2005－SE－147（13）
2005／3／18

機能量測定と連携した組み込みソフトウェア要件分析手法

鶴見 知生[i]*　小澤 陽平[ii]
岡本 鉄兵[iii]　小泉 寿男[ii]

主としてリアルタイムソフトウェアの機能量測定法として COSMIC-FFP が近年 ISO において規格化された。しかしながら、測定法の原理／原則は述べられているが、ソフトウェア工学手法と連携していないため、メトリックスの原単位として採用する場合に致命的な測定者によるバラツキが避けられない。また、要求仕様の変更は機能量にも影響を与えるため、機能量はプロジェクト開始時だけではなく、開発プロセスのマイルストーンで継続的に測定することが必要となる。このためには測定手法が開発手法とリンクしていないと測定のためのプロセスが発生することになり、継続的な測定が難しくなる。本論文はこれらの問題を解消し、機能量を原単位とするメトリックス管理を可能にするために、UML のいくつかのダイアグラムを利用した機能量測定と連携したオブジェクト指向アプローチの要件分析手法を提案するものである。

A Functional Analysis Method for Embedded Software Associated with Function Size Measurement

TSURUMI Tomoo[i], OZAWA Yohei[ii]
Okamoto Teppei[iii], Koizuni Hisao[ii]

In recent years, a method of FSM (stands for Functional Size Measurement) methods named COSMIC-FFP has been standardized in ISO for measurement of functional size, mainly of a real-time software. The definition, principles and rules of terms are stated in the method, but those are not associated with any software engineering method. Hence, there is no way to avoid scattering of measured value by measurers, and this will result in the fact that it is unavailable to apply this method for the software metrics.
In addition, the measurement of functional size should be performed not only once at the beginning of the project, but also at many times whenever a functional user requirement is changed. Therefore, if not associated with any software engineering method, the measurement of the functional size of the software to be developed imposes an additional work on developers, which makes it difficult to measure the functional size continuously in the progress of project.

1. はじめに

ソフトウェアの開発コスト、開発工数の早期見積り、ドキュメント量の予測、検証工数の予測が KKD（勘と経験と度胸）ではなく、工学的に可能になるといことで、開発対象のソフトウェアの機能量を測定が開発者側からだけではなく、発注者側からも注目され、情報システム向けのソフトウェアの分野では IFPUG（International Function Point Users Group）の FP 法と呼ばれる測定法が定着しつつある*1。しかしながら、この方法はデータベースを中心としたシステムには適用できるが、データベースが要求されない組み込みソフトウェアに適用するには問題があるということで、組み込みソフトウェア向けの機能量測定法として COSMIC 法が注目され、近年 IFPUG 法と共に ISO として規格化された。

*1 IFPUGのFP法については、JFPUG（Japan Function Point Users Group、下記URL参照）の下で日本における普及活動が行われ、COSMIC法についてもJFPUGの活動の一環として検討が行われている。
http://www.jfpug.gr.jp/

しかしながら、測定に関する概念／原理／原則は定義されているものの、どのようにして利用者機能要件を抽出す

[i] ティーティーテック AG
TTTech Compuertechnik AG
[ii] 東京電機大学大学院理工学研究科
Graduate School of Science and Engeneering, Tokyo Denki University
[iii] ペンタックス(株)
PENTAX Corporation

るかの方法論は確立していないため、メトリックスとして採用する場合に致命的な測定者によるバラツキが避けられないし、バラツキの要因がどこにあるか識別することも難しい。さらに、ソフトウェア開発手法とはリンクしていないため、仕様変更によるプロジェクトへの影響を管理するためには開発プロセスを通しての継続的な測定が必須であるが、測定が開発者に追加的な作業を強いることになり、納期に追われているような場合には現実的には難しいという問題がある。

これらの問題を解決するために、本論文では、定義された粒度の機能要件を抽出し、発注者が判読しやすい方法で機能要件の仕様を記述することによって、仕様の見落としや曖昧さを防止すると共に、分析成果物に基づいて機能量を機械的に測定できる分析手法を論じる。

さらに、この分析手法の利点は、複数開発者によって要件分析が実施された場合に、結果として得られる機能量を比較することによって要件分析結果の差異を識別できる点である。

2. FSM 連携分析手法の概要

ソフトウェア開発プロジェクトの最初のプロセスは要件分析プロセスで、このプロセスの主たる目的は開発ソフトウェアに対して要求される機能を見落としなく、正確に抽出することで、このプロセスの成果物に基づいて開発対象ソフトウェアの機能量を機械的に測定できれば、適正なプロジェクトスケジュールを立案することができるようになる。

このような観点から、本論文で機能量測定という追加的なプロセスが不要な、機能量測定と連携した要件分析手法－FSM（Functional Size Measurement）連携分析手法と呼ぶ－を提案する。

機能の抽出において重要なのは、その**粒度**と**機能仕様の判読性**である。粒度が粗いと仕様の見落とし、曖昧さが発生し、細か過ぎると利用者が理解するのが難しいという問題が発生する。また、機能仕様（シナリオなど）を自然言語だけで記述する場合には、記述レベルが問題になり、粒度の場合と同様に仕様の見落とし、曖昧さという問題が発生する。

従って、FSM 連携分析手法では、以下のように機能の粒度を定義すると共に機能仕様を表現している。

① 主アクタが人であるユースケースを機能集合と考え、このユースケースに対するステートチャート図に登場するアクションを機能と捉える。
② ソフトウェア部品というオブジェクト指向的な概念を導入し、局所的な機能をソフトウェア部品にカプセル化し、アクションとソフトウェア部品との相互作用をコラボレーション図に表現することによって、アクションの機能性を判読することが容易になる。

以上のように機能の粒度を定義し、仕様を表現することによって、組み込みソフトウェアの機能量はソフトウェア部品との間の相互作用数に依存すると仮定すれば、分析成果を COSMIC 法のソフトウェアモデルにマッピングすることができ、機能量を機械的に測定することができるということになる。

3 COSMIC 法の概要

COSMIC 法では、ソフトウェアの機能量は二つのフェーズ、すなわち①マッピングフェーズと②測定フェーズを通して測定される。以下に各フェーズの概要について説明する。なお、説明は COSMIC 法マニュアル（バージョン 2.2）に基づいて行われている。

3.1 マッピングフェーズ

マッピングフェーズでは、利用者機能要件を図 3-1 に示される COSMIC 汎用ソフトウェアモデルにマッピングする。

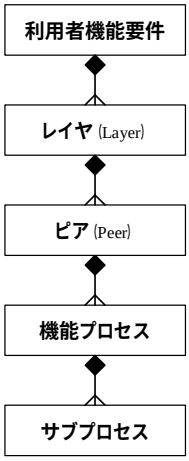

図3-1: COSMIC**汎用ソフトウェアモデル**

以下にこのソフトウェアモデルにおいて用いられている用語の定義について説明する。

① **レイヤ**は、「すべての含まれている機能プロセスが同じ抽象化レベルで実行するようにソフトウェア環境を機能的に分割した結果得られるパーティション」と定義されている。レイヤ間には上位／下位の関係があり、下位レイヤのソフトウェアは上位レイヤのソフトウェアの要求に基づいて、その機能サービスを提供する。また最上位のレイヤ（図 2-2 ではレイヤ－X）は、アプリケーションレイヤと呼ばれている。

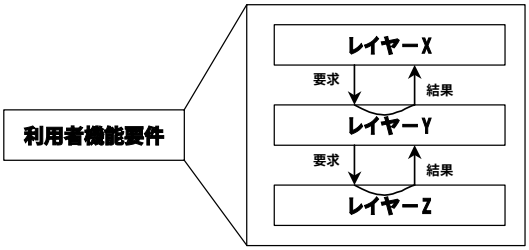

図3-2：ソフトウェアのレイヤ構成例

② **ピア**（Peer）は、「同一レイヤ内のソフトウェアを機能的に水平分割したパーティション」と定義されていて、サブシステムと同義語である。図 3-3 は、レイヤーXが三つのピアに分割されていること示している。レイヤーY、Z に関しても、境界の設定が必要な場合には、ピアに分割しなければならない。

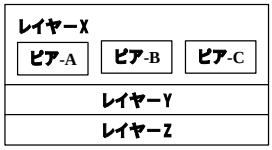

図3-3：ピア（Peer）構成例

③ COSMIC 法では、ＥＲモデルで実体の概念に対して用いられる用語“エンティティタイプ”の同義語として**注目オブジェクト**と呼ばれる用語が用いられ、この注目オブジェクトを説明する属性の集合をデータグループと呼んでいる。（エンティティタイプと異なり、過渡的なエンティティ（DBMS に対する問合せおよび問合せ結果）も注目オブジェクトとして扱われ、対応するデータグループが識別される。）

④ **サブプロセス**とは、ひとつのデータグループに属する一つまたは複数のデータ属性を移動する機能要素で、データグループ属性を移動する場所と方向に従って図 3-4 に図示されるように四つのタイプに分類される。エントリはデータグループ属性を**境界**外のユーザから境界内に移動し、エグジットはデータグループ属性を境界内から境界外のユーザに移動する。一方、リードはデータグループ属性を持続的記憶装置から機能プロセスへ移動し、ライトは機能プロセスから持続的記憶装置へ移動する。いずれのサブプロセスタイプにもデータ移動機能に加えて、データ移動に関連するデータ操作機能（サブプロセスタイプによって異なり、例えばエントリの場合には入力データの合理性チェック、エグジットの場合には出力データの計算など）が該当する）も含まれている。

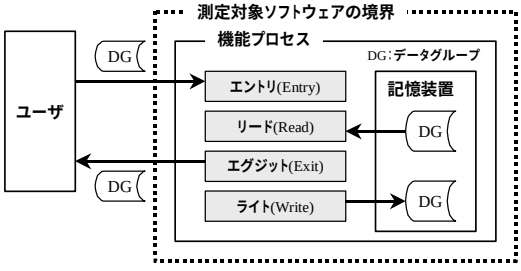

図3-4：サブプロセスタイプ

また、どのサブプロセスタイプも一つの機能プロセス（⑥参照のこと）において移動対象データグループとデータ操作はユニークでなければならない。従って、一つのサブプロセスタイプが一つの機能プロセスの中で複数回発生しても、移動対象データグループとデータ操作が同じであれば、唯一のサブプロセスタイプが識別されるだけである。

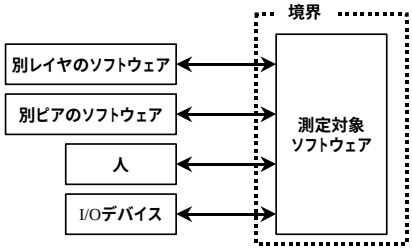

図 3-5：境界の概念

⑤ 図 3-4 において、測定対象ソフトウェアの**境界**とは機能量の測定対象のソフトウェアとこのソフトウェアと相互作用するユーザ（人、別レイヤのソフトウェア、別ピアのソフトウェア、I/O デバイスがユーザ候補である。）との間に存在する概念的な境で、境界を設定することによって測定対象ソフトウェアによって提供される機能を曖昧さなしに識別することができる。

⑥ **機能プロセス**とは、サブプロセスの集合で、機能量測定の対象ソフトウェアの境界外で発生したイベント（トリガイベントと呼ばれる）によって起動され、一連のサブプロセスを実行し、自身によって誘導されるサブプロセスが存在しなくなった時点で終了する。

【注意】機能プロセスを起動するイベントは境界外でなければならないことから、一つの機能プロセスが他の機能プロセスを起動する場合には、これらの機能プロセスは異なるレイヤまたは異なるピア内に配置しなければならない。

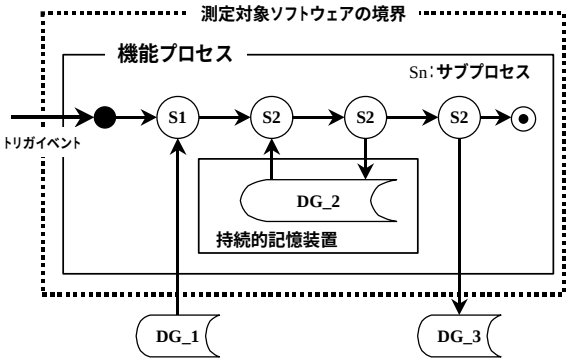

図3-6：機能プロセス

3.2 測定フェーズ

測定フェーズでは、利用者機能要件をマッピングして得られた COSMIC ソフトウェアモデルのインスタンスにおいて、ひとつのサブプロセスを機能量の単位（単位名：Cfsu）と定義することによって、測定対象ソフトウェアの機能量を測定する。すなわち、最初に機能プロセスに含まれるサブプロセスの総数を集計することによってこの機能プロセスの機能量を求め、次にこのソフトウェアの中で識別されるすべての機能プロセスについてその機能量を集計することによって測定対象ソフトウェアの機能量を求める。

例えば、図 3-6 の場合には、機能量は下表の通りで、

4 Cfsu である。

NO.	データ	移動方向	サブプロセス
1	DG1	境界外⇒	エントリ
2	DG2	持続的記憶装置⇒	リード
3	DG2	⇒持続的記憶装置	ライト
4	DG3	⇒境界外	エグジット
機能量			4 Cfsu

4. FSM 連携分析手法の詳細

COSMIC連携分析手法では、図　4-1に図示される六つのステップを通して、測定対象ソフトウェアの機能要件を分析し、最終的にコラボレーション図に基づいて測定対象

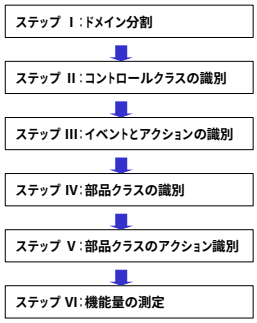

図 4-1：FSM連携分析手法の手順

ソフトウェアの機能量を機械的に求めている。
以下に各ステップについて論じる。

4.1 ドメイン分割

このステップでは、開発対象ソフトウェアの問題領域を分割するドメインの概念を導入し、機能量測定の対象となっている利用者機能要件を三つのドメイン、①アプリケーション・ドメインと②部品制御ドメイン③I/O ドメインに分割する。これらのドメインは図 4-2 に図示されるように COSMIC 法のレイヤに 1:1 に対応する。

ここで、部品制御ドメインには利用者機能要件を実装するために必要なソフトウェア部品を制御する機能（例：タイマ機能、PID／オンオフ温度調節機能、温度監視機能）、アプリケーション・ドメインにはソフトウェア部品と協調して利用者機能要件を実装する機能、I/O ドメインには外

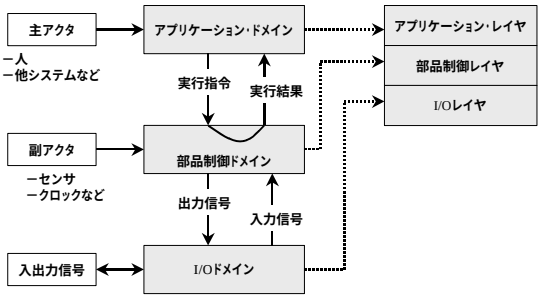

図 4-2：ドメイン分割とレイヤとの関係

部入出力信号とインタフェースする機能が含まれる。このようにドメインを識別することによって、アプリケーション・ドメインと部品制御ドメインとの間にも部品制御ドメインと I/O ドメインの間にも上位／下位の関係が存在することから、それぞれのドメインを COSMIC 法のレイヤに対応させることができる。なお、実際のシステムでは、ユーザインタフェース機能を別ドメインとすることが想定されるが、本提案手法ではこの機能は部品制御ドメインに含まれるものとしている。

機能量を測定する場合に、測定視点（発注者視点、または開発者視点）を明確にすることが重要であるが、アプリケーション・ドメインの機能は主アクタによって直接起動され、部品制御ドメインの機能は主アクタではなく、副アクタ（システム内部のアクタ）によって起動されることから、アプリケーション・ドメインの機能量は発注者視点の機能量で、アプリケーション・ドメインと部品制御ドメインの機能量合計は開発者視点の機能量とみなすことができる。

4.2 コントロールクラスの識別

このステップでは、アプリケーション・ドメインに含まれるコントロールクラスを識別する。コントロールクラスとは、主アクタによって直接的に起動される互いに依存性が低い機能要素で、主アクタによって引き起こされるユースケースと１：１に対応する。機能要素は互いに依存性が低くなければならないことから、ユーザケースも互いに疎となるように識別することが必要である。また、コントロールクラスはピアとも１：１に対応し、複数のコントロールクラスが識別される場合には、各コントロールクラスは別々のピアに配置される。

図4-2は話題沸騰ポットで識別される三つのコントロールクラス(キッチンタイマを制御するコントロールクラス、ポット内の水をモードによって決定される一定温度まで沸騰し、沸騰後に保温するコントロールクラス、およびポットの湯を給湯するコントロールクラス）を図示している。

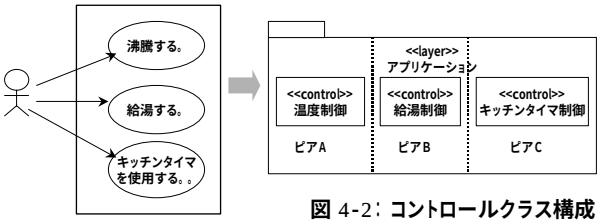

図 4-2: **コントロールクラス構成**

4.3 イベントとアクションの識別

このステップでは、コントロールクラスの時間的な挙動をステート遷移図によって定義することによって、遷移イベントとこのイベントによって起動されなければならないアクション（一連の制御アクティビティ）を識別する。ステート遷移図で定義されるイベント／アクションはCOSMIC法におけるトリガイベント／機能プロセスにそれぞれ対応する。

図4-3は話題沸騰アプリケーションのコントロールクラス「沸騰制御」のステート遷移を示し、「停止中」にイベント「沸騰開始ボタン押下」が検出されるとステートは「沸騰中」に遷移し、同時にアクション「沸騰開始」が起動されることを示している。

【注意】　COSMIC 法では、アクションを起動するイベントは境界外で発生したイベントでなければならないことから、図 4-3 で保温開始は沸騰中のアクションの中で検出されるイベントで起動することはできず、境界外で検出されるイベントで起動することが必要である。このため、沸騰開始アクションで沸騰温度到達メッセージを受けたときに起動される部品クラス「タイマ」のオブジェクト「保温開始タイマ」のタイムアップ通知が保温開始アクションのイベントになる。（図 4-5 を参照のこと）

4.4 部品クラスの識別

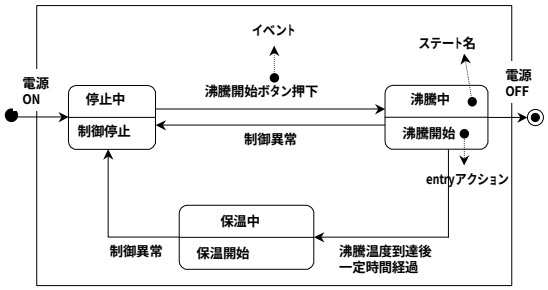

図 4-3: **ステートチャート図**

このステップでは、アクションにおいて使用される部品クラスを識別し、アクションと部品クラスのコラボレーション図を作成する。部品クラスとはハードウェアで使用される電子部品に対応するソフトウェア部品のことで、部品クラスにカプセル化される機能はアクション毎に発注者の視点でコラボレーション図を作成しながら識別され、洗練される。発注者視点からのアクションのもつ機能の判読性は、どのような部品クラスを識別するかに依存するため非常に重要なステップである。

コラボレーション図上には、

① アクションと自身が属するクラス属性の相互作用
② アクションと部品クラスの相互作用
③ アクションと入出力信号の相互作用

の３種類の相互作用が図示される。図 4-4 に図示されるように、①の相互作用には自身が属するクラスの属性の参照／更新が、③の相互作用には I/O ドメインで管理される入出力信号の入出力が含まれる。一方、②の相互作用は形態

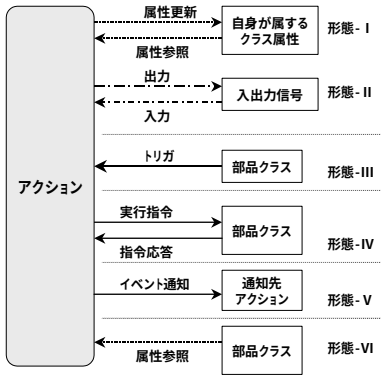

図4-4: **アクションと部品クラス間のインタフェース**

III～VIの４形態に分類される。

形態-IIIでは部品クラスからのメッセージによってアクションが起動される。

形態-IVではアクションは部品クラスに対して指定したサービス（例：検索）の実行指令のためのメッセージを送信し、部品クラスはサービスが完了したときに実行結果（例：検索結果）をアクションにメッセージで引き渡す。

形態-Ⅴでは、アクションにおいて検出されたイベントがメッセージで指定されたアクションに通知される。

形態-VIでは、アクションは部品クラスに対して属性の送

信を要求し、部品クラスの属性を参照する。ただし、簡便性のために属性の送信要求は表現しない。

コラボレーション図では、形態-ⅠとⅤのインタフェースは破線矢印で表現されるのに対して、それ以外の形態ではインタフェースはメッセージ交換によって行われ、実線矢印で表現される。

また、各部品クラスも他の部品クラスとの相互作用があると想定されることから、各部品クラスの制御ソフトウェアはそれぞれ別のピアに配置されるものとする。

図4-5はコントロールクラス「温度制御」内のアクション「沸騰開始」のコラボレーション図を示している。このコラボレーション図は、アクション「沸騰開始」は部品クラス「操作ボタン」から「開始ボタン押下」イベントによってが起動され、最初にオンオフ制御器に目標温度(固定値、例：１００℃）付きで制御開始指令を送信し、オンオフ制御器から目標温度到達メッセージを受信すると、一定時間後に保温制御を開始するために、タイマにタイムアップ時

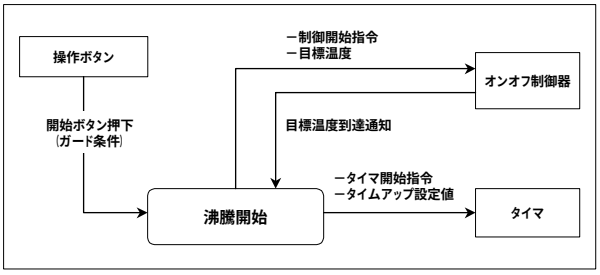

図4-5：アクション「沸騰開始」コラボレーション図

間付きでタイマ開始指令を送信することを示している。(なお、図4-3のステートチャート図は、タイマからのタイムアップイベントによって保温開始アクションが起動されることを示している。)

4.5 部品クラスのアクションの識別

このステップでは、コントロールクラスのアクションと同様に部品クラスの時間的な挙動をステート遷移図で定義し、遷移イベントとイベントによって起動されるアクションを識別し、アクション毎にコラボレーション図を作成する。部品クラスのコラボレーション図の作成においては、サービスを要求した利用ソフトウェアに対するサービス完了を通知するためにメッセージ出力が必要となる。

4.6 機能量の集計

このステップでは、ステップⅣとⅤで作成されたコラボレーション図でアクションと部品クラスとの相互作用を図4-6に図示される相互作用形態とサブプロセスタイプの対応に従って、サブプロセスタイプ別にサブプロセスの総数をカウントすることによって、当該アクションのサブプロセスタイプ別の機能量を機械的に求めることができる。

従って、図4-5のアクション「沸騰開始」の機能量は4Cfsu（エントリ＝2、エグジット=2）となる。さらにすべてのアクションの機能量を集計することによって測定対象ソフトウェアの機能量を測定することができる。

なお、アクションの時間的な流れにおいて一つの部品クラスとの相互作用が複数回発生するような場合には、コラボレーション図に加えてシーケンス図で相互作用の流れを定義することが必要となる。例えば、最初に部品クラス属性を参照したときに実行されるデータ操作と次に同じ部品クラス属性を参照したときに実行されるデータ操作が異なるような場合である。理由は、COSMIC 法ではデータ移動タイプを識別することによって機能量を測定するのであって、発生回数を識別するのではないからである。

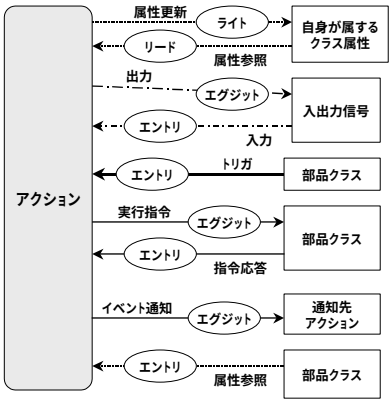

図4-6：相互作用形態とサブプロセスタイプの対

5 事例と評価

5.1 事例

FSM連携分析手法をウェブで公開されているCOSMIC法のケーススタディ「炊飯器」に適用した場合について考察する。

このケーススタディの利用者機能要件は図5-1に示されている通りで、以下にこの利用者機能要件で要求されている

(1) 3種類のモード(早炊き、普通炊き、粥炊き)で炊飯できること。
(2) モードを選択し、次に開始ボタンを押して炊飯を開始する。ただし、モードを選択しないで、開始ボタンを押した時には普通モードで炊飯を開始する。
(3) 30秒周期で、選択モードおよび経過時間に基づき目標炊飯温度を更新する。
(4) 5秒周期でヒータを制御する。すなわち、炊飯温度＜目標炊飯温度の場合にはヒータをオン、そうでない場合にはオフとする。
(5) 30秒周期で炊飯器状態(炊飯、保温)を判定し、対対応する状態ランプを点灯する。開始ボタンが押されてから一定時間経過するまでは“炊飯状態”、それ以降は“保温状態”とする。一定時間はモード毎に決められている。

図 5-1：ケーススタディ「炊飯器」の利用者機能要件

機能要件をFSM連携要件分析手法のステップに従って分析し、その機能量を測定する。

ステップⅠ：ドメインの分割

図3-1に図示されるようにアプリケーション・ドメイン、部品制御ドメイン、I/Oドメインの三つのドメインに分割することができる。

ステップⅡ：コントロールクラスの識別

人を主アクタとするユースケースは図5-1に、このユースケースに対応するコントロールクラスのステートチャート図は図5-2に図示する通りである。

図 5-1：炊飯器のユースケース図

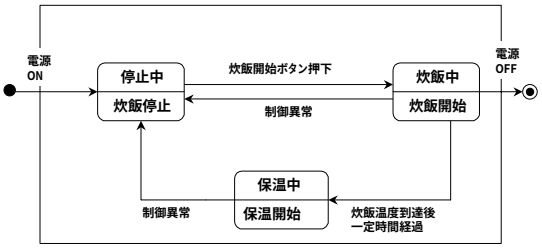

図 5-2：コントロールクラス「温度制御」のステートチャート図

ステップⅢ：イベントとアクションの識別

図5-2から識別されるアクションは、①炊飯開始、②保温開始、③制御停止の三つである。

ステップⅣ：部品クラスの識別

上記の三つのアクションにおいて使用される部品クラスとこれらの部品クラスとのコラボレーション図は図5-3の通

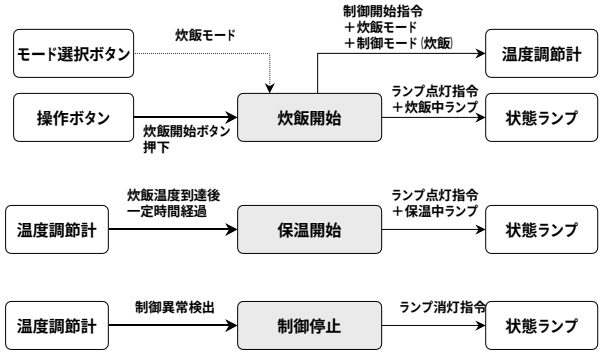

図 5-3：アクション「制御停止」のコラボレーション図

りである。

ステップⅤ：部品クラスのアクション識別

部品クラスについても、コントロールクラスと同様にステート遷移図からアクションを識別し、コラボレーション図を作成することによって部品クラス毎の機能量を測定することができる。

図5-4は、炊飯開始アクションにおいて使用される部品クラス「温度調節計」のステートチャート図と識別される二つのアクションのコラボレーション図を示している。

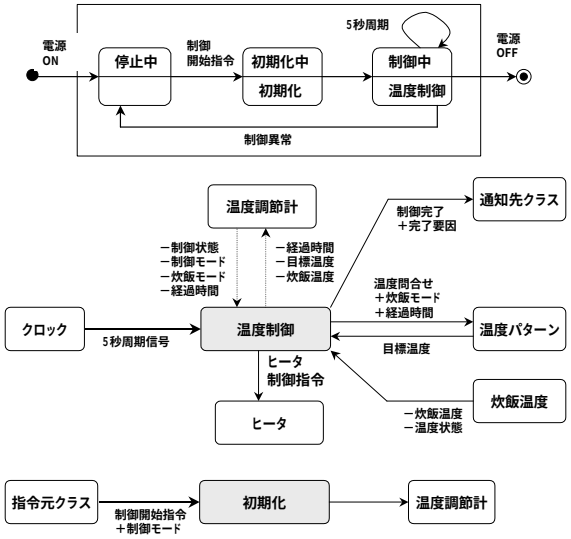

図 5-4 部品クラス「温度調節計」の ステートチャート図とコラボレーション図

ステップⅥ：機能量の測定

図5-3と図5-4のコラボレーション図からアクションと部品クラス間の相互作用数をカウントすればコントロールクラスと部品クラス「温度調節計」のアクションの機能量が測定できる。結果は表5-1に記載の通りである。なお、表5-1には、温度調節計以外の部品クラスに関して、識別されるアクションとそのコラボレーション図から測定されたアクションの機能量も記載されている。

5.2 評価

表5-1において、増分はケーススタディの測定結果と比較したときの機能量の増加分を、また理由欄には機能量が増加した理由を示している。

理由1： 目標炊飯温度および保温温度は炊飯モードと炊飯経過時間に基づいて決定されるが、ケーススタディでは機能量として計上されないデータ移動に伴うデータ操作として扱っているため。（＋5）

理由２：ケーススタディでは考慮されていないため。（＋2）

理由３：部品クラス導入のため。（＋10）

理由1の増分は、温度パターンの保守が必要と考えたためのもので、理由2の増分も炊飯器に何らかの異常が発生すれば異常処理が必要であるとの考え方に因るもので、測定法に依存するものではなく、機能要件の捉えかたの問題で

ある。しかしながら、部品クラスという概念の導入に

NO.	クラス	クラスタイプ	アクション	機能量	増分	理由
1	炊飯制御	コントロール	炊飯開始	4	-	
2			保温開始	2	-	
3			制御停止	2	+2	2
3	温度調節制御	部品	初期化	2	+2	3
4			温度制御	8	+2	1
5	温度パターン管理	部品	温度問合せ	3	+3	1
6	モード管理	部品	モード管理	3	+3	3
7	状態ランプ制御	部品	状態ランプ制御	2	+2	3
8	温度監視	部品	温度監視	3	+3	3
合計				29	17	

表5-1： 機能量測定結果

伴って発生した理由3の増分は、システムの内部的なインタフェースが増加したことによるもので、発注者視点からの機能量ではなく、開発者視点の機能量であるとも考えられる。

従って、ソフトウェア部品の再利用性に考慮を払う必要がなく、インタフェースのための機能量を排除したい場合には、部品クラスの機能を利用しているアクションに移植することで可能であるが、以下の理由によって部品クラスの概念を導入すべきであると考える。

① ユーザ視点の機能量と開発者視点の機能量とを明確に区別することができる。すなわち、コントロールクラスのアクションの機能量は明らかにユーザ視点の機能量であり、この機能量と部品クラスの機能量の合計は基本的に開発者視点の機能量とみなすことができる。場合によっては、部品クラスの機能量も発注者視点の機能量と捉えられる場合がある。

② 生産性につながる再利用性の向上。

③ 部品クラスへの機能のカプセル化による開発ソフトウェアによって提供される機能の判読性の向上

6 まとめ

本論文では、オブジェクト指向的アプローチとUMLのダイアグラムを使用した機能量測定と連携した要件分析手法を提案しケーススタディ「炊飯器」に適用して評価を行った結果、FSM連携要件分析手法は、開発プロジェクトの先頭プロセスである要件分析プロセスにおいて発注者の機能要件を見落としなく、かつ正確に抽出することができることに加えて、ISOとして標準化されている機能量測定法に準拠して機能量を計測することができるということを確認できた。

今後の課題は、以下の二点であると考えている。

① 互いに独立したユースケース抽出手法の検討

② 要件分析の成果物の設計プロセスへのシームレス・インタフェース

③ 測定値バラツキ要因の解析

参考文献

(1) 渡辺 博之、渡辺 政彦、堀松 和人、渡守 武和：「組込み UML eUML によるオブジェクト指向組込みシステム開発」，翔泳社, (2002)

(2) 渡辺 政彦、飯田周作、石田 哲史、山本 修二、浅利康二：「UML 動的モデルによる組込み開発」，オーム社, (2003)

(3) 細川 卓誠･鶴見 知生･岡本 鉄兵･小泉 寿男：「組込みソフトウェア開発におけるユースケース分析･設計方式の提案」，情報処理学会第 140 回ソフトウェア工学研究会, 140-2 pp9-14 (2003)

(4) 組込みソフトウェア技術者・管理者育成研究会 SESSAME「話題沸騰ポットGOMA GOMA-1015 型」」 sessam@blues.tqm.t.u-tokyo.ac.jp

(5) COSMIC 法計測マニュアル(V2.2) http://www.lrgl.uqam.ca/cosmic-ffp/manual.jsp

(6) COSMIC 法ケーススタディ「炊飯器 Rice Cooker」 http://www.lrgl.uqam.ca/cosmic-ffp/casestudies/