# 第 2 章

# 関連研究調査結果

## 2.1 オブジェクト指向のメトリクス

オブジェクトの進化メトリクスを決定するに当たり，これまでのオブジェクト指向メトリクスの研究について調査を行った.

### 2.1.1 Chidamber & Kemerer のメトリクス

**概要**

Chidamber & Kemerer が提唱したオブジェクト指向設計メトリクス（以下 CK メトリクス）は，次の６種類に分類される [4].

- WMC（Weigted Methods per Class）

  各クラスについて，クラス内のメソッドの複雑度によって重みづけられたメソッドの値を合計したクラスの複雑度を表わすメトリクス．ただし，メソッドの複雑度を計測する方法については議論していない.

  Henderson-Sellers はメソッドの複雑度を表現するメトリクスについて，McCabe の cyclomatic メトリクス，Henry と Kafura の fan-in/fan-out メトリクスをオブジェクト指向に適用するための議論を行った [9].

- CBO（Coupling Between Objects）

  あるクラスが関連を持つクラスの総数でクラス間の結合度を表わすメトリクス.

- LCOM（Lack of Cohesion of Methods）

  クラスに定義されている各メソッドについて，メソッドが参照しているインスタンス変数の集合を求め，メソッド間で，インスタンス変数の集合の交わりを求める．このメトリクスは，この交わりの要素の数で，これをクラスの凝集度と定義した．もし，LCOM の値が小さければ，そのクラスの凝集度は低く分割可能であることを表わす.

- NOC（Number of Children）

  注目するクラスの直下に定義されているサブクラスの数．クラスの変更のしやすさを表わすメトリクス．この値が大きいということは，そのクラスを変更したときに，その影響を受けるクラスが多いと判断される.

- DIT（Deapth of Inheritance Tree）

  継承の深さ．注目するクラスを理解するために理解しなければならないクラスの数で，クラスの可読性を表わすと共に，継承によって，他のクラスの仕様変更の影響を直接受ける可能性の大きさを表わす．DIT の値が大きいということは，継承しているクラスの仕様変更の影響を受ける可能性が高く，可読性も低いことを意味している．

- RFC（Response for Class）

  そのクラスに応答するメッセージ数．RFC は，注目しているクラスが呼び出しているメッセージの数と自分自身に定義してあるメッセージの数の和で，クラスの可読性を表わす．

CK メトリクスは，クラスの理解容易性，可読性，変更のしやすさを定量化することを目的に定義された．

#### 本研究における位置付け

我々は，CK メトリクスを，進化メトリクスの基本とした．ただし，計測対象に Smalltalk で開発されたシステムを選択したので，オブジェクト間の関連をソースコードから計測することはできない．そこで，プログラム実行時にオブジェクト間の関連の状態を計測する動的メトリクスを定義した．

### 2.1.2 Lorenz & Kidd の設計メトリクス

#### 概要

Lorenz & Kidd が提案した設計メトリクスは，メソッドの大きさ，メソッドの内部構造，クラスのサイズ，クラス継承，メソッド継承，クラス構造，クラス間の関係といった観点で分類されている．各々の観点で定義されたメトリクスは次のとおりである [13].

- メソッドの大きさに関するメトリクス
  - メソッドから送られるメッセージの数
  - メソッドの内の文の数
  - メソッドの内の行数
  - プロジェクトで定義された全メソッドの平均値
- メソッドの内部構造に関するメトリクス
  - 各文に重みづけを行った合計値（メソッドの複雑度）
  - カスケードメッセージの長さ（Smalltalk 固有）
- クラスの大きさに関するメトリクス
  - 1 クラスに定義されているインスタンスメソッドの数
  - 1 クラスに定義されているインスタンス変数の数
  - 1 クラスに定義されているクラスメソッドの数
  - 1 クラスに定義されているクラス変数の数
  - 1 プロジェクトにおける 1 クラスに定義されているインスタンスメソッドの数の平均値
  - 1 プロジェクトにおける 1 クラスに定義されているインスタンス変数の数の平均値
- クラス継承に関するメトリクス

- クラス階層の深さ
- 抽象クラスの数
- 多重継承しているクラスの数

- メソッド継承に関するメトリクス
  - サブクラスで再定義されているメソッドの数
  - サブクラスで継承されているメソッドの数
  - サブクラスで追加されているメソッドの数

$$\frac{\text{再定義したメソッドの数} * \text{クラス階層の深さ}}{\text{メソッド数}}$$

- クラスの内部構造に関するメトリクス
  - 広域変数の数
  - クラス変数の数
  - プール辞書の数
  - 手続き指向によるコードの占有率
  - コメントを持っているメソッドの数の全メソッド数に対する割合
  - １クラス，あるいは１メッセージ送受信当たりのエラー報告件数
  - １メソッド当たりの引き数の数のプロジェクト内の平均
  - １メソッド当たりのコメント行数のプロジェクト内の平均値

- クラス間の関係に関するメトリクス
  - クラス結合度
  - クラスが再利用された回数
  - 捨てられたクラスやメソッドの数

以上のメトリクスについて，Smalltalk および C++を用いて開発されたシステムの計測値で，標準的な値を示すと共に，これらのメトリクスを用いて計測を行った結果の意味，対処を与えている．

#### 本研究における位置付け

システムの開発には，各々固有の状況から受ける制約が存在する．Lorenz らは，一般的な平均値から外れた値が得られたときにはメンターの指導を受けることを提案している．しかし，システム開発過程の計測結果を時系列に並べ，相互に比較すれば個々のシステムの状況を相殺することができるので，システムの開発状況に応じた設計方針の検討を行うためには，システム毎の開発経過に従って変化する計測値を観察することも重要であろう．

### 2.1.3 Basili らによる品質評価のためのメトリクス

#### 概要

システムの開発時および保守時のエラーの数と CK メトリクスに準拠したメトリクスを用いて計測した値との相関関係を求め，各計測値とソフトウェアの品質の関係について議論した．計測にあたり，Basili らは，CK メトリクスについて独自の解釈を与えた．たとえば，WMC について，各メソッドの重みづけは恣意的に決めるのは適切ではないので，重みづけは行わず，WMC を１クラス当たりのメソッド数として解釈した．彼等はソフトウェアの品質とメトリクスとの関係について，次のような仮説を提起した [1].

1. H-WMC: 他のクラスに比べてメソッド数が多いクラスは，他のクラスより複雑であることを意味している．したがって，より多くのエラーを含んでいる可能性が高い．
2. H-DIT: クラス階層中，深い位置にあるクラスは，多くの定義を祖先から継承しているので，浅い位置にあるクラスより多くのエラーを含んでいる可能性が高い．
3. H-NOC: 多くの直下のサブクラスをもっているクラスは変更しづらく，より多くのテストが必要であるのが普通である．また，多くの子供クラスをもっているクラスは，様々なコンテキストでサービスを提供しなければならないこともあるので，より柔軟でなければならない．このような状態はクラス設計に複雑さをもたらすので，H-NOC が大きいクラスは，小さいクラスより多くのエラーを含んでいる可能性が高い．
4. H-CBO: 高い結合度を持っているクラスは弱い結合度を持っているクラスより，他のクラスに定義されているメソッドやオブジェクトに多くを依存しているので，より多くのエラーを含んでいる可能性が高い．
5. H-RFC: 大きな応答集合を持っているクラスは，より複雑な機能を実装しているので，より多くのエラーを含んでいる可能性が高い．
6. H-LCOM: メソッド間の凝集度が低いクラスは，不適切な設計がなされていることを表わしているので，より多くのエラーを含んでいる可能性が高い．

検証実験では，オブジェクト指向の基礎教育を受けた学生を，3 人ずつ 8 つグループに分けてサンプルシステムを開発した．この開発で，テスト時に発見されたエラーの数と上記のメトリクスを用いた計測値を収集し，これらの値の間の相関分析を行って仮説の検証を行った．

#### 本研究における位置付け

システムやシステムを構成するクラスの進化過程を定量化することによって，初期に設計された内容が，進化によってどのように変化するかを明らかにすることを目指している．Bassili らの研究成果から，CK メトリクスを適用して計測したあるクラスの値が，時系列で増大するとき，そのクラスに内在するエラーの可能性が増大していく傾向も表わしていると考えることができる．

### 2.1.4 Sharble & Cohen のメトリクス

#### 概要

Sharble & Cohen は，データ駆動型 [5, 3, 15] と振る舞い駆動型 [7, 22] の方法論を用いて開発された成果物に CK メトリクスを拡張したメトリクスを適用して計測し，方法論によるクラスやメソッドの複雑度の違いを比較した [17].

Sharble らは CK メトリクスに次のメトリクスを追加した．

- WAC（Weighted Attributes per Class）：各クラスに定義されている属性に重みづけを行い，その値をクラス毎に合計した値
- NOT（Number of Tramps）：メソッドに与えられるパラメータのうち，メソッド本体で参照されないパラメータの数
- VOD（Violations of the Law of Demeter）：インスタンス変数，メソッドの引き数，クラス内で生成されたオブジェクトが属しているクラスの数．CBO と似ているが，CBO がクラス間の相互作用を計測しているメトリクスであるのに対して，VOD は複合クラスに結合しているクラスの数を計測するためのメトリクスである．

以上のメトリクスと CK メトリクスはクラスの複雑度や可読性を表現するためのメトリクスである．複雑度は，クラスの複雑度（RFC, LCOM, WAC, WMC），クラス間の相互作用における複雑度（RFC, VOD, CBO），継承によるクラス間の関連の複雑度（DIT, NOC）に分けらる．一般に，結合度を低く，凝集度を高くすることによって複雑度を小さくできると言われているが，各メトリクスについて，結合度は VOD, CBO で計測し，凝集度は LCOM によって計測するとした．たとえば，高い VOD の値は，複合クラスがシステム内で相互に依存しあっていることを意味しているので，変更が及ぶ可能性のある範囲が広いと理解する．また，継承によるクラス間の関連の複雑度は，DIT，NOC 共に値が大きいほうが複雑度は高いと評価する．

さらに，良くカプセル化されたオブジェクトとは，自分自身にとって自然でないサービスを実行しないこと，そして，自分にとって自然な振る舞いを実行するときに必要な基本的な情報を他のオブジェクトに問い合わせたりしないオブジェクトであると定義した．したがって，良くカプセル化されたオブジェクトは，その CBO の値が小さく，LCOM の値も小さくなければならないと結論づけた．

以上の評価基準に基づいて，2 種類の方法論によって定義されたモデルを比較した結果，振る舞い駆動型の方法論を用いた方が，複雑度が低く，凝集度も高い，良くカプセル化されたクラスを開発できたと報告している．

#### 本研究における位置付け

我々は，開発に用いた方法論については調査を行っていないが，今後は，方法論によるオブジェクトの進化の差を議論する必要もあるだろう．

## 2.2 Belady & Lehman の進化法則

### 概要

Belady & Lehman は，IBM360 開発において，システムの進化について研究し，その成果をまとめた．彼等がまとめた 5 つの進化法則は次のとおり [11].

1. プログラムの継続的な変化

   プログラムが使われると，現実世界に影響を与えるようになり，変化し続け，その結果，徐々にプログラムの有効性を低下させる．このような傾向は，より費用対効果が大きい効率的なシステムによって置き換えられるまで継続する．

2. 複雑度の増加

   進化するプログラムでは，構造の劣化を反映させる複雑度を増加させないような保守を行わない限り，その複雑度は増加する．

3. 統計的な制御可能性

   プロジェクト全体の属性やシステムの属性を計測することによって，統計的に判断可能な傾向や不変性によって，プログラミング過程を自己制御できるようになる．

### 本研究における位置付け

我々は，オブジェクト指向システムの進化過程を調査することによって，Belady と Lehman が定義した進化法則がオブジェクト指向システムにも当てはまるか，また，当てはまらないとしたらどのような現象が開発時に起きているのかを明らかにすることを目指す．