情報処理学会論文誌 Vol.58 No.4 795–806 (Apr. 2017)

クラス責務割当ての ファジィ制約充足問題としての定式化

林 晋平[1,a)] 柳田 拓人[2,†1] 佐伯 元司[1] 三村 秀典[2]

受付日 2016年8月10日, 採録日 2017年1月10日

概要：本論文ではクラスへの責務割当てをファジィ制約充足問題として定式化することにより自動化を図る手法を提案する．責務とは各クラスのインスタンスが果たすべき役割を指し，それらをクラスへ適切に割り当てることによって，高品質な設計が実現される．割当てに際しては，疎結合かつ高凝集な割当てが望ましいなど，様々な観点を考慮することが必要となる．しかし，そのような観点の間にはトレード・オフがあるため，現実的には条件を適度に満たす割当てが求められ，計算機による支援が重要である．そこで，様々な条件をファジィ制約として表現し，責務割当てをファジィ制約充足問題として定式化する．これによって，既存の汎用的なアルゴリズムを適用できるようになり，解としての責務割当ての導出が可能となる．例題に対して解を導出した結果を示す．

キーワード：責務割当て，ソフトウェア設計，ファジィ制約充足問題

Formalizing Class Responsibility Assignment as Fuzzy Constraint Satisfaction Problem

Shinpei Hayashi[1,a)] Takuto Yanagida[2,†1] Motoshi Saeki[1] Hidenori Mimura[2]

Received: August 10, 2016, Accepted: January 10, 2017

Abstract: The authors formulate the class responsibility assignment (CRA) problem as the fuzzy constraint satisfaction problem (FCSP) to automate CRA, and show the results of automatic assignments of examples. Responsibilities are contracts or obligations of objects that they should assume; by aligning them to classes appropriately, designs of high quality realize. Typical aspects of a desirable design are having a low coupling between highly cohesive classes. However, because of a trade-off among such aspects, solutions that satisfy the conditions moderately are desired, and computer assistance is needed. The authors represent the conditions of such aspects as fuzzy constraints, and formulate CRA as FCSP. That enables us to apply common algorithms that solve FCSP to the problem, and to derive solutions representing a CRA.

Keywords: class responsibility assignment, software design, fuzzy constraint satisfaction problems

1. はじめに

オブジェクト指向開発において高品質な設計を実現するために重要な作業として，クラスへの責務割当て（Class Responsibility Assignment; CRA）がある．責務（Responsibility）とは，各クラスのインスタンスが果たすべき役割であり [2]，責務割当てとは抽出された責務が所属するクラスを決定する作業を指す． 設計の際に適切な責務割当てを行うことにより，明確性や保守性，拡張性などに優れた高品質な設計が実現できる．たとえば，互いに関連のある責務を 1 カ所にまとめて割り当てることにより，それらの責

1 東京工業大学情報理工学院
School of Computing, Tokyo Institute of Technology, Meguro, Tokyo 152–8552, Japan
2 静岡大学電子工学研究所
Research Institute of Electronics, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
†1 現在，株式会社スペースタイム
Presently with Space-Time Inc.
a) hayashi@c.titech.ac.jp

本論文は，IWESEP 2014 での我々の発表 [1] の内容を発展させたものである．

© 2017 Information Processing Society of Japan

務に変更が生じた際にプログラムの他の部分に影響が波及することを防ぐことができる．

責務割当てのための方法論や様々な指針が提案されているものの，課題も残っている．たとえば，Wirfs-Brock らの責務駆動アプローチ（Responsibility Driven Approach）[2] などの手法がある．また，責務抽出をもとにクラス設計を行う CRC カード法 [3] も提案されており，実用にも付されている [4], [5]．これらの手法を用いることで，クラスとそのクラスが果たすべき責務を抽出することができるものの，設計の質の観点からは不十分であり，得られた結果をそのまま設計に利用するわけにはいかない．ここで，典型的な設計の質的観点は，疎結合かつ高凝集，すなわち，クラス間をまたがる責務の依存関係が少なく，かつ同一クラス内の責務が協調し合っているような割当てを好ましいとするものである [6]．CRC カード法は，人間の認知過程や能力に大きく依存しており，開発者の直感的な理解には適しているものの，変更容易性などの保守性において最適とはいえない．これらの手法で，クラス，責務を抽出する最中や抽出した後に，高品質の設計になるように既存の割当てを書き換えたり，議論の開始点とするための設計の叩き台を導出したりすることが望ましく，そのためには責務割当ての自動化が重要である．

自動化においては，様々な状況の考慮が必要とされる．実際，責務割当ては過制約な（over-constrained）問題である．前述した疎結合や高凝集といった責務割当ての観点にはトレード・オフがあり，多くの場合，一方の向上が他方の低下を引き起こす．そのため，その両方をよく満たす設計を得ることは簡単ではなく，現実的には設計の条件を適度に満たす解の導出が求められる．また，設計時には設計者による様々な試行錯誤が発生するため，割当てを繰り返し行えるよう十分な速度であること，また途中までの割当て結果を尊重しながら後続の割当てを行うことも求められる．このように，責務の質を示す観点には様々なものがあり，現実的にはこれら複数の観点を混在させたり，切り替えて利用したりするなど高い柔軟性も求められる．

本論文では，責務割当ての自動化として，ファジィ制約充足問題（Fuzzy Constraint Satisfaction Problem; FCSP）への帰着を試みる．提案手法では，結合性や凝集性をはじめとした様々な条件をファジィ制約として表現し，割当て問題を FCSP として定式化することによって，モノリシックな評価関数を導入することなく，各種の設計の質的観点や既存の割当ての維持といったあいまいなルールを自然に表現することが可能となる．そして，既存の汎用的な FCSP ソルバを適用し，解として責務割当てを求めることが可能となる．また，今後，FCSP の様々な知見を広く応用することが可能となる．提案手法に既存の FCSP ソルバを適用し，いくつかの例題に適用したところ，提案手法によって高品質な責務割当てが実現できる可能性が示唆された．

本論文の貢献は以下のとおりである．

(1) 責務割当てについて，その支援ツールを構築するうえでの観点を整理したこと．特に，責務割当ての自動化において従来考慮されてきた質的観点だけでなく，設計維持のための安定性（stability）や開発者の持つ意図（intention）の反映が重要である点を明らかにしたこと．
(2) 上記の観点を考慮した責務割当てに対して，FCSP による定式化を初めて与えたこと．
(3) FCSP の新しい応用を示したこと．本論文は，ソフトウェア工学に対する人工知能，特に FCSP の新たな応用例を示すものである．
(4) 例題への適用により，提案手法を設計支援ツールに組み込むことの有用性および実現可能性の分析を行ったこと．

以降の本論文の構成は次のとおりである．まず，2 章で準備として責務割当ておよびファジィ制約充足問題について説明する．次に 3 章で FCSP による責務割当ての定式化を示す．例題に適用した結果を 4 章で示し，また行った考察を 5 章にまとめる．関連研究を 6 章で示し，最後に 7 章でまとめと今後の課題を述べる．

2. 準備

他の文献にも詳しく，よく知られたことであるが，後の説明に必要となるので，本章で責務割当ておよびファジィ制約充足問題の諸概念を簡潔に導入しておく．

2.1 責務割当て

責務（Responsibility）とは，各クラスのインスタンスが果たすべき役割であり [2]，それらが行う動作（動作責務；Doing Responsibility）と持つ知識（情報把握責務；Knowing Responsibility）の 2 つに大きく分類される [6]．責務割当てとは，あらかじめ抽出しておいた責務が所属するクラスを決定する作業を指す．以降，責務を $m \in M$ で，責務の割当て対象となるクラスを $k \in K$ で表す．単純に述べれば，責務割当ては責務の集合 M からクラスの集合への写像 $A : M \to K$ を決定することに相当する．ただし，現実的には，割当て作業中に新しいクラスを作成したり，既存の割当てを修正したりする試行錯誤を経て割当てが完了することに注意されたい．

2.2 ファジィ制約充足問題

ファジィ制約充足問題（FCSP）は，変数の有限集合 X，個々の変数に対応するドメインの有限集合 $D = \{ D_x \mid x \in X \}$，制約の有限集合 C から構成される [7]．制約 $c \in C$ は変数の部分集合 $S_c \subseteq X$ におけるファジィ関係 μR_c によって表される．S_c を μR_c のスコープという．S_c の要素数 $|S_c|$ が

© 2017 Information Processing Society of Japan

表 1 疎結合性，高凝集性の定式化の方針
Table 1 Formalization strategy for low coupling and high cohesion.

性質	疎結合性	高凝集性
着目する関係	異なるクラスに所属する責務間の関係	同一クラス内の責務間の関係
	↓	↓
注目する達成阻害要因	結合：クラス外の責務との強い関連	凝集性の欠如：クラス内の無関係な責務の存在
	↓	↓
阻害要因の FCSP での表現	関連性の高い責務が遠いクラスに存在	関連性の低い責務が近いクラスに存在

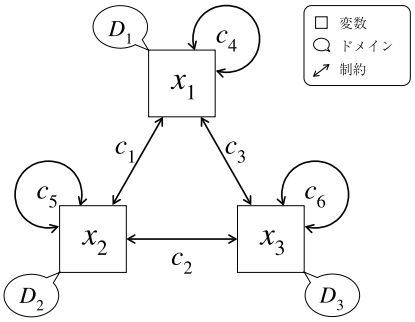


図 1 ファジィ制約充足問題の模式例
Fig. 1 Illustrative example of an FCSP.

1 もしくは 2 のとき，この関係をそれぞれ単項関係，2 項関係という．ファジィ関係 μR_c を，以下の形式で与えるメンバシップ関数と同一視する．

$$\mu R_c : \prod_{x \in S_c} D_x \to [0,1]$$

すなわち，メンバシップ値は制約 c のスコープ S_c における変数への割当て $v[S_c]$ によって定義される．これを c の制約充足度という．FCSP ではすべての制約充足度のファジィ論理積を FCSP 全体の充足度とする．すなわち，すべての制約の中で，最も低い充足度 $C_{\min}$ を FCSP の充足度とする．

$$C_{\min}(v) = \min_{c \in C} \mu R_c(v[S_c])$$

なお，以降本論文では簡略化のため，$v[S_c] = \{d_1, \ldots, d_{|S_c|}\}$ のとき，$\mu R_c(d_1, \ldots, d_{|S_c|})$ がメンバシップ値，すなわち制約充足度を表すと見なす．

図 1 に FCSP の模式例を示す．図中の四角は変数，吹き出しは変数に対応するドメイン，矢印は変数間の制約を表す．この例では，3 変数 $X = \{x_1, x_2, x_3\}$ それぞれに対してドメイン $D = \{D_1, D_2, D_3\}$ が対応しており，6 つの制約 $C = \{c_1, \ldots, c_6\}$ が定義されている．制約 c_1, c_2, c_3 は変数間の 2 項関係を規定するものとして，c_4, c_5, c_6 は各変数における単項関係を規定するものとして定義されている．

3. 提案手法

3.1 定式化の方針

責務割当てを FCSP として定式化するためには，良い割当ての指針をどのように制約として表現するかが重要となる．本論文では，以下で述べる条件をできるだけ満たすように，クラスに対して責務を割り当てるような定式化を試みる．

疎結合性と高凝集性の定式化方針を**表 1** に示す．本論文における定式化では，疎結合性の達成を阻害する要因として，結合，すなわち責務がクラス外の責務と強く関連している場合を考え，これが存在する場合に制約充足度を下げる．また，高凝集性の達成を阻害する要因として，凝集性の欠如，すなわち同一クラス内に無関係な責務が存在する場合を考え，これが存在する場合に制約充足度を下げる．

- **疎結合性**：関連性の高い責務が遠いクラスにできるだけ存在しない．
- **高凝集性**：関連性の低い責務が近いクラスにできるだけ存在しない．

また，たとえば設計の途中であるなど，一部の割当て結果がユーザ設計として与えられている際には，提案手法の出力がユーザ設計を逸脱しないことが好ましい．そこで，ユーザ設計からの距離ができるだけ近いことも責務割当ての条件に加える．

- **設計維持**：責務割当てにユーザ設計が与えられる場合，できるだけユーザ設計によるクラスからのクラス間距離の近いクラスへの変更にとどめたい．

さらに，たとえば同一のクラスに割り当てられるべき責務が複数のクラスに分散してしまったなど，提案手法の出力に不満がある場合，これらを制約として付加できるよう，特定の責務に対するクラス割当てのユーザの意図も扱う．

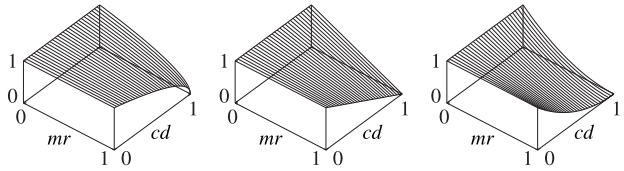
- **意図反映**：ユーザによる，同クラスに割り当てたい，もしくは異クラスに割り当てたいという意図をできるだけ反映させる．

クラス間の距離，ならびに責務の関連性は様々なメトリクスが考えられる．本論文では，用いたメトリクスの定義を 4 章で説明し，どのような既存のメトリクスを利用可能であるかについて 5 章で考察する．

3.2 定式化

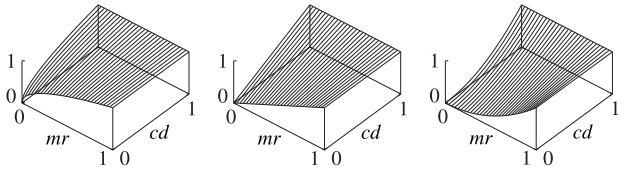
FCSP としての定式化は，責務にクラスを割り当てる問題と換言できる．すなわち，FCSP における変数を責務 $m \in M$ に，そのドメインを割当て対象のクラスの集合 K に，先に述べた責務割当ての指針をその変数間のファジィ

© 2017 Information Processing Society of Japan



(a) $w^{\mathrm{lc}} = 0.5$ (b) $w^{\mathrm{lc}} = 1$ (c) $w^{\mathrm{lc}} = 2$

図 2 制約充足度 $\mu R_{c^{\mathrm{lc}}}$ の分布
Fig. 2 Distribution of $\mu R_{c^{\mathrm{lc}}}$.



(a) $w^{\mathrm{hc}} = 0.5$ (b) $w^{\mathrm{hc}} = 1$ (c) $w^{\mathrm{hc}} = 2$

図 3 制約充足度 $\mu R_{c^{\mathrm{hc}}}$ の分布
Fig. 3 Distribution of $\mu R_{c^{\mathrm{hc}}}$.

制約に対応づける．したがって，責務数 n の責務割当て問題は，n 変数の FCSP として定式化される．

3.2.1 クラス間の距離と責務間の関連性

定式化の際には，次の形式で定義される，解こうとする問題領域内において正規化されたクラス間の距離を表す関数 cd と，同じく正規化された責務間の関連性を表す関数 mr を用いる．

$$cd : K^2 \to [0,1]$$
$$mr : M^2 \to [0,1]$$

任意のクラス $k_1, k_2 \in K$ に対して，k_1 と k_2 が同一であるとき，またそのときに限り $cd(k_1,k_2) = 0$ となるよう，また，k_1 と k_2 の距離が最も遠いとき，またそのときに限り $cd(k_1,k_2) = 1$ となるよう，cd を構成する．同様に，任意の責務 $m_1, m_2 \in M$ に対して，m_1 と m_2 にまったく関連性がないとき，またそのときに限り $mr(m_1,m_2) = 0$ となるよう，また，m_1 と m_2 の関連性が最も高いとき，またそのときに限り，$mr(m_1,m_2) = 1$ となるよう，mr を構成する．

これらの関数の値は提案手法の入力として，求解に先立ってあらかじめ計算しておくため，FCSP ソルバの実行中に値が変化することはない．しかし，クラス間の距離の情報源に既存のクラス構成を使用した場合など，責務割当て結果がこれらの定義に影響を与えることもある．こういった場合，割当ての更新後に値を計算し直し，ソルバを実行し直すことにより，値の更新に対応できる．割当て結果を開発者に推薦し，受け入れの可否を選択させるような支援ツールを想定すれば，推薦を受け入れて割当てが更新された際に，値の再計算とソルバの再実行を自動的に行うことにより，連鎖的な推薦が可能になる．

3.2.2 ファジィ制約

疎結合性を表現する制約 c^{lc} は，問題におけるすべての変数（責務）のペアに対して設けられる 2 項制約であり，責務間の関連性が高いときに，クラス間距離が大きくなると充足度が低下する．すなわち，関連する責務はできるだけ遠いクラスに配置されないようになる．2 責務 m_1，m_2 間の制約 $c^{\mathrm{lc}}_{m_1,m_2}$ の充足度 $\mu R_{c^{\mathrm{lc}}_{m_1,m_2}}$ は，割り当てられた 2 クラス間の距離から，次のように求める．

$$\mu R_{c^{\mathrm{lc}}_{m_1,m_2}}(k_1,k_2) = \{-mr(m_1,m_2) \cdot cd(k_1,k_2) + 1\}^{w^{\mathrm{lc}}}$$

ここで，w^{lc} は疎結合性に対する重みである．**図 2** に示すように，$w^{\mathrm{lc}} = 1$ のとき，充足度 $\mu R_{c^{\mathrm{lc}}}$ の分布はクラス間距離と責務の関連性に対して線形となる．

高凝集性を表現する制約 c^{hc} は，先ほどと同様に問題におけるすべての変数のペアに対して設けられる 2 項制約であり，責務間の関連性が低いときに，クラス間距離が小さいと充足度が低下する．すなわち，関連しない責務はできるだけ近いクラスに配置されないようになる．2 責務 m_1，m_2 間の制約 $c^{\mathrm{hc}}_{m_1,m_2}$ の充足度 $\mu R_{c^{\mathrm{hc}}_{m_1,m_2}}$ は，割り当てられた 2 クラス間の距離から，次のように求める．

$$\mu R_{c^{\mathrm{hc}}_{m_1,m_2}}(k_1,k_2) = [\{1 - mr(m_1,m_2)\} \cdot cd(k_1,k_2) + mr(m_1,m_2)]^{w^{\mathrm{hc}}}$$

ここで，w^{hc} は高凝集性に対する重みである．$\mu R_{c^{\mathrm{lc}}}$ 同様，**図 3** に示すように，$w^{\mathrm{hc}} = 1$ のとき，充足度 $\mu R_{c^{\mathrm{hc}}}$ の分布もクラス間距離と責務の関連性に対して線形となる．

設計からの少変更性を表現する制約 c^{s} は，各々の変数に対する単項制約として設けられ，設計として与えられたクラス割当てと変更されたクラスとの距離が遠くなると充足度が低下する．充足度 $\mu R_{c^{\mathrm{s}}_m}$ は設計におけるクラス割当てを k_{orig}，現在のクラス割当てを k としたとき，次のように求める．

$$\mu R_{c^{\mathrm{s}}_m}(k) = \{1 - cd(k_{\mathrm{orig}}, k)\}^{w^{\mathrm{s}}}$$

ここで，w^{s} は少変更性に対する重みである．

特定の責務を同一クラスに割り当てる，あるいは他クラスに割り当てるというユーザの意図は，それぞれ，c^{same} と c^{diff} という 2 項制約を対象となる責務のペアに対して設けることによって表現し，その充足度 $\mu R_{c^{\mathrm{same}}}$ と $\mu R_{c^{\mathrm{diff}}}$ は，割り当てられた 2 クラス間の距離 cd より，それぞれ次のように求める．

$$\mu R_{c^{\mathrm{same}}_{m_1,m_2}}(k_1,k_2) = \{1 - cd(k_1,k_2)\}^{w^{\mathrm{same}}}$$
$$\mu R_{c^{\mathrm{diff}}_{m_1,m_2}}(k_1,k_2) = cd(k_1,k_2)^{w^{\mathrm{diff}}}$$

ここで，w^{same} と w^{diff} はそれぞれ，同一クラス制約，他クラス制約に対する重みである．

図 4 に責務割当ての FCSP による表現例を示す．この例では，2 つのクラス `ClassA`，`ClassB` に対して，3 つの

© 2017 Information Processing Society of Japan

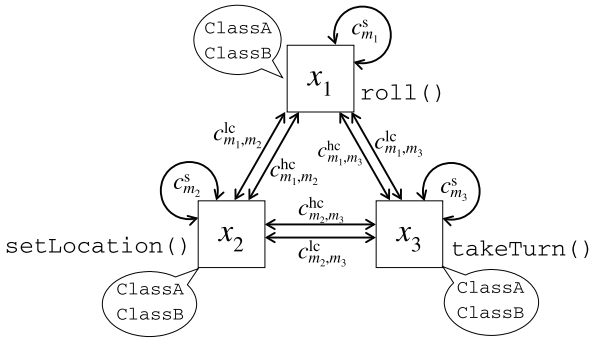


図 4 責務割当ての FCSP による表現例
Fig. 4 Example of an FCSP expressing a CRA.

責務 `roll()`，`setLocation()`，`takeTurn()` を割り当てようとしている．まず，各責務は変数 $X = \{x_1, x_2, x_3\}$ で表される．また，割当て先のクラスは，変数に対するドメインとして，全変数に共通で $D_i = \{\texttt{ClassA}, \texttt{ClassB}\}$ を対応付けることにより表現する．変数間には，疎結合性に関する制約 c^{lc} および高凝集性に関する制約 c^{hc} が 2 項関係として定義されている．また各変数に，少変更性に関する制約 c^{s} が単項関係として定義されている．この例では，ユーザの意図に関する制約は付与されていない．

各制約の定義の妥当性については 5 章でも議論する．

4. 実験による予備評価

4.1 評価項目

ゆくゆくは提案手法を設計ツールに組み込むための準備として，提案手法が有効に機能することを確認するため，大規模な例題への適用に先立ち，我々は複数の小規模な例題に対して予備評価を行った．本評価では，既存の例題を用い，以下の質問（evaluation question; EQ）に答えることを目的とする．

EQ1：割当て精度．提案手法の責務割当ての精度はどの程度か．特に，新しく割り当てる責務とすでに割り当てられた責務の割合により，精度がどのように変動するか．

EQ2：割当てに要する時間．提案手法が必要とする計算時間はどの程度か．特に，新しく割り当てる責務の数により計算時間はどのように変動するか．

EQ3：意図制約の利用．利用者の意図が追加で与えられた場合，提案手法が割当て結果を修正できるか．

EQ1 および EQ2 では，提案手法で割当て対象とする責務の数を変動させながら，割当ての精度や計算時間を計測する．開発者の設計作業の最中にインクリメンタルに割当ての支援を行う場合は，それまでの割当て結果を既知としながら追加の割当てを推薦することとなるため，少数を割り当てる場合に相当する．一方，スクラッチから全体を自動で割り当てて結果を分析したり，既存の結果の改善箇所を提案したりするような場合は，多数を割り当てる場合に相当する．EQ1 では，こういった異なる利用方法でも提案手法が機能する状況があるかを確認する．また，多数を割り当てる場合のほうが計算コストは高いため，EQ2 では，将来的に対話的なツールを構築するうえで，どれだけの割当て数までがリアルタイムの支援として許容可能な速度で行えるかを分析する．EQ3 では，新しい種類の制約を組み込み可能であることを調べるため，EQ1 の実験で正解の設計と食い違いの見られた責務に対して，それを修正するように意図を反映する制約を設定し，再度割り当てさせた際に，割当てが修正されるかを確認する．

表 2 評価対象
Table 2 Evaluation targets.

システム名	クラス数	動作責務数
Monopoly [6]	6	26
NextGen [6]	9	29
ARENA [8]	7	56

4.2 方法

4.2.1 対象

責務割当てを解説する書籍 [6] に掲載されている 2 つの例題，Monopoly と NextGen，および既存の責務割当て研究 [9] で使われた例題 ARENA [8] を評価に用いた．Monopoly はモノポリー・ゲーム，NextGen は POS システム，ARENA はトーナメント・ウェブ・システムのためのフレームワークの実装である．

Monopoly と NextGen については，書籍に掲載されていたクラス図に割り当てられているメソッドを動作責務と見なし，これらを割当ての対象とした．クラス図が表現している責務割当てをオリジナルの設計，すなわち正解と見なした．ARENA については，既存文献 [9] に掲載されているクラス図に表現されているメソッドを動作責務とし，そのうち公開されているソースコード中から発見できなかったなど不適切と判断したものを除いて利用した．

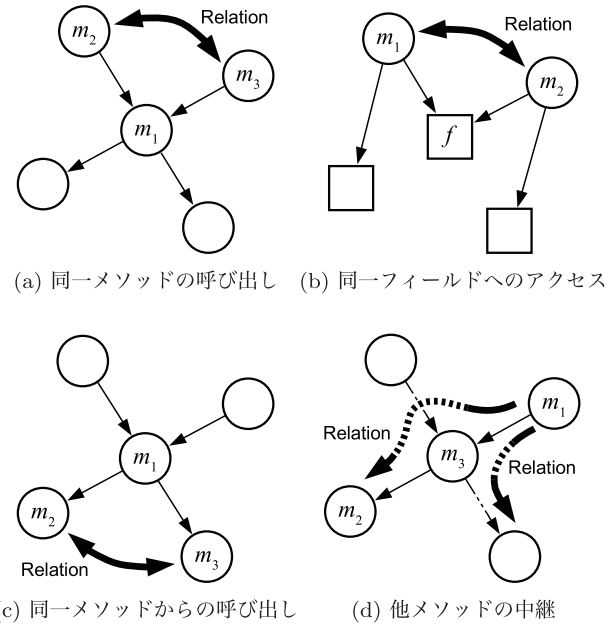
クラス間距離 cd および責務間の関連性 mr を計算するため，クラス図およびそのもとになったソースコードに基づき，クラス間の関係や責務間の関係を抽出した．ここで，責務間の関係については，メソッド間の呼び出し関係だけでなく，フィールドも情報把握責務を見なし，これへのアクセス関係も抽出した．これらを用いた cd と mr の計算については，次節で述べる．

各システムの規模を**表 2** に示す．いずれも現実的なシステムよりは小規模であるものの，責務割当ての困難さを引き起こす煩雑な関連性を備えた例題だと考えている．

4.2.2 実装

クラス間距離 cd および責務間の関連性 mr については，以下のように定義して用いた．まず，クラス間距離 cd は，クラス図において，クラスを節，関係を辺と見なしたとき

© 2017 Information Processing Society of Japan



(a) 同一メソッドの呼び出し (b) 同一フィールドへのアクセス

(c) 同一メソッドからの呼び出し (d) 他メソッドの中継

図 5 メソッド間の関連性．丸ノードはメソッド，四角ノードはフィールド，矢印はそれぞれに対するアクセスを表す

Fig. 5 Relations between methods. Circle nodes, square nodes, and arrows respectively represent methods, fields, and their accesses.

の 2 節間の最短経路長を，区間 $[0,1]$ に線形正規化したものを用いた．

また，責務間の関連性 mr については，抽出した責務間の関係から以下のように求めた．

(1) メソッド間の呼び出し関係をその呼び出し数によって隣接行列として表現し，それを転置したものを加算することにより，呼び出し関係と被呼び出し関係を同一視する．また，同様にメソッドからフィールドへのアクセス回数もメソッドとフィールドとの関連性として記録しておく．

(2) 同一メソッドを呼び出すメソッド間には関係があるとし，該当の呼び出しにおける関連性の平均値に重み α を乗じたものを，同一メソッドを呼び出すメソッド間の相互の関連性の値に加算する．**図 5** (a) の例では，m_2，m_3 が m_1 を共通に呼び出しているので，m_2，m_3 の相互の関連性の値に $\alpha\{mr'(m_2,m_1)+mr'(m_3,m_1)\}/2$ を加える．ここで，mr' は直前の手順までに得られた関連性の値を表す．

(3) 同一のフィールドにアクセスするメソッドの集合には関係があるとし，該当のアクセスにおける関連性の値の平均値に重み β を乗じたものを，その集合に含まれるメソッド間の相互の関連性の値に加算する．図 5 (b) の例では，m_1，m_2 が f に共通にアクセスしているので，m_1，m_2 の相互の関連性の値に $\beta\{mr'(m_1,f)+mr'(m_2,f)\}/2$ を加える．

(4) 同じメソッドから呼び出されるメソッド間にも同様に，その関連性の値の平均値に重み γ を乗じた値を加算する．図 5 (c) の例では，m_1 が m_2 と m_3 を共通に呼び出しているので，m_2，m_3 の相互の関連性の値に $\gamma\{mr'(m_1,m_2)+mr'(m_1,m_3)\}/2$ を加える．

(5) メソッド間に前述のいずれの関連性もない場合は，これまでに得られた関連性をたどり，間接的な関連性を求め，その呼び出しグラフにおける距離 n のパスに規定される関連性の値の平均値に重み δ^n を乗じたものを用いる．図 5 (d) の例では，m_1 と m_2 の間の距離 2 のパス $m_1 \to m_3 \to m_2$ を考え，関連性の値として $\delta^2\{mr'(m_1,m_3)+mr'(m_3,m_2)\}/2$ を用いる．

(6) 得られた関連性の値を区間 $[0,1]$ に線形正規化する．

α，γ，δ は，メソッド間の関連性を，既知の関連性から間接的に求める際の重みであり，間接的な関連性が直接的な関連性よりも強くならないよう，$(0,1)$ の範囲内で定めることを想定している．β はフィールドアクセスを共有するメソッド間の関係に付与される関連性の重みであり，直接的なデータ依存関係に対する関連性の強さを表現している．

我々は今回の実装において，たとえば LCOM*などの既存の結合度や凝集度のメトリクスを用いず，独自に定義した．これは，FCSP のフレームワークにおいて解を探索する際に，探索空間における勾配が空間内のどの点においても存在していることが，解の導出に有利であるためである．通常，メソッド間の呼び出し関係などを表す行列はスパースになるため，こういった特性を満たさない．しかし，制約充足度を求める際には，できるだけすべての責務の組合せに対して関連性が設定されていることが望ましいため，距離や関連性の大きさがなめらかに変化するよう，前述の定義を用いている．もちろん，既存のメトリクスの応用も可能と考えており，今後検討したい．

既存の FCSP ライブラリ [10] を用いて，FCSP に基づく責務割当て問題のを Java 8 上（Windows 7，Intel Core i7 3820, 3.60 GHz）に実装した．FCSP ソルバには Fuzzy forward checking を採用した．また，制約における重み w^* をすべて 1 に，間接的な関連性は半分に減衰するものとして α，γ，δ を 0.5 に，データ依存による関連性はメソッド呼び出しの倍の関連性を持つものとして β を 2 に設定した．なお，重み w の調整については，5 章で議論する．

4.3 結果

4.3.1 EQ1：割当て精度

提案手法による責務割当ての精度の分布を**図 6** に示す．このグラフは，割り当てる責務数に対する精度を，例題ごとに示している．例題における全責務数を N，ある試行で割り当てる責務数を a とすると，この試行では $N-a$ の割当て対象外の責務に対しては，既知の設計として正解の割当てを与えておく．ここで，既知の設計は，再割当てを行う責務以外の責務に対応する変数のドメインの要素を，正

© 2017 Information Processing Society of Japan

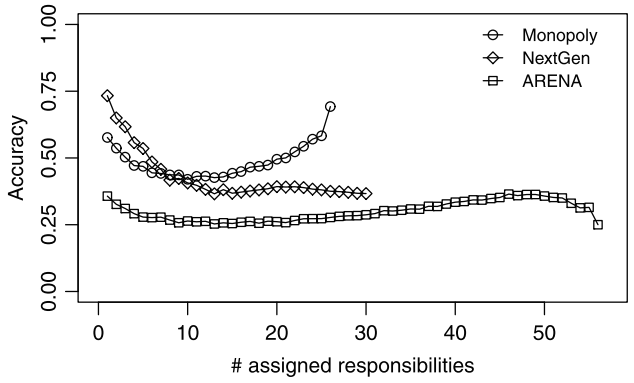


図 6 責務の割当て精度
Fig. 6 Acurracy of responsibility assignments.

解となるクラスのみに限定することによって表現される．既知の設計を与える対象の責務の全組合せを試すことは現実的ではないため，$1 < a < N - 1$ の場合には無作為に作成した 1,000 の充足結果の平均値を精度の値としてプロットしている．$a = 1$，$a = N - 1$ または $a = N$ の場合には，全組合せの平均値を与えている．

それぞれの例題での最も右側，すなわち割当て数が最大となる点は，スクラッチからの割当て，すなわちいずれの責務も割り当てられていない状態からすべての責務を一度に割り当てた際の精度を表している．その結果，それぞれ Monopoly では 69%（26 責務中 18），NextGen では 33%（30 責務中 10），ARENA では 25%（56 責務中 14）の責務が正解どおりのクラス割当てとなっている．

一方で最も左側，すなわち割当て数が 1 となる点は，正解となる責務割当てから取り除いた 1 つの責務を割り当て直す際の精度を表している．その結果，それぞれ Monopoly では 58%（26 責務中 15），NextGen では 73%（30 責務中 22），ARENA では 36%（56 責務中 20）の責務が正しく割り当てられたことになる．

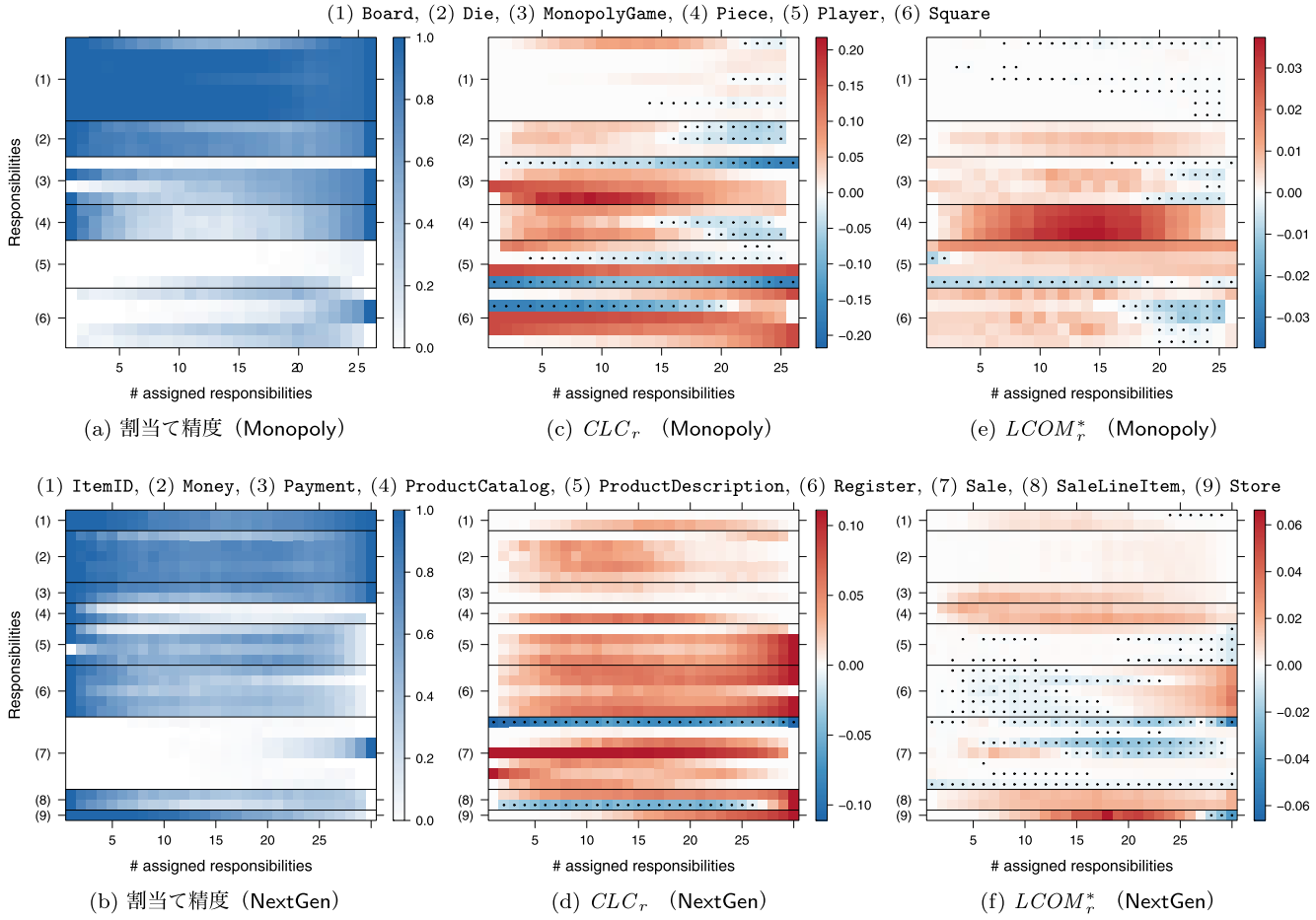
ARENA に対する精度は他の例題に対して低かった．その原因を調査したところ，ARENA 中の特定のクラスに割り当てられるべきメソッドの間の関係性が希薄であることが要因の 1 つであると分かった．該当クラス `TournamentStyle` は，トーナメントの方式を切り替えるための Strategy として機能しており，情報把握責務（フィールド）を持っていなかった．また，所属している責務間の関連性が希薄で独立しており，疎結合性や高凝集性を満たすことを目指しても，それによって得られた割当ては正解に近づかないものだった．解決のためには，特定のデザインパターン [11] や責務割当てパターン [6] などの設計作法に適合する部分を高く評価するための制約が必要と考える．

多数の責務を一度に割り当てるものは，インタラクティブなツールを実現するうえでは実際の利用形態を想定したものではないものの，これらの結果は，FCSP による定式化が基礎的な技術としてある程度有効であることを示していると考える．また，Monopoly と NextGen においては，割り当てる責務数が少ない場合（$a < 4$）は，50%以上の精度で割当てを実現できている．一般に設計には複数の解が存在するため，今回正解として与えた割当ては可能な解の 1 つであることを考慮すると，与えられた正解のうち半数を割当て可能であることは，提案手法が責務割当ての支援システムとして応用可能であることを示唆している．開発者が着目している責務やそれに関連する近隣の責務に対して割当て箇所の推薦や自動的な初期割当てを行うモデリングツールやコーディングツールは，こういった少数の割当てのみでも実現が可能と考える．

図 7 (a)，(b) は，割り当てる責務数に応じた，責務ごとの割当て精度をヒートマップとして可視化したものである．それぞれのグラフの横軸は割り当てる責務数，縦軸は責務の一覧を表しており，セル横 1 列分が，特定の責務がどのように割当てに成功するかを示している．ヒートマップ中の黒線は，正解におけるクラス境界を表している．おおまかに，クラスによって傾向が異なり，割当てに成功する責務が多いものと失敗する責務が多いものがある．たとえば，Monopoly においては，最上部の `Board` クラスは割当てに成功する傾向にあり，一方で 4 番目の `Player` は失敗する傾向にある．また，割当て数が少ない場合に成功する傾向があるものや，多い場合に成功する傾向にあるものがある．こういった傾向には，正解における疎結合性や高凝集性の満たされやすさが影響している．割当て数が大きくならないと割当てに成功しない責務は，周辺の構造に大きな変化がなければ該当の責務が正解に割り当てられないことを示唆している．また，割当て数が少ない場合に割当てに成功する責務は，正解において，その近傍における疎結合性や高凝集性がよく満たされていることを示唆している．このような，責務ごとの割当て成功傾向の違いが，システム全体の割当て精度に影響している．

また，図 7 (c)–(f) は，割り当てられた個々の責務が，割当て全体の設計の品質メトリクスにどのように影響しているかを分析したものである．それぞれのヒートマップ中のセルは，責務割当てによりある責務が正解以外のクラスに配置された際に，該当の責務を正解に割り当てたときと比べて割当て全体のメトリクスの値がどれだけ変動するかの平均値を表している．図 7 (c)，(d) は結合性のメトリクスである CLC_r [12], [13], [14] を，図 7 (e)，(f) は凝集性の欠如のメトリクスである $LCOM_r^*$ [14], [15] に基づいている．いずれのメトリクスも値が高いと低品質であることを表すため，変動が負となる，黒点付きの青色を帯びたセルは，正解の割当てにおけるクラスよりも高品質であったことを示している．図 7 (a)，(b) と (c)–(f) を比較すると，正解との比較に基づいた割当て精度の悪い責務も，メトリクスの観点では値が負になっており，正解を離れる割当てにより設計が改善される可能性が示唆されている．

© 2017 Information Processing Society of Japan



(a) 割当て精度（Monopoly） (c) CLC_r （Monopoly） (e) $LCOM_r^*$ （Monopoly）

(b) 割当て精度（NextGen） (d) CLC_r （NextGen） (f) $LCOM_r^*$ （NextGen）

図 7 責務ごとの割当て精度，CLC_r，$LCOM_r^*$ の変動値

Fig. 7 Fluctuation of accuracy, CLC_r, and $LCOM_r^*$ values.

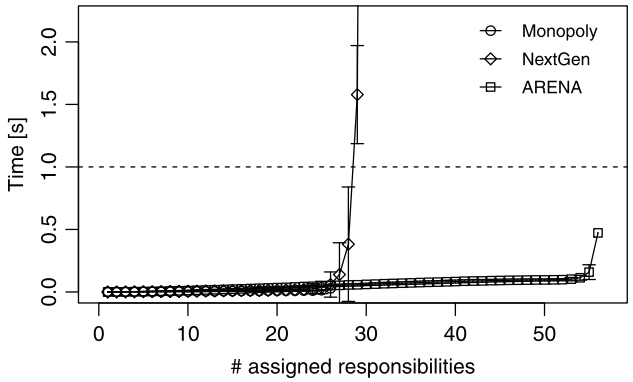


図 8 責務の割当ての計算時間

Fig. 8 Time spent in responsibility assignments.

4.3.2 EQ2：割当てに要する時間

提案手法による責務割当ての計算時間の分布を**図 8** に示す．このグラフは，図 6 と同様に割り当てる責務数を横軸にとり，縦軸は無作為の 1000 試行の平均計算時間を表している．割当て数が少ない場合，実行時間は十分に短く，たとえば，$a < 25$ に対して，いずれの例題でも割当て時間が 200 ミリ秒を下回っている．しかし，計算時間は，割当てが必要となる責務の数の増大に対して，指数的に大きくなることが知られており，グラフの右側の形からもそれがうかがえる．たとえば，NextGen では，$a > 28$ の際に割当て時間が 1 秒を超えている．ユーザ・インタフェースの設計として，利用者が待たされていると感じない目安として，1 秒程度が許容されている [16]．割当て問題の規模に応じて，解の導出時間は大きくなるため，多数の割当ての限界については別途検討する必要があるものの，提案手法で少数の責務を割り当てる際には，現在の枠組みで十分に実現できることが分かる．開発者が着目している責務やそれに関連する近隣の責務に対して割当て箇所の推薦や自動的な初期割当てを行うモデリング・ツールやコーディング・ツールは，こういった少数の割当てのみでも実現が可能と考える．

4.3.3 EQ3：意図制約の利用

意図制約の付与の例として，EQ1 の Monopoly に対する全割当ての試行において目標設計と食い違いの見られた責務に対して，意図に基づく制約（c^{same}，c^{diff}）の付与を試みた．**図 9** に割当て結果のクラス図を示す．ここで，正解と異なる割当てとなった責務には，責務名の後に [*class*] のように正しい割当て先のクラスを示してある．なお，我々の定式化においてはそれぞれのクラスは無名であるため，ク

© 2017 Information Processing Society of Japan

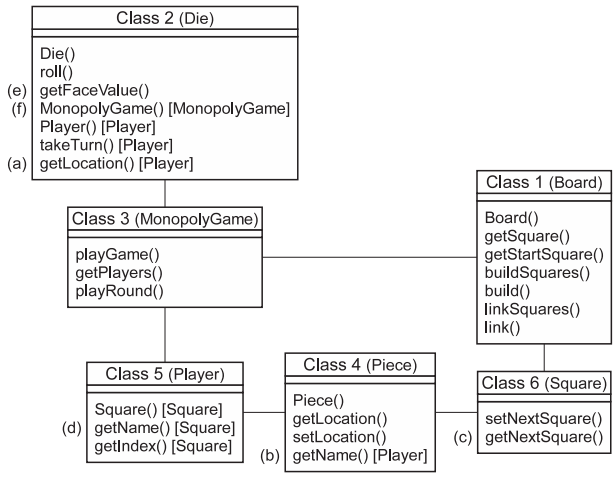


図 9 Monopoly における割当て
Fig. 9 Assignment of all responsibilities (Monopoly).

表 3 ユーザ意図を付加した場合の割当て
Table 3 Assignment using intentional constraints.

責務	制約	制約付与前	制約付与後
(a) `getLocation()`	c^{same}	Class 2 (Die)	Class 4 (Piece)
(b) `getName()`		Class 4 (Piece)	
(c) `getNextSquare()`	c^{same}	Class 6 (Square)	Class 6 (Square)
(d) `getName()`		Class 5 (Player)	
(e) `getFaceValue()`	c^{diff}	Class 2 (Die)	Class 2 (Die)
(f) `MonopolyGame()`			Class 4 (Piece)

ラス間関係導出時に用いたクラスの名前を括弧で付してある．この結果に対して，図 9 (a) と (b)，(c) と (d) がそれぞれ同じクラスに割り当てられるよう c^{same} を，(e) と (f) が異なるクラスに割り当てられるよう c^{diff} を設定し，解を求めさせた．その結果，すべての制約が反映された（**表 3**）．また，この実験においては，ソルバの実行時間は 20 ミリ秒以下だった．

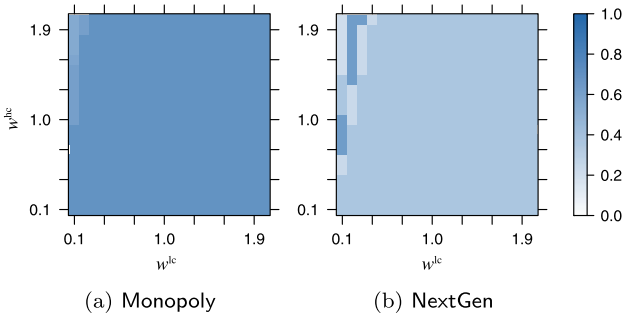
この結果は，このようなユーザの意図を，疎結合，高凝集という全体的な指標ではない，部分的な制約として追加可能であることを示していると考える．ただし，制約が満たされた結果，必ずしもすべての責務が正解のクラスに割り当てられたわけではない．ユーザが適切な制約を選択したり，制約付与の試行錯誤を行ったりすることを支援する環境の提供が有用と考える．

5. 議論

5.1 パラメータの選定

今回はいずれの実験においても，全制約の重みを経験的に決定したが，ユーザの目的に合わせた設定も可能である．しかしながら，適切な重みの値は，対象とする問題に強く依存することに注意が必要である．

例として，重み w^{lc}，w^{hc} の適切値の設定を，予備評価で用いた例題で検討した．**図 10** は，各重みを $0.1 \leq w \leq 1.9$ の範囲で 0.1 刻みに変動させた際の，スクラッチからの割



(a) Monopoly (b) NextGen

図 10 重み w^{lc}，w^{hc} による精度への影響
Fig. 10 Effects of the weights w^{lc} and w^{hc} to the accuracy.

当て結果の精度の値の変動を示している．色の濃い領域が高精度の結果を示しているが，Monopoly と NextGen の 2 例題で値の変動傾向が異なる．いずれの例題でも，w^{lc} と w^{hc} に低めの値を設定した際に精度が変動しているが，精度の増加，減少傾向に違いがある．

このことから，現実的には，対象システムの特徴や利用者の設計に対する嗜好などを考慮可能な，重みの割当てメカニズム，インタフェースが必要となると考える．

5.2 利用したメトリクスや制約の妥当性

本論文ではメソッドを実行責務と見なし，その関係性を算出するのに，4.2.2 項で述べたとおり，メソッドの呼び出し関係やフィールドに対するアクセス関係を利用した．しかしながら，メソッドの関連性として，アクセサ・メソッドを経由した間接的なフィールド・アクセス，ならびに，フィールドに対するアクセスそのものを考慮すべきかどうかには議論の余地がある．すなわち，概念モデルに表現されたクラス間の関係から割当てが既知なアクセサ・メソッドは責務割当て問題の対象にすべきではないという議論がある．今回は研究の初期段階として，また FCSP への親和性を重視して単純な関係を用いたものの，今後，LCOM* [15] などをはじめとする，責務間の関係性をより適切に表現するソフトウェア・メトリクスを，FCSP で利用しやすいよう加工して用いることを検討したい．

また，上記に鑑み，疎結合性と高凝集性のファジィ制約の検討も必要であると考える．たとえば，3.2 節の定義における高凝集性の制約で，責務間の関連性が低い場合，たとえそれらが同クラスに所属していなかったとしても，クラス間距離が小さければペナルティを受けるよう設定されている．これは，本実験で行ったように，責務間の関連性が複数の責務を経た間接的なものも含めて定義されていることを前提としている．ファジィ制約充足が機能するよう，なめらかな充足度の分布に従う制約を既存のメトリクスから実現する必要がある．

5.3 他の最適化手法との比較

提案手法は，FCSP としての責務割当て問題の定式化を

© 2017 Information Processing Society of Japan

表 4 最適化手法の比較
Table 4 Comparison of optimization-based techniques.

	最適化の観点					
	疎結合	高凝集	設計維持	意図	定式化とその粒度	求解アルゴリズム
Glavaš と Fertalj [17]	√	√			観点全体の単一の適合関数	GA，山登り，焼きなまし，粒子群最適化
Bowman ら [18]	√	√			観点ごとの複数の適合関数	多目的 GA（SPEA2）
提案手法	√	√	√	√	責務ごとの多種のファジィ制約	先読みチェック探索

初めて行ったという意味で新規性を有していると考える一方，責務割当てを最適化問題として扱うこと自体は既存である．たとえば，Bowman らは遺伝的アルゴリズム（GA）を用いてクラス図やシーケンス図を分析し，結合度と凝集度を最適化するようなメソッドと属性の配置を探索する手法を提案しており [9], [18]，提案手法と同様に，責務割当てを最適化問題と見なし，既存のアルゴリズムにより求解を行っている．他の様々なメタヒューリスティクスと同様，類似の評価関数を用意することにより，提案手法と類似の最適化を実現できると考える．

提案手法を含めた，いくつかの最適化に基づく責務割当て手法の比較を**表 4** に示す．この表には，各手法が扱う最適化の観点に加え，それらが最適化問題として表現される際の定式化の方式とその粒度，求解アルゴリズムがまとめられている．すべての手法が，疎結合性や高凝集性に基づいた最適化の観点を扱っている．提案手法はこれに加えて，設計維持や意図反映などの支援ツール構築に向けた観点を扱っている．Glavaš と Fertalj の手法 [17] は単目的の最適化を扱っており，疎結合性と高凝集性を組み合わせた単一の適合関数を用いて，様々な求解アルゴリズムを適用している．Bowman らの手法では多目的 GA が用いられており，観点ごとに適合関数が定義されている．一方，我々の手法はファジィ制約により定式化されており，各責務に対して多種のファジィ制約が定義され，それらの組合せとして全体の制約充足度が定義される．

このように，責務レベルの制約に基づいて定式化が行われていることは，個々の制約の調整のしやすさにつながると考える．これは，局所的な制約の追加が直感的に行えることを意味している．実際，FCSP に対して EQ3 の実験で新しく 2 種類の制約を追加することは容易であったし，求解も問題なく行われた．また，制約可視化ツール [19] を用いて，個々の制約の充足度を把握することにより，適宜意図制約を追加することも可能と考える．一方，FCSP 全体の充足度として個々の制約の充足度の最小値を用いているため，最小値の向上に貢献しないような部分的な最適化が促されない可能性がある．実験での例題で，きわめて高い精度が得られなかった理由としてこの可能性が残るため，さらなる分析を加えたい．

6. 関連研究

設計の質を向上させる様々な手法がすでに提案されている．前述のとおり，Bowman らは GA を用いた責務割当て手法を提案している [9], [18]．また，Tsantails らは，メソッドや属性の参照関係に基づき，結合度を抑えて凝集度を上げるメソッド配置を特定し，開発者に提示する手法を提案している [20]．この手法では，各メソッドについて最もクラス間の参照関係を抑えられる割当て先クラスが調査され，発見されたクラスへのメソッド移動リファクタリングが提示される．これらの手法は，用いられるアルゴリズムや注目点は異なるものの，結合性と凝集性に基づいた最適化を行っている点で我々の手法と類似している．

UML モデルの問題箇所を検出するものとしては，Zamani らの手法 [21] がある．この手法では，ステレオタイプを用いて UML モデルを分析し，エンタープライズ・アプリケーション・アーキテクチャ・パターンが正しく適用されているかどうかを事前に定義した OCL 条件式に基づいて判断し，不適切な場合は問題を開発者に知らせる．この手法は複数クラスの相互関係などに関するパターンを対象としたものであり，我々が扱っている結合性，凝集性に基づいたものとは異なる．

Sunyé らは UML クラス図および状態遷移図におけるモデルリファクタリングを OCL で定義した変換規則の列として定式化している [22]．しかし，この手法では割当てを自動的に特定したり，改善すべき箇所を特定したりすることについては論じられていない．Trifu らは設計欠陥と直接観測可能な指針との関係について論じている [23]．設計欠陥の仕様を文脈と指針，指針と自然言語で書かれた修正戦略を用いた診断戦略も含めて定義している．また，彼らは設計欠陥を発見するためのツールを実装している．彼らの設計欠陥発見のための指針は設計メトリクスと構造情報の組合せとして定義されている．ClassCompass [24] もソフトウェア設計の自動的な評価システムであり，自然言語で記述された規則に基づく設計修正案の推薦が可能である．ただし，これらは適切な責務割当てを目的としているわけではない．その他，メトリクスを利用した設計の不吉な臭いの検出 [25]，メトリクス値の分析に基づくアンチパターンの検出 [26]，メトリクスとプログラムの構造的特徴に基づく設計の臭いの検出 [27] など，ソースコードの分析

© 2017 Information Processing Society of Japan

に基づいたソフトウェア・メトリクスを利用して設計の不吉な臭いを検出する手法も提案されている．

メトリクス以外にも，様々な責務割当ての指標が提案されている．たとえばPoOOD [28] や，過去の設計の事例を分析して汎用的な9つの原則を抽出してパターン化したGRASP [6] などの指針がある．Akiyamaらは，GRASPを用いた責務割当て支援手法を提案している [14]．この手法は，責務記述の類似性に基づき既存の責務割当てから不吉な臭いを発見し，これを解消するようGRASPのパターンの適用箇所を推薦する．これらの指針をファジィ制約として定式化することにより，提案手法でも，必要に応じてパターンの適用を考慮したような割当てが可能になると考える．

FCSPは，従来の制約充足問題にファジィ制約を導入したモデルであり，曖昧さを表す充足度によって不完全に充足される解を求め，実世界問題の解決に有用な情報の提供を可能としている [7]．たとえば，色覚特性を考慮した配色の調整がFCSPにより実現されている [29]．エンド・ユーザを想定した応用例として，グラフィカル・ユーザ・インタフェースのウィジェット・レイアウトの最適化問題に適用した研究がある [30]．

7. まとめ

本論文では，ソフトウェア工学におけるクラス責務割当て問題を，人工知能ならびにソフト・コンピューティング分野におけるフレームワークであるファジィ制約充足問題として定式化し，また例題において解を導出できることを示した．今後の課題として，以下を考えている．

さらなる例題への適用． 本論文で用いた例題は小規模なプログラムであるため，提案手法が実システムの規模においても十分に実用的かどうかは未確認であり，他の例題への適用を重ねる必要がある．

より高いスケーラビリティの確保． 一般に，最適化問題はスケーラビリティが課題となる．提案手法もそれについては同様であるが，FCSPとして定式化しているため，FCSP用のソルバであればいずれも適用可能である．今後，定式化手法そのもののスケーラビリティの検証や適切なFCSPソルバの選定を予定している．

他の品質メトリクスや制約の利用の検討． 5章で述べたように，責務間の関連性をより適切に表現するメトリクスを検討し，疎結合性や高凝集性などの定義を検討したい．また，たとえば責務割当ての指針の1つであるGRASP [6] など，本論文で取り上げた以外の観点がファジィ制約として表現できるかも確かめたい．

CASEツールの実現． 設計者の意図の注釈を付与できる，対話的な責務割当ての支援ツールを実現し，そのユーザビリティや有用性を評価することが望ましい．

謝辞 本研究の一部は科学研究費補助金（JP15K15970, JP15H02683，JP15H02685）の助成を受けた．

参考文献

[1] Hayashi, S., Yanagida, T., Saeki, M. and Mimura, H.: Class Responsibility Assignment as Fuzzy Constraint Satisfaction, *Proc. 6th International Workshop on Empirical Software Engineering in Practice*, pp.19–24 (2014).

[2] Wirfs-Brock, R. and McKean, A.: *Object Design: Roles, Responsibilities, and Collaborations*, Addison-Wesley (2002).

[3] Bellin, D. and Simone, S.S.: *The CRC Card Book*, Addison-Wesley Professional (1997).

[4] Beck, K. and Cunningham, W.: A Laboratory for Teaching Object-Oriented Thinking, *Proc.Conference on Object-Oriented Programming Systems, Languages and Applications*, pp.1–6 (1989).

[5] Sharble, R.C. and Cohen, S.: The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods, *ACM SIGSOFT Software Engineering Notes*, Vol.18, No.2, pp.60–73 (1993).

[6] Larman, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edition*, Prentice Hall (2005).

[7] Ruttkay, Z.: Fuzzy Constraint Satisfaction, *Proc. 3rd IEEE Conference on Fuzzy Systems*, Vol.2, pp.1263–1268 (1994).

[8] Bruegge, B. and Dutoit, A.H.: *Object-Oriented Software Engineering Using UML, Patterns, and Java, 3rd edition*, Prentice Hall (2009).

[9] Bowman, M., Briand, L.C. and Labiche, Y.: Solving the Class Responsibility Assignment Problem in Object-Oriented Analysis with Multi-Objective Genetic Algorithms, *IEEE Trans. Softw. Eng.*, Vol.36, No.6, pp.817–837 (2010).

[10] 柳田拓人，須藤康裕：ファジィ制約充足問題ライブラリStlics，入手先 ⟨http://takty.stxst.com/res/stlics/⟩（参照2016-08-10）．

[11] Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1994).

[12] Briand, L.C., Daly, J.W. and Wüst, J.K.: A Unified Framework for Coupling Measurement in Object-Oriented Systems, *IEEE Trans. Softw. Eng.*, Vol.25, No.1, pp.91–121 (1999).

[13] Hitz, M. and Montazeri, B.: Measuring Coupling and Cohesion in Object-Oriented Systems, *Proc. International Symposium on Applied Corporate Computing* (1995).

[14] Akiyama, M., Hayashi, S., Kobayashi, T. and Saeki, M.: Supporting Design Model Refactoring for Improving Class Responsibility Assignment, *Proc. ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems*, pp.455–469 (2011).

[15] Henderson-Sellers, B.: *Object-Oriented Metrics: Measures of Complexity*, Prentice Hall (1995).

[16] Card, S.K., Robertson, G.G. and Mackinlay, J.D.: The Information Visualizer, an Information Workspace, *Proc. Conference on Human Factors in Computing*, pp.181–188 (1991).

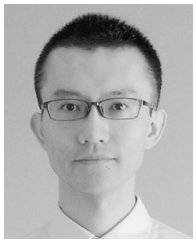
[17] Glavaš, G. and Fertalj, K.: Solving the Class Responsibility Assignment Problem Using Metaheuristic Approach, *Journal of Computing and Information Technology*, Vol.19, No.4, pp.275–283 (2011).

© 2017 Information Processing Society of Japan

[18] Bowman, M., Briand, L.C. and Labiche, Y.: Multi-Objective Genetic Algorithms to Support Class Responsibility Assignment, *Proc. 23rd IEEE International Conference on Software Maintenance*, pp.124–133 (2007).
[19] Yanagida, T., Kurihara, M. and Nonaka, H.: A Tool for Visualizing the Behavior of Fuzzy Constraint Satisfaction Solvers, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol.14, No.5, pp.425–430 (2010).
[20] Tsantalis, N. and Chatzigeorgiou, A.: Identification of Move Method Refactoring Opportunities, *IEEE Trans. Softw. Eng.*, Vol.35, No.3, pp.347–367 (2009).
[21] Zamani, B. and Butler, G.: Smell Detection in UML Designs which Utilize Pattern Languages, *Iranian Journal of Electrical and Computer Engineering*, Vol.8, No.1, pp.47–52 (2009).
[22] Sunyé, G., Pollet, D., Traon, Y.L. and Jézéquel, J.-M.: Refactoring UML Models, *Proc. 4th International Conference on the Unified Modeling Language*, pp.134–148 (2001).
[23] Trifu, A. and Reupke, U.: Towards Automated Restructuring of Object Oriented Systems, *Proc. 12th Working Conference on Reverse Engineering*, pp.39–48 (2007).
[24] Coelho, W. and Murphy, G.: ClassCompass: A Software Design Mentoring System, *Educational Resources in Computing*, Vol.7, No.1, pp.1–18 (2007).
[25] Marinescu, R.: Detection Strategies: Metrics-based Rules for Detecting Design Flaws, *Proc. 20th International Conference on Software Maintenance*, pp.350–359 (2004).
[26] Oliveto, R., Khomh, F., Antoniol, G. and Guéhéneuc, Y.-G.: Numerical Signatures of Antipatterns: An Approach Based on B-Splines, *Proc. 14th European Conference on Software Maintenance and Reengineering*, pp.248–251 (2010).
[27] Moha, N., Guéhéneuc, Y.-G., Duchien, L. and Meur, A.-F.L.: DECOR: A Method for the Specification and Detection of Code and Design Smells, *IEEE Trans. Softw. Eng.*, Vol.36, No.1, pp.20–36 (2010).
[28] Martin, R.C.: *Agile Software Development: Principles, Patterns, and Practices*, Prentice Hall (2002).
[29] Yanagida, T., Okajima, K. and Mimura, H.: Color Scheme Adjustment by Fuzzy Constraint Satisfaction for Color Vision Deficiencies, *Color Research & Application*, Vol.40, No.5, pp.446–464 (2015).
[30] Yanagida, T. and Nonaka, H.: Flexible Widget Layout Formulated as Fuzzy Constraint Satisfaction Problem, *Proc. 1st KES International Symposium on Intelligent Decision Technologies*, pp.73–83 (2009).

林 晋平 （正会員）

2008 年東京工業大学大学院情報理工学研究科計算工学専攻博士後期課程修了．2009 年同専攻助教．現在，同大学情報理工学院助教．博士（工学）．ソフトウェア変更やソフトウェア開発環境の研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE-CS，ACM 各会員．



柳田 拓人

2009 年北海道大学大学院情報科学研究科コンピュータサイエンス専攻博士後期課程修了．2009 年静岡大学電子工学研究所助教．ヒューマン・コンピュータ・インタラクションおよび人工知能の研究に従事．2014 年より株式会社スペースタイムにて，小中学生向けサイエンス&プログラミング教室ラッコラのカリキュラム開発と運営に従事，現在に至る．博士（情報科学）．

佐伯 元司 （正会員）

1983 年東京工業大学大学院工学研究科情報工学専攻博士後期課程修了．同大学助手，助教授を経て，2000 年同大学大学院情報理工学研究科計算工学専攻教授．現在，同大学情報理工学院教授．工学博士．要求工学やソフトウェア開発技法等の研究に従事．電子情報通信学会，人工知能学会，ソフトウェア科学会，IEEE，ACM 各会員．

三村 秀典

1987 年静岡大学大学院電子科学研究科電子応用工学専攻博士課程修了．1987 年新日本製鉄第一技術研究所主任研究員．1994 年国際電気通信基礎技術研究所（ATR）主任研究員．1996 年東北大学電気通信研究所助教授．2003 年静岡大学電子工学研究所教授，2007 年同研究所所長，現在に至る．工学博士．真空ナノ・エレクトロニクスの研究に従事．映像情報メディア学会，電子情報通信学会，応用物理学会，IEEE，Society for Information Display（SID）各会員．

© 2017 Information Processing Society of Japan