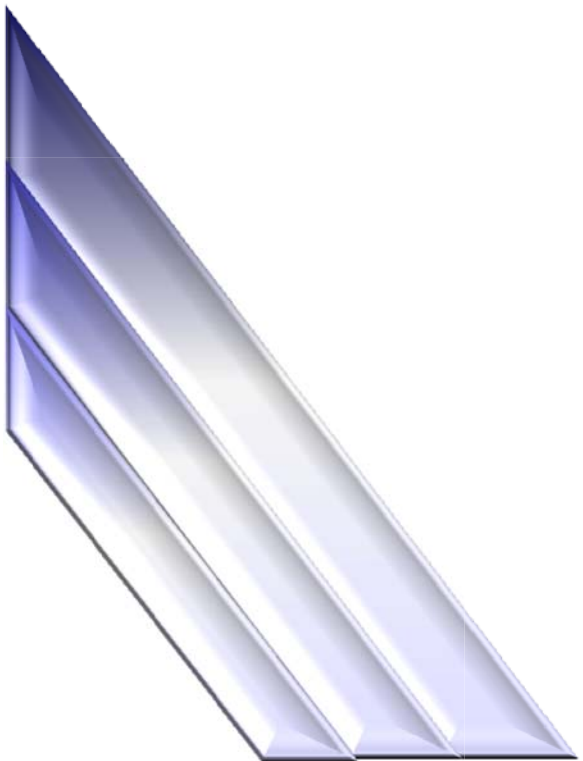
BACKSLASH DESIGN

組込みソフトウェア開発が抱える課題と対応

～求められる経営課題としての取り組み～



2008.01.23
近藤　満
ビースラッシュ株式会社
mitsuru.kondoh@bslash.co.jp

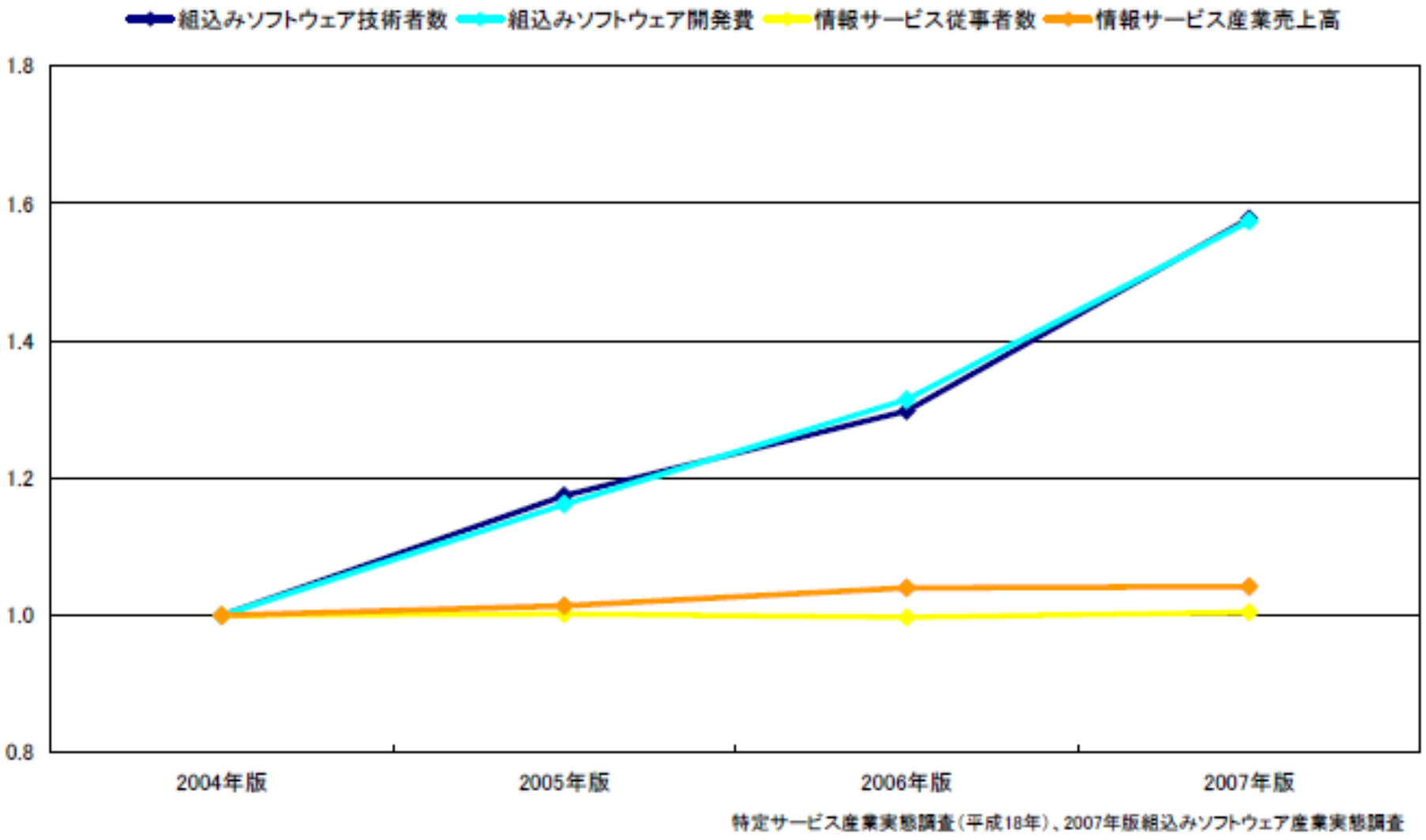
目次

- **開発現場が抱える問題・課題**
  - 経産省実施(2004～2007年)の「組込みSW産業実態調査」結果
  - 企業事例
- **開発課題の分析**
- **課題への対応**
- **開発プロセス改善・改革の壁**

開発現場が抱える問題・課題 -経産省実態調査結果(1)-

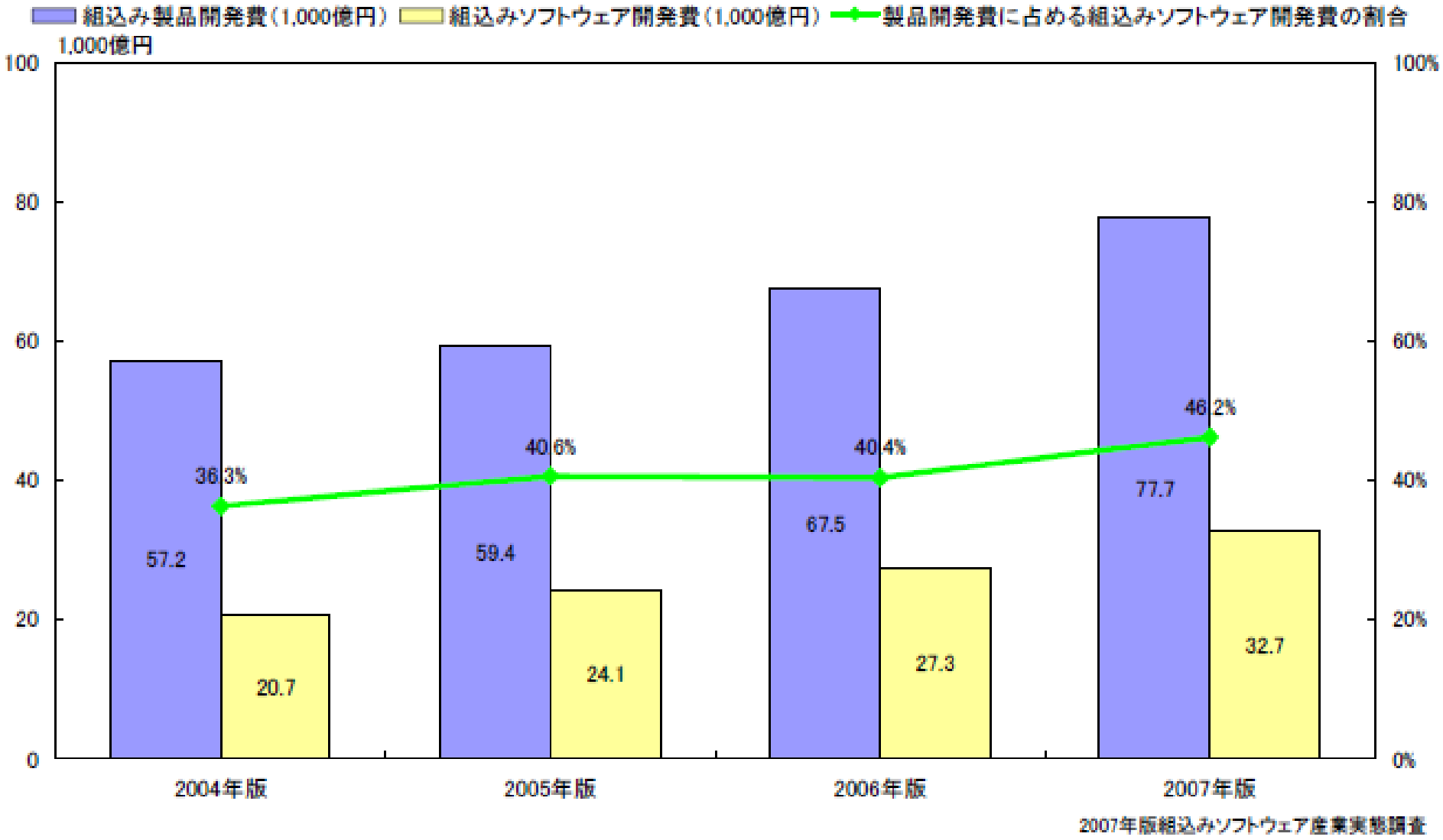
- 増大中の組込みSW開発費
  - プロジェクト開発費に占めるSW開発費の割合は62.3%(2007年調査)
  - プロジェクトの最大費用は組込みSW開発費で、年率平均12.0%で増大中(2007年調査)。
- 慢性的な組込みSW技術者の不足
  - 要求人員数は年率平均12.1%で増大中で、9.9万人の不足状況にあり(2007年調査)、慢性的人員不足。
  - 技術者不足を外部委託で対処
    - ◆外部委託の一番理由は社内リソースの不足(大手企業で80%回答、2007年調査)
- 高い外部委託工数比率
  - 内部開発工数比率は41.9% (2007年調査)
    - ◆外部依存型で技術流出、技術の空洞化の恐れあり。
  - 外部委託企業比率は82%と、米国(47%)、欧州(35%)に比較して高い(2004年調査)
- 難しい外部委託の品質管理
  - 外部委託の上位課題として、品質管理の難しさ、人材の継続的な確保を挙げている(2007年調査)

組込ソフトウェア規模の推移



特定サービス産業実態調査（平成18年）、2007年版組込みソフトウェア産業実態調査

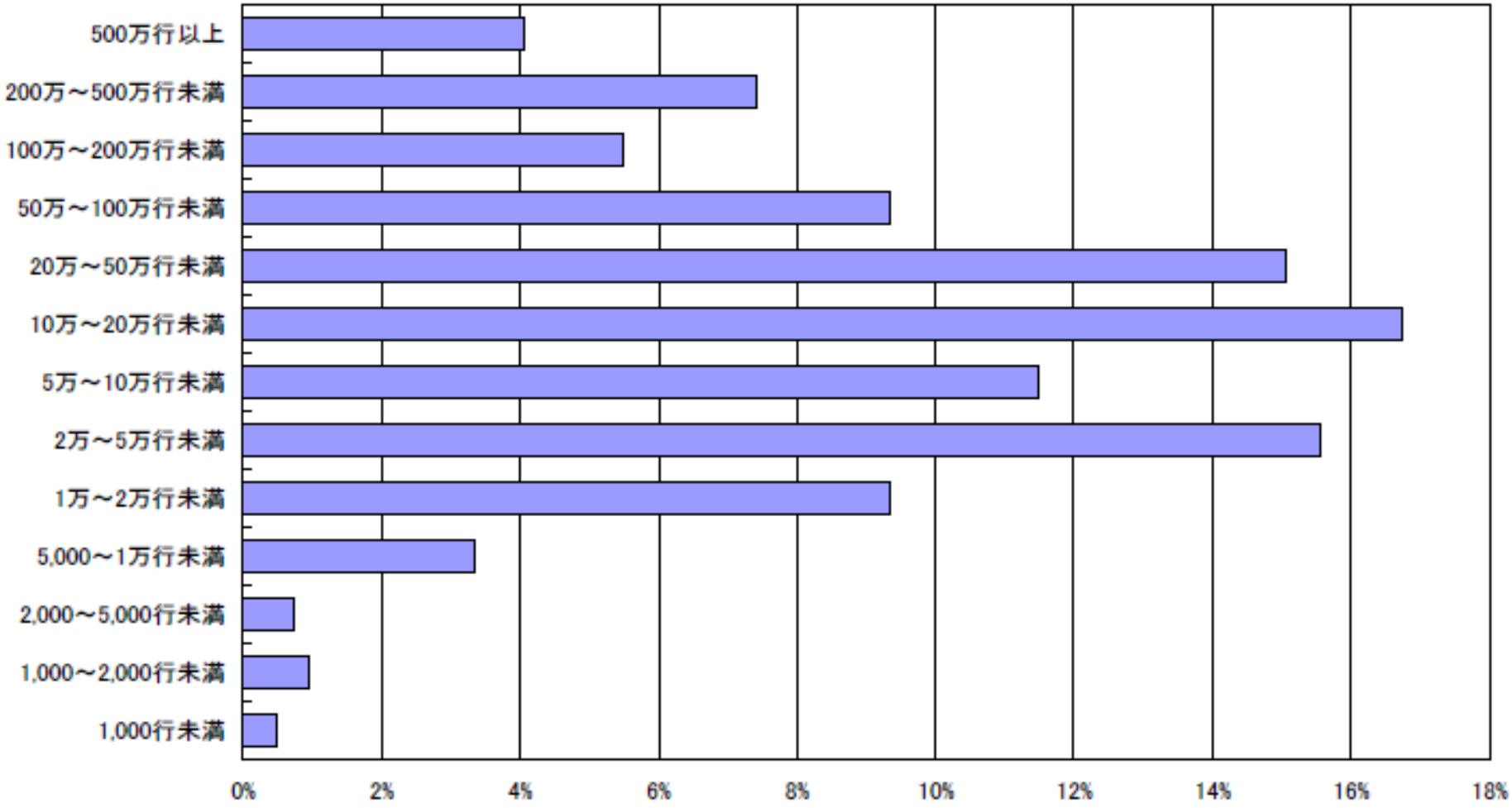
組込みソフトウェア開発費割合の推移



開発現場が抱える問題・課題 -経産省実態調査結果(2)-

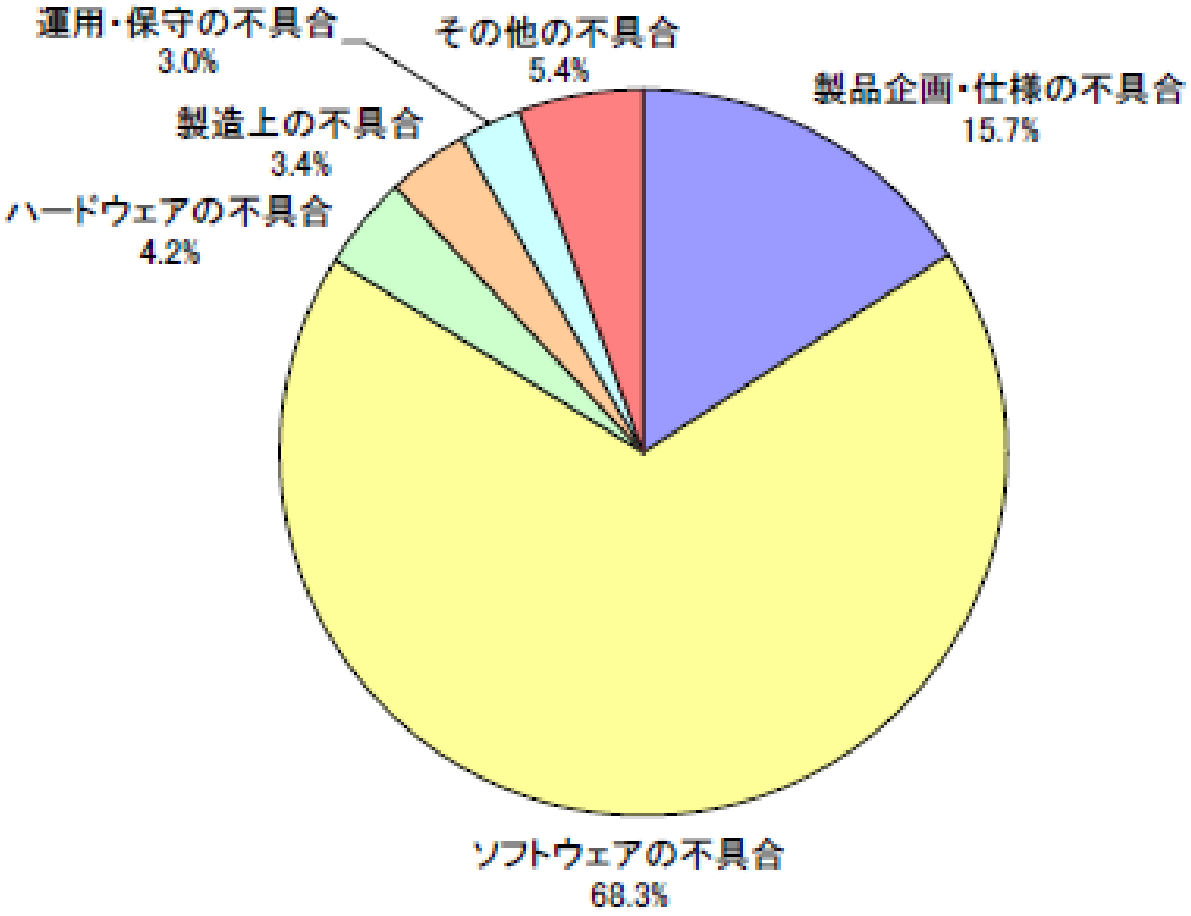
- 肥大化するSWサイズ
  - 10～20万行にピーク、500万行以上との回答も4%（2007年調査）
- 製品出荷後に多発するSW不具合
  - 68.3%がSW原因の不具合（2007年調査）
- 手戻りの最大原因は要求仕様の不備にあり
  - 一番が要求仕様の不備、二番はソフトウェア設計上の不具合（2004年調査）
- 低い、システム設計の自組織実施率
  - 組込みシステム製品のコア技術であるシステム設計を100%自組織行っている割合が13%と、米国(36%)、欧州（25%）に比べて低い。
- 欧米に比べて低い技術者のスキル標準／スキル評価制度の保有
  - 日本（18%）、米国（約40%）、欧州（70%）
- 技術者スキル不足
  - 開発現場は、今後のプロジェクトの最優先課題を技術者スキル向上・育成と認識（2007年調査）
  - 大学での教育強化に最大期待 （2004年調査）

開発全行数（新規開発と既存の合計）



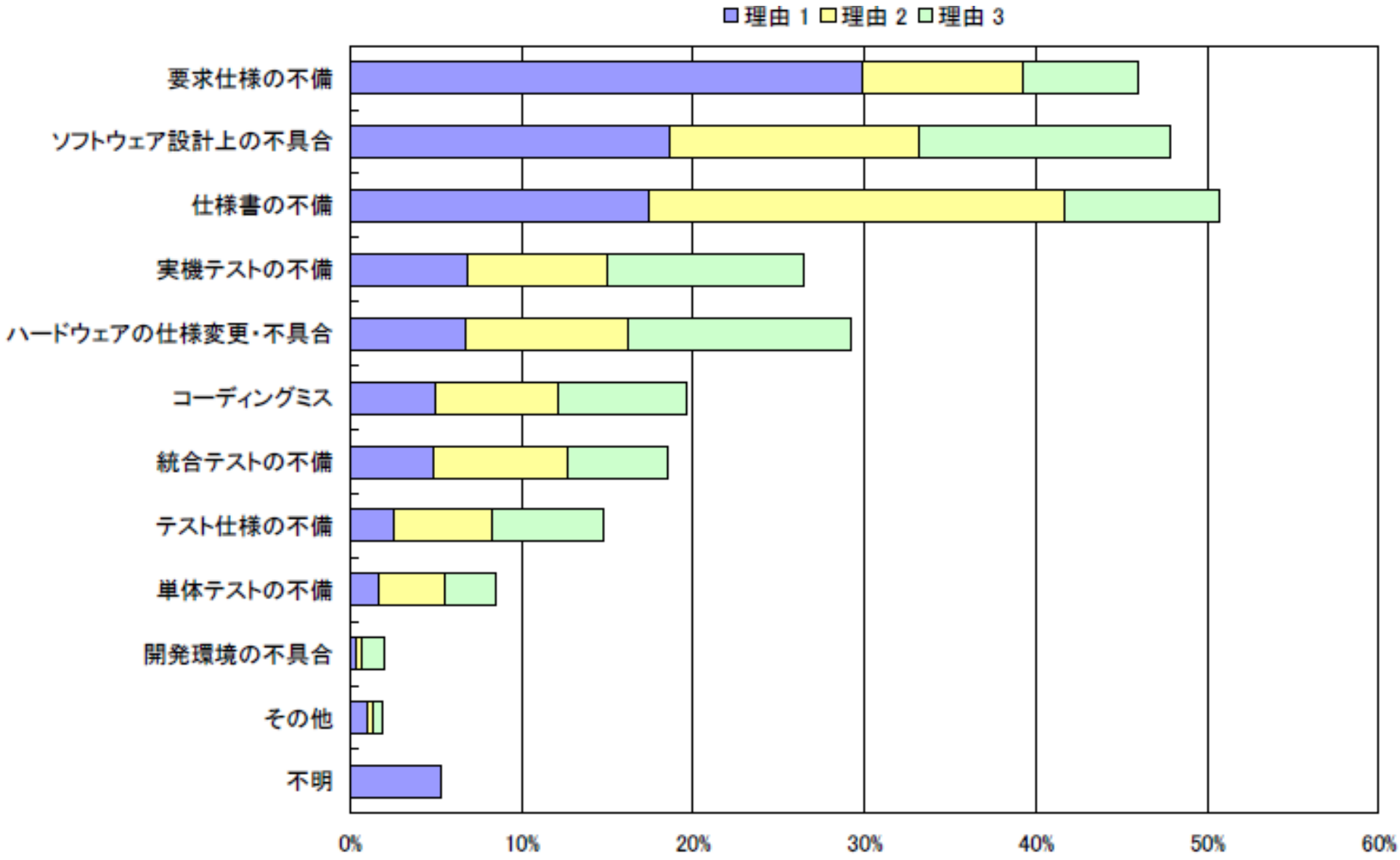
出典：経産省 2007年版「組込みソフトウェア産業実態調査報告書」

製品出荷後の不具合件数



出典：経産省 2007年版「組込みソフトウェア産業実態調査報告書」

組込みソフトウェア手戻りの原因



出典：経産省 2004年版「組込みソフトウェア産業実態調査報告書」

開発現場が抱える問題・課題 -企業事例(1)-

- 機構系主体製品時代の開発プロセス採用
  - 設計試作→技術試作→量産試作の積み重ねによる品質の作り込み
  - ソフトウェア開発は最下流工程
- ソフトウェア原因の品質問題の発生
  - 出荷間際の安易な仕様変更の実施
  - テストで検出しきれないバグ
  - ベテランもポカミス
- 肥大化速度に追従出来ない生産性の向上
  - 既に高いコードの再利用率
  - 進まない再利用方法の改革
    - ◆規模の肥大化と、大量のコード資産保有
    - ◆高い技術的ハードル
    - ◆NIH
    - ◆プロジェクト毎のPM制を施行、PMの使命はプロジェクトのQCD必達
      - 組織に技術／資産を残すこはプロジェクトのミッション外

開発現場が抱える問題・課題 -企業事例(2)-

- 開発経費の肥大化
  - 肥大化速度に追従出来ていない開発生産性
    - ◆変わらずコードの再利用が中心
  - テスト工数の肥大化
    - ◆テスト依存の品質確保
    - ◆実機依存の評価で大量の試作機を要求
      - – 評価期間短縮に実機台数を増加
  - 大きな手戻り工数の発生
    - ◆要求仕様の不備
      - – 開発工程の後半で仕様変更多発
- ソフトウェアの価値判断の不在
  - 直接原価として積み上げられないソフトウェア開発費
    - ◆変わらず一過性の経費扱いのまま
    - ◆プロジェクト(製品テーマ)毎のP/Lで評価
  - 組織の資産開発との行動が促進されない
    - ◆再利用開発の遅れ

開発現場が抱える問題・課題 -企業事例(3)-

- ベテラン程テーマのQCD達成に忙しく、新規な技術を学ぶ(充電する)機会／時間が不足
  - 絶え間ないプロジェクトへの参加
- 伸びない新人の技術力
  - 新人に外部委託管理を依存
    - ◆増大する外部委託管理に大きなパワーを要求
    - ◆技術力を有する人は開発担当に
- スキルパス、キャリアパスに基づいた教育体系の不在
  - 部下の教育受講の管理が上司の評価対象外
  - 新人教育＝OJTとの安易な考えが支配的
    - ◆環境変化に対応しきれていない社内技術力
- スキルに基づく処遇の遅れ
  - ソフトウェア技術者のスキル判断尺度の不在
    - ◆経験年数／担当プロジェクト数による判断
  - 担当テーマのP/L、QCD達成が主体の人事評価／処遇

目次

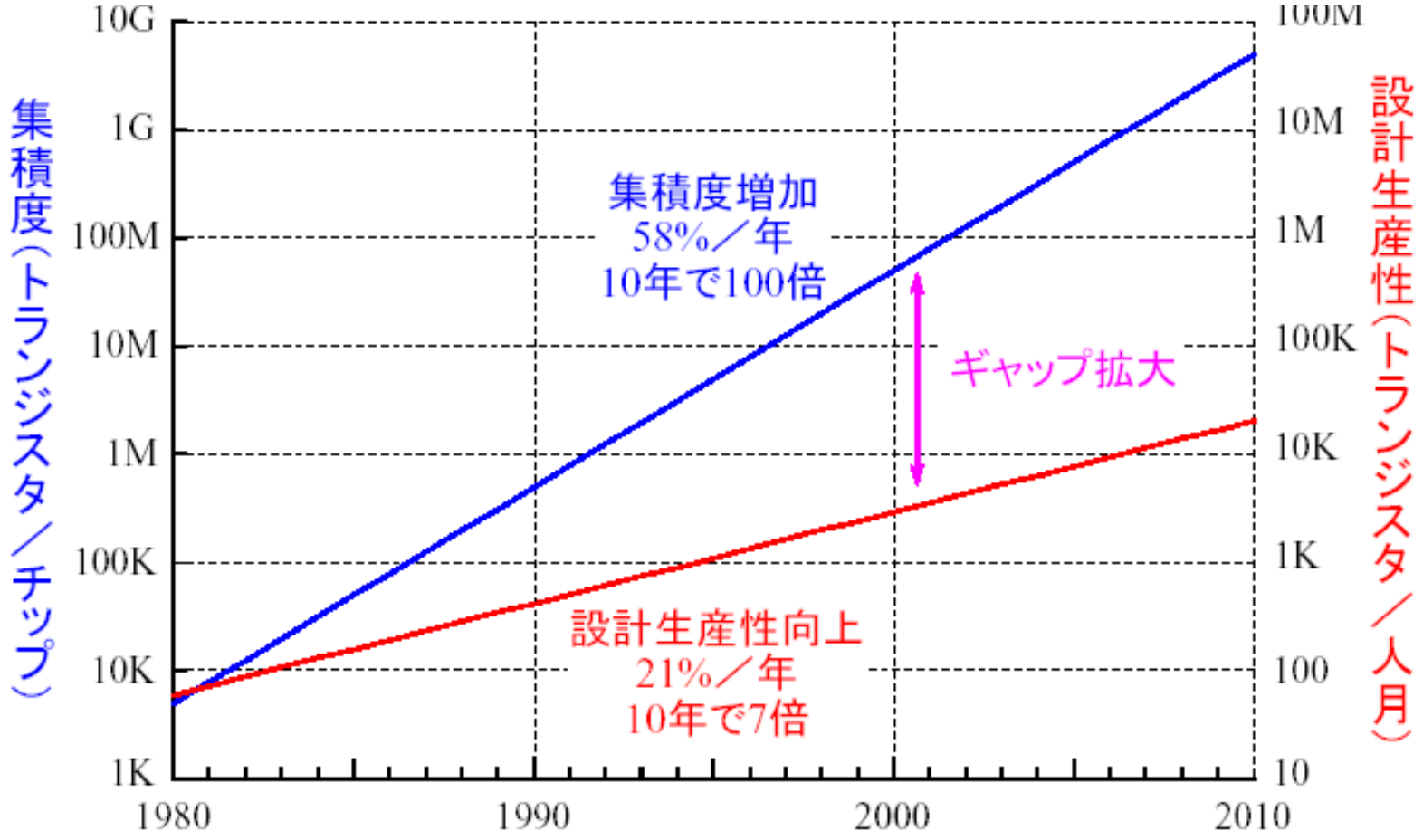
- 開発現場が抱える問題・課題
  - 経産省実施(2004～2007年)の「組込みSW産業実態調査」結果
  - 企業事例
- **開発課題の分析**
- 課題への対応
- 開発プロセス改善・改革の壁

課題の分析(1)

- ソフトウェア開発の2大課題
  - 高いソフトウェア品質
  - 開発生産性の大幅向上
- 肥大化、複雑化さの増加要因
  - 多機能化、複数バージョンの混在、利用者ごとのカスタマイズ
  - 共同開発
  - 半導体微細化技術の継続的な進歩
- 肥大化／複雑化への対処は従来延長線の開発方法では困難
  - 肥大化速度に生産性向上が追いつかない
    - ◆個人の生産性向上には限界がある
    - ◆コードの再利用では求める生産性向上が得られない
  - 開発工数はプログラム規模の冪乗になる ※「人月の神話」フレデリック・P・ブルックス
    - ◆モジュール数の増加でモジュール間I/F、システムの複雑さは指数関数的に増大
  - 開発要員増と開発スピードは比例しない
    - ◆プロジェクトの人数が増えるほど生産性は低下
  - テスト依存では品質確保に限界あり
    - ◆莫大な数のテストケースを要求
    - ◆バグを取り切れない

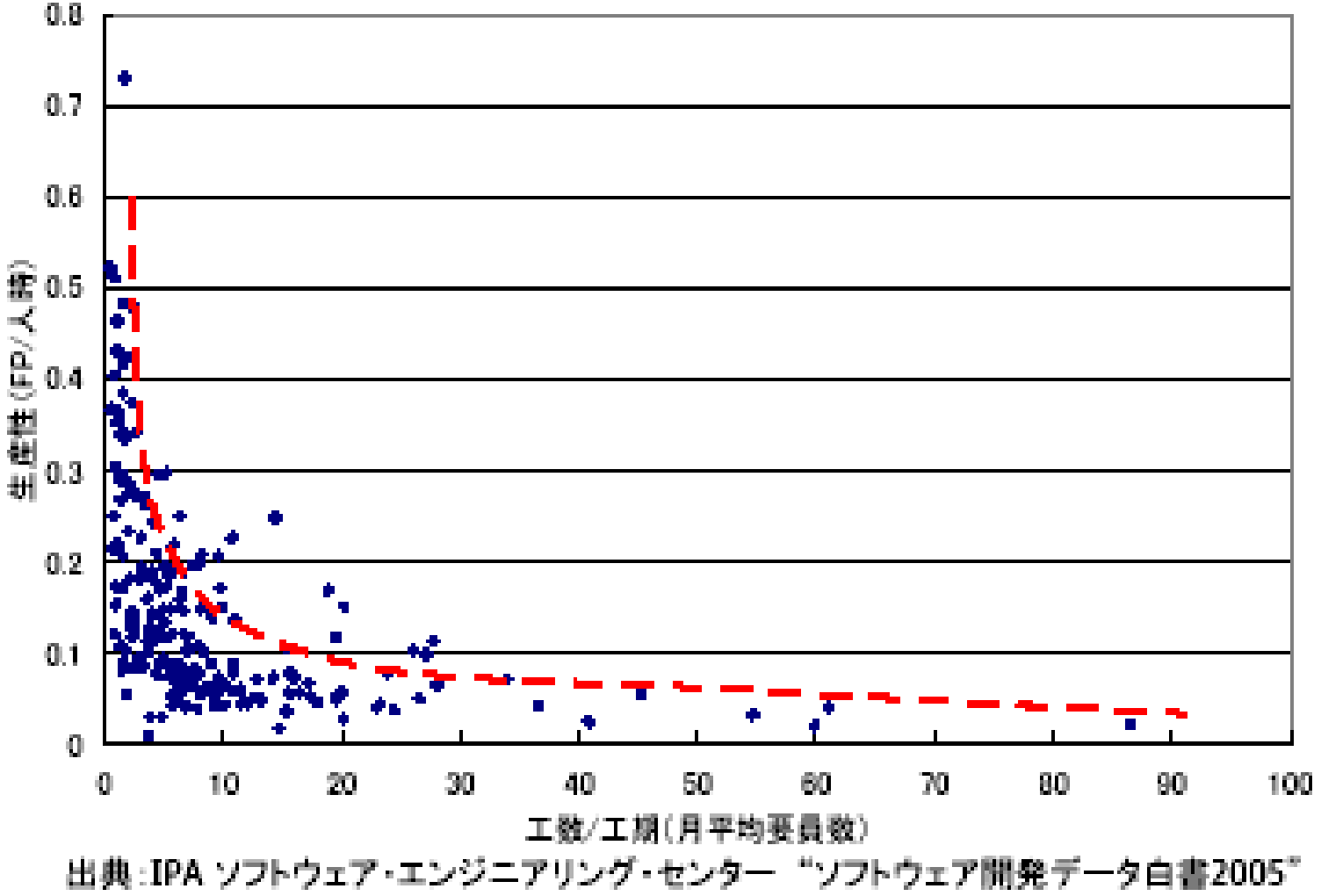
半導体に見る集積度と設計生産性のギャップ

出典：電子情報通信学会東海支部学生講演会（2002/5/29）



ブルックスの法則は健在

- 遅れているソフトウェアプロジェクトへの要員追加はさらに遅らせる
- プロジェクト中に人数が増えるほど生産性は低い

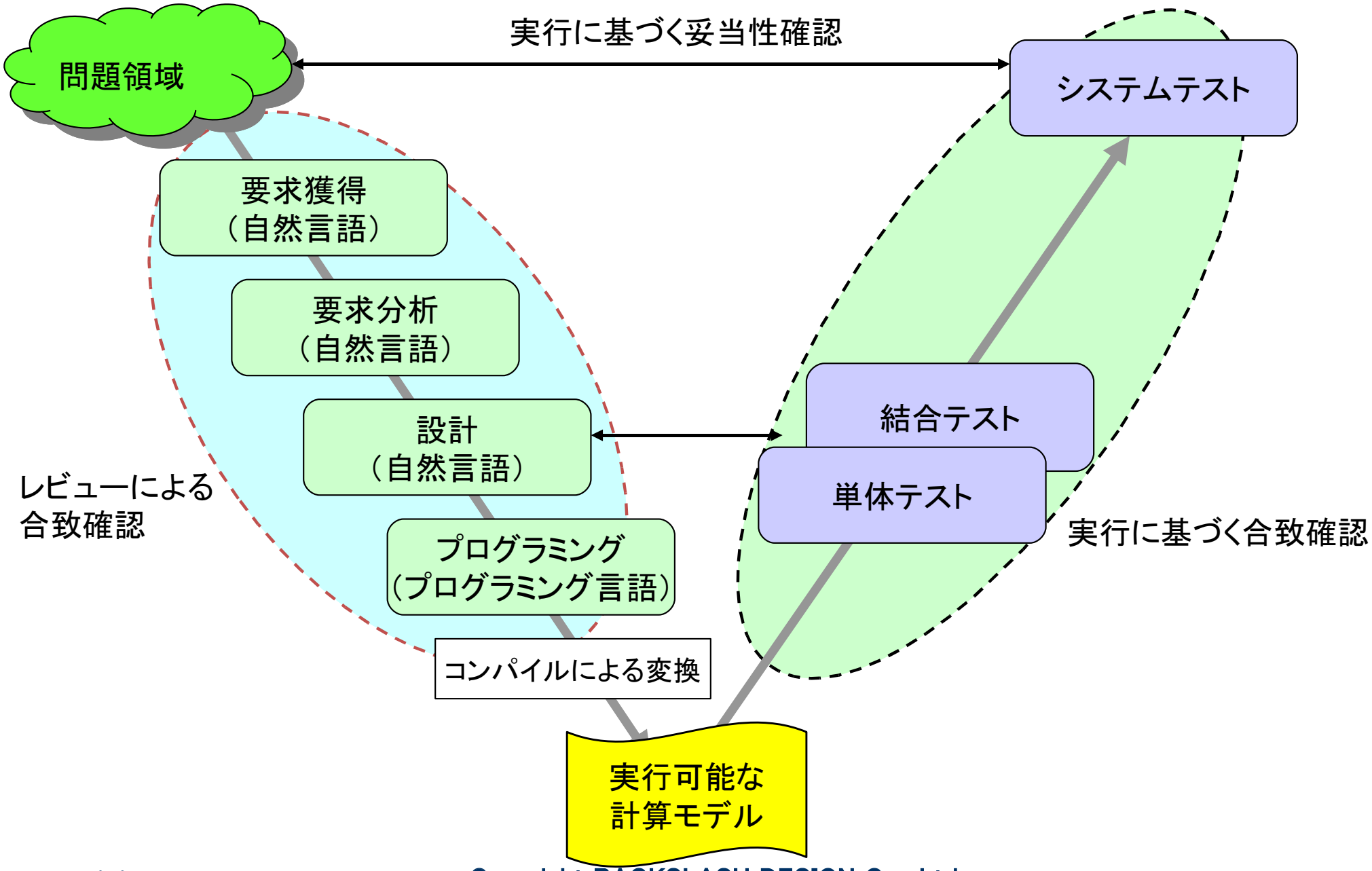


出典:IPA ソフトウェア・エンジニアリング・センター "ソフトウェア開発データ白書2005"

課題の分析(2)

- 全て自前の開発・設計では肥大化／開発期間要求に対応不可能
  - 肥大化と生産性向上のギャップに外部リソースへの依存は必然
    - ◆ 日本の理工系卒業者数では組込み技術者不足は解消できない
- 要求仕様獲得の不備
  - システムテスト依存の仕様検証では狙いの品質と生産性が達成できない
- 設計力の不足
  - 全体像把握力の不足、力ずくの設計で肥大化／複雑化に対応不十分
- 教育体制の不備
  - コア技術者ほど新規技術修得の機会を損失
  - OJT一辺倒では最新技術を修得できない
- ソフトウェア価値判断の不在
  - ソフトウェア資産化への開発が進まない
  - 再利用開発が評価されない
- 開発プロセス改善・改革への組織的な取組みの遅れ
  - 個々のプロセス改善・改革行動では環境変化に対応し切れない
  - NIHは技術者に共通な行動パターン

V字モデルによる品質保証



目次

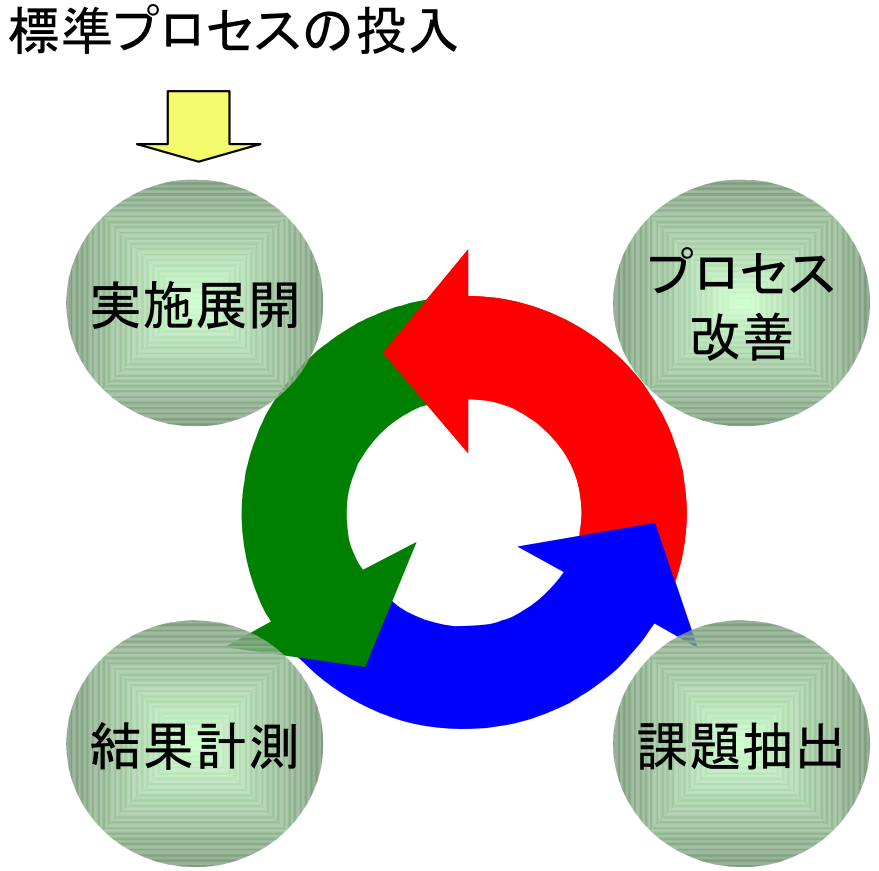
- 開発現場が抱える問題・課題
  - 経産省実施(2004～2007年)の「組込みSW産業実態調査」結果
  - 企業事例
- 開発課題の分析
- **課題への対応**
- 開発プロセス改善・改革の壁

開発課題への対応(1)

- 開発プロセスの改善
  - 組織／プロジェクトに標準プロセスを導入
  - プロセス成熟度の向上
    - ◆PDCAサイクルの実施
- トップダウン設計の実施
  - 将来の拡張性／発展性を見込んだアーキテクチャを開発すること
  - 何を作るかを関係者に分かるようにすること(可視化)
  - 上流設計工程と下流工程を分離し、上流設計工程を重視すること
- 抽象化設計の実施
  - システム全体を把握すること
  - 設計の抽象度向上で生産性をアップ
- 工学的手法の導入 “ソフトウェア工学の成果を開発現場に確りと適用 ”
  - 開発プロセス／開発手法・技法の活用
  - 再利用方法の革新
    - ◆再利用粒度の拡大
    - ◆結果の再利用から予め再利用を予定した開発へ
    - ◆コードの再利用から、包括的な再利用へ

開発プロセスの成熟度向上

- 標準開発プロセスの用意
  - プロセスの定義
- 実施展開
  - プロセス教育の実施
  - 組織／プロジェクトへの実践
- 適用効果の計測
  - メトリクスの設定
  - 計測システムの構築
  - 結果／効果の把握
- 課題の抽出
  - 結果／効果の分析
  - 課題の発見
- プロセスの改善
  - プロセス問題点の修正



工学的手法の導入

- ソフトウェア工学の要素と技術領域
  - 「再利用」、「 ValidationとVerification」
    - 再利用はソフトウェア工学の規範
  - 「開発プロセス」、「方法論」、「人材」
- 開発プロセス
  - 分析、設計、実装、テスト、保守のライフサイクルを対象とした開発管理
    - ウォーターフォール、反復型(UP)、アジャイル(XP、SCRUM)
  - 定義、計測、マネージメントのサイクルでプロセス成熟度を向上
- 方法論
  - 多くの技術者が均質な設計品質を達成するには、共通規範となる設計方法論が有効
    - 属人性の排除
  - 設計対象を表現する記法
    - プログラム言語、モデル表記法、フォーマルメソド
  - 設計手順を定める運用
    - トップダウンアプローチ、データ指向アプローチ
    - 抽象データ型／構造化分析・設計／オブジェクト指向分析・設計
    - ドメインエンジニアリング、プロダクトラインエンジニアリング

要求工学

- システム開発の最大課題は要求仕様にあり
  - 要求仕様が不完全／曖昧、要求仕様が顧客の目標に合致していない、顧客求める要求仕様変更に追従出来ない
  - 組込みソフトの手戻りのトップ原因は要求仕様の不備／仕様書の不備にあり
- 要求工学とは
  - 要求工学の狙い
    - ソフトウェアは目的をもって作成される。仕様と目的の摺り合わせ、あるいは曖昧な目的を明瞭な目的に洗練化し、それを最大限忠実に表現する仕様を作成すること（プログラムが完成する前に仕様のバグを発見すること）
  - Verification とValidation
    - Verificationとは仕様を絶対的公理と考え、この仕様にソフトウェアが合致するか否かを確かめること
    - Validationとは、完成したソフトウェアを動作させることで、それが仕様以前の要求、つまりソフトウェアが構築された本来の目的に合致するかを調べること
- 要求工学のプロセス
  - 要求の抽出／要求分析／要求の妥当性確認／要求管理／ステークホルダー分析／ユースケース分析／変動性と機能の分析

モデルベースのソフトウェア開発

- 従来開発方式の問題点
  - システム全体を貫くアーキテクチャーを開発せず、いきなりシステムの構築を進めてしまっては拡張性に乏しく、機能追加が困難であり、それを無理やり実施している内にシステムはLegacy（遺産）化してしまう
- 対応策
  - アーキテクチャの検討
    - 先ず将来の拡張性／発展性を見込んだアーキテクチャーを検討することから開始すべきで、そのアーキテクチャー検討用のツールにUMLがある。
  - 先ず何を作るかを関係者に分るように明確にすること
    - その為には、関係者に「分る形」「見える形」で何を作るかを表現することが必要であり、その鍵は「抽象化」と「可視化」にある。この「抽象化」と「可視化」実行手段としてモデル化がある。
  - モデル化
    - エンジニアリング・モデルの主目的はリスクを低減すること、様々なシステム関係者にお金と工数をかけて実際に建築する前に複雑なシステムの特徴を理解してもらう手助けをすること、設計のアイディアを伝えることにある。
    - モデル設計手法の有力な候補にオブジェクト指向分析・設計がある

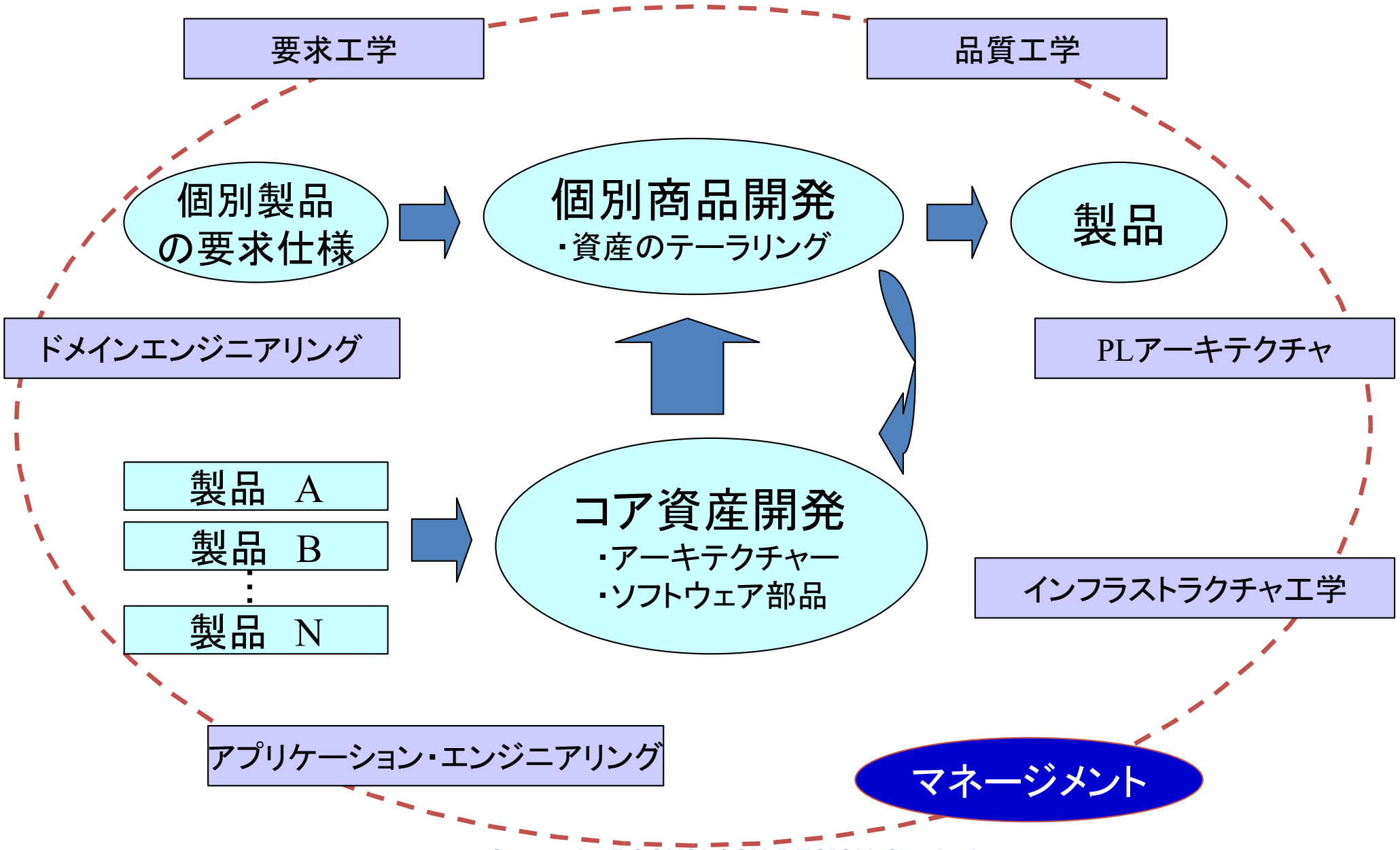
再利用設計の実施

- トップダウン・システム設計
  - 要求分析の重視
  - システムアーキテクチャー設計の重視
  - 下流工程成果物の再利用から上流工程成果物の再利用へ
  - 結果の再利用から、予め再利用を意図した設計へ
- 可視化設計
  - 特異的で少数の人のみが理解できるコードからの脱却し、モデル設計へ
- 再利用粒度の拡大
  - 個々のコンポーネント再利用からコンポーネント群の再利用へ
  - コンポーネントの再利用からシステムの再利用へ
  - 個別商品群での再利用から、類似商品群での再利用へ
  - 変動部を考慮した設計
    - ◆魅力ある機能追加部の予測
    - ◆技術革新部の変動扱い
- 全設計資産の再利用
  - コンポーネントの再利用から、要求仕様、モデル、アーキテクチャ、テストプラン、ドキュメントの包括再利用へ

プロダクトライン・エンジニアリング(PLE)

- 狙い
  - 従来の再利用プロセスに比して、更なる粒度の大きな大規模再利用の実現
- 技術ベース
  - ドメイン・エンジニアリングを基盤とし、その持つ課題への対応に加え、更なる上流工程開発へ発展させた包括的な再利用プロセス
- 技術のポイント
  - 固定と変動設計
  - 全ての資産を単一の製品だけではなく、予め製品ライン(ファミリー)を対象として再利用することを意図して開発
  - コア資産を注意深く設計し、組織に知的財産として資産化していくことで、各製品開発をこのコア資産のテーラリング(インスタント)行為化
  - アーキテクチャーの再利用を重視し、プロダクトライン・アーキテクチャーとして予め製品ラインの変動を支持するように設計

PLE を導入した開発構造



開発課題への対応(2)

- 既存遺産の資産化
  - 理解性の向上、再利用性の向上
    - リバースエンジニアリングの実施
    - モデル化、洗練化
- コンピュータパワーの活用
  - 実行(シミュレーション)可能なモデル記述で、V字型品質保証からの脱却
  - 下流工程の自動化
- 人材の育成　“ソフトウェア工学で最も重要な資源は人材”
  - 技術者スキル標準の普及
  - キャリアパス、スキルパスに基づいた教育体系の整備
  - スキルに照らした人事評価の実施
    - 人事評価制度への組込み
- 魅力ある組込みソフトウェア職種づくり
  - 応分の報酬
  - 社会的地位向上
  - 3K排除
- 外部委託プロセスの改善
  - 品質の向上
  - 技術の空洞化／流出の防止が出来るプロセス

人材の育成

- 体系的なキャリアパス／スキルパスの制定
  - スキル標準の制定
    - ◆事業戦略をスキル要件に反映
  - キャリアパスとスキルパスの連携
- スキル育成計画の作成
  - 計画的なスキル育成
  - 人材育成を上司の評価尺度化
- 教育体制の整備
  - 社内教育スキルと外部委託教育スキルの区分
  - 組織資産を活用した教育
    - ◆組織に資産の蓄積
- スキル評価を反映した処遇
  - プロセスの重視
- 新卒採用者に求めるスキル要件の明確化
  - 学に期待するスキル、産で育成する技術の区分
  - 要求するスキル要件を明確にした採用

目次

- 開発現場が抱える問題・課題
  - 経産省実施(2004～2007年)の「組込みSW産業実態調査」結果
  - 企業事例
- 開発課題の分析
- 課題への対応
- **開発プロセス改善・改革の壁**

開発プロセス改善・改革の壁

- 中長期視野での体質改善実施
  - 短期業績重視でプロセス改革を実行し切れない
    - 業績が悪化するまで実行しきれない改革
    - 競合に偏った競争意識
  - 結果重視、プロセス軽視
- トップマネージメントによる組込みソフトウェアの重要性認識
  - 組込みソフトウェアが日本の組込みシステムの産業競争力の鍵
    - 今や組み込みソフトウェアが機能・性能達成の要である現状
  - 届かない実態調査結果
- 組込みソフトウェア開発方法改革の必要性認識
  - 環境変化の認識が十分でない
- 技術者のスキル向上
  - 技術者スキル標準の導入
  - スキルに照らした処遇
- 新卒採用方法の改革
  - 基礎学力偏重の採用からの脱却
    - 配属先／担当職種を明確にした採用

参照文献

1) 組込みソフトウェア産業実態調査報告書
2004年版、2005年版、2006年版、2007年版
経済産業省商務情報製作局

2)UML for Real:Design of Embedded Real-Time Systems
Lucian Lavagno

3) オブジェクト指向システム設計
E・ヨードン著　松原友夫訳
株式会社トッパン発行　ISBN4-8101-8594-X

4) バーチャルエンジニア
Howard C. Crabb　吉村信敏、他訳
日経BP社　ISBN4-8222-1860-0

5) A Framework for Software Product Line Practice
Version4.1Carnegie Mellon Software Engineering Institute
http://www.sei.cmu.edu/plp/framework.html, 他

6) Component-Based Product Line Engineering with UML Collin Atkinson, 他
ISBN 0-201-73791-4

7)　モデル駆動型ソフトウェアテストの可能性
http://www.atmarikit.co.jpfarc/rensai/test01/test01.html

8) モデル駆動開発とその周辺
http://www.metabolics.co.jp/mmw/excutable-knoledge/model-driven-development/30e230eb9

9) 二つの合理性と日本のソフトウェア工学
神戸大工学部　客員研究官　林　晋

10)生産性の飛躍的向上をめざし包括的再利用の実現にチャレンジ
組込みソフトウェア2006　p188～199
日経E／日経BYTE

11)ソフトウェア工学の勧め
東京情報大学総合情報学部環境情報学科教授
玉置彰宏

12)ソフトウェア工学
鹿児島大学 工学部 情報工学科
渕田孝康

13)200 1 年SDP(Software Design and Productivity)
米国情報技術研究開発政策