65

セーフティ＆セキュリティ設計のための モジュラーアプローチと準形式手法の設計現場への導入[1]

～ソフトウェアの５Ｓ３定で欠陥の少ないソフトウェアを作る～

1. はじめに

セーフティ（安全）を求められる機器のソフトウェア開発、またセキュリティを求められる機器の開発では、システマチック故障の原因となる欠陥の予防が論点の一つになる。このような欠陥を予防するため、機能安全規格の JIS C 0508-3:2014 (IEC 61508-3:2010)では、モジュラーアプローチや準形式手法の導入を強く推奨している[2][1]。

これらが意図通りの効果を発揮するには、単に形をまねるだけではなく、設計者が技術の本質を理解し、適切に適用する必要がある。例えばモジュラーアプローチでは、情報隠蔽の規定がある。しかし浅い理解の基に、安易にアクセサメソッドを用いれば情報隠蔽はできない。このような設計は、実質的にグローバル変数と同じである。変数を参照する処理がソフトウェア内に散在し、故障発生時の影響範囲を特定できない。そのためセーフティ設計は困難になる。同じくモジュールサイズの制限も規定されている。しかし数百行にもなる関数が散在するなら、この規定は満たせない。残念ながら設計現場では、このようなコードは稀ではない。

本事例では、一般的な設計者が、このような欠陥予防の技術を適切に扱えるように、①分かりやすい言葉による説明で導入の敷居を下げ、②誰でもできる手順と規準とで技術を扱えるようにし、③自動化の仕組みを作ることで良し悪しを直ぐに確認できるようにした。

この分かりやすい言葉による説明には、製造業で一般的な５Ｓ（整理、整頓、清掃、清潔、しつけ）と３定（定位、定品、定量）を採用した。ソフトウェア設計に５Ｓ３定を取り入れることで、モジュラーアプローチと準形式手法のスムーズな導入につながった。

2. セーフティおよびセキュリティ設計のポイント

最初にセーフティおよびセキュリティ設計の基本的な考え方を説明する。次にソフトウェア設計に固有の論点として欠陥を起因とするシステマチック故障との関係を説明する。最後に欠陥予防の技術としてモジュラーアプローチと準形式手法とを説明する。

2.1 セーフティ＆セキュリティ設計の考え方

ISO/IEC GUIDE 51:1999 によれば、セーフティとは「受容できないリスクが無いこと」と

---

[1] 事例提供: セイコーエプソン株式会社 ソフトウェア品質・生産技術部 萩原 豊隆 氏

[2] 「強く推奨」となる場合、技法を使用しないときは、使用しない理論的根拠を示す必要がある。

定義されている。リスクとは「危害の発生確率及びその危害の程度の組み合わせ」である。受容できるかは、最終的には設計対象の製品がもたらす効用と、その製品の使用によって引き起こされるリスクとのバランスによって決定される。

従ってセーフティ設計は、図 65-1 に示すように、①リスク分析によってリスクを推定する、②推定したリスクを評価し、受容できない場合は、リスクコントロールを実施する、③リスクコントロールによって受容可能なリスクになったかを再度評価する、という一連のリスクマネジメントプロセスを伴う[3] [2]。単に欠陥がなく故障が発生しないだけでは、セーフティは達成できない。リスクマネジメントプロセスによってリスクが受容可能であるかを評価する必要がある。故障なく常に危害をもたらす仕様の製品は、信頼性は高くともセーフティとは言えない。

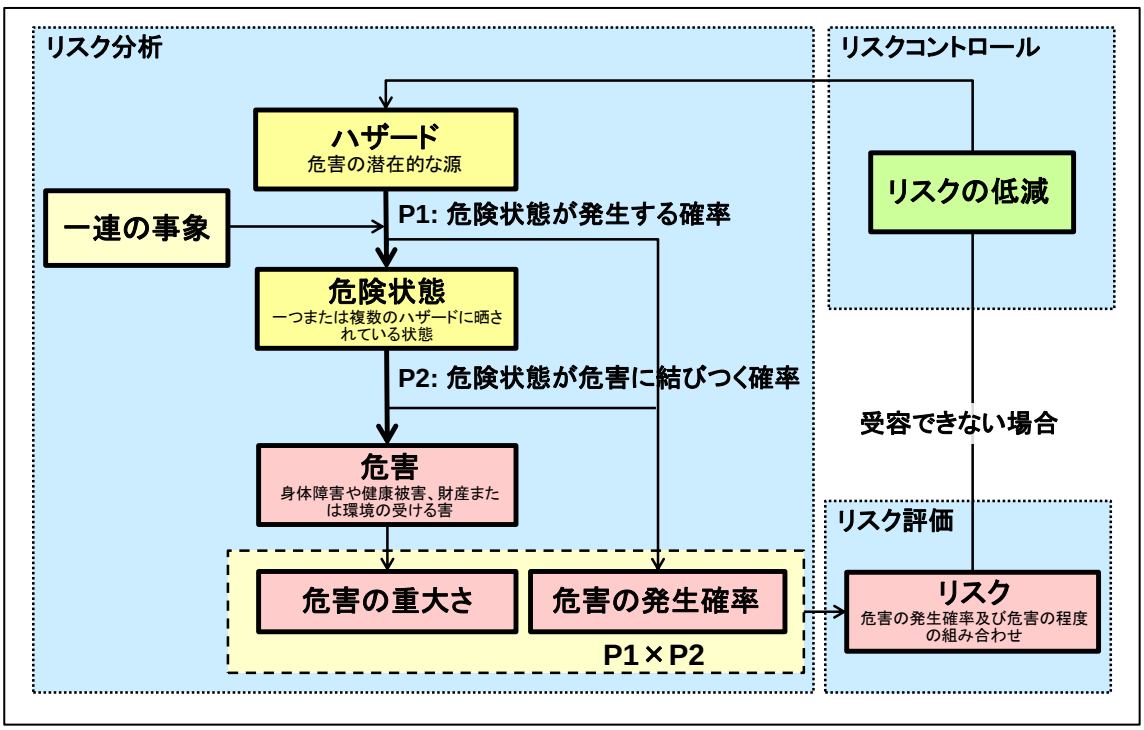

図 65-1　リスクマネジメントのプロセス

またセキュリティもリスクマネジメントプロセスと切り離すことはできない。セーフティのためのセキュリティ設計では、ハザードが危険状態に至る一連の事象に、セキュリティ上の脅威が関与しないように対策する必要がある。

セーフティ設計で、ハザードが危険状態に至る一連の事象を分析する際には、図 65-2 に示す正常使用の範囲を考慮すればよい[4] [3]。しかしセキュリティ設計では、使用者の行為の範囲

[3] JIS T 14971:2012(ISO 14971:2007) 図 E.1 「ハザード、一連の事象、危険状態および危害の関係の図式」を元に一部修正し作成した。

[4] IEC 62366:2007 医療機器－医療機器へのユーザビリティエンジニアリングの適用　図 B.1 「予測可能な使用者の行為のカテゴリ」を基に作成した。

に関してネットワーク等を経由した使用についても考慮が必要となる。また意図する行為のうち、異常使用、つまり禁止されている行為やあり得ない違反行為などに分類される行為までも考慮が必要になる。

このような異常使用は無限に考えうるため、脅威分析をすることで範囲を限定する。例えばソフトウェアが関与するサイバーセキュリティでは、危険状態の一因になるソフトウェアアイテム[5]を特定し、そのソフトウェアアイテムを保護資産として脅威分析を実施する[6]。

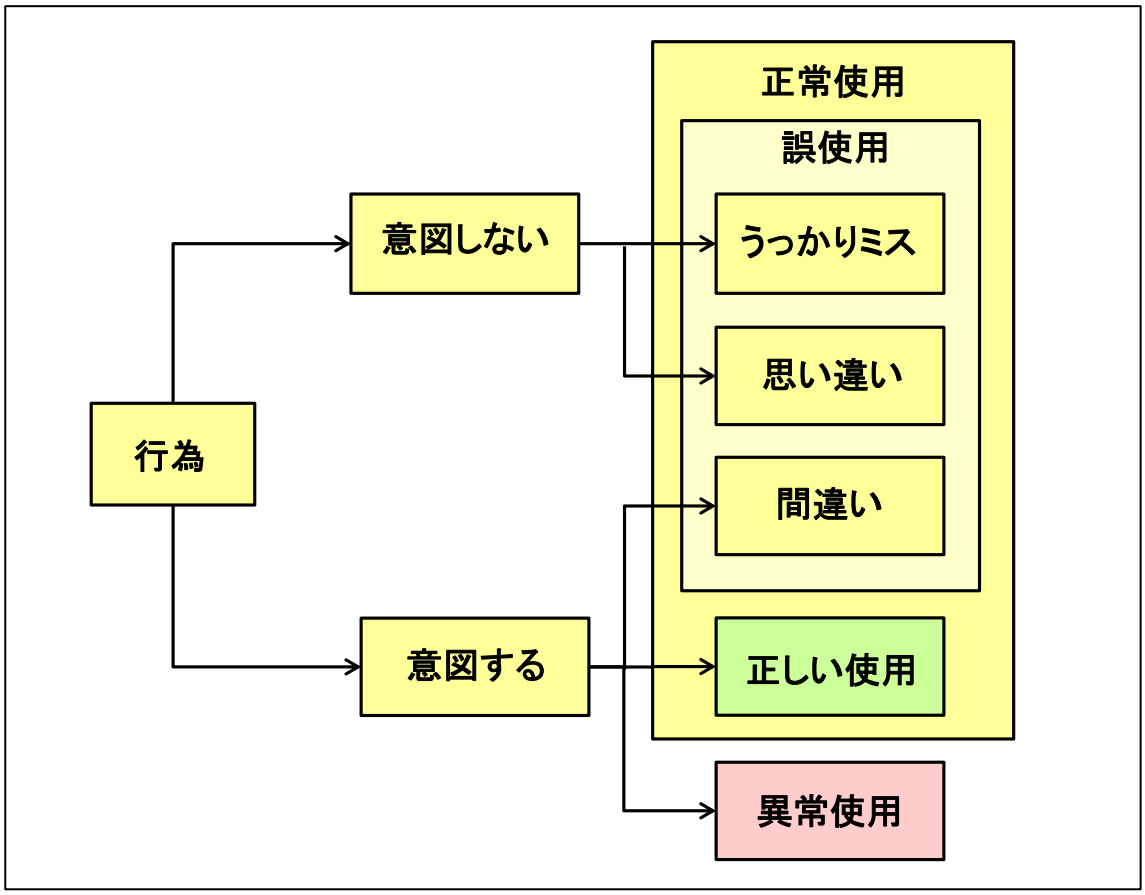

図 65-2　予測可能な使用者の行為のカテゴリ

[5] 「コンピュータプログラムの識別可能な部分」を示す(JIS T 2304:2012)。

[6] これだけでは「他機器に対するサイバーセキュリティ攻撃の踏み台にする」等の行為を防げない。別途、情報セキュリティの範囲でも、保護資産を特定し脅威分析する必要がある。

2.2 ソフトウェア設計に固有の論点

ソフトウェア設計に固有の論点として、システマチック故障とセーフティおよびセキュリティ設計との関係について説明する。

2.2.1 セーフティ設計

ハザードが危険状態に至る一連の事象にソフトウェアが関与する場合、誤った、または不完全な仕様、合理的に予見可能な誤使用の他、ソフトウェアのシステマチック故障がその要因になる[4]。従ってソフトウェアのセーフティ設計の際にはシステマチック故障の防止の対策を取る。さらに故障発生が危険状態に結びつかないように設計上の対策も併用する。

システマチック故障の防止では、欠陥の予防および除去を実施する。ソフトウェアは、実行中に欠陥に遭遇すると、期待した機能から逸脱し故障となる。従ってこの欠陥を予防し除去することが故障の防止に有効である。欠陥の予防は、設計時に適切なエンジニアリング技術を導入ことで達成する。欠陥の除去は、検証面を強化した開発プロセスを導入し、残存する欠陥を除去することで達成する。

また設計上の対策では、故障の伝搬防止を基本とする安全信頼機構を設計する[5]。ソフトウェアアイテムの局所的な故障が、ソフトウェアサブシステムあるいはシステムの故障に結びつかなければ危険状態に至らない。従って、故障の伝搬つまりカスケード故障の防止策となる技術を安全信頼機構に採用する。これらは分離[7]あるいはパーティショニング[8]と言われる技術になる。なお安全信頼機構自体にも欠陥が混入する可能性がある。安全信頼機構に対する欠陥予防および除去の実施と、安全信頼機構の導入に伴うリスクの再評価も必要となる[9]。

2.2.2 セーフティのためのセキュリティ設計

システマチック故障が一連の事象に関与すると危険状態が生じる。セーフティ設計では、この関与を考える際に、図 65-2 における正常使用の範囲を考慮する。しかしセキュリティ設計では、意図する行為のうち、異常使用までを考慮する必要がある。

一般的にサイバーセキュリティ攻撃は、正常使用の範囲ではシステマチック故障として顕在化しない欠陥を対象に行われる。異常使用に類する攻撃により、潜在的な欠陥をシステマチック故障として顕在化させ、ハザードを危険状態に結びつける。例えば、バッファサイズのチェック漏れという欠陥は、正常使用では問題にならない。しかし通常ではありえない使用方法によって、サイズをオーバーするデータを与えることで故障を引き起こせる。この故障が危険状態に至る一連の事象に関与すれば、セーフティは実現できない。

最初にセーフティ設計で、危険状態の一因になるソフトウェアアイテムを特定し、欠陥の予防および除去を実施する。これによって設計あるいはコードに起因する脆弱性を塞ぐ。次にセ

---

[7] JIS T 2304:2012 (IEC 62304:2006) による用語

[8] ISO26262 Part 6, Part9 による用語

[9] 安全信頼機構は、設計に複雑さをもたらす。従って機構の導入によりシステマチック故障の可能性が増す。導入によって新たなリスクが生じないか、リスクを再評価する必要がある。

キュリティ設計で、それらアイテムを保護資産として脅威を分析し、異常使用によってハザードが危険状態に至る一連の事象を現実的な範囲に限定する。限定により、その事象に関与する故障が、直ちに危険状態に結びつかないように設計上の対策を導くことができる。

2.3 欠陥を予防する代表的な技術

欠陥予防に関わるエンジニアリング技術として代表的なものを表 65-1 に示す。これらは、JIS C 0508-3:2014 (IEC 61508-3:2010) で推奨される技術の抜粋である[10]。セーフティ設計では、危険状態の一因になるソフトウェアアイテムの特定した上で、これらの技術を適用する。

なお昨今では、実質的にセーフティであるだけでなく、第三者に対する説明責任も求められる。しかし職人技による開発ではセーフティであることの理論的根拠が示せない。プロセス的な対策も併用し、このような現在の技術水準における妥当な対策（state-of-the-art）となる技術を適切に適用させ、説明責任を果たせる状態にすることも急務となる。

表 65-1　欠陥を予防するエンジニアリング技術[11] [12]

番号	手法及び技法	SIL1	SIL2	SIL3	SIL4
7	モジュラーアプローチ	HR	HR	HR	HR
8	信頼性確認及び検証済ソフトウェア要素の使用（利用できる場合）	R	HR	HR	HR
9	ソフトウェア安全要求仕様及びソフトウェアアーキテクチャとの間の前方トレーサビリティ	R	R	HR	HR
10	ソフトウェア安全要求仕様及びソフトウェアアーキテクチャ間の後方トレーサビリティ	R	R	HR	HR
11a	構造化図表法	HR	HR	HR	HR
11b	準形式手法	R	R	HR	HR
11c	形式的設計手法及び精緻化手法	－	R	R	HR
11d	自動ソフトウェア生成	R	R	R	R
12	コンピュータ支援仕様書作成ツール コンピュータ支援設計ツール	R	R	HR	HR

本事例では、これらのエンジニアリング技術のうち、「モジュラーアプローチ」と「準形式手法」とを取り上げる。この２つの技術は、安全信頼機構の設計においても基礎として必要となる技術であり、その他のエンジニアリング技術の基礎となる技術でもある。

[10] ここに取り上げていない技術は、主に安全信頼機構の設計に関わる技術である。

[11] JIS C 0508-3:2014 (IEC 61508-3:2010) 表 A.2 ソフトウェア設計及び開発－ソフトウェアアーキテクチャ設計。

[12] SIL は安全度水準を意味する。SIL1 が低く SIL4 が最も高い。HR は「該当安全度水準において技法又は手段を使用することを強く推奨する」、R は「HR よりも低い推奨事項としての技法又は手段が望ましい」である。例えばモジュラーアプローチは、全ての安全度水準で強く推奨となっている。

2.3.1　モジュラーアプローチ

モジュラーアプローチは、1970 年代の D.L.Parnas によるデータ抽象と情報隠蔽の提案に起源をもつ。いわゆるオブジェクト指向もこの考え方の延長に存在する。表 65-2 にモジュラーアプローチの構成要素を示す。

表 65-2　モジュラーアプローチ[13]

番号	技法および手段	SIL1	SIL2	SIL3	SIL4
1	ソフトウェアモジュールサイズの制限	HR	HR	HR	HR
2	ソフトウェアの複雑さの制御	R	R	HR	HR
3	情報隠蔽およびカプセル化	R	HR	HR	HR
4	パラメータ数制限及び固定数のサブプログラムパラメータ	R	R	R	R
5	サブルーチン及び関数における１入口点又は１出口点	HR	HR	HR	HR
6	完全に定義したインタフェース	HR	HR	HR	HR

モジュラーアプローチでは、「ソフトウェアモジュールサイズの制限」と「ソフトウェアの複雑さの制御」をする。これらはレビューによる検証の精度を向上させるために有効である。レビューでは、原理的に人間の認知能力を超える対象に対して精度よく欠陥を摘出することはできない。人間が同時に認識できる塊は７±２に過ぎない[6]。従ってモジュールサイズや複雑さの制限は、レビュー精度を向上させるための有効な手段となる。

「情報隠蔽およびカプセル化」は、複雑さを押さえてレビュー精度の向上に寄与する他、ソフトウェアアイテム間の分離の基盤を作るために有効である[14]。分離を実現するには、その前提として、アイテム間の関係を最小限に抑える必要がある。そのためにはアイテムの内部事情（データ構造、処理手順、処理の起動条件）を外部に晒さないようにすること、外部の不正な関与から内部事情を保護することの二点が必要になる。この二点を実現する方法が情報隠蔽とカプセル化である。情報隠蔽によって内部事情を外部に晒さないようにし、カプセル化によって外部から内部事情を保護する。

残りの「パラメータ数制限及び固定数のサブプログラムパラメータ」、「サブルーチン及び関数における１入口点又は１出口点」、「完全に定義したインタフェース」も、検証精度を向上させる手段となる。これらはレビューによる検証だけでなく、テストによる検証にも有効である。例えば、パラメータ数の制限は、構造的カバレッジを実現するテストケースの量を減らす効果がある。またテストケースの設計には、完全に定義されたインタフェースが必要である。

---

[13] JIS C 0508-3:2014 (IEC 61508-3:2010) 表 B.9 モジュラーアプローチ

[14] この基盤の上に、エラーの検出とハンドリングの仕組みを設計することで分離を実現する。システマチック故障によってソフトウェアアイテム間の意図しない副次的悪影響が生じる。このような干渉をエラー検出によって発見し、エラーハンドリングによって正常範囲に是正する。

2.3.2　準形式手法

準形式手法とは、「記法を厳密に定義し、さらにツールによって定義された記法からの逸脱を防止する」手法を指す。モデリングツールを用いた UML によるモデルの記述は、準形式手法の条件を満たす。厳密に定義された記法とそこからの逸脱の防止により曖昧さを排除し、精度良くレビューを実施できる。

なお UML のモデルとコードとのマッピングルールを定めることで、準形式手法とモジュラーアプローチを併用できる。図 65-3 にクラス図とコードとのマッピングルール例を示す[15]。

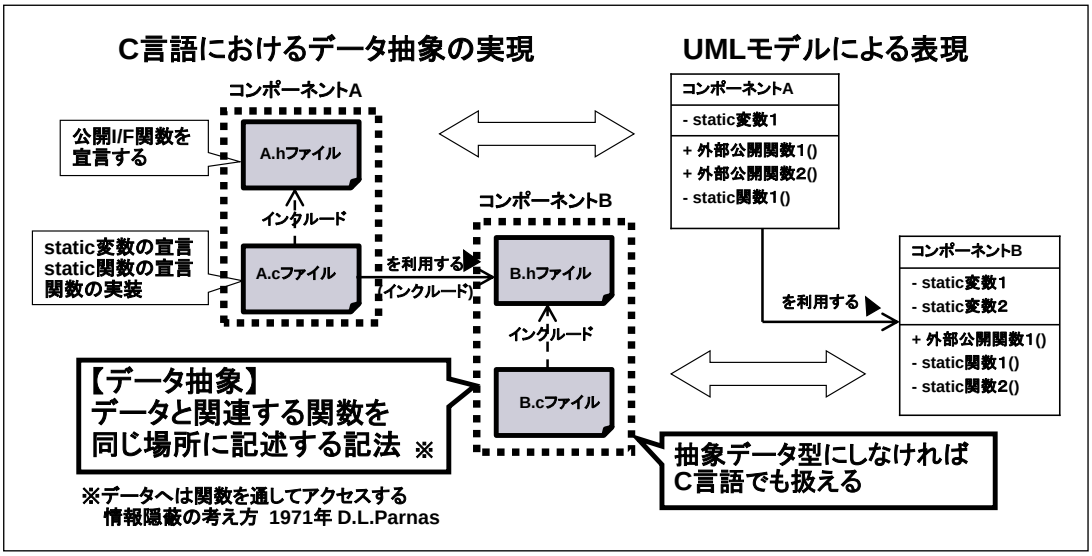

図 65-3　クラス図の C 言語へのマッピングルール例

C 言語でモジュラーアプローチを実践するには、「.c ファイル」に変数（データ）とその変数に関係する関数とを実装する。ファイル外に公開しない変数および関数については static 宣言する[16]。さらに変数に対しては関数を通してアクセスするようにし、アクセス用関数を同名の「.h ファイル」に定義する。この考え方をデータ抽象と言う。このときアクセス用の関数を適切に設計すれば情報隠蔽も実現できる。

マッピングルールを定義することで、このコード上のモジュラーアプローチと UML のモデルとを厳密に対応させる。対応によってクラス図上でもモジュラーアプローチの検討ができる。またマッピングルールは自動ソフトウェア生成につながる。例えばコンピュータ支援設計ツールを用いる MDD（モデル駆動開発：Model-Driven Development）では、UML モデルからのコード生成に対応している。

---

[15] これはソフトウェアユニット（他のアイテムに分割できないソフトウェアアイテム、JIS T 2304:2012）へのマッピングルールである。他にも複数アイテムを内部構造にもつソフトウェアアイテムとの対応も定義できる。この場合、クラス図はアーキテクチャ設計レベルの構造を表現する。

[16] C 言語の規約上、static 宣言された変数と関数は、ファイルスコープとなる。

3. モジュラーアプローチの導入

一般的な設計者が、モジュラーアプローチを活用するためには、以下の３つの敷居がある。これらの敷居を超える施策がなければ、技術を適切に適用できない。

①具体的な使用イメージが理解できない
②使用イメージが理解できても手が動かない
③手が動いたとしても習慣として根付かずすぐに元に戻る

第一の敷居は、技術が具体的な使用イメージを伴って理解できないことである。設計者にとってモジュラーアプローチの構成要素は、おそらく一度は聞いたものである。しかし単に言葉を知っているのと、本質を理解しているのとは異なる。言葉として聞き流すのではなく理解に至らせるように、行動がイメージできる言葉で働きかける必要がある。

第二の敷居は、理解しても手が動かないことである。成果につなげるためには、理解を行動に結びつける必要がある。具体的な手順と規準とで行動を規定することで、設計者は初めて動けるようになる。

第三の敷居は、行動が習慣として定着しないことである。安定的な成果となるには、行動が一過性であってはならない。行動の良し悪しを検証し、すぐにフィードバックする。これを繰り返すことで習慣になり安定的な成果となる。

3.1 ソフトウェアの５Ｓ３定で心に引っかかりを作る

モジュラーアプローチに関する具体的な使用イメージを設計者に理解させる。そのために聞き流されないようにする。ワンフレーズでスローガンを語り、合わせて具体的な行動を提示するとよい。

そこで表 65-3 のソフトウェアの５Ｓ（整理、整頓、清掃、清潔、しつけ）を考案した。５Ｓは、製造業では一般的である。これをソフトウェア設計に適用したのがソフトウェアの５Ｓである。普段社内で使われる言葉は忘れがたい。また５Ｓの具体的な行動についても、一度は指導を受け実践している。そのためソフトウェアに置き換えたとしても、具体的な行動がイメージしやすい。

このソフトウェアの５Ｓをクラス図上で実施したものが図 65-4 である[17]。クラスとコードとのマッピングルールを図 65-3 のものとすると、クラス名がファイル名に相当する。従って適切なクラス名は、適切なファイル名に相当する。ファイルは変数と関数の置き場所となる。そのファイルに誰でも分かる名前をつけて、名前に相応しい変数と関数とを集める。５Ｓの「整頓」に相当することをクラス図上で実現している。

[17] このクラス図は、SESSAME　話題沸騰ポット 要求仕様書 第 7 版を元に作成した。
http://www.sessame.jp/workinggroup/WorkingGroup2/POT_Specification_v7.PDF

表 65-3　ソフトウェアの５Ｓ[18]

項目	説明
整理	要るものと要らないものとを明確に分けて、要らない変数・関数、インクルードを削除する。
整頓	必要な時に、必要な変数と関数が使えるように、置き方や位置を決め、置き場所となるファイルに名前をつけて、誰にでもわかるように明示する。
清掃	適宜にコードを掃除し、関数名や変数名、処理やコメントを見直し、きれいにしておく。
清潔	整理・整頓・清掃を繰り返し行うことで、３Ｓを維持する。
しつけ	設計ルールなど、決められた通りに守る習慣をつける。

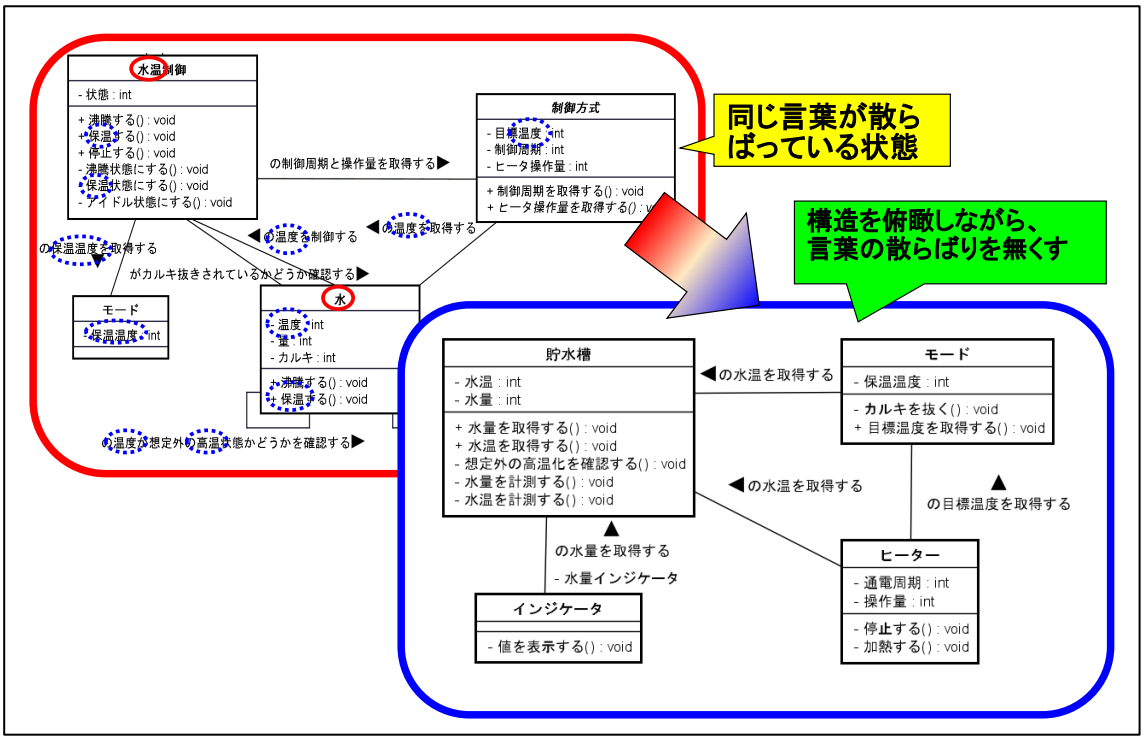

図 65-4　クラス図による 5S の実施例

[18] これはファイルレベルの５Ｓになる。関数レベルでも、演算の置き場所となる関数に名前を付けることで、同様に５Ｓを実施できる。

なおソフトウェアの５Ｓをソフトウェア工学的な側面から説明すると、図 65-5 のようになる。５Ｓで変数（データ）に対して関係する関数（機能）が集まる。これはモジュラーアプローチの起源となるデータ抽象の実現に相当する。データに対して機能を集めることで、ソフトウェアアイテムの凝集度を改善し、同時にアイテム間の結合度を下げている。この状態を達成後、適切なインタフェースを設計することで、モジュラーアプローチの構成要素である情報隠蔽とカプセル化を実現する。

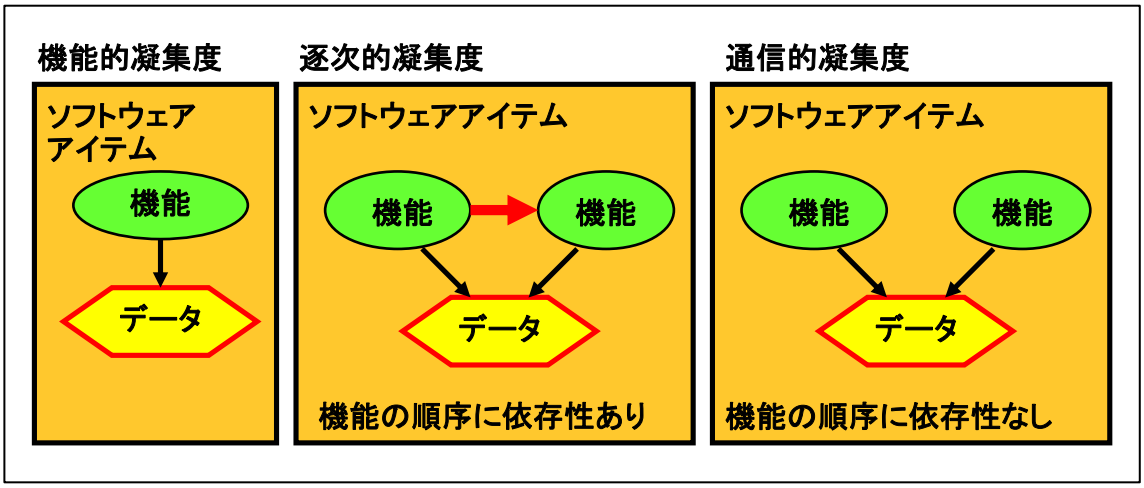

図 65-5　凝集度と結合度の改善

なお５Ｓ活動では、同時に３定管理（定位、定品、定量）も実施する。ソフトウェアの３定管理の項目は、表 65-4 のようになる。これらを定量的に管理することで３定の状態を維持できる[19]。これはモジュラーアプローチの構成要素である「ソフトウェアモジュールサイズの制限」と「ソフトウェアの複雑さの制御」に相当する。

表 65-4　ソフトウェアの３定管理

	ソフトウェアアイテム	関数
定位 （どこに）	■ 変数と関数の置き場所をファイル名で明示する。	■ 演算の置き場所を関数名で明示する。 ■ データの置き場所を変数名で明示する。
定品 （何を）	■ 変数と、その変数にアクセスする関数は同じファイルに置く。	■ 管理に関わる演算と実行に関わる演算とは別の関数に置く。
定量 （いくつ）	■ 関数 20 個まで。 (公開 10 個、非公開 10 個)	■ 50 行まで(推奨 25 行)。

[19] インクルードの数に関する制限を加えるのもよい。関数の個数制限との両立には設計が必要となる。

3.2　PBR シナリオで具体的な手順と規準とを明示する

理解が行動に結びつくことで成果になる。モジュラーアプローチ、つまりソフトウェアの５Ｓ３定でも、行動のイメージを掴んだ後は手を動かすことが求められる。このとき設計者の手が動くように、具体的な手順と規準とを示す必要がある。

手を動かすためには、まずは動かすための環境を用意する。コードレベルでソフトウェアの５Ｓを実施することは不可能ではない。しかし図 65-4 に例示したようにソフトウェア構造を UML のクラス図で表現すれば、構造を俯瞰しながら５Ｓ３定を進めることができる。コード上で変数と関数の置き場所をファイル名に明示しても、構造を俯瞰できなければ両者の置き場所の適正化は困難である。図 65-3 のような UML のモデルとコードとのマッピングルールを定めた上で、モデルを用いて置き場所を適正化するとよい。

次に考慮すべき点は、手順と規準の示し方である。実務上は、図 65-4 の左側のクラス図のように５Ｓ３定が不徹底な状態に対して、構造を改善できる手順と規準が望ましい。設計者は意図していなくとも、結果的に不適切な設計構造を作ることがある。これをレビュー等で是正できれば、５Ｓ３定の観点で良い構造を作ることができる。従って手順と規準はレビューのシナリオとして示すこととする。

ソフトウェアの５Ｓ３定を実現するためのレビューシナリオを表 65-5 に示す。これは PBR（Perspective - Based Reading）と呼ばれるレビュー技法に基づくシナリオである。PBR レビューでは視点を定めて査読する。シナリオには役割、視点、査読手順が記載されており、これらに従いレビューを実施する。何度かレビューを繰り返すことで、役割に固有の視点と査読手順が身に付く。教育効果も見込めるため、設計者の力量向上を期待できる。

なおこの PBR シナリオを有効に活用するためには、UML において変数名、関数名等を母国語で記載することが望ましい。そのため国内では分析モデル作成だけでなく、設計モデル作成の後半まで日本語で記載し、最後に実装用にアルファベットに置き換えるようにする。モデリングツールによっては別名記載ができるため、別名によって日本語とアルファベット表記とを切り替えるのもよい。

ところで、表 65-5 に示した PBR シナリオだけでは、ソフトウェアの５Ｓ３定には十分でも、モジュラーアプローチには不足する。モジュラーアプローチでは、「情報隠蔽およびカプセル化」を実施する。これを検討するためには、表 65-6 に示すコミュニケーション図の PBR シナリオも必要となる。このシナリオは関数レベルの５Ｓ３定に相当する。

この PBR シナリオを用いて、コミュニケーション図上の関数の呼び出し階層を適正化する。関数呼び出し階層が適正化できれば、その延長線上で情報隠蔽およびカプセル化の検討も容易になる。

表 65-5　PBR シナリオの例（凝集度と結合度の改善）

役割名	クラス図の凝集度と結合度に対する検証者
説明	クラス図として凝集度が低く、結合度が高い場合がある。このような場合は、クラス図を対象に、凝集度と結合度の改善を行う。本査読手順では、このクラス図の凝集度と結合度に関して検証する。
視点	クラス図に関して以下の視点から検証する。 1. クラス図の範囲に含まれない言葉は残っていませんか？ 2. クラス図内の言葉の散らばりが解消されていますか？ 3. 改善に伴う修正によって不整合は生じていませんか？
査読手順	凝集度と結合度の改善が適切に実施されているかを以下の手順で検証する。 **■手順１： クラス図内の言葉はクラス図の名前が示す範囲に収まっていますか？** ✓　記述範囲を特定できるクラス図の名前であるかを確認する。 ✓　クラス図の名前で特定された範囲内の言葉のみがクラス図にあることを確認する。 クラス図の名前は、図に含まれるクラス群の振る舞いの範囲を表現している。従って範囲外の言葉は、そのクラス図に含まれてはならない。なお範囲の表現では、動詞に着目する。そのもの固有のはたらきを動詞で表現し、それを組み合わせることで振る舞いの範囲が明確になる。名前と内容が一致しない場合は、名前を見直すか、クラス図を修正する。図の修正では、範囲外の概念を別のクラス図へ移動してもよい。 **■手順２： 同じような言葉がクラス図内で散らばっていませんか？** ✓　同義語となる言葉が複数のクラスに分散していないかを確認する。 ある言葉がクラス図内で散らばる状態であれば、その言葉の関与する変更理由によって多数のクラスが変更される可能性がある。従って言葉の散らばりを抑えることが保守性と移植性の向上につながる。同じものには同じ名前をつけることで、同義語を解消する。さらに同じカテゴリの言葉をもつ属性とメソッドは同じクラスに集める。 **■手順３： クラス図の構成要素を読み上げると、意味の通る文章になりますか？** ✓　クラスと関連を読み上げ、意味が通る文章になるかを確認する。 ✓　クラスの属性とメソッドを読み上げ、意味が通る文章になるかを確認する。 言葉の散らばりを解消するために属性やメソッドを移動する。移動に伴い属性名、メソッド名、関連のつながりも修正する。これらの修正を適正に行わないと不整合が生じる。この不整合はクラス図の要素を読み上げることで検出する。クラスと関連は、関連の両端のクラスを主語と目的語とし、「（主語）は（目的語）を～する」と読み上げる。クラスとその属性は、「○○の××」という形で読み上げる。同様にメソッドは「○○を××する」という形で読み上げる。読み上げは、不整合を検出するための有効な手段である。

表 65-6　PBR シナリオの例（メソッド呼び出しの改善）

役割名	メソッド呼び出しの検証者
説明	コミュニケーション図を用いてメソッド呼び出しを洗練する。本査読手順では、メソッド呼び出しの階層構造に関して検証する。
視点	コミュニケーション図に関して以下の視点から確認する。 1. メッセージ名から予期できないメッセージの階層化はありませんか？ 2. メッセージの粒度は揃っていますか？ 3. メッセージ階層の広がりやバランスは適切ですか？
査読手順	**■手順１： 階層化で展開したメッセージは元メッセージの範囲に入りますか？** ✓　元のメッセージ名から予期できない呼び出しがないかを確認する。 コミュニケーション図に配置されたメッセージ呼び出しの階層は、コード上の関数呼び出しの階層に一致する。従って元メッセージから予期できないメッセージ呼び出しが存在すると、親関数の名前から予期できない子関数の呼び出しになる。これは保守性を損なう要因となるため、呼び出しの構造を変更するか、あるいは適切なメッセージ名に修整する。 **■手順２： 階層化の際に展開したメッセージの粒度は揃っていますか？** ✓　粒度の合わないメッセージが展開先に含まれていないかを確認する。 メッセージ呼び出しを階層化する際に、展開したメッセージの粒度が異なると、展開した処理が必要十分であるかの検証が難しくなる。そのため粒度の合わないメッセージは、統合や分割で粒度をそろえる。 **■手順３：階層化の際に展開したメッセージ数やネストの深さは適切ですか？** ✓　展開したメッセージ数が３～４程度に収まっているかを確認する。 ✓　呼び出しネストの深さが揃っているかを確認する。 メッセージ呼び出しを階層化する際に、展開したメッセージ数が多すぎると、展開した処理が必要十分であるかの検証が難しくなる。またメッセージ呼び出しのネストの深さが揃っていないと、同じく処理が必要十分であるかの検証が難しくなる。このような場合には、メッセージ配置の見直しや、統合で呼び出し構造を調整する。 この査読手順は、構造化設計おいて、モジュール構造図に対して行うシステム形状の検証と同じ効果を持つ。一般にモジュール構造図は、左右均等で中間層が膨れたモスク形の形状が望ましいとされる。上記手順３で検証すると、構造図を記述した場合のシステム形状はモスク形に近づく。なおシステム形状の検証は、各モジュールの粒度が均一で、その名前付けが適切であることが前提になる。手順１と手順２でこの前提を整える。

コミュニケーション図を用いた情報隠蔽とカプセル化の検討について補足する。この図は、同じ相互作用図であるシーケンス図とは異なり構造も表現する。構造を表現することでソフトウェアアイテムの内部と外部とを明示でき、情報隠蔽とカプセル化を検討できる。図 65-6 にコミュニケーション図の例を示す。

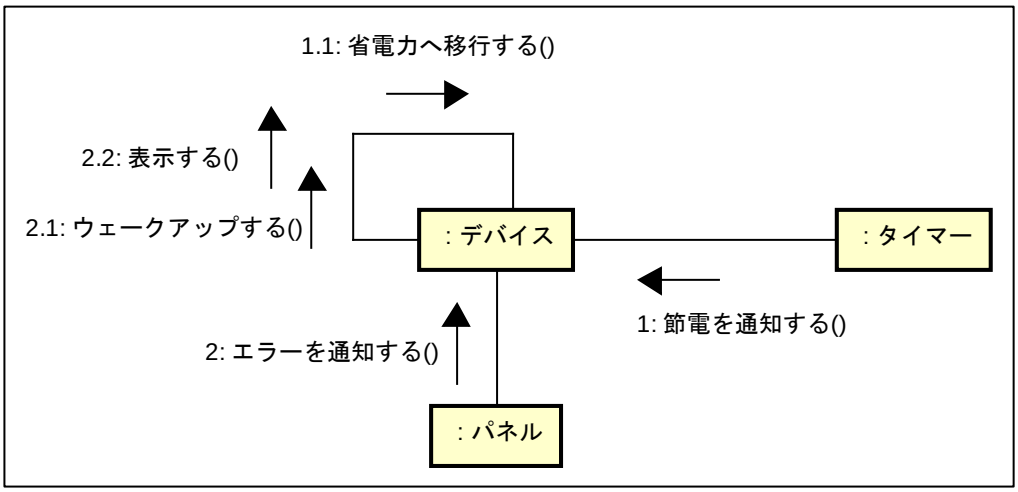

図 65-6　コミュニケーション図の例

例えば図において、「2:エラーを通知する()」は、「デバイス」というファイルの公開関数である。この公開関数が、内部処理となる「2.1:ウェークアップする()」や「2.2:表示する()」 を情報隠蔽できるか検討する。

検討では、内部事情となるデータ構造、処理手順、処理の起動条件が外側に透けていないかを検証する。透けて見えるなら情報隠蔽は実現できていない。コミュニケーション図であればこの情報隠蔽の度合いを公開と非公開の関数を比較することで検証できる[20]。

例えば、「外部に処理手順に関する知識を持ち、その手順に従って関数を次々に呼んでいる」「アクセサメソッドによって変数を参照し、外部で処理の起動条件を判断している」等がコミュニケーション図から読み取れるなら、内部事情が外部に透けている。従って情報隠蔽は実現できていない。処理手順あるいは起動条件に関する知識をもつ関数を、それらの参照先の変数がデータ構造として定義されたソフトウェアアイテムに移動すると共に、公開関数と非公開関数との区分けを適正化する。

なおモジュラーアプローチの残りの要素となる「カプセル化」や「パラメータ数制限」は、関数の引数を用いて検討できる。構造化設計の結合度の指標を用いて、引数をデータ結合あるいはスタンプ結合に制限する[21]。また同時に引数の数を最小限に抑えるとよい。

---

[20] シーケンス図では、公開関数と非公開関数の記述場所が離れることが多い。どちらかがスクロールアウトされると両者を比較しての検討は困難である。

[21] 制御結合、外部結合、共通結合の３つは、内部処理や内部データに対して外部から干渉できるため、カプセル化は達成できない。

3.3　３定管理を自動化し習慣化につなげる

ソフトウェアの５Ｓ３定のうち、 ３定管理項目についてはコード等を対象にして自動計測ができる。計測結果を定期的に設計者にフィードバックすることで５Ｓ３定を習慣化する。違反を即時にフィードバックすると改善につながりやすい。図 65-7 のような CI（継続的インテグレーション：Continuous Integration）環境を構築し、解析を自動化するとよい。

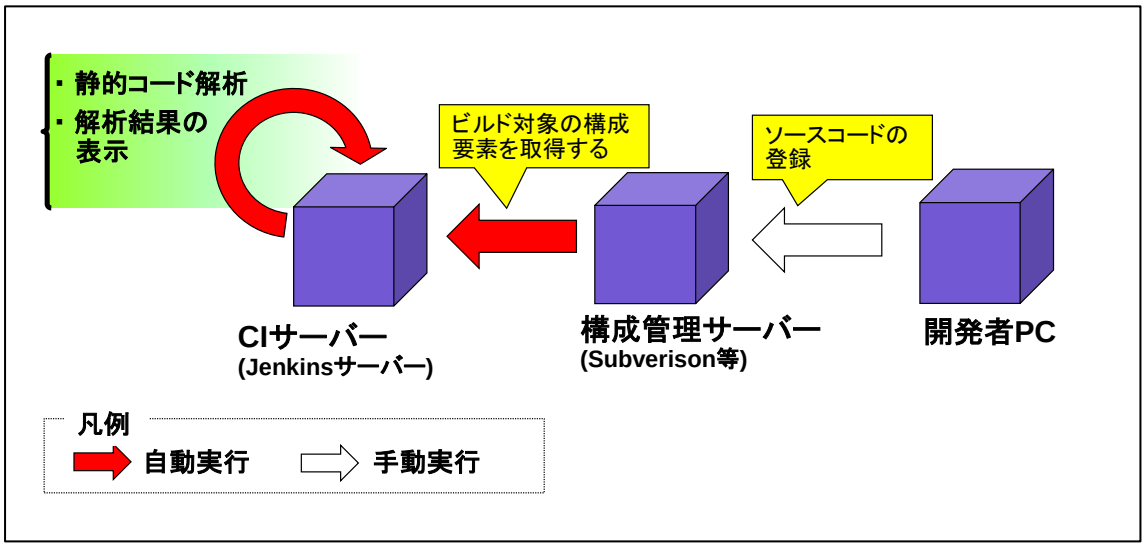

図 65-7　CI 環境の例

クラス図における関数数の ３定管理だけでは、「ソフトウェアモジュールサイズの制限」は達成できない。コード上で巨大な関数にならないような制限も必要となる。UML モデルを対象にしたレビューによる ３定管理と合わせて、コード解析による ３定管理を併用することが、モジュラーアプローチの実効性を高める有効な手段となる。

4. 準形式手法の導入

PBR シナリオの活用によるソフトウェアの５Ｓ３定活動は、準形式手法の UML モデルの活用が前提となる。しかし設計現場への UML モデルの導入は容易ではない。そのため誰でもできるモデリング手法が求められる。ある程度の精度でクラス図を作成できれば、PBR シナリオを活用し、レビューを通して５Ｓ３定の観点で良い構造に改善できる。

そこでモデリングに特化した手法として、図 65-8 のような手順と規準を作った[22]。この手法では目的語と動詞で表現した機能をクラス図に変換する。トップダウン的に進む手順と規準を規定しており、初学者にとっても実施しやすいものとなっている[7]。

このモデリング手法によって UML の導入が容易になる。また PBR シナリオを併用することでモジュラーアプローチもシームレスに実現でき、セーフティ設計およびセキュリティ設計の前提を整えることができる。

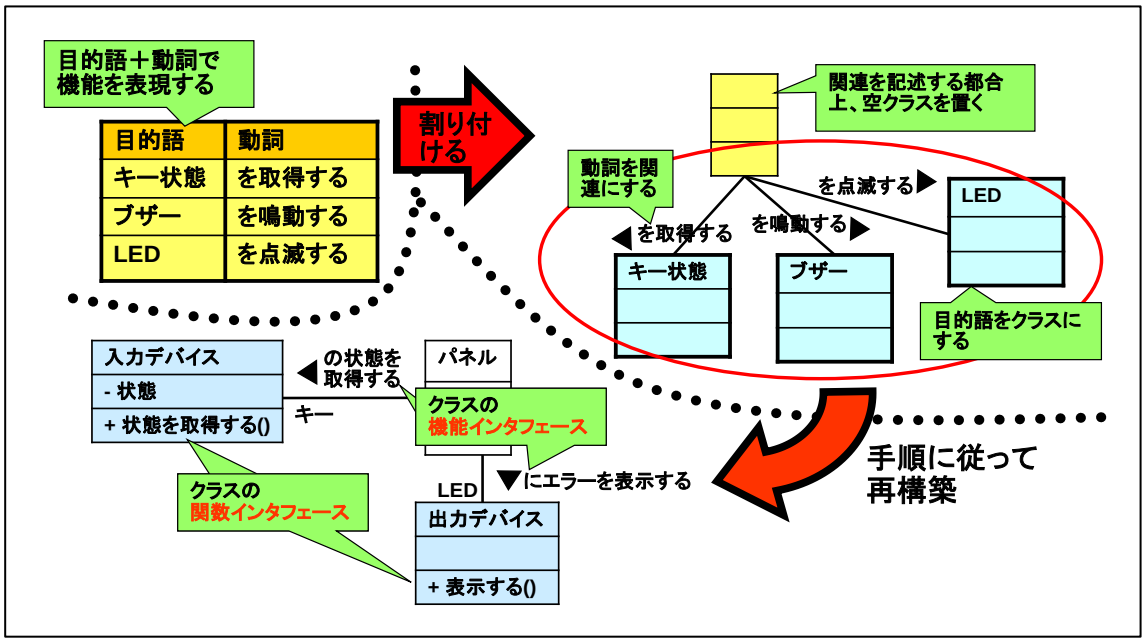

図 65-8　機能からクラス図を作成する手順

なおモジュラーアプローチの構成要素のうち、「完全に定義したインタフェース」は UML で記載することで達成する。設計モデルになると、UML の規約に従って、関数名、引数名、戻り値、それぞれの名前と型など、インタフェースの定義に必要な項目をすべて記載する。これはモデリングツールの設定項目として規定されているため、人によるブレもない。

[22] <http://ja.areyoumodeling.com/2015/06/15/verb_modeling_2/>に動画による解説がある。また解説用スライドは、<http://www.slideshare.net/ChangeVision/modeling-by-verb/>に掲載されている。

5. 導入効果と考察

モジュラーアプローチおよび準形式手法を設計現場に導入したことにで、表 65-7 に示す効果を得ることができた。なおこれらは基礎となる技術であるため、単独で導入することは少ない。通常は、各種のプロセス改善施策と同時に実施する。この場合、技術単独での定量的導入効果は示せない。以下では「小規模組み込みファームウェア」のみ他の影響をある程度排除できたため、品質および生産性の定量的な改善値を示した[23]。

表 65-7　手法の導入効果

事例	導入効果
小規模組み込みファームウェア	≪定量効果≫ ● 準形式手法（UML）の導入により、従来プロジェクト比で欠陥を 80%削減し、開発期間を 30%削減した。 ● さらにモデリング手法およびソフトウェア５Ｓ３定の導入したところ、欠陥密度は引き続き低く抑えられ、生産性がさらに 40%向上した。 ≪定性効果≫ ● モデリング作業や再構成が、システム全体の理解に大変役立った。 ● 設計がモデル化され、かつ複雑さが減ることで、属人性を減らせた。
ネットワークミドルウェア	≪定量効果≫ ● 事前に設定したソフトウェア５Ｓ３定の ３定項目については､関数サイズが 96%の順守率となった以外は、100%遵守することができた。 ≪定性効果≫ ● ３定項目を日々計測することで､規定値をオーバーした箇所について､都度リファクタリングをかけることができた。
セーフティ機器のファームウェア	≪定量効果≫ ● 事前に設定したソフトウェア５Ｓ３定の ３定項目については、100%遵守することができた。 ≪定性効果≫ ● ソフトウェアの複雑度が低下したため、単体テストおよび結合テストにおいて､高い構造的カバレッジ率となるテストの設計と実施が容易になった。

「小規模組み込みファームウェア」の事例では、品質と生産性の向上を確認できた。このプロジェクトでは品質と生産性が悪化傾向にあった。その対策として準形式手法である UML を導入した。これにより、まずは品質の悪化に歯止めをかけることができた。さらに次のプロジェクトで５Ｓ３定相当の活動を通してモジュラーアプローチも導入した。これによりソフトウェアの部品化が進み生産性が向上した。この事例は、モジュラーアプローチおよび準形式手法

[23] この事例も厳密に言うと、SPLE（Software Product Line Engineering）を意識した取り組みを実施している。従って技術の単独導入よりも、高い生産性となっている可能性がある。

が、欠陥予防だけでなく生産性向上の基礎技術になることを示している。

「ネットワークミドルウェア」の事例では、日々のコード計測を併用した。５Ｓ３定は、ルールを設定するだけでなく、守り続ける必要がある。設計者の感想には、「油断するとルールから逸脱してしまうが、日々指摘があることでルールを遵守できた」という趣旨の発言もあった。これはルール順守に関して CI 環境を用いた計測の有効性を示唆している。

「セーフティ機器のファームウェア」の事例では、モジュラーアプローチおよび準形式手法だけでなく、検証面を強化した開発プロセスを導入している。この開発プロセスでは、要求仕様と設計要素との双方向トレーサビリティ、単体テスト、結合および結合テストなど、精度の高い検証を実現するための仕組みが数多く導入されている。

これらの活動に関する労力低減には、検証面を考慮した設計がポイントとなる。このような設計を導く方法の一つが、モジュラーアプローチおよび準形式手法である。例えば、凝集度が悪く、同じ機能に関する処理がソフトウェア中に散在していれば、トレーサビリティを取る手間が増える。また関数の循環的複雑度と、単体テストの C1 カバレッジを実現するテストケース数は一致する。従って単体テストの労力軽減には、関数サイズを小さく設計するなどの対策が有効である。設計者による定性的な評価もこれを裏付けるものとなっている。

6. 今後の取り組み

本事例では、一般的な設計者が、欠陥予防のためにモジュラーアプローチを活用できるように以下の対策を取った。

① ソフトウェア５Ｓ３定で具体的な使用イメージを設計者が理解できるようにした。
② PBR レビューシナリオの形式で、誰でもできる手順と規準を明示し、モジュラーアプローチを実践できるようにした。
③ CI 環境による静的コード解析など、自動化の仕組みを併用することで、ルール違反を日々摘出し、早期に是正処置を取れるようにした。

なお上記②の PBR レビューシナリオは、準形式手法の UML の活用が前提となる。このため目的語と動詞で表現した機能をクラス図に変換するモデリング手法も合わせて作った。

これらの活動で、一般的な設計者であってもモジュラーアプローチおよび準形式手法を適切に適用でき、欠陥予防による品質向上と生産性の向上とを実現できた。

今後はこれらの応用技術となる SPLE（Software Product Line Engineering）などに幅広く広く取り組み、さらなる品質と生産性の改善に取り組んでいきたい。またアジャイル・モデリングなどへも応用し、より早くお客様に価値をお届けできるようにしたい。

参考文献

[1] JIS C0508-3:2014 (IEC 61508-3:2010) 電気・電子・プログラマブル電子安全関連系の機能安全－第 3 部：ソフトウェア要求事項
[2] JIS T 14971:2012(ISO 14971:2007) 医療機器－リスクマネジメントの医療機器への適用
[3] IEC 62366:2007 Medical devices – Application of usability engineering to medical devices
[4] JIS T 2304:2012 (IEC 62304:2006) 医療機器ソフトウェアーソフトウェアライフサイクルプロセス
[5] 一般社団法人 JASPAR：機能安全対応のための解説書【ソフトウェア・パーティショニング編】、2013
[6] ハーバート・A. サイモン（著)、稲葉 元吉、吉原 英樹（翻訳)：システムの科学 第 3 版、パーソナルメディア、1999
[7] スキルマネージメント協会 監修：モデルベース開発とエンジニア育成の最前線 「7.4 セイコーエプソンにおけるモデリング技術教育事例」、TechShare、2014

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（IPA/SEC）