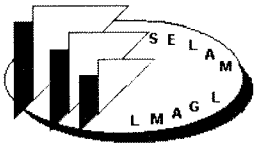
Université du Québec à Montréal

ケベック大学（モントリオール）




# フルファンクションポイント：算出マニュアル

**FULL FUNCTION POINTS:**
**COUNTING PRACTICE MANUAL**

## 技術報告書1997-04

編集

ソフトウエア経営工学研究所
Software Engineering Management Research Laboratory
ソフトウエア応用計測研究所
Software Engineering Laboratory in Applied Metrics (SELAM)

共同研究

Nortel
Bell Canada
Hydro-Québec
JECS System Research

執筆者

Denis St-Pierre
Marcela Maya
Alain Abran
Jean-Marc Desharnais
Pierre Bourque

1997年9月

株式会社ジェックスシステムリサーチ　荒木　盛次　訳（1998年1月）

# 目 次

[1] FFP算出手順とルールに関する文献は次の技術レポートでも公開されている。

- http://lmagl.qc.ca/rapport16.pdf または
- http://saturne.info.uqam.ca/Labo_Recherche/Lrgl/publi/treports/LRGL-1997-018A.pdf

# 1. はじめに

最近、製品やサービスに対するソフトウエアの重要性が高まり、法人や団体などの予算に占めるソフトウエアの比率も高くなりつつある。従って、ソフトウエア費用も、他の費用と同様にもっとも進んだ投資効率事例と比較検討して、管理することが大切である。そのためには、ソフトウエアの計測とそれを用いた見積もりモデルの確立が必要となる。これらの計測は、たとえばソフトウエアの開発、メンテナンス時の品質や生産性の解析に不可欠である。

開発者の視点から製品やサービスの実施効率を把握するために、技術面の計測が必要となる。たとえば技術面の計測は、開発段階の設計効率の悪さを改善するのに役に立つ。またユーザの視点から製品やサービスの品質や生産性の高さを知るために、機能性の計測が必要となる。機能性の計測は、開発の際の技術的な意思決定や実現の際の意思決定と無関係でなければならない。これらの計測は、個別の開発方法や技術の生産性を比較する上で役に立つ。

「ファンクションポイント法 (<FP 法> : Function Point Analysis)」は、このような機能性の計測技法のひとつである。主にFP法 はMIS領域でよく使われ、大変役に立っている。特にソフトウエアの生産性分析と見積もりによく使われている（Abran,1996； Desharnais,1988, Jones,1996 ; Kemerer,1987）。 FP法は、MISソフトウエアに特有の機能特性を把握するのに適している。

しかしながら、FP法 は、必ずしもどんなタイプのソフトウエアにもむくわけではないという批判を受けてきた（Conte,1996 ; Galea,1995 ; Grady,1992 ; Hetzel,1993 ; Ince,1991 ; Jones,1988 ; Jones,1991 ; Kan,1993 ; Whitmire,1992）。次に、ファンクションポイント<Function Point>（FP）の適用範囲に関する D.C.Inceの見解を引用する。

> *”ファンクションポイント アプローチ <Function Point Approach> の問題点は、そこで想定されるアプリケーションのタイプが、特に銀行や建設、販売など大型ファイル ベースのシステムに限られていることだ。通信を多用した在庫管理システムなどのハイブリッド システムには当てはまらない” (Ince、1991)。*

FP法はリアルタイム ソフトウエアの機能特性を完全にはとらえていない。勿論、FP法 をリアルタイム ソフトウエアに適用し、算出結果を得ることはできる。しかし結果は適切なサイズを表さない。現在の FP法では、リアルタイム ソフトウエアに対しては、MIS領域とは計測詳細内容が異なり、同程度の意味ある結果は得られない。すなわち、FP法では、リアルタイム ソフトウエアのフルサイズ

の機能性をもとに、意味ある生産性ベンチマークや見積もりモデルを確立するのは難しい。

この報告書は、Software Engineering Management Research Laboratory と産業界の共同研究パートナである SELAM が、FP法をリアルタイム ソフトウエアに適用した試みを記述したものである。第２章では、International Function Point Users Group （IFPUG） の算出マニュアル 第４版をもとにFP法について概説する。第３章では、リアルタイム ソフトウエアの定義について述べる。第４章では,現在のFP法をリアルタイム ソフトウエアに適用した場合の限界について述べる。第５章では、われわれが提案する拡張コンセプトについて紹介する。第6章では新しい算出の手順と規則について説明する。次の第7章では、算出の事例について説明する。第8章ではこの拡張手法をアプリケーション計測に実際に適用した際に確認できた事柄について述べる。

# 2. FP法 概要

ファンクションポイント法(Function Point Analysis) は、1979年に、FP法の開発者である IBM の Allan Albrecht がはじめて発表した。また1984年に、International Function Point Users Group が設立され、FP法の規則の明確化、標準化および普及発展を進めてきた。FP法の規則と標準はこれまでに4回大幅に改定された。ここで説明する FP法の規則と標準は1994年改定の最新版にもとづいている。

FP法の計算の第１ステップは算出する対象の 境界<boudary> を区分けすることである。すなわち、この境界というのは、計測対象のアプリケーションまたはプロジェクトと、外部のアプリケーションまたはユーザの領域を分ける境界線である。これにより、ファンクションポイント 算出(Function Point Count)のどの機能が、境界内にどれだけあるかがわかる。図1はアプリケーションとユーザの間の境界を示している。次のステップは、*未調整ファンクションポイント数 <unadjusted function point count> (UFP数)* の算出である。

UFP 数は、プロジェクトまたはアプリケーションがユーザに提供する特有の機能性を反映しており、これをポイントとして算出する。 UFPの計算は、プロジェクトまたはアプリケーションの５つの ファンクションタイプ<function types> の計測からはじめる。５つの ファンクションタイプとは、２つの データ ファンクションタイプ<data function types> と３つの トランザクション ファンクションタイプである。次にこれらの５つの ファンクションタイプについて説明する（図１）。

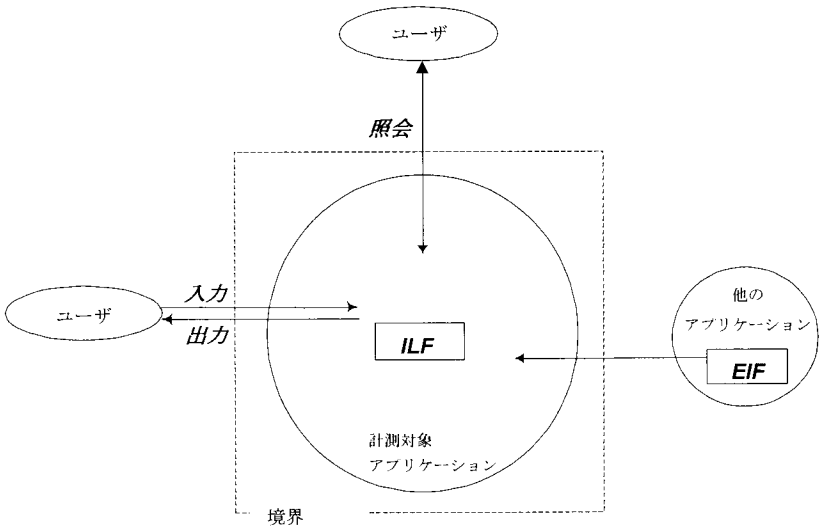



**図1 - FP法 components**

データ ファンクションタイプ：

内部論理ファイル<Internal Logical File> (ILF)：算出するアプリケーション境界内で維持管理される、一連の論理的に関連するデータまたは制御情報のグループで、ユーザが認識可能なものである。

外部インターフェース ファイル<External Interface File> (EIF)： 算出するアプリケーションによって参照される、一連の論理的に関連するデータまたは制御情報のグループで、ユーザが認識可能なものである。それは算出アプリケーションの境界の外で維持管理される。つまり、EIF は他のアプリケーションで ILF として算出できるものでなければならない。

トランザクション ファンクションタイプ<Transactional function types>：

外部入力 (EI) ：アプリケーションの境界の外から入ってくるデータまたは制御情報を処理する。外部入力それ自体はひとつの要素処理<elementary process>[2]である。

外部出力 (EO) ：アプリケーションの境界の外に送り出すデータまたは制御情報を生成する。外部出力自体はひとつの要素処理である。

外部照会 (EQ) ：入力（検索要求）と出力（検索結果）の組みあわせである。外部照会は情報を検索するための要素処理である。出力側には、導出デ

[2] 要素処理(Elementary process): 業務上、エンドユーザにとって意味のある活動の最小単位。この要素処理は、自己完結していなければならず、算出するアプリケーションの業務を首尾一貫した状態におかなければならない。

ータ<Derived data>[3] は含まない。この処理内では、ILF の維持管理はしない。

次に、５つの ファンクションタイプの複雑度を「低」、「中」、「高」にランクづけする。その後、ファンクションタイプごとに重みづけを行い UFP を算出する。

最後に、プロジェクトまたはアプリケーションの環境や処理の難易度をもとに、値調整係数<value adjustment factor> を使い UFP に調整を加えて、最終的な ファンクションポイント<Adjusted function points> 数を計算する。値調整係数は14の項目からなる一般システム特性<general system characteristics> を使って求める。14の項目は影響度が０から５までの６段階に格付けされている。各格付けは、プロジェクトまたはアプリケーション特有の言葉で表現されている。この係数は UFP に対しプラス、マイナス最大35%の範囲で調整を加え、*調整済みファンクションポイント*数を算出する。

FP法 の定義、手順、算出規則、事例についての詳細は、IFPUG が発行した Function Point Counting Practices Manual, Release 4.0 を参照願いたい。

# 3. リアルタイム ソフトウエア

「リアルタイム ソフトウエア」に関する定義は多くの文献に見られる。

IEEE (1990):

*「外部のプロセスをタイムリーに制御し、モニタし、応答するために、外部のプロセス処理時間内に、システム内のコンピュータ処理が終了するシステムである。」*

The Oxford Dictionary of Computing (Illingworth 1991):

*「出力が生成されるまでの時間が極めて重要なあらゆるシステムである。通常、この種のシステムの入力は、物理的な動きに呼応し、同様に出力も物理的な動きを必要とするからである。入力から出力までのタイムラグは、許容時間に対して十分小さくなければならない。」*

[3] 導出データ(Derived data):ILFとEIFからの情報をそのまま取り出し編集したデータではなく、なんらかの処理が施されたデータ。

Laplante (1993) and Cooling (1991):

*「コンピュータの応答が、時間的制約をこえると性能が劣化し、また機能不全となるシステム。」*

Stankovic and Ramamritham (1988):

*「リアルタイム システムは、システムの正確さが、計算の正確さだけでなく、結果を出すまでの時間によっても測られるシステムと定義される。」*

最後の定義は、リアルタイム ソフトウエアに関する２つの重要な側面を示している。タイミングおよび外部エンティティとの相互作用である。この相互作用の目的は、外部エンティティから情報を受け取り、外部エンティティを制御することである。この相互作用は厳しい時間的制約の中で仕事を選択し完了しなければならない。リアルタイム ソフトウエアは、タイミングが厳しい制約をうけるという点で、他のシステムとは異なる。この種のソフトウエアは、このような厳しい制約の下での処理を余儀なくされている。このような時間的制約を守らなければ、システム不良が発生する。結局、リアルタイム ソフトウエアの結果の正確さは、タイミングに帰着する。

# 4. リアルタイム ソフトウエアでのFP法の限界

## 4.1 データ の限界

制御データ構造は、マルチオカレンス データグループ<multiple occurrence groups of data> と シングルオカレンス データグループ<single occurrence data groups> の２種類である。マルチオカレンス データグループは同一のレコードタイプに複数のインスタンスを持つことができる[4]。一方、シングルオカレンス データグループは、レコードタイプにインスタンスを1つしか持つことができない。典型的なリアルタイム ソフトウエアは、このようなシングルオカレンス データグループを多数含んでいる。このデータグループは、ILF や EIF のグループとしては扱いにくい。シングルオカレンスデータグループを適切に計測するためには、ILFやEIFのルールを拡張する必要がある。

[4] 例えば、エンジン制御アプリケーションは、各シリンダーに関する情報（シリンダー番号、点火タイミング、気圧などの制御データを持っている。このようなデータグループは、複数のマルチ オカレンス＜multiple occurrence＞構造（各シリンダーにひとつのレコードを）持っている。つまり、このシリンダーのレコードは何度も繰り返して使用される。

## 4.2 トランザクション の限界

通常のリアルタイム ソフトウエアのプロセスは、サブプロセスの数が大幅に異なる特有の トランザクション特性<transactional characteristic> を持っている。たとえばあるプロセスにはサブプロセスが数個しかないのに対して、別のプロセスには膨大な数のサブプロセスがある。リアルタイムの機能計測法の場合、この点を考慮しなければならない。次にこのような現象を、次の２つの事例を用いて説明する。

### 例 1 - エンジン温度制御プロセス　（サブプロセス数の少ないプロセス）

このプロセスの主な目的は、必要な場合に、エンジン冷却システムを作動させることである。センサーはエンジンの温度をアプリケーションに送り（サブプロセス１）、過熱温度敷居値と比較し（サブプロセス２）、必要な場合に、作動オン メッセージを冷却システムに送り出す（サブプロセス３）。

この例の場合、温度制御プロセスの*サブプロセスの数は３つである*（表1）。なおこのプロセスはアプリケーションではなく、数あるエンジン制御アプリケーションのひとつにすぎない。このアプリケーションは、エンジン温度制御プロセスのすべてのサブプロセスが完了しない限り、処理が終了したことにはならない。従って、この場合、要素処理がひとつあるだけである（IFPUG、1994）。

| プロセス | サブプロセス | サブプロセス数 |
|---|---|---|
| エンジン温度制御 | 温度入力 | 1 |
| | 比較のための制御値読みとり | 1 |
| | 作動オンメッセージの送り出し | 1 |
| | 計 | **3** |

**表 1 - エンジン温度制御プロセスのサブプロセス**

この場合、標準FP法に従えば、要素処理がひとつしかないので、トランザクション ファンクション<transactional function> がひとつ識別されるだけである。

### 例 2 - エンジン診断プロセス(サブプロセスの数の多いプロセス)

このプロセスの主な目的は、必要に応じて警告を発することである。15種類の異なるエンジン センサーが、診断プロセスにデータを送り、各センサーから受取ったデータを外部のファイルから読みとった敷居値と比較する（センサーごとにユニークなサブプロセスが1個づつ加わるので、サブプロセスは

15個になる）。条件が揃えば、ダッシュ ボードの警告器を作動させる警告メッセージ が送り出される（ひとつのサブプロセス）。

この例の場合、エンジン診断プロセスのサブプロセスは31個である（表2）。このプロセスはアプリケーションではなく、数あるエンジン制御システムのプロセスのひとつである。診断プロセスがすべてのサブプロセスを完了しない限り、アプリケーションは処理を終了したことにはならない。

| プロセス | サブプロセス | サブプロセス数 |
|---|---|---|
| エンジン診断 | センサ データ入力 | 15 |
| | 比較のための敷居値読みとり | 15 |
| | 警告メッセージの出力 | 1 |
| | 計 | **31** |

表 2 – 例2のサブプロセス

FP法 標準に従えば、トランザクション ファンクションはサブプロセスベースではなく、要素処理をもとにしているため、数個のトランザクション ポイント<transactional points> が算出されるだけである。従って、現在のIFPUGの算出手順に従えば、例１と２のポイント数には、大きな違いはない。

# 5. リアルタイムへの拡張

フルファンクションポイント <Full Function Point=FFP> は標準 FP法 をもとにした機能計測技法である。FFPはMISやリアルタイム ソフトウエアの双方に使えるよう設計されている。またFFPは、標準FP法を拡張したものであるため、IFPUGの規則がすべてこの新しい計測技法に含まれている。これまでIFPUG規則の中で扱われてきた一部の制御機能は、FFPの新コンセプトの中ではかなり大きく拡張されている。リアルタイムソフトウエアの制御面の機能は新しいファンクションタイプによって計測される。

## 5.1 FFP ファンクションタイプ

サブプロセスの数が大幅に異なるソフトウエアの機能特性を正しく計測するには、 FP法（の要素処理ベース）で規定したプロセスだけでなく、サブプロセスも考慮する必要がある。さらに、典型的な制御データに固有の構造も考慮しなければならない。 FFP はこれらの点を考慮し、新しい データ や トランザクション ファンクションタイプ を追加した。

新しい２つの制御データファンクションタイプはIFPUGのデータファンクションタイプと似た構造になっている。

アップデーテッド コントロールグループ<Updated control group>（UCG）：計測対象アプリケーションが更新する制御データグループである。それは機能的視点より認識可能なものである[5]。その制御データは複数のトランザクションのために存在する[6]。

リードオンリー コントロールグループ<Read-only control group>（RCG）：計測対象アプリケーションが参照はするが、更新はしない制御データグループである。機能的視点より認識可能なものである。その制御データは複数のトランザクションのために存在する[7]。

新しい４つの制御トランザクション ファンクションタイプはリアルタイム ソフトウエアのサブプロセスの機能を表している。

エクスターナル コントロールエントリ<External control entry>（ECE）：ユニークなサブプロセスであり、機能的視点より認識可能なものである[8]。ECEは境界の外からの入力制御データを処理する。なお、それはデータグループの処理プロセスを分解した最小レベルである。従って、制御プロセス が２つの異なるデータグループを受け取るのであれば、ECE は少なくとも２つ存在する。ECEはデータの更新機能を持たない。更新機能は別の制御ファンクションタイプで扱われる（インターナル コントロールライト）。

エンジン診断の例の場合（4.2章）、15のセンサーがアプリケーションにデータを送り、各センサーごとにユニークな15のサブプロセスがあるため、ECE は15個となる。

エクスターナル コントロールエグジット<External control exit>（ECX）：ユニークなサブプロセスであり、機能的視点より認識可能なものである。ECX は境界の外への出力を処理する。なお、それはデータグループを処理するプロセスを分解した最小レベルである。従って、制御プロセス

---

[5] つまり、アプリケーションの要求仕様が完全でありさえすれば、データグループが確認できる。

[6] ４章の事例２の場合、センサーからの入力データはひとつのトランザクションのために存在している。診断プロセス処理が終了したあとは、システムはデータを記憶しておく必要はない。一方、敷居値は、新しい入力があるたびに利用される。従って、敷居値は一回だけでなく、複数のトランザクションのために存在する。他の例については、７章を参照。

[7] FFPはリアルタイム ソフトウエア特有のデータを計測する必要があるので、UCGおよびRCGルールはFP法のILFおよびEIFルールとは異なっている。ILFおよびEIFルールは、MISの典型的なデータ構造に対して適用されると思われる。

[8] つまり、要求仕様が完全でありさえすれば、このサブプロセスは確認できる。

が２つの異なるデータグループを送り出すのであれば、ECX は少なくとも２つ存在する。ECX はデータの読みとり機能を持たない。読みとり機能は別の制御ファンクションタイプで扱われる（インターナル コントロールリード）。
エンジン診断の例の場合、ダッシュボードにメッセージ（境界の外部に送り出される制御情報）を送り出すサブプロセスが ECX である。

インターナル コントロールリード<Internal control read> (ICR)：ユニークなサブプロセスであり、機能的視点より認識可能なものである。ICR は制御データを読み込む。なお、それはデータグループ を処理するプロセスを分解した最小レベルである。従って、制御プロセスが２つの異なるデータグループを読む場合、ICR は少なくとも２つ存在する。

エンジン診断例の場合、敷居値を読むサブプロセスが ICR である。この例の場合、15の異なるサブプロセスが、異なる敷居値を異なる時間に読み、比較を行う。従ってこの場合、15の ICR が存在する。

インターナル コントロールライト<Internal control write> (ICW) ： ユニークなサブプロセスであり、機能的視点より認識可能なものである。ICW は制御データを書き込む。なお、それはデータグループ を処理するプロセスを分解した最小レベルである。従って、ひとつの制御プロセスが２つのデータグループを書き込むのであれば、ICW は少なくとも２つ存在する。

エンジン診断例での書き込みサブプロセス例は次のとおりである。「15セットの制御データがある。これらはすべて異なる時間に別々のファイルにストアされる（15の異なったサブプロセス）」。15種類のセンサ制御データが異なる時間に更新されるので(15のユニークなサブプロセス）、新しいFFP計測法に従えば、15の ICW が存在する。

次に、制御トランザクションファンクションと制御プロセスの関係を示す。

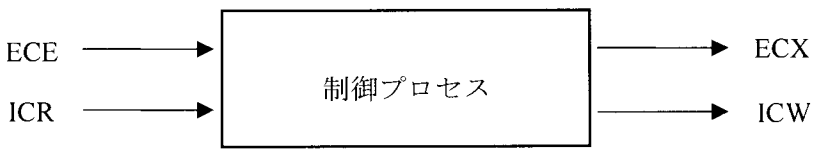



**図2 - 制御トランザクションファンクション図**

FP法 の場合と同じく、新しいファンクションタイプはすべてアプリケーションの技術的視点ではなく、機能的視点にもとづいている。従って、標準 FP法と新しい拡張法の異なる点は、新たにファンクションタイプ（UCG, RCG, ECE,

ECX，ICWおよびICR）が追加されることである。この追加されたファンクションタイプは*制御データと制御プロセスを計測*するだけである。ここで、管理データや管理プロセスと称する他のタイプのデータやプロセスは、標準FP法を用いて算出する（図３および表３参照）。

| | |
|---|---|
| FFP 管理ファンクションタイプ: | |
| 内部論理ファイル (ILF) | FFPでもFP法と変らず |
| 外部インタフェース ファイル (EIF) | FFPでもFP法と変らず |
| 外部入力 (EI) | FFPでもFP法と変らず |
| 外部出力 (EO) | FFPでもFP法と変らず |
| 外部照会 (EQ) | FFPでもFP法と変らず |
| FFP 制御ファンクションタイプ: | |
| ｱｯﾌﾟﾃﾞｰﾃｯﾄﾞ ｺﾝﾄﾛｰﾙｸﾞﾙｰﾌﾟ(UCG) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、ILF に類似 |
| ﾘｰﾄﾞｵﾝﾘｰ ｺﾝﾄﾛｰﾙｸﾞﾙｰﾌﾟ(RCG) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、EIF に類似 |
| ｴｸｽﾀｰﾅﾙ ｺﾝﾄﾛｰﾙｴﾝﾄﾘ(ECE) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、EIの 一部分に類似 |
| ｴｸｽﾀｰﾅﾙ ｺﾝﾄﾛｰﾙｴｸﾞｼﾞｯﾄ(ECX) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、EO/EQ の一部分に類似 |
| ｲﾝﾀｰﾅﾙ ｺﾝﾄﾛｰﾙﾘｰﾄﾞ(ICR) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、EI/EO/EQの一部分に類似 |
| ｲﾝﾀｰﾅﾙ ｺﾝﾄﾛｰﾙﾗｲﾄ(ICW) | 新ﾌｧﾝｸｼｮﾝﾀｲﾌﾟ、EI の一部分に類似 |

**表3 -ファンクションタイプ 表**

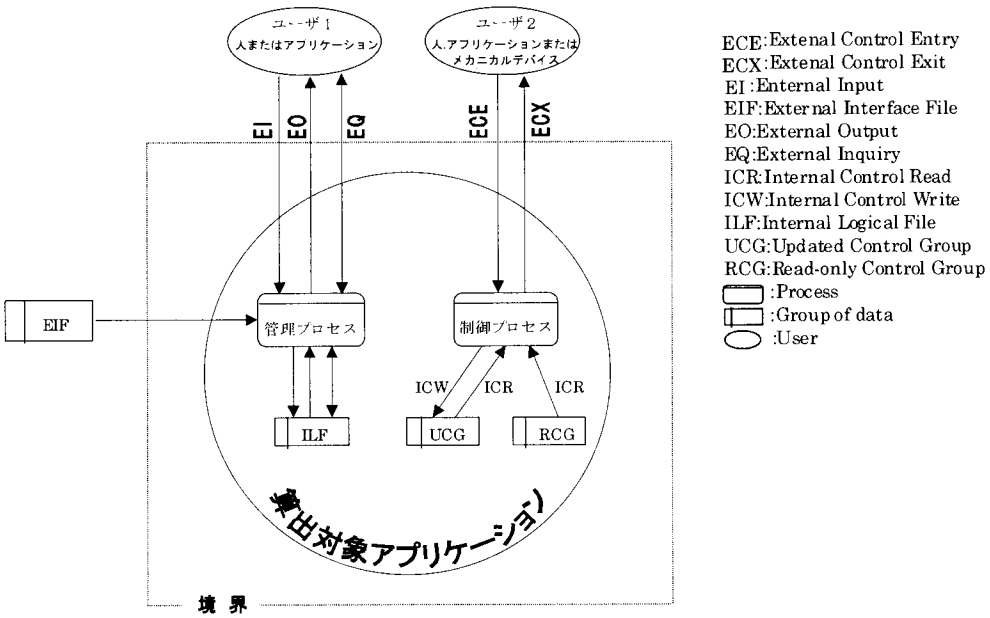



**図3 - FFP ファンクションタイプ 図**

次に、新しい拡張技法（FFP）をもとにした、アプリケーションの未調整ファンクションポイント数の式を示す。

FFP= 管理 FP + 制御 FP
= (FP法 – 制御情報) + 制御 FP

## 5.2 ポイント の割付け

前節ではファンクションポイントの抽出方法について述べた。次に、計測アプリケーションで抽出した各ファンクションに対するポイント割付けについて述べる。

### 5.2.1 管理ファンクションタイプに対するポイント割付け

FFPの場合も、管理ファンクションタイプ（ILF、EIF、EI、EO および EQ： 表3）の一部に対するポイントの割付け手順は標準 FP法と同じである。従って、ここでは IFPUG（IFPUG、1994）ポイント割付け手順がそのまま適用される。

### 5.2.2 制御ファンクションタイプに対するポイント割付け

#### 5.2.2.1 制御データ ファンクションタイプ に対するポイント割付け

##### 5.2.2.1.1 マルチオカレンス データグループ

マルチオカレンス データグループ (4.1章)は、FP法における ILF や EIF と同じ構造である。従って、マルチオカレンス データグループは、この２つの FP法ファンクションタイプとまったく同じ要領で、算出する。すなわち、データ エレメントタイプ（DET）とレコード エレメントタイプ（RET） の数から対応する複雑度表（IFPUG 1994定義と複雑度表参照）を使ってポイントを割付ける。

##### 5.2.2.1.2 シングルオカレンス データグループ

シングルオカレンス データグループ（4.1節参照）のポイント割付け数はDETの数のみで決まる。DETの数が決まればUCGのポイント数は ＜（DET数／５）＋５>の式で計算される。また、RCGのポイント数は（DET数／５）の式で計算される。これらの式は、シングルオカレンス データグループのサイズと、FP法のILFやEIFのサイズとの釣り合いを考慮して決定した。

計測アプリケーションの、シングルオカレンスUCGはすべて、（機能視点より見た場合）更新制御シングルデータから成り立っている。アプリケーション内のこれらすべてのシングル値をひとつにまとめて、全データから構成されるひとつのUCGとして扱うものとする。従って、アプリケーション内の、マルチオカレンスUCGは複数のこともあるが、シングルオカレンスUCGは常にひとつである。シングルオカレンスRCGも同様である。

典型的なリアルタイム アプリケーションの場合、このようなシングル値の数のバラツキは、数個から数百の幅に及ぶ。FP法のポイントを割付ける際に、標準 FP法 のポイント割付けの３段階複雑度評価方式ではなく、数式を使ったのはこのためである。FFPの場合、シングルオカレンス データグループの数が大きくなっても十分に使えるよう考慮されている。以下の表は数式の計算結果の例を示している。

シングルオカレンス UCG ポイント数の割付け例： <（DET数／５）＋５>:

| **Number of DET:** | 10 DET | 35 DET | 50 DET |
|---|---|---|---|
| **Points:** | 7 | 12 | 15 |

**表4: UCG ポイント割付け例**

シングルオカレンス RCG ポイント 数の割付け例： （DET 数 ／ ５）：

| **Number of DET:** | 10 DET | 35 DET | 50 DET |
|---|---|---|---|
| **Points:** | 2 | 7 | 10 |

**表5: RCG ポイント割付け例**

シングルオカレンス UCG と RCG の他の例については、第7章を参照のこと。

#### 5.2.2.2 制御トランザクション ファンクションタイプに対するポイントの割付け

制御トランザクションファンクションタイプ（ECE, ECX, ICWおよびICR）に対するポイントの割付け数は、DET[9]の数によって決まる。DETの数が決まれば、次の表を使ってポイント数に換算する。[10]:

| DET: | 1 to 19 DET | 20 to 50 DET | 51 + DET |
|---|---|---|---|
| Points: | 1 | 2 | 3 |

表6: 制御トランザクションファンクションタイプに対するポイントの割付け

これらの区分とポイントの割付けは、制御トランザクションファンクションタイプのサイズとFP法のサイズの釣り合いを考慮して決定した。

# 6. FFP 算出手順と規則

次の図は、FP法の現行の算出手順の概略を示している。

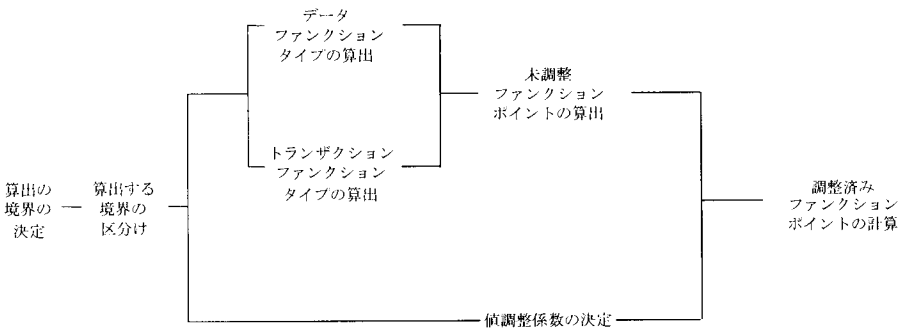



図4 – FP法の算出手順図

算出手順は次の通りである（IFPUG,1994)。：

1. ファンクションポイントのタイプの判定
2. 算出境界の区分け

[9] DET：ユニークで、ユーザが識別でき、繰り返しのないフィールド(IFPUG 94,用語集）。ファンクションタイプが入力、出力、読みとり、書き込みを行うフィールドのいずれかである。

[10] エンジンの診断例の場合、ダッシュボードに警告メッセージを送るサブプロセスの DETは２つ（ボタンの番号とスイッチオンのアクションコード）である。従って、ポイント数は１（表６の「1〜19DET」列）である。

3. データ ファンクションタイプの未調整ファンクションポイント数の算出
4. トランザクション ファンクションタイプの未調整ファンクションポイント数の算出
5. 値調整係数の決定
6. 最終調整済みファンクションポイント数の算出

FFPの場合も、ステップの 1,2,5及び6 はFP法とまったく同じである。ステップ3と4は図5に示すように、管理ファンクションタイプと制御ファンクションタイプにわかれる。

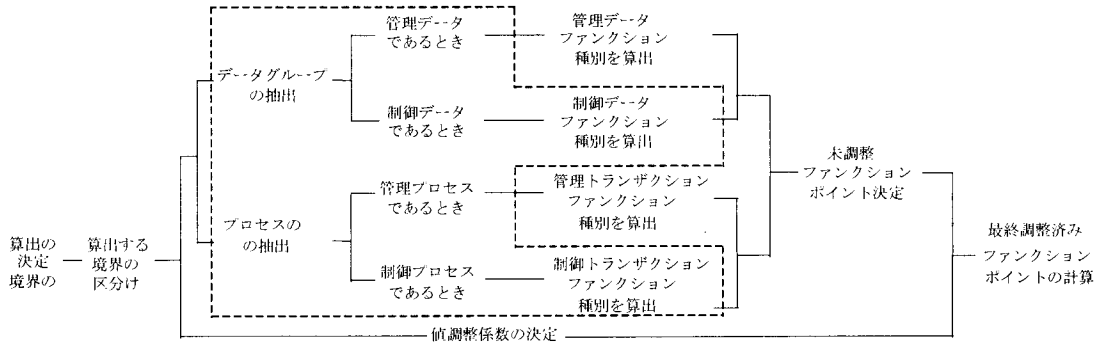



**図5 – FFPの算出手順図**

点線枠内の手順は、FFP手法の新ステップの部分であり、これについてはこの章で説明する。残りはの部分は、FP法の算出マニュアルの通りである（IFPUG 1994）。

## 6.1 データグループの抽出

このステップは、計測対象アプリケーションが、ユーザにデータを提供する機能を受け持つデータグループを抽出する。また、データグループを抽出した後、ファンクションタイプに関する定義と規則を適用し、抽出データグループをファンクションタイプとして算出できるかどうかを判定する。

### 定義

**データグループ**: 機能視点より認識され、グループ化されたデータ。

**管理データ**: 特にビジネス情報や経営情報などを管理するユーザ支援アプリケーションデータ。

制御データ: アプリケーションや機械装置の振る舞いを直接または間接的に制御するアプリケーションデータ。

### 抽出手順

次にデータグループの候補を抽出する手順を示す。

1. 機能的視点から識別可能なデータグループを探す。

   注: アプリケーションが更新するすべてのシングル制御データ（たとえば4.2章例1の温度敷居値）は、それぞれ更新や参照にユニークなデータグループとして統合する。リードオンリーコントロールデータも同様である。

2. 前述の規則に従い、データグループが*管理データグループか制御データグループか*を判定する。管理データグループの場合、既存のILFやEIFの算出手順と規則（IFPUG 1994）を適用する。制御データグループに対しては以下の算出手順と規則を適用する。

## 6.2 制御データファンクションタイプの算出

### 定義

アップデーテッドコントロールグループ (UCG): 計測対象アプリケーションが更新する制御データグループであり、機能的視点より認識可能なものである。その制御データは複数のトランザクションのために存在する。

リードオンリーコントロールグループ (RCG): 計測対象アプリケーションが参照はするが、更新はしない制御データグループであり、機能的視点より認識可能なものである。その制御データは複数のトランザクションのために存在する。

### 組込型関連用語の定義

機能的視点：アプリケーションがもたらす機能面からの視点。あくまでも技術面、実現面からの配慮を取り除いたものである。

トランザクション：外部のトリガーの発生に関連して発生するすべてのプロセス処理

### 算出手順

前節のステップで、制御データグループとして抽出された各データグループに対し、

1. 定義と規則を使い、その制御データグループがUCGかRCGかを決定する。

2. UCGまたはRCGの点数を割付け、未調整ファンクションポイント数（ポイントの割付け）を求める。

### 抽出規則

UCG 抽出規則：

- ❒ グループが論理的に関連するデータグループかあるいはシングルオカレンスデータグループである。
- ❒ データグループがアプリケーションの境界内で更新される。
- ❒ データグループが複数のトランザクションのために存在する。
- ❒ データグループがアプリケーションのRCG, ILFまたはEIFとして算出されてはいない。

機能的視点より、これらの算出規則がすべて適用され、すべての規則が満たされなければUCGとして抽出できない。

RCG 抽出規則：

- ❒ グループが論理的に関連するグループかあるいはシングルオカレンスデータグループである。
- ❒ データグループがアプリケーションによって更新されない。
- ❒ データグループがアプリケーションによって参照される。
- ❒ データグループが複数のトランザクションのために存在する。
- ❒ データグループがそのアプリケーションのUCG, ILFまたはEIFとして算出されてはいない。

機能的視点より、これらの算出規則がすべて適用され、すべての規則が満たされなければRCGとして抽出できない。

### ポイントの割付け

UCGとRCGの割付けポイント数は、制御データグループの種類（シングルオカレンスまたはマルチオカレンス）によって決まる。マルチオカレンスデータグループはFP法のILFやEIFと構造が同じであるため、FP法のファンクションタイプのカウントとまったく同じ要領で算出する。すなわち、DET数やRET数、関連複雑度マトリックスがそのまま用いられる。

シングルオカレンスデータグループの場合，ポイント数は、DETの数のみで決定する。ILFやEIFの場合と同じ規則でDETの数を求め、次の公式を使ってポイント数を計算する。：

UCG: <(DET数／5) + 5>
RCG: (DET数／5)

## 6.3 プロセスの抽出

（管理や制御の）データファンクションポイントを算出した後、トランザクションファンクションタイプを抽出する。トランザクションファンクションタイプはアプリケーションがユーザに提供するデータの処理機能をあらわす。従って、トランザクションファンクションタイプを抽出するには、まずアプリケーションのプロセスを識別しなければならない。

### 抽出手順

以下のステップでプロセスを識別する。:

1. 機能的視点より、アプリケーション内の異なるプロセスを見つける。
2. プロセスが管理プロセスか制御プロセスかを次の定義を用いて判定する。:

   管理プロセス:プロセスの目的が、ビジネス情報や経営情報などを管理するユーザ支援である。

   制御プロセス:アプリケーションや機械装置の振る舞いを直接または間接的に制御するプロセス。

3. もしプロセスが制御プロセスである場合は、新しい４つのトランザクションファンクションタイプの定義と規則を適用する。またプロセスが管理プロセスである場合は、既存のFP法のトランザクションファンクションタイプの定義と規則を適用する。

## 6.4 制御トランザクション ファンクションタイプの算出

### 定義

エクスターナル コントロールエントリ (ECE): ユニークなサブプロセスであり、機能的視点より認識可能なものである。ECE は境界の外からの入力制御データを処理する。なお、それはデータグループを処理するプロセスを分解した最小レベルである。従って、制御プロセスが２つの異なるデータグループを受け取る場合、ECE は少なくとも２つ存在する。またECEはデータ

の更新を含まない。更新機能は、別の制御ファンクションタイプで扱われる（インターナル コントロールライト）。

エクスターナル コントロールエグジット (ECX): ユニークなサブプロセスであり、機能的視点より認識可能なものである。ECX は境界外への出力を処理する。なお、それはデータグループを処理するプロセスを分解した最小レベルである。従って、制御プロセスが２つの異なるデータグループを送り出す場合、ECXは少なくとも２つで存在する。ECXはデータの読みとりを含まない。読みとり機能は、別の制御ファンクションタイプで扱われる（インターナル コントロールリード）。

インターナル コントロールリード (ICR): ユニークなサブプロセスであり、機能的視点より認識可能なものである。ICR は制御データを読みとる。なお、それはデータグループを処理するプロセスを分解した最小レベルである。従って、ひとつの制御プロセスが２つの異なるデータグループを読みとる場合、ICRは少なくとも２つ存在する。

インターナル コントロールライト (ICW): ユニークなサブプロセスであり、機能的視点より認識可能なものである。ICW は制御データを書き込む。なお、それはデータグループを処理するプロセスを分解した最小レベルである。従って、ひとつの制御プロセスが２つのデータグループを書き込む場合、ICWは少なくとも２つ存在する。

## 組み込み型の用語の定義

サブプロセス:エントリ、エグジット、リード、ライトなど、機能的視点より抽出できる最小のプロセス。

制御プロセス:アプリケーションや機械装置の振る舞いを直接的または間接的に制御するプロセス。

機能的視点:アプリケーションが与える機能の視点。技術上または実施上の配慮は一切排除される。

制御データ:アプリケーションや機械装置の振る舞いを直接的または間接的に制御するデータ。

ユーザ:計測対象アプリケーションと情報のやり取りをする人、アプリケーション、または機械装置。

## 算出手順

プロセスが制御プロセスとして判定された場合、次のステップに従う。:

制御プロセスのすべての機能的サブプロセスを抽出する（技術的サブプロセスではない）。

定義と規則をもとに、各サブプロセスがECE，ECX，ICRまたはICWかどうかを判定する。

ECE，ECX，ICRまたはICWの点数を割付け、未調整ファンクションポイント数（ポイントの割付け）を求める。

サブプロセス識別ステップ:

1. プロセス内のサブプロセスの論理的な実行順序に従い、最初の実行サブプロセス（ECE，ECX，ICRまたはICWのいずれか）を抽出する。
2. ECE，ECX，ICRまたはICWの規則を適用する。
3. ECE，ECX，ICRまたはICWの点数を割付け、未調整ファンクションポイント数（ポイントの割付け）を求める。
4. 再び、実行順序に従い、次のサブプロセス（ECE，ECX，ICRまたはICWのいずれか）を抽出する。次のサブプロセスはひとつではなく複数の場合もある（例えば、２つの選択肢を持つ"IF"ステートメントなど）。その場合、すべての関係経路を調べ、新しいサブプロセスが抽出されるかどうかを確かめる。
5. 2～4までのステップを繰り返し、すべてのプロセスのサブプロセスを抽出する。
6. 最後に、サブプロセスの重複がないかどうかを確認し、重複があれば取り除く（同一のプロセス処理[11]や同一のDETなど)。

   注: ひとつのサブプロセスが複数の制御プロセスに関係している場合、各制御プロセスごとに算出する。

[11] プロセス処理は入力、出力、読みとりまたは、書き込みだけでなくこれらの抽出したサブプロセスと結びついた他の種類のプロセス処理がある。（計算、フィルタリング、比較等に）。例えばエンジン診断の場合、算出対象であるスイッチオン・メッセージの送り出しプロセス処理ECXの他に敷居値と入力温度の比較のサブプロセスが存在する。しかし、このサブプロセスはECX処理に含まれる。

## 抽出規則

ECE 抽出規則:

- ❒ サブプロセスがアプリケーションの境界の外から制御データグループを受け取る。
- ❒ サブプロセスがひとつのデータグループを受け取る。複数のデータグループを受け取る場合、データグループごとにひとつのECEとして算出する。
- ❒ サブプロセスがデータの送り出し、読みとりまたは書き込みを行わない。
- ❒ サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のECEのものとは異なっている。

注 1:時計のトリガーは外部からのものと考える。従って、3秒ごとに起こるイベントは、1DETのECEとして算出する。しかし、周期的に発生するイベントは無視する。

注 2:内部での時間の読みとりは、特別のプロセス処理がない限り算出しない。例えば、プロセスが時刻を書き込む場合、内部クロック値の読みとりに対してICRとして算出はしない。

機能的視点より、ここに記述した算出規則がすべて適用され、すべての規則が満たされなければ、ICRとして抽出できない。

ECX 抽出規則:*

- ❒ サブプロセスがアプリケーションの境界の外に制御データを送り出す。
- ❒ サブプロセスがひとつのデータグループを送り出す。もし複数のデータグループをアプリケーションの境界の外に送り出す時は、各データグループごとにひとつのECXとして算出する。
- ❒ サブプロセスがデータの受け取り、読みとりまたは書き込みを行わない。
- ❒ サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のECXのものとは異なっている。

注: ユーザデータがないメッセージ（確認とエラーメッセージなど）はすべてECXとして算出する。DETの数は異なるタイプのメッセージの数である。

機能的視点より、ここに記述したすべての規則が満たされなければ、ECXとして抽出できない。

---

* たとえばエンジン診断例では、作動オンメッセージの送り出しがECXとして算出される。なおメッセージを作成するためのデータ比較の機能はこの作動オンメッセージ送り出し機能のECXの中に含まれると考える。

ICR抽出規則:

- サブプロセスが制御データグループを読みとる。
- サブプロセスがひとつのデータグループを読みとる。サブプロセスが複数のデータグループを読みとる場合は、データグループごとにひとつのICRとして算出する。
- サブプロセスがデータの受け取り、送り出しまたは書き込みを行わない。
- サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICRのものと異なっている。

機能的視点より、ここに記述したすべての規則が満たされなければ、ICRとして抽出できない。

ICW抽出規則:

- サブプロセスが制御データグループを書き込みとる。
- サブプロセスがひとつのデータグループを書き込む。複数のデータグループを書き込む場合は、データグループごとにひとつのICWとして算出する。
- サブプロセスがデータの受け取り、送り出しまたは読みとりを行わない。
- サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICWのものと異なっている。

機能的視点より、ここに記述したすべての規則が満たされなければ、ICWとして抽出できない。

### ポイントの割付け

制御トランザクションファンクションタイプ(ECE, ECX, ICRおよびICW)の割付けポイント数は、DETの数によって決まる。DETは以下の規則を適用して算出する。:

ECEとECXの場合:

❒ ユーザが認識でき、ユニークで繰り返しがなく、アプリケーション境界をクロスするフィールド（データ項目）を、ひとつのDETとして算出する。

ICRの場合:

❒ ユーザが認識でき、ユニークで繰り返しがなく、ILF、EIF、UCG、RCGから読みとられるフィールド（データ項目）を、キーも含めてひとつのDETとして算出する。

ICWの場合:

❒ ユーザが認識でき、ユニークで繰り返しがなく、ILFまたはUCGに書き込まれたフィールド（データ項目）を、キーも含めてひとつのICWとして算出する。

DETの数が決まったら、表6を用いてポイント数に換算する。

# 7. 算出事例

簡略型オーブン制御とモニタの事例

**要求仕様書**

データグループ:

アプリケーションが維持管理するデータグループ:

-メッセージログ[12]

フィールド:

-メッセージタイプ（オーブンスイッチオン、スイッチオフ）

-時刻（メッセージを送った時間）

-テンポラリフィールド:

-期待制御温度（オーブン温度として到達し、維持されるべき温度）

-警告メッセージ送付記録（送った警告メッセージ回数）

アプリケーションが読みとるデータグループ:

---

[12] ログデータ(Log data)例:

| メッセージタイプ | | 時刻 |
|---|---|---|
| オーブンエレメント | スイッチオン | 12:00 |
| オーブンエレメント | スイッチオン | 12:01 |
| オーブンエレメント | スイッチオン | 12:02 |
| ... | | |
| オーブンエレメント | スイッチオフ | 12:15 |

-アプリケーションが使用する要求定数:

-警告リミットタイム（警告メッセージが出るまでの連続加熱許容時間）

-シャットオフ敷居値（オーブンが自動シャット オフされる前に送られる警告メッセージ回数）

プロセス:

-温度制御:

オーブン温度はセンサーから受け取る

オーブン温度と期待制御温度の関係でスイッチをオン･オフさせるメッセージをオーブンヒーテイングエレメントに送る。

新しい記録がメッセージログ フィールドで作られる。:

-オーブンのモニタ

外部クロックのトリガーによりプロセスが作動する（アプリケーション境界の外部）。

サブプロセスは、警告リミットタイムを手がかりに、メッセージログから過熱状態を探す[13]。このサブプロセスが使用するデータ:

-メッセージログ：メッセージタイプと時刻

-リードオンリーコントロールグループ：シャットオフ敷居値と警告リミットタイム

過熱と判定された場合、警告メッセージがコントロールパネルに送られる。

モニタ中に過熱と判定された場合、警告メッセージ送付記録が更新される（送出された警告メッセージ回数が一回増える）過熱状態でなければ、元のままである。

警告メッセージ回数がシャットオフ敷居値に達した場合、オーブンにシャットオフメッセージが送られる。[14]

事例の単純化のため、ここでは期待制御温度の更新など他のユーザ要求仕様の説明は省略してある。しかし、後章で記述する算出事例では、期待制御温度の更新があるものとしてそれらを含めて算出する。

---

[13] 警告リミットタイムが30分の場合、メッセージログが30分間スイッチオン状態であれば、警告メッセージが送られる。オーブン温度が30分を過ぎても期待温度に達しなければ、センサは故障と判断し、警告メッセージが送られる。

[14] シャットオフ敷居値の警告回数が３回である場合、警告メッセージを３回送った時点でオーブンは自動的に動作を停止する。

## 7.2 FFP 算出

### 7.2.1 データグループの識別

ユーザの要求仕様書から３つのデータグループがあることがわかる。:
-メッセージログ
　フィールド:メッセージタイプ、時刻

-アップデーテッド シングルオカレンス データグループ
　フィールド:期待制御温度、警告メッセージ送付記録

-リードオンリー シングルオカレンス データグループ
　フィールド: 警告リミットタイム、シャットオフ敷居値

これらのデータグループは、アプリケーションの振る舞いを制御するために用いる制御データである。

### 7.2.2 制御データファンクションタイプの算出

次に、メッセージログがUCGかどうか判定する一覧表を示す。

メッセージログ

| UCG 算出規則 | 規則の適用可否？ |
|---|---|
| グループが論理的に関連するデータグループかあるいはシングルオカレンスデータグループである。 | はい。メッセージログは論理的に関連するデータグループである。 |
| データグループがアプリケーションの境界内で更新される。 | はい。温度制御プロセスがメッセージログを更新する。 |
| データグループが複数のトランザクションのために存在する。 | はい。メッセージをオーブンヒーティングエレメントに送り出すたびにオーブンモニタプロセスはメッセージログを使う。また多くのログ値がメッセージログに入り、オーブンモニタプロセスで使われる。 |
| データグループがアプリケーションのRCG，ILFまたはEIFとして算出されてはいない。 | はい。データグループはRCG，ILFまたはEIFとして算出されておらず、規則に適合している。 |

この判定に従えば、メッセージログはUCGである。次にRETとDETの算出とポイントの割付けを示す。

RET

| **RET 算出規則** | **規則の適用可否?** |
|---|---|
| 任意または必須の各UCGサブグループに対してひとつのRETとして算出する。 | メッセージログにサブグループはない。 |
| *Or* | |
| サブグループがない場合、UCGをひとつのRETとして算出する。 | サブグループがないため、1RETとして算出する。 |

1 RET

メッセージタイプと時刻

| **DET 算出規則** | **DET** |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、UCGの各フィールドを、1DETとして算出する。 | -メッセージのタイプ<br>-時刻 |
| 各外部キーに対して、ひとつのDETとして算出する。 | このタイプのデータはない。 |
| 実現技術関連のデータ項目は、データ項目のフィールド全体を、ひとつのDETとして算出する。 | このタイプのデータはない。 |

2 DET

UCGにIFPUGのILF表（IFPUG 1994）をUCGにも適用し、RETが１で、DETが２であるから、メッセージログは７ポイントとなる。

アップデーテッド シングルオカレンス コントロールデータ

要求仕様書によれば、アップデーテッド シングルオカレンス コントロールデータがひとつある。次に、これがUCGかどうかを判定する一覧表を示す。

フィールド:

-期待制御温度（オーブン温度として到達し、維持されるべき温度）

-警告メッセージ送付記録（送出した警告メッセージの回数）

| **UCG 算出規則** | **規則の適用可否?** |
|---|---|
| グループが論理的に関連するデータグループか、あるいはシングルオカレンス データグループである。 | はい。グループはシングルオカレンスデータグループである。 |
| データグループがアプリケーション内で更新される。 | はい。プロセスによって警告メッセージ送付記録が更新される。期待制御温度は別のプロセスによって更新される。後者は事例を簡単にする理由で要求仕様書の中では記述していない。しかし、後述の算出要約章では当表の前提で算出している。 |
| データグループが複数のトランザクションのために存在する。 | はい。期待制御温度と警告メッセージ送付記録は温度制御やオーブンモニタの多くのトランザクションに使用されている。 |
| データグループがアプリケーションのRCG, ILFまたはEIFとして算出されてはいない。 | はい。期待制御温度と警告メッセージ送付記録は、RCG, ILFまたはEIFとして算出されておらず、規則に適合している。 |

この判定によれば、このシングルオカレンス データグループはUCGである。次にこのDETの算出とポイントの割付けを示す。

期待制御温度と警告メッセージ送付記録

| **DET 算出規則** | **DET** |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、UCGの各フィールドを、ひとつのDETとして算出する。 | -期待制御温度<br>-警告メッセージ送付記録 |
| 各外部キーを、それぞれひとつのDETとして算出する。 | このタイプのデータはない。 |
| 実現技術関連のデータ項目は、データ項目のフィールド全体を、ひとつのDETとして算出する。 | このタイプのフィールドはない。 |

2 DET

注: 通常、シングルオカレンスUCGのDETの数は、計測アプリケーションのすべてのシングルオカレンスフィールドを含むため、これよりはるかに多くなる。

**識別したデータグループのポイント割付け:**

ポイント ＝<(DET数／5) + 5>
ポイント ＝<(2／5) +5> = 5.4

## リードオンリー シングルオカレンス コントロールデータ

要求仕様書によれば、リードオンリー シングルオカレンス コントロールデータがひとつある。次に、これがRCGかどうかを判定する一覧表を示す。

フィールド:
-警告リミットタイム（警告メッセージが出るまでの連続加熱許容時間）
-シャットオフ敷居値（オーブンが自動的にシャットオフする前に送出される警告メッセージ回数）

| RCG 算出規則 | 規則の適用可否? |
|---|---|
| グループが論理的に関連するデータグループか、あるいはシングルオカレンス データグループである。 | はい。グループはシングルオカレンス データグループである。 |
| データグループがアプリケーションによって更新されない。 | はい。 |
| データグループが複数のトランザクションのために存在する。 | はい。警告リミットタイムとシャットオフ敷居値は多くのオーブンモニタプロセスのトランザクションで使用されている。 |
| データグループがアプリケーションのUCG，ILFまたはEIFとして算出されない。 | はい。警告リミットタイムとシャットオフ敷居値は、UCG，ILFまたはEIFとして算出されておらず、規則に適合している。 |

この判定によれば、このシングルオカレンス データグループはRCGである。次にこのDETの算出とポイントの割付けを示す。

## 警告リミットタイムとシャットオフ敷居値

| DET 算出規則 | DET |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、RCGの各フィールドを、1DETとして算出する。 | -警告リミットタイム<br>-シャットオフ敷居値 |
| 各外部キーをそれぞれひとつのDETとして算出する。 | このタイプのデータはない。 |

| | |
|---|---|
| 実現技術関連のデータ項目は、データ項目のフィールド全体を、ひとつのDETとして算出する。 | このタイプのフィールドはない。 |

2 DET

注: 通常、シングルオカレンスRCGのDETの数は、計測アプリケーションのすべてのリードオンリーシングルオカレンスフィールドを含むため、これよりはるかに多くなる。

**識別したデータグループのポイント割付け:**

ポイント =(DET数／5)
ポイント=(2／5) = 0.4

### 7.2.3 プロセスの抽出

ユーザの要求仕様書から２つのプロセスがあることがわかる。:
-温度制御
-オーブンモニタ

これらは、アプリケーションの振る舞いを制御する制御プロセスである。従って、ECE，ECX，ICRまたはICWの規則を用いて算出する。

### 7.2.4 制御トランザクション ファンクションタイプの算出

識別した各制御プロセスに対してまずサブプロセス候補を抽出し、次に各サブプロセス候補に対してサブプロセスのタイプを判定しDETの数を求め、これをもとにポイント数を算出する。次に例を示す。:

**温度制御**

この制御プロセスでサブプロセスと思われるのは次のものである:
1. センサーからオーブン温度を受け取る。
2. シングルオカレンス データグループから期待制御温度を読みとる。
3. オーブン温度と期待制御温度を比較する。
4. サブプロセス３の結果をもとにスイッチのオン・オフメッセージを送る。
5. メッセージログを更新する。

１番目のサブプロセス（センサーからオーブン温度を受け取る）は、アプリケーションの境界の外から制御データを受け取る。従って、これはECEである可能性がある。次に、このサブプロセスがECEかどうかを判定する一覧表を示す。ECEである場合、規則をもとにDETを求め、ポイント数を算出する。

センサーからのオーブン温度の受け取り

| ECE 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスがアプリケーションの境界の外から制御データグループを受け取る。 | はい。サブプロセスがセンサーからオーブン温度を受取る。 |
| サブプロセスがひとつのデータグループを受け取る。複数のデータグループを受け取る場合、データグループごとにひとつのECEとして算出する。 | はい。ひとつのデータグループを受取る。 |
| サブプロセスがデータの送り出し、読みとりまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のECEのものと異なっている。 | はい。 |

この判定によれば、このオーブン温度を受取るサブプロセスはECEである。

オーブン温度

| DET 算出規則 | DET |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、アプリケーションの境界をクロスするフィールドを、それぞれ1DETとして算出する。 | -オーブン温度 |

1 DET

ポイントの割付け

制御トランザクション ファンクションタイプ換算表（表６）とDET数１から、このオーブンの温度受け取りサブプロセスは１ポイントとなる。

２番目のサブプロセスは、データ（期待制御温度）を読みとるためのものである。これはICRである可能性がある。次に、このサブプロセスがICRかどうかを判定する一覧表を示す。ICRである場合、規則をもとに、DETを求め、ポイント数を算出する。

シングルオカレンス データグループ内の期待制御温度の読みとり

| ICR 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを読みとる。 | はい。シングルオカレンスUCGから期待制御温度を読みとる。 |
| サブプロセスがひとつのデータグループを読みとる。サブプロセスが複数のデータグループを読みとる場合は、データグループごとにひとつのICRとして算出する。 | はい。ひとつのデータグループを読みとる。 |
| サブプロセスがデータの受け取り、送り出しまたは書き込みを行わない。 | はい。 |

| | |
|---|---|
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICRのものと異なっている。 | はい。 |

期待制御温度

| DET 算出規則 | DET |
|---|---|
| ILF，EIF，UCGまたはRCGから読みとるユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -期待制御温度 |

1 DET

制御トランザクション ファンクションタイプ換算表（表6）とDET数１から、この期待制御温度読みとりサブプロセスは１ポイントとなる。

２つの温度（オーブン温度と期待制御温度）を比較する３番目のサブプロセス候補は、制御データの受け取り、送り出し、読みとりまたは書き込みを行っていない。従って、このサブプロセス候補は、制御トランザクションファンクションとして算出しない。

４番目のサブプロセスは、アプリケーションの境界の外へ制御データを送り出すためのものである。これはECXである可能性がある。次に、このサブプロセスがECXかどうかを判定する一覧表を示す。ECXである場合、規則をもとにDETを求め、ポイント数を算出する。

オーブンエレメントへのメッセージの送り出し

| ECX 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスがアプリケーションの境界外に制御データグループを送り出す。 | はい。メッセージをアプリケーションの境界の外に送り出す。 |
| サブプロセスがひとつのデータグループを送り出す。もし複数のデータグループをアプリケーションの境界の外に送り出すときは、各データグループごとにひとつのECXとして算出する。 | はい。ひとつタイプのメッセージを送り出す。 |
| サブプロセスがデータの受け取り、読みとりまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のECXのものと異なっている | はい。 |

アクションタイプ

| DET 算出規則 | DET |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、アプリケーションの境界をクロスするフィールドを、それぞれひとつのDETとして算出する。 | -アクションタイプ（スイッチオン、スイッチオフ） |

1 DET

制御トランザクション ファンクションタイプ換算表（表6）とDET数１から、オーブンエレメントへメッセージを送り出すサブプロセスは１ポイントとなる。

最後の５番目のサブプロセスはメッセージログの更新、すなわち制御データを書き込むためのものである。これはICWである可能性がある。次に、このサブプロセスがICWかどうかを判定する一覧表を示す。ICWである場合、規則をもとにDETを求め、ポイント数を算出する。

メッセージログの更新

| ICW 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを書き込む。 | はい。メッセージを送るたびにメッセージログに記録する。 |
| サブプロセスがひとつのデータグループを書き込む。複数のデータグループを書き込む場合は、データグループごとにひとつのICWとして算出する。 | はい。メッセージログを更新する。 |
| サブプロセスがデータの受け取り、送り出しまたは読みとりを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したデータエレメントが同一プロセスの他のICWのものと異なっている。 | はい。 |

メッセージタイプと時刻

| DET 算出規則 | DET |
|---|---|
| ILFまたはUCGに書き込んだユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -メッセージタイプ<br>-時刻 |

1 DET

制御トランザクション ファンクションタイプ換算表（表6）とDET数１から、オーブンエレメントへメッセージを送り出すサブプロセスは１ポイントとなる。

温度制御の重複サブプロセスの削除について

算出手順（6章　6.4節）で述べたように、何らかの理由で同一プロセス内で同じサブプロセスが２回以上抽出された場合、ひとつだけを算出する。

上記の温度制御プロセスの例の場合、識別されたサブプロセスはすべて、ユニークなプロセス処理とユニークなデータのみであり、サブプロセスの重複はない。

次に２番目のプロセス（7.2.3節）の判定結果を示す。:

オーブンモニタ

この制御プロセスのサブプロセス候補は次の通りである。:

1　外部のクロックからトリガーを受け取る。
2　シングルオカレンスRCGから警告リミットタイムとシャットオフ敷居値を読みとる。
3　メッセージログからメッセージタイプと時刻を読みとる。
4　読みとったデータから過熱状態を探す。
5　過熱状態がある場合（スイッチオンメッセージ継続時期が警告リミットタイムを超えた場合）、警告メッセージを送る。
6　シングルオカレンスUCGから送られた警告メッセージを読みとる。
7　警告メッセージ送付記録を更新する。
8　警告回数がシャットオフ敷居値に達したら（警告メッセージ送付記録の回数＞シャットオフ敷居の回数）、シャットオフメッセージを送り出す。

１番目のサブプロセス（外部のクロックからトリガーを受け取る）は、アプリケーションの境界の外から制御データを受け取る。従って、これはECEである可能性がある。次に、このサブプロセスがECEかどうかを判定する一覧表を示す。ECEである場合、規則をもとにDETを求め、ポイント数を算出する。

外部のクロックからトリガーを受け取る

| ECE 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスがアプリケーションの境界の外から制御データグループを受け取る。 | はい。サブプロセスがアプリケーションの境界の外からトリガーを受け取る。 |
| サブプロセスがひとつのデータグループを受け取る。複数のデータグループを受け取る場合、データグループごとにひとつのECEとして算出する。 | はい。ひとつのトリガーを受け取る。 |
| サブプロセスがデータの送り出し、読みと | はい。 |

| | |
|---|---|
| りまたは書き込みを行わない。 | |
| サブプロセスがユニークであり、抽出したサブプロセスのプロセス処理とデータエレメントが同一プロセスの他のECEのものとは異なっている。 | はい。 |

トリガー

| DET 算出規則 | DET |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、アプリケーションの境界をクロスするフィールドを、それぞれひとつのDETとして算出する。 | -トリガー |

1 DET

制御トランザクションファンクションタイプ換算表(表6)とDET数1から外部のクロックからトリガーを受け取るサブプロセスは1ポイントとなる。

２番目のサブプロセスはデータを読むためのものである。従ってこれはICRである可能性がある。

**RCGからの警告リミットタイムとシャットオフ敷居値の読みとり**

| ICR 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを読みとる。 | はい。シングルオカレンスRCGから警告リミットタイムとシャットオフ敷居値を読みとる。 |
| サブプロセスがひとつのデータグループを読みとる。サブプロセスが複数のデータグループを読みとる場合は、データグループごとにひとつのICRとして算出する。 | はい。ひとつのデータグループ(シングルオカレンスRCG）を読みとる。 |
| サブプロセスがデータの受け取り、送り出しまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICRのものと異なっている。 | はい。 |

警告リミットタイムとシャットオフ敷居値

| DET 算出規則 | DET |
|---|---|
| ILF，EIF，UCGまたはRCGから読みとるユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -警告リミットタイム<br>-シャットオフ敷居値 |

2 DET

制御トランザクション ファンクションタイプ換算表（表６）とDET数２から、こ

のサブプロセスは１ポイントとなる。

３番目のサブプロセス候補は、メッセージログから異なるデータグループを読みとる。：

メッセージログからのメッセージタイプと時刻の読みとり

| ICR 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを読みとる。 | はい。UCG(メッセージログ）からメッセージタイプと時刻を読みとる。 |
| サブプロセスがひとつのデータグループを読みとる。サブプロセスが複数のデータグループを読みとる場合、データグループごとにひとつのICRとして算出する。 | はい。ひとつのデータグループ(UCG：ログメッセージ）を読みとる。 |
| サブプロセスがデータの受け取り、送り出しまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントがプロセスの他のICRのものと異なっている。 | はい。 |

メッセージタイプと時刻

| DET 算出規則 | DET |
|---|---|
| ILF，EIF，またはRCGから読みとるユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -メッセージタイプ<br>-時刻 |

2 DET

制御トランザクションファンクションタイプ換算表（表６）とDET数２から、このサブプロセスは１ポイントとなる。

「読みとったデータから過熱状態を探す」４番目のサブプロセス候補は、データの受け取り、送り出し、読みとりまたは書き込みを行わない。従って、このサブプロセス候補はファンクションとして算出しない。

５番目のサブプロセスは、アプリケーションの境界の外に制御データを送り出す。従ってこれはECXである可能性がある。

過熱状態の時の警告メッセージの送り出し

| ECX 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループをアプリケーションの外に送り出す。 | はい。メッセージをアプリケーションの境界の外に送り出す。 |
| サブプロセスがひとつのデータグループを送り出す。もし複数のデータグループをアプリケーションの境界の外に送り出すときは、各データグループごとにひとつの | はい。ひとつタイプのメッセージを送り出す。 |

| | |
|---|---|
| ECXとして算出する。 | |
| サブプロセスがデータの受け取り、読み取りまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のECXのものと異なっている。 | はい。 |

メッセージ

| DET 算出規則 | DET |
|---|---|
| ユニークでユーザが認識でき、繰り返しがない、アプリケーションの境界をクロスするフィールドを、それぞれひとつのDETとして算出する。 | -メッセージ |

1 DET

制御トランザクション ファンクションタイプ換算表（表６）とDET数１から、このサブプロセスは１ポイントとなる。

６番目のサブプロセスはデータを読みとる。従ってこれはICRである可能性がある。

シングルオカレンスUCGからの警告メッセージの読みとり

| ICR 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを読みとる。 | はい。シングルオカレンスUCGから警告メッセージ送付記録を読みとる。 |
| サブプロセスがひとつのデータグループを読みとる。サブプロセスが複数のデータグループを読みとる場合は、データグループごとにひとつのICRとして算出する。 | はい。ひとつのデータグループ(シングルオカレンスUCG）を読みとる。 |
| サブプロセスがデータの受け取り、送り出しまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICRのものと異なっている。 | はい。 |

警告メッセージ記録

| DET 算出規則 | DET |
|---|---|
| ILF，EIF，UCGまたはRCGからユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -警告メッセージ送付記録 |

1 DET

制御トランザクション ファンクションタイプ換算表（表６）とDET数１から、

このサブプロセスは１ポイントとなる。

７番目のサブプロセスは警告メッセージ送付記録を書き込む。これはICWである可能性がある。

送られてきた警告メッセージの書き込み

| ICW 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスが制御データグループを書き込む。 | はい。シングルオカレンスUCGを更新する。 |
| サブプロセスがひとつのデータグループを書き込む。複数のデータグループを書き込む場合は、データグループごとにひとつのICWを算出する。 | はい。ひとつのシングルオカレンスUCGだけを更新する。 |
| サブプロセスがデータの受け取り、送り出しまたは読みとりを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントが同一プロセスの他のICWのものと異なっている。 | はい。 |

メッセージ

| DET 算出規則 | DET |
|---|---|
| ILFまたはUCGに書き込むユニークでユーザが認識でき、繰り返しがないフィールドを、キーも含めてそれぞれひとつのDETとして算出する。 | -メッセージ |

1 DET

制御トランザクションファンクションタイプ換算表（表６）とDET数１から、この警告メッセージ送付記録の書き込みサブプロセスは１ポイントとなる。

８番目のサブプロセスは、アプリケーションの境界の外に制御データを送出する。従ってこれはECXである可能性がある。

過熱状態の時の警告メッセージの送り出し

| ECX 算出規則 | 規則の適用可否? |
|---|---|
| サブプロセスがアプリケーションの外に制御データを送り出す。 | はい。メッセージをアプリケーションの境界の外に送り出す。 |
| サブプロセスがひとつのデータグループを送り出す。もし複数のデータグループを | はい。ひとつタイプのメッセージを送り出す。 |

| | |
|---|---|
| アプリケーションの境界の外に送り出すときは各データグループごとにひとつのECXとして算出する。 | |
| サブプロセスがデータの受け取り、読みとりまたは書き込みを行わない。 | はい。 |
| サブプロセスがユニークであり、抽出したプロセス処理とデータエレメントがプロセスの他のECXのものと異なっている。 | はい。 |

メッセージ

| **DET 算出規則** | **DET** |
|---|---|
| ユニークでユーザが認識でき、繰り返しがないフィールドを、それぞれひとつのDETとして算出する。 | -メッセージ |

1 DET

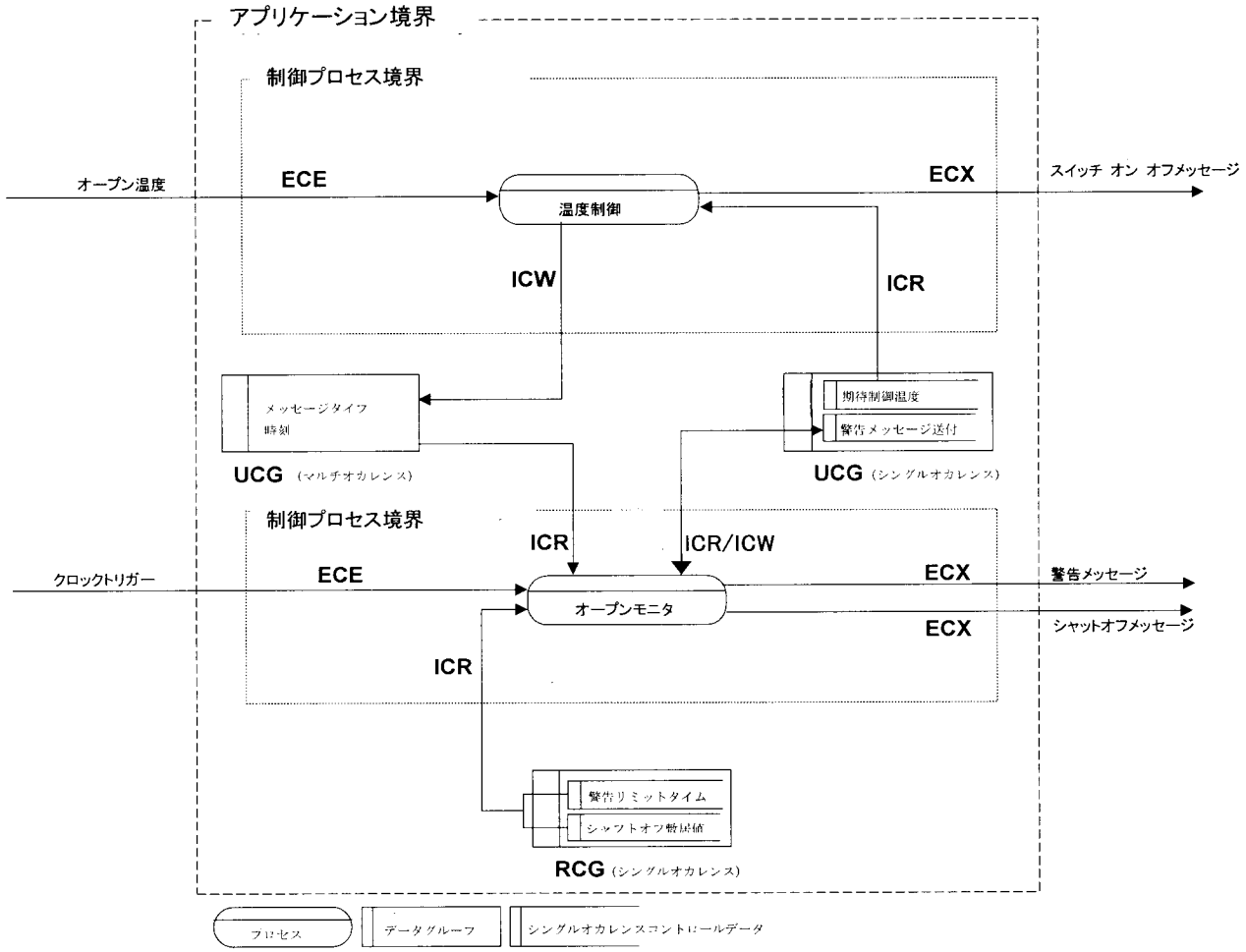
制御トランザクションファンクションタイプ換算表（表６）とDET数１から、このサブプロセスは１ポイントとなる。

### 自己診断の重複サブプロセスの削除について

算出手順の説明で述べたように、各制御プロセス内に重複したサブプロセスがある場合、重複サブプロセスを削除しなければならない。このオーブンモニタプロセスの場合、上記の温度制御プロセスの場合と同様に、抽出したすべてのサブプロセスに重複はない。

### 7.2.5 オーブン例の算出要約

本例の算出要約を以下に図示する。




注：アプリケーションによって更新されるすべての制御データをシングルオカレンス UCGとしてまとめる
アプリケーションによって読みとられるすべての制御データをシングルオカレンス RCGとしてまとめる

図 6 – オーブン例 算出要約図

次の表に、すべてのファンクションのポイント割り付けと未調整ファンクションポイント数を要約して示す。

| ファンクションの説明 | ファンクション タイプ | FTR/ RET | DET | Point |
|---|---|---|---|---|
| | | | | |
| **制御データファンクションタイプ** | | | | |
| メッセージログ[15] | UCG | 1 | 2 | 7.0 |
| 更新シングルオカレンスデータグループ[16] | UCG | N/A | 2 | 5.4 |
| リードオンリーシングルオカレンスデータグループ[17] | RCG | N/A | 2 | 0.4 |
| **データポイント計:** | | | | **12.8** |
| | | | | |
| **制御トランザクション ファンクションタイプ** | | | | |
| 温度制御 | | | | |
| センサーからのオーブン温度受け取り | ECE | N/A | 1 | 1 |
| UCGからの期待制御温度の読みとり | ICR | N/A | 1 | 1 |
| スイッチオン、オフの送り出し | ECX | N/A | 1 | 1 |
| メッセージログの更新 | ICW | N/A | 2 | 1 |
| **温度制御プロセスポイント:** | | | | **4** |
| | | | | |
| オーブンモニタ | | | | |
| 外部のクロックからのトリガーの受け取り | ECE | N/A | 1 | 1 |
| 警告リミットタイムとシャットオフ敷居値の読みとり | ICR | N/A | 2 | 1 |
| メッセージタイプと時刻の読みとり | ICR | N/A | 2 | 1 |
| 過熱状態時の警告メッセージの送り出し | ECX | N/A | 1 | 1 |
| UCGからの警告メッセージの読みとり | ICR | N/A | 1 | 1 |
| UCGからの警告メッセージの書き込み | ICW | N/A | 1 | 1 |
| 敷居値到達時のシャットオフメッセージ送り出し | ECX | N/A | 1 | 1 |
| **オーブンモニタプロセスのポイント:** | | | | **7** |
| | | | | |
| | | | | |
| **未調整ファンクションポイント数:** | | | | **23.8** |
| | | | | |

**第 7 表 - オーブン事例の算出要約**

[15] メッセージ ログフィールド(Message Log Fields) : メッセージタイプと時刻

[16] アップデーテッド シングルオカレンス データフィールドグループ(Updated Single Occurrence Group of data field):期待制御温度と警告メッセージ送付記録

[17] リードオンリー シングルオカレンス データフィールドグループ(Read-only Single Occurrence Group of data fields):警告リミットタイムとシャットオフ敷居値

我々は事例を用いて算出の内容を詳細に説明した。実際の算出では、これほど詳細な検討の必要はない。算出実務の面では、実際には（表７）程度の内容がわかれば十分に算出できる。

事例を簡略化するため、非常に小型のアプリケーションを例として用いた。典型的なリアルタイムアプリケーションの場合には、制御プロセス当たりのサブプロセス（ECE, ECX, ICRおよびICW）の数がこれよりもはるかに多くなる。この例の場合、プロセス当たりのサブプロセスは最大７つだが、実際のリアルタイムアプリケーションの中には、ひとつのプロセス内に５０個のサブプロセスがあるものもある。また、実際のリアルタイムアプリケーションの場合、DET数ははるかに多く、シングルオカレンス制御グループもはるかに複雑である。実際のリアルタイムソフトウエアの中には、シングルレコード制御グループが１００個以上のDETをもつ例もある。

# 8. FFP 適用トライからのフィードバック

この報告書はこれまで、事例を用いて、FFPのコンセプト、定義、カウント手順について説明した。次に、FFPの実際の使用状況について述べる。FFPはこれまで、産業界で実際に多くのリアルタイム アプリケーションに適用され、カウントを行っている。次に、実用現場からのフィードバックについて報告する。

## 8.1 理解のしやすさ

FFPのカウントはすべて、リアルタイムアプリケーションのエキスパートの支援を得て行われている（産業界のエキスパートは計測対象アプリケーションを熟知している）。アプリケーションのエキスパートは制御ファンクションタイプの定義を理解しさえすれば、それらを問題なく抽出できた。実際彼らは、FFPのカウントを丸一日行った後は、FFPの計測専門家の助けを借りずに算出できるようになった。

## 8.2 算出労力

カウンティングのフィールドテストにおけるカウント労力は、FP法の場合とほぼ同レベルだった。なお、FFPはFP法に比べ多くの機能を抽出することが判明した。また、FFPは簡単であるため[18]、新しい制御トランザクションファンクションタイプをすばやく抽出できることも判明した。さらに、制御トランザクションフ

[18] コントロール トランザクショナル ファンクション タイプ(Control Transactional Function Types)は、ひとつのサブ プロセスだけに関係するため比較的単純である。一般に、マネジメント トランザクショナル ファンクションタイプ(Management Transactional Function Types)は、多くのサブプロセスで構成され、エレメンタリ プロセスに組み込まれているため、識別するのに時間がかかる。

ファンクションタイプがよりシンプルであるため、アプリケーション専門家は、ファンクション計測専門家の助けをそれほど借りずに算出できることもわかった。

## 8.3 ドキュメントの重要性

アプリケーションを計測するには、関連機能情報資料を入手する必要がある。これらの機能情報は、アプリケーションのエキスパートやアプリケーションのドキュメントなどから入手できる。FFPも、FP法の場合と同様に機能情報の質が決め手となる。これまでのフィールドテストの結果をみる限り、FFPは産業界の通常の資料で十分に算出できる。

## 8.4 初期段階でのFFP算出

機能要求書が入手できれば、開発の初期段階でも制御ファンクションタイプは算出できる。あとからファンクションタイプが追加されても、FFPの詳細レベルはFP法の場合とほぼ同じである。初期段階でFP法を使用する場合も、すべての機能情報が揃っているわけではないので、概算的カウントとならざるをえない。FP法の場合は、初期段階の概略見積もり法が長年の間に多数開発されてきた。FFPの場合も、今後、見積もり法が時間をかけて徐々に開発されていくものと思われる。[19].

# 9. 要約

リアルタイムソフトウエアの機能計測法の開発の試みは偉大なる挑戦である。一般に、この種の計測法は、機能規模に基づく価値のある生産性評価基準や見積もりモデルの開発に道を開くものでなければならない。リアルタイム ソフトウエアに対する生産性評価基準や見積もりモデルの開発の第一歩は、計測者が同一ルールで計測できる環境を与えることである。同一ルールが確立されさえすれば、計測者は他のこれまでの実績をもとに計測することができる。独自のルールや既存ルールの独自解釈によるゼロからのスタートではなく、計測者はこれまで他で達成されたものをベースに計測することが可能となる。これは、ＭＩＳアプリケーションのファンクションポイントの適用の経緯と通じるものがある。Albrechtが1984年にファンクションポイントの構造を初めて発表して以来(Albrecht1984)、多くの人々がこの計測法に取り組んできた。また、IFPUGが10年以上にわたり大事に育て上げたルールをもとに、多くのプロジェクトのデータベース[20]を集積した。リアルタイムソフトウエアの機能性計測ルールも業界が合意し、リアルタイ

[19] 勿論、現時点では、FFPの概略見積もり法がFP法と同程度に成熟した手法であるとはだれも期待していない。

[20] 例えば、Gartner Group、International Software Benchmarking Standards Group(ISBSG)、Howard Rubin Associates、Software Productivity Researchなど

ム ソフトウエアに対する生産性評価基準モデルや見積もりモデルの開発を促進することが望まれる。

この報告書はFFPの発表に関する1997年9月版である。

我々は、このFFP計測技術がファンクションポイントの適用可能領域をひろげ、業界においてファンクションポイントの重要性が増大することを確信している。また我々は、法人及び計測協会が次のFFPの出版に向けての共同研究に参加されることを歓迎している。

# 用語集

以下に、IFPUGの用語集(IFPUG1994)に加えて使用する、FFP用語の定義を示す。

**制御データ<Control data>**: アプリケーションや機械装置の振る舞いを直接または間接的に制御するアプリケーションデータ。

**制御プロセス<Control process>**: アプリケーションや機械装置の振る舞いを直接または間接的に[21]制御するプロセス。

注: 複数のトリガーを持つプロセス：プロセスによっては複数のトリガーを持つものもある(はじめの主要なひとつの制御イベントとそれに付随するイベント)。このケースでは計測上は、全体のプロセスをひとつの制御プロセスとみなして処理する。

**データファンクションタイプ<Data function type>**: データ要求を満たすためにユーザに提供する機能性。データタイプはUCG, RCG, ILFおよびEIF として定義される。

**機能的視点<Functional perspective>**: アプリケーションが与える機能の視点。技術上または実現上の配慮は一切排除される。

**データグループ<Group of data>**: 機能視点より認識可能な、論理的に関連するデータグループ。

**管理データ<Management data>**: 特にビジネス情報や経営情報などを管理するユーザ支援アプリケーションデータ。

**管理プロセス<Management process>**: プロセスの目的が、ビジネス情報や経営情報などを管理するユーザ支援である。

**MISアプリケーション<MIS applications>**: 事務処理業務情報をサポートするアプリケーション。

**プロセス<Process>**: 入力が結果をもたらす過程の処理作業または活動。

**サブプロセス<Sub-process>**: この報告書では、エントリ、エグジット、リードおよびライトなど、機能的視点より抽出できる最小のプロセス。

---

[21] 例えば、人に報告する診断プロセスは、アプリケーションや機械装置を直接に制御しない。しかしこれも制御プロセスである。

トランザクション<**Transaction**>: 外部のトリガーの発生に関連して発生するすべてのプロセス処理。

トランザクション ファンクションタイプ<**Transactional function type**>: データを処理するアプリケーションがユーザに提供する機能性。トランザクションファンクションタイプはECE，ECX，ICR，ICW，EI，EOおよびEQとして定義される。

トリガー<**Trigger**>: 機能的視点よりプロセス処理を引き起こすと見られるイベント。イベントはアプリケーション境界の外部から入り込むものである。時計とタイミングイベントはトリガーの例といえる。

注1：複数のトリガーを持つプロセス[22]：プロセスによっては複数のトリガーを持つものもある(はじめの主要なひとつの制御イベントとそれに付随するイベント)。このケースでは計測上は、はじめの主要なイベントだけを取り上げてひとつのトリガーとして扱う。

注2：グラフィカルユーザインターフェース(GUI)：プロセス処理を引き起こすイベントの場合を除き、ユーザデータ(すなわちマウスクリック及びナビゲーションコマンド)を持たないGUIイベントは、カウント上は、トリガーではない。

更新<**Update**>: データ値を変更する能力。

ユーザ<**User**>: 計測対象アプリケーションと情報のやり取りをする人、アプリケーション、機械装置。

[22] すべての制御プロセスには、プロセス処理を引き起こす、はじめの主要なひとつのトリガーがある。しかしプロセスによっては更にそれに影響を与える追加の外部イベントを持つものもある。

# References


Abran, A., *Analysis of the measurement process of Function Points Analysis*, PhD. Thesis, Département de génie électrique et de génie informatique, École Polytechnique de Montréal, 1994, 405 pages.

Abran, A., and Robillard, P. N., *Function Point Analysis, An Empirical Study of its Measurement Processes* IEEE Transactions on Software Engineering, vol. 22, no. 12, pp. 895-909, Dec. 1996.

Albrecht, A.J., *AD/M Productivity Measurement and Estimate Validation*, IBM Corporate Information Systems, IBM Corp., Purchase, N.Y., May 1984.

Conte, S.D., Shen, V.Y., and Dunsmore, H.E., *Software Engineering Metrics and Models*, Benjamin Cummins Publishing, 1986, 396 pages.

Cooling, J. E., *Software Design for Real-Time Systems,* Chapman and Hall, 1991.

Desharnais, J. M., *Statistical Analysis on the Productivity of Data Processing with Development Projects using the Function Point Technique.* Université du Québec à Montréal. 1988.

Galea, S., *The Boeing Company: 3D Function Point Extensions, V2.0, Release 1.0*, Boeing Information and Support Services, Research and Technology Software Engineering, June 1995.

Grady, R. B., *Practical software metrics for project management and process improvement* Prentice Hall, New Jersey, 1992, 270 pages.

Hetzel, B., *Making Software Measurement Work,* QEB Publishing Group, 1993, 290 pages.

IEEE, *IEEE Standard Computer Dictionary: A compilation of IEEE Standard Computer Glossaries,* IEEE Std 610-1990, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, 1990.

IFPUG (1994). *Function Point Counting Practices Manual, Release 4.0,* International Function Point Users Group - IFPUG, Westerville, Ohio, 1994.

Illingworth, V., (1991) (editor), *Dictionary of Computing*, Oxford University Press, 3rd edition, 1991, 510 pages.

Ince, D. C., *History and industrial applications*, in Fenton, N.E., *Software Metrics: A Rigorous Approach*, Chapman & Hall, UK, 1991, 337 pages.

Jones, C., *A Short History of Function Points and Feature Points*, Software Productivity Research, Inc., Cambridge, Mass, 1988.

Jones, C., *Applied Software Measurement - Assuring Productivity and Quality*, McGraw-Hill, 1991, 493 pages.

Jones, C., *Applied Software Measurement - Assuring Productivity and Quality*, McGraw-Hill, 1996, 618 pages.

Kan, S. H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, 1993, 344 pages.

Kemerer, C. F., *An Empirical Validation of Software Cost Estimation Models*, Communications of the ACM, vol. 36, pp. 85-97, 1993.

Laplante, P., *Real-Time Systems Design and Analysis: An Engineer's Handbook*, The Institute of Electrical and Electronics Engineers Inc., New York, NY, 1993, 339 pages.

Stankovic, J. A. and Ramamritham, K., *Tutorial Hard Real-Time Systems,* IEEE Computer Society Press, Washington D.C., 1988, 618 pages.

Whitmire, S. A., *3-D Function Points: Scientific and Real-Time Extensions to Function Points Proceedings of the 1992 Pacific Northwest Software Quality Conference*, 1992.

**For more information on FFP:**

**Alain Abran Ph.D.**

Software Engineering Management Research Laboratory
Departement d'informatique
Universite du Quebec a Montreal
C.p. 8888 succursale centre-ville
Montreal (Quebec) Canada H3C 3P8

Telephone: 514 - 987-3000 (8900)
Fax: 514 - 987-8477
E-mail: abran.alain@uqam.ca
Web site: http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

**Jean-Marc Desharnais M.Sc.A., C.F.P.S.**

Software Engineering Laboratory in Applied Metrics
7415 Beaubien East suite 509
Anjou (Quebec) Canada H1M 3R5

Telephone: 514 - 355-2872
Fax: 514 - 355-3600
E-mail: Desharnais.Jean-Marc@uqam.ca
Web site: http://www.lmagl.qc.ca

**Marcela Maya M.Sc.A., C.F.P.S.**

Software Engineering Management Research Laboratory
Departement d'informatique
Universite du Quebec a Montreal
C.p. 8888 succursale centre-ville
Montreal (Quebec) Canada H3C 3P8

Telephone: 514 - 987-3000 (6647)
Fax: 514 - 987-8477
E-mail: Maya.Marcela@uqam.ca
Web site: http://saturne.info.uqam.ca/Labo_Recherche/lrgl.html

**Denis St-Pierre M.Sc., C.F.P.S.**

Software Engineering Laboratory in Applied Metrics
7415 Beaubien East suite 509
Anjou (Quebec) Canada H1M 3R5

Telephone: 514 - 355-2872
Fax: 514 - 355-3600
E-mail: Denis.St-Pierre@CRIM.CA
Web site: http://www.lmagl.qc.ca