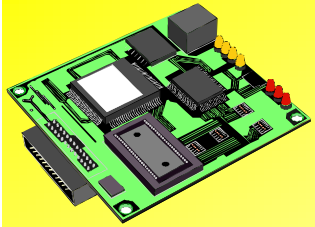
組込み・ユビキタスネットワークシステム開発におけるソフトウェア工学の展開

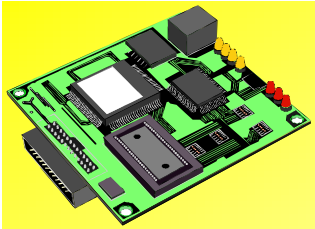
青山 幹雄
南山大学 数理情報学部 情報通信学科
mikio.aoyama@nifty.com
http://www.seto.nanzan-u.ac.jp/~amikio/NISE/

2003年10月1日



シナリオ

- 今，そこにある機会と危機
- 組込みソフトウェア工学とは
- わが国の組込みソフトウェア工学の課題
- まとめ



今，そこにある機会と危機
PC時代から組込み・ユビキタスネットワーク時代へ

- ユビキタス化・ネットワーク化: 情報技術の転換点
  - ユビキタス化: PC(数億台/年)から組込み(100億台/年) の新市場へ
  - すべてがネットワーク化: システムが点から面へ[個人から社会へ]
- 巨大なビジネス機会

ケータイからメガバンクまで

ネットワーク[企業・公共ソフトウェア・サービス]

ユビキタス(Ubiquitous)・組込み(Embedded)

インターネット

インターネット(Web)

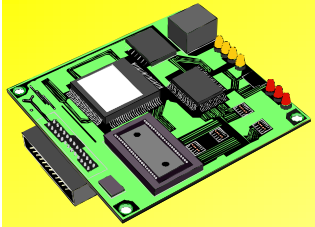
ユビキタス・ネットワークの時代 (2000年代) [100億台/年]

パソコンの時代(大衆化:80-90年代)[1億台/年]

ダウンサイジング

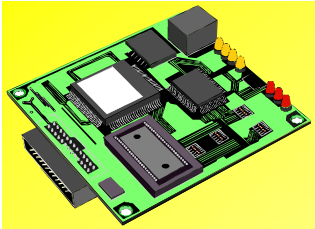
メインフレームの時代 (60-70年代)[1万台/年]

参考文献：野村総合研究所(編), ユビキタス・ネットワークと新社会システム, 野村総合研究所, 2002.



今，そこにある機会と危機
ソフトウェア開発とビジネスの転換と進化

- メインフレーム(70-80年代): 重厚長大システム
  - 大規模基幹システムの受託開発: 工数(人月)ビジネス
    - 低収益, 人件費の増大に対応不可 ⇒ 外注化[人件費削減] ⇒ 限界
  - 大規模, 高品質, ハードウェアで収益を稼ぐ
- PCとクライアント/サーバ(80-90年代): 軽薄短小システム
  - PCのソフトウェアパッケージ製品: 大量生産ビジネス
    - 高リスク高リターン: ベンチャ型(米国優位性発揮), 日本の立遅れ
  - 早期市場占有, 低価格，ほどほど品質(Good Enough Quality)
- ユビキタスネットワーク(2000年代): 社会(コミュニティ)システム
  - ソフトウェアのサービス化: ユーティリティビジネス
    - 仲介（ゲートウェイ）型: ユーザ(社会・企業)主導[オンデマンド]
  - 強い寡占モデル(少数ベンダ)とニッチ, 安全性・信頼性

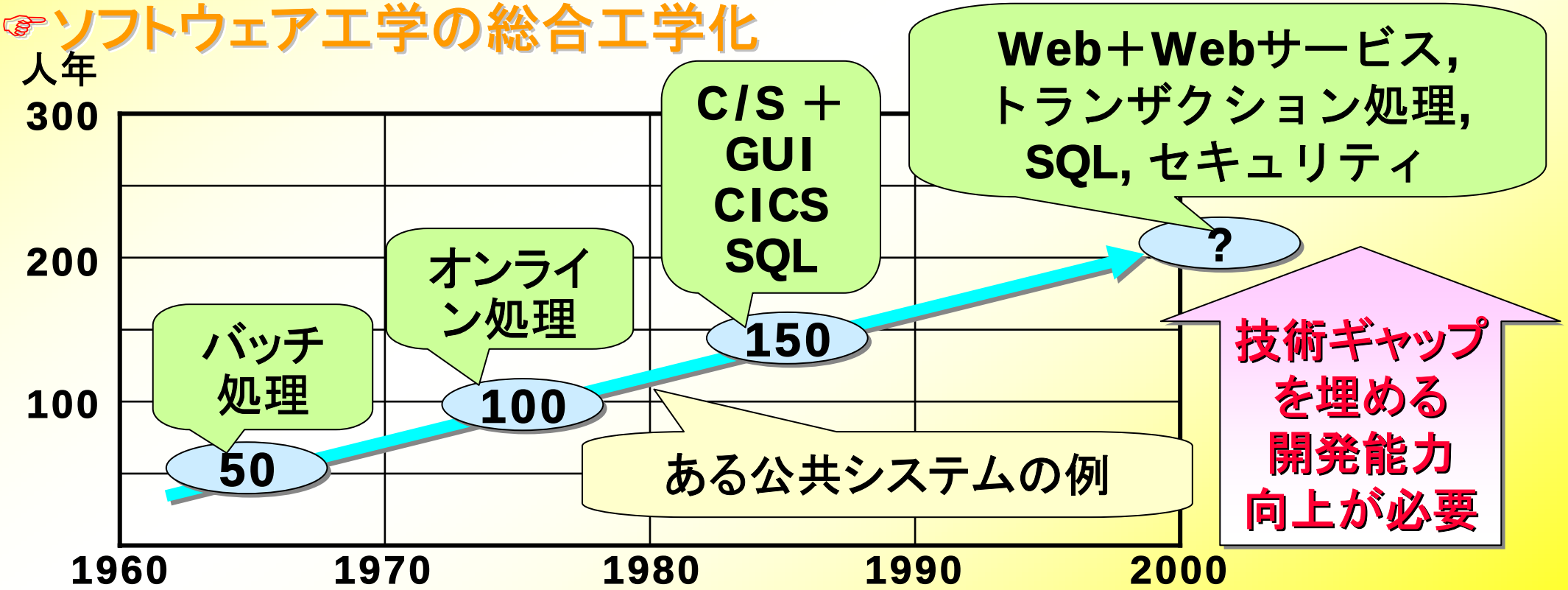


今，そこにある機会と危機
ソフトウェア開発の高度化とリスク増大

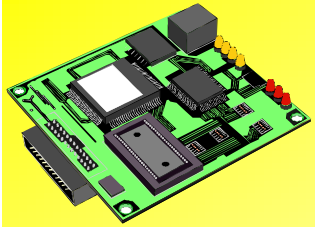
☞開発技術の高度化とリスクの増大: 技術障壁の高まり

- 同一「機能」システムの開発でより高度な技術が必要
- 技術能力による淘汰・市場分化，寡占化
  - 例: オブジェクト指向技術 ⇒ 分散オブジェクト環境(ミドルウェア)

☞ソフトウェア工学の総合工学化



参考文献: P. T. L. Lloyd and G. M. Galambos, Technical Reference Architectures, IBM Systems Journal, Vol. 38, No. 1, 1999, pp. 51-75.



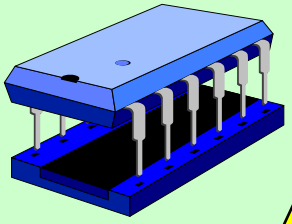
今，そこにある機会と危機
組込みソフトウェア開発の変化と危機

☞組込みソフトウェアの「危機」⇒ 開発の仕組みを変える

- 「危機」の連鎖: メインフレーム ⇒ PC ⇒ 組込み
  - 組込みソフトウェア開発の現状は60-70年代のメインフレーム
- 人海戦術の破綻: ソフトウェア開発の仕組みを変える

いつか
来た路?…

従来型
組込みソフトウェア



ソフトウェア＝
ハードウェアの一部

大規模・複雑化

規模増大 ⇒数100K～M

戦略的転換

個人技から工学へ

オープン化

垂直統合から水平統合へ

戦略的転換

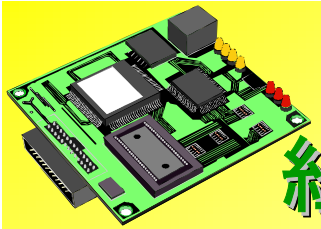
単一システムから複合システムへ

社会基盤化

あらゆる機器へソフトウェアが浸透

戦略的転換

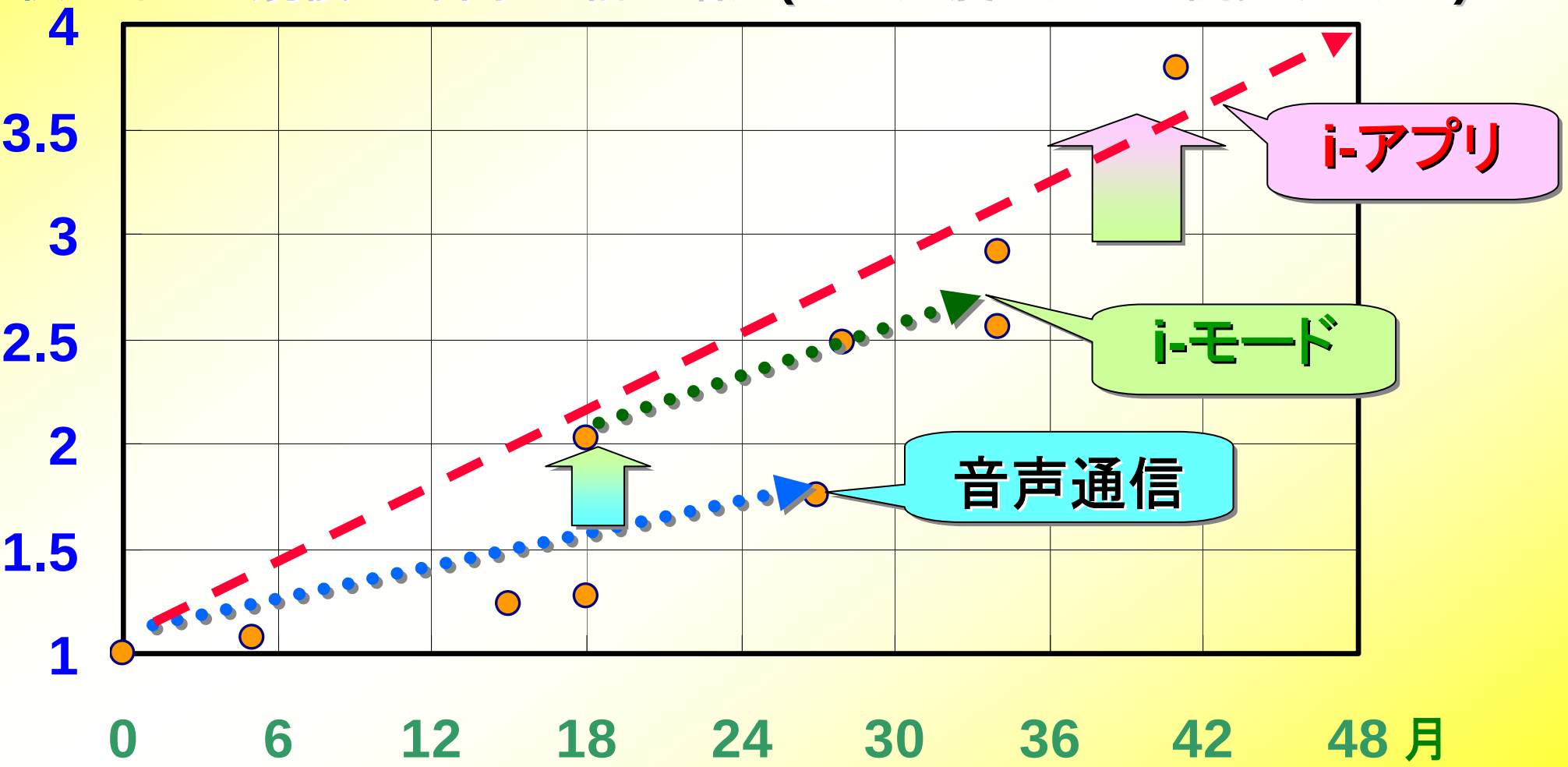
社会基盤としての信頼性・安全性



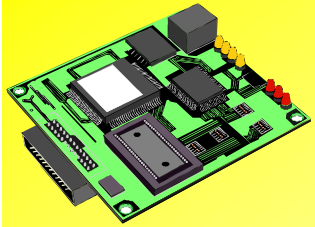
今，そこにある機会と危機
組込みソフトウェア開発の変化と危機: 規模の増大

☞携帯電話: 規模の飛躍的増大と開発期間の短縮

規模比率 ☝ 規模: 4年間で4倍に増大(進化速度の速さが開発リスクに)



参考文献: M. Aoyama, Continuous and Discontinuous Software Evolution: Aspects of Software Evolution across Multiple Product Lines, Proc. IWPSE 2001, ACM Press, Sep. 2001, pp. 87-90.



今，そこにある機会と危機
組込みソフトウェアの多様化と分化

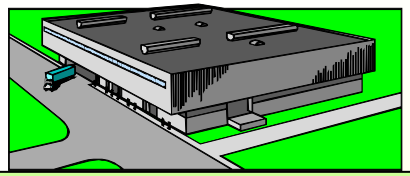
☞ 多様な組込みソフトウェア

- 対象の機器・製品分野による要求・リスクの多様性

☞「組込みソフトウェア」一般で捉えることの難しさ: ドメイン分化

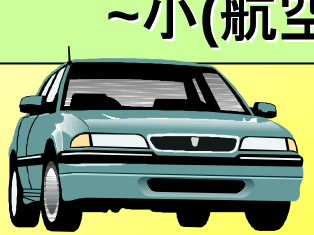
- 通信ソフトウェア工学(Telecommunications Software Engineering)
- 自動車ソフトウェア工学(Automotive Software Engineering)

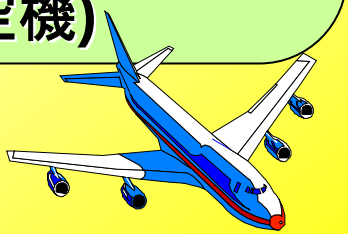
家電製品制御
変更可能性: 不可
稼動連続性: 利用時
リスク: 比較的小
(品質低下)
コスト制約: 大
大量生産

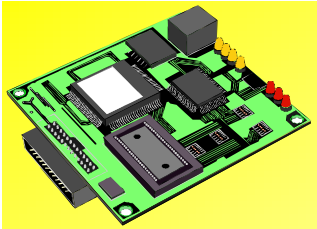


プラント制御・通信
変更可能性: 可/不可
稼動連続性: 運用中(1~20年)
リスクの大きさ: 中
(多数のビジネスに関わる)
コスト制約: 中

運転・飛行制御
変更可能性: 不可
稼動連続性: 運用中(~24H)
リスクの大きさ: 大
(人命にかかわる)
コスト制約: 大(自動車)
~小(航空機)







今，そこにある機会と危機
組込みソフトウェアの多様化と分化

☞組込みソフトウェアの適用領域の拡大と多様化

PCの主たる適用領域

大型機器

航空機用

プラント用

通信機器用

自動車用

情報家電

携帯情報機器
パームトップPC,
携帯電話

従来の組込み
ソフトウェアの主たる
適用領域

パーソナル

(組込み)機器

ゲーム

カーナビ

組み込み機器制御用ソフトウェア部品

より小規模なハードウェア制御の
コンピュータによる代替の拡大

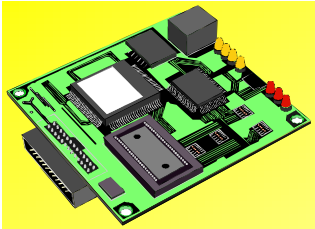
ICカード

IP(Intellectual Property)
ハードウェア部品とソフトウェア部品を
一体化した部品化

SoC

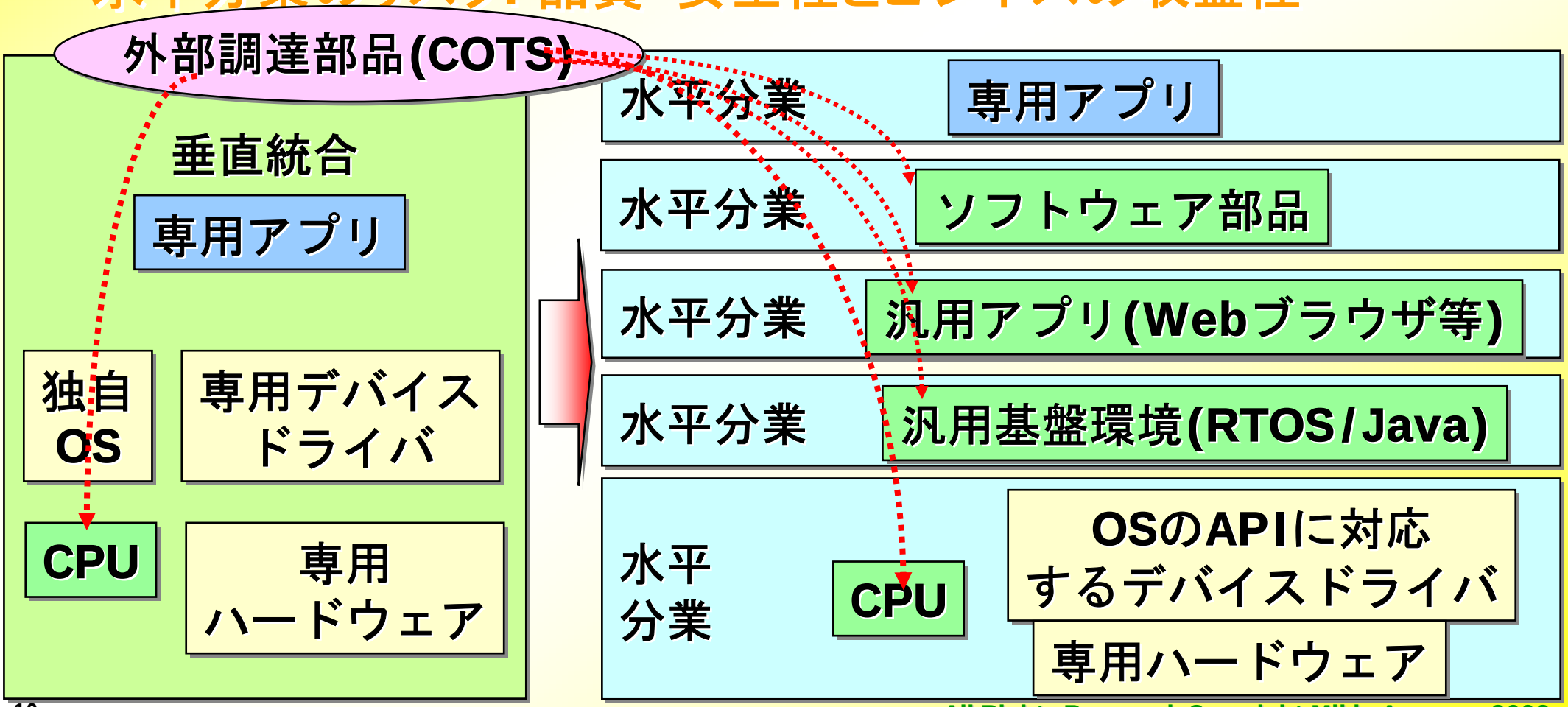
小型機器

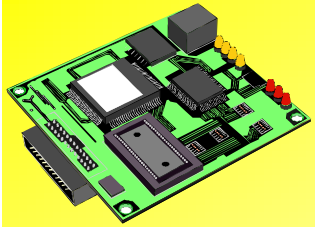
従来ハードウェアで制御していた適用領域



今，そこにある機会と危機
開発モデルの転換: 垂直統合から水平分業へ

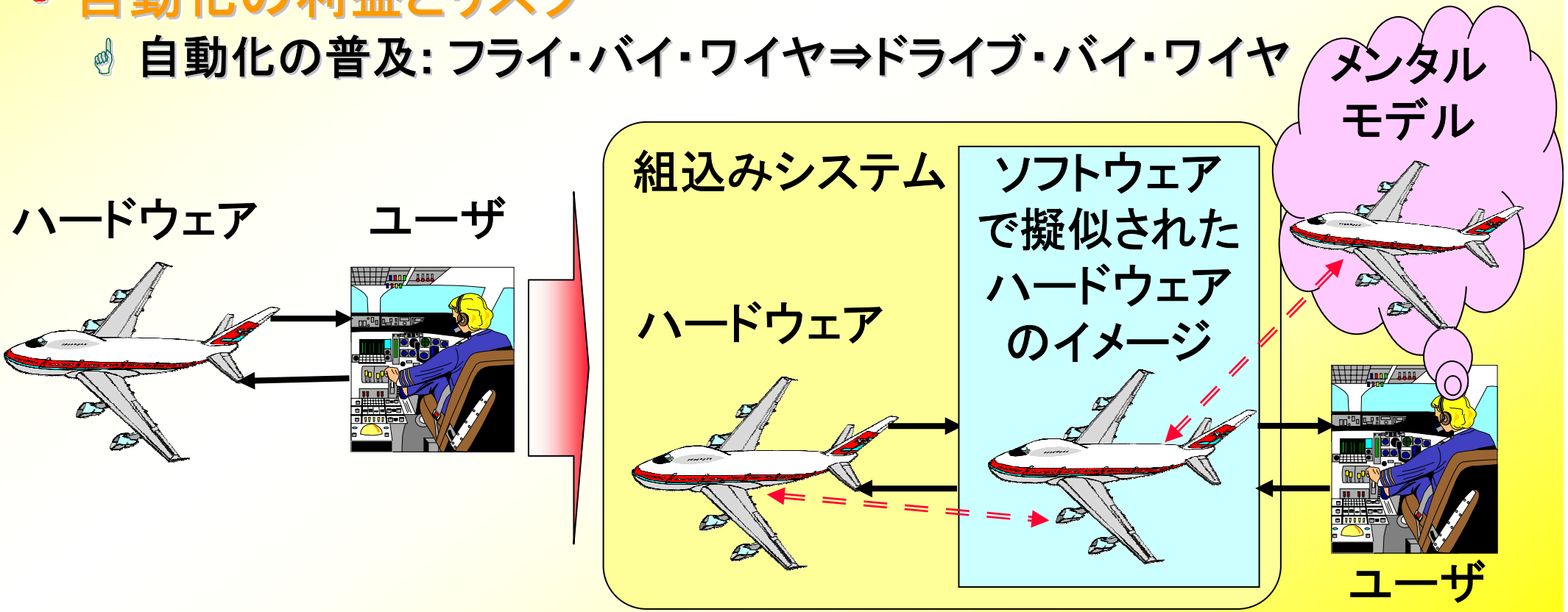
- 垂直統合から水平分業へ: 開発モデルの転換
  - 市販部品(COTS: Commercial Off-The-Shelf)を再利用した開発
    - 例: Java, Webブラウザ, リアルタイムOS（RTOS)の導入
- 水平分業のリスク: 品質･安全性とビジネスの収益性

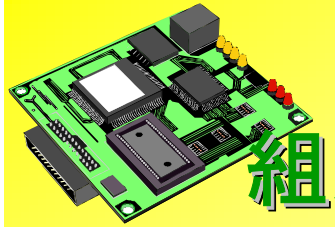




今，そこにある機会と危機
組込みソフトウェアのユーザインタフェース問題

- コンピュータの存在を意識していないユーザ
  - ユーザ＝コンピュータの非専門家
- 組込みソフトウェア=実世界の擬似
  - 実世界, ソフトウェアのモデル, ユーザのメンタルモデル
- 自動化の利益とリスク
  - 自動化の普及: フライ・バイ・ワイヤ⇒ドライブ・バイ・ワイヤ

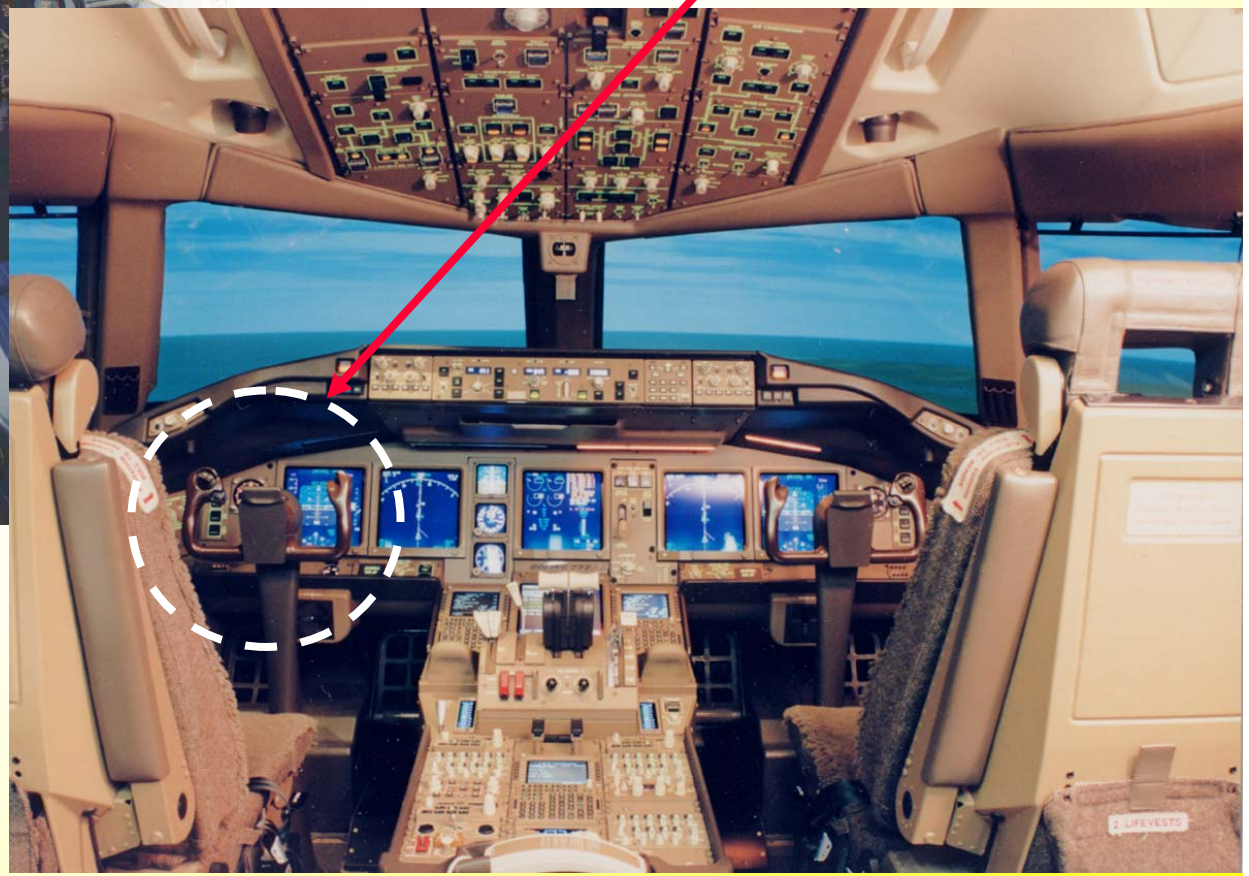




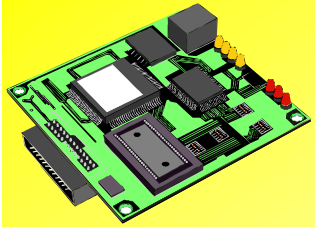
今，そこにある機会と危機
組込みソフトウェアのユーザインタフェース問題:事例

エアバスA320のコクピット
フライ・バイ・ワイヤによる自動化
とジョイスティックの使用

ボーイング777のコクピット
フライ・バイ・ワイヤでも
操縦桿を使用



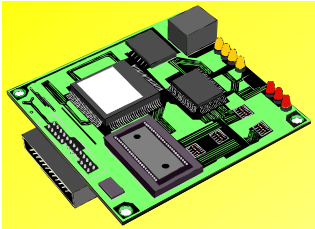
参考文献:青山幹雄(編著), 航空とIT技術,
共立出版, 2002.



今，そこにある機会と危機
品質(安全性・安心)への回帰

- 「ほどほど品質(Good Enough Quality)」から「真の品質」へ
  - ソフトウェアの社会的リスクの認識
    - 「我々は脆弱な大規模ソフトウェアに依存する危険な状態にある」*
  - セキュリティ問題
- 社会全体がソフトウェアの生産性・品質(リスク)に依存
  - ソフトウェア(工学)が社会生活の質(Quality of Life)を規定する
  - 組込みソフトウェアは社会基盤
- ソフトウェアリスクの分類
  - 開発リスク: 予定のコスト, 期間で開発できない
  - 運用リスク: 運用停止, 情報セキュリティの毀損など
  - 進化(保守)リスク: 運用中のソフトウェアを変更できない

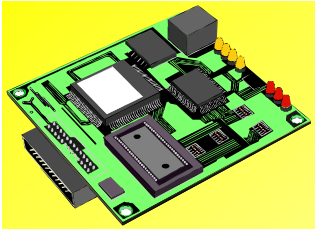
*参考文献: President's Information Technology Advisory Committee, Report to the President, Information Technology Research: Investing in Our Future, Feb. 1999.



今，そこにある機会と危機
大規模ソフトウェアシステムの問題事例とリスク

- 国内における大規模ソフトウェアシステムの問題事例とリスク
  - 近年の金融，交通，通信などにおける問題発生
- 米国政府における大規模ソフトウェア開発破綻事例とリスク調査
  - 連邦航空局(FAA)システム更新[GAO報告書(1999)](4大事例の一つ)*
    - 計画: 予算=$2.5B(約3,000億円)，1996年完了予定
    - 実績: 経費=$45B(約5.4兆円), 2004年?(経費20倍, 10年遅れ)
  - ソフトウェアの不十分な検証による社会損失
    - NIST報告書(2002): $59.8B(7.2兆円)/年[GDPの0.6%]損失**
    - 日本のGDP=約500兆円(2002年度）に換算: 3兆円/年の損失
- キュリティ問題: セキュリティ問題の主因はソフトウェア設計***
    - 47%＝入力検証誤り(バッファオーバフロー含む), 26%=他の設計誤り

参考文献:
*GAO, Observations on FAA's Air Traffic Control Modernization Program, AIMD-99-137, 1999.
**NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing, 2002.
***E. H. Spafford, Relating Software Engineering and Information Security, ICSE (International Conference on Software Engineering) 2003 Presentation, May 2003.

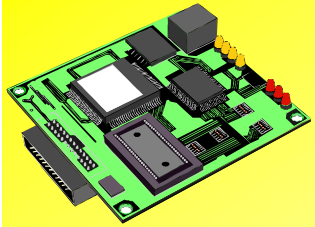


今，そこにある機会と危機
組込みソフトウェアの失敗事例

- 医 療
  - 放射線治療装置 Therac-25の事例*
    - カナダAECL社製造, 北米の病院で利用
    - ’85-’87年:バグによる6人の過剰被爆, 内3人死亡 ⇒使用停止に
- 通 信
  - 交換機の停止
  - 携帯電話の回収
- 交 通
  - (航空管制システムの停止)
  - 航空機: パイロットとのユーザインタフェース問題
  - 自動車: シートベルト制御ソフトウェアのバグによる幼児死亡
- 宇宙・防衛
  - アリアン-5 ロケット: プログラム更新における誤りで打ち上げ失敗
  - パトリオットミサイル: 継続時間処理誤りにより湾岸戦争で迎撃失敗

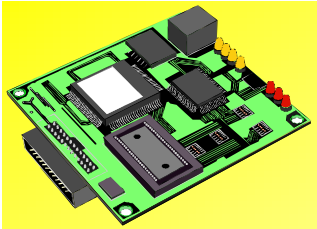
安全性の問題は致命的
になる場合がある

*参考文献, N. Leveson, Safeware, Addison Wesley, 1995.
The Risk Digest, http://catless.ncl.ac.uk/Risks/.



シナリオ

- 今，そこにある機会と危機
- 組込みソフトウェア工学とは
- わが国の組込みソフトウェア工学の課題
- まとめ



組込みソフトウェア工学とは
組込みソフトウェアとは

- 多くの制約がもたらす設計の複雑さと困難さ
  - 非機能的特性(時間制約, 高信頼性): 機能横断的設計条件
  - リソース制約: 従来の設計技術の対象外
- 「開発(能)力」,「設計(能)力」が問われる: 量から質へ

連続性(信頼性)
多数の入出力
(多重処理)

時間制約

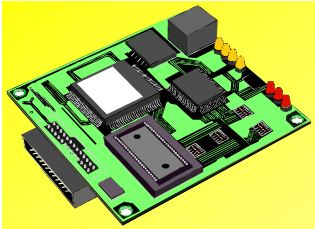
高信頼性,
プログラム
変更不可,
リソース制約
(メモリ/CPU),

リアクティブソフトウェア
(外部信号に反応して何らかの
制御を行う)

リアルタイムソフトウェア
(時間制約のあるソフトウェア)

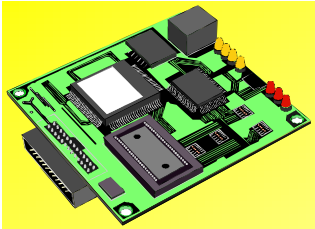
組込みソフトウェア
(機器に組み込まれるリアクティブ
ソフトウェアで通常リアルタイム性
を要求される)

データ変換
ソフトウェア
(データを
変換処理)
[業務処理]
メイン
フレーム,
PCの
ソフトウェア



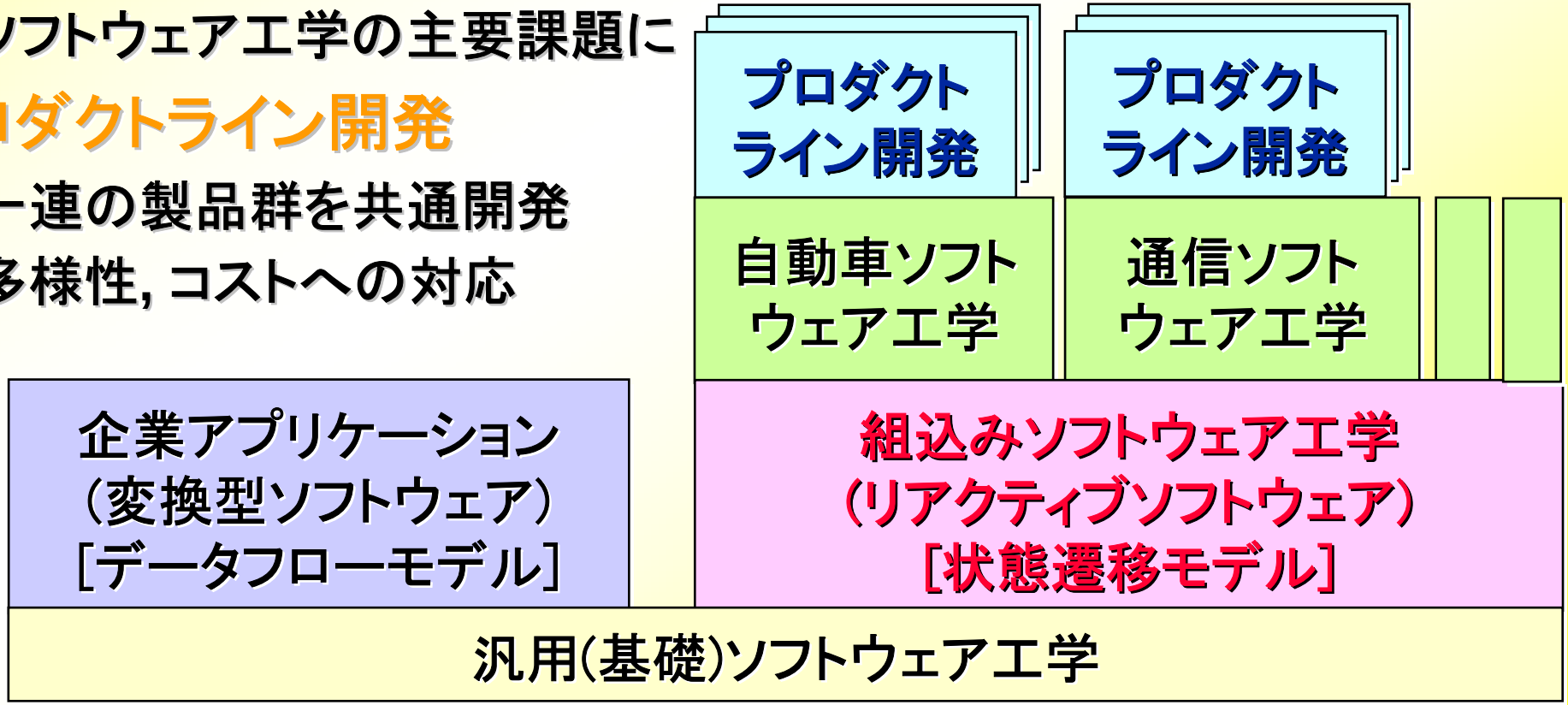
組込みソフトウェア工学とは
ソフトウェア工学の3つの枠組み

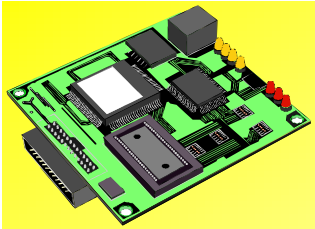
ソフトウェア開発＝プロセス(作り方・手順: How)
＋プロダクト(作る物の構造: What)
＋ピープル(作る人: Do)



組込みソフトウェア工学とは
組込みソフトウェア工学の位置づけ

- ソフトウェア工学の深化と分化
- ドメイン指向ソフトウェア工学としての組込みソフトウェア工学
  - ドメイン: 対象とする問題やソフトウェア(アプリケーション)の集まり
  - ドメイン指向: ドメインの特性に対応できる開発技術の深化
  - ソフトウェア工学の主要課題に
- プロダクトライン開発
  - 一連の製品群を共通開発
  - 多様性, コストへの対応





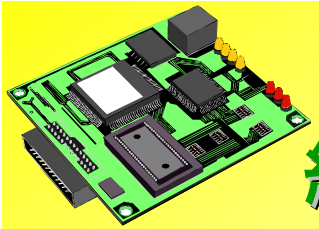
組込みソフトウェア工学とは
ソフトウェア工学の進化と総合化

☞ソフトウェア工学: 90年代の急速な進化・総合化

☝ 技術の体系的開発・評価・導入の仕組みが必要

国内の対応?

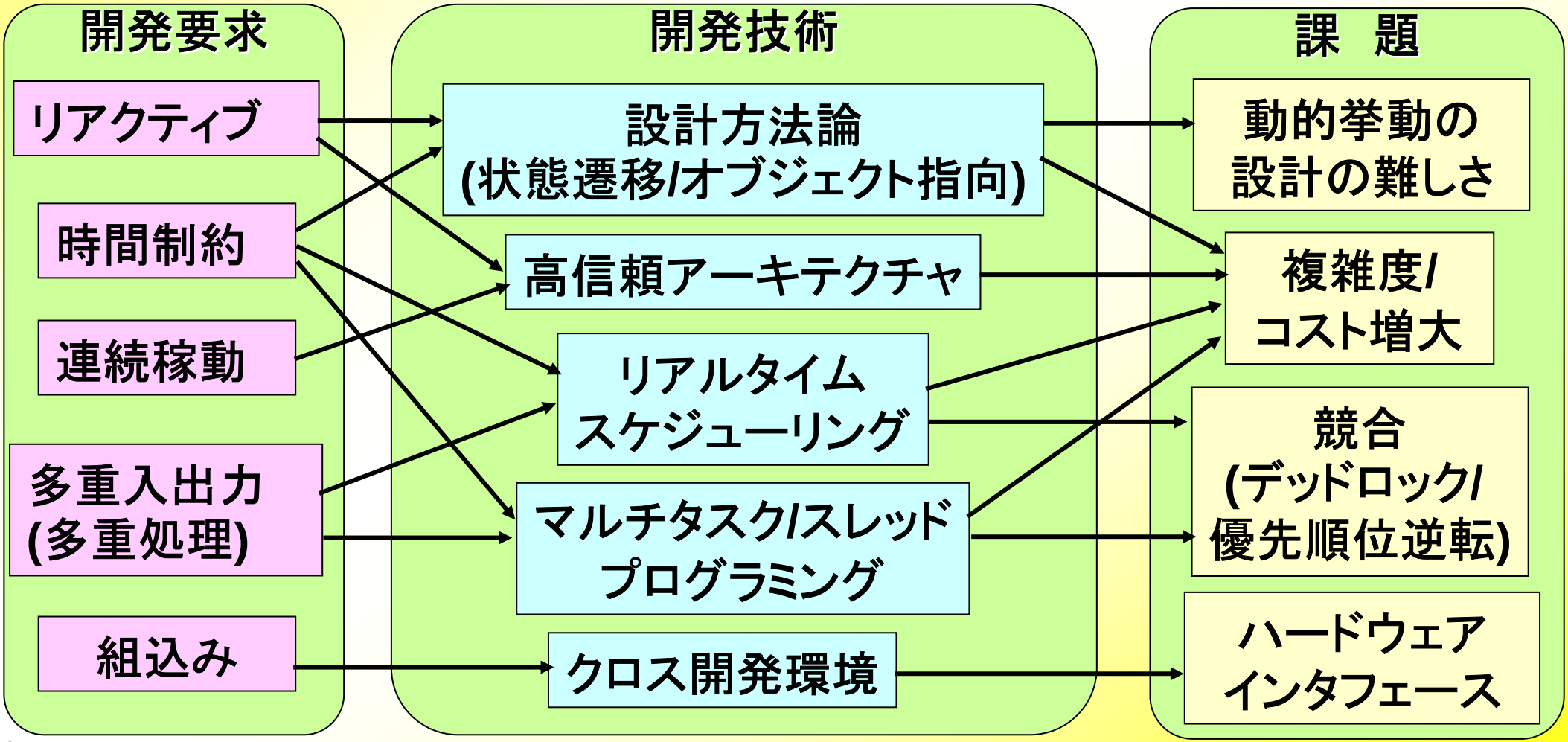
	80-84	85-89	90-94	95-99	00-
開発・プロダクト技術	オブジェクト指向プログラミング		オブジェクト指向分析/設計・要求工学		
			UML		
			コンポーネント指向・サービス指向		
			Web / Java / XML		
			アーキテクチャ・フレームワーク・パターン		
			ドメイン指向	プロダクトライン	
プロセス技術	ソフトウェアファクトリ		プロセス工学(構造と改善CMM/CMMI))		
			プロジェクト管理(PMBOK)		ISO-15504
					ISO-10006
研究組織			SEI(米)		PITAC
			ESPRIT(~98)(欧)		FraunhoferIESE(独)

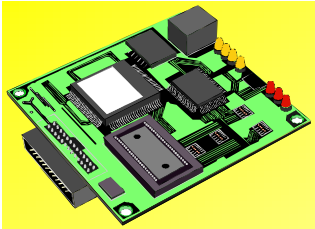


組込みソフトウェア工学
組込みソフトウェア工学への要求と開発技術・課題

☞組込みソフトウェア固有の開発技術と課題

- 対象ドメインの知識 × ソフトウェア工学
- ドメインの特性に応じた開発技術の体系化





組込みソフトウェア工学とは
組込みソフトウェア開発のプロセスと開発技術

システム分析(ハードウェア・ソフトウェア機能分割)

ハードウェア開発

ソフトウェア開発

分析 → 設計 → 実装 → 試験

要求工学

組込み(リアルタイム)オブジェクト指向

性能設計[性能工学]

組込み設計・実装技術(メモリ/処理能力制約)

マルチタスク/スレッドプログラミング

リアルタイムスケジューリング(RTOS等)

品質評価(信頼性/性能評価)

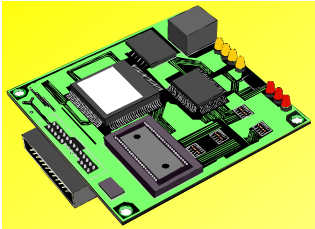
高信頼性ソフトウェアアーキテクチャ設計(冗長設計, 例外/障害処理, 動的再構成, ソフトウェア監査)

クロスコンパイラ

クロス試験(シミュレータ/クロスデバッガ)

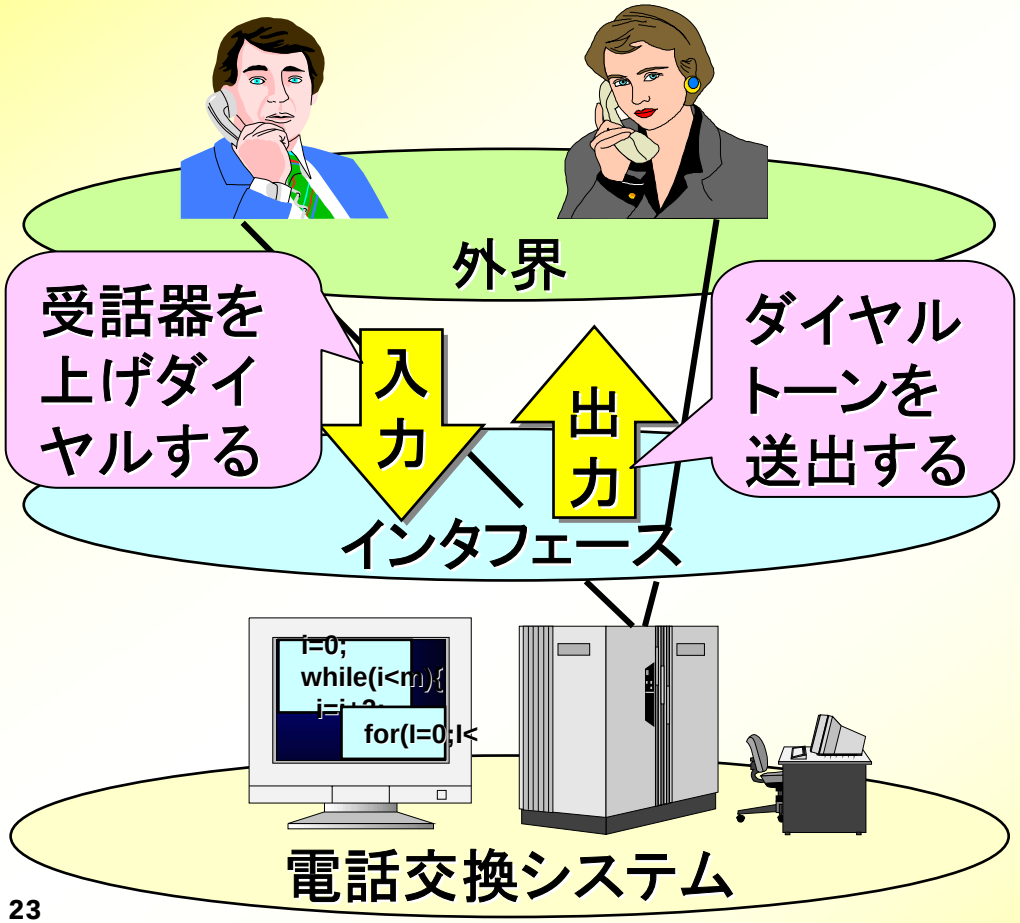
実機試験(ICE, JTAGなど)

検証(モデル検査)

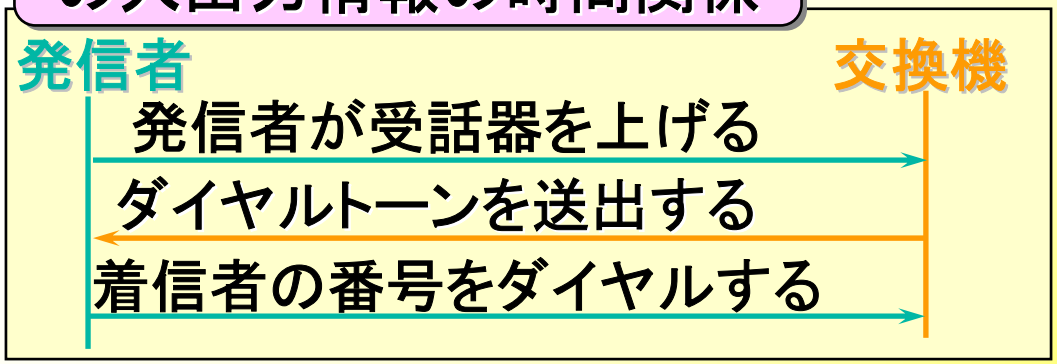


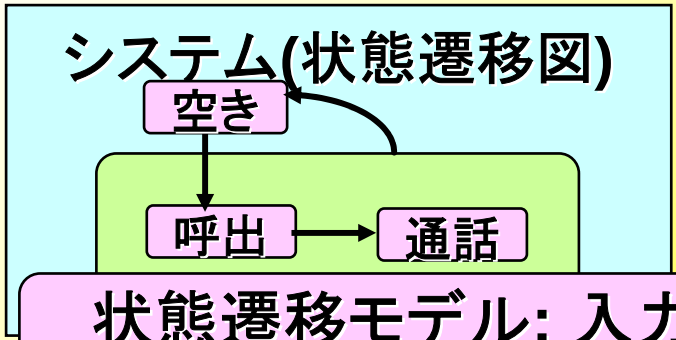
組込みソフトウェア工学とは
開発技術: 分析・設計方法論

- 組込みソフトウェア＝外界の擬似: 状態遷移/相互作用モデル
- 分析: 良いモデルの構築
  - ハードウェアなどの制約・不確定性

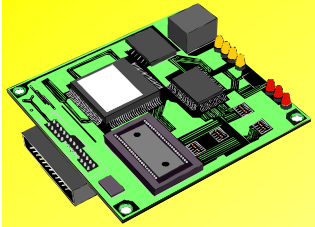


相互作用モデル: 外界との入出力情報の時間関係





状態遷移モデル: 入力から出力に至るシステムの挙動



組込みソフトウェア工学とは
開発技術: 複数視点による良いモデル化の方法

☞複数視点によるモデル化

- ☝ 複雑なシステムの正しいモデル化には複数の視点が必要

システムの果たす機能とその処理の流れ(何を)

構造の視点
(クラス図)

電話機

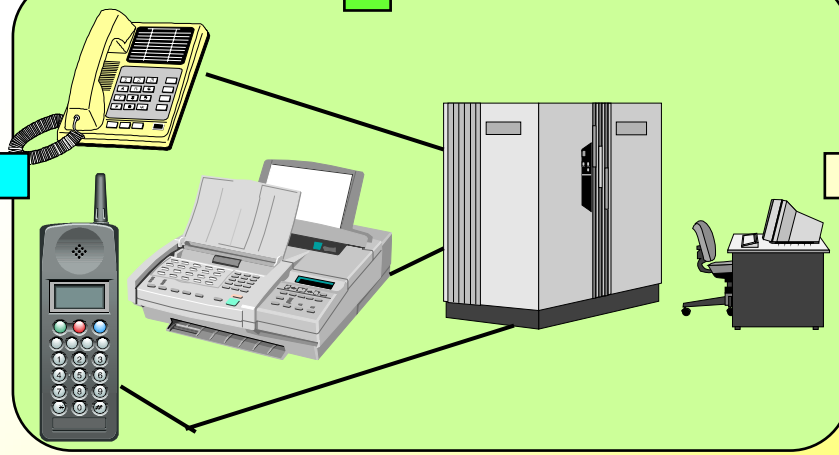
一般電話　携帯電話

システムの構成要素とその関係(誰が・何が)

システムの実行順序とそのタイミング(いつ)

機能の視点
(データフロー図)

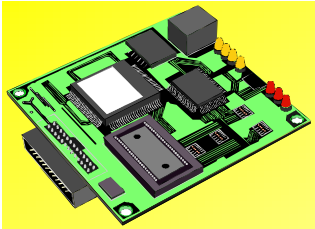
発信処理　課金処理



挙動の視点
(状態遷移図)

空き

呼出 → 通話



組込みソフトウェア工学とは
開発技術: 複数視点に対応するモデル表現方法

☞複数視点を活かした良いモデルの構築力とその表現方法

- 標準表記法: 組込み向けUML
- モデル構築力
  - ドメインの知識

システムの構成要素とその関係(継承・集約)[だれ，組織]

構造の視点

データモデリング

E-R図

システムの果たす機能とその処理の流れ[何を]

システムの実行順序とそのタイミング[いつ]

クラス図

機能の視点

構造化分析・設計

DFD

アクティビティ図

UML

挙動の視点

シーケンス図

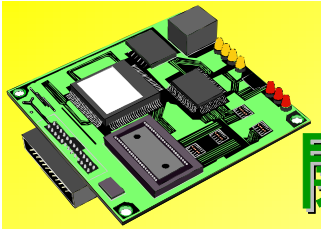
コラボレーション図

ステートチャート

リアルタイム構造化分析・設計

STD

状態遷移表



組込みソフトウェア工学とは
開発技術: 組込みオブジェクト指向ソフトウェア開発

- オブジェクト指向: 良いモデル化/モジュール化の技術
- 組込みオブジェクト指向 = オブジェクト指向 × 組込み開発技術

高度な開発力

組込みソフトウェア開発

オブジェクト指向
オブジェクト指向概念
分析・設計方法論
カプセル化
コンポーネント

組込みオブジェクト指向ソフトウェア開発

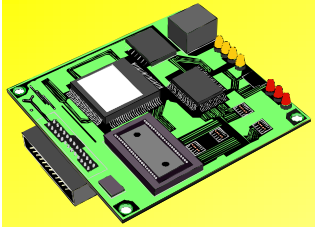
組込み向け開発技術
状態遷移モデル
マルチタスク／スレッド
タイミング／性能設計

組込み向け開発技術
状態遷移モデル
マルチタスク／スレッド
タイミング／性能設計

構造化／機能指向ソフトウェア開発

オブジェクト指向
オブジェクト指向概念
分析・設計方法論
カプセル化
コンポーネント

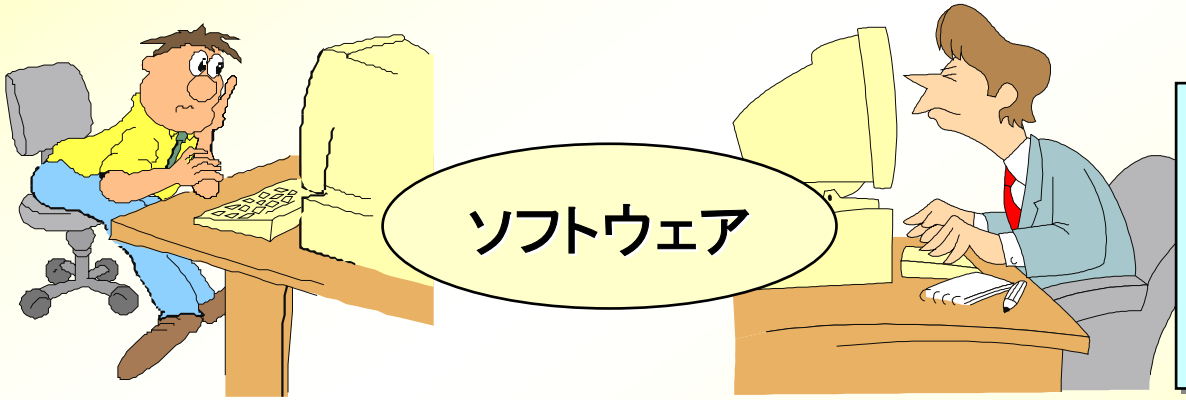
オブジェクト指向ソフトウェア開発



組込みソフトウェア工学とは
開発技術: 品質設計

- 欠陥除去から品質の作りこみ(品質設計)へ
  - 組込みソフトウェア開発:「工数の50%以上が試験工数」?
- 「設計力」が生産性と品質の源泉

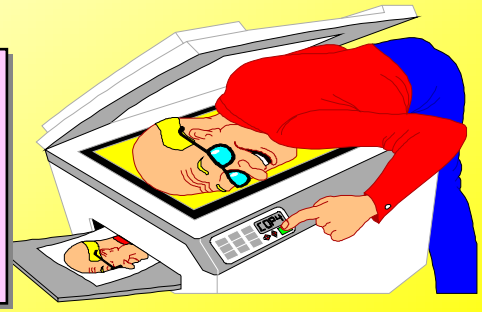
利用者の視点
「使いの質」:
信頼性,
使用性
(使い勝手)
など



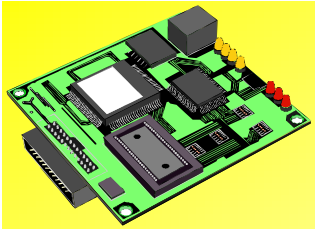
開発者の視点
「作りの質」:
理解容易性,
保守容易性など

・良い設計とは

- **当たり前品質**: 満たすべき最低限の品質
- **魅力的品質**: 付加価値を与える品質

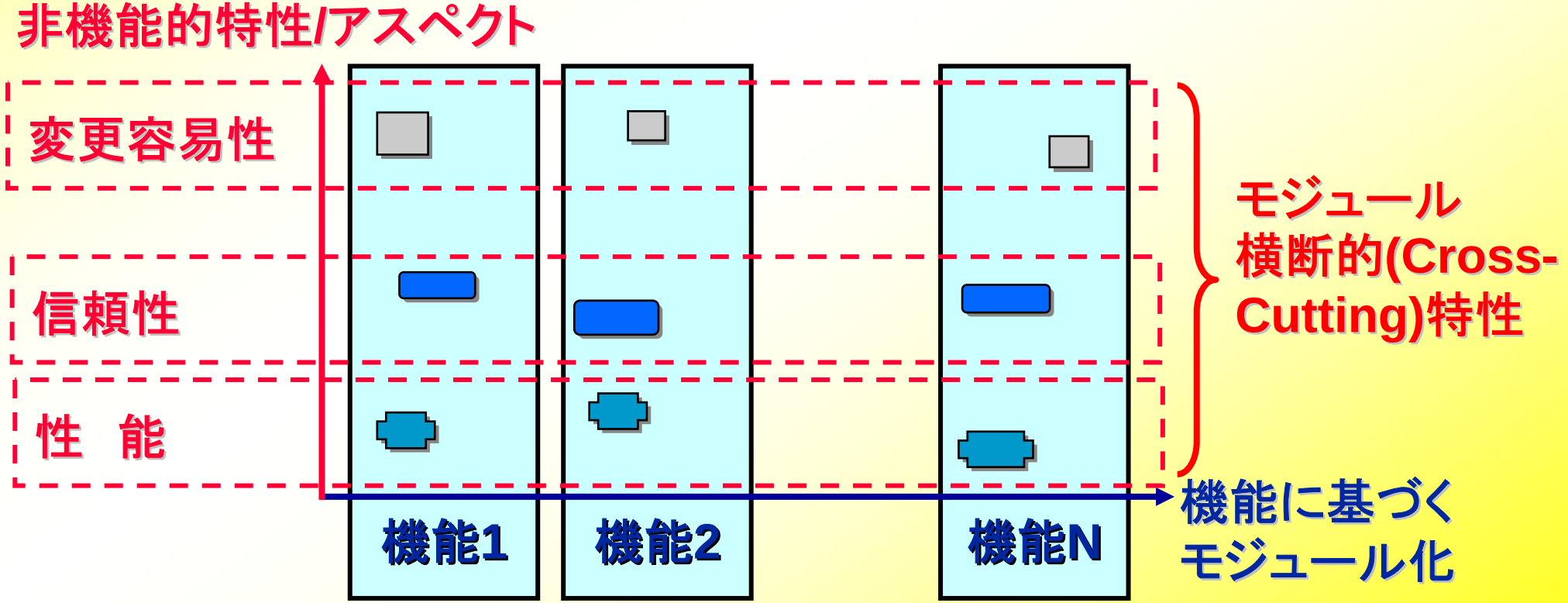


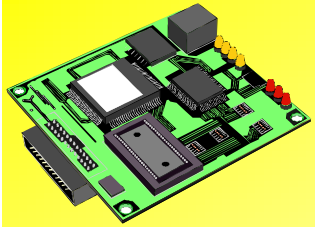
品質向上のアプローチ
- 高品質ソフトウェアの設計: 品質の作り込み
- 欠陥除去: 設計で混入したバグを除去



組込みソフトウェア工学とは
開発技術: アスペクト指向によるモジュール化

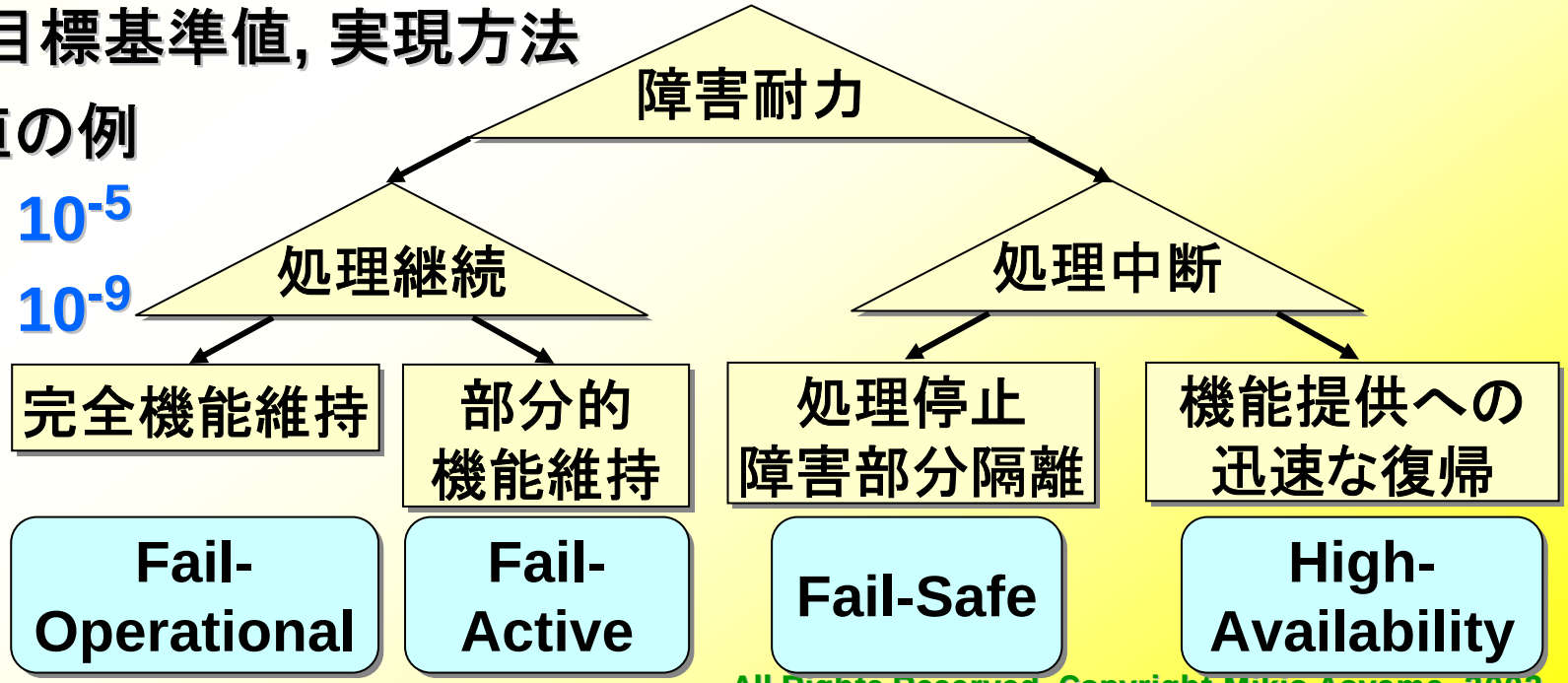
- 非機能的特性(品質)の設計の困難さ
- アスペクト指向ソフトウェア開発 AOSD (Aspect-Oriented Software Development)
  - アスペクトによる多次元モジュール化: 非機能特性によるモジュール
  - オブジェクト指向を補う

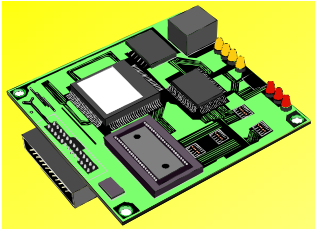




組込みソフトウェア工学とは
開発技術: 信頼性と安全性の設計

- 信頼性(Reliability)と安全性(Safety)の要求
  - 信頼性: 機能を提供すること(結果は問わない) ⇒処理の連続性
    - 障害を発生させない, 障害が発生しても迅速に復旧可能
  - 安全性: 結果が悪影響をもたらさない
    - 誤りの防止, 誤りに対して悪影響を防止(インターロックなど)
- 信頼性・安全性のモデル化と実現: システム個別の要求
  - モデル化, 目標基準値, 実現方法
  - 目標基準値の例
    - 交換機: 10^{-5}
    - 旅客機: 10^{-9}

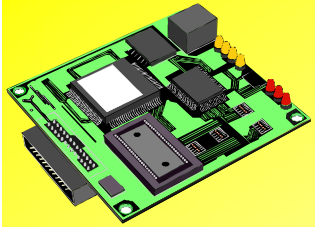




組込みソフトウェア工学とは
開発技術: 検証・試験技術

- 静的解析と動的実行の相互補完
- 静的解析: ソースプログラム, 仕様の解析(擬似実行を含む)
  - HASE (High Assurance System/Software Engineering)
  - コンピュータ能力を活用した検証: モデル検査(Model Checking)
- 動的試験: プログラム実行
  - 試験項目の設計・実行評価・再利用のツール支援

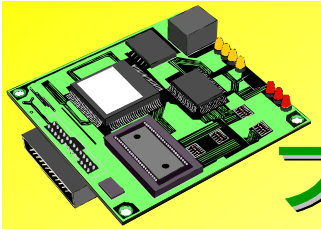
<table>
<tr><th>工程</th><th colspan="2">受動的(Passive)除去技術</th><th>能動的(Active)除去技術</th></tr>
<tr><td>分析
設計
実装</td><td rowspan="2">・レビュー: インスペクションとウォークスルー(要求仕様, 設計, プログラム, 試験仕様など)</td><td>・形式的検証 : 仕様の形式的検証</td><td rowspan="2">・シミュレーション :
実行可能仕様のシミュレーション,プログラムの実行による動的解析</td></tr>
<tr><td>・静的解析: プログラムソースコードからその構造などの解析を行う</td></tr>
<tr><td>試験</td><td colspan="2">・構造試験(ホワイトボックス試験) : 内部構造を踏まえた試験で単体試験に適用
・機能試験(ブラックボックス試験) : 内部構造によらずモジュールの外部から機能などを確認する試験, 機能試験, 結合試験に適用.
・非機能的試験:性能など試験で, システム試験に適用</td></tr>
</table>



組込みソフトウェア工学とは
開発技術: 検証・試験技術

- ☞状態遷移モデルを基礎とする試験の必要性
- ☞非機能的特性の試験の必要性
- ☞ハードウェアとソフトウェアの結合試験

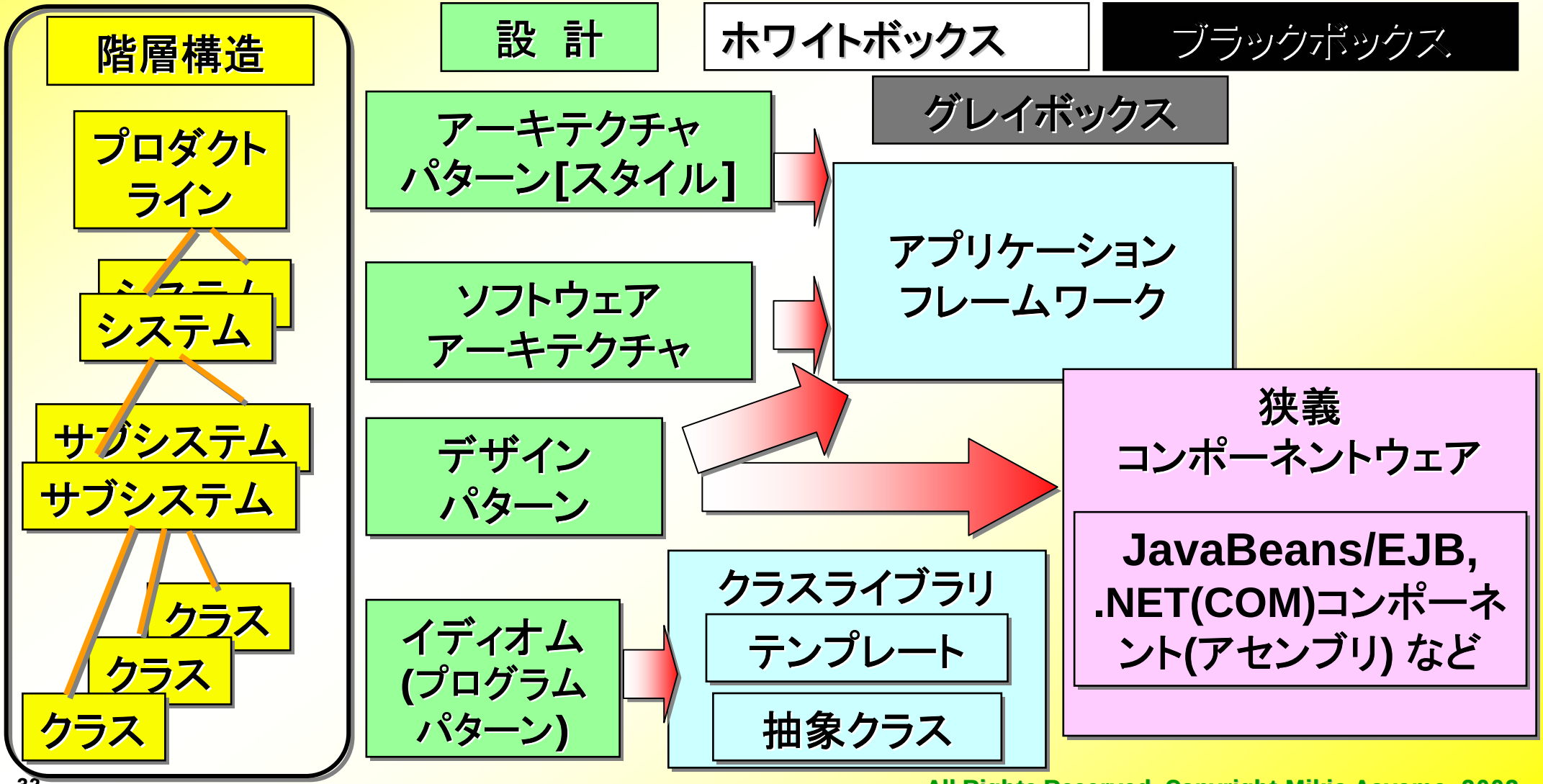
項目	試験内容	試験内容
機能性	システム全体の機能	システム全体として要求された仕様を満たすか
頑健性	過負荷	使用で規定された値以上の負荷に対する耐性
	信頼性	例:連続運転における障害の有無
	回復性	障害からの回復やデータの再生.
	セキュリティ	不法侵入や不法なデータアクセスなどからの保護
互換性	システム互換性	現行のシステムとの互換性
	標準互換性	標準(国際, 業界, 社内)に合致しているか
性 能	スループット	一定時間の処理量[例:トランザクション／秒]
	応答時間	ある操作に対する応答時間[秒]
保守性	ー	障害やシステムの構成変更に対する保守容易性
使い易さ(ユーザビリティ)		ユーザインタフェースを中心に使いやすさを試験

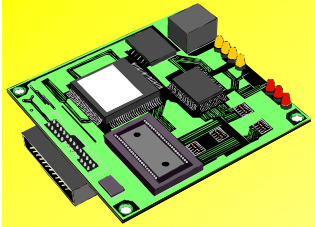


組込みソフトウェア工学とは
プロダクト: ソフトウェアコンポーネントと再利用技術

☞システムの階層とプロセスにより段階的に再利用する仕組み

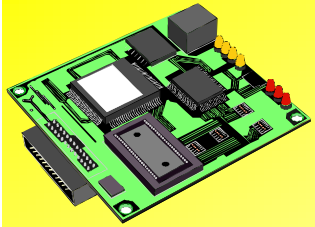
☝ コンポーネント開発とそれを組み合わせたコンポーネント指向開発





組込みソフトウェア工学とは
プロダクト: ソフトウェアアーキテクチャ

- ソフトウェアアーキテクチャ
  - ソフトウェアシステムのマクロ(全体的)な構造
  - アーキテクチャ＝コンポーネント＋関係(コネクタ)
- ソフトウェアアーキテクチャの意義
  - システム全体の構造を規定
    - 振舞い(制御構造)の設計, 情報モデルの設計
  - システムの品質を規定: システムの良さを決める
    - 性能・信頼性の設計
    - 同一機能を実現するソフトウェアアーキテクチャは複数ある
  - 設計規範となる: パターン, スタイル
    - 参照アーキテクチャ: OSI 7層モデル, RM-ODP 等
- 組込み開発におけるアーキテクチャの重要性
  - リアルタイムソフトウェア ⇒ 振舞の設計が鍵
  - 非機能的特性(品質設計)を支配



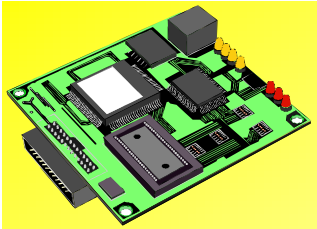
組込みソフトウェア工学とは
プロダクト: デザインパターン

☞デザインパターン: 設計の良い手本

- 問題と解法を対にして示す, 設計の語彙となる

☞デザインパターンを用いたPOSフレームワークの設計

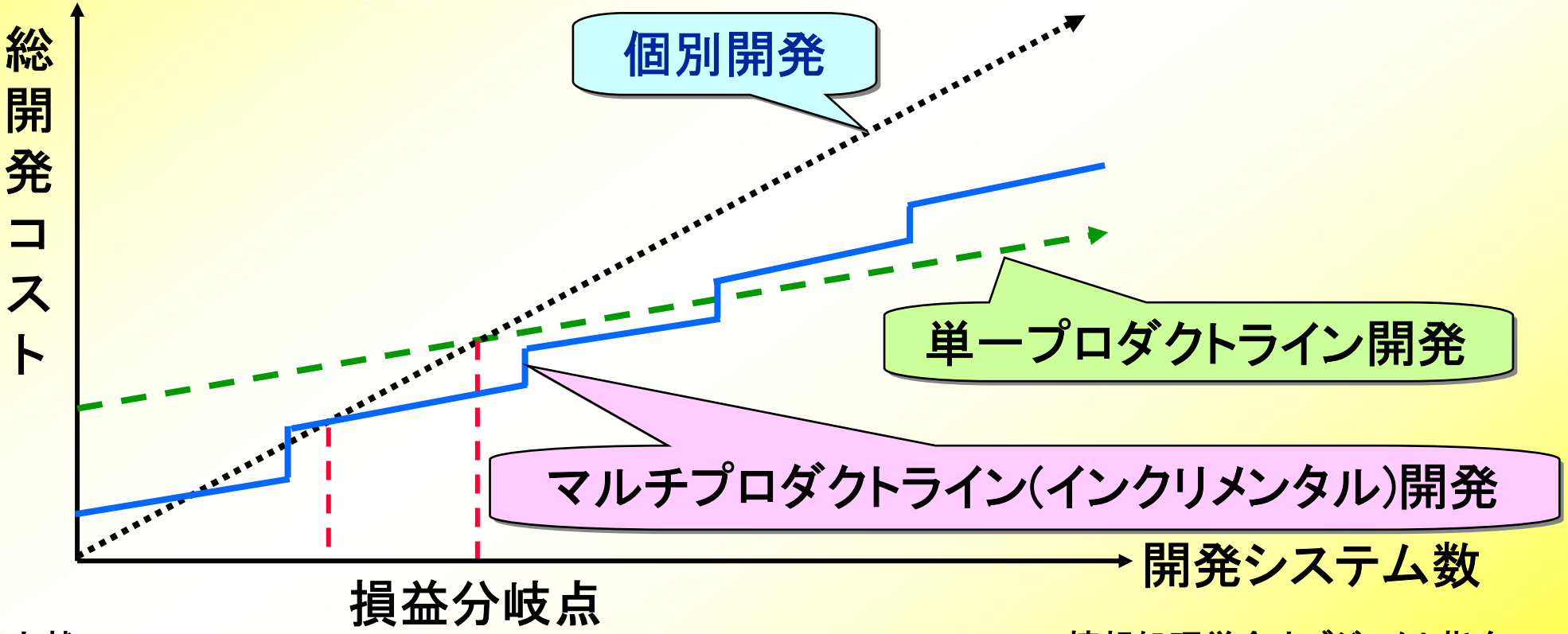
参考文献: 情報技術コンソーシアム(ITC), 次世代コンポーネント
ウェア技術に関する研究開発報告書, Mar. 2000.



組込みソフトウェア工学とは
プロダクト: プロダクトラインソフトウェア開発

☞プロダクトライン開発: システム単位の再利用

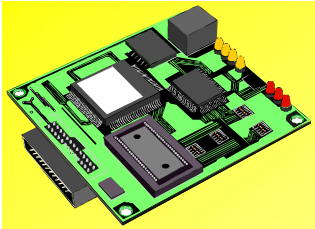
- システムの「流用」を体系的に実現する仕組み
- 大きな再利用効果



参考文献：D. Garlan, Software Architecture: Past, Present, and Future, 情報処理学会オブジェクト指向2000シンポジウム講演資料集, Aug. 2000.

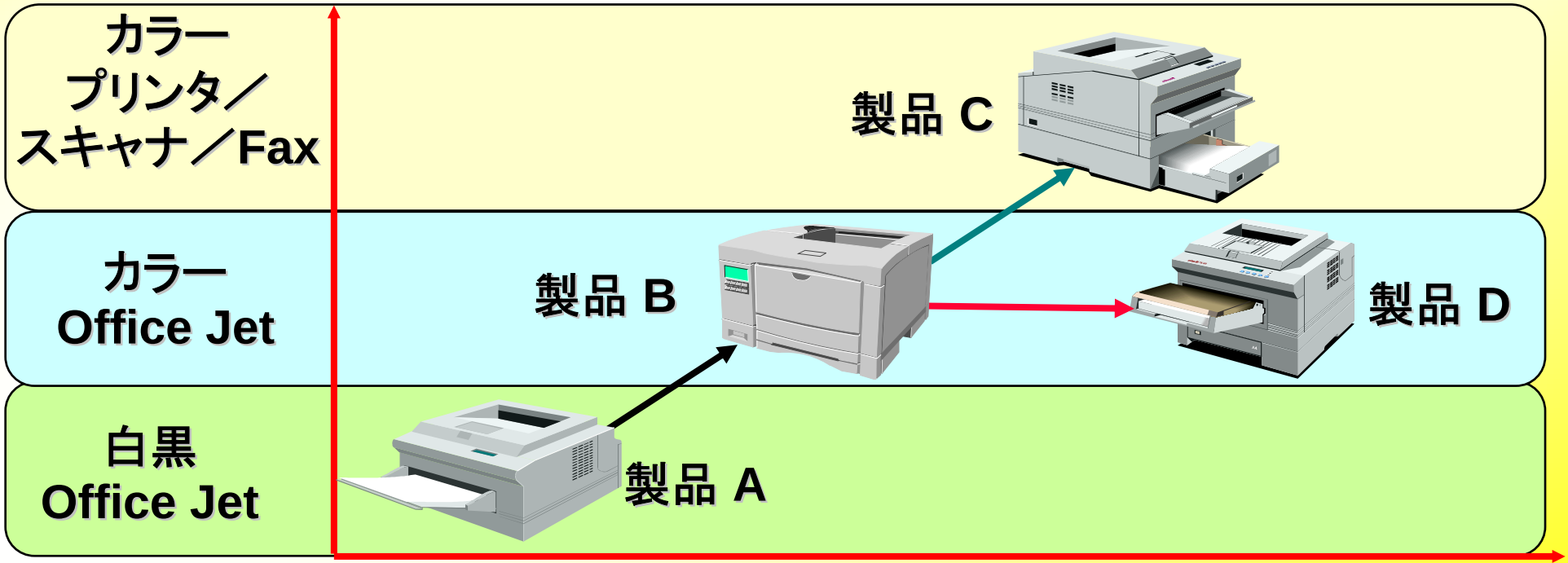
P. Clements and L. Northrop, *Software Product Lines: Practices and Patterns*, Addison Wesley, 2002.

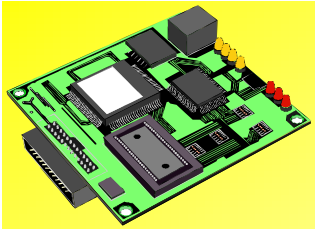
SEIのプロダクトラインのWebページ, http://www.sei.cmu.edu/programs/pls/pl_program.html



組込みソフトウェア工学とは
プロダクト:プロダクトラインソフトウェア開発

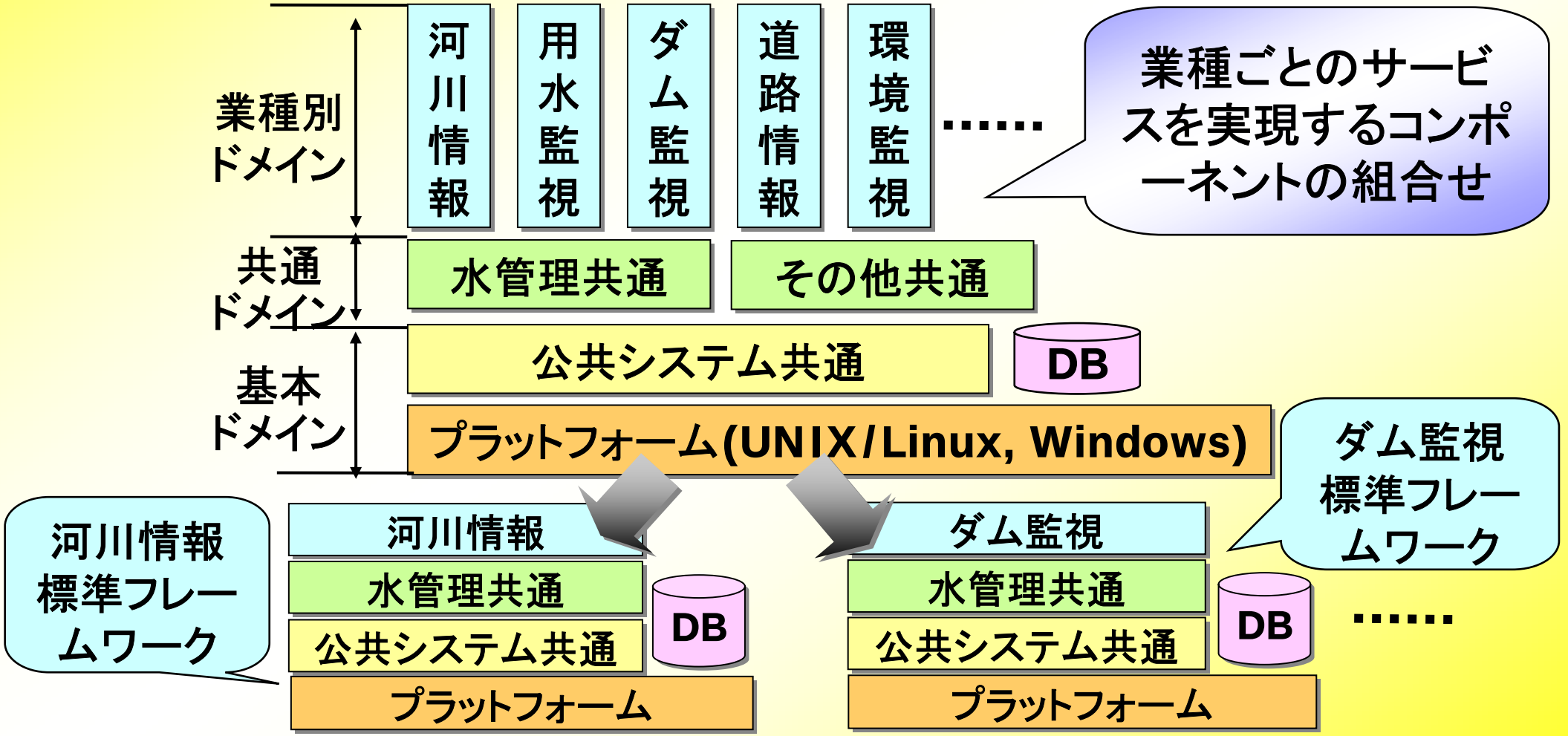
- 事例: HPオールインワンプリンタプロダクトライン
  - プラットフォームアーキテクチャ化: プロダクトライン共通部
  - プラットフォームの再利用
    - 製品Dのソフトウェアの66%はプラットフォーム再利用
- その他
  - Nokia: 携帯電話ソフトウェア, Bosch: 自動車部品ソフトウェア, 等



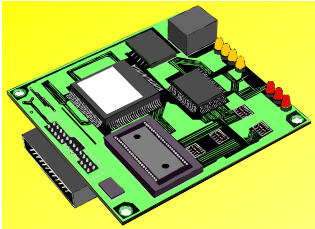


組込みソフトウェア工学とは
プロダクト: プロダクトラインソフトウェア開発

☞事例: 公共向けコンポーネント群をフレームワーク化した
プロダクトラインアプリケーション開発



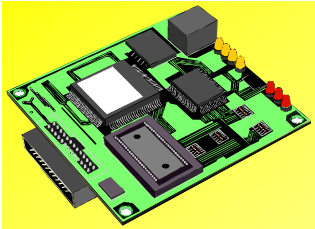
参考文献: 西尾 裕, 森脇 康之, 渡辺 健一郎, 青山 幹雄, 公共向けアプリケーションフレームワークのソフトウェアアーキテクチャの分析, 情報処理学会ソフトウェア工学研究会, No. 133-6, Sep. 2001, pp. 39-46.



組込みソフトウェア工学とは
プロセス: ソフトウェアプロセス技術の進化

プロセスモデルの黎明　多様化と形式化　標準化と改善

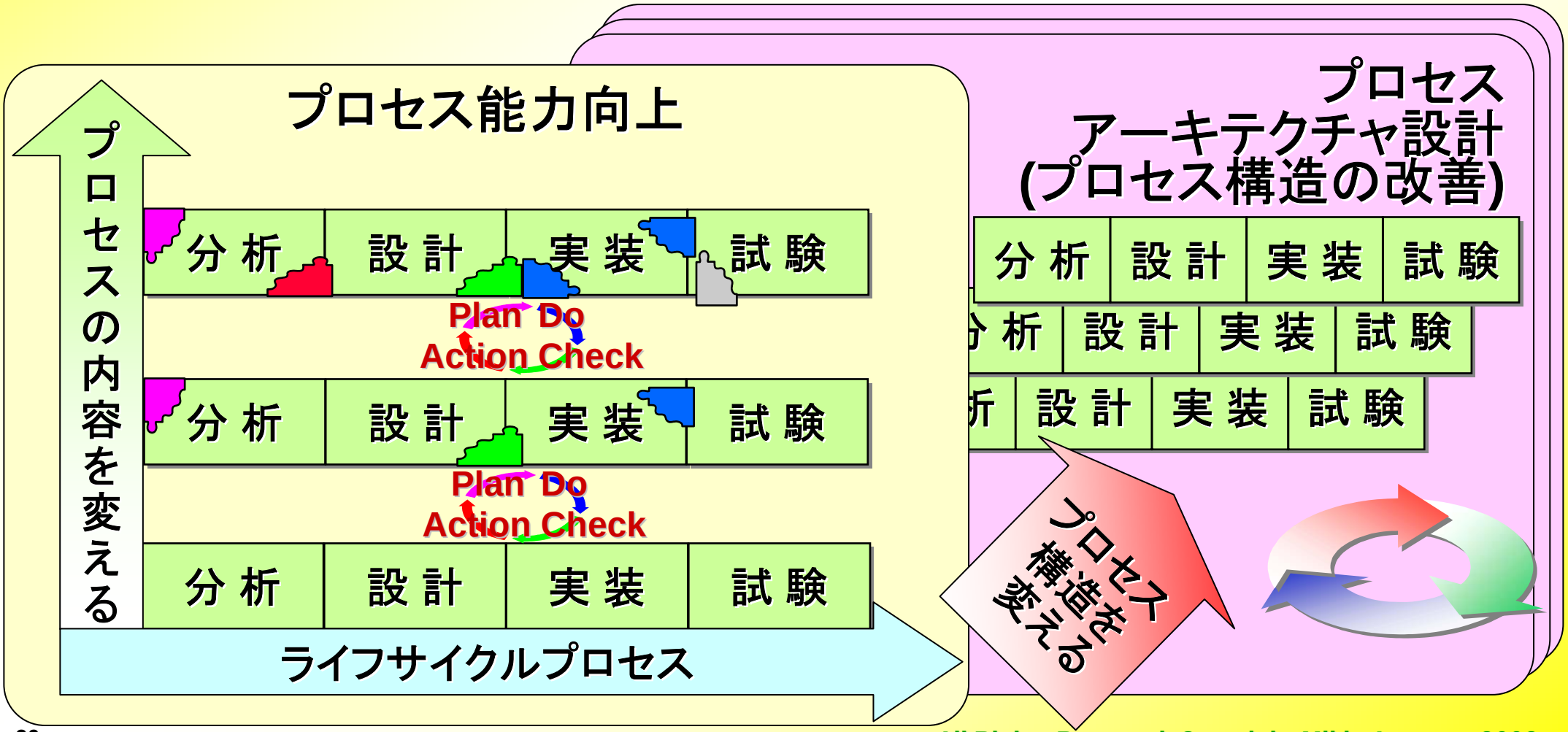
年　代	1970 年代	1980 年代	1990 年代	2K 年代
プロセスモデル	ウォーターフォールプロセス インクリメンタルデリバリ	プロトタイピング スパイラル 分散開発 RAD	インクリメンタル&イテラティブ コンカレント オープンソース	軽量プロセス(XP/アジャイル)
プロセス形式化と改善		プロセスプログラミング CMMとKPA	ベストプラクティス PSP	CMMI/SCAMPI
標準化			ISO-12207/X0160(SLCP) ISO-15504 ISO-9000 ISO-10006	

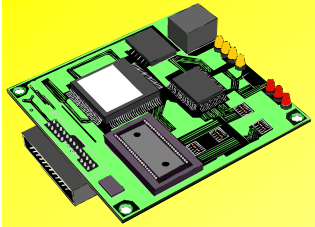


組込みソフトウェア工学とは
プロセス: プロセス工学

☞プロセス工学＝プロセスアーキテクチャ設計＋プロセス能力向上

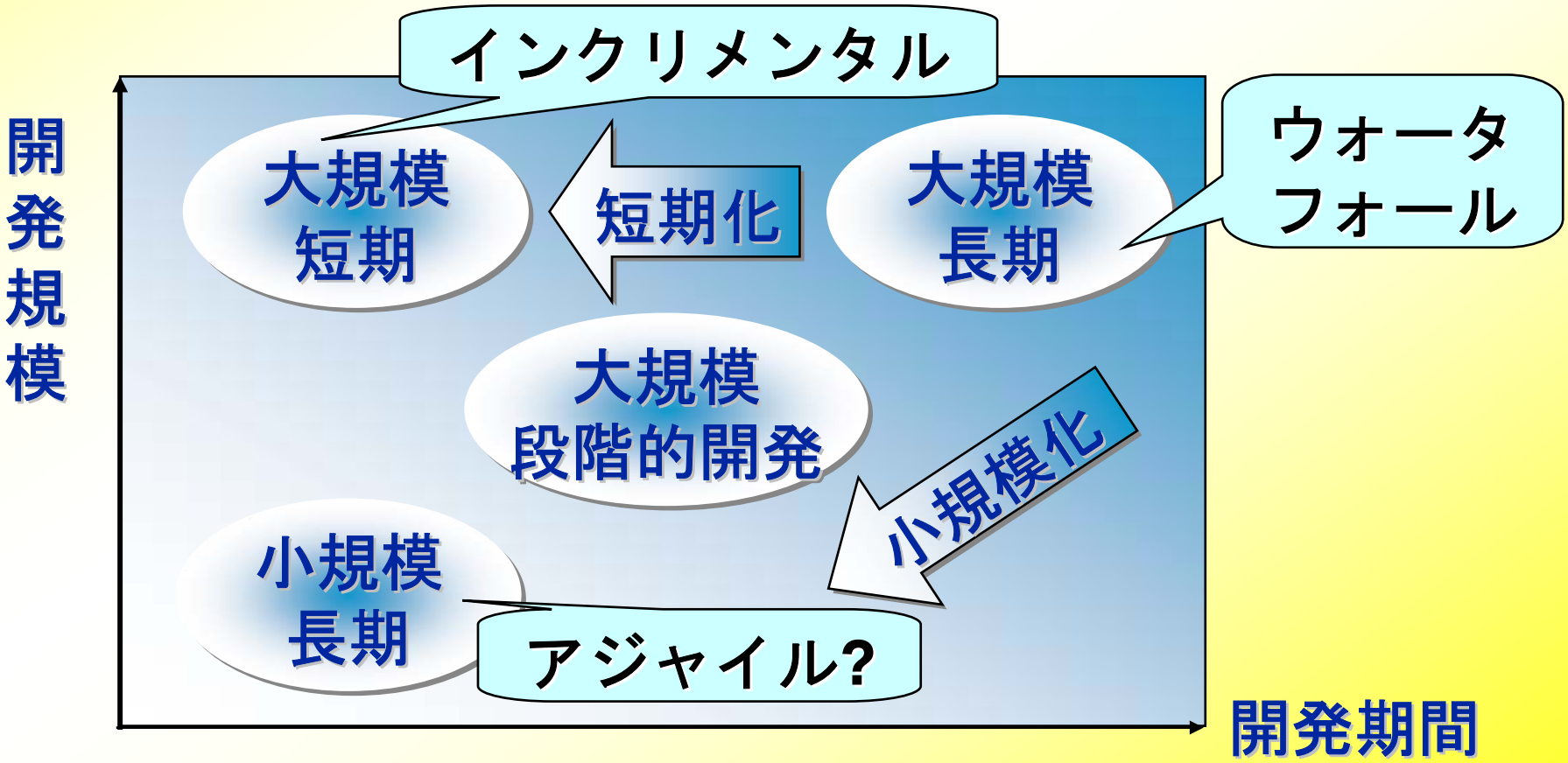
- プロセスアーキテクチャ: 開発要求を満たすプロセス構造の設計
- プロセス能力: プロセス内の活動, 利用技術の改善

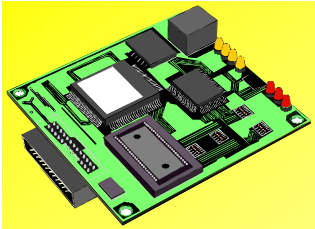




組込みソフトウェア工学とは
プロセス: 開発対象の多様化に対応したプロセス

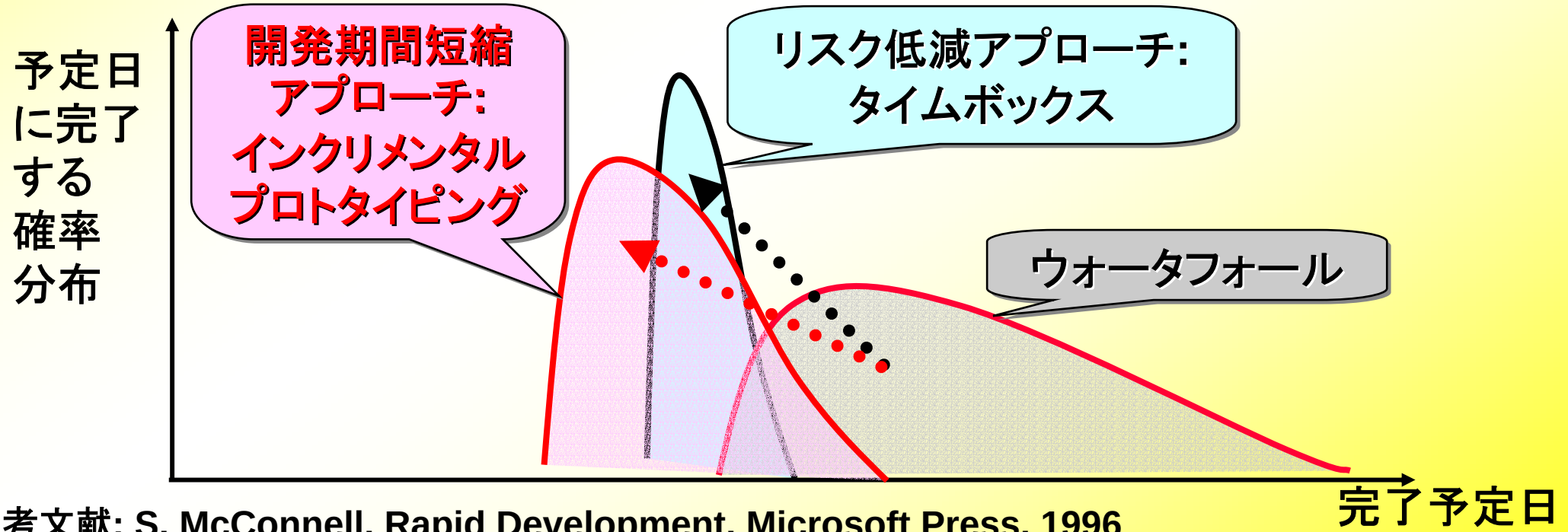
- ソフトウェアの多様化 ⇒ ソフトウェアプロセスの多様化
  - 規模と期間
- 単一のプロセスモデル(ウォータフォール)の適合不全



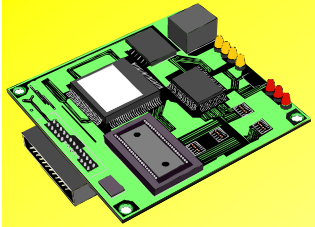


組込みソフトウェア工学とは
プロセス: 開発条件に適したプロセスの選択

- 開発の特性とプロセスの特性との適合性
  - GUI中心: プロトタイピング・RAD
  - 機能的処理: インクリメンタル（＋タイムボックス）
- プロセス能力(制御性): インクリメンタル, タイムボックスが優位



参考文献: S. McConnell, Rapid Development, Microsoft Press, 1996
[日立インフォメーションアカデミー(訳), ラピッドデベロップメント, アスキー出版局, 1998]



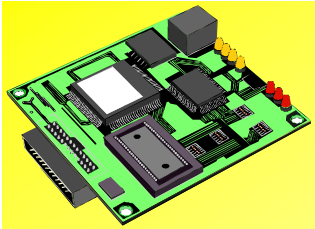
組込みソフトウェア工学とは
開発管理: 開発管理の仕組み

☞組込みソフトウェアの開発管理システム

- 本質的には同一 ⇒ より高い品質要求の達成

プロセス	部門					
	経営	企画・研究	設計	品証	サポート	ユーザ
方針	方針	→各部門へ				
調査		情報収集	→各部門へ			
製品企画		企画				
研究		研究				
設計			設計			
実装試験			レビュー / 外注 / 内製			
品証				受入検査 / 試験 / 検査 / 出荷検査		
出荷					出荷	
運用					サポート	運用

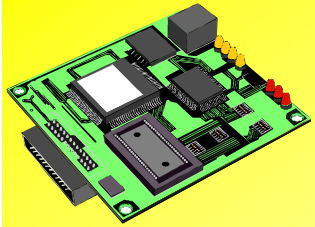
凡例
- → 製品の流れ
- → 品質管理の流れ
- ⋯→ 保守の流れ



組込みソフトウェア工学とは
ピープル: アーキテクト育成の必要性

☞Capacity(量)からCapability(能力=競争力)へ

- 開発技術の高度化に即した高度な人材の育成
  - ソフトウェア/システムアーキテクト: ソフトウェア＋ハードウェアの知識
  - 企業内アーキテクト育成プログラムの開発: 実際に設計する
- ITスキル標準など: 組込みソフトウェア開発への対応

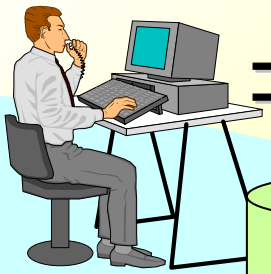


組込みソフトウェア工学とは
ピープル: アーキテクト育成の必要性

- 競争力の核となる人材としてのソフトウェアアーキテクト
  - プログラム(プロジェクト)マネジャ
    - 製品マーケティングの観点からプロジェクト管理
  - システム/ソフトウェアアーキテクト
    - システム/ソフトウェアアーキテクチャ設計の統括

プログラム
(プロジェクト)
マネジャ

チーフテクニカル
オフィサ(CTO)

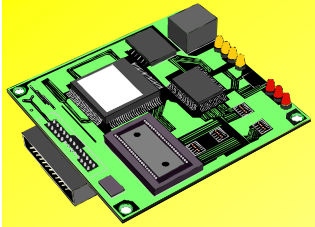


コンポーネント
開発者

ソフトウェアアーキテクト

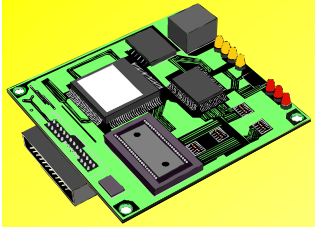
ドメイン
エンジニア

インテグレータ



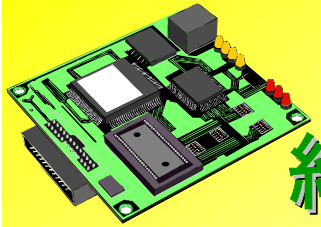
シナリオ

- 今，そこにある機会と危機
- 組込みソフトウェア工学とは
- わが国の組込みソフトウェア工学の課題
- まとめ



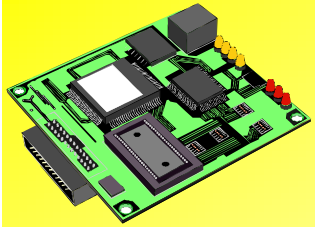
わが国の組込みソフトウェア工学の課題
組込みソフトウェア工学の実践の課題

- 組込み機器の付加価値の源泉はソフトウェアへ
- しかし,...
  - 組込み機器ベンダ(組込みソフトウェア開発者)の特性
    - 本来はコンピュータやソフトウェアベンダでない
    - ソフトウェア開発の問題認識, 技術開発・人材育成の認識
  - ソフトウェア工学導入の阻害要因
    - 開発リスク: 新しい技術の実証が得られない ⇒ 現状の開発方式を維持(変化への抵抗)
    - 現場は超繁忙: ソフトウェア工学を研究・実践する余裕がない
  - ビジネス慣行の問題
    - (一括)外注化: 問題が見えない



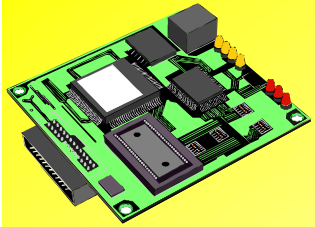
わが国の組込みソフトウェア工学の課題
組込みソフトウェア工学の研究と教育の課題

- 組込みソフトウェア工学研究者の不足
  - 大学における組込みソフトウェアへの認識不足
    - 関心は増大しているが…
  - 企業における組込みソフトウェアの情報発信の不足
- 組込みソフトウェア研究の組織的取組みの欠如
  - 実証実験する場の欠如
  - 産学連携の欠如
- 組込みソフトウェア教育の課題
  - ソフトウェア工学教育の不足
  - 組込みソフトウェア工学の教育の不足



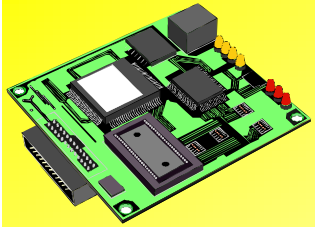
わが国の組込みソフトウェア工学の課題
学会・業界における課題

- 組込みソフトウェアの技術交流のための組織的取組みの欠如
  - 情報処理学会ソフトウェア工学研究会での取組み
    - 組込みソフトウェアWG: 2003年4月再編成
    - ワークショップでの討議: 2002年から
    - 組込みソフトウェアシンポジウムの開催: 2002年から
      - 2003年10月16-17日, 機械振興会館
      - http://www.seto.nanzan-u.ac.jp/~watanabe/ess03/
  - 各分野の学会などの活動
  - 業界団体と標準化団体の活動
- 研究開発と技術移転の組織的基盤の必要性
  - わが国全体で支援する仕組みの必要性
  - グローバルな連携の仕組み



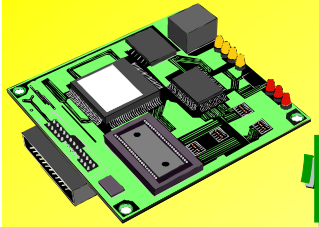
わが国の組込みソフトウェア工学の課題
わが国の組込みソフトウェア産業の優位性?

- 多くの長所に気付いていない?
- 研究開発と教育
  - 大学, 企業における継続的研究開発努力
  - 基礎となる技術水準の高さ
- 産業競争力
  - 多数のグローバル規模の企業
  - 競争優位のドメイン: 情報家電, 自動車, デジカメ, ...
- 日本の社会の品質希求性
  - 社会全体の高い品質要求性向
  - ものづくりの伝統: Made in Japan



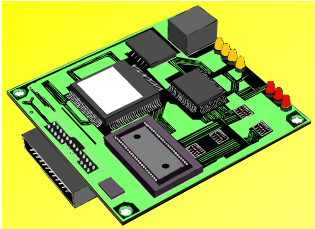
まとめ

- 組込み・ユビキタスネットワーク
  - 大きな機会と大きな危機
- 組込みソフトウェア開発とソフトウェア工学
  - 組込みソフトウェア開発におけるソフトウェア工学の必要性
  - ソフトウェア工学における組込みソフトウェア開発の重要性
- 組込みソフトウェア開発の変革の必要性
  - 人海戦術・アドホックな開発から組織的・工学的開発へ
  - ソフトウェア工学の総合工学化: 技術による淘汰
- 組込みソフトウェア工学の研究開発と実践の推進
  - ベクトルを合わせよう!
  - **Software Made in Japan** を掲げよう



参考資料
情報処理学会ソフトウェア工学研究会の活動

- 研究会の位置づけ
  - 情報処理学会内の研究会(1977年設立, 現在の会員＝約550人)
  - http://www.ipsj.or.jp/sig/se/
- 活　動
  - 定例研究会: 年4回
  - シンポジウム
    - オブジェクト指向[1995年～], 組込みソフトウェア工学[2001年～]
  - ワークショップ: 1993年～
  - 国際会議
    - APSEC(アジア太平洋ソフトウェア工学)[1994年～], ほか
  - ワーキンググループ
    - 要求工学, 組込み, パターン
- 組込みソフトウェアシポジウム: 2003年10月16-17日
  - http://www.seto.nanzan-u.ac.jp/~watanabe/ess03/
    あるいは上記研究会のWebページから



参考文献
(より深く知るために)

1) 青山 幹雄, 佐伯 元司, 深澤 良彰, 本位田 真一(編), ソフトウェアテクノロジーシリーズ, 全12巻, 共立出版, 1999~[刊行中].
2) 青山 幹雄, ソフトウェアプロセス・リエンジニアリング, 情報処理, Vol. 36, No. 5, May 1995, pp. 442-447.
3) 青山 幹雄, 中所武司, 向山 博(編著), コンポーネントウェア,共立出版, 1998.
4) 青山 幹雄, 組込みソフトウェアの転機, 情報処理, Vol. 39, No. 7, Jul. 1998, pp. 689-692.
5) 青山 幹雄, ソフトウェア技術者のグローバルスタンダード化, 情報処理, Vol. 39, No. 11, Nov. 1998, pp. 1144-1147.
6) 青山 幹雄, コンピュータが消える日: インターネット時代のソフトウェア, 情報処理, Vol. 41, No. 5, May 2000, pp. 523-527.
7) 青山 幹雄, ソフトウェアサービス技術へのいざない, 情報処理, Vol.42, No. 9, Sep. 2001, pp. 857-862.
8) 青山 幹雄, オープンソースソフトウェアの現状, 情報処理, Vol.43, No. 12, Dec. 2002, pp. 1319-1324.
9) 青山 幹雄, 中谷 多哉子(編著), オブジェクト指向に強くなる, 技術評論社, 2003.
10) A. Finkelstein (ed.), *The Future of Software Engineering*, ACM Press, 2000.
11) National Research Council, *Making IT Better: Expanding IT Research to Meet Society's Needs*, National Academy Press, 2000.
12) NIST, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, May 2002.
13)野村総合研究所(編), ユビキタス・ネットワークと新社会システム, 野村総合研究所, 2002.
14) PITAC Webページ, http://www.itrd.gov/ac/index.html
15) 田中 辰雄, ソフトウェア産業, 後藤 晃ほか(編), サイエンス産業, NTT出版, 2003, pp. 253-278.