JEITA　ソフトウェア事業基盤専門委員会　委員

松下電器産業株式会社
システムエンジニアリングセンター
春名　修介

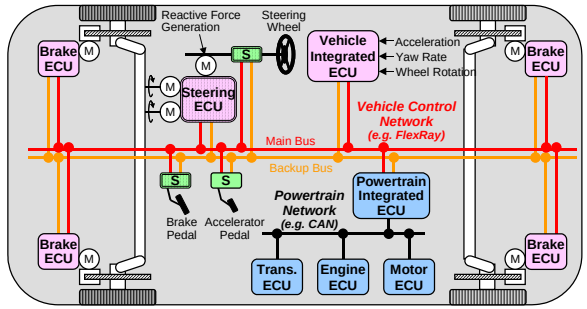
組込みソフトウェアの現状：デジタル家電

- 小規模であった頃の開発文化を引きずりながら，デジタル化による急激な規模拡大に遭遇



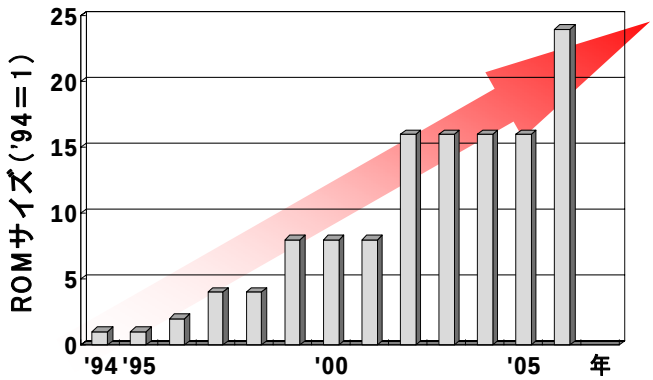
組込みソフトウェアの現状：車両制御

- **電子化の進む車両制御システム**
  - 制御ユニットの大規模・複雑化
    - » 制御ユニット数：70ユニット以上（1車両あたり）
    - » 総プログラムサイズ：7M Step以上



車両制御ネットワークの例

- **各社の取り組み**
  - 開発効率向上：モデルベース開発，コード自動生成
  - 開発量削減：ソフトウェア再利用
  - オブジェクト指向設計，社内標準アーキテクチャ



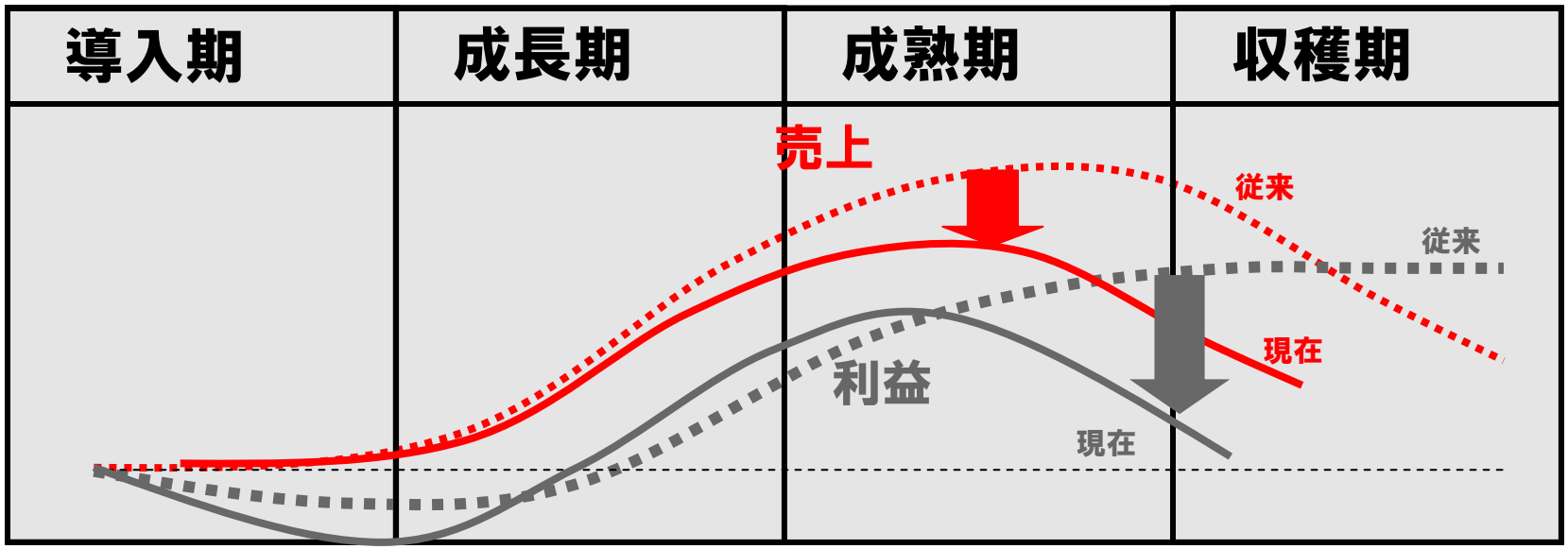
エンジン制御用マイコンメモリサイズの一例

各社個別の対応に限界
↓
業界標準によるソフト再利用

商品ライフサイクルの短縮：デジタル家電

- アナログ家電
  - 成熟期以降の製品寿命が長かった
- デジタル家電
  - 開発した機能の陳腐化が早い
  - 常に付加価値部分を積上げての価格維持
  - 多様な顧客層への対応による機種の増加

⇨ 短納期化，多機種化の加速



現場で起こっていること：開発事例①

- **コーディング主体の開発**

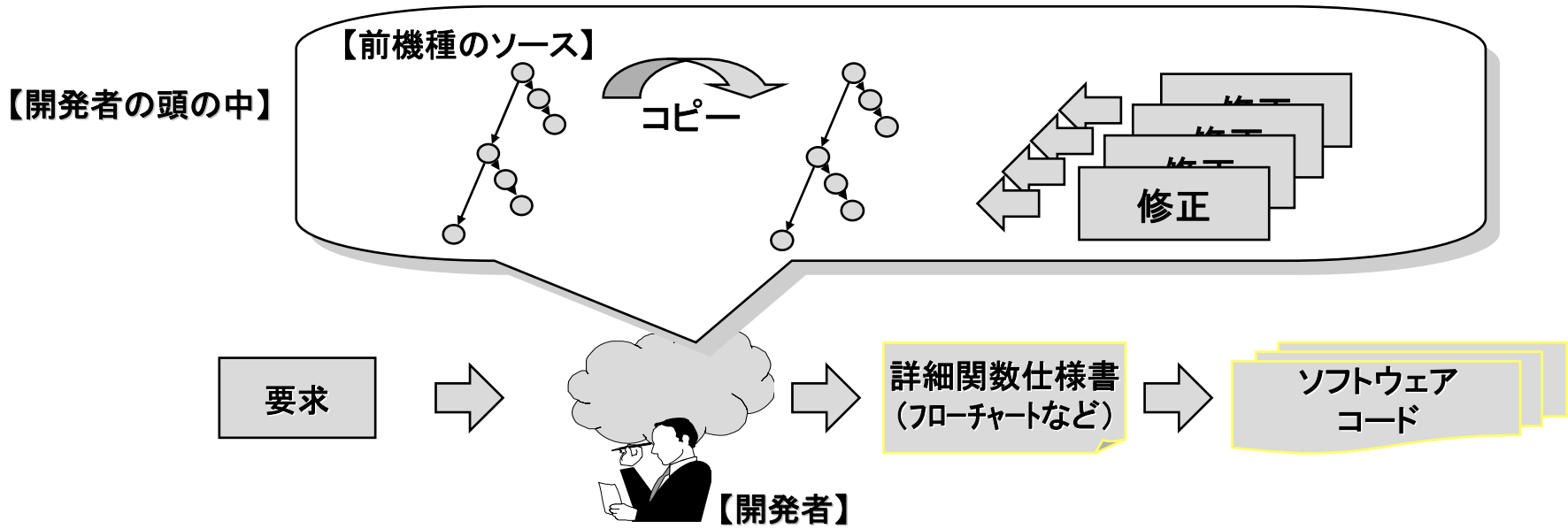
  ＜小規模時代の開発のなごり，短納期のプレッシャ＞

  – **以前の機種のソフトウェアをコピーし，必要な部分のみを修正・追加（差分開発，コピー＆ペースト開発）**

    » **要求からコードへのブレークダウン過程が開発者の頭の中（属人的な開発）**

    » **場当たり的な修正によるコードの複雑化**

    » **機種開発数の増加，担当者の変更で急激に開発効率が低下**



現場で起こっていること：開発事例②

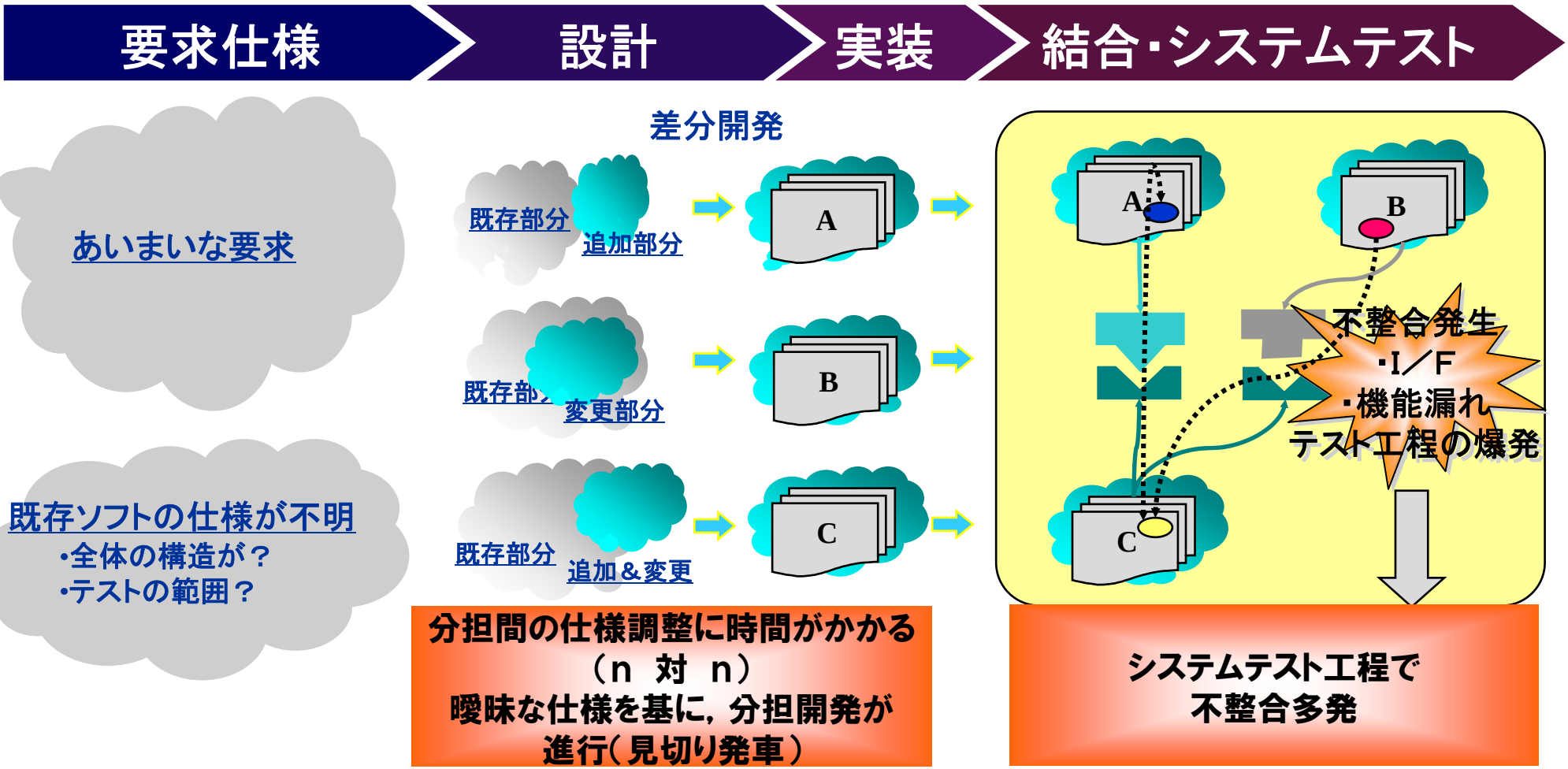
- 生産性向上には再利用が効果的であるが，再利用が有効に機能していない
  - 流用率は高いが生産性は思った程，向上していない

- 修正箇所特定のための**コード解析**
- 修正による影響範囲が不明なために**全体再テストと改造の繰り返し**

- 全体構造が不明確なまま開が進行．全体を俯瞰できる仕組みの欠如．作ってからの再利用（アドホック再利用）の限界

現場で起こっていること：開発事例③

■ 分担のみ決まっており，全体把握ができていない



有効な施策と，現実のギャップ

- 開発規模の拡大，多機種開発への対処としては，本来，再利用有効なはずであるが，下流工程での擦り合せ開発が横行

  既存ソフトを流用．上手く行く筈・・・でも，動かない！

  - 原因を特定しようと夜を徹して調べるけど判らない！
  - では，かつての開発者に聞いてみよう！
  - 残念ながら，その開発者は，もう居ない・・・

⇩

- 再利用は，昔から叫ばれているが，現場に定着した例は？
  - キーマンが変れば，元の木阿弥．．．．

⇩

- 作ってからの再利用は，効果が薄く，再利用を考えた戦略的な開発への発想転換が必要

⇩

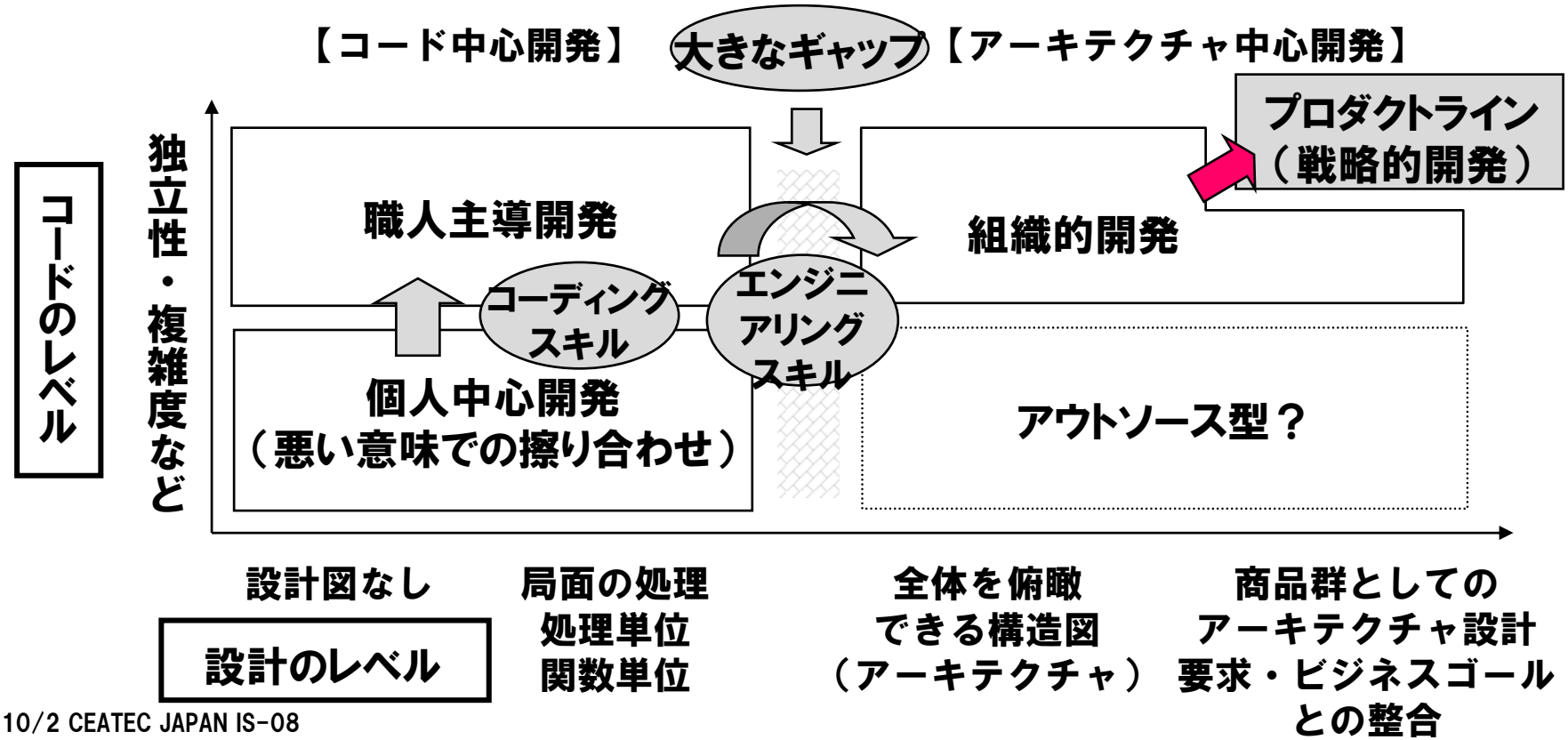
- 再利用を考えたアーキテクチャ設計とコンポーネント設計

組込みソフトウェア開発の課題：まとめ

- コーディング主体の開発形態が浸透
  - 小規模時代の開発のなごり，短納期のプレッシャ
  - 資源制約化での開発が長く続いたため，最適化のためのコーディング技法が重要視されてきた
  - 統合後の擦り合せが不可欠というハードウェア制御の特性が全体を支配
- 全体を俯瞰する仕組みが確立されていない
  - 全体構造をコーディング前に確定するアーキテクチャ設計がなされていない
  - アーキテクチャ設計を実施するアーキテクトが不明確．組込み分野でのアーキテクトとしての要件が未確立であり，育成ができていない
- トップダウン設計の全面採用は難しい
  - 擦り合せ開発の要素は排除できない
    - » ハードウェア制御，日本の強みである高品質開発，集団合議による意思決定体制など
- 既存資産を捨てることはできず，新アーキテクチャへの移行が難しい

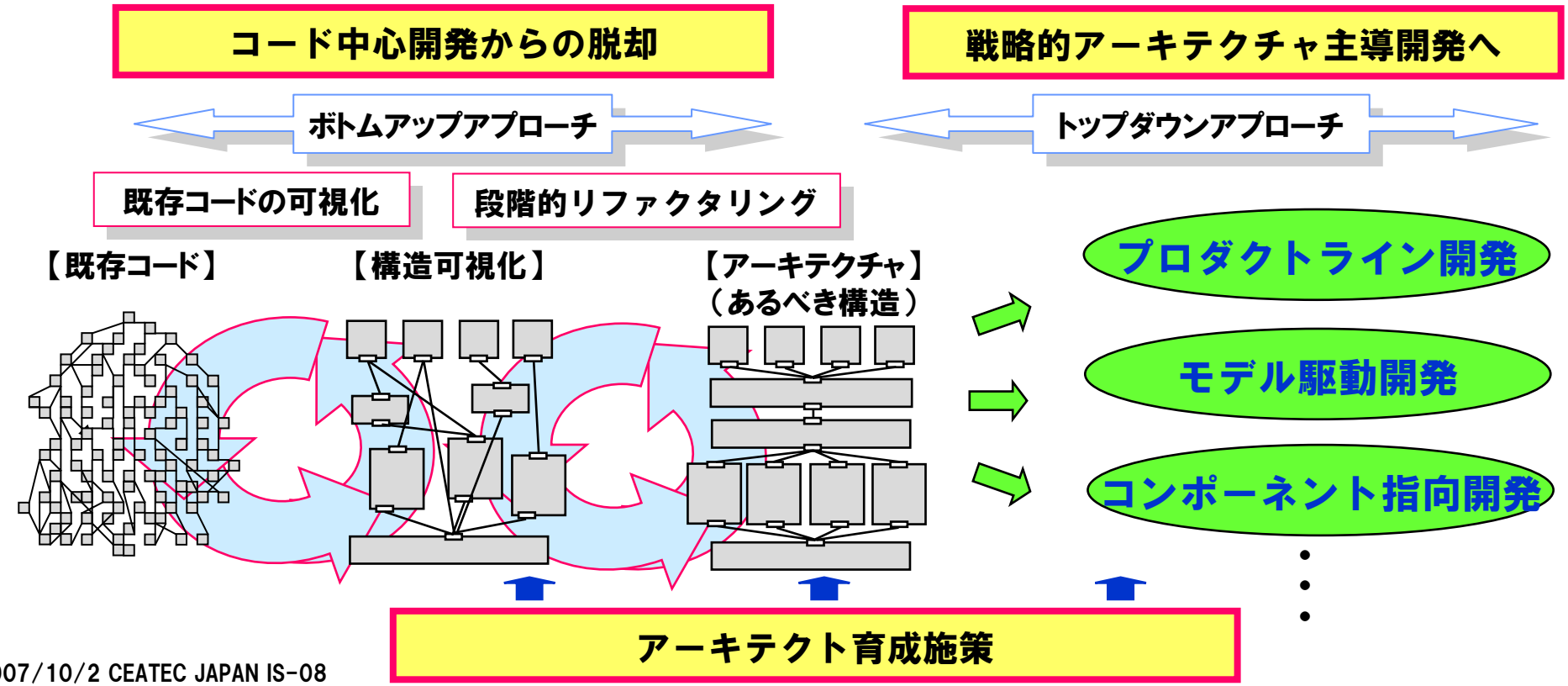
現状認識の重要性

- 開発レベルの認識とレベルに合った処方箋が必要
  - 個人中心開発の状況から，いきなり戦略的な再利用（プロダクトライン）には行けない
  - まだまだ，コード中心開発の現場が多いのではないか？



今後の進むべき方向

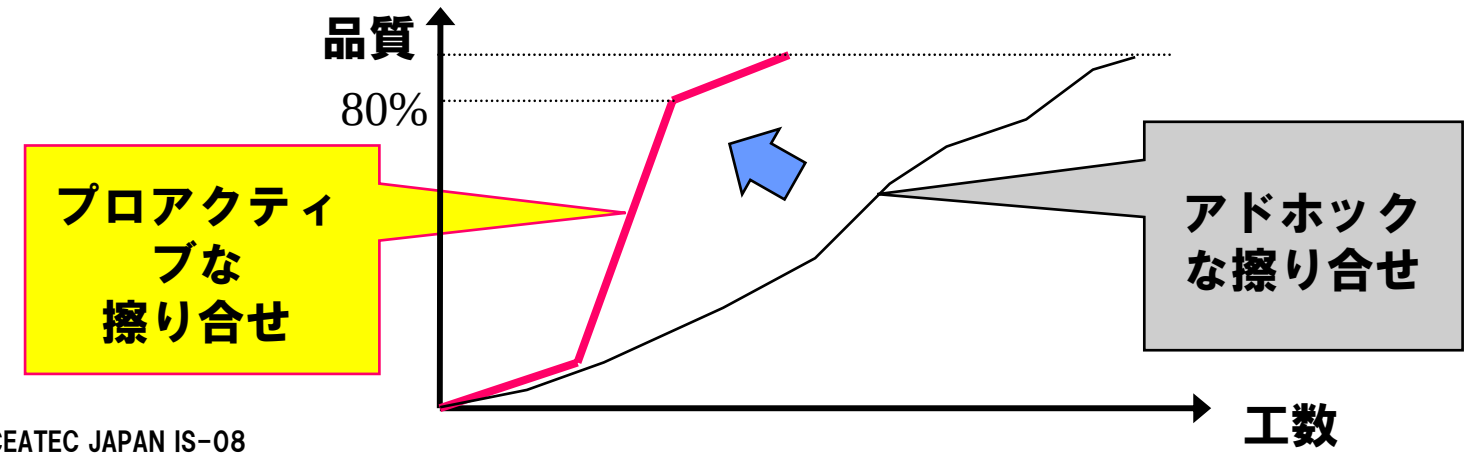
- ステップ１：コード中心開発からの脱却
  - 既存ソフトウェアの資産価値向上，アーキテクト育成
- ステップ２：戦略的アーキテクチャ主導開発へ（PLE，MDDなど）



日本の強み・弱み

- 擦り合せによる高品質開発が日本の競争力の源泉
- しかし，全体が見えない時点からの「アドホックな擦り合わせ」では，大規模化・短納期化に対応できない

- アドホックな擦り合わせから，プロアクティブな擦り合わせへ
  - 組み合わせ（設計・アーキテクチャ力）の補完により，8割まですぐにできるようにする
  - 残りの2割をすり合わせる



今後のJEITAの活動

■ 日本の強みをいかしたベストプラクティスの調査を実施

	施策提案	内容
1	既存コードの資産化プロセスの調査	組込み機器の既存コードを多く抱えているのが日本の特徴である．その強みを活かす意味でも，既存コードを有効活用するプロセスが大切である．既存コードを資産化そして再利用するための設計手法として，コードの評価指標や，改善のためのリバース方法などの，実際の取り組みを調査し，形式化し，水平展開する仕組みが求められる．
2	現場主導アプローチの調査	欧米発のトップダウンアプローチの採用は，日本の開発現場とのギャップが大きく，現実的ではない．段階的なボトムアップアプローチを採用して，モデル駆動開発や再利用・プロダクトラインへ展開している事例を調査し，その成功（あるいは失敗）の本質を明確にしたい．
3	設計力の向上	日本が得意なすり合わせを補完する技術として，全体を俯瞰する設計力が必要である．既存資産を活かしつつ，次に向けてのアーキテクチャの仕込みと展開を行う役割を担うアーキテクトの育成が急務である．アーキテクト育成の事例を調査し，その仕組みを解明したい．