AOO開発手法のご紹介
～表形式と日本語によるモデル駆動開発～

三菱電機　設計システム技術センター
岩橋正実

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

目次

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

1．開発上の課題

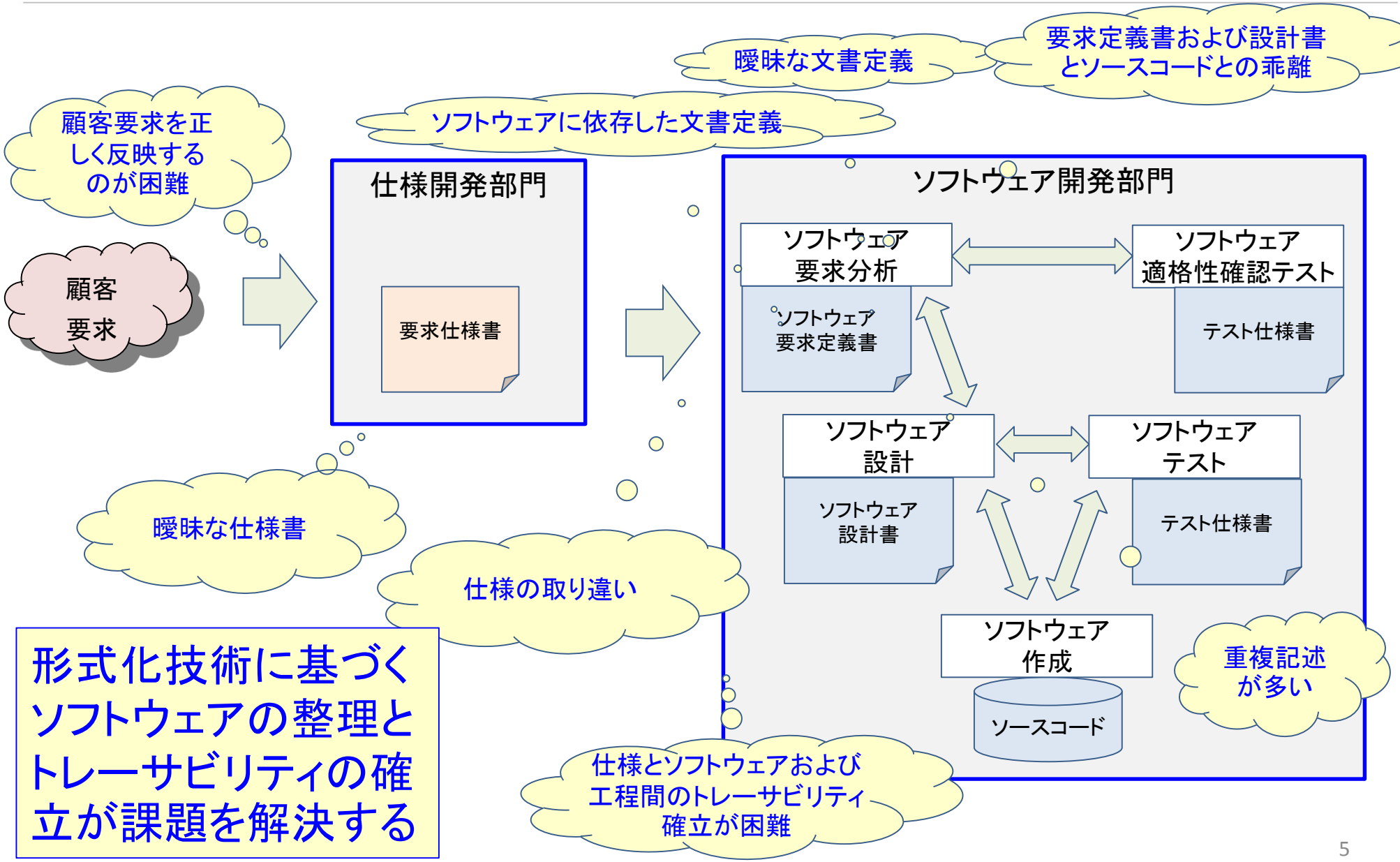
COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

組込みソフトウェアの開発上の課題

- 無くならない仕様とコードのギャップ
- 無くならない仕様の取違い
- 決まらない仕様と変わらない納期
- 似て非なる仕様とコードの増加
- 進まないソフトウェアの部品化
- 増加し続ける機能と機種バリエーション
- 脱却できない個人依存の開発
- 止まらない流出不具合

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

ソフトウェア開発上の課題と解決



形式化技術に基づくソフトウェアの整理とトレーサビリティの確立が課題を解決する

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

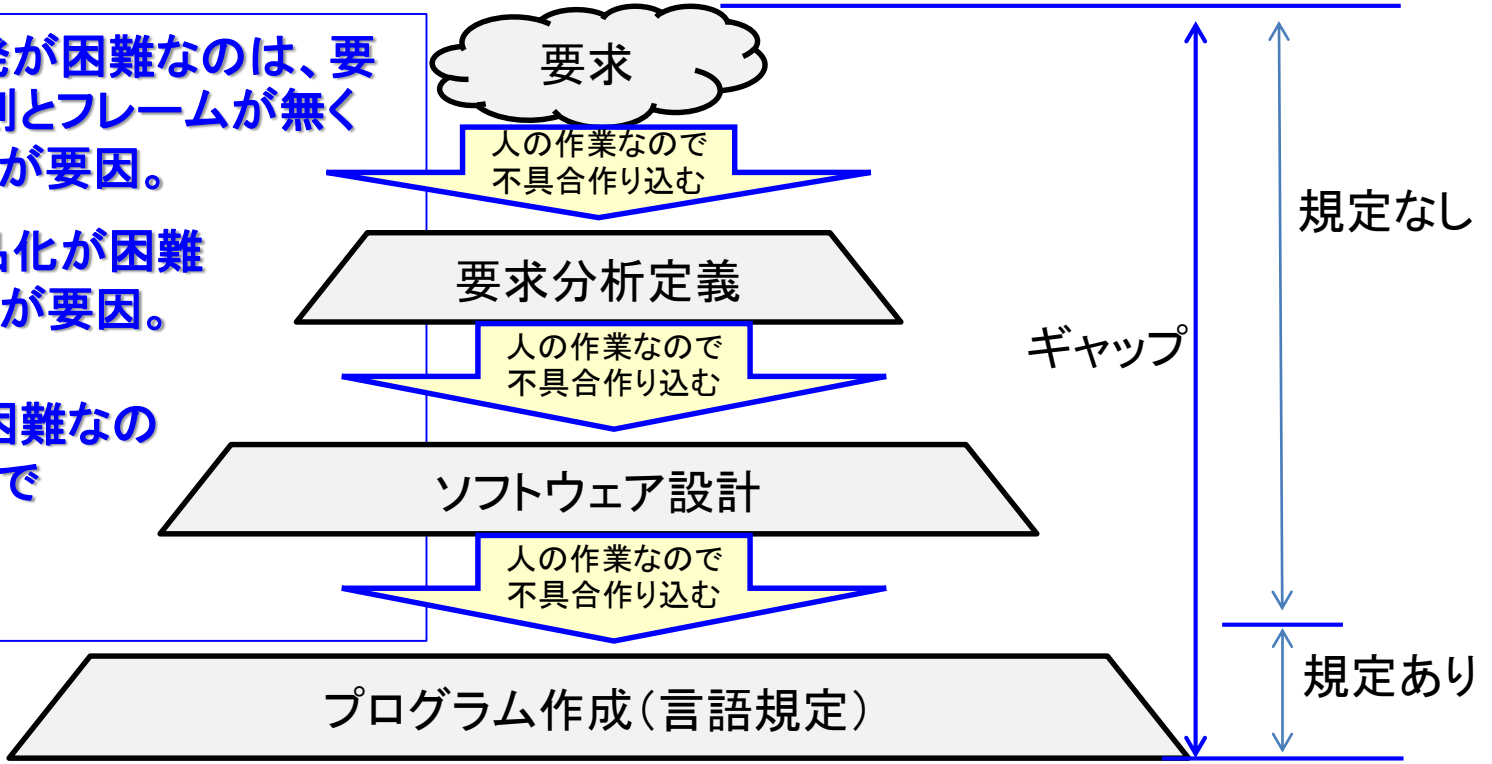
ソフトウェア開発が困難な要因

ソフトウェア開発を困難にしている要因は、抽象度の高い要求を厳密に定義せずソフトウェアを規則性なく作り込むためである。

ソフトウェアの開発が困難なのは、要求の整理変換規則とフレームが無く個人依存となるのが要因。

ソフトウェアの部品化が困難なのは、仕様変動が要因。

仕様の部品化が困難なのは、仕様を日本語で形式化する方法が無いのが要因。



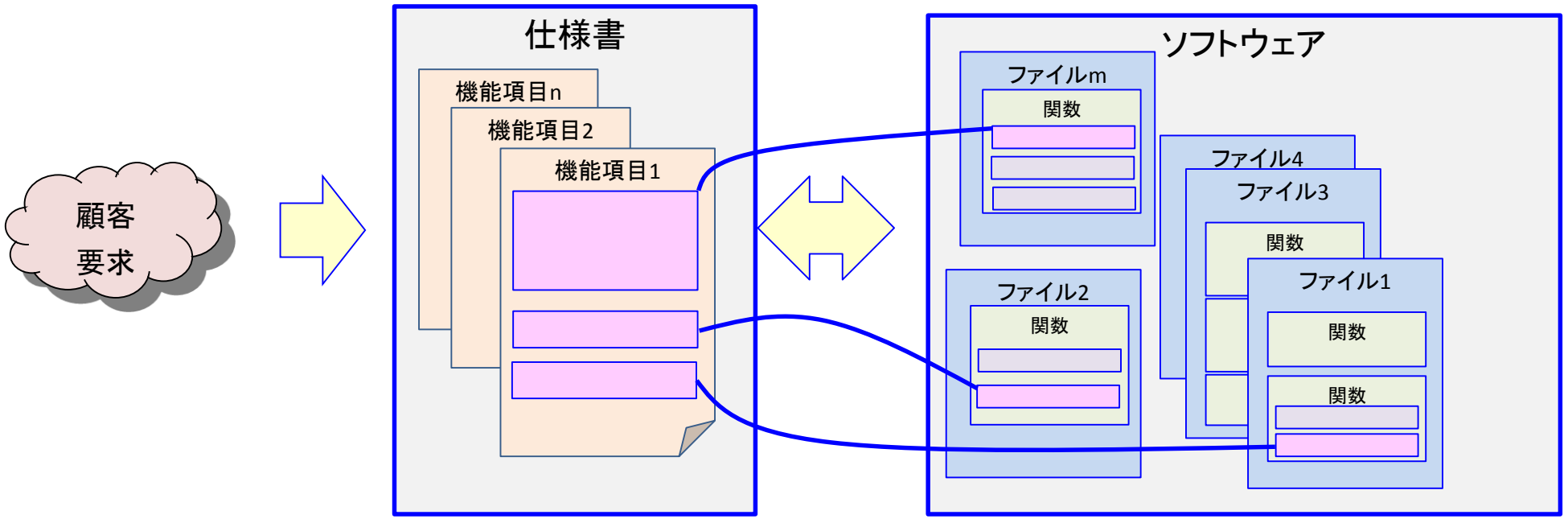
COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

ソフトウェアの開発が困難な理由

1機能項目の記述が複数のファイル内の関数のソースコード記述に対応

- 仕様とソフトウェアの関係の解析が困難
- 1機能項目に対するソースコード記述間の関係の解析が困難
- 仕様記述の要素と設計/実装要素との関係の解析が困難

整理されていない曖昧な仕様記述や仕様記述の重複・不整合がある。
更に仕様とソフトウェアのトレーサビリティが複雑で仕様変更が困難な状況となる



COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

2. AOOでの派生開発

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

派生開発時の課題

仕様およびソフトウェア分析/設計書が不完全である。派生開発時に仕様書およびソフトウェア分析/設計書を変更してソースコードを変更するが不完全な文書記述に基づき開発を進めるため手戻りが多い課題がある。更に派生機種の増加による仕様定義が複雑になる傾向がある。

解決する課題

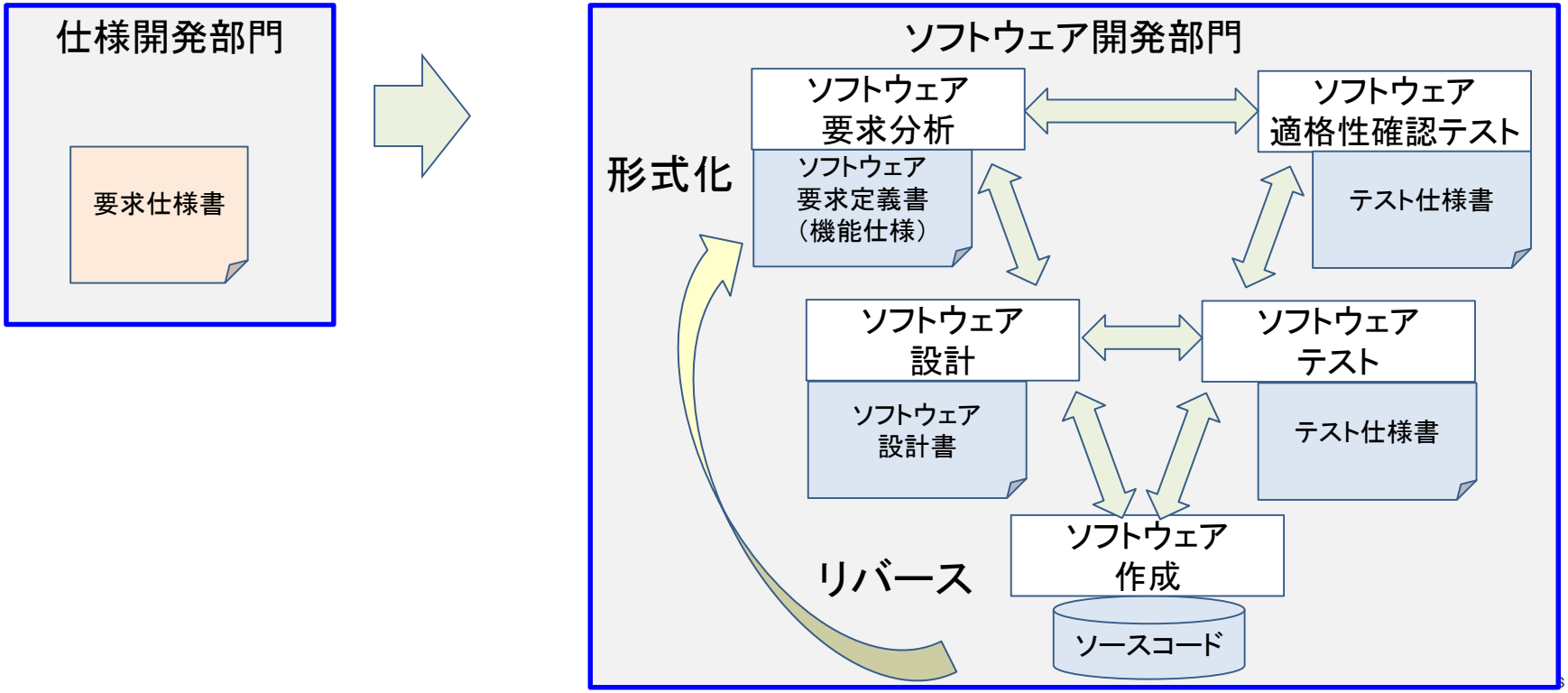
- 文書の記述が曖昧であり重複や矛盾があり完全ではない。
- 文書とソースコードとの乖離がある 。
- 派生した機種の増加により派生した仕様定義が複雑になる。

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

課題解決(上流設計の強化)

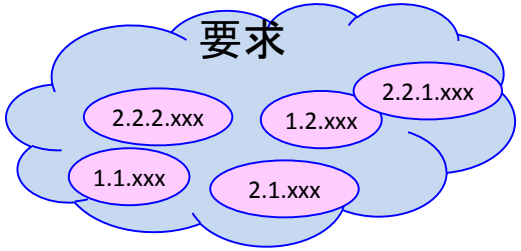
仕様が安定した項目からソースコードを解析して日本語と表形式による形式化で機能仕様をリバースする。リバースした機能仕様に基づき変更開発を実施。機能仕様の記述を設計書およびテスト仕様に変換することで設計品質およびテスト品質向上を図る。

- 分析精度向上:ソースコードから機能仕様をリバースすることでソースコードの分析精度を向上
- 仕様取違い抑制:リバースした機能仕様を仕様開発部門と読み合わせることで仕様の取違いを抑制
- 手戻り抑制:仕様不安定時はソースコード修正を優先して分析/設計/テストを簡略化
- 仕様部品化:機能仕様を分類整理することで重複部分の整理と仕様の部品化を実現
- 記述複雑化抑制:バリエーションの形式記述による仕様定義の複雑化を抑制



AOOによる派生開発手順

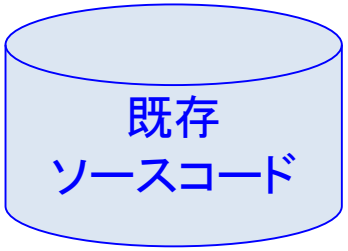
①対象の製品の機能項目リストを定義する（未定義の製品）
②機能項目に対して変更粒度を定義する（FA,FC,FS,FD,RA,RC,RS）
③機能項目に対して仕様安定度を定義する（安定/不安定（H/M/L）/未定）
④変更するソースコードを特定する
⑤仕様が安定したらソースコードから機能仕様書をリバースして機能仕様書を変更する
⑥機能仕様書を仕様開発部門と読み合わせを実施する
⑦承認を得た仕様を開発プロセスに基づきソフトウェアを開発する



作成

機能項目リスト

機能項目			要求	発生源	価値	目的	操作	属性	変更粒度	安定度
XXシステム										
	1.xxx									
		1.1.xxx							FC	安定
		1.2.xxx								
	2.xxx									
		2.1.xxx								



生成

機能仕様書（形式化）

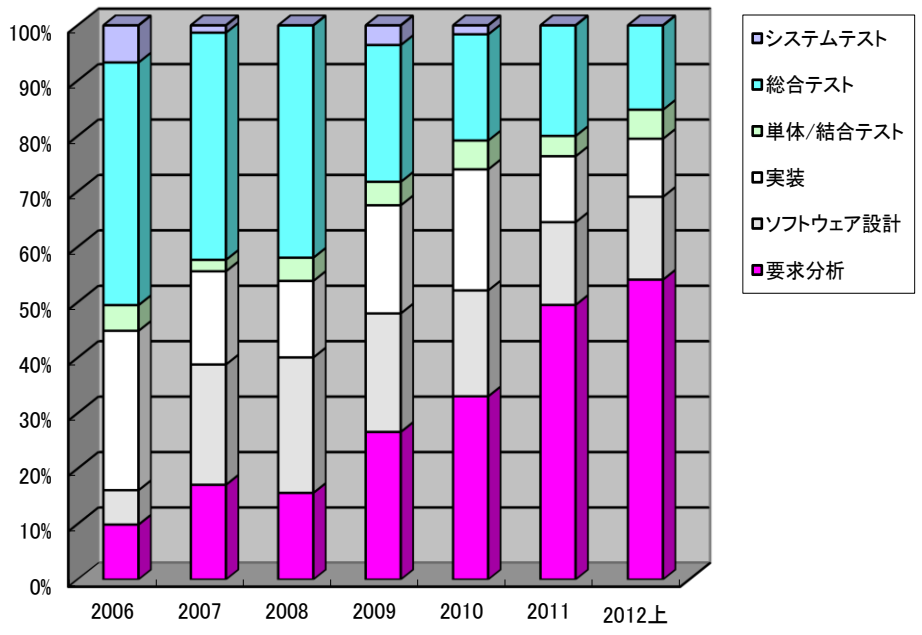
1.1.xxx

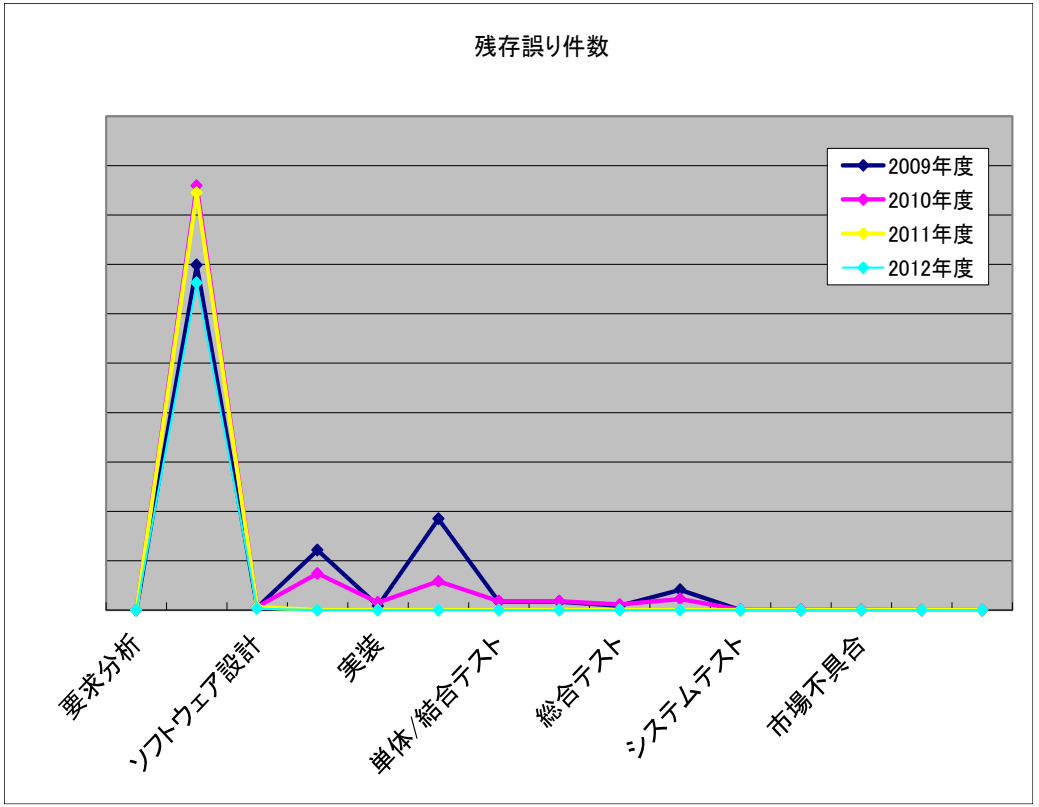
①and②
①外気温度＞10℃
②運転モード＝冷房
・XXX制御状態←制御中
【XXX制御状態＝制御中】

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

AOO派生開発適用の効果

フロントローディングが進み生産性約130%改善

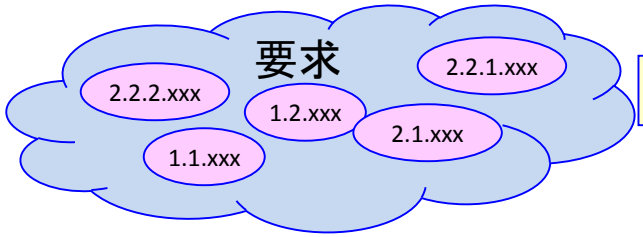




COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

外部ツール連携

仕様定義の手法とのエンジニアリング手法との連携による上流品質の向上



作成

機能項目 AOO	要求	発生源	価値	目的	操作	属性
XXシステム						
1.xxx						
1.1.xxx						
1.2.xxx						
2.xxx						
2.1.xxx						
2.2.xxx						
2.2.1.xxx						
2.2.2.xxx						

UMLクラス図との双方向の連携

連携

UML

仕様記述にAOO形式記述一部適用

作成

USDM

インポート

生成

USDM/AOO双方向連携
USDM仕様生成

要求 → エンジニアリング

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

USDMとの連携

要求仕様の記述およびバリエーションをインポートすることでUSDMに基づいたAOO開発を可能にする。。
効果1：変更仕様に基づき機能仕様書を変更。既存コードからの機能仕様のリバースでコードの自動生成を実現（効率化）
効果2：要求のバリエーションポイントとバリアントをソースコードに紐づけすることでSPL開発を実現（効率化）。

変更要求仕様

1.xxx（記号）	要求	（要求番号）	要求記述	備考欄
		理由	理由記述	
		説明	説明記述	
	<<1.1.xxx>>			
	□	（仕様番号）	仕様記述	
	□	（仕様番号）		
	<<1.2.xxx>>			
2.xxx（記号）				考欄
	<2.2.1>			
	□	（仕様番号）	仕様記述	
	<2.2.2>			
	□	（仕様番号）	仕様記述	

OOは<<保温判定>>を使用する
判定が成立するとヒーターをONする

<湯温保温判定>
①センサ<90℃
<コーヒー保温判定>
①センサ<80℃

インポート

変更仕様に基づく実現仕様の定義

機能項目		要求	発生源	価値	目的	バリエーション				
						バリエーションポイント			バリアント	
						名称	ラベル	形式	名称	ラベル
1.Xxx										
	1.1.xxx					保温判定			湯温保温判定	
									コーヒー保温判定	
	1.2.xxx					変換処理				
2.xxx	2.1.xxx					異常検知				

機能項目リスト

1.1.xxx

①<<保温判定>>=TRUE
・ヒーター←ON

・<OOO>
・<BBB>

<BBB>
XXXXXXXXXXXXXXXX

機能仕様書

既存ソースコード

COPYRIGHT © MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

USDM仕様記述の形式記法の適用

USDM仕様記述にバリエーションの記述の記法を適用した例

バリエーションポイント

バリアント

OOは＜＜保温判定＞＞を使用する
判定が成立するとヒーターをONする

＜湯温保温判定＞
①センサ<90℃
＜コーヒー保温判定＞
①センサ<80℃

コーヒーポット選択（仕様生成）

OOは＜コーヒー保温判定＞を使用する
判定が成立するとヒーターをONする

＜コーヒー保温判定＞
①センサ<80℃

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

3. AOO開発手法のご紹介

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

AOO指針

AOO（Autonomic architecture base Object-Oriented development technique）は、自主研究により1998年に組込みソフトウェア開発向けオブジェクト指向の開発手法として発表したオープンな技術である。その後、AOOは、プロセス(AOO_PRS)、プロダクトライン(AOO_SPL)、見積り(AOO_EST)、形式手法(AOO_DSL)と手法を拡張してツールとして提供準備中。
「整理」「定義」「変換」の3つの指針で構成される。要求の発生源、目的、目的達成で得られる価値を定義する。要求は1機能1目的でカテゴリに階層的に整理して機能仕様を定義する記法と表形式により分析・設計・実装・テストの双方向変換ルールを用いトレーサビリティを確立。更にドメイン特化した言語を定義・再利用することで増大する要求に対応する。カテゴリ共通部とカテゴリ間インタフェースの定義により対象ドメインのフレームワーク構築を可能にする。AOOは、開発上の課題に基づく解決技術であり開発上で発生する課題の真因を分析して未然防止するためのフレーム（ドキュメントフレーム/ソースコードフレーム）とフレームに実装するための開発プロセスを定義することで組織的な課題抑制を可能にする。

【整理】目的指向による形式的モデル定義

- 自律化（要素間の依存関係の抑制）
- 目的での要求分解とカテゴリ化
- 共通化（カテゴリ内の共通部の抽出）
- 類型化と集約（カテゴリ内間の共通フレームの抽出）
- 製品間の可変性定義と統合化
- 無駄取りの徹底（重複情報、重複作業の除去）

【定義】課題抑制のフレームとプロセス定義

- 課題抑制のフレームとプロセス定義
- 形式化された日本語と表形式による仕様定義

【変換】変換ルール化

- 工程間の双方向変換のルール化
- 構造と振る舞いの変換のパターン化

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

仕様品質特性からみた課題解決

ソフトウェア要求仕様の品質評価指標の標準規格
IEEE Std. 830-1998 (Revision of IEEE Std.830-1993) Recommended Practice for Software Requirements Specifications

- Correct（妥当性）
- Unambiguos（非曖昧性）
- Complete（完全性）
- Consistent（無矛盾性）
- Ranked for importance and/or stability（重要度と安定性のランク付け）
- Verifiable（検証可能性）
- Modifiable（変更可能性）
- Traceable（追跡可能性）

①整理技術：要求を分解整理する形式的なフレームが必要
Correct, Complete, Consistent, Modifiable
②定義技術：仕様定義フレームと日本語による形式化が必要
Unambiguos
③変換技術：工程間の双方向一貫性を確立するプロセスが必要
Traceable

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

整理する技術

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

要求の整理

➢ 価値定義による要求開発

1）目的定義：「～したい」の文言で要求の目的を定義
2）発生源定義：要求の発生源を定義
3）価値定義：発生源に対して目的達成で生じる価値を定義

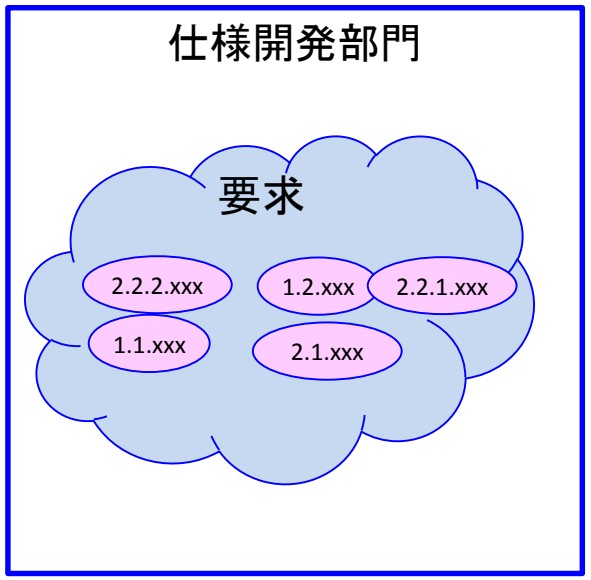
価値の深堀による
真の目的を定義。
- ➢ 真の価値
- ➢ 真の目的

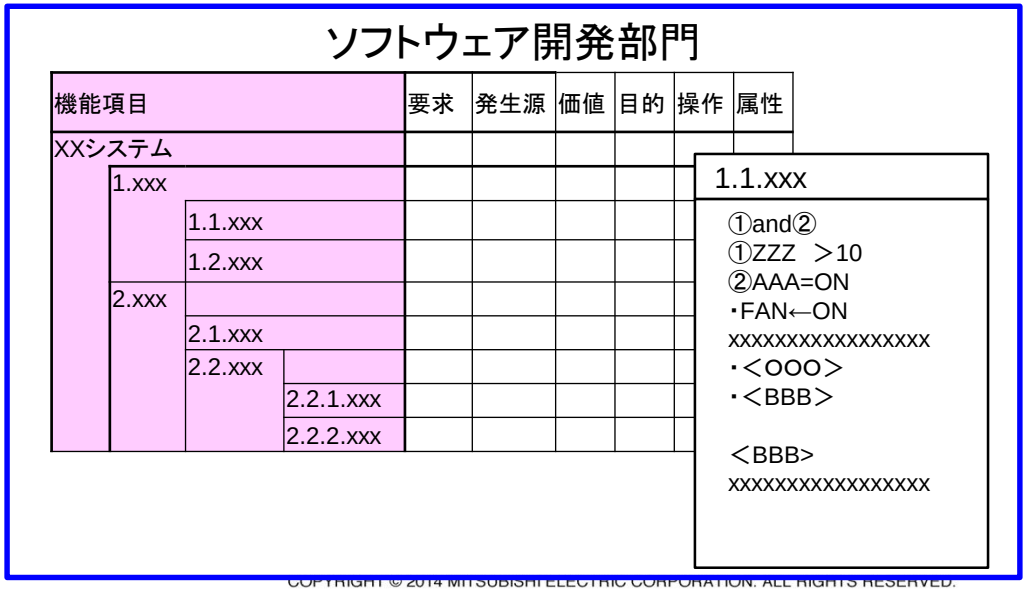
➢ 目的定義による要求の階層化

各要求が持つ目的を定義して、目的を表す機能項目名称を定義する。 同種の目的でカテゴリを定義して要求を階層的に整理する。 最下位層の機能項目が要求の実現手段であり機能仕様を形式的に定義する。

➢ 表形式と日本語記述による形式化

仕様開発部門とソフトウェア開発部門が共通理解可能な言語で要求を形式的に定義する。抽象化技術を使用してモデル化を進め仕様定義の可読性とソースコードとの双方向のトレーサビリティを確保する。





COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

目的指向モデリング

①最上位目的定義

アクターの目的達成の価値を分析して最上位目的を定義する

②副目的を定義（上位目的達成のための目的定義）

目的達成のための下位の目的を定義する

③目的での分類によるカテゴリ化

目的で分類して1つの目的を持つカテゴリを定義する

④目的達成の機能項目の定義

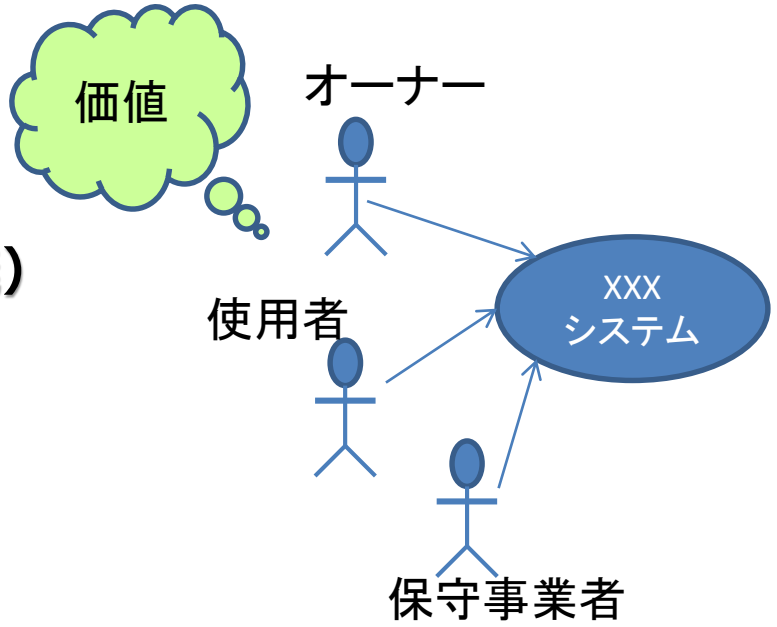
目的を達成するための実現機能を機能項目として実現方法を機能仕様に定義する

⑤カテゴリ内共通機能の抽出

カテゴリ内の機能項目間で共通機能を抽出する

⑥カテゴリ間のインタフェースを定義する

カテゴリ間のインタフェースを定義してフレームワークを定義する



COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

目的定義フレーム

対象システムに関連するアクターすなわち要求の発生源に対してどのような価値を提供することが目的であるかを定義する。
システムの目的を定義してトップの目的を達成する副目的を定義、最下位層は機能ブロックの目的を定義する。

カテゴリおよび機能項目が持つ目的を定義

機能ブロックが持つ目的を定義する

機能項目リスト

機能項目	要求	発生源	価値	目的	機能ブロック			属性							
					名称	ラベル	目的	名称	ラベル	単位	初期値	範囲	項目値		
													名称	ラベル	値
XXXシステム															
1.xxx															
1.1.xxx															
1.2.xxx															
2.xxx															
2.1.xxx															

お客様のご要望に基づき要求の発生源、目的達成の価値を定義

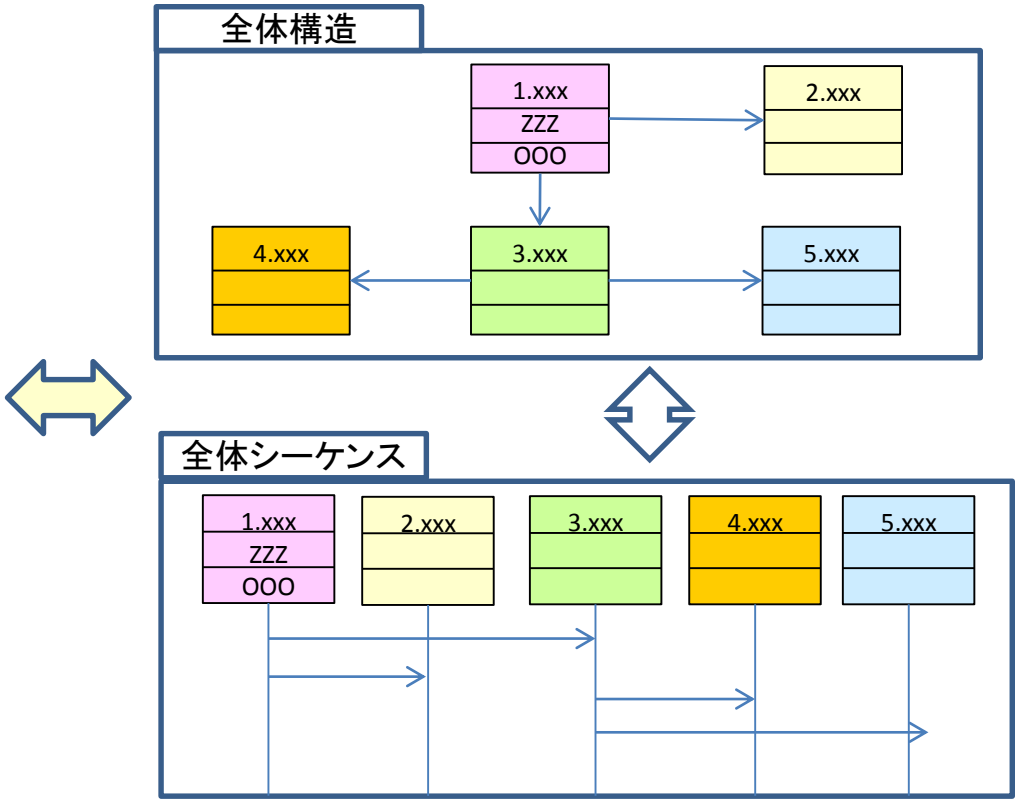
COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

カテゴリ化による整理

①ドメインモデル構築
　対象ドメインに特化したカテゴリで全体構造と振る舞いを定義

②共通部抽出
　要素間で共通な機能ブロック/名詞の抽出⇒類似要素の統合化

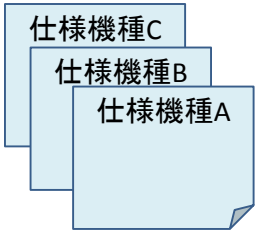
オブジェクト			目的	属性	目的	操作	状態
1.xxx				ZZZ		OOO	
	1.1.xxx						
	1.2.xxx			AAA		BBB	
2.xxx							
	2.1.xxx						
	2.2.xxx	2.2.1.xxx					
		2.2.2.xxx					
3.xxx							
4.xxx							
5.xxx							



COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

表形式モデルによるSPL開発

増加し続ける機種バリエーションの仕様記述に形式記法を用いることでバリエーション記述を分離して仕様記述の複雑化を抑制する。 製品を特徴づける要素であるフィーチャー（物理および機能構成要素）を分析して、フィーチャーをリストに階層的に定義する。これにより製品の特徴の可視化と他社差別化のための戦略的なフィーチャーの先行開発を可能にする。
仕様の可変ポイント（バリエーションポイント）＋可変仕様（バリアント）とフィーチャーを紐づけることでフィーチャーを選択して新製品の仕様構築の加速を可能にする。
バリエーションポイントとバリアントとソースコードを紐づけることで新機種のソフトウェア開発の加速を可能にする。



作成

フィーチャーリスト		
センサ	温度センサ	ＸＸ方式
		ＯＯ方式
	圧力センサ	
人感センサ	赤外線方式	
	簡易方式	
	カメラ方式	

インポート

機能項目リスト

機能項目		操作			属性			バリエーション					システム名称			フィチャーモデル				
													システムラベル			センサ		人感センサ		
		目的	名称	ラベル	目的	名称	ラベル	バリエーションポイント			バリアント		機種A	機種B	新機種	温度センサ XX方式	温度センサ OO方式	赤外線方式	カメラ方式	簡易方式
								名称	ラベル	形式	名称	ラベル	ラベル	ラベル	ラベル					
1.xxx	1.1.xxx							入力変換			XX方式			○	○	○				
											OO方式		○				○			
	1.2.xxx							変換処理			画像			○	○				○	
											ON/OFF		○					○		○
2.xxx	2.1.xxx							異常検知			異常信号			○	○			○	○	
											簡易検知		○							○

仕様の可変ポイント　可変する部分仕様

仕様定義

1.xxxx

① and ②
①＜＜異常検知＞＞＝TRUE
②モード＝運転
・異常状態 ← 異常停止

＜＜異常検知＞＞
[System＝機種A]
＜簡易検知＞
・xxxxxxxxxxxxxxxxxxx
[System＝機種Bor 新機種]
＜異常信号＞
・xxxxxxxxxxxxxxxxxxx

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

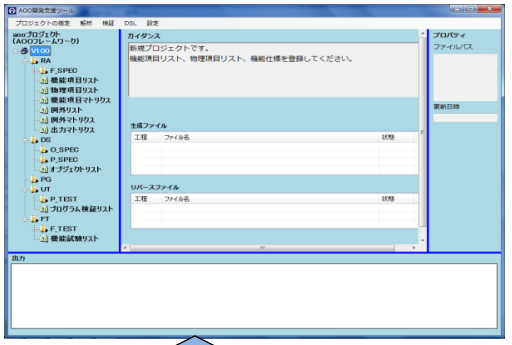
定義する技術

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

表形式と日本語による形式化

機能仕様記述で使用される名詞および機能ブロック(機能仕様記述の部分仕様)を登録・利用を実現することで要求の再利用と編集時に利用可能な名詞および機能ブロックの自動表示と選択を可能にする。

PCシステムで再利用可能な機能は、.NET Framework やJavaのクラス ライブラリに定義されており、必要なクラスやメソッドを選択利用することで効率化を実現している。
機器組込システムでは、クラスライブラリを要求レベルで構築する仕組みと選択利用する仕組みが必要である。対象ドメインに仕様をクラス化して部分仕様を登録して選択使用を可能にすることでツール活用部門独自にライブラリが進化して継続的な効率化を可能にする。



登録DSL選択利用

1.xxx_1.1.xxx_1.1.1.xxx

① 外気温度
運転モード
XXX制御状態

・< 運転判定
タイマ制御
XXX初期化
XXX判定

要求
要求
要求

機能項目リスト

機能項目			目的	属性	機能ブロック
1.xxx				外気温度 運転モード	運転判定 タイマ制御
	1.1.xxx			○○○制御状態	○○○初期化 ○○○判定
		1.1.1.xxx			
		1.1.2.xxx			
	1.2.xxx				
2.xxx	2.1.xxx				
	2.2.xxx	2.2.1.xxx			
		2.2.2.xxx			

物理項目リスト

物理項目			目的	属性	目的
1.xxx	1.1.xxx				
	1.2.xxx				
	1.3.xxx				
2.xxx	2.1.xxx	2.1.1.xxx			
		2.1.2.xxx			

1機能項目1目的として階層的にカテゴリ化を進める

利用

登録

開発支援ツール

ドメイン共通表形式記法・機能ブロック
ライブラリ

共通形式記述言語

機能仕様で扱う名詞と機能ブロックを定義してライブラリに登録・利用することで仕様の部品化とコード自動生成を継続的に進める

期待効果

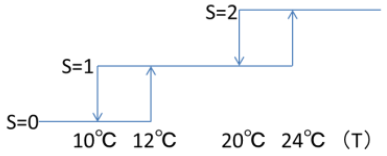
- 仕様の厳密化と抽象化による可読性向上
- ドメインごとの仕様の部品化推進による生産性向上
- 仕様からコードの自動生成による生産性向上

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

モデル駆動開発の構成

STEP3:仕様から抽出したドメインで共通な仕様の目的を示す名称を付加して表形式モデルとする。共通仕様を隠ぺいして可変部のモデルとなることで仕様開発スピードおよび可読性の向上を図る。

表形式によるドメイン共通モデル
＜共通仕様の可変点のモデル化＞



HYSTERESIS_TB

		T								
			10		12		20		24	
S	初期値	0	0	0	1	1	1	1	2	2
	変化UP		-		1		-		2	
	変化DOWN		0		-		1		-	

STEP2:仕様の部分記述を機能ブロック及び処理ブロックとして目的を表す文言で抽象化と厳密化を実現させて可読性を向上させる。更に機能ブロック/処理ブロックのユーザー登録を可能にして仕様の再利用を促進させる。

ユーザー定義モデル（機能ブロック/処理ブロック）
＜ドメインごとに再利用可能な仕様記述を蓄積/再利用＞

＜正常運転判定＞
①and②
①運転／停止指令 ＝ 運転
②組合せ異常状態 ＝ 正常

＜＜組合わせ異常切替判定＞＞
[システム＝AHU]
・＜組合わせ異常切替判定AHU＞
[システム＝マルチS]
・＜組合わせ異常切替判定マルチS＞

Step1:形式記述の記載に意味がある部分を表形式モデルとすることで可読性を向上させる。

表形式によるモデル化

JugeTB：

JG		OP	
JG		OP	

SelectSetDataTB：

JD		SD	
JV		SV	
JV		SV	

Step1:日本語による形式化と仕様と設計の要素を関連付けることによりモデルとコードの双方向のトレーサビリティを確立してモデル駆動開発を実現する。

日本語による形式化

① and ② and ③
①＜組合わせ異常切替マルチS＞ ＝ TRUE
②運転／停止指令 ＝ 運転
③組合せ異常状態 ＝ 正常
・組合せ異常状態 ← 異常停止

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

図式表現と日本語により形式化

1）形式記号による記述仕様の形式化

①～⑳	条件記号
：TRUE	条件満足時の処理を示す。
：FALSE	条件不満足時の処理を示す。
・	操作（処理）の記述を示す
←	代入を示す
【条件】	以下に続く処理の実行イベントを示す
【オブジェクト：イベント】	オブジェクトに対するイベント記述を示す
<<>>	バリエーションポイントを示す。
<>	機能ブロック（バリアント含む）を示す。
xxx[]	複数要素を示す。

2）表形式による形式化

JugeTB:

		JG		JG	
JG		PS		PS	
JG		PS		PS	

SelectTB:

JD			
JV		PS	
JV		PS	
JV		PS	

SelectSetDataTB:

JD		SD	
JV		SV	
JV		SV	
JV		SV	

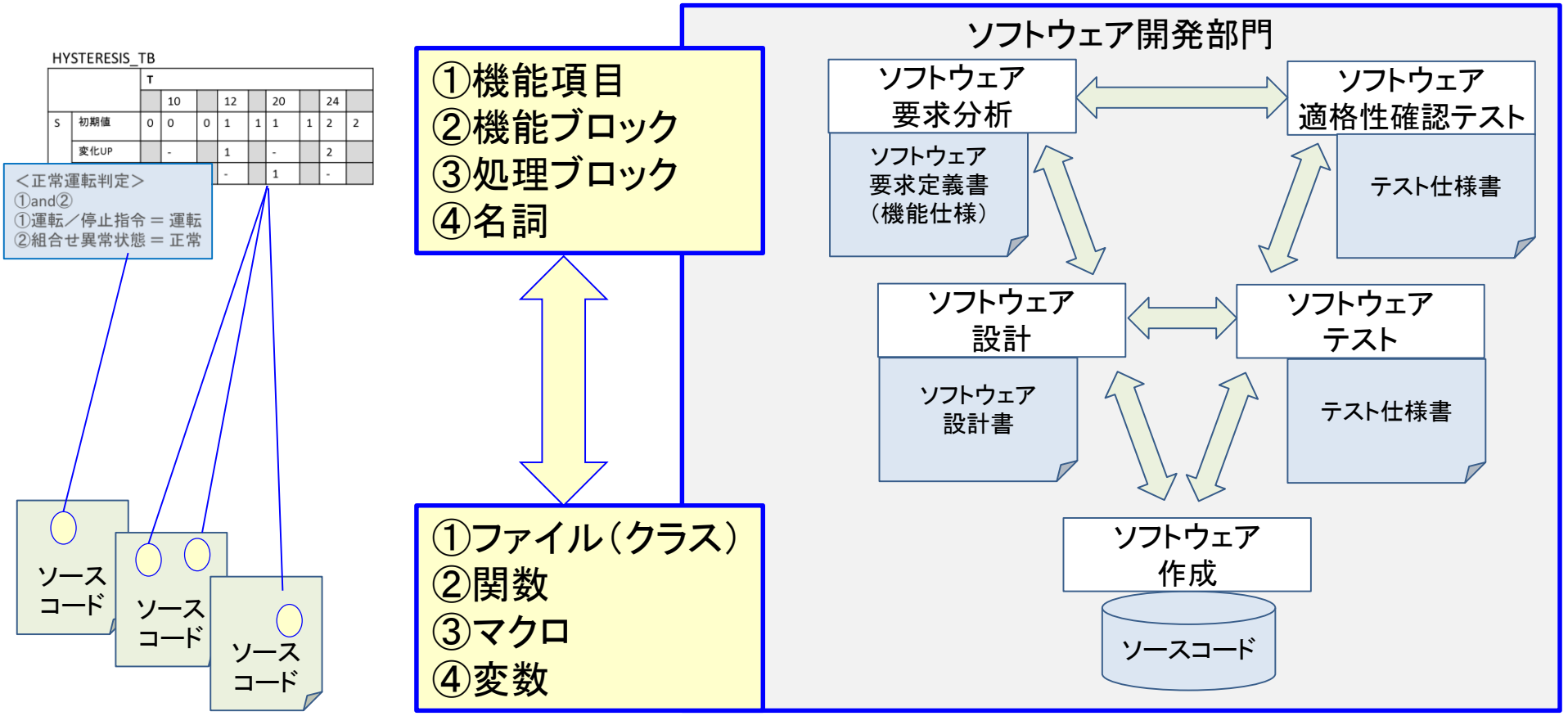
COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

変換する技術

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

日本語の要素とコードの要素の紐づけ

機能仕様書の日本語記述内の要素とソースコード内の要素を紐づける。
共通仕様とソースコードに紐づけにより仕様とコードの資産化を実現する。



COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

まとめ

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

AOOのよる派生開発まとめ

既存ソースコードから日本語と表形式の形式化で機能仕様をリバースする。
機能仕様に対して変更追加を行うことで上流設計で派生開発を実現する。

既存ソースコードから機能仕様書をリバースする作業負荷が大きいためツールにより機能仕様のリバースを支援する。
機能仕様をリバースした時点で、ソースコードと機能仕様が紐づく。
これによりモデルベース開発が可能になる。
更に機能仕様の記述のクローンを抽出して、クローン記述に特定の目的があれば、目的を示す名称で仕様のライブラリ化を図る。この作業は対象ドメインで固有な言語を構築する作業になりツールでそれを支援する。
再利用可能な仕様とソースコードが紐づくことで既存ソースコードの資産化が進む。更に、プロダクトラインのバリアントとソースコードを紐づけることで既存ソースコードの資産化が可能になる。

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.

ご清聴ありがとうございました

COPYRIGHT © 2014 MITSUBISHI ELECTRIC CORPORATION. ALL RIGHTS RESERVED.