2008年度JEITAソフトウェア事業委員会セミナー

ソフトウェア事業基盤専門委員会報告

組込み系ソフトウェア開発の課題分析と提言

～大規模化、複雑化、短納期化、多機種化の波にどのように立ち向かうべきか～

2008年7月10日

社団法人電子情報技術産業協会
ソフトウェア事業基盤専門委員会
専門委員長　福嶋 慎一

1．組込み系ソフトウェア開発に関する問題意識

- 「組込み系ソフトウェアは
  我が国の強みの源泉であり価値創出のキー」と言われているが、
  組込み対象となるハードウェア機器は強いとしても、

  ソフトウェア開発力が国際的に見ても本当に強いのであろうか?

- 「擦り合わせ」なるものが日本の開発力の強みと言われているが、
  急激に増大している開発規模や短納期化の中で、

  現在でも「擦り合わせ」が強みになっているのであろうか?

- **何を強くすれば、**
  我が国の組込み系ソフトウェア開発力の国際競争力を強化し、

  真に「我が国の強みの源泉」たりうるものにできるのであろうか?

本専門委員会参加企業

インタフェース、沖ソフトウェア、セイコーエプソン、東芝、日本電気、
日立製作所、富士ゼロックス、三菱電機、松下電器産業、リコー

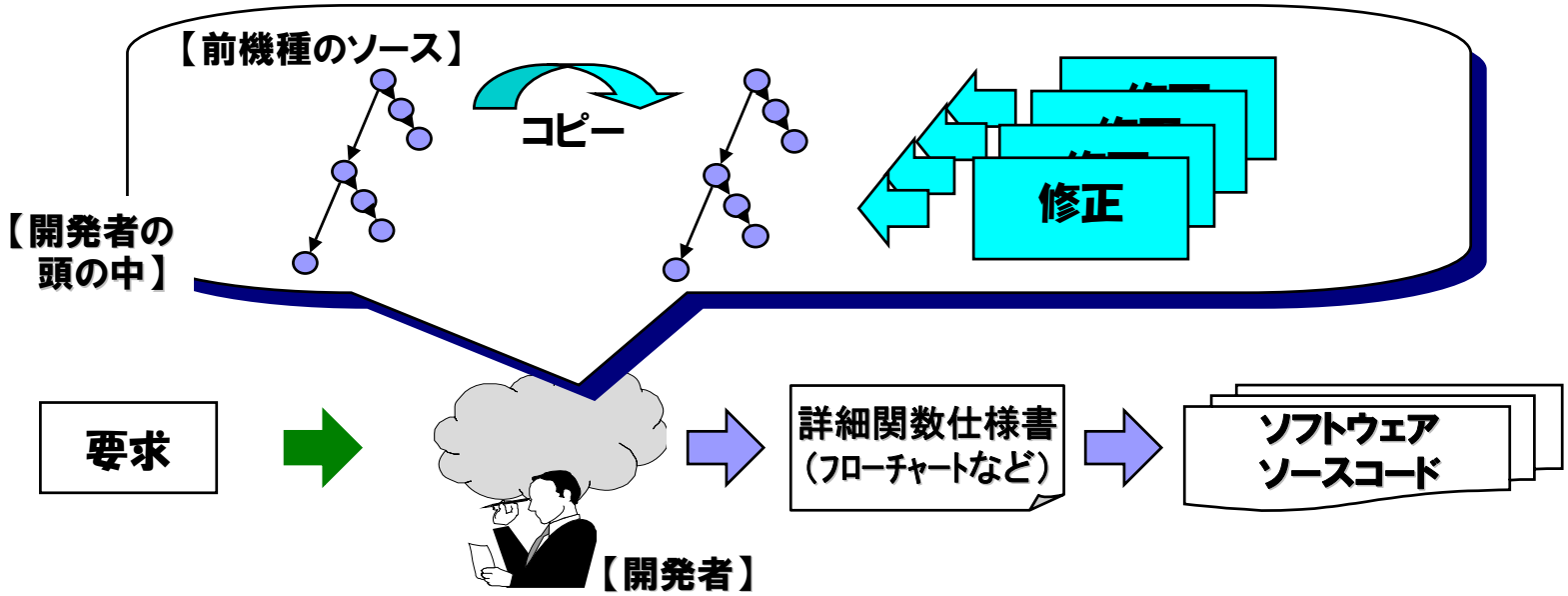
2. 開発現場の現状 と 現場から見た課題

2.1 コード中心の開発

■ コードを作りながら全体構造を決めていく開発スタイル

- 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）
- 小規模時代の開発のなごり、短納期開発のプレッシャ

◇ 属人的な開発：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
◇ 場当たり的な修正によるコードの複雑化
◇ 開発する機種数の増加、担当者の変更により、急激に開発効率が低下



2.2 有効に機能しない再利用、不明確な全体構造

■ 生産性向上には再利用が効果的であるはずだが、
再利用が有効に機能していない
・・・流用率は高いが、生産性は思ったほど向上していない

- 全体構造が不明確なまま、開発が進行
- **全体を俯瞰できる仕組みが欠如**

◇ **修正箇所特定のためにコード解析**
◇ 修正による影響範囲が把握できないために
全体再テストと改造の繰り返し
◇ 作ってからの再利用（**アドホックな再利用**）が**限界に**

2.3 全体構造が把握されないまま進行する開発

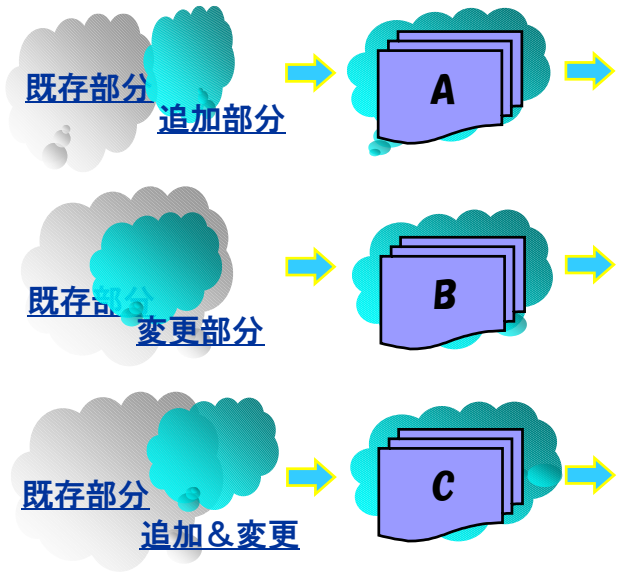
■ 分担だけは決まっているが、全体把握ができていない

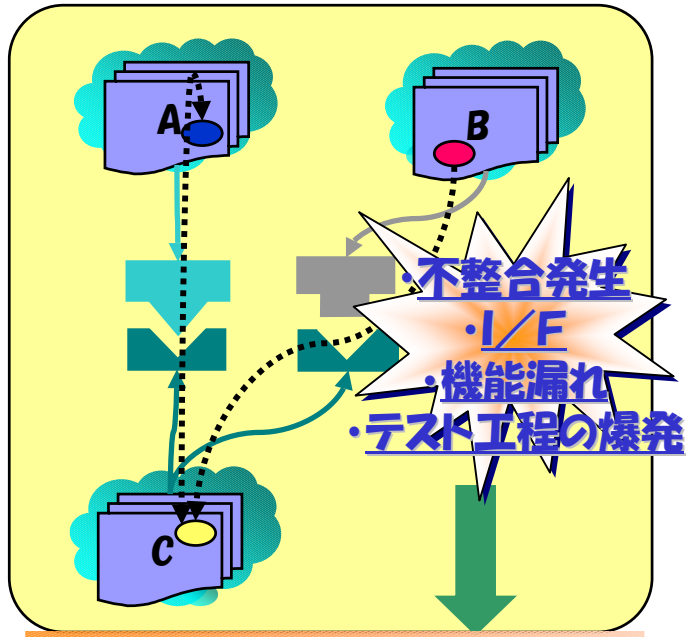
要求仕様 → 設計 → 実装 → 結合・システムテスト

あいまいな要求

全体構造と
担当間のインタフェースが
事前に決まっていない

差分開発





◇ 分担間の仕様調整に
時間がかかる（n 対 m）
◇ 曖昧な仕様を基に、分担開発
が進行（見切り発車）

◇ システムテスト工程で
不整合多発!!

2.4 有効な施策と現実のギャップ

■ 開発の大規模化、多機種開発への対処として、本来ならば、
再利用が有効なはずだが、下流工程での擦り合わせ開発が横行

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徹夜で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い
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

3．課題解決に向けての本専門委員会の取組み

- 2005年度：“足元を知る”
- 2006年度：“品質確保”問題に集中
- 2007年度：“効果的な取組み”の実態把握

3.1 2005年度：実態・課題の把握、足元を知る

- "**足元の確認**"：
  - **日本の開発現場が抱える問題点、課題、今後の方向性の把握と分析**
  - 組込み系ソフトウェア開発に関するアンケート調査
    （JEITA参加企業：30社、70プロジェクト）
    - **品質確保、外部委託活用、OSS利用**

- **アンケート調査結果**：
  - **開発規模の増大**が品質に大きな影響を与えている
  - **短納期化**が進行している
  - **開発要員も増大傾向**。開発組織のマネージメントの改善要望が強い
  - テスト工程での品質確保が困難。**上流工程での品質確保施策が必要**
  - 小規模であった頃の体質を引きり、**全体を統括する仕組みが弱い**
  - **外部委託**：**9割強のプロジェクトで採用**、社内工数が不足する実態
  - **OSS利用**：その**期待は高いが、課題も多い**

3.2 2006年度:“品質確保”問題に集中(1)

外を知る：欧州との情報交流

- ドイツ Fraunhofer IESE研究所、Fraunhofer工科大学と情報交流(2006年6月):
  - ソフトウェアの品質劣化要因:
    - 大規模化 (Cost)
    - 短納期化 (Time)
    - 複雑化 (Complexity)
    - 多機種化 (Variety)
  - ソフトウェアアーキテクチャの重要性
  - 統一された品質管理/計測システム導入の重要性
  - 外部委託に関する“スマートグローバリゼーション”の考え方
  - OSSの信頼性の問題

組込み系ソフトウェア開発
に同時に押し寄せる
4つの大波!!

3.2 2006年度：“品質確保”問題に集中 (2)

- “品質確保の問題”に焦点を絞ってアンケート調査（59プロジェクト）：
  - 品質施策（品質の計測と管理の仕組み）：
    - 下流工程重視から上流工程重視へ移行しつつある
    - 水平展開などを始めとする多機種開発に対する施策が弱い
  - 品質プロセス（品質確保のための開発プロセス）：
    - 大規模化 に対して：
      - 開発体制や開発プロセス/手法の整備が追いついていない
      - 設計の文書化/モデル化を行い、上流工程重視の方向にある
    - 多機種化 に対して：
      - ウォーターフォールプロセスの採用が多く、もはや実態に合わないプロセスを、現場開発者が適時、工夫変更しながら開発を続けている
      - プロダクトラインエンジニアリングには期待は大きいが、本格的な導入はこれからという状況
    - システム化（擦り合わせ）に対して：
      - ソフトウェア開発を配慮したハードウェアとソフトウェアの統合開発プロセス・開発方法論が必要となっている

2006年度から2007年度の活動へ

組込み系ソフトウェア開発の現場は・・・

・大規模化
・複雑化
・短納期化
・多機種開発化
(複数機種並行開発)

このような多重の困難の中で・・・

開発現場は
品質確保の課題
に取り組んでいる

この現状と今後を
調査・分析、
課題解決への提言

・「品質施策」の観点
・「品質プロセス」の観点

■「品質施策」に関する提言(提案)

・コストと効果を考慮した品質施策を実施せよ
・品質施策の重点は、上流工程へシフトせよ
・多機種開発を前提としたツールを利用せよ
・品質関連の手法等は外部委託先と共有せよ

■「品質プロセス」に関する提言(提案)

・大規模化に適した開発プロセス事例を収集し、ベストプラクティスの情報共有を図るべし
・多機種開発を考慮した工学的な開発プロセスを研究せよ
・日本の強みと弱みを意識した擦り合わせ技術を模索せよ
・アーキテクトの育成を急げ

2006年度の活動

2年間の調査・分析活動の纏め

2007年度の活動

課題解決に向けた提言(提案)を具体化している
各社の取組み・施策を収集・分析する

課題解決に
直結する分野を
対象に
具体的な取組み・施策を
アンケート調査

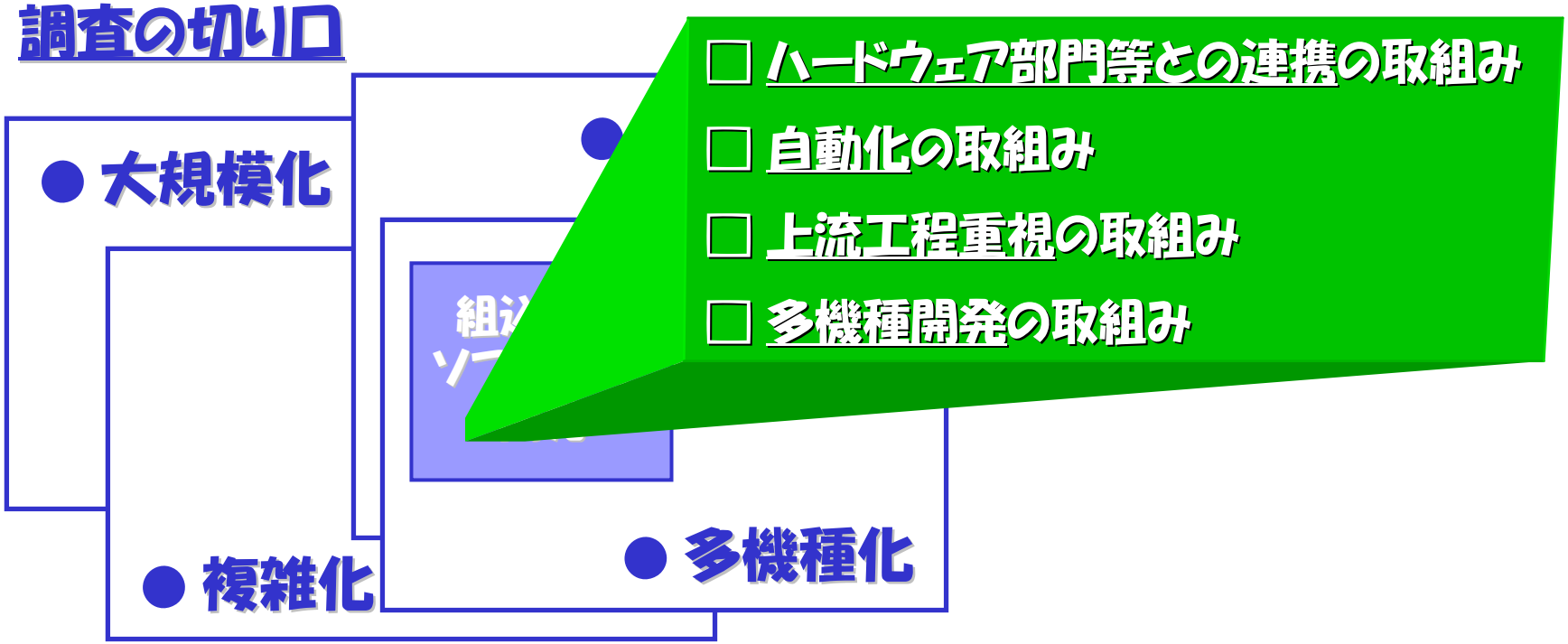
- 「ハード部門等との連携」
- 「自動化」
- 「上流工程重視」
- 「多機種開発」

効果的な取組み・施策を
組込み系ソフトウェア業界の
関係者で情報共有!!

組込み系ソフトウェア
業界の発展に寄与

3.3 2007年度：“効果的な取組み”の実態把握（1）

- 大規模化、複雑化、短納期化、多機種化の波に立ち向かう具体的な取組み・施策について、アンケート調査を実施
- アンケート調査対象企業を拡大：57社、69プロジェクト
  - 関西経済連合会「組込みソフト産業推進会議」
  - JEITA参加企業
- 調査の切り口



3.3 2007年度：“効果的な取組み”の実態把握 (2)

2007年度アンケート調査結果

- **品質低下の要因**
  - 「最も重要視する要因」：
    トップは「短納期化」

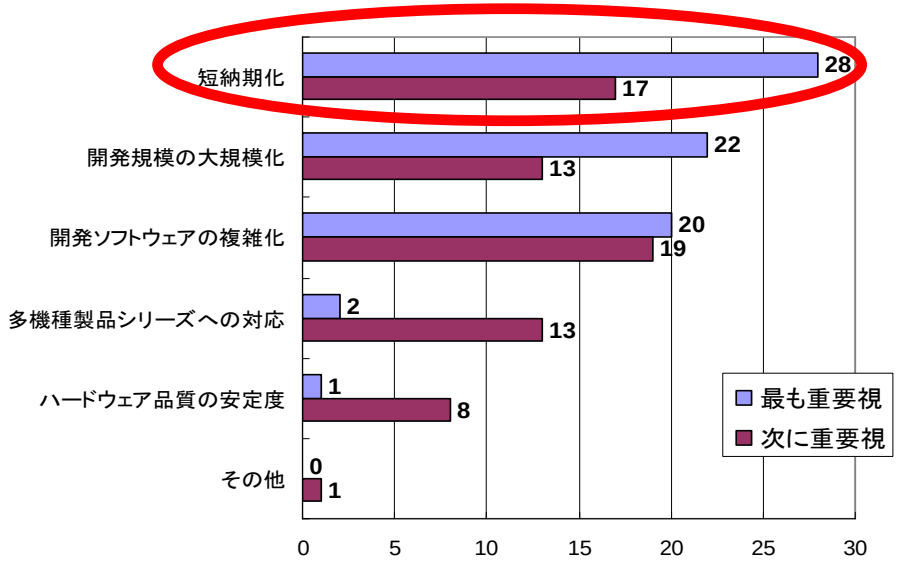


図5.4-2 品質に影響を与えている主な要因

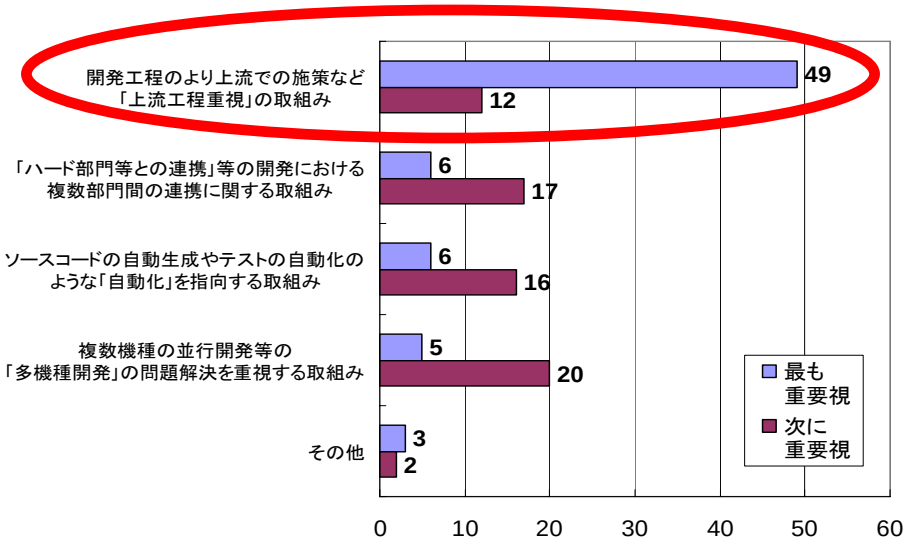


図5.4-3 品質の要因の解決策として、重要視する取組み・施策

- **重視する取組み・施策**
  - 最も重要視する取組み：
    トップは
    「上流工程重視の取組み」

4. ハードウェア部門等との連携の取組み

- ハードウェア部門等との共同作業の標準化
- 要求と仕様定義における相互連携
- 上流工程での評価連携（シミュレーションの活用）

4.1「ハードウェア部門等との連携の取組み」の状況

- **ハードウェア部門等との共同作業の標準化**
  - 統合化された開発プロセス標準が存在するプロジェクトは非常に少ない
  - ハード・ソフト個別の開発プロセスをゲートモデルによって整合と同期
  - 共同作業の工程としては、広く全工程にわたっている
- **要求と仕様定義における相互連携**
  - システム分析・設計手法が存在しない（79%）
    進捗や障害、構成にかかわる管理手法がない（42%）
  - 両分野に跨る障害解決の解析活動や役割分担については、
    現実的に工夫されたプロセスや手法が存在する
- **上流工程での評価連携（シミュレーションの活用）**
  - システム全体をシミュレートしている例はまだ少ない
  - 機械CADのデータを使ったシミュレーション環境の構築など、
    可能な限り実機の環境に近づける取組みが行われている
  - ハードウェアデバイスの変動分を開発プロセスや設計手法の中に織り込み、
    これを様々なソフトウェア設計手法により吸収するといった開発スタイルが定着

4.2 ハードウェア部門等との共同作業の標準化（1）

- **統合化された開発プロセスの存在**
  - ハードウェア開発とソフトウェア開発を統合して**1つの開発プロセス標準として存在するプロジェクトは、わずか6％**

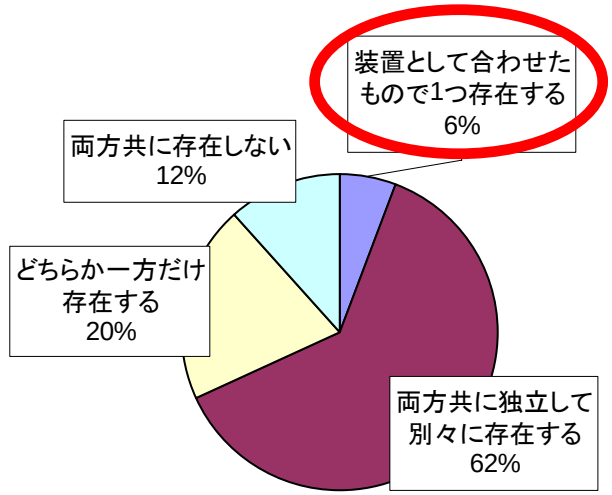


図6.2－1ハードウェアとソフトウェアの開発プロセス標準化の割合

- **共同作業のための開発プロセスモデル**
  - 開発プロセスとしては**ゲートモデルの採用が圧倒的**
  - ゲートにおいて**合同レビュー等によって仕様の整合**を図り、**開発同期**を取っている

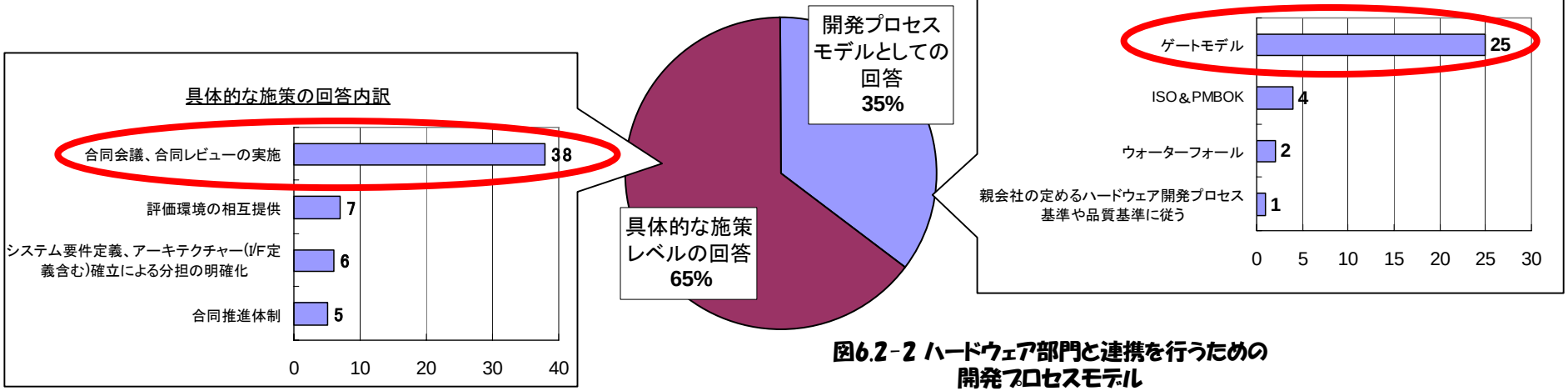


図6.2－2 ハードウェア部門と連携を行うための開発プロセスモデル

4.2 ハードウェア部門等との共同作業の標準化(2)

- **共同作業の対象工程**
  - 要求共同作業の工程としては、
    システム全体に関わる工程に重点はあるものの、
    広く全工程にわたっている

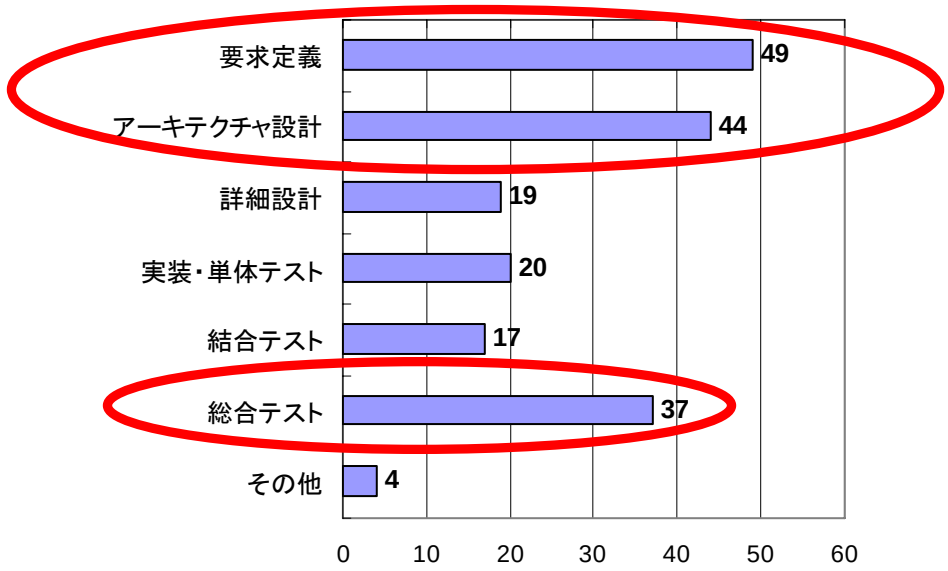


図6.2-4 H/W, S/Wなどの異分野のチームによる
共同作業が計画されている工程

4.3 要求と仕様定義における相互連携

- **システム分析・設計手法**
  - 要求性能を満たすシステム分析・設計手法が未確立（79%）
  - 進捗や障害、構成にかかわる管理手法がない（42%）

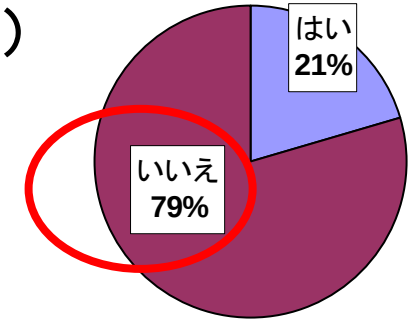


図6.3-2 要求性能を満たすシステム分析と設計手法

- **障害解決の解析活動や役割分担**
  - 両分野に跨る障害解決の解析活動や役割分担については組織と製品の特性に適合するよう現実的に工夫されたプロセスや手法が存在する

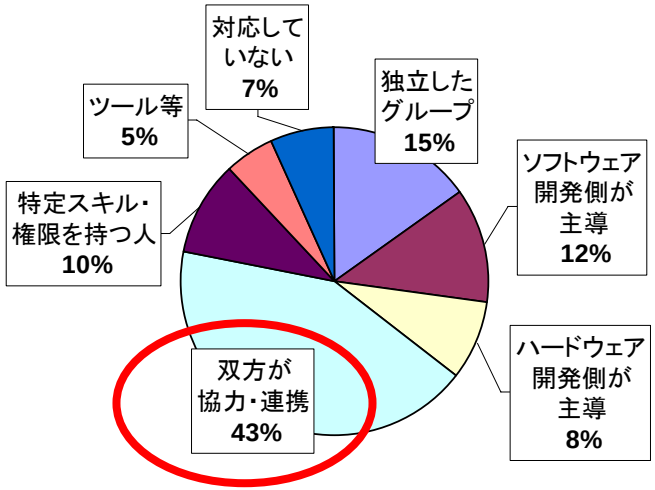


図6.3-6 開発途上に生ずる様々な逸脱事項、特にハードウェア、ソフトウェアの両方に跨る問題への対応

4.4 上流工程での評価連携(シミュレーションの活用)

- **ハードウェア変動分のシミュレーション**
  - システム全体をシミュレートしている例は少ない
  - 機械CADのデータを使ったシミュレーション環境の構築など、可能な限り実機の環境に近づける取組みが報告されている
  - ハードウェアデバイスの変動分を開発プロセスや設計手法の中に織り込み、これを様々なソフトウェア設計手法により吸収・局所化するといった開発スタイルが定着している

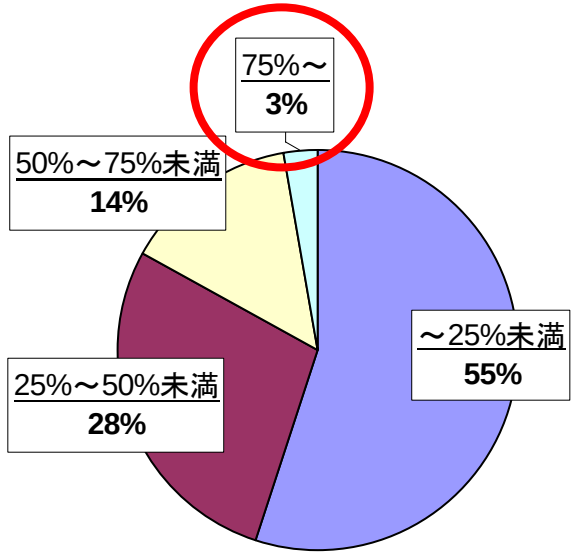


図6.4-1 実機を使わずにシミュレータで検証できる範囲

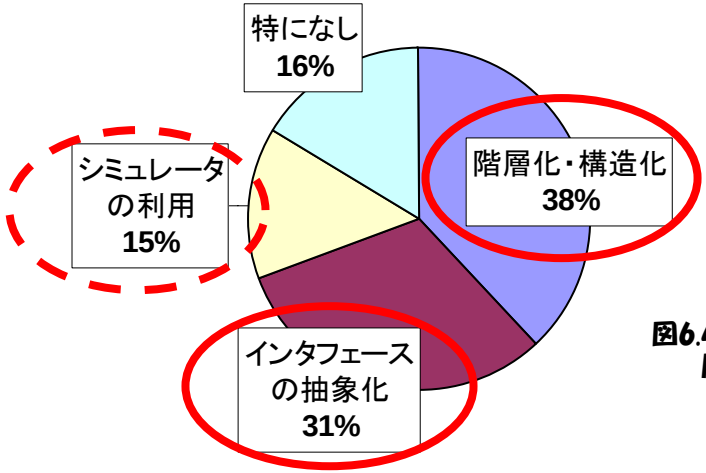


図6.4-4 ハードウェアの影響を局所化する工夫, 手法

4.5「ハードウェア部門等との連携の取組み」に関する提言

- **ソフトウェア・ハードウェアの統一的な開発プロセス標準の整備**
  - ハードウェア開発とソフトウェア開発を包含した統一的な開発プロセス標準を整備する
  - アドホックな擦り合わせ開発でなく、プロアクティブな擦り合わせ開発を目指す
- **作業者の開発技術に関するスキルの底上げ**
  - 作業者によるバラツキを抑えることは、プロセスや手法の整備だけでは限界。作業者の全体的なスキルの底上げを図る
- **ステークホルダー間の合意と文書化**
  - アーキテクトによるシステム分析・設計を実施できる体制の構築
  - 文書化によるステークホルダー間の合意形成を推進する
- **シミュレーションの適用工程の上流工程への拡張**
- **変動部に対する品質設計**
  - 「変動部の共同管理」の下で、「変動部不具合の未然防止」、「固定部への影響のコントロール」といったプロアクティブな品質設計を目指す

5. 自動化の取組み

- ソースコードの自動生成
- テストの自動化
- その他の自動化

5.1 「自動化の取組み」の状況

開発の大規模化
短納期化
多機種化
→ ツールによる自動化

- ツールの分類
  - ソースコードの自動生成
  - テストの自動化
  - その他の自動化
- 調査観点
  - 自動化の効果・適用範囲
  - 自動化のコスト効果計測
  - 自動化の課題と解決方法

- 取組み状況
  - 構成管理ツールや下流工程のテストツールの使用は当たり前
  - 設計ツールなど上流工程のツールの使用はまだ少ない
  - 生産性の向上よりも、大規模化に伴う様々なメンバーによる開発品質を一定水準に保つ目的が多い
  - 効果や適用範囲の定量化の意識は高いが、実際に適用する困難さが浮かび上がっている
  - ツールの適用範囲の狭さも課題

5.2 ソースコードの自動生成 (1)

- **ツールの使用状況**
  - 半数以上が使っていない
  - クラス図からのコード生成ツールが多い
  - 種々の自動化手法が導入

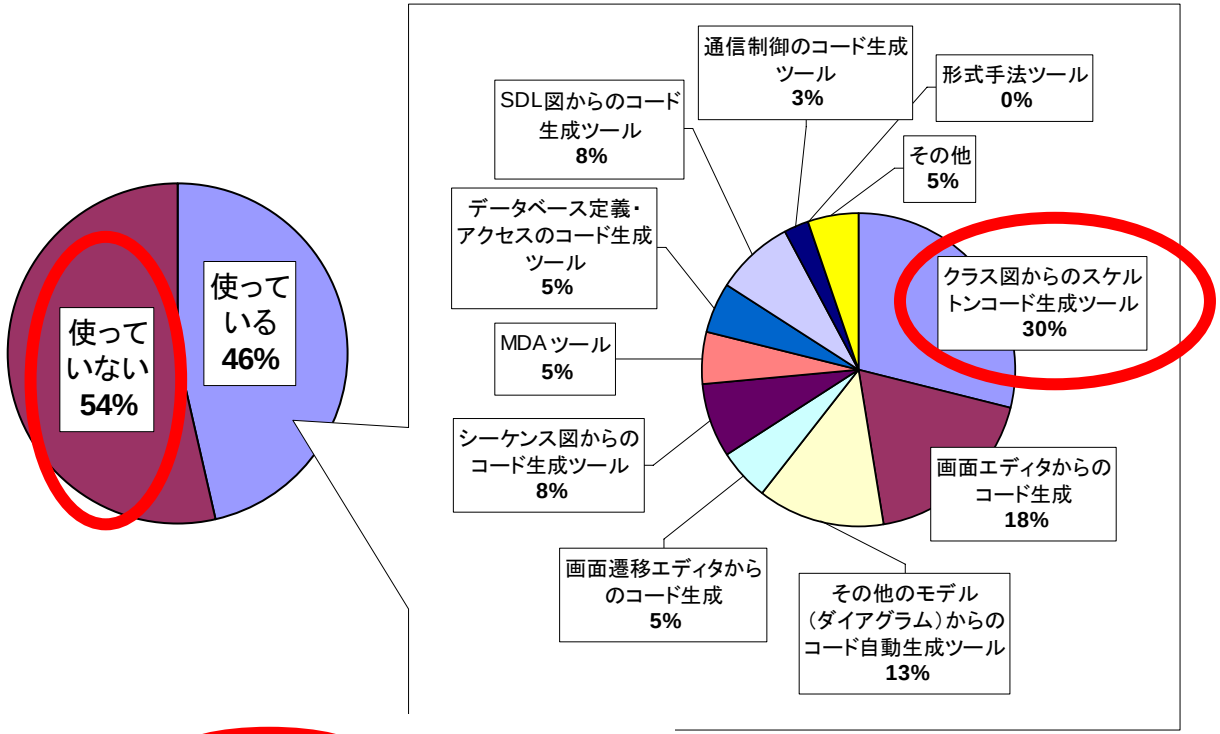


図7.2-1 ソースコードの自動生成ツールとして、どのような種類のツールを使っていますか

- **使用ツールの種類**
  - 様々なツールを使用
  - 自作ツールが多い
  - 標準のツールはない

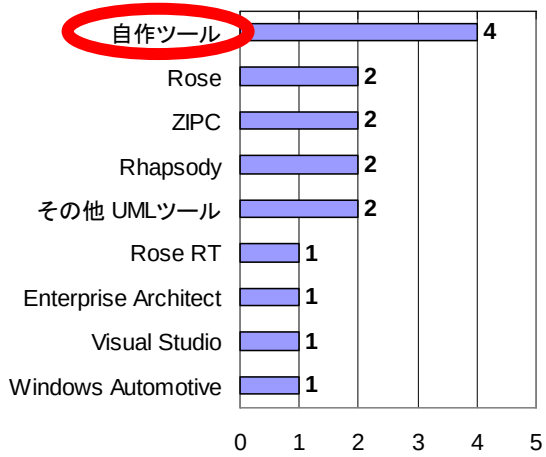


図7.2-2 ソースコードの自動生成ツールとして、どのようなツールを使っていますか

5.2 ソースコードの自動生成 (2)

- **ツールの適用範囲**
  - 適用範囲は25%未満
  - 限られた範囲で使用
  - 算出基準は「ステップ数・開発規模」

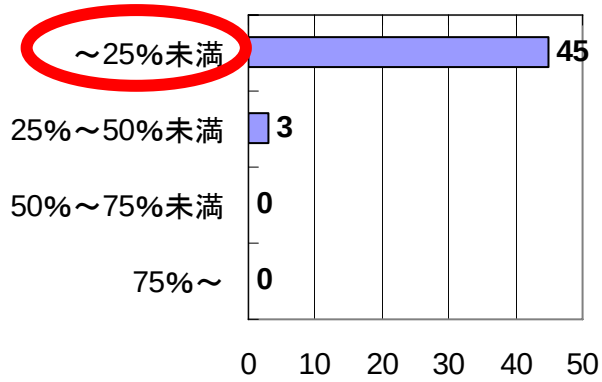


図7.2-2 自動生成ツールの適用範囲は全体のどの程度でしょうか

- **導入における課題**
  - 性能／信頼性
  - スキル／教育期間

- **課題の解決方法**
  - 学習、ガイドライン、調査
  - 適用範囲の狭さは未解決（ツールベンダ任せ）

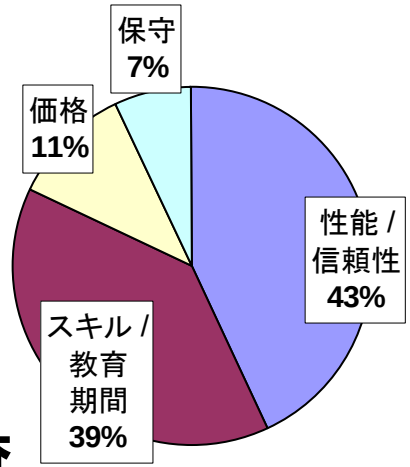


図7.2-6 自動生成ツールの導入にあたっての課題は何でしょうか

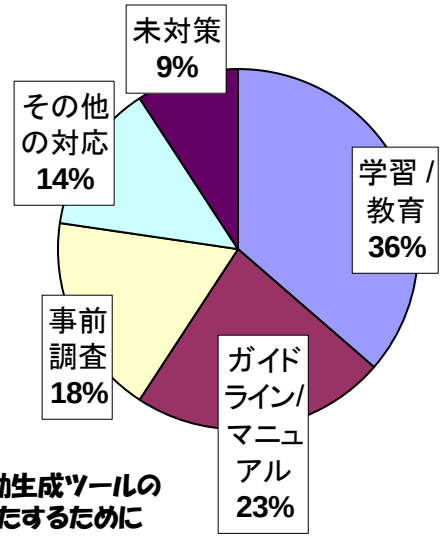
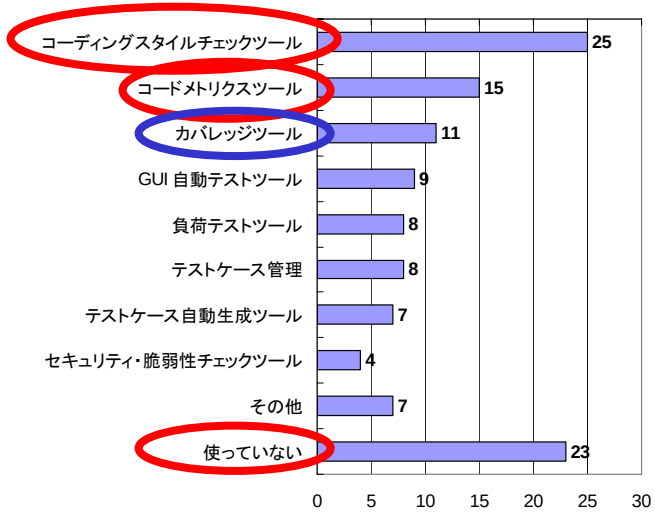


図7.2-7 自動生成ツールの課題を解決するためにどのような対応をされていますか

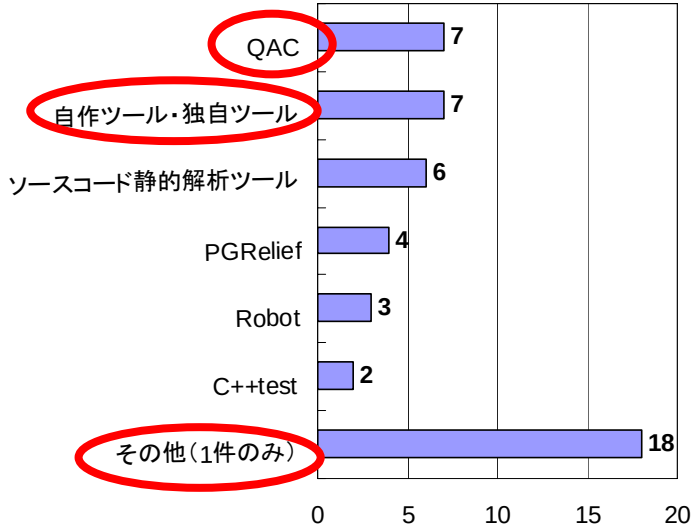
5.3 テストの自動化（1）

- **ツールの使用状況**
  - ほとんどのプロジェクトで使用
    しかし20％で使っていない
  - コーディングスタイルチェック／
    コードメトリクスツールが多い
  - テストカバレッジツールは少ない



- **使用ツールの種類**
  - QAC が多い
  - 自作ツールも多い
  - その他のツールも多い
  - プロジェクトでばらばら

  組込み系の特徴！



5.3 テストの自動化(2)

ツールの適用範囲

- 適用範囲は25%未満が大半

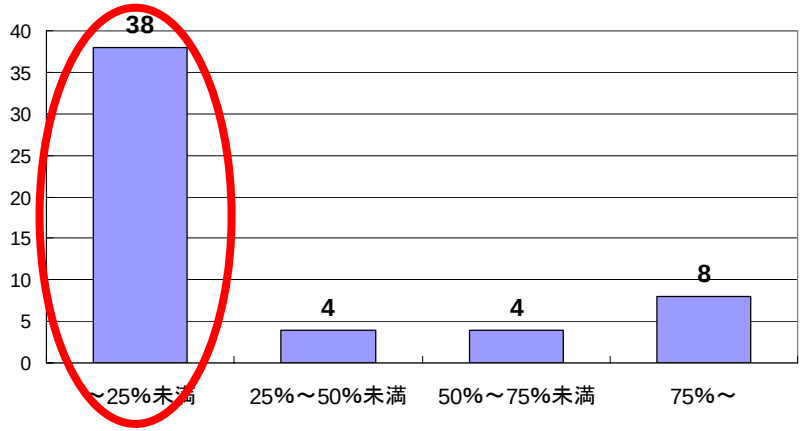


図7.3-3(a) テストの自動化ツールの適用範囲は全体のどの程度でしょうか

ツールの導入効果

定性的な効果

- 自己チェック効率の向上
- バグを事前に発見
- 単純なバグを発見
- コーディング段階でのバグの削減
- テストのドキュメントの作成効率が向上
- 夜間テストの実施
- 典型的なバグの網羅的な検出
- テストカバレッジの把握

定性的な効果(続き)

- 予想外の指摘
- 再現しにくいテストに利用
- 目安すらない時代に比べて格段に向上

欠点

- 生産性の寄与はそれほど大きくない
- ツールを使うためのコストが必要
- 新規開発では自動試験部分の開発コストが大幅増

5.4「自動化の取組み」への提言

- 上流工程への自動化ツール適用の拡大
- 支援体制の構築とガイドラインの策定
- 試行による実績の積み重ね
- 全社施策としての取組みと啓蒙

6. 上流工程重視の取組み

- 要求（要求定義・要求管理）
- 設計（アーキテクチャ設計・設計評価）
- 人材（アーキテクト育成）

6.1「上流工程重視の取組み」の状況

- **要求（要求定義・要求管理）**
  - 下流での手戻りのうち、要求定義の不備が原因であるものが少なくない
  - 要求のインプット、アウトプットは、自然言語で記述した文書が中心
  - 要求定義では、ソフトウェア開発部門が大きな役割
  - 要求定義での留意点：優先順位づけと、背景の明確化
  - 要求変更は多いが、要求管理と変更追跡を支援するツールはまだ不十分
- **設計（アーキテクチャ設計・設計評価）**
  - 大半のプロジェクトでアーキテクチャ設計に課題を認識
  - スキルのある人材が不足していることが指摘されている
  - アーキテクチャ設計を支援するツールの導入は不十分で、評価もレビュー以外の方法がない。設計のプロセスと成果物に関する業界標準がない
- **人材（アーキテクト育成）**
  - アーキテクトには総合的なスキルが要求され、その育成に苦慮している状況
  - 組込み分野でのアーキテクトのスキルなど、必要要件が業界でも明確ではないことが更に育成を困難にしていると思われる

6.2 要求（要求定義・要求管理）(1)

- **要求定義に起因する手戻り**
  - 下流工程で発生する手戻りのうち、**要求定義の不備が原因である割合は、25％未満が多い・・・**

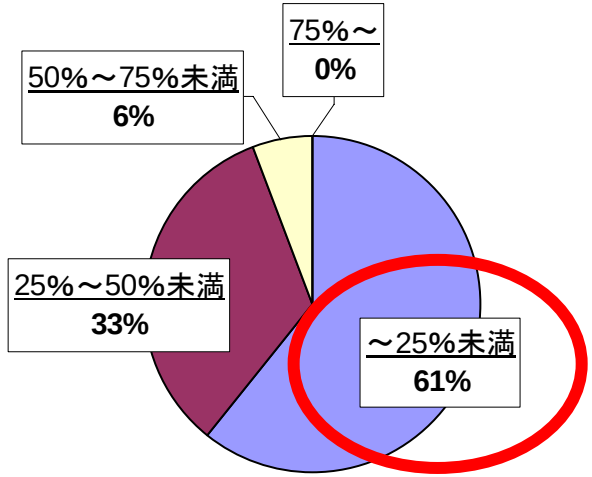


図8.2-1 下流工程で発生する手戻りのうち、要求定義の不備が原因であるものの割合

- **要求定義の表現**
  - 専用ツールは使われていない
  - 自然言語による記述や口頭による場合が多い

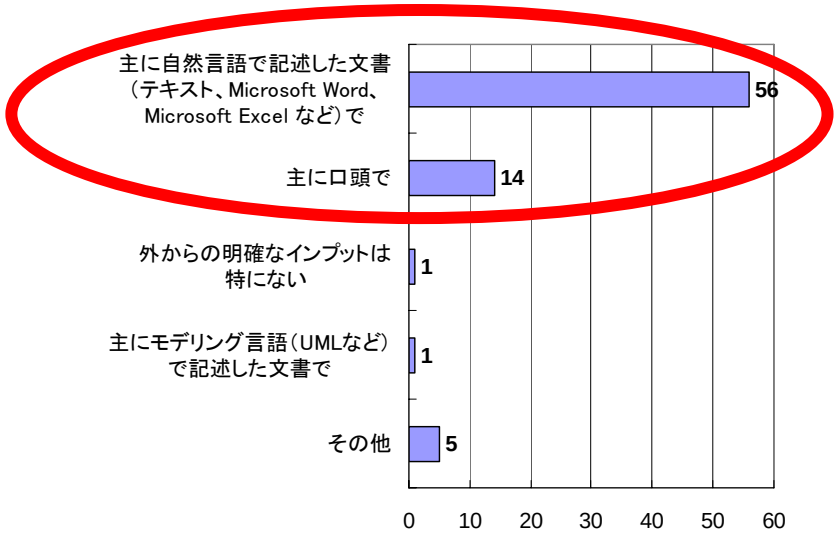


図8.2-3(a) 要求のインプット形式

6.2 要求（要求定義・要求管理）(2)

- **要求変更の発生状況**
  - 要求の変更が多発：30％が毎週、45％が毎月の頻度
  - 要求変更が最終工程に至ってまで発生
    - ほぼ半数がテスト工程まで

- **要求変更の発生原因**
  - 「過剰な機能追加」「技術の未確立」という要求定義段階での検討不足も半数存在

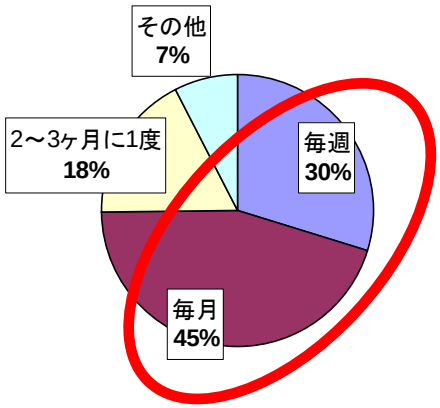


図8.2-6(a) 要求変更の発生頻度

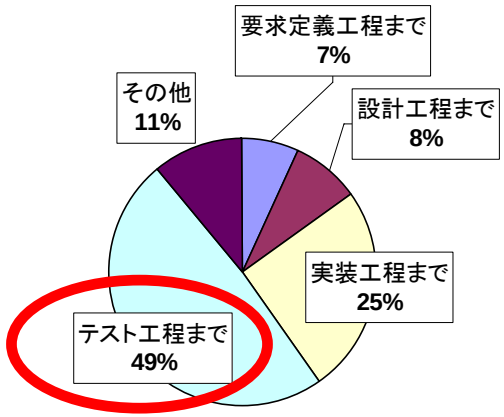


図8.2-6(b) 要求変更の発生工程

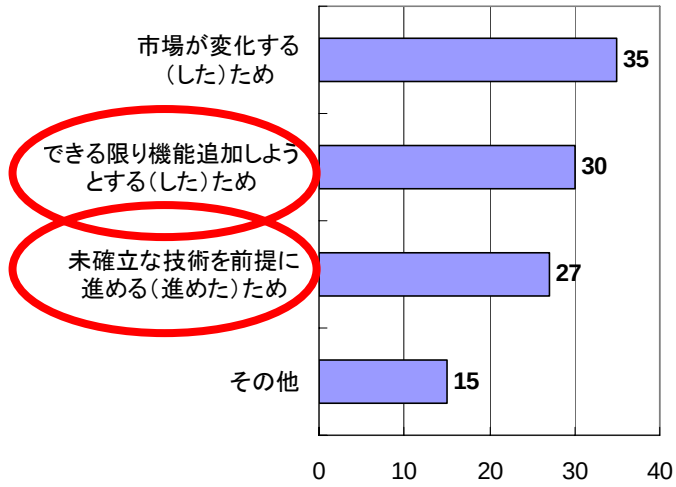


図8.2-7 変更要求の発生原因

6.3 設計（アーキテクチャ設計・設計評価）(1)

- **アーキテクチャ設計の実施状況**
  - アーキテクチャ設計は行われている認識にも関わらず**有効なアーキテクチャ設計が行われていない実態**
    - **アーキテクチャの寿命**は、**3年未満の回答が約4割**
- **アーキテクチャ設計での注力ポイント**
  - **再利用性、拡張性の確保といった非機能要件の実現が注力ポイント**

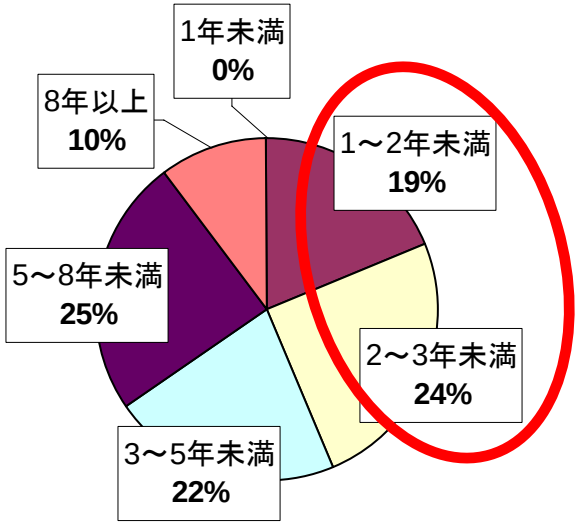


図8.3-2 アーキテクチャの寿命

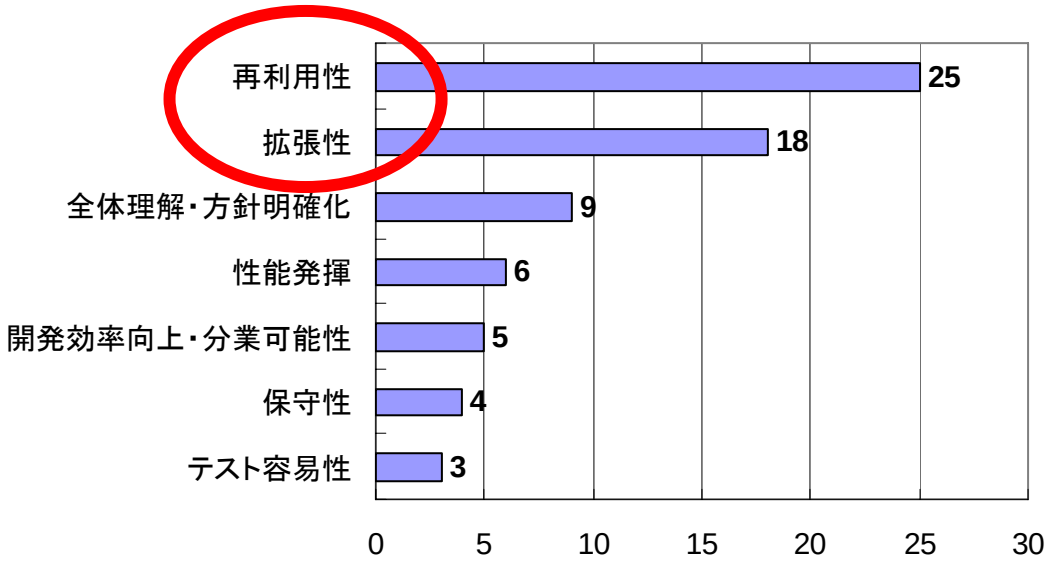


図8.3-6 アーキテクチャ設計での注力ポイント

6.3 設計（アーキテクチャ設計・設計評価）(2)

- **アーキテクチャ設計の実態**
  - アーキテクチャ設計書の定義やアーキテクチャ設計と詳細設計との境界など、**アーキテクチャ設計の具体的な業務内容は、プロジェクトでまちまち**
  - **アーキテクチャ設計とは何をするべきかの標準が定まっていない**

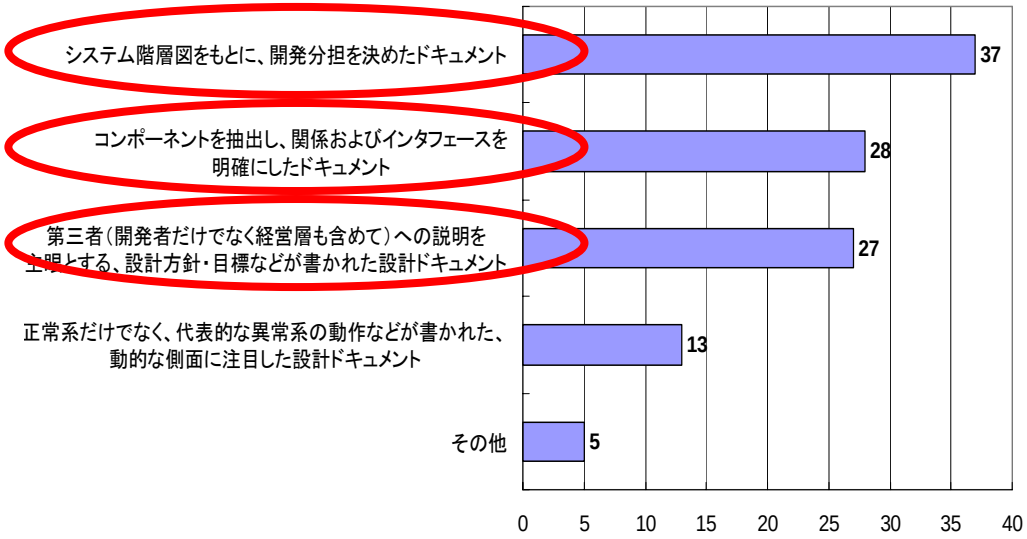


図8.3-7 アーキテクチャ設計書の定義

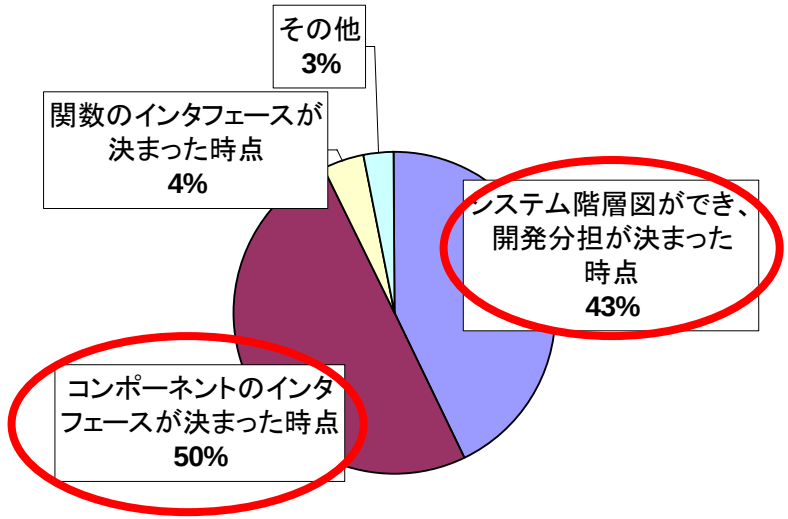


図8.3-8 アーキテクチャ設計と詳細設計との境界

6.4 人材（アーキテクト育成）(1)

- **アーキテクチャ設計の課題**
  - 90％の回答で、アーキテクチャ設計に困難を感じる
  - 最大の要因は、スキルのある人材の不足
- **アーキテクトの育成状況**
  - 84％が 育成の仕組みを持たず模索状況

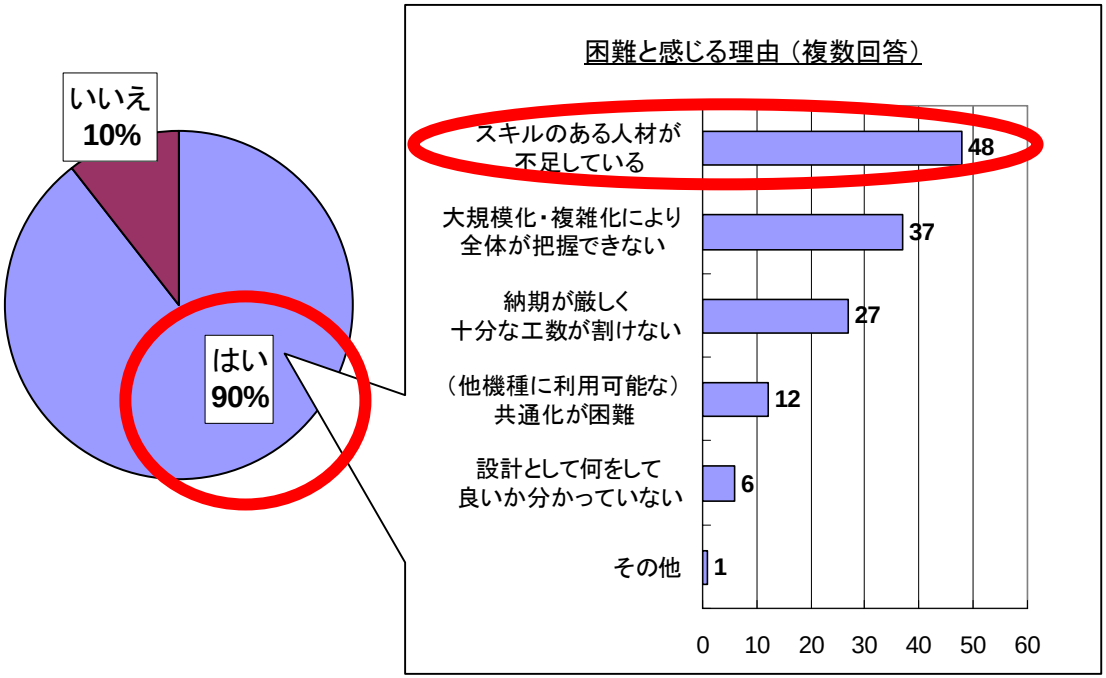


図8.3-3 アーキテクチャ設計の課題

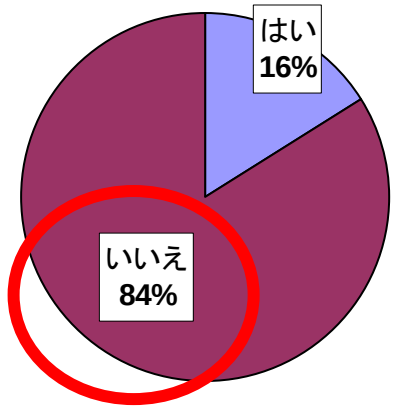


図8.4-1 アーキテクトを育成する仕組みの有無

6.4 人材（アーキテクト育成）(2)

- **アーキテクトに必要な能力の認識**
  - 技術とビジネスの両面に精通したリーダーシップを発揮できる人材
    - 最も必要であるとされたのは「抽象化能力」
    - 「高度な技術力」「モデリング技術」も同等程度に重要視
    - 「（将来に向けた）ビジョンの構想力」という回答も多く、育成の困難さを物語る

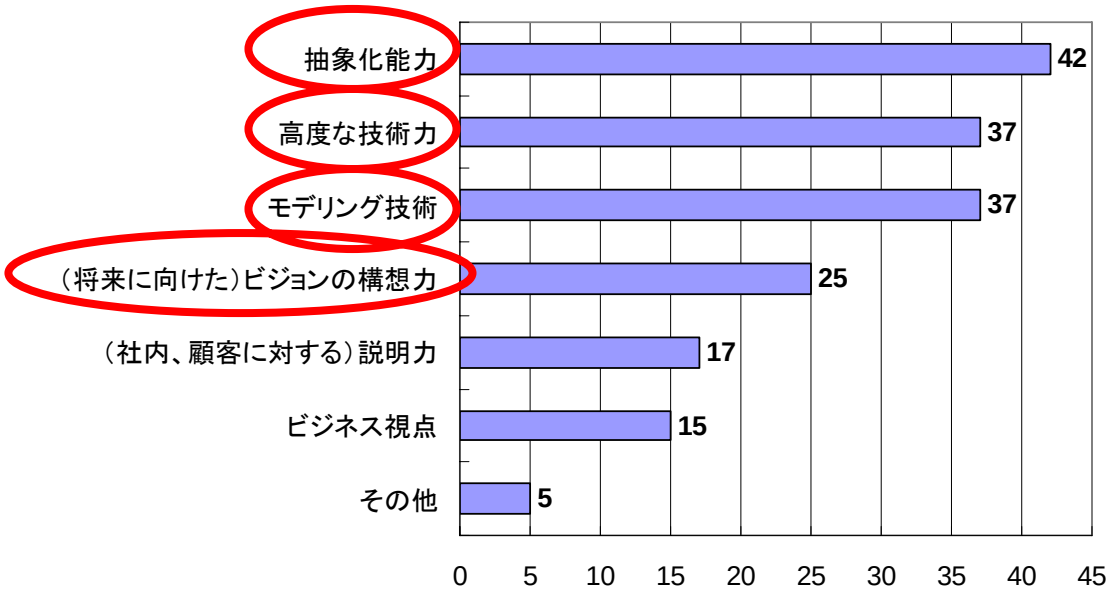


図8.4-2 アーキテクトに必要なスキル

6.5「上流工程重視の取組み」への提言

- **要求定義プロセスの効率化**
  - 要求記述ツールを使って、曖昧性を排除、変更管理・追跡を効率化
  - 顧客要望を適切に取り込み、機能を絞り込む（作り過ぎない）ことも必要
  - アーキテクチャ設計との連携による変更対応性の向上
    - 要求変更は不可避であり、機能の追加・削除が容易なコンポーネント設計を行うことが重要
- **設計プロセスと成果物の標準化**
  - 業界あげてのアーキテクチャ設計業務の標準化が必要
    - 設計書のテンプレートなど成果物の標準化
    - 設計評価の手法・設計ツールの評価・普及 など
- **アーキテクトの育成加速**
  - 企業の枠を超え、産官学が連携したアーキテクト育成施策の実施
    - 組込み系向けアーキテクトの人材モデルとスキルセットの明確化
    - カリキュラムと適切な教材の提供 など

7. 多機種開発の取組み

- 設計（機種共通部分の設計・
  共通部分を再利用した機種設計）
- マネージメント
  （商品戦略・開発体制・開発プロセス）
- ソフトウェアプロダクトライン

7.1 「多機種開発の取組み」の状況

- **設計（機種共通部分の設計・共通部分を再利用した機種設計）**
  - アーキテクチャ設計の重要性は、6割のプロジェクトで認識されている
    しかし、従来どおりのコード中心の開発形態も依然多い
  - トップダウンの設計を行っている回答も多いものの、変動点管理の方法は、
    ソースコード中心の対応(条件コンパイル等)も多い
  - アーキテクチャの再構築の必要性は認めるものの、人材・時間の
    制約により行われていない(84％)
- **マネージメント（商品戦略・開発体制・開発プロセス）**
  - 多機種開発を商品戦略と一体化させる状況には至っていない
- **ソフトウェアプロダクトライン**
  - 関心は高く、取組みを始めているプロジェクトも増加している
  - 従来からの取組みをそのまま当てはめているだけの場合がある
  - 再利用を戦略的に管理された状態で実施するというソフトウェア
    プロダクトラインの本来の目的に沿った活動には至っていない

7.2 設計（機種共通部分の設計・共通部分を再利用した機種設計）（1）

- **アーキテクチャ設計の重要性の認識**
  - **重要性は浸透しているものの、**
    アーキテクチャ中心に開発しているプロジェクトは、半数

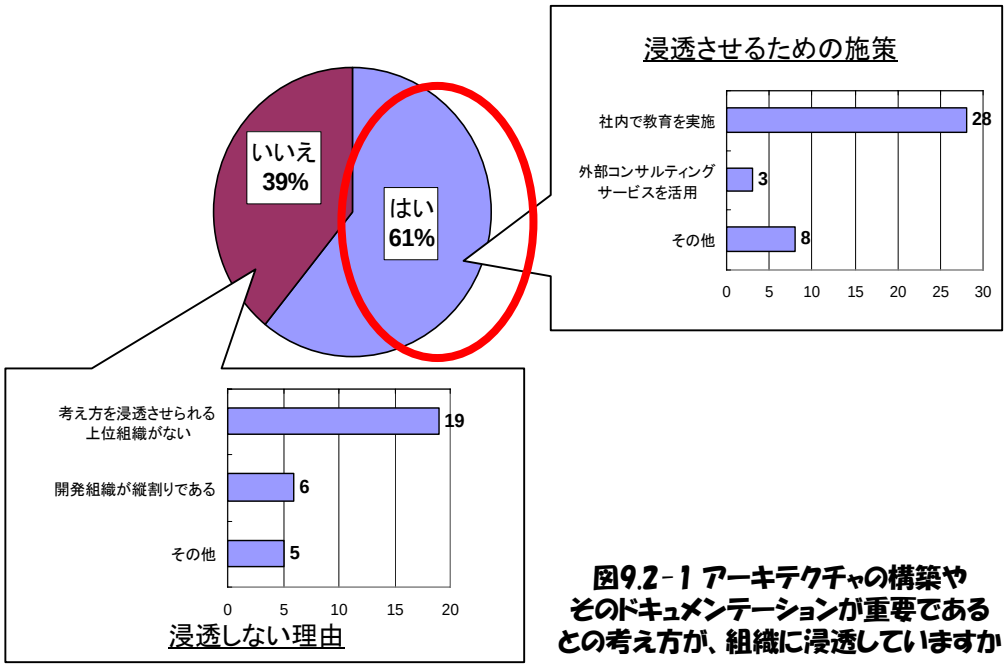


図9.2-1 アーキテクチャの構築やそのドキュメンテーションが重要であるとの考え方が、組織に浸透していますか

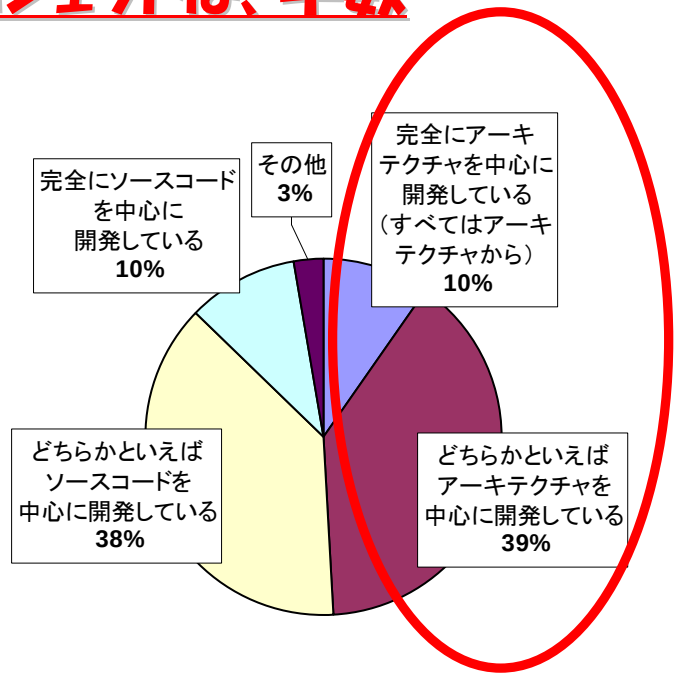


図9.2-2 多機種開発における、アーキテクチャの位置付けはどのようなものですか

7.2 設計（機種共通部分の設計・共通部分を再利用した機種設計）(2)

- **変動点管理**
  - **ソースコードレベルでの管理（条件コンパイルと構成管理ツール）が、半数超**
  - **機種が増えた場合、ソースコードに条件コンパイルを付加するという コードレベルでの対処が最も多い**

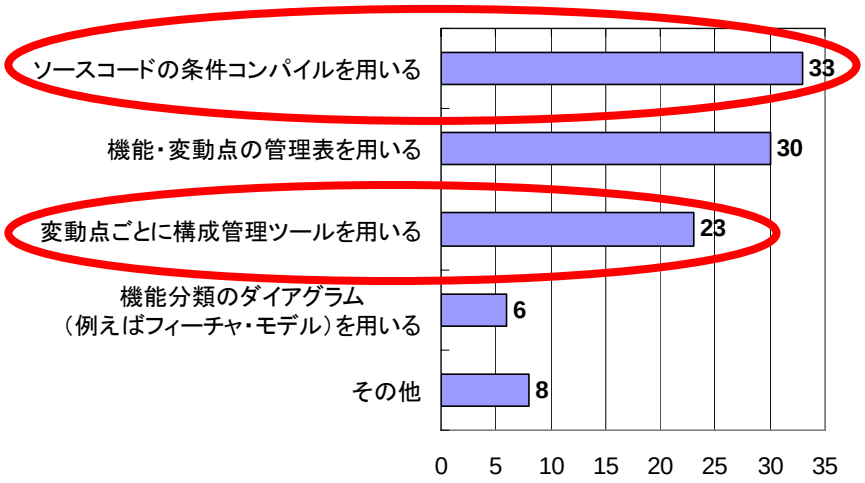


図9.2-5 機種ごとの変動点を どのように管理していますか

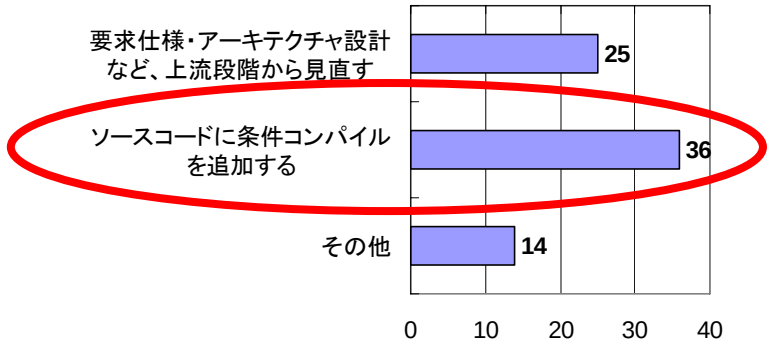


図9.2-6 機種が増えた場合に、 どんな方法で対応しますか

7.2 設計（機種共通部分の設計・共通部分を再利用した機種設計）(3)

- **アーキテクチャの評価・改善**
  - 84％のプロジェクトで、機種共通部分の再設計がタイムリーにできていない
    - 一度構築したアーキテクチャの改善の困難さ
    - アーキテクトの育成が課題
  - 「追加する機能が当初の想定を越えた」「構造劣化が進んだ」とき、アーキテクチャの見直しが行われる

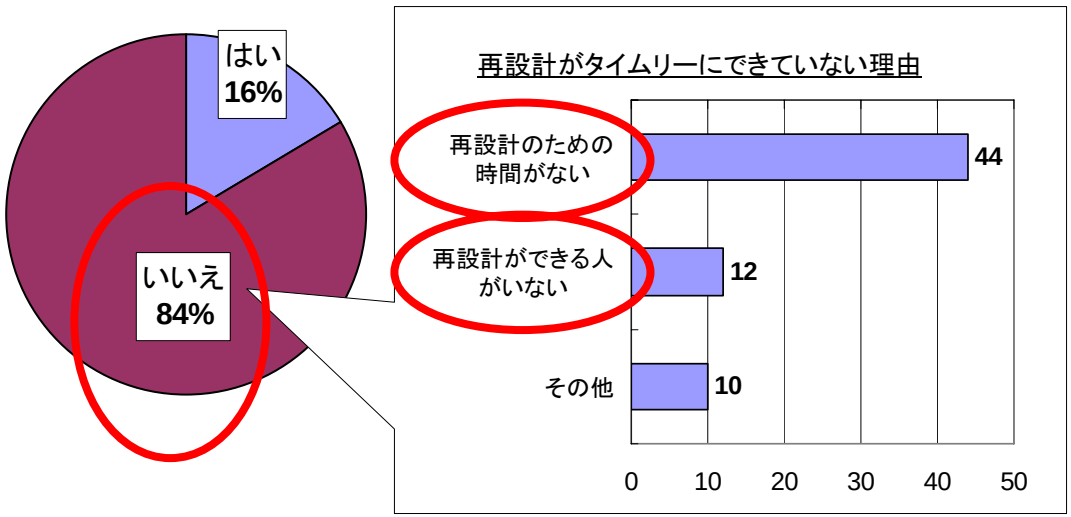


図9.2-7 製品ライフサイクルや技術変化に対応した機種共通部分の再設計が、タイムリーにできていますか

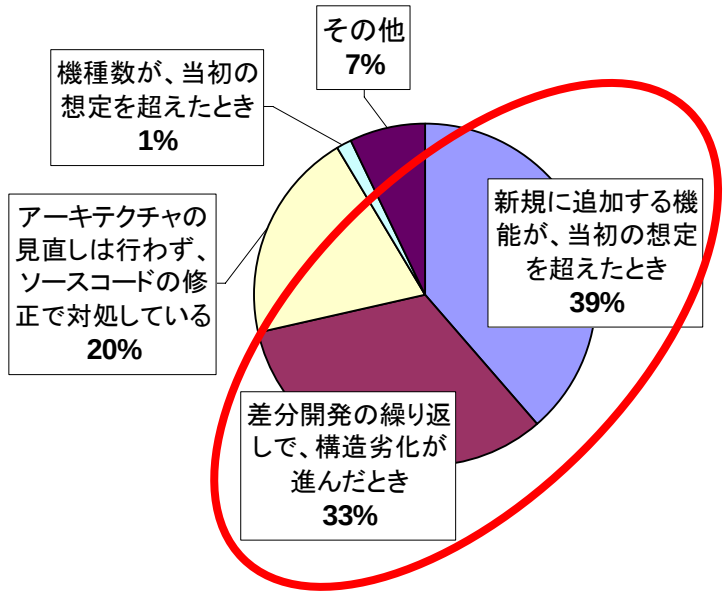


図9.2-8 アーキテクチャの見直しはどの段階で行われますか

7.3 マネージメント（商品戦略・開発体制・開発プロセス）

商品戦略との整合

68％のプロジェクトが、戦略の変化に対して改訂

- 変化に柔軟に対応していると見ることもできるが、
  別の見方をすれば、戦略的な開発が行われておらず、アドホックに対応している　とも考えられる

企画部門と開発部門間の連携の重要性は認識

- **回答の中には、連携の難しさも指摘されている**：
  「まったく連携していない」
  「企画優先で開発側の事情が考慮されない」
  「企画部門は仕様変更が与える影響を把握できていない」
  「開発部門は、企画部門に対して仕様変更の遅延・影響度合いを的確に説明できていない」・・・

多機種開発を、商品戦略と一体化させる状況には、至っていない

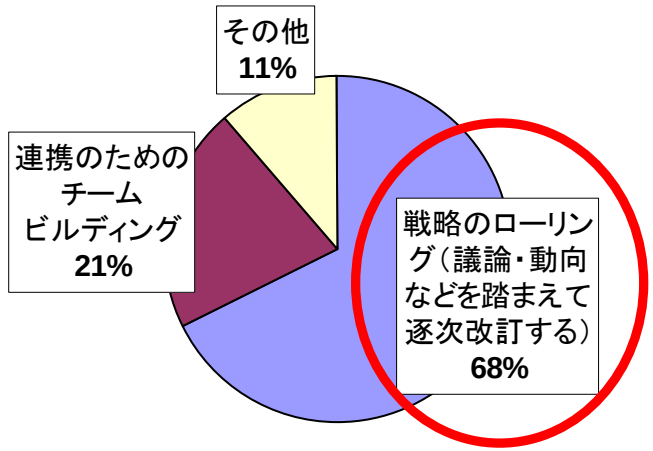


図9.3-1 機種系列に対して有効なアーキテクチャ設計を行うために、企画部門と開発部門とはどのように連携していますか

7.4 ソフトウェアプロダクトライン（1）

- **ソフトウェアプロダクトラインの導入状況**
  - 再利用を戦略的に行うトップダウンアプローチが実態にあっている：66%
  - ソフトウェアプロダクトラインに関心がある：8割超
    - すでに導入（14%）、導入に向けて取組み中（16%）

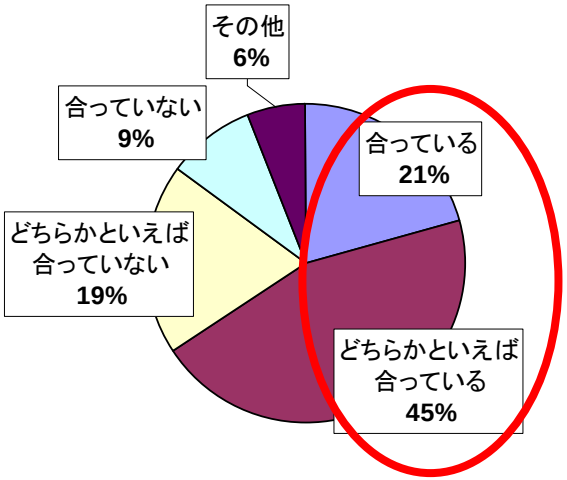


図9.4-1 再利用を戦略的に行うトップダウン的な手法は、貴プロジェクト、貴部門または貴社の実態に合っていますか

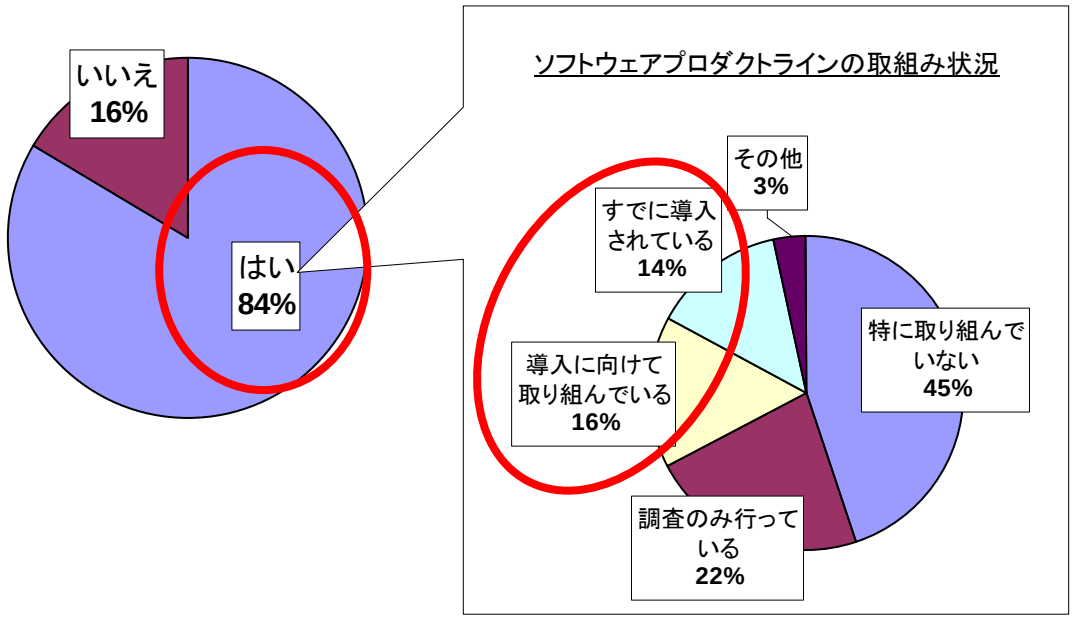


図9.4-2 ソフトウェアプロダクトラインに関心がありますか

7.4 ソフトウェアプロダクトライン (2)

- **ソフトウェアプロダクトラインの取組みを阻害する要因**
  - 「現行機種開発の納期の厳しさ」と「人的リソースの不足」が上位
    - 現状に追われていて、戦略的な取組みになっていないことを示唆
  - 「アーキテクチャの移行戦略の難しさ」と「事業・商品戦略立案層の知識・意識不足」が次に
    - 戦略的な活動の難しさを物語っている

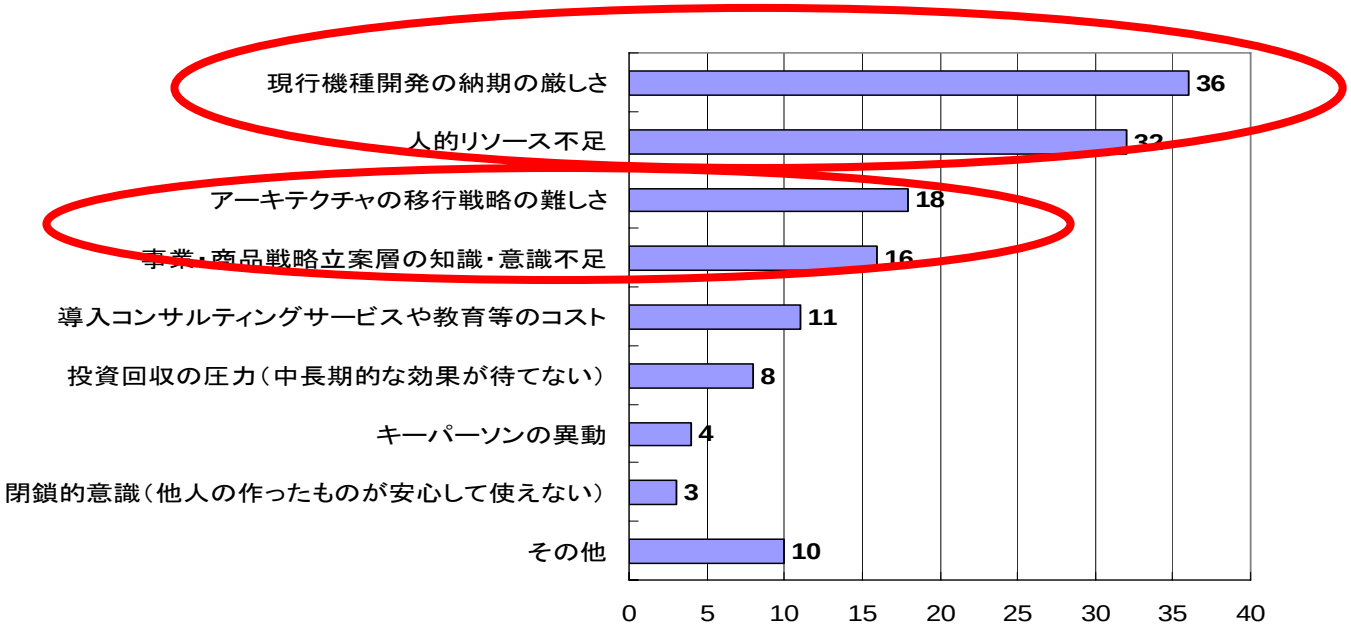


図9.4-3 ソフトウェアプロダクトラインを導入にするにあたって、
これを阻害する要因は何でしょうか

7.5「多機種開発の取組み」への提言

- **日本の開発特性に合わせたソフトウェアプロダクトライン、アーキテクチャ移行の方法論の確立**
  - ソフトウェアプロダクトラインの本来求めていることを明確にし、日本の開発実態に合った方法論の確立を行う
  - アーキテクチャの移行・改善に対して新しい焦点を当て、産学連携で取組む
- **経営層・技術者自身の意識改革**
  - 上流のアーキテクチャ設計・移行に対して、経営レベルでのリソース確保の決断
  - 視点を設計レベルまで上げることにより、木を見て森を見ない開発からの脱却
- **組込みソフトウェア分野でのアーキテクト育成の推進**
  - これまでのアーキテクトの技術に加えて、ソフトウェアプロダクトラインの考え方を熟知し、スコーピングされ変動性が識別された要求から、プロダクトラインを設定し変動性を実現していける技術を持つアーキテクトの育成
  - 経営や戦略の視点を持つアーキテクトの育成

８．課題解決に向けた今後の取組みへの総括的提言

8.1 今後強化すべき施策

日本の組込み系ソフトウェア開発の
国際競争力を強化するためには、
以下の5つの施策を強化していく必要がある

- □ アーキテクチャ主導開発(トップダウンアプローチ)
- □ 上流工程重視開発(フロントローディング)
- □ トップダウンアプローチとボトムアップアプローチの融合
- □ それらを遂行できる能力のある技術者チームの構築
- □ 戦略的・長期的な視点に立ったアーキテクトの育成

8.2 アーキテクチャ主導開発（トップダウンアプローチ）

組込み系ソフトウェア開発

- 大規模化
- 複雑化
- 短納期化
- 多機種化

アーキテクチャ

- トップダウンにシステム/ソフトウェア**全体を俯瞰できる**構造図
- **見える化された**ソフトウェアの全体構造

アーキテクチャをベースとしたトップダウンアプローチ開発

■ **開発量の抑制**：
- 全体構造図に基づく**自前開発のコア部分の特定**
- **自前開発量の絞り込み**
- 非コア部分の外部調達/外部委託の活用

■ **重複開発の抑制**：

● **社内開発での重複排除**
- 構造図に基づく共通部分と個別部分の識別

● **業界内での重複開発排除**
- 共通構造モデル・アーキテクチャの（業界）標準化
- 競争領域と非競争領域の特定
- 非競争領域での共同開発、分担開発、無開発（利用）

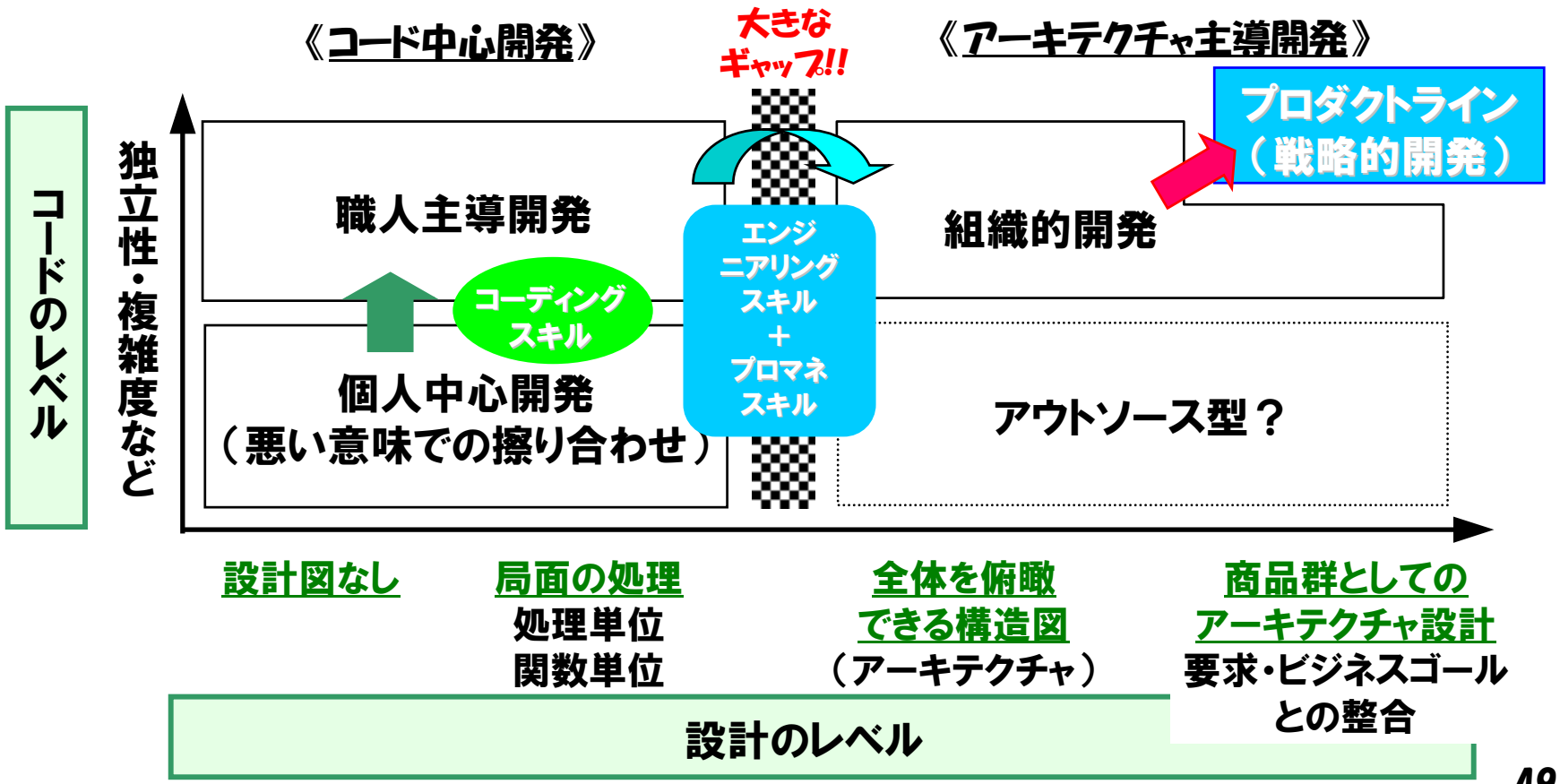
■ **計画的再利用、戦略的再利用の促進**：
- 構造図に基いて**事前に計画された再利用**
- コンポーネント化によるアドホックな再利用から離別

■ **「擦り合わせ」から「組み合わせ」開発の促進**：
- 構造図に基づいた**コンポーネント切り出しと組合せ**
- 多機種開発での開発量の「積算」から「加算」へ
  - 動く部分（変動部）と動かざる部分（共通部）の分離とコンポーネント化による **N×M→N+M化**

8.2.1 現状からの脱出：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？



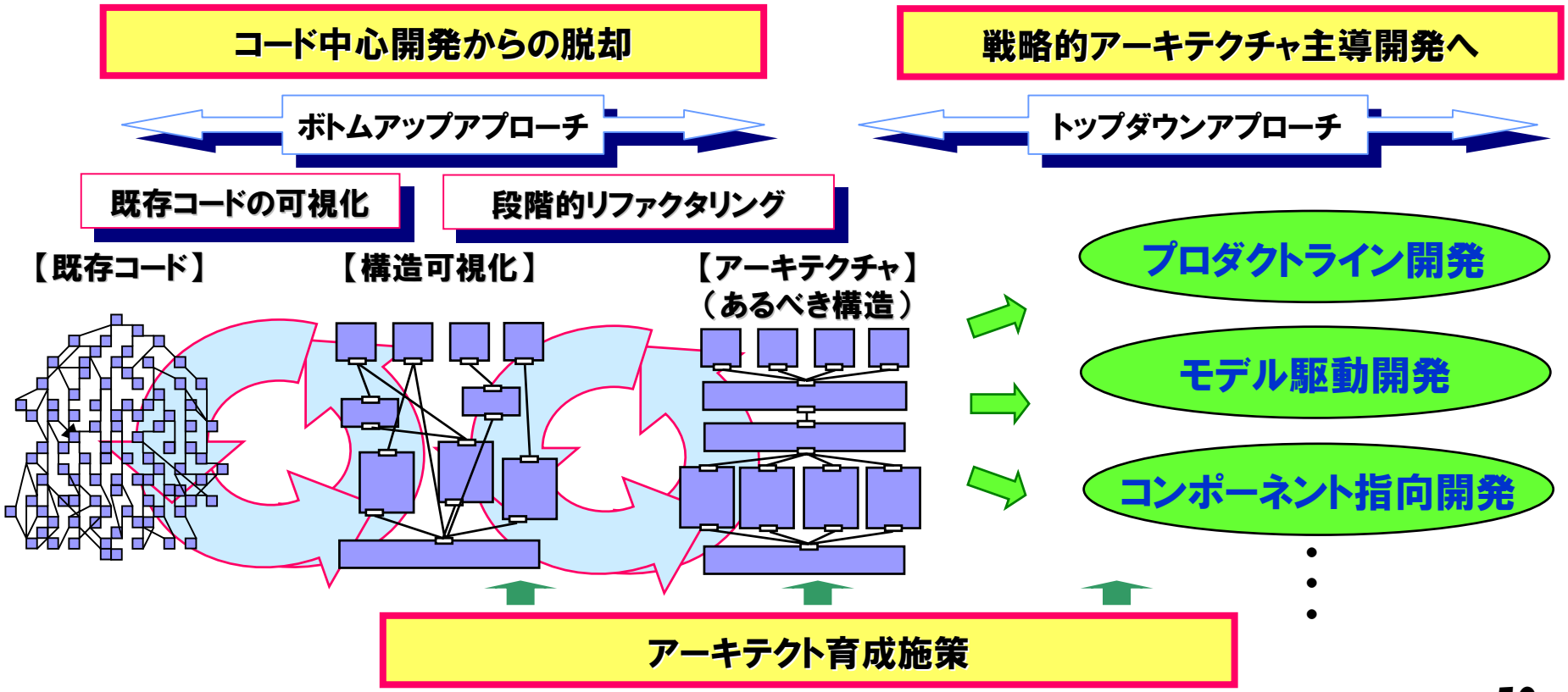
8.2.2 コード中心開発から戦略的アーキテクチャ主導開発へ

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

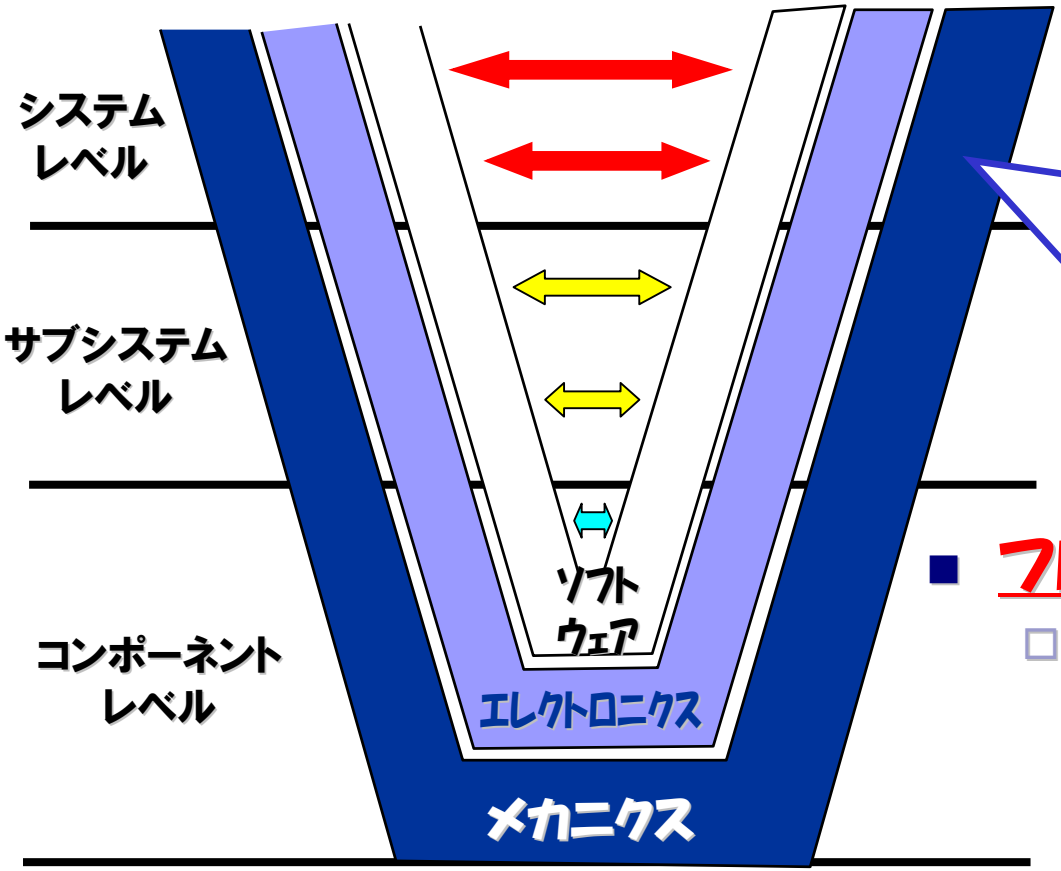
ステップ2 ： 戦略的アーキテクチャ主導開発へ

- プロダクトライン開発、モデル駆動開発など



8.3 上流工程重視開発（フロントローディング）

組込みシステム開発のV字モデル：三位一体



最下流工程での不具合：
最上流工程まで遡ることが多く
影響範囲も大きくなる

●日数を要するハードウェアの
手直しなどにより、
市場機会の損失のおそれも・・・

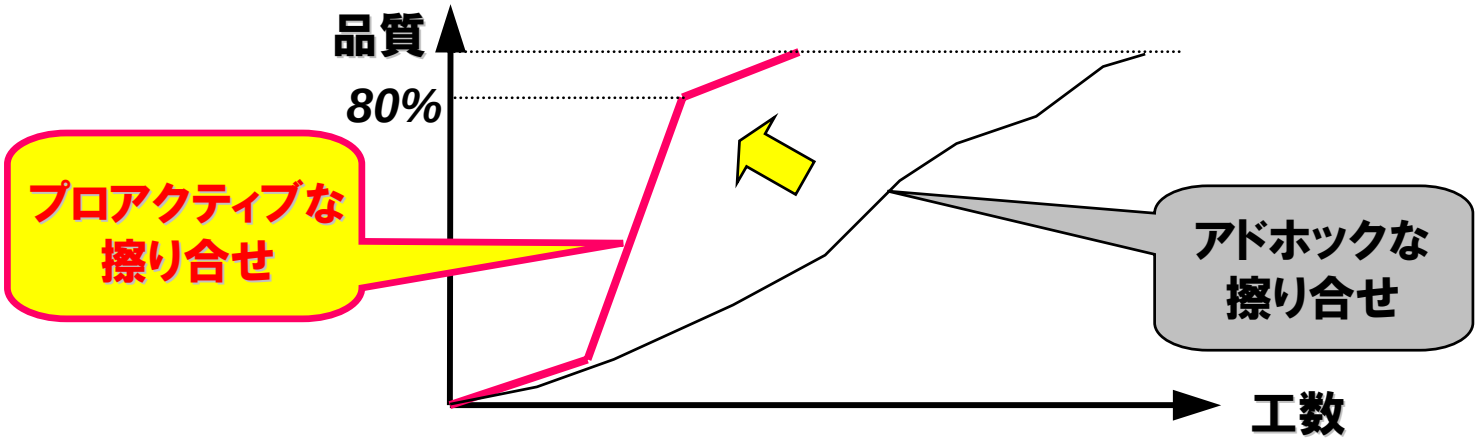
- フロントローディングの取組み：
  - 上流工程の早い段階で
    ハードウェアとソフトウェアの
    擦り合わせを行って
    問題点を先取り!!
    - シミュレータの計画的/戦略的活用など

8.4 トップダウンアプローチとボトムアップアプローチの融合

■ 擦り合わ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・短納期化などに対応できない

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

- 8割までは、組み合わせ（設計・アーキテクチャ力）の補完により、“すぐに”
- 残りの2割を、擦り合わせで



8.5 技術者チームの構築、アーキテクトの育成

- **遂行能力をもつ技術者チームの構築**
  - 構造を作るアーキテクト
  - 擦り合わせにより敢えて構造を崩せるスーパープログラマ
  - その両者が共存し補完しあうような技術者チームを構築してこそ、日本の強みを活かせるソフトウェア開発形態となる

- **戦略的・長期的な視点に立ったアーキテクトの育成**
  - 座学などの学校教育だけでの育成は困難
  - 一企業内の活動でもその育成は難しい
  - ”メンタリング（徒弟のような育成活動）“を通してスキルアップなど
  - 産官学一体となったアーキテクト育成への取組み・活動が望まれる

9. 2008年度の活動

組込み系ソフトウェアの開発スピードアップに向けて

● **開発を如何にスピードアップして、国際的競争に打ち勝っていくか**

これまでの活動（品質確保の課題分析と解決に向た提案）の
次ステップの“攻めのテーマ”として取り組む

(1)「開発スピードアップ」の主要課題の検討

(2)「開発スピードアップ」に関するセミナー／ワークショップ

- ●テーマ：　組込み系ソフトウェア開発をスピードアップ！（課題）
- ● 開催日：2008年8月27日(水) 13:30 ～ 17:30(予定)
- ● 会場：　ベルサール神保町（千代田ファーストビル南館）
- ● 案内：　7月下旬(予定)

(3)「開発スピードアップ」に関するアンケート調査

(4)「開発スピードアップ」の取組みに関する海外調査

(5) CEATEC 2008での活動成果報告講演

参 考 資 料

□ 平成19年度 ソフトウェアに関する調査報告書 I、II、III(IS-08-情シ-1、2、3)

- ソフトウェア技術者の育成に関する調査報告書と提言
- 組込み系ソフトウェア開発の課題分析と提言
- ソフトウェアリソースの最適活用に関する調査報告書

http://home.jeita.or.jp/is/committee/software/080625report/index.html

□ 2007 IESE/JEITA共同ワークショップ(2007年7月3日)

- 組込み系ソフトウェア開発の課題分析と提言

http://home.jeita.or.jp/is/committee/software/070906/index.html

□ CEATEC JAPAN 2007 インダストリアルシステムトラック講演(2007年10月2日)

- 組込み系ソフトウェア開発の課題分析と提言

http://home.jeita.or.jp/is/committee/software/071002/index.html

ご清聴ありがとうございました