クラス責務割り当て問題へのファジィ制約充足問題の適用

柳田 拓人[†]，林 晋平[††]，佐伯 元司[††]，三村 秀典[†]

[†]静岡大学電子工学研究所
[††]東京工業大学大学院情報理工学研究科

クラス責務割り当て問題（1/4）

- オブジェクト指向開発において……
  - 責務（Responsibility）
    - 各クラスのインスタンスが果たすべき役割
    - 動作責務，知識（情報把握）責務
    - $m \in M$
  - 責務割り当て（Class Responsibility Assignment; CRA）
    - 責務が所属するクラス $k \in K$ の決定
    - 写像 $M \rightarrow K$ の決定に相当

クラス責務割り当て問題（2/4）

- 適切な責務割り当ての重要性

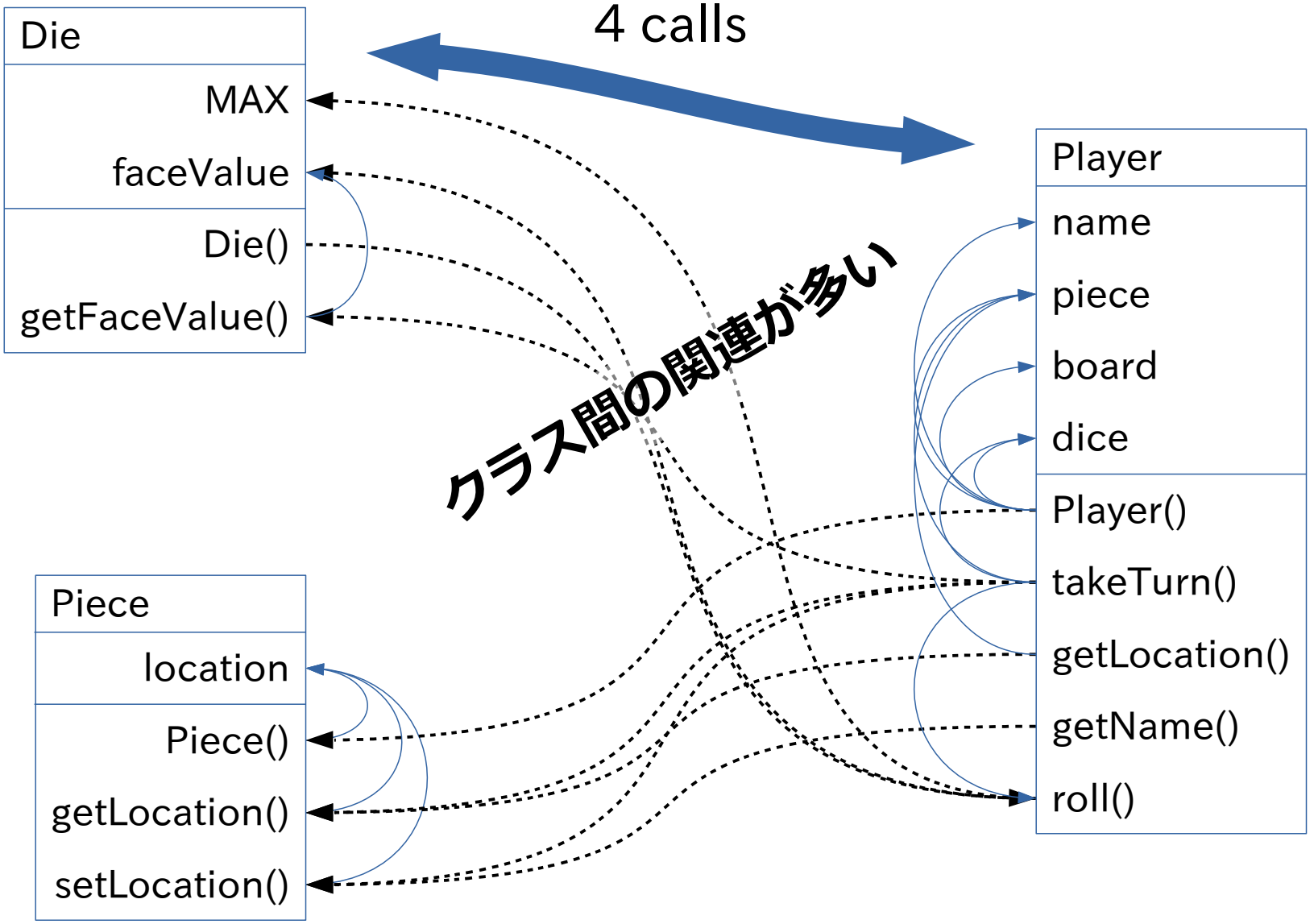
→ 高品質な設計（明確性，保守性，拡張性）

互いに関連のある責務を1箇所に割り当て

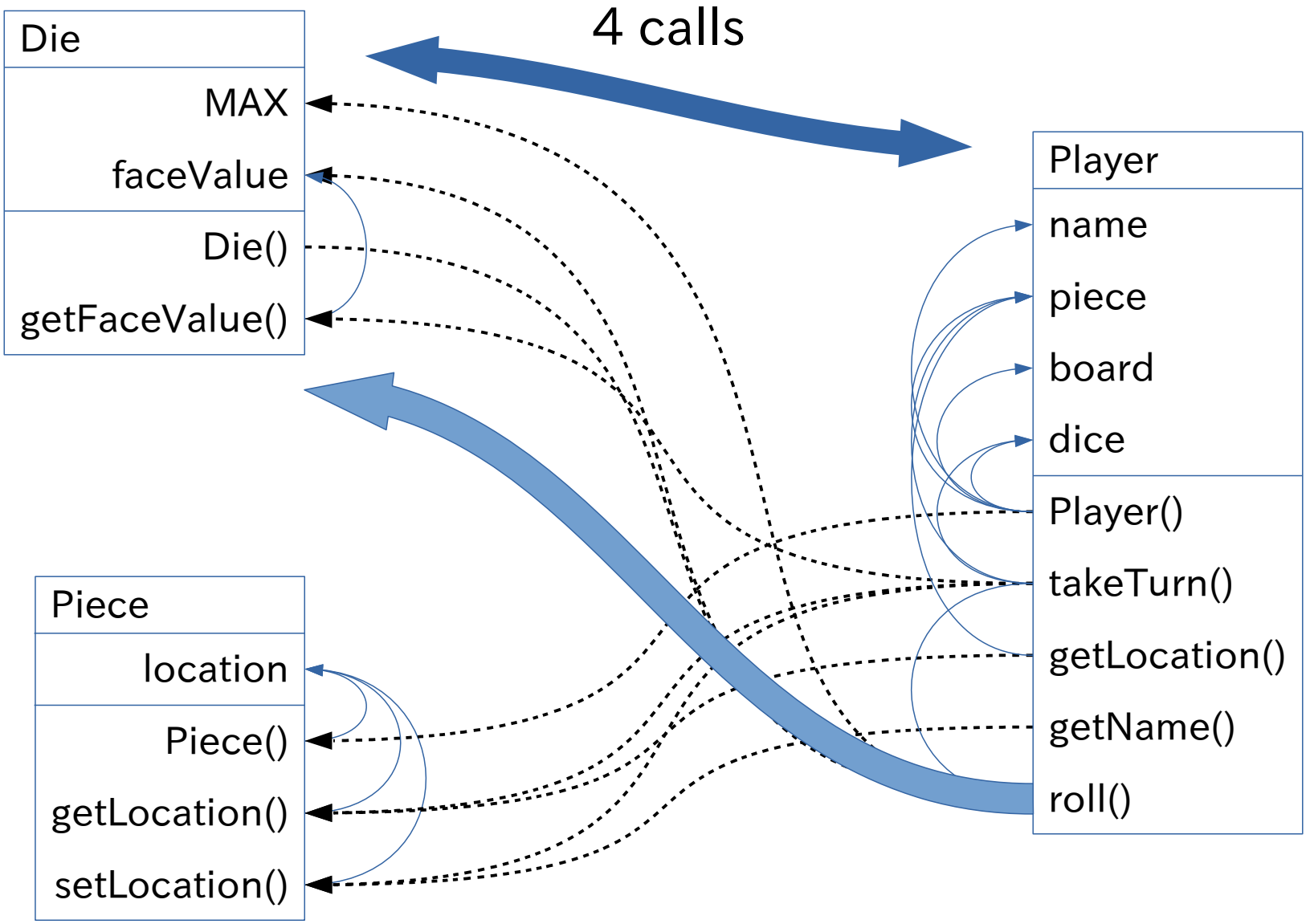
↓

それらの責務に変更が生じても，
ソフトウェアの他の部分に影響が波及しない

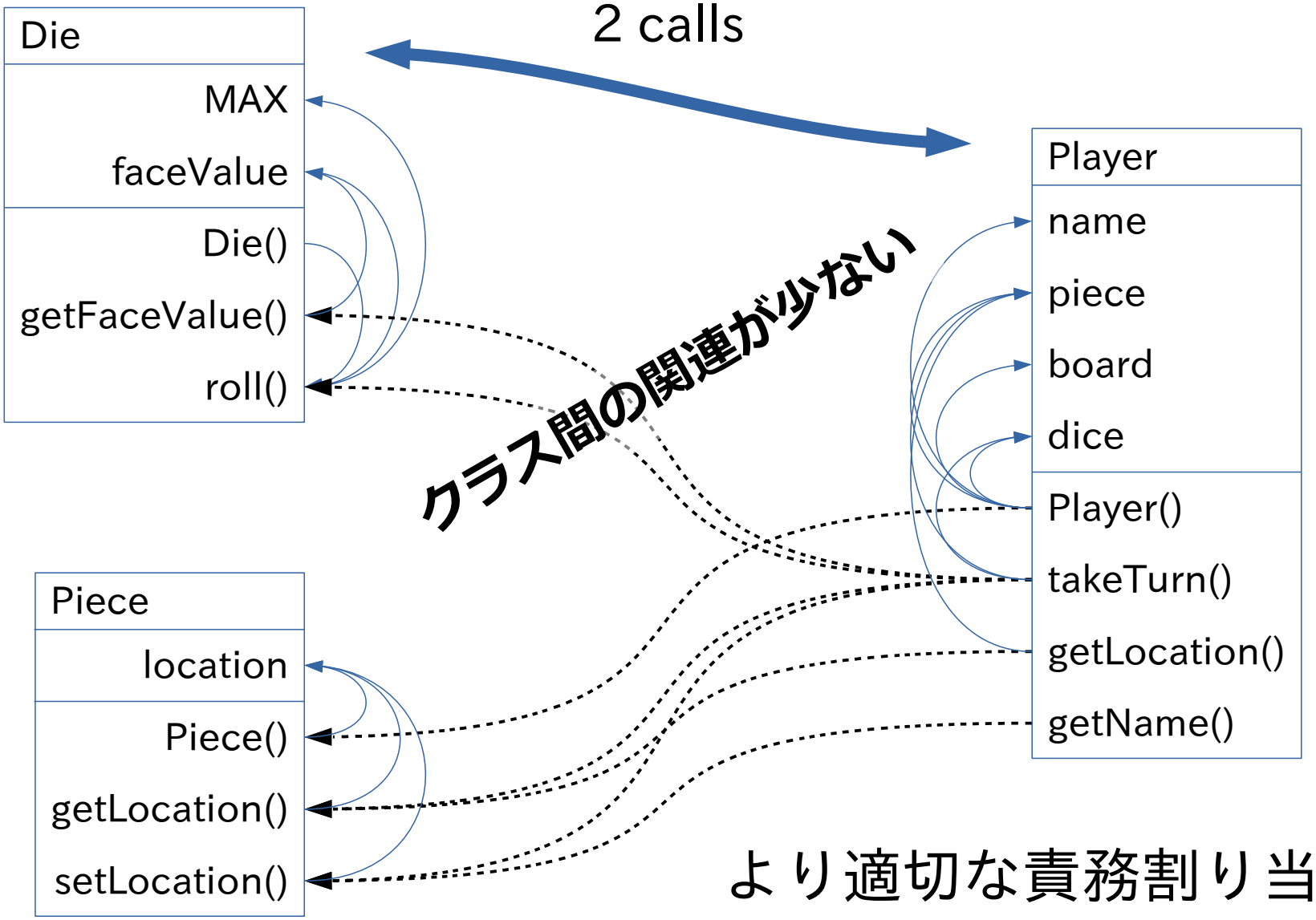
クラス責務割り当て問題（3/4）



クラス責務割り当て問題（3/4）



クラス責務割り当て問題（4/4）



より適切な責務割り当て

方法論

- 責務割り当てのための様々な方法
  - 責務駆動アプローチ（Responsibility Driven Approach）[1]
  - CRCカード法 [2]，その実用 [3, 4]
- クラスと責務の抽出は可能
  - 質的観点からは不十分（そのまま設計に使えない）

[1] R. Wirfs-Brock and A. McKean, Object Design: Roles, Responsibilities, and Collaborations, Addison-Wesley, 2002.
[2] D. Bellin and S.S. Simone, The CRC Card Book, Addison-Wesley Professional, 1997.
[3] K. Beck and W. Cunningham, “A laboratory for teaching object-oriented thinking,” Proc. Conference on Object-Oriented Programming Systems, Languages and Applications, pp.1–6, 1989.
[4] R.C. Sharble and Samuel S.Cohen, “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.

典型的な観点と自動化

- 設計の質的観点
  - 疎結合（クラスをまたがる責務に依存関係が少ない）
  - 高凝集（同一クラス内の責務が協調し合っている）

- クラスと責務の抽出後:
  - 既存の設計の高品質化
  - 議論の開始点となる設計原案の導出

責務割り当ての自動化が重要

自動化の課題

- 様々な状況と求められる対応
  - **トレードオフ**を考慮した，
    指針を適度に満たす解の導出
  - 設計者の**試行錯誤**に応えられる高速割り当て
    途中経過を尊重した後続の割り当て
  - 責務の質を示す観点の混在，切り替えなどの
    高い**柔軟性**

自動化の指針（1/2）

- **疎結合性**
  - アクセスする責務のクラス間距離を出来るだけ短くしたい
- **高凝集性**
  - 近いクラスにある責務間の関連性を出来るだけ高くしたい

トレードオフ

※クラス間距離，責務の関連性は様々なメトリクスを利用

自動化の指針（2/2）

- **設計維持**
  - ユーザ設計に出来るだけ近いクラスに割り当てたい
- **意図反映**
  - 出来るだけ同じクラスに割り当てたい
  - 出来るだけ異なるクラスに割り当てたい

試行錯誤

それぞれの指針を**柔軟**に設定可能に……

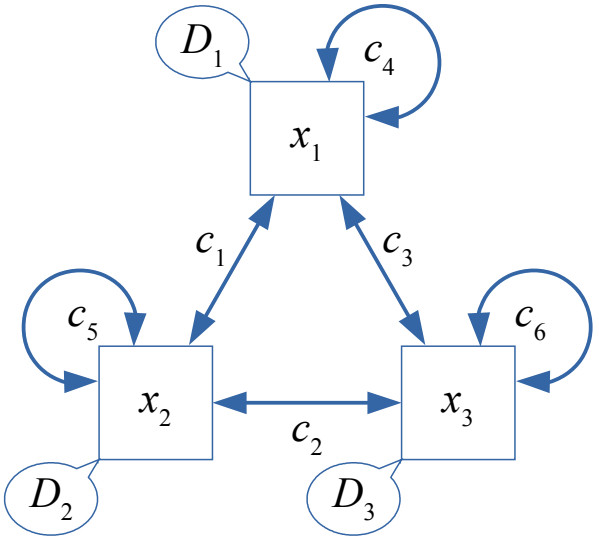
提案手法

- ファジィ制約充足問題（FCSP）[14] による責務割り当ての自動化
  - FCSP
    - 組合せ探索問題の枠組み（人工知能分野）
    - 汎用ソルバが存在
  - FCSPとしての定式化
    - 様々な指針をファジィ制約として自然に表現
    - 評価関数を導入する必要なし

[14] Z. Ruttkay, “Fuzzy constraint satisfaction,” Proc. the 3rd IEEE Conference on Fuzzy Systems, vol.2, pp.1263–1268, 1994.

ファジィ制約充足問題（1/2）

- 変数: $x \in X$
  - 変数への値の割り当てが解を表現
- ドメイン: $D = \{ D_x \mid x \in X \}$
  - 対応する変数が取り得る値の集合
- ファジィ制約: $c \in C$
  - 可能な値の組合せ
  - 組合せ毎に制約充足度



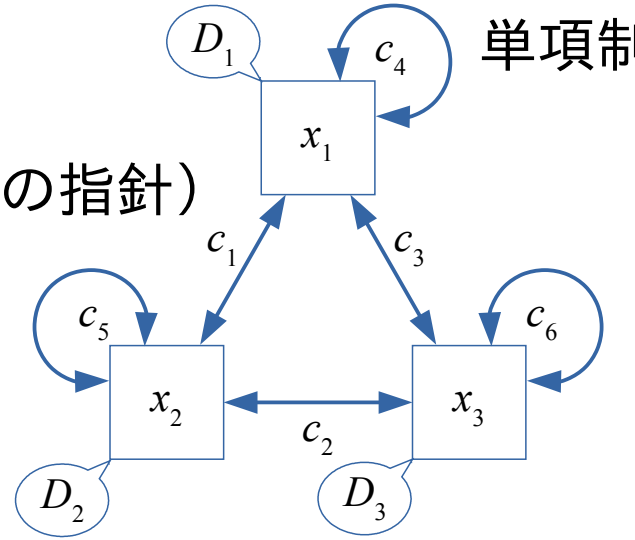
ファジィ制約充足問題（2/2）

- 解: 全ての変数への値割り当て
  - 全ての制約充足度の最小値:
    - $C_{\min}(v) = \min_{c \in C}(\mu R_c(v[S_c]))$
    - $C_{\min}(v) > 0$ のとき，割り当てvは解
  - 解は制約充足度 $(0, 1]$ を持つ
    - ソルバは$C_{\min}(v)$が最大の解を探索

クラスへの責務の割り当て

定式化

- 変数: x ←「責務」$m \in M$
- ドメイン: D ←「クラスの集合」K
- ファジィ制約: c ←「割り当て指針」



単項制約（設計維持）

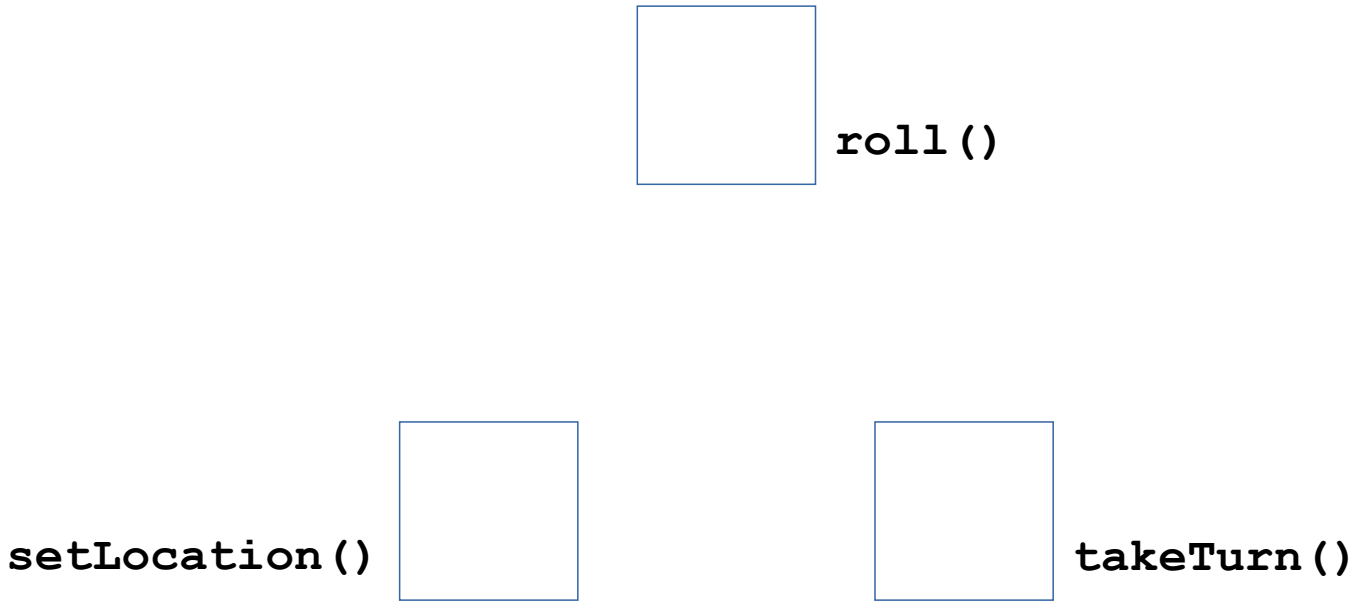
二項制約（その他の指針）

ドメイン
（クラスの集合）

責務数3の場合

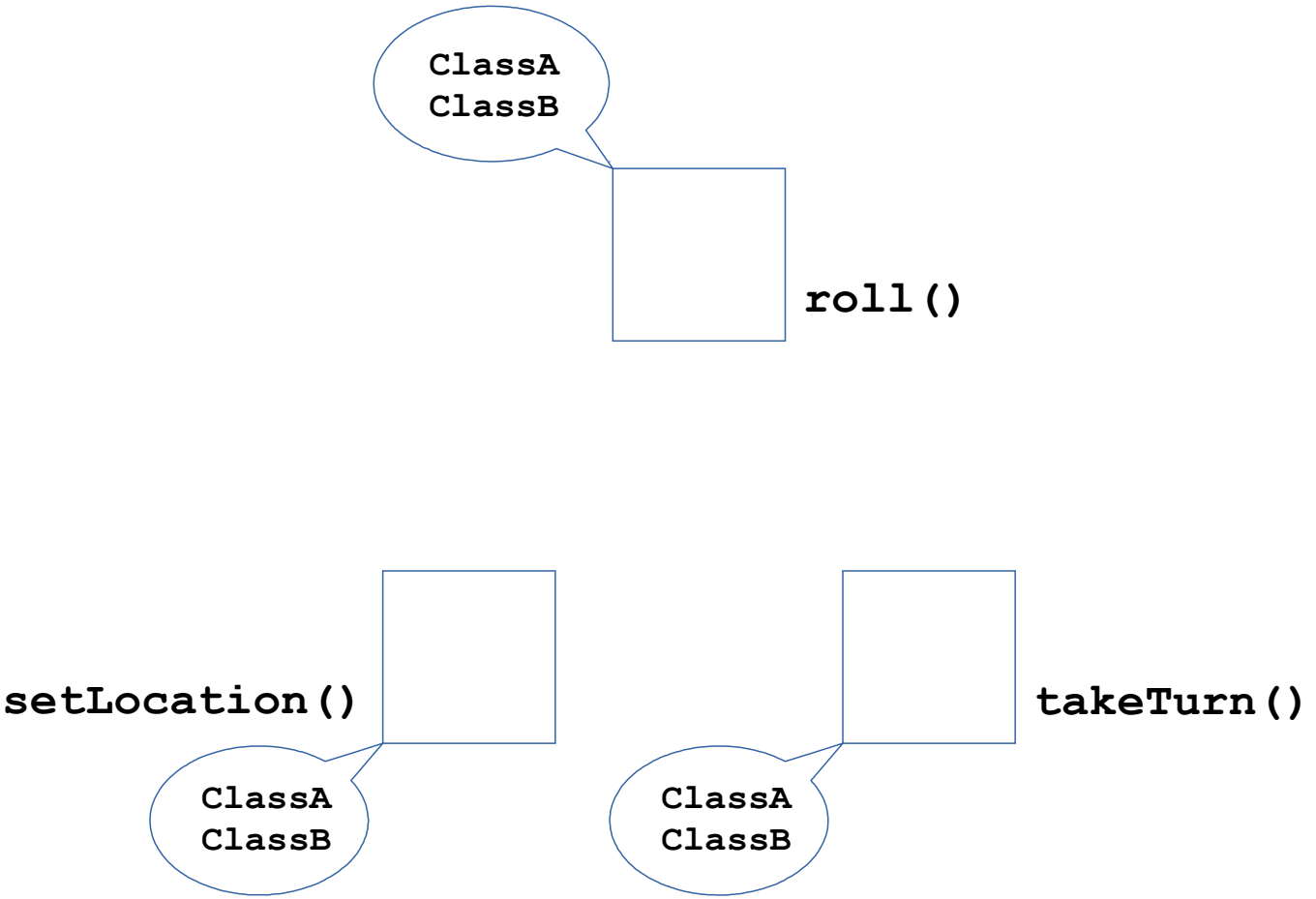
定式化の例（1/2）

3責務→2クラス



定式化の例（1/2）

3責務→2クラス



制約定義のための関数

- クラス間の正規化距離 $cd : K^2 \rightarrow [0, 1]$
  - k_1とk_2が同一である時: $cd(k_1, k_2) = 0$
  - k_1とk_2の距離が最も遠い時: $cd = 1$
- 責務間の正規化関連性 $mr : M^2 \rightarrow [0, 1]$
  - m_1とm_2の関連性が全くない時: $mr(m_1, m_2) = 0$
  - m_1とm_2の関連性が最も高い時: $mr = 1$

※ $cd(<k_1, k_2>)$ を単に $cd(k_1, k_2)$ と表記

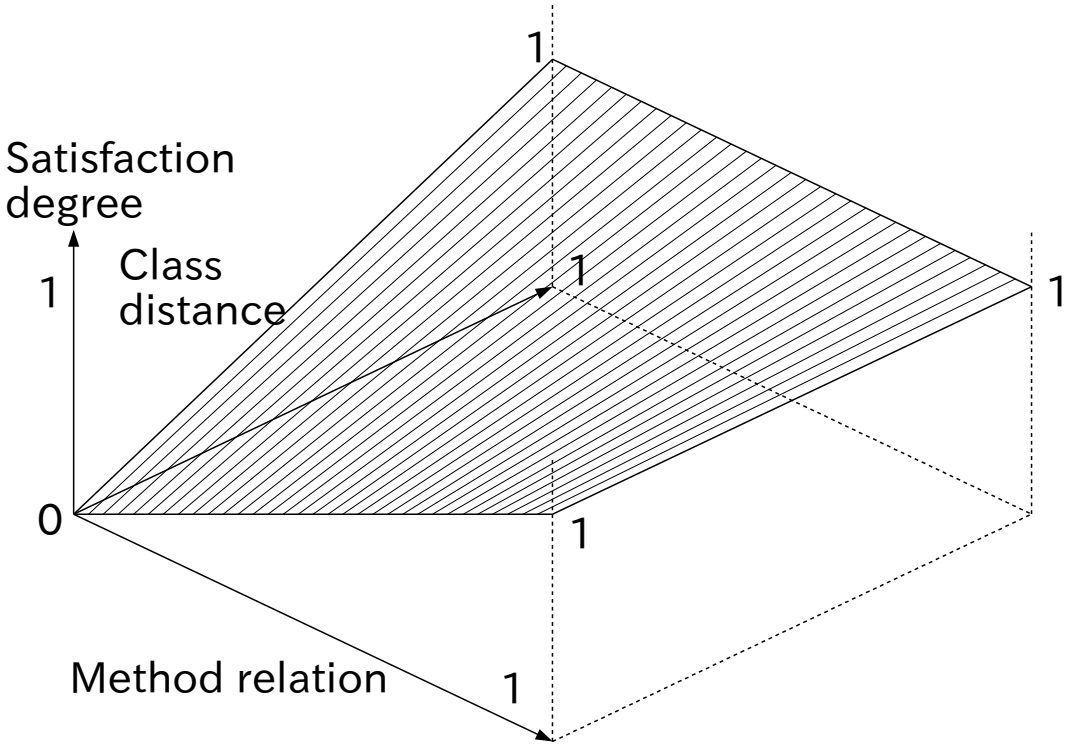
疎結合性の制約（1/2）

- 制約 c_{lcp}
  - **関連しない責務は出来るだけ遠いクラスに……**
  - 全変数（責務）のペアに設けられる二項制約
    - 責務間の関連性が低い時に，
      クラス間距離が小さくなると充足度が低下

$$\mu R_{\mathrm{c}^{\mathrm{lcp}}_{\mathrm{m}_1,\mathrm{m}_2}}(k_1,k_2)$$
$$= \{(1-mr(m_1,m_2))cd(k_1,k_2)+mr(m_1,m_2)\}^{w^{\mathrm{lcp}}}$$

w^{lcp}は重み

疎結合性の制約（2/2）



$$w^{\mathrm{lcp}} = 1$$

$$\mu R_{\mathrm{c}_{\mathrm{m}_1,\mathrm{m}_2}^{\mathrm{lcp}}}(k_1, k_2)$$
$$= \{(1 - mr(m_1, m_2))cd(k_1, k_2) + mr(m_1, m_2)\}^{w^{\mathrm{lcp}}}$$

責務間の関連性が低い時，クラス間距離が小さくなると充足度が低下

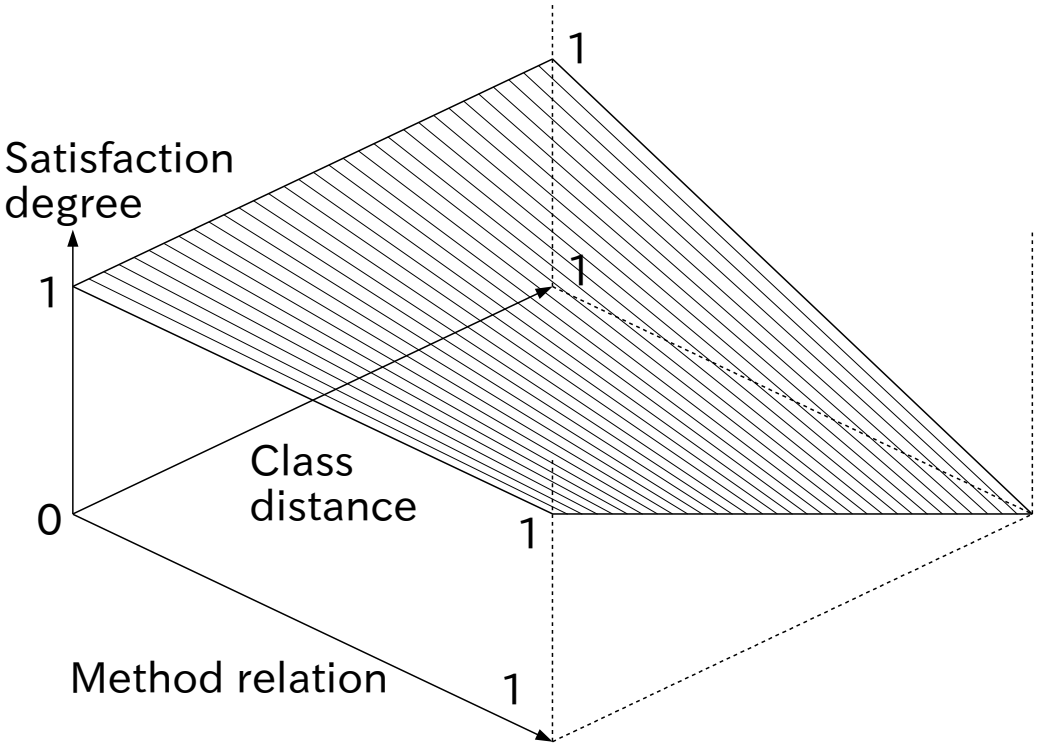
高凝集性の制約（1/2）

- 制約 c_{hch}
  - **関連する責務は出来るだけ近いクラスに……**
  - 全変数のペアに設けられる二項制約
    - 責務間の関連性が高い時に，
      クラス間距離が大きくなると充足度が低下

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

w^{hch}は重み

高凝集性の制約（2/2）



$$w^{\mathrm{hch}} = 1$$

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

責務間の関連性が高い時，クラス間距離が大きくなると充足度が低下

設計維持の制約

- 制約 c_{dc}
  - **出来るだけ設計のクラスと近いクラスに……**
  - 各々の変数に対する単項制約
    - 設計として与えられたクラス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

$$\mu R_{c_m^{\mathrm{dc}}}(k) = \left(1 - cd(k_{\mathrm{orig}}, k)\right)^{w^{\mathrm{dc}}}$$

k_{orig}は設計のクラス，w^{dc}は重み

意図反映の制約

- 制約 c_{same}, c_{diff}
  - **特定の責務は出来るだけ近いクラスに……**
  - **出来るだけ遠いクラスに……**
  - 任意の変数のペアに設定される二項制約

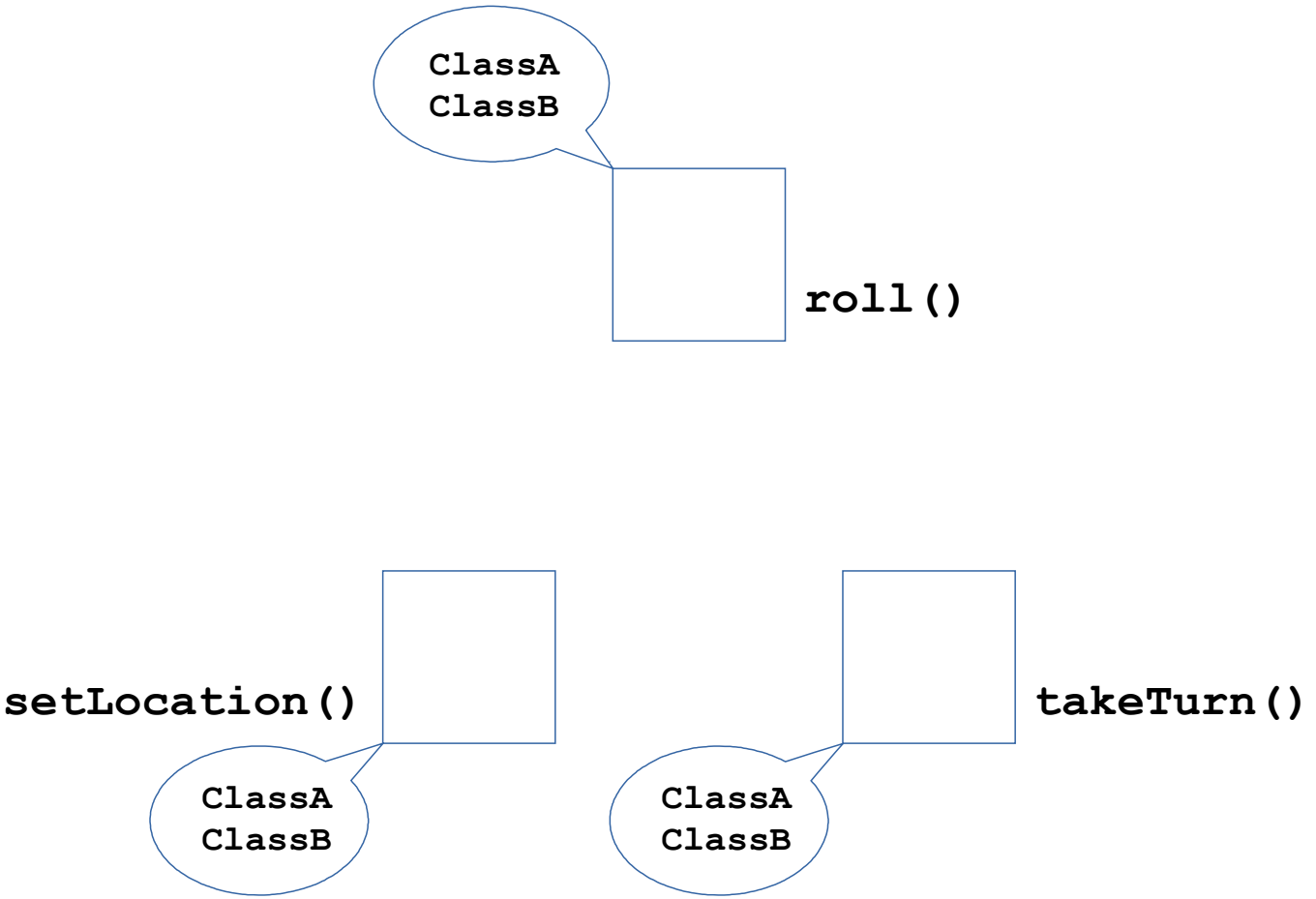
$$\mu R_{c^{\text{same}}_{m_1,m_2}}(k_1,k_2) = (1 - cd(k_1,k_2))^{w^{\text{same}}}$$

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

w^{same}, w^{diff}は重み

定式化の例（2/2）

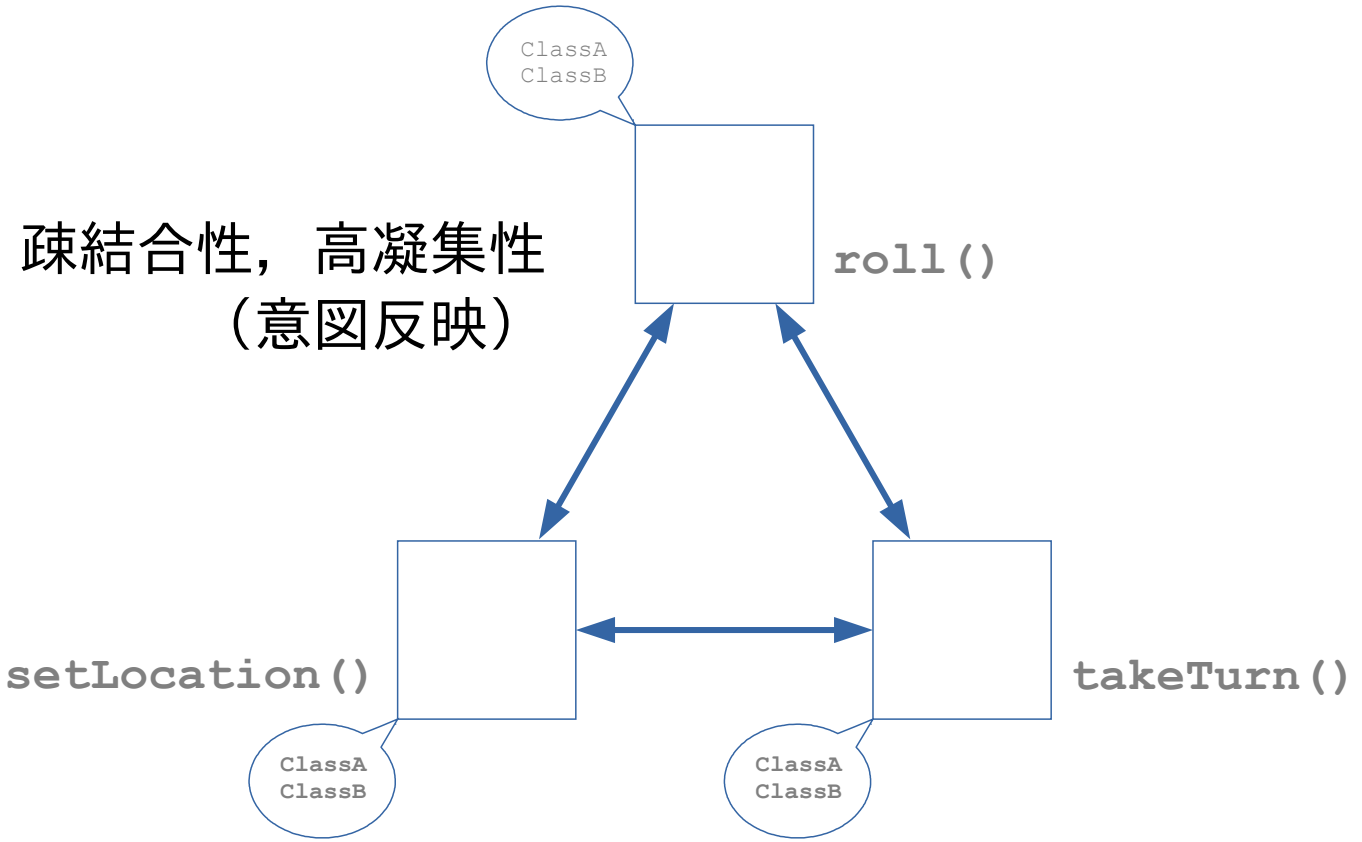
3責務→2クラス



定式化の例（2/2）

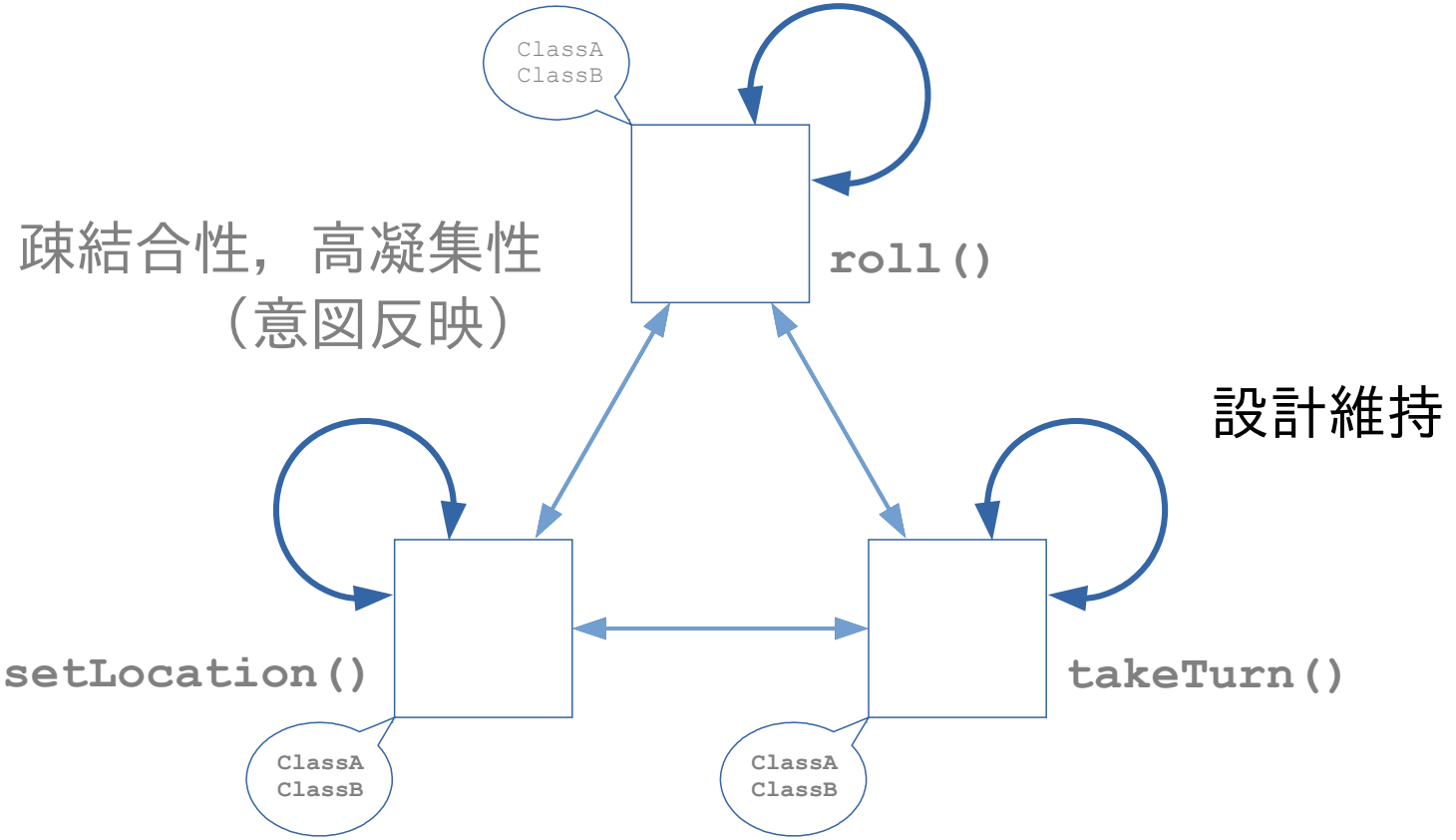
3責務→2クラス

疎結合性，高凝集性
（意図反映）



定式化の例（2/2）

3責務→2クラス



予備実験（1/3）

- 実装
  - 既存のFCSPライブラリ
  - Java 7（Win 7，Intel Core i7，2.93GHz）
  - ソルバー: Fuzzy forward checking
- 設定
  - 重みwを全て1

予備実験（2/3）

- 実験1
  - 0からの割り当てでどれだけ目標設計を再現できるか？
- 実験2
  - 目標設計の責務を1つだけ未割り当てにしたとき，それを元に戻せるか？
- 実験3
  - 意図と異なる割り当てに対してユーザが制約を追加したとき，それが反映されるか？
- 十分な速度で動作するか？（各実験で確認）

予備実験（3/3）

- 対象システム [5]
  - Monopoly（6クラス，26責務）
  - NextGenPos（9クラス，30責務）
    - 既存のクラス図から，
      メソッドを実行責務と見なして抽出
    - 責務間関係については，フィールドも情報把握
      責務と見なし，これとの関係も利用

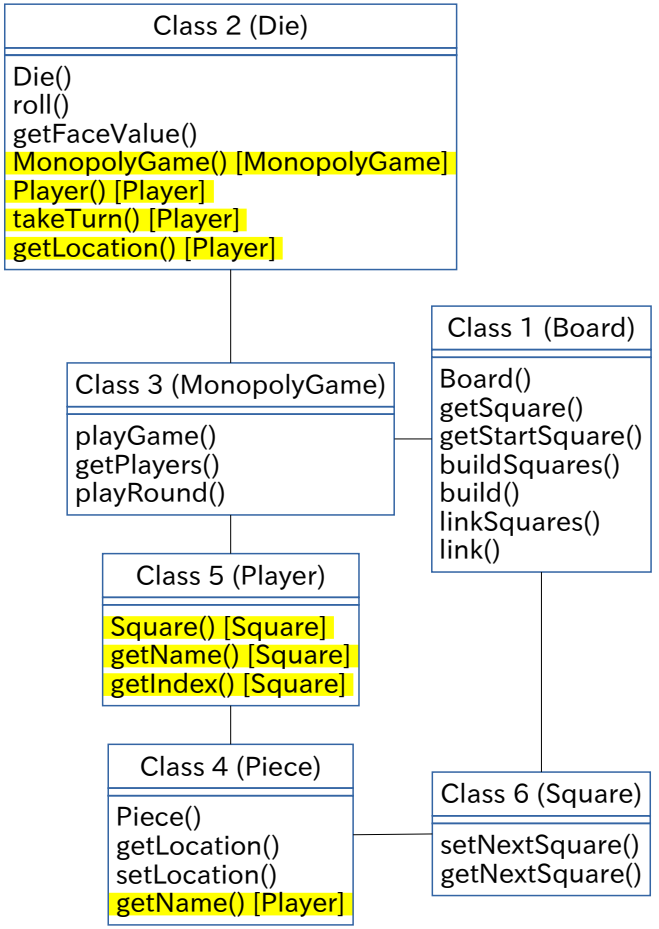
目標設計（望ましい設計）

⟹ 既存のクラス図の責務割り当て

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

実験1（1/2）

0からの割り当てでどれだけ目標設計を再現できるか？



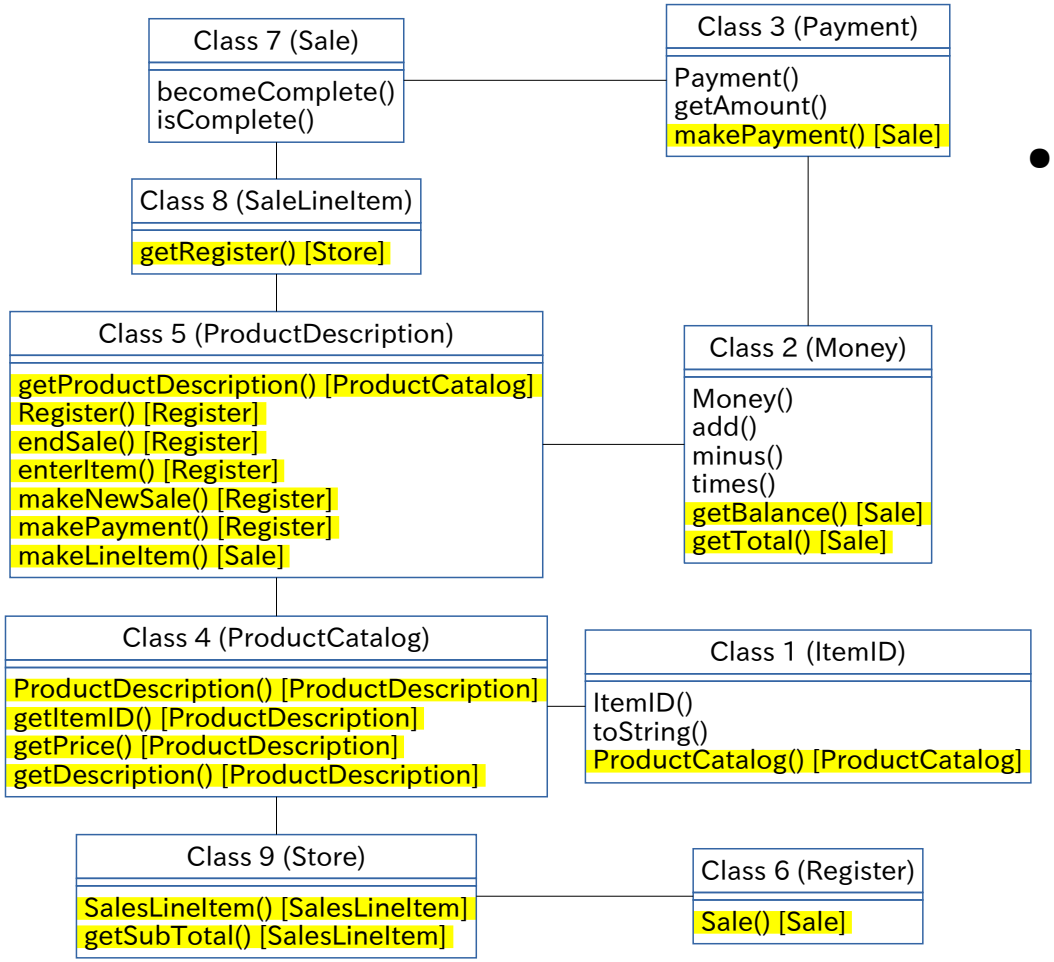
- Monopoly
  - 69%（18個）

それぞれのクラスは無名，括弧内はクラス間関係導出時に用いたクラスの名前

責務名の後の[className]は目標設計の割り当てクラス

実験1（2/2）

0からの割り当てでどれだけ目標設計を再現できるか？



- NextGenPos
  - 33%（10個）

定式化が基礎的な技術としてある程度有効

実験2（1/2）

目標設計の責務を1つだけ未割り当てにしたとき，それを元に戻せるか？

問題の表現

- 再割り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，設計におけるクラスのみにすることによって表現

実験2（2/2）

目標設計の責務を1つだけ未割り当てにしたとき，それを元に戻せるか？

- Monopoly: 58%（15責務）
- NextGenPos: 73%（22責務）

目標設計とは異なる場合

- クラス間距離の近いものが割り当て

より適切なメトリクスの設定によって，定式化手法は有効

実験3（1/2）

意図と異なる割り当てに対してユーザが制約を追加したとき，それが反映されるか？

Monopolyにおいて目標設計と食い違った責務に制約を追加，0から割り当て

- c_{same}: 責務ペア2つに設定
- c_{diff}: 責務ペア1つに設定

実験3（2/2）

意図と異なる割り当てに対してユーザが制約を追加したとき，それが反映されるか？

結果

- c_{same}: 2つ中1つは反映
- c_{diff}: 1つ中1つ反映

ユーザ意図を部分的な制約として追加可能

実行時間

- 実験1（≠実際の利用形態）
  - Monopoly: 20 ms
  - NextGenPos: 8550 ms
- 実験2
  - いずれも1 ms以下
- 実験3
  - 20 ms

十分高速な解導出

考察

- 全制約の重み
  - 今回は経験的に決定
  - ユーザの目的に合わせた設定が可能
- 特定の責務に対する制約
  - アドホックに追加可能

ファジィCSPとして定式化したことによる柔軟性

関連研究（1/2）

- 責務やメソッドの自動割り当て
  - 遺伝的アルゴリズムの利用 [9]
  - メソッドや属性の参照関係に基づくメソッド配置 [10]
- 提案手法とは……
  - 相違点: アルゴリズムや注目点
  - 類似点: 結合性と凝集性に基づいた最適化

[9] M. Bowman, L.C. Briand, and Y. Labiche, “Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms,” IEEE Transactions on Software Engineering, vol.36, no.6, pp.817–837, 2010.
[10] N. Tsantalis and A. Chatzigeorgiou, “Identification of move method refactoring opportunities,” IEEE Transactions on Software Engineering, vol.35, no.3, pp.347–367, 2009.

関連研究（2/2）

- モデル・リファクタリング
  - ステレオタイプの利用 [11]
  - GRASP 等の指針の利用 [13]

複数クラスの相互関係等に関するパターンを対象

提案手法: 大域的な結合性，凝集性に基づく

[11] B. Zamani and G. Butler, “Smell detection in UML designs which utilize pattern languages,” Iranian Journal of Electrical and Computer Engineering, vol.8, no.1, pp.47–52, 2009.
[13] M. Akiyama, S. Hayashi, T. Kobayashi, and M. Saeki, “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.

まとめ

- クラス責務割り当ての自動化
  - ファジィ制約充足問題として定式化
    - 設計のあいまいなルールを自然に表現
    - FCSPの既存の汎用的ソルバを適用可能
    - 今後，FCSPの様々な知見を広く応用可能
  - いくつかの例題への適用

ソフトウェア工学に対する人工知能（FCSP）の応用例

今後の課題

- スケーラビリティの検証
  - 実システムの規模における実用性の検討
  - 適切なFCSP用ソルバーの選定
- 他のソフトウェア・メトリクスの検討
  - 責務間関係のより適切なメトリクスの検討
  - 疎結合性や高凝集性など以外の指針のファジィ制約としての表現

開発者向けのツールとしての実装