3D² Value Chain
3D (SD-DVD-DTV) X 3D (HD-UD-Device)

JEITA組込み系開発
スピードアップ・ワークショップ2008

モデル駆動型開発の導入による生産性向上の事例

2008年8月27日

松下電器産業株式会社
パナソニックAVCネットワークス社
技術統括センター
南光　孝彦

Panasonic ideas for life

松下電器 PAVC社の事業領域と事業戦略

松下電器産業株式会社　パナソニックAVCネットワークス社（PAVC社）

・97年4月社内分社として発足　約29,500名

映像・ディスプレイデバイス事業グループ

・デジタルTV・・　PDP、液晶、CRT
・CATV/BS/CS受信機
・ディスプレイデバイス

ネットワーク事業グループ

・DVD、SD-AV機器
・ビデオカメラ、DSC
・オーディオ機器

システム事業グループ

・業務用AVシステム
・アビオニクス
・PC
・プロジェクタ

デバイス事業グループ

・光ピックアップ、レンズ
・SDメモリカード、DVD－RAM

3D² Value Chain
3D (SD-DVD-DTV) X 3D (HD-UD-Device)

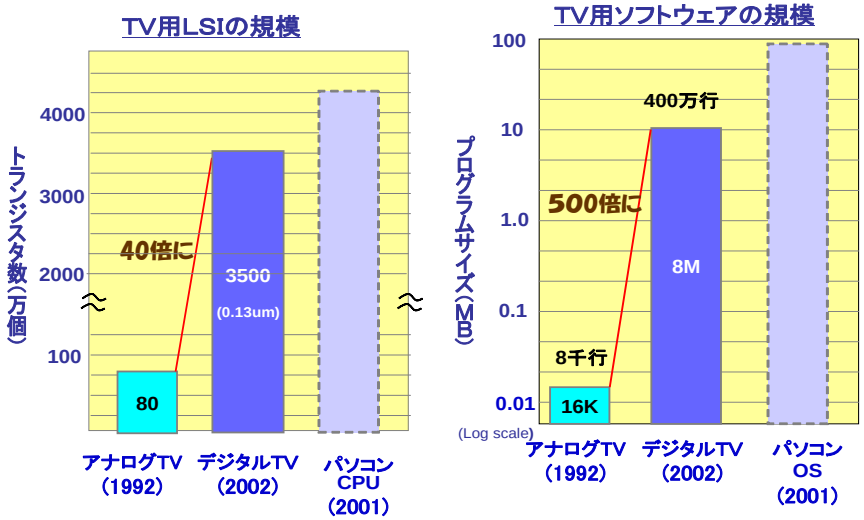
『SDカード関連商品』、『DVD関連商品』、『デジタルテレビ（DTV）関連商品』の3つの商品軸『D』に加え、
ハイ・デフィニションの『HD』、ユニバーサルデザインの『UD』、
『DEVICE』の3つの要素軸『D』を掛け合わせ、
そのバリューチェーン・シナジーによりお客様価値を生み出します。
そしてこの取り組みを徹底し、クオリティーの高い商品を
グローバル市場に提供していきます。

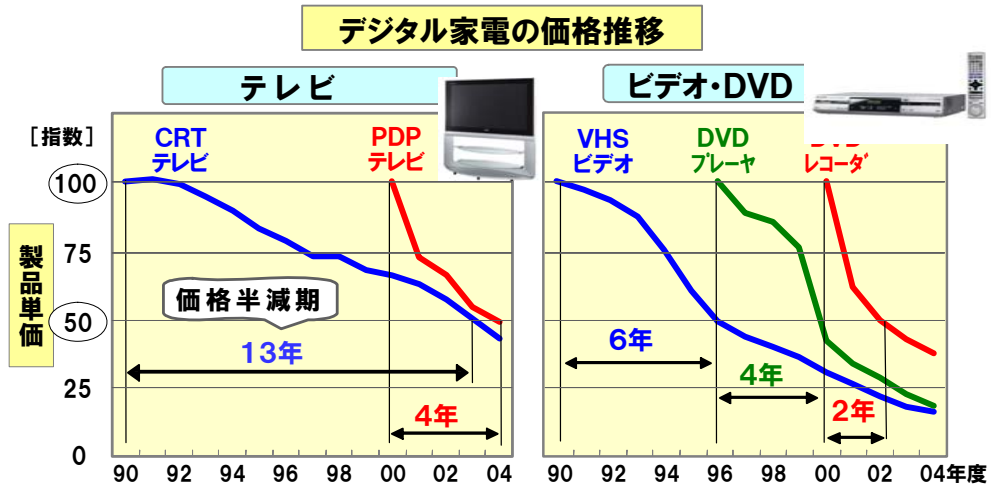
Panasonic ideas for life

AV機器向け組込みソフト開発を取り巻く環境変化

- **ソフトウェア開発の重要性が10～20年で急激に高まる**
  - AV機器・放送のデジタル化
    - ⇒ ソフトウェアで実現する機能の増加 ＝ ソフト大規模化
- **グローバルでの急激な価格下落への対応**
  - グローバル多機種展開・グローバル同時立上げの事業戦略
    - ⇒ ソフトウェア開発へのプレッシャーが増大

半導体とソフトウェアの規模の急増





Panasonic ideas for life

PAVC社のソフトウェア開発の特徴

■PAVC社商品のソフトウェア規模は数千～数百万行と幅広く開発形態も多様

■機器のデジタル化・ネットワーク化に伴い、ソフトウェア規模は増加傾向

Panasonic ideas for life

グローバル機種増への対応

デジタル化・ネットワーク化に伴うソフトウェア規模の増大に加え

- ■デジタルTV、DVD、SD関連機器を中心に機種数が増大
- ■世界同時垂直立上げに伴う機種開発の同時化
- ■機器の多機能化・ネットワーク化に伴う検証項目数の急増

デジタルTVの場合

デジタルスチルカメラの場合

Panasonic ideas for life

松下電器・PAVC社ソフトウェア開発力強化の取組み

① 組織横断のソフトウェアプロセス改善

複数の事業場を跨る組織横断活動の推進 ～ ベストプラクティス交流

○事業場を跨る相互アセスメントの実施（SW-CMM/CMMI）

○メトリクス標準化による事業場での活用促進

② プラットフォーム型開発

○デジタルTVのグローバルプラットフォーム技術による世界各国対応の実現

○商品間を跨るプラットフォームの共通化（UniPhier®）

③ 検証体制の強化

○システム共同検証センターの設立によるAV機器の相互接続検証強化
（http://panasonic.co.jp/pavc/csvc/index.html）

④ エンジニアリング領域強化による生産性向上

★ モデル駆動型開発（MDD）の試行導入（2005年より）

○静コード解析などのツール活用促進

Panasonic ideas for life

モデル駆動型開発（MDD）の導入事例

モデル駆動型開発手法導入の背景

PAVC社ソフトウェア開発力強化の視点より

【これまで】

・プラットフォーム型開発によるソフトウェア再利用を推進し、
ソフトウェアの大規模化、多機種同時開発化に対応

【今後】

・グローバル機種展開の拡大、開発リードタイム短縮
に向けて更なる効率化が必要

・ソフトウェア再利用率が高レベルになってきた現状を踏まえ、
効率化に向けた新たな開発方法論・手法の導入が必要

「モデル駆動型開発」の効果検証～導入

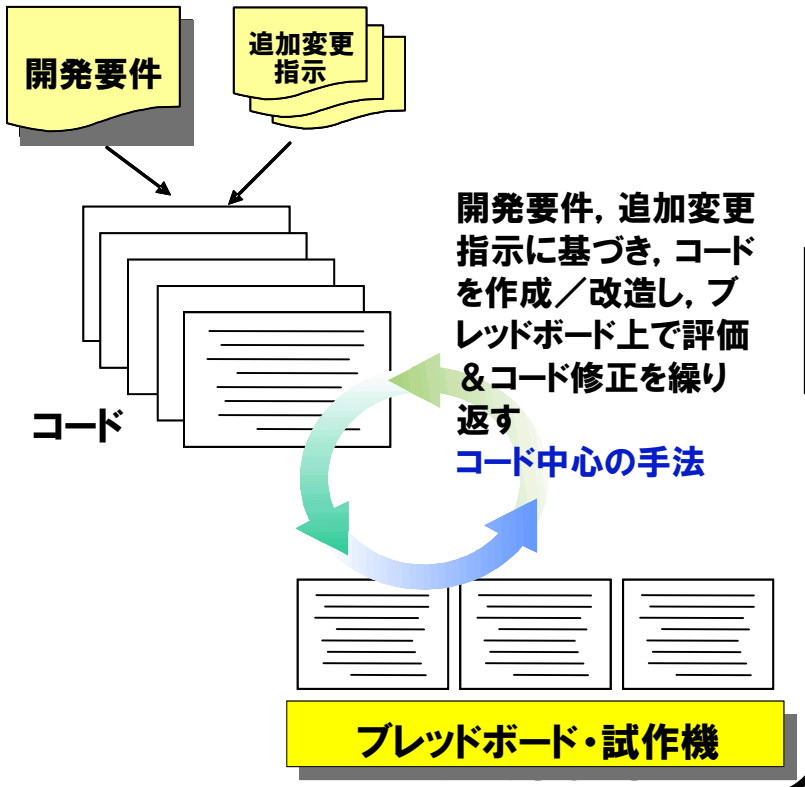
Panasonic ideas for life

モデル駆動型開発(Model Driven Development)概要

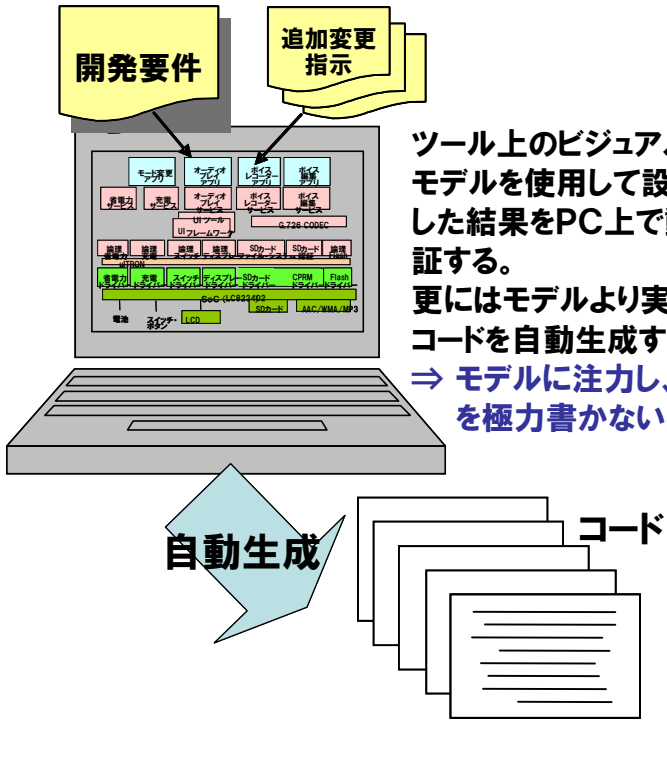
■ ツールを用い、モデル(図)で構造設計・動作設計を行い、モデルで動作検証まで実施
■ モデルからプログラムコードを自動生成し、そのまま実機のソフトとして利用(手作業の排除)

★*LSI CADやメカの3次元CADのように、設計段階で実機レスの設計・動作検証が可能*

従来のソフトウェア開発



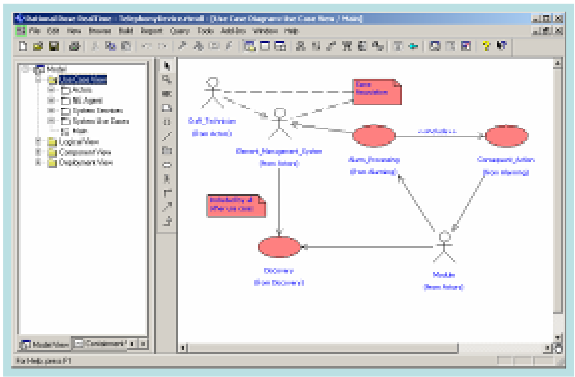
モデル駆動型開発開発



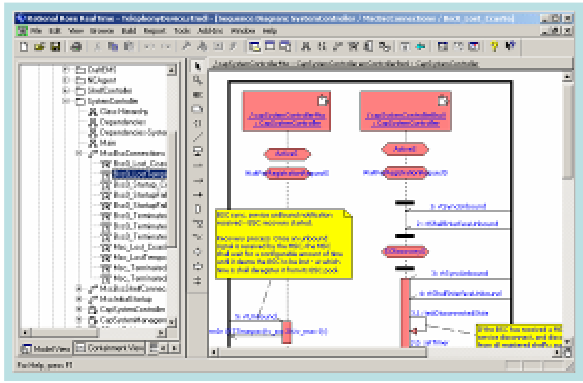
Panasonic ideas for life

MDDツール　Rational Rose Realtime概要

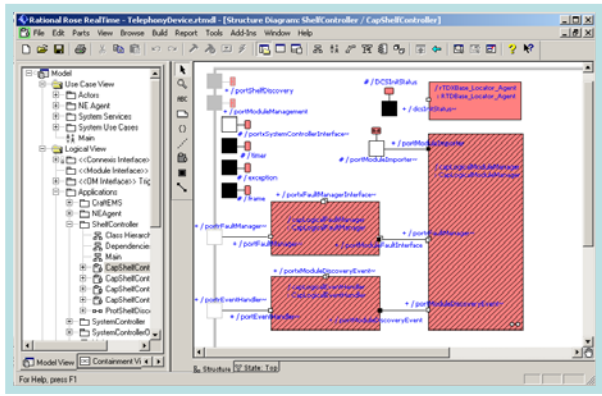
■ 世界標準記法であるUML（Unified Modeling Language）を用い、要求分析・設計・実装
■ UMLで記述したチャート（一部C／C++で記述）から実行可能なコードを自動生成



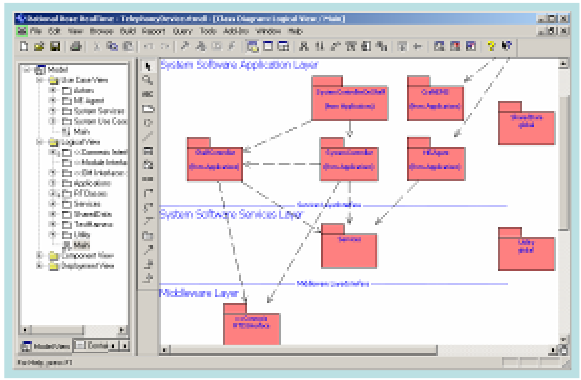
ユースケース図
（要求分析）



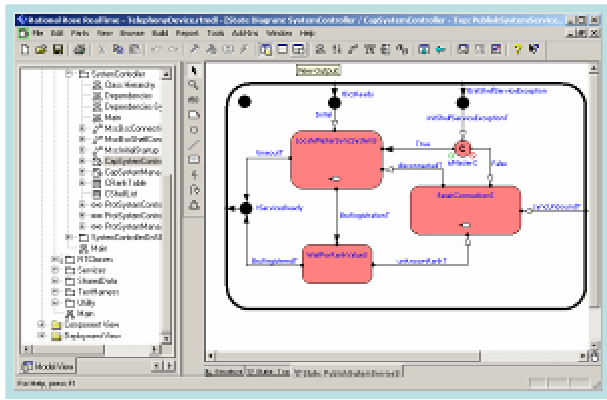
シーケンス図
（動作設計）



配置図
（実装）



パッケージ図
（アーキテクチャ設計）



ステートチャート図
（実装）

Panasonic ideas for life

モデル駆動型開発手法導入　サマリ

【検証フェーズ】 検証プロジェクトを立ち上げ、手法の有効性を検証

・商品開発プロジェクト（従来手法：C言語ベタ書き）と並行し
モデル駆動型開発手法を用いた検証プロジェクトで同じ商品ソフトを
開発、両者のメトリクスを比較分析

＜指標＞・生産性向上（工数削減度合い）・品質（デバッグ効率）
・メモリー（ROM/RAM容量）　・消費電力　など

【結果】手法の有効性を確認、商品開発への導入を決定

【導入フェーズ】事業場メンバーによる商品開発への手法導入

・検証プロジェクトのモデル駆動型開発手法による
成果物をベースに次期商品開発を推進

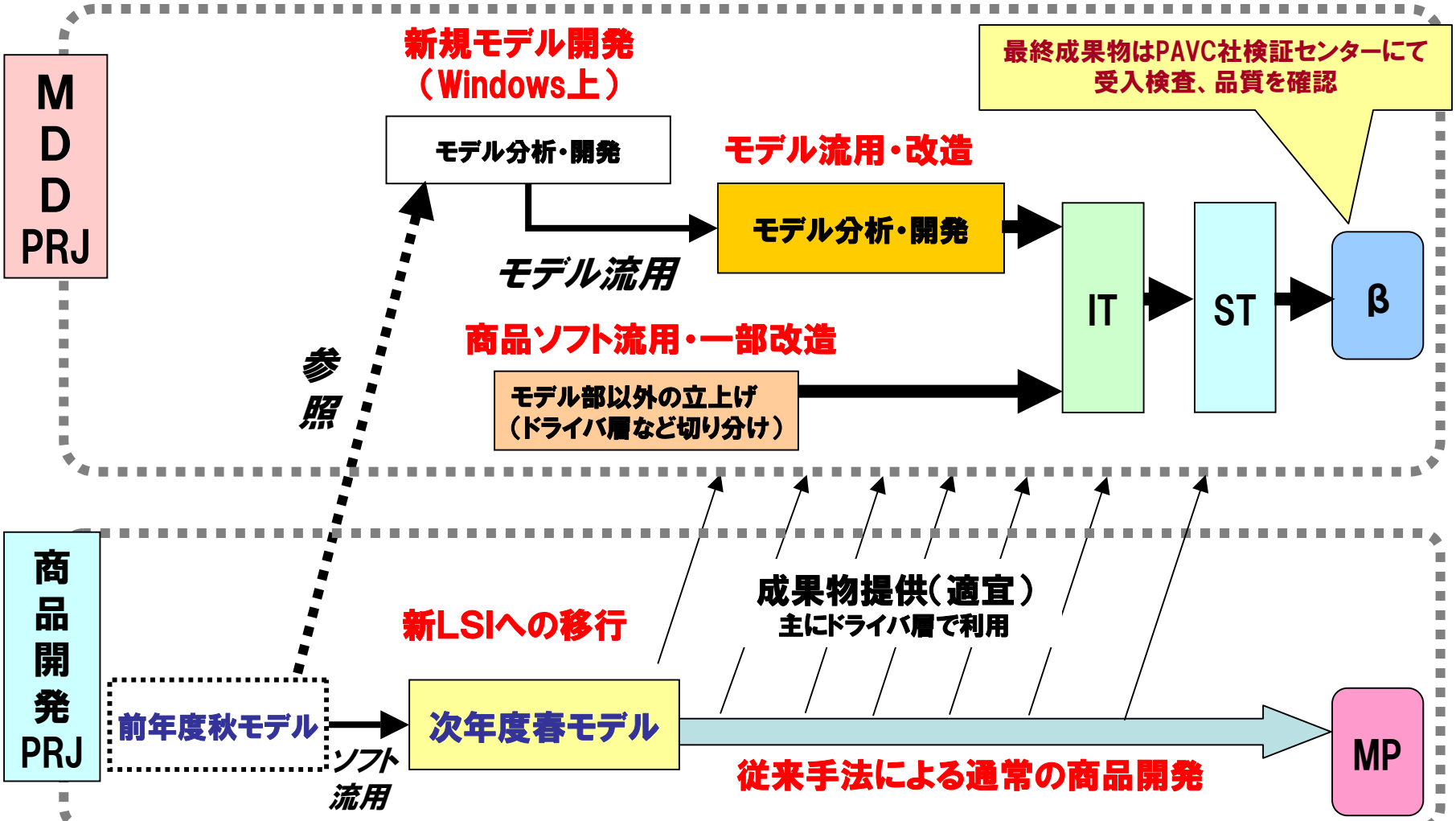
＜取組み＞・ソフト開発プロセス変更(インクリメンタル型)
・既存資産活用検討　・アーキテクチャ見直し　など

【結果】ソフト日程遅れ無しで無事商品化を完了

Panasonic ideas for life

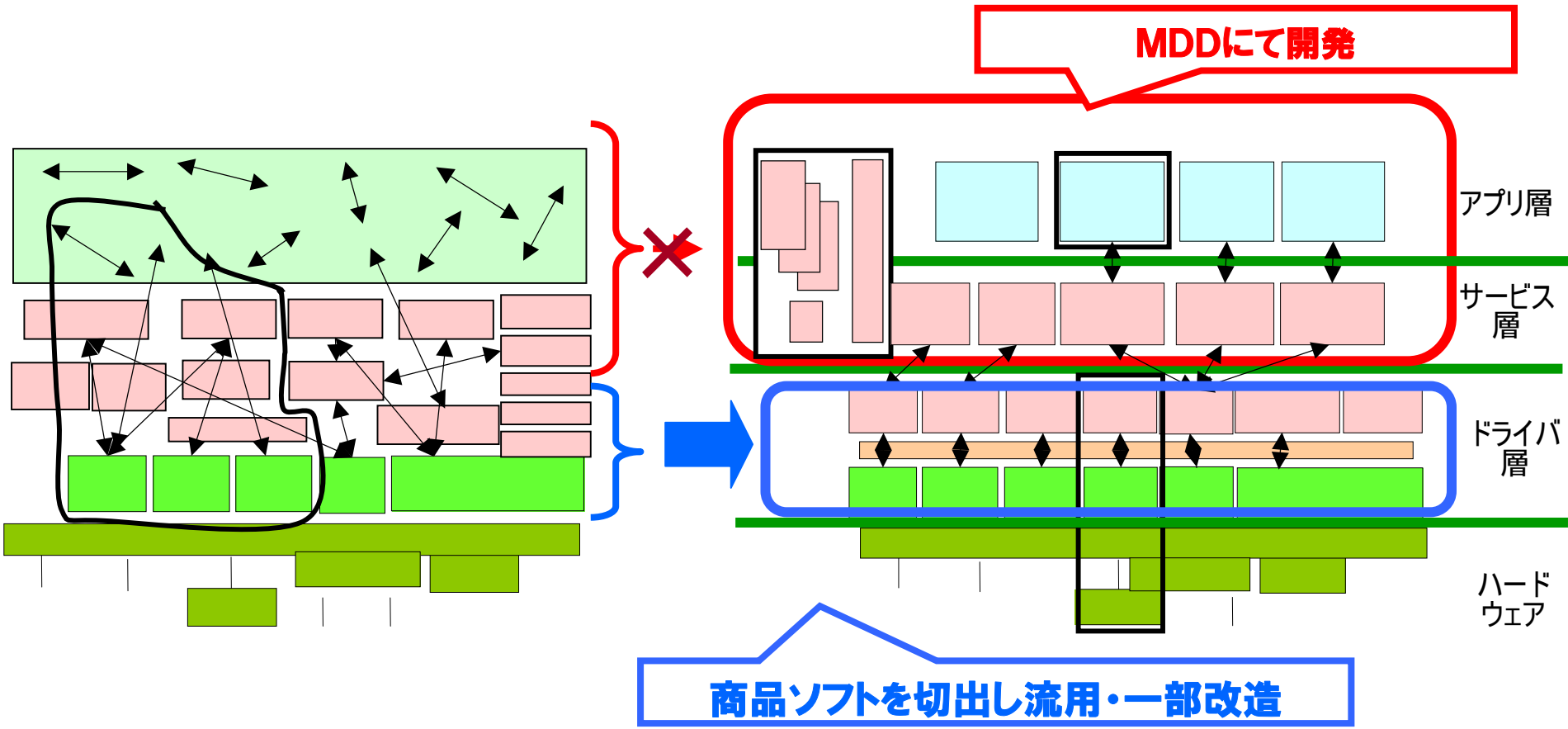
検証プロジェクトと商品開発プロジェクトとの関係

■ PAVC社商品開発と並行して、検証プロジェクトにて同じ商品の
ソフトウェア開発をモデル駆動型開発手法にて実施



MDDプロジェクトのソフトウェアアーキテクチャ

- 商品ソフトウェアを分析し、アーキテクチャの再設計を実施
- MDDプロジェクトがモデル駆動型開発で開発したのはソフトウェア全体の約50%



Panasonic ideas for life

【検証フェーズ】評価結果

評価指標	結果
① 生産性	モデル部開発工数は従来手法開発の約7割 （生産性は約1．5倍）
② 品質	モデル部でのデバッグ効率が良い（モデル部で速度4倍）
③ ROM・RAM容量	ほぼ同等 （ROM：2割増、RAM：同等）
④ パフォーマンス・消費電力	ほぼ同等 （パフォーマンス：同等、消費電力：微増）

- モデル駆動型開発導入により開発効率向上の効果（見込）を確認
- 懸念されていたHWリソース・性能への影響が小さいことを確認

次年度商品開発への「モデル駆動型開発手法」導入を決定

Panasonic ideas for life

【導入フェーズ】商品開発プロジェクトへの手法導入

【課題】商品開発へ手法導入にあたってのリスク

①ウォーターフォール的プロセスの採用への不安

→ ・事業場の商品開発部門にモデル駆動型開発の経験者がほとんどいない

・開発終盤での問題発生や設計変更が困難となり、日程遅れ発生のリスクが大

②細かい商品仕様のノウハウに関わるモジュールのモデル化の労力

→ これまでの商品化ノウハウが凝縮されたミドルウェアをモデルに移植するのは労力大

③特定のモジュールの巨大化・複雑化

→ 特定のモジュールが巨大化・複雑化し、品質の作りこみに不安

【取組み】上記課題に対して、以下の3点の対策を実施

① インクリメンタル型開発プロセスの導入

② 実績のあるC言語資産の有効活用

③ 製品動作仕様の再分析と基本アーキテクチャの変更

【導入フェーズ】評価結果

ほぼ検証プロジェクトと同等の結果が得られた

本製品のソフトウェア開発では、今後もモデル駆動型開発を継続し さらなる効率化を目指す

	実績	考察
生産性向上	MDD手法導入部の工数は従来開発の約8割に	・技術者が手法に不慣れであったことや、一部設計変更を実施したことが原因 ・次機種では更なる効率化を目指す
バグ対応容易性	モデル部では通常のC言語部に比べて解決速度が速い	・検証プロジェクトでの結果と同等
手法適用規模	総規模の約50％をモデルで設計開発	・ほぼ想定どおり ・今後適用範囲拡大を検討
ROM容量	前モデルより2割程度増加	・当初目標の3割増以内には入ったが今後も容量増加には注意が必要
コード自動生成	平均生成率75％	・検証プロジェクト評価結果と同等
バグの前出し	システムテストまでに全バグの半分以上を検出	・図で設計するため設計レビューが容易になっていると考えられる

Panasonic ideas for life

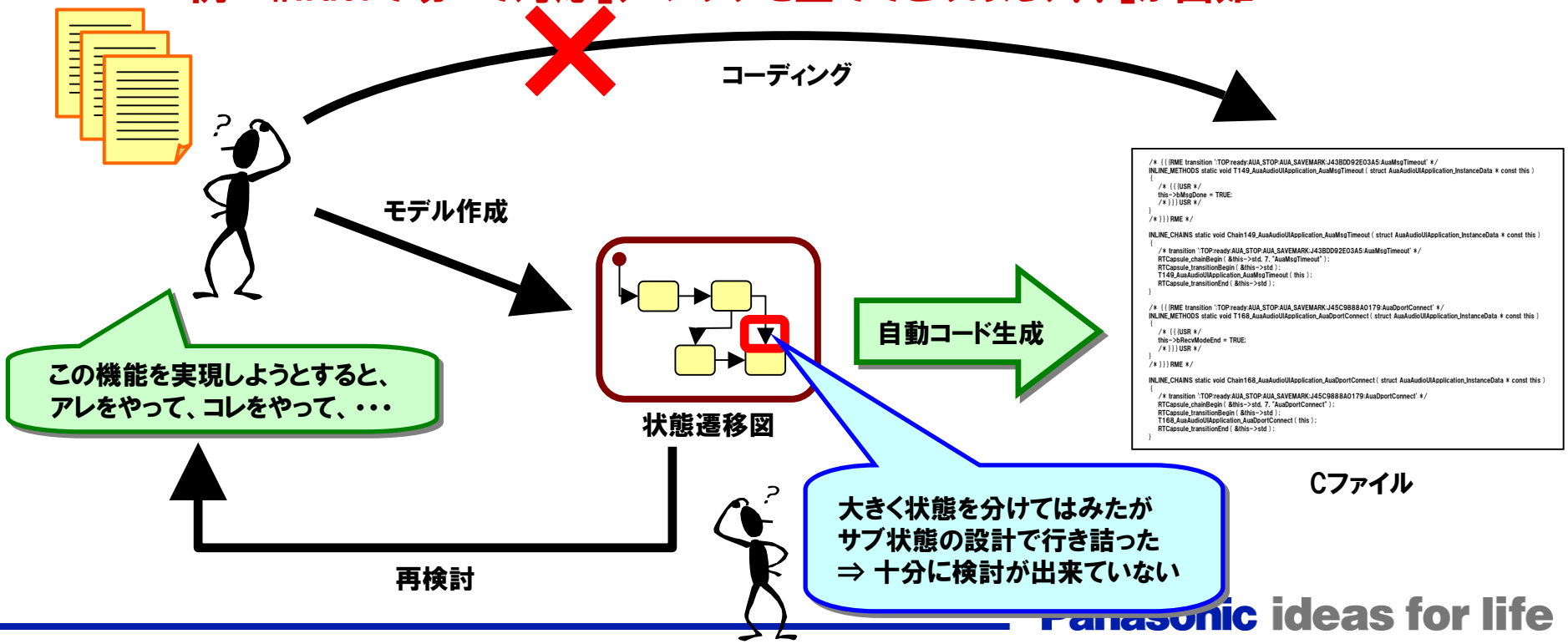
考察
モデル駆動型開発の効果

モデル駆動型開発の効果（１）

トップダウン設計が強制される＝「いきなりコーディング」が出来ない

○図を作成することで設計検討内容の再確認、再整理に繋がる
○曖昧に検討した部分は詳細にモデル化できない
○最終的には成果物のメンテナンス性が向上する

【注】付け焼刃的対応が難しく、必ず設計段階に立戻る必要がある
例：「#ifdefで切って対応」、「フラグを立ててとりあえず。。」が困難

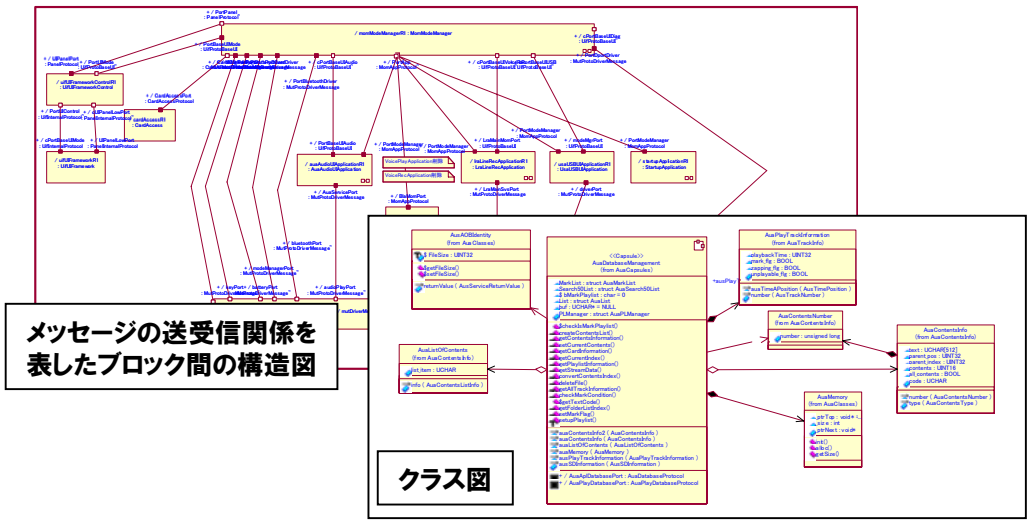


Panasonic ideas for life

モデル駆動型開発の効果(２)

ソフトウェア構成が整理される

○モジュールの役割・責務とモジュール間の相互関係を明確にできる
○自動コード生成のため、設計情報(UML)と実装(C言語)が常に同期
○状態遷移図で設計し状態遷移表でチェックすることでモレ・ヌケを防止
○問題発生時の原因ポイントが特定しやすい
【注】全体構造(アーキテクチャ)設計の方針を誤ると、実装が複雑化する



自動コード生成

```
/* {{{RME transition 'TOP:ready:AUA_STOP:AUA_SAVEMARK:J43BDD92E03A5:AuaMsgTimeout' */
INLINE_METHODS static void T149_AuaAudioUIApplication_AuaMsgTimeout ( struct AuaAudioUIApplication_InstanceData * const this )
{
    /* {{{USR */
    this->bMsgDone = TRUE;
    /* }}} USR */
}
/* }}} RME */

INLINE_CHAINS static void Chain149_AuaAudioUIApplication_AuaMsgTimeout ( struct AuaAudioUIApplication_InstanceData * const this )
{
    /* transition 'TOP:ready:AUA_STOP:AUA_SAVEMARK:J43BDD92E03A5:AuaMsgTimeout' */
    RTCapsule_chainBegin ( &this->std, 7, "AuaMsgTimeout" ) ;
    RTCapsule_transitionBegin ( &this->std ) ;
    T149_AuaAudioUIApplication_AuaMsgTimeout ( this ) ;
    RTCapsule_transitionEnd ( &this->std ) ;
}

/* {{{RME transition 'TOP:ready:AUA_STOP:AUA_SAVEMARK:J45C9888A0179:AuaDportConnect' */
INLINE_METHODS static void T168_AuaAudioUIApplication_AuaDportConnect ( struct AuaAudioUIApplication_InstanceData * const this )
{
    /* {{{USR */
    this->bRecvModeEnd = TRUE;
    /* }}} USR */
}
/* }}} RME */

INLINE_CHAINS static void Chain168_AuaAudioUIApplication_AuaDportConnect ( struct AuaAudioUIApplication_InstanceData * const this )
{
    /* transition 'TOP:ready:AUA_STOP:AUA_SAVEMARK:J45C9888A0179:AuaDportConnect' */
    RTCapsule_chainBegin ( &this->std, 7, "AuaDportConnect" ) ;
    RTCapsule_transitionBegin ( &this->std ) ;
    T168_AuaAudioUIApplication_AuaDportConnect ( this ) ;
    RTCapsule_transitionEnd ( &this->std ) ;
}
```

図で書いた関係が
そのままインクルード関係となる

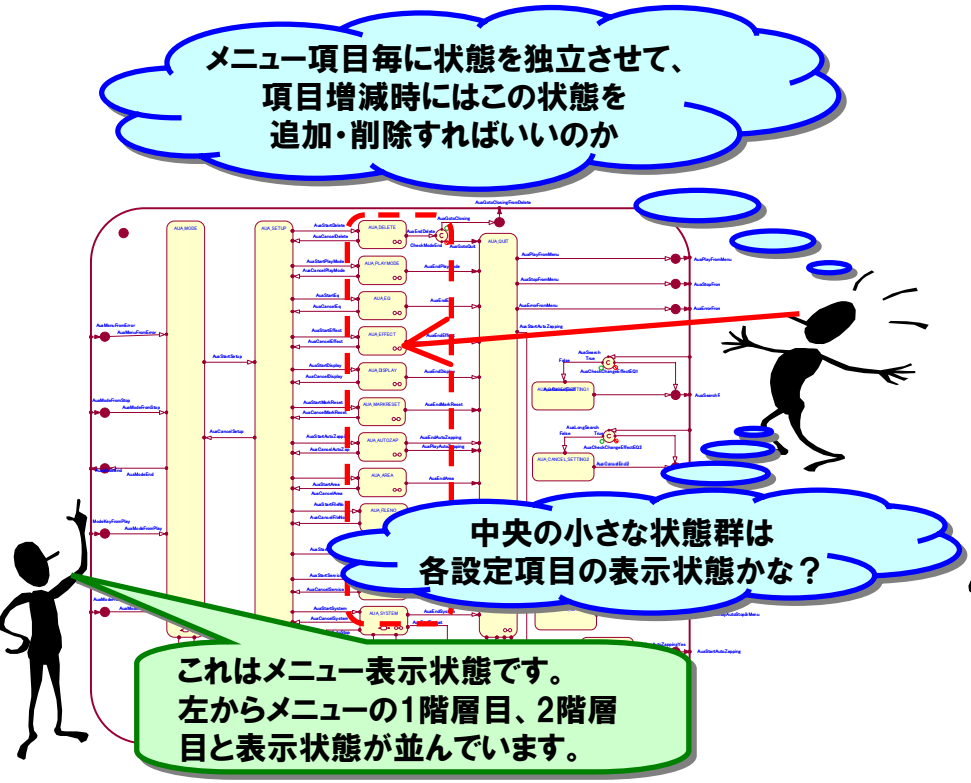
Cファイル

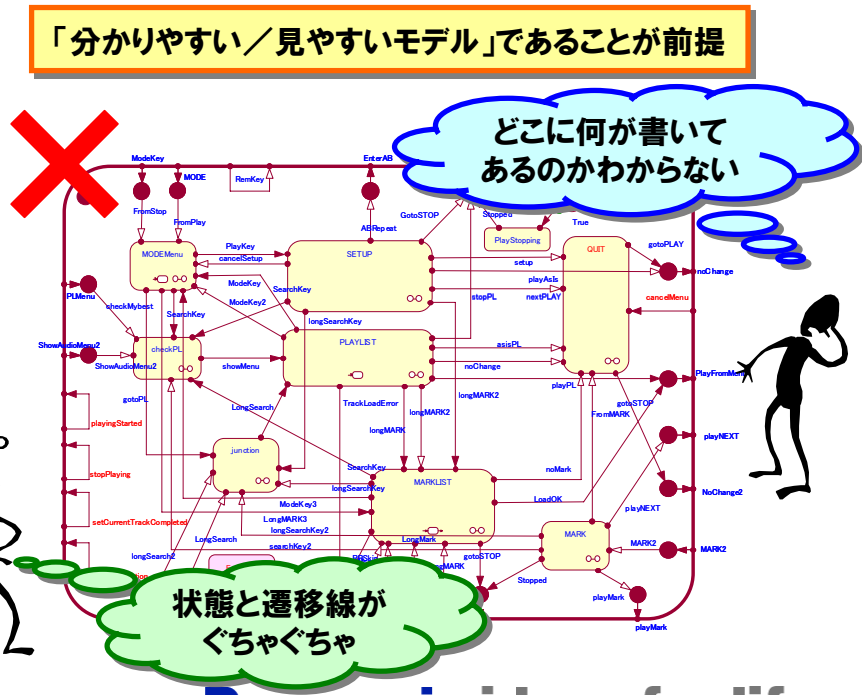
Panasonic ideas for life

モデル駆動型開発の効果(３)

開発者間での設計レビューが容易になる

○UMLという共通言語を用いた記法で統一される
○図的表記であるため、レビュアは設計内容を理解しやすい
○モデルレビューがソースコードレビューを兼ねる(自動生成)
【注】「レビューしやすい図」を書くための「設計・表記ルール」の徹底が重要





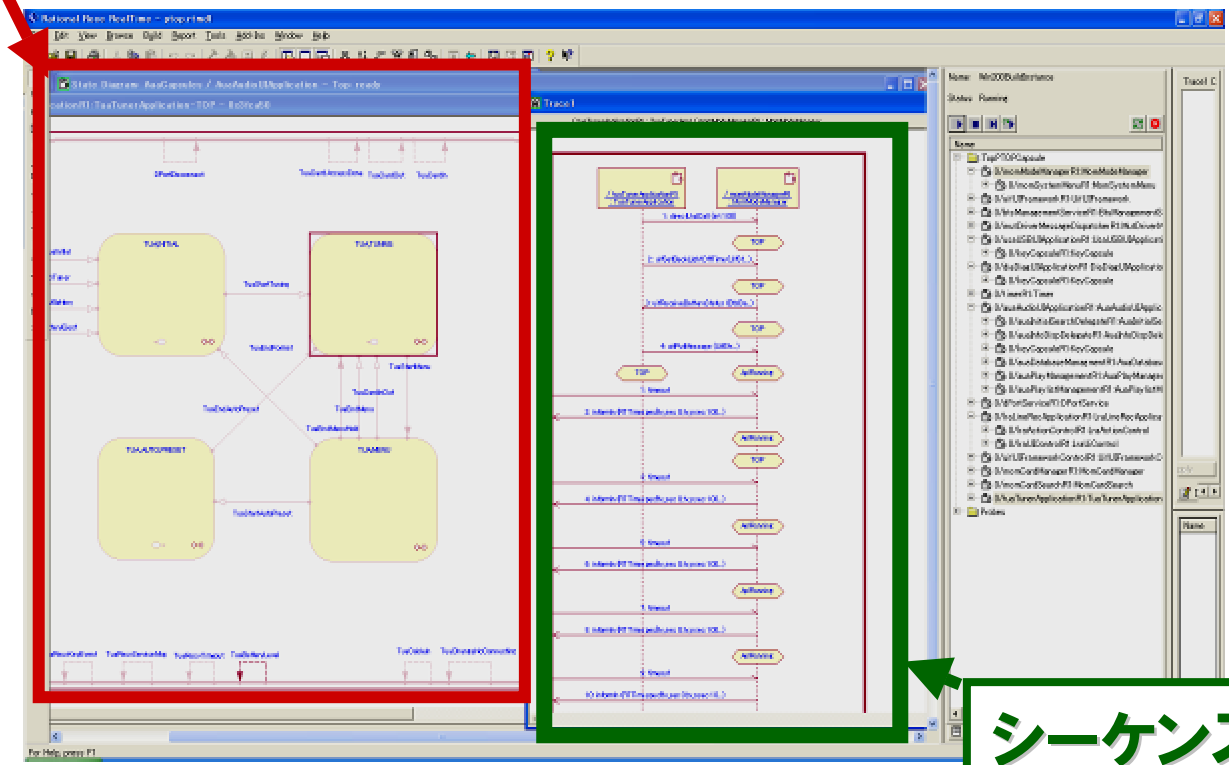
Panasonic ideas for life

モデル駆動型開発の効果(４)

ツール上でモデルベースのデバッグが可能(実機レス)

○Windows上で状態遷移、イベント送受信を視覚的に確認しながらデバッグ
→実機開発環境がなくても不具合箇所を早期に特定

デバッグ中の状態遷移図



シーケンス図

Rational Rose RealTime

Panasonic ideas for life

まとめ

■モデル駆動型開発は有効な手法であるが、、、

1. モデル駆動型開発を使えば、何でも誰でも効率化が出来るというわけではない

- MDDは有効なツールだが、重要なのはやはり『良い設計』と『良いプロセス』
- モデリングと商品アーキテクチャの両面に精通した人が必要（＝アーキテクト）
- 自動コード生成によるコーディング作業省力化、設計と実装の同期は効果有り

2. プラットフォーム化が既に進んでいるときなどは投資効果の事前検証が重要

- 導入オーバーヘッド（教育・ツール投資・既存ソフトの置換え）と導入リスク
- トップダウンでの方針／段階的な導入計画・目論見が無いと、なかなか導入は難しい

3. 一気に全体をモデル駆動型開発に移行するのは難しい

- 現状課題の解決に向け出来るところから少しずつ導入し、アーキテクチャ改良のサイクルを廻す

狼人間を倒す銀の弾丸など無い

The Mythical Man-Month : essays on software engineering, Brooks, F. P.,

Panasonic ideas for life