抽象化に基づくリモートコンポーネントベースシステムの複雑度測定

中川知基† 鷲崎弘宜‡ 斉藤勇樹† 深澤良彰†

†早稲田大学大学院理工学研究科
‡国立情報学研究所

ソフトウェアテストシンポジウム2005

研究概要

- **目的**
  コンポーネントベースシステム（CBS）の保守性を測る方法として、CBSの特徴を考慮した複雑度を測定する手法を提案
- **方法**
  CBSのクラス集合を段階的に抽象化することにより、コンポーネント間の関連性から複雑度を測定
- **結果**
  実験により、従来の測定法にくらべ、本手法ではよりコンポーネントレベルでの保守性を反映することを確認

コンポーネントベース開発

コンポーネントベース開発の達成すべきことは？

コンポーネント合を組み合わせることで、新しい高品質・大規模・多様なソフトウェアを迅速に開発すること

- **コンポーネント開発への取り組み**
  - 基盤となるコンポーネントアーキテクチャの決定
    - JavaBeans、ActiveX、EnterpriseJavaBensなど
  - アーキテクチャの規格に従い、動作可能なソフトウェア部品としてコンポーネントを再利用／新規開発

コンポーネントアーキテクチャ

コンポーネントを組み合わせるための規格と、様々な調整サービスを提供する基盤環境

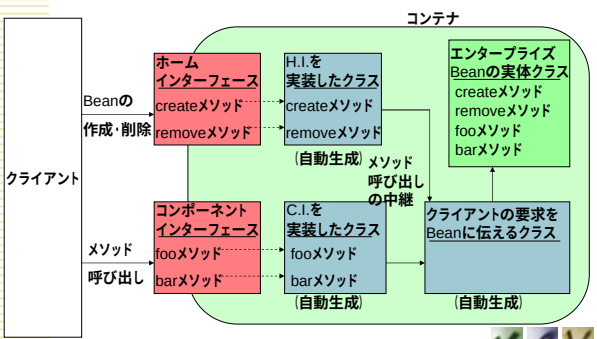
- **ローカルコンポーネントアーキテクチャ**
  - コンポーネントが、コンポーネントの実体が存在する環境と同一環境上に存在する外部のソフトウェアから利用される
    - OLE/COM, JavaBeans, IntelligentPad
    - GUI部品などの再利用
- **リモートコンポーネントアーキテクチャ**
  - コンポーネント、コンポーネントの実体が存在する環境とはネットワークを介して異なる環境上に存在する外部のソフトウェアから利用される
    - DCOM/ActiveX, Enterprise JavaBeans (EJB), CORBA/IDL, CORBA Component Model (CCM)/CIDL
    - ビジネスロジック部品などの再利用

EnterpriseJavaBeans（EJB）

ビジネスロジックとビジネスエンティティをカプセル化するためのリモートコンポーネントアーキテクチャ

- **EJBにおけるコンポーネント**
  - エンタープライズBean
    - エンタープライズBeanの実体クラス
    - ホームインターフェース
    - コンポーネントインターフェース
- **EJBコンテナ内でエンタープライズBeanを組み合わせて使用**
  - コンテナがトランザクションなどの機能を自動追加

EJBの仕組み



保守性

仕様変更時や障害修正時の必要な労力を現す品質特性

- **システムの長期運用を考えるとき、高い保守性を兼ね備える必要がある**
- **保守作業にかかるコストは、全体の50%-75%を占める**
  - *B.W. Boehm, Software Engineering Economics, 1981*
- **保守性を測定する方法**
  - 複雑度測定法
- **従来の測定法**
  - 結合度測定法としてCOF(Coupling Factor)
  - オブジェクト指向クラス間の関係に注目

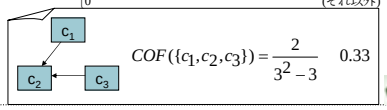
オブジェクト指向複雑度測定法

- **COF(Coupling Factor)**

$$COF(S)=\begin{cases}\dfrac{\sum_{i=1}^{|S|}\sum_{j=1}^{|S|}isClient(c_i,c_j)}{|S|^2-|S|-(2\sum_{k=1}^{|S|}|Descendents(c_k)|)} & (|S|\neq 0,1)\\ 0 & (|S|=0,1)\end{cases}$$

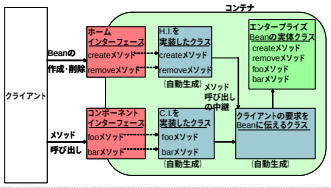
S＝{c1,…,c|S|}は、測定対象システム
cは、システムを構成するクラス
Descendents(c)は、cのサブクラス数

$$isClient(c_1,c_2)=\begin{cases}1 & (c_1,c_2\text{が継承関係になく、}c_1\neq c_2\text{かつ}c_1\text{が}c_2\text{を参照})\\ 0 & (\text{それ以外})\end{cases}$$



オブジェクト指向複雑度測定法の欠点

- **COFは、オブジェクト指向クラスの関連をそのまま測定対象としている**
  → **コンポーネントを単位とした保守性を反映しない**
- **例) EJBの場合**
  - 自動生成されるクラスには、プログラミングレベルでは関連がない



コンポーネントベース複雑度測定法

CBSの特徴を考慮した複雑度を測定する手法を提案

- **CCOF(Component Coupling Factor)**
  - COFの仕組みを発展させ、オブジェクト指向クラス集合を抽象化し、その抽象化された構造に対して、複雑度を測定

$$CCOF(S)=\begin{cases}\dfrac{\sum_{i=1}^{|S|}\sum_{j=1}^{|S|}hasReferences(B_i,B_j)}{|S|^2-|S|} & (|S|\neq 0,1)\\ 0 & (|S|=0,1)\end{cases}$$

Sは、測定対象システム
Bは、１つのエンタープライズBeanもしくは同程度の抽象化クラス

$$hasReferences(B_1,B_2)=\begin{cases}1 & (B_1\neq B_2\text{かつ}B_1\text{が}B_2\text{を参照})\\ 0 & (\text{それ以外})\end{cases}$$

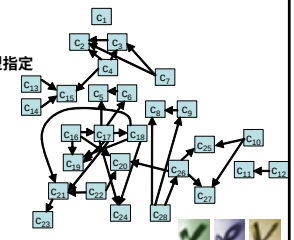
システムの抽象化

- **以下の3つの手順でシステムを抽象化**

1. クラス関連グラフの作成
2. コンポーネントの特定
3. コンポーネント以外のクラス部分集合の抽象化

1. クラス関連グラフの作成

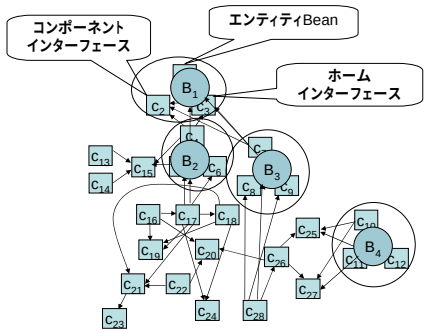
- **コンポーネントベースシステムのクラス集合のクラス関連グラフ (CRG)**
  - 頂点Vには、
    - クラスおよびインターフェース
  - 辺Eには、(有向)
    - フィールドの参照
    - メソッド参照
    - クラス･インターフェースの型指定



2.コンポーネントの特定

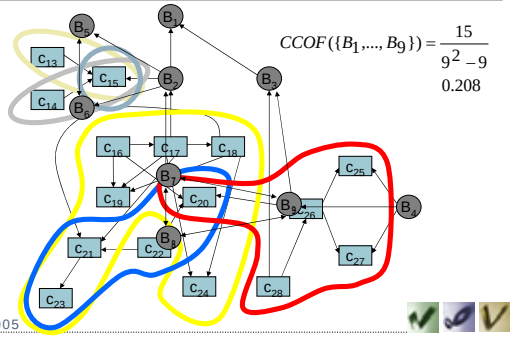
- 以下を1つのコンポーネントにまとめる
  - エンタープライズBeanの実装クラス
  - コンポーネントインターフェース
  - ホームインターフェース
- これらへの外部からの参照は、抽象化後の抽象化クラスへの参照へ対応付ける

2.コンポーネントの特定



3.コンポーネント以外のクラス部分集合の抽象化

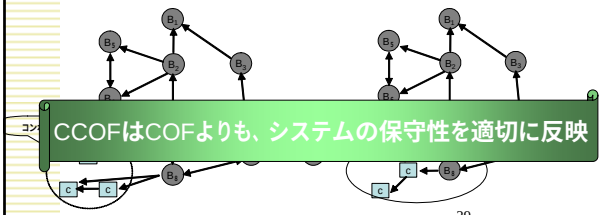
参照／継承到達可能な最大集合を求める



$$CCOF(\{B_1,...,B_9\}) = \frac{15}{9^2-9} \quad 0.208$$

機能追加後の保守作業の考察

安定的な再利用性の高いコンポーネントを追加　　軽微／補助的な機能を追加



CCOFはCOFよりも、システムの保守性を適切に反映

$$COF(S') = \frac{39}{31^2-31} \quad 0.043 \qquad CCOF(S') = \frac{16}{10^2-10} \quad 0.178$$

$$COF(S'') = \frac{39}{31^2-31} \quad 0.043 \qquad CCOF(S'') = \frac{15}{9^2-9} \quad 0.208$$

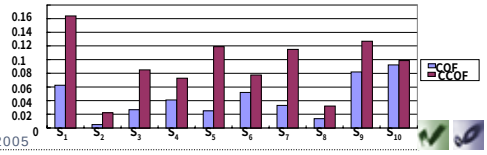
実験

- 実際使われているWeb上・書籍上のEJBアプリケーション（S_1～S_{10}）に、COF/CCOFを測定

ID	プロジェクト名	抽象化前のクラス数	抽象化後のクラス数
S_1	Financial Brokerage Service	72	11
S_2	Financial System Analysis	72	39
S_3	ITracker	198	25
S_4	Roomba	82	17
S_5	Virtual Shopping Mall	172	23
S_6	WebStore	89	10
S_7	xPetStore	110	13
S_8	PetStore	326	37
S_9	Book	52	11
S_{10}	C3J	45	12

考察

- 値大小の逆転：COF(S)>COF(S')∧CCOF(S)<CCOF(S')
  - 逆転が見られたシステムの対は13個存在した
- 例: S1とS9、COF(S1) < COF(S9) ∧ CCOF(S1) > CCOF(S9)
  - クラス数: 72と52、抽象化クラス数: 11と11
  - クラス数が多いためクラスレベルではS9がより複雑
  - 抽象化クラス数は同一となり、コンポーネントレベルではS1がより複雑
  - つまりS9は、コンポーネント内が高凝集、全体して祖結合
  - レビューの結果、S9はコンポーネントを単位としてより保守が容易
- CCOF はコンポーネントを単位とした保守作業の容易さを反映



CCOFの有効性の検証1

- **結合度に基づく測定法が正当であることの検証**
- **BMB Frameworkの適用[Briand, 1997]**
  - 正規性：0≦CCOF(S)≦1
  - 非負性：CCOF(S)≧0
  - ヌル値：S=φ ⇒ CCOF(S)=0
  - 単調性：S⊆S' ⇒ CCOF(S)≦CCOF(S')
  - クラスの併合：
    2つのコンポーネントを1つにまとめたとき、CCOF(S)≧CCOF(S')
  - 無関連クラスの併合：
    無関連のコンポーネントを1つにまとめたとき、CCOF(S)=CCOF(S')

6条件のうち5つを満たす

CCOFの有効性の検証2

- **主成分分析・相関分析による定量的検証法**
  - 実験結果に対し、主成分分析を施し累積寄与率を計算

主成分	累積寄与率
第1主成分	0.805
第2主成分	1.000

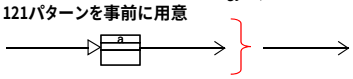
    - 結果: 第1主成分の累積寄与率<0.85より、実験データは2次元の広がりを持つ

主成分	COF	CCOF
COF	1	0.610
CCOF	0.610	1

  - 両測定法の測定値間の相関係数を計算
    - 結果: 相関係数<0.8より、強い正の相関は確認できない
  - CCOFは、COFに対して冗長ではない

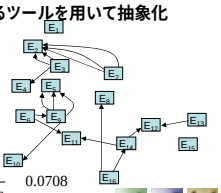
関連研究

- **様々な抽象化手法**
  - パターンに基づくクラス図抽象化 [Egyed, 2002]
    - 121パターンを事前に用意



  - 先のサンプルに対し、公開されているツールを用いて抽象化

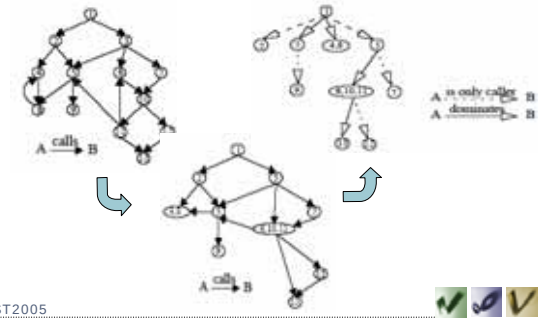
→リモートコンポーネントアーキテクチャの特徴を考慮していないので、保守性を適切に反映しない



$$CCOF(\{E_1,\ldots,E_{16}\}) = \frac{17}{16^2-16} \quad 0.0708$$

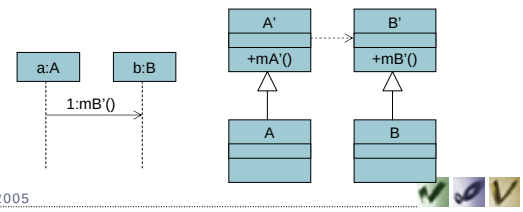
関連研究

- **支配解析に基づくクラスタリング [Girard, 1997]**



動的な結合度測定法

- オブジェクト間の相互作用の数に基づくオブジェクトとクラス間の動的結合度の強さを捉える測定法[Erik Arisholm 2003]
- アプリケーションクラスのインスタンスの振る舞いから見たライブラリ等の外部リソースとの動的結合度を測る測定法



まとめ

- **リモートコンポーネントシステムの複雑度測定法CCOFを提案した**
  - コンポーネントの特定と抽象化
  - 結合度測定法としての必要条件を満たすことを確認
  - 冗長ではないことを確認
- **本研究の貢献**
  - コンポーネントをシステムの構成単位とした保守作業の容易さを反映することが可能である
  - 他のシステムのCCOF値との大小関係からコンポーネントレベルでのシステムの複雑度を比較することができる
- **今後の課題**
  - 他の抽象化手法との比較
  - 大規模な測定実験