社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－SE－154 (9)
2006／11／27

文書間類似度によるソフトウェアパターン間関連分析と複合関連の導出

久保淳人[†] 鷲崎弘宜[††]
高須淳宏[††] 深澤良彰[†]

ソフトウェアパターンとは，ソフトウェア開発において頻出する問題と問題が発生する状況，解法の選択にかかる制約条件，および，具体的な解法の組である．パターンを用いることで熟練技術者の持つ知識やノウハウの共有や再利用を促進することができる．パターンの有機的な利用にはパターン間関連の分析が重要であるが，人手による網羅的なパターン間関連の分析は困難である．本稿では，自然言語処理技術を用いて，ソフトウェアパターン間の関連を自動的に抽出する手法を提案する．本手法を用いることで，人手による分析よりも多数のパターンについて，網羅的な関連分析が可能になる．

Automatic Analysis of Advanced Inter-pattern Relationships

Atsuto Kubo,[†] Hironori Washizaki,[††] Atsuhiro Takasu[††]
and Yoshiaki Fukazawa [†]

By using software patterns, developers can reuse knowledge obtained from experts' empirical rule. Analysis of Inter-pattern relationships is important, but analysis by human costs a lot. In this paper, we propose an automated inter-pattern relationship analysis method. Our method includes a pattern model and two inter-pattern relationship models, and uses several natural language processing techniques such as TF-IDF weighting.

1. はじめに

ソフトウェアパターンは，ソフトウェア開発において頻出する問題に対する実証済みの解法である．熟練したソフトウェア技術者が持つソフトウェア開発に関する知識や経験則をソフトウェアパターンとして記述することで，その知識の共有および再利用が可能になる．現在までに多数のパターンが書籍や World Wide Web(WWW) で公開されている．以下では，ソフトウェアパターンを単にパターンと記述する．

パターンを記述した文書をパターン文書と呼び，パターン文書中の見出し語句と本文の組を項目と呼ぶ．パターン文書の大多数は，図1のような項目集合という構造において共通している．各文書断片は見出し語句によって内容が明示され，カタログ化が容易である．各文書断片の内容を示す見出し語句の構成はパターン形式と呼ばれる．標準的なパターン形式 (Canonical Form[1]) を，表1に示す．

既存パターンの利用プロセスは，「選択」「拡張」「利用」「評価」の4段階に分類できる[2]．我々は，左記の4段階のうち，選択に注目する．パターンの選択には3つの方法がある．

まず，キーワード検索を用いて直面した問題に適したパターンを選択する方法がある．主に検索エンジンや書籍の索引が利用され，検索対象は書籍や WWW 上に存在するパターン文書である．次に，パターン同

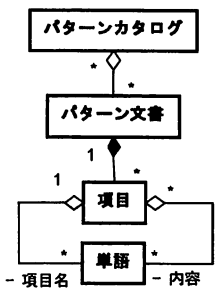

図1 パターン文書モデル

† 早稲田大学
Waseda University
†† 国立情報学研究所
National Institute of Infomatics

士の関連性を利用してパターンを選択する方法がある．パターン間には，2章で示される数種類のパターン間関連の存在が指摘されており，多くのパターン形式で，パターン間の関連性を記述するための項目が存在する．利用者は，パターン間関連に関する記述を参照して，他のパターンとの関係を理解し，パターンを利用する．パターン同士の関連性を利用したパターン選択では，検討対象となるパターン集合の各パターンについて，パターン間の関連性があらかじめ分析されていることが前提となる．第3に，あらかじめ分類されたパターン集合からパターンを選択する方法がある．例えば，Gamma らのデザインパターン[3]では，収録された各パターンを，適用対象の言語要素と，問題の種類の2軸で分類を行った．

検索によるパターン選択は実現が容易であるが，関連パターンなど，パターンに特有の情報を扱うことができない．分類によるパターン選択は，情報量が多く利用しやすいが，分類方法が問題領域に特化しているため，計算機による選択支援を行うのが困難である．パターン間関連によるパターン選択では，各パターン間に汎用的なパターン間関連が指摘されているため，統一された方法で計算機による支援を行うことができる．したがって，本稿ではパターン間関連によるパターン選択を扱い，計算機を用いてパターン間関連によるパターン選択を支援する手法を提案する．

以下の章では，まず既存のパターン間関連を分類し，次にそれらを表現可能なパターンモデルとパターン間関連モデルを提案する．Gamma らのデザインパターンについて提案手法を適用する実験を行い，結果を考察する．最後に，提案手法の長所・短所について考察する．

2. パターン間関連

パターン間関連に関する研究は多数存在する．Gamma らはパターン間関連を有向グラフの辺として表現した[3]．Zimmer は，Gamma らのデザインパターンについて3種類の関連を新たに提案した[4]．同様に，Buschmann らは3種類の汎用的なパターン間関連を提案した[5]．Noble らは，Primari Relationship として3種類の汎用的なパターン間関連と，Secondary Relationship として9種類の汎用的なパターン間関連を提案した[6]．Volter らは，サーバサイドコンポーネントに関するパターンカタログ中で，3種類の汎用的なパターン間関連を提案した[7]．各パターン間関連の詳細を表2に示す．

表1 Canonical Form

項目名	内容
Name	パターン名
Problem	パターンが解決する問題
Context	問題が発現する状況
Forces	問題解決のための制約条件
Solution	具体的な解法・指針
Resulting Context	パターン適用後の文脈
Known Uses	適用例
Related Patterns	関連パターン

問題の単純化のため，本稿では，2つのパターンの間の関連のみを扱う．そのため，多数のパターン間の関連である Ns9 と，パターン自身との関連である Ns8 は除外する．また，Np1 と Ns1〜Ns6 は表中の他の文献が出典になっているため除外する．さらに，Np2 と B1 は同じ関連であるので，B1 に統合する．以上から，本稿では，Z1(利用)，Z2(変形による利用)，Z3(類似)，B1(洗練)，B2(変形)，B3(組み合わせ)，Np3(衝突)，Ns7(要求)，V1(文脈提供)，V2(依存)，V3(特化)について検討する．

3. パターンモデル

本章では，前章で列挙したパターン間関連を表現可能なパターン関連モデルを作成する．分析対象のパターン記述は既に得られていることを仮定する．つまり，表3の形式でパターン記述が得られていることが前提である．パターン記述を得る方法は特に限定しない．例えば，HTML の構造を利用してパターン文書からパターン記述を抽出する手法がある[8]．

パターンには，パターンの適用前後でパターンの適用対象となった設計等がどのような状況であるか，という情報が記述される．具体的には，パターン適用前に関する記述は，表1では Context(文脈) の項目に含まれ，パターン適用後に関する記述は Resulting Context(結果文脈) の項目に含まれる．本稿では，この対比を明確にするために，上述の Context(文脈) を，特に開始文脈と呼ぶ．

同一の問題であっても，制約条件により最適な解法は異なる．整列アルゴリズムの選択であれば，要素を整列するという同一の問題に対して，バブルソートやクイックソートなど複数の整列アルゴリズムが存在する．これらのアルゴリズムの差異は制約条件となって現れる．例えば，バブルソートは低速である反面，実装が容易である．逆にクイックソートは高速に整列できる反面，実装難度はやや高い．このとき，制約条件が「整列の速度は重要ではなく，実装難度が低いことが重要である」であれば，バブルソートが最適であると判断できる．パターンには，上述のような判断を

表 2 既存の汎用パターン間関連

提案者	記号	名前	内容
Zimmer	Z1	利用	パターン X は，その解法の中でパターン Y を利用する
	Z2	変形による利用	パターン X の変形がパターン Y を利用する.
	Z3	類似	パターン X とパターン Y は類似している
Buschmann	B1	洗練	パターン X は，より小さなパターン Y の組み合わせで表現可能である
	B2	変形	パターン Y は，パターン X の変形である.
	B3	組み合わせ	パターン X とパターン Y は組み合わせて利用できる
Noble	Np1	利用	あるパターンが他のパターンを利用する (Z1 と同)
	Np2	洗練	特化したパターンが，汎用的なパターンを洗練する
	Np3	衝突	あるパターンは他のパターンと同じ問題を扱う.
Noble	Ns1	被利用	(Np1 の逆)
	Ns2	被洗練	(Np2 の逆)
	Ns3	変形	あるパターンはよく知られた他のパターンの変形である. (B2 と同)
	Ns4	変形による利用	パターンの変形が他のパターンを利用する. (Z2 と同)
	Ns5	類似	パターン X とパターン Y は類似している. (Z3 と同)
	Ns6	結合	パターン X とパターン Y を組み合わせることで問題を解決する. (B3 と同)
	Ns7	要求	あるパターンが他のパターンの解法を要求する.
	Ns8	タイリング	パターンが自分自身を利用する.
	Ns9	詳細化	単純から複雑にいたるパターンの列.
Volter	V1	文脈提供	パターン X はパターン Y に文脈を提供する
	V2	依存	パターン X はパターン Y に依存する
	V3	特化	パターン Y はパターン X の特化である

表 3 Composite パターン

カタログ	パターン	見出し	内容
GoF	Composite	Context	You want to represent whole-part hierarchies. You don't want clients to know the difference between compositions (nodes) and individual (leaf) objects ...
GoF	Composite	Problem	Compose objects into tree structures to represent whole-part hierarchies in a manner that lets clients treat objects and compositions uniformly. ...
GoF	Composite	Solution	Create an abstract Component class which defines the interface of all tree objects and implements default common behavior. ...
GoF	Composite	Resulting Contexts	A near pure implementation of 1:N Recursive Connection, thus gains its advantages (and drawbacks, of course!) ...
GoF	Composite	Related Patterns	Composite and Decorator have similar structure diagrams, reflecting the fact that both rely on recursive composition to organize an open-ended number of objects. ...

行うための情報が Forces(制約条件) の項目に記述される.

上記の議論から，パターンは 3 つの項目 (開始文脈, 制約条件，結果文脈) を持つものとする. 構造はパターン文書モデル (図 1) と同じであるが，持つことのできる項目は，開始文脈，制約条件，結果文脈のみである. 項目ごとの対応関係を指定することで，パターン文書からパターンを得ることができる. 項目ごとの対応関係の例を表 4 に示す.

パターンモデルモデルをラベル付有向グラフとして表現したものが図 2 である. 頂点が文脈，辺がパターンの適用，ラベルが制約条件をそれぞれ表す.

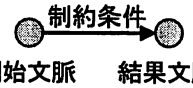

図 2 パターンモデルのラベル付有向グラフ表現

表 4 パターン項目の対応関係

パターン文書の項目名	対応するパターンの項目
Context, Applicability	開始文脈
Force, Pressure, Motivation	制約条件
Consequences, Resulting Context, Benefits	結果文脈

4. パターン間関連モデル

本章では，前章で得られたパターンモデル上でパターン間関連を表現する. 最初に共通して用いられる要素技術を列挙し，続いて 2 種類のパターン間関連を提案する.

4.1 文書間類似度

後述されるパターン間関連モデルは，すべて文書間類似度を基礎にしている．本稿では，文書間類似度の算出法として，TF-IDF 法による単語重み付けと，ベクトルモデル上の余弦類似度を用いる．

パターン文書は，単語列から構成される

TF-IDF 法は，文書中の単語重みの算出手法の一つで，単語の出現頻度 (Term Frequency) と単語による文書の特定性 (Inverse Document Frequency) の積を単語の重みとする手法である．ある文書 d 中の単語 t の出現回数を $tf(d,t)$ とし，ある単語 t が出現する文書数を $df(t)$，文書数を N する．このとき，単語 t の文書 d における重み $tfidf(d,t)$ を，

$$tfidf(d,t) = tf(d,t) \cdot \left(\log \frac{N}{dt(t)} + 1 \right)$$

と定義する．上式より，文書 d について単語重み $tfidf(d,t)$ の値によるベクトル $w(d)$ が得られる．2 つの文書 d_1, d_2 について，それぞれ $w(d_1), w(d_2)$ が得られ,d_1, d_2 間の類似度 $s(d_1, d_2)$ は，ベクトルの余弦値を用いて以下の式で定義される．

$$s(d_1, d_2) = \frac{w(d_1) \cdot w(d_2)}{|w(d_1)|\,|w(d_2)|}$$

また，パターン文書のほとんどは英語で記述されているため，英語に特化した自然言語処理技術を用いる．不要誤除去処理では，特定の意味を持たない語 (冠詞，be 動詞) を不要語として除去する．接辞処理では，語尾が変化 (3 人称単数現在，複数形，分詞など) した語を原形に戻す．

4.2 基本パターン間関連

基本パターン間関連として，表 5 に示す 4 種類を提案する．各関連が存在すると判定するための条件が「条件」の列に記述される．閾値を変化させることで，パターン間関連検出の感度を調整することができる．最右列には，各パターン間関連と既存のパターン間関連との対応関係が示されている．

4.3 複合パターン間関連

上で得られた基本パターン間関連の組み合わせを用いて，表 6 に示される複合パターン間関連を得ることができる．記号は，複合パターン間関連を特定する名前である．SS・RS・RR・FF の各欄は，適当な閾値を設定したとき，対応する基本パターン関連が存在する場合に○，しない場合に×となる.「意味」の欄は各関連の意味付けである．2 章で挙げた関連のうち，多くの関連について提案モデル上の関連として表現可能であることがわかる．

関連 Same 2 つのパターンは強く類似してるか，同一のパターンである．

関連 SubInContext 一方の開始文脈と，他方の開始・結果文脈が類似している．他方のパターンの記述内容が一方のパターンの適用前の状況に完全に含まれると解釈される．

関連 SimilarOverall 開始文脈同士・結果文脈同士がそれぞれ類似しているので，単なる類似パターンだと解釈される．

関連 SimilarProblem 開始文脈同士が類似しているので，類似した問題に対する異なる解決を提供するパターンである．基本パターン間関連 SS と同一である．

関連 SubInResulting 一方の結果文脈と，他方の開始・結果文脈が類似している．他方のパターンの記述内容が一方のパターンの適用後の状況に完全に含まれると解釈される．例えば,「Model-View-Controller パターンは，その解法で Observer パターンを利用する」という関連がこの関連に相当する．

関連 Continuous 基本パターン間関連 RS と同一で，連続して適用できる可能性がある．

関連 SimilarResulting 結果文脈同士が類似しているので，異なる問題に対して適用すると類似した結果をもたらす．基本パターン間関連 RR と同一である．

関連 Corresponds 文脈は類似していないが制約条件が類似している場合で，異なる問題領域において，類似した制約条件のもとで適用可能なパターンである．Flyweight パターンは，記憶領域の節約が求められる場合に利用できる．また，プログラミング言語固有の記憶領域を節約する技法も存在する．これらのパターンは，問題領域は異なっているものの，記憶領域の節約という制約条件が共通しているため，開発のそれぞれの段階で利用を検討することができる．

関連 None 2 つのパターン間に関連は存在しない．

Gamma らのデザインパターンカタログ内で用いられているパターン間関連について，著者らが分類を行った結果を表 7 に示す．例えば表の第 1 行では，Abstract Factory パターンと Factory Method パターンの関連について述べられているが，その関連は，提案した複合パターン間関連のうち SubInResulting に相当する．

5. 実　　験

Java 言語を用いて提案手法を実現するシステムを

表 5 基本パターン間関連モデル

関連名	記号	条件	意味	対応する既存の関連
開始文脈の類似	SS	2 つのパターンの開始文脈同士の類似度が別に設定する閾値より高い	2 つのパターンは類似した問題を扱う	Z3 B2
結果文脈の類似	RR	2 つのパターンの結果文脈同士の類似度が別に設定する閾値より高い	2 つのパターンは，似た適用結果をもたらす	Z3
結果・開始文脈の類似	RS	パターン p_1 の結果文脈と，他のパターン p_2 の開始文脈との類似度が，別に設定する閾値より高い	p_1 の後に p_2 を適用できる可能性がある	Ns7 V1
制約条件の類似	FF	2 つのパターンの制約条件同士の類似度が別に設定する閾値より高い	問題領域や粒度が異なっていても，類似した制約条件のもとで適用できる．	B3

表 6 複合パターン間関連モデル

記号	SS	RS	RR	FF	意味	対応
Same	○	○	○	○/×	同一パターン (強い類似)	
SubInContext	○	○	×	○/×	一方が他方の開始文脈に含まれる	
SimilarOverall	○	×	○	○/×	問題と結果が類似	Z3
SimilarProblem	○	×	×	○/×	問題のみ類似	B2 Np3
SubInResulting	×	○	○	○/×	一方が他方の結果文脈に含まれる	Z1 B1 Np2
Continuous	×	○	×	○/×	連続して適用できる可能性がある	Ns7 V1
SimilarResult	×	×	○	○/×	結果のみ類似	
Corresponds	×	×	×	○	異なる問題領域で対応するパターン	B3
None	×	×	×	×	関連なし	

表 7 提案関連モデルによる GoF デザインパターン間関連の分類 (一部)

パターン間関連記述	筆者らによる分類
Abstract Factory クラスは，しばしば factory method を使って実装される	SubInResulting
factory method は通常，template method の中で呼ばれる	SubInResulting
Visitor パターンは，Composite パターンで定義されるオブジェクト構造上にオペレーションを適用するために使うことができる	Corresponds
Mediator パターンは，既存のクラスの機能を抽出しているという点で Facade パターンと似ている	SimilarOverall
command がその実行結果を取り消すことができるように，Memento パターンで状態を保存しておくことができる	SubInResulting
Interpreter パターンでは，Visitor パターンを言語の解釈のために適用してもよい	SubInResulting

作成し，実験を行った．実験に先立って，Gamma らのデザインパターン間の関連を分類した (表 7)．実験中では，このデータを仮の正解として評価を行った．不要語リストとして SMART によるリスト[9] を，接辞処理手法として Paice の方法[10] 用いた．

5.1 基本パターン間関連の分析実験

Gamma らのデザインパターンを記述したパターン文書[11] を WWW から取得し，提案手法を適用した．得られた基本パターン間関連のうち，上位 10 件を表 8 に示す．「原著中の言及」列は，3) 内でパターン間関連が存在する旨の指摘がある場合に○，そうでない場合に×とした．

得られたパターン間関連のうち，強さが閾値 0.18 以上の関連について，パターン間関連を示したグラフを作成した (図 3)．ラベル付の矩形が一つのパターンを表す．パターン中には，楕円で表される文脈が存在し，開始文脈と結果文脈は矢印で結ばれている．破線は得られたパターン間関連を表し，ラベルに関連の種類と強さが表示される．文脈同士の関連は文脈を直接連結し，制約条件同士の関連はパターンの矩形同士を連結する．例えば，Abstract Factory パターンと Factory Method パターンの結果文脈が類似していることは，両パターンの結果文脈を表す楕円が破線で連結されていることから分かる．

表 9 に，Abstract Factory と Factory Method の結果文脈を示す．product や creation といった単語は，共通していたため，結果文脈同士の類似が検出された．

5.1.1 考　　察

実験 1 では，上位においては，Gamma らが主張するパターン間関連を正確に検出できた．例えば，Visitor パターンと Iterator パターンは，要素の走査処理を扱う点で共通しているし，Mediator パターンの実現として Observer パターンを用いることができる．

表中下位のパターン組については，Gamma らの言及がないにも関わらず，パターン間関連が検出された．

表 8 GoF デザインパターン間の基本パターン間関連

パターン 1	パターン 2	種類	強さ	原著中の言及
Visitor	Iterator	SS	0.27	○
AbstractFactory	FactoryMethod	RR	0.26	○
FactoryMethod	TemplateMethod	RR	0.23	○
Composite	Visitor	FF	0.23	○
Mediator	Observer	RS	0.23	○
Mediator	Facade	RR	0.20	○
Interpreter	Visitor	RR	0.19	○
FactoryMethod	Prototype	RR	0.19	○
Iterator	Prototype	RR	0.19	×
Visitor	Prototype	RR	0.18	×

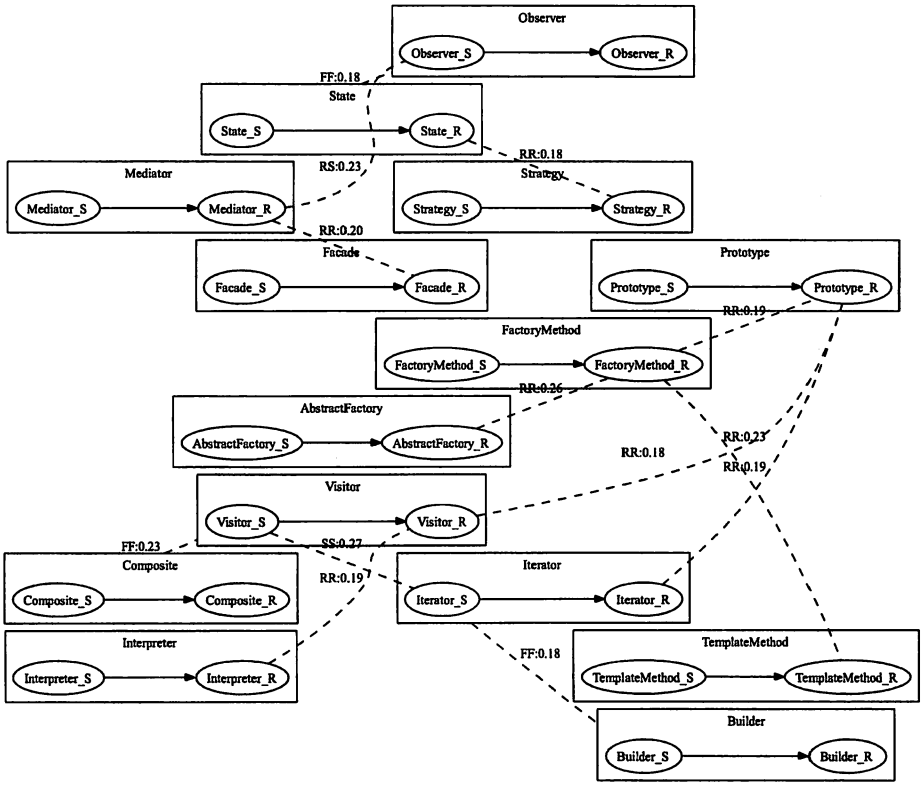

図 3 提案手法によって得られた GoF デザインパターン間の基本パターン間関連

表 9 Abstract Factory パターンと Factory Method パターンのパターン記述 (抜粋)
太字は著者によるもので，2 つの文書断片間に共通する単語を示す

パターン	項目	内容
Abstract Factory	結果文脈	Isolates client code from **object creation**. Makes exchanging **product** families easy – client code written to work with one family of **products** can be changed to work with a completely **different** set of **products** with little or no modification. Supporting new categories of **products** is **difficult** – the set of **products** is fixed by the AbstractFactory. ...
Factory Method	結果文脈	Factory methods may **create** multiple kinds of **objects** in a single function; however, this will make the FactoryMethod code more complex and may make it more **difficult** to extend the **Creator** class to support new **product object** types. Factory Method is to **creating objects** as Template Method is to implementing an algorithm. ...

Iterator-Prototype の組については，実際には検出されたようなパターン間関連は無く，誤検出であると考えられる．Visitor-Prototype の組は，振る舞いが実行時の型によって決定されるようになる点において共通する．提案手法を用いることで，Gamma らが言及しなかったパターン組について関連を示唆することができた．

5.2 複合パターン間関連の分析実験

実験 1 と同様，WWW から取得した Gamma らのデザインパターン文書について，提案手法を適用した．基本パターン間関連の有無を判定する閾値は 0.14 を用いた．得られた複合パターン間関連のうち，上位 20 件を表 10 に示す．「原著中の言及」列は，3) 内でパターン間関連が存在する旨の指摘がない場合×，原著中で指摘され，その関連の種類も含め正しく検出した場合○，検出した関連の種類が異なる場合は△とした．

5.2.1 考　　察

20 個の関連中，9 個の関連 (○印と△印) は，Gamma らが指摘したパターン間関連を検出した．また，同じく 3 個の関連 (○印) は，Gamma らが指摘したパターン間関連を，その種類も含めて正しく検出した．しかし，検出された他の関連はいずれも Gamma らによって指摘されていない関連だった．

表 10 中，Visitor パターンと Composite パターンの組について，提案手法により関連 Corresponds が示された．Visitor パターンは，Gamma らによって Composite パターンで表現される木構造の走査に用いることが指摘されているが，提案手法はそのことを正しく検出した．

Abstract Factory - Factory Method の組については，原著中では SubInResulting に相当する関連が示されているが，システムは SimilarResult の関連であると検出した．2 つのパターン間の関連 RS の重みが，実験で設定した閾値より小さかったため，異なる関連が検出された．閾値を 0.10 程度に設定すると，正しく検出された．

6. 関 連 研 究

直面した問題に適したパターンの選択を支援する試みは，本稿で示したパターン選択方法のうち，検索によるパターン選択[12)13)] と分類によるパターン選択[3)14)] に関しては研究が存在する．対して，提案手法はパターン間関連を基礎にしたパターン選択を支援する．

Borchers は，パターンを項目の集合とするモデルを提案している[15)]．しかし，Borchers らのモデルは，問題や解法を表す項目が存在するとしただけで，具体的な利用にはふれておらず，標準的な項目の構成 (パターン形式) も示されていない．本稿のパターンモデルでは，Borchers らのモデルと同様の構造を持つが，パターン間関連分析のために項目の構成を特化させた．

2 章で挙げたように，パターン間関連に関する研究は多数存在する[3)~7)]．本稿では，上述の関連のうち多くの関連を表現可能なパターン間関連モデルを提案した．パターン間関連の利用として，有向関連による参照関係からパターンの抽象度を算出する手法も提案されている[16)]．

7. おわりに

ソフトウェアパターンの選択は，今後のパターン利用において重要な位置を占めるが，パターン選択支援に関する試みはあまり多くない．パターン選択には，検索によるパターン選択，パターン間関連によるパターン選択，分類によるパターン選択が考えられる．検索によるパターン選択と分類によるパターン選択は広く行われている．対して，パターン間関連の人手による網羅的な分析が難しいため，パターン間関連によるパターン選択はあまり行われていない．そこで，我々は自動的なパターン間関連分析手法を提案した．提案手法は，パターン間関連によるパターン選択支援の基礎となることが期待される．

提案手法には，以下の長所がある．まず，自然言語によるパターン記述を分析するため，どのようなパターン文書でも扱うことができる点がある．そのため，問題領域や開発工程を横断したパターン間関連分析を行うことができる．また，提案パターンモデルでは，既知の汎用的なパターン間関連の多くを検出可能である．

逆に，提案手法には以下の短所や制限がある．まず，利用する自然言語処理技術に分析結果が強く依存する．特に，作者による表現や語彙の差異が影響する可能性が高い．また，文書間類似度を基礎としているため，全く異なるパターンの有機的な組み合わせを発見するのが困難である．例えば，Command パターンによるコマンドオブジェクトを，Composite パターンを用いて木構造化しマクロコマンドとする組み合わせがある．しかし，Composite パターンと Command パターンとの文書間類似度は低いため，提案手法では上述の組み合わせを検出するのは困難である．さらに，分析の過程で図による情報を捨てるため，デザインパターンのクラス構造などを考慮した分析は不可能である．最後に，複合パターン間関連では，閾値の設定の変化に対して，検出されるパターン間関連の種類が大きく変

表 10 GoF デザインパターン間の複合パターン間関連

パターン 1	パターン 2	関連の種類	重み	原著中の る言及	原著中での 関連の種類
AbstractFactory	FactoryMethod	SimilarResult	0.244	△	SubInResulting
TemplateMethod	FactoryMethod	SimilarResult	0.225	△	SubInResulting
Visitor	Composite	Corresponds	0.202	○	Correnponds
Mediator	Facade	SimilarResult	0.193	△	SimilarOverall
Visitor	Command	SimilarResult	0.180	×	
Iterator	Strategy	SimilarResult	0.173	×	
Iterator	Builder	Corresponds	0.171	×	
State	Strategy	SimilarResult	0.170	×	
Strategy	FactoryMethod	SimilarResult	0.166	×	
Singleton	Proxy	Corresponds	0.166	×	
Visitor	FactoryMethod	SimilarResult	0.160	×	
Strategy	AbstractFactory	SimilarResult	0.159	×	
Memento	Command	Corresponds	0.157	△	SubInResulting
State	Observer	Corresponds	0.157	×	
Decorator	Bridge	Corresponds	0.156	×	
Visitor	Iterator	Same	0.156	△	SubInResulting
ChainOfResponsibility	Command	Corresponds	0.155	×	
Interpreter	Visitor	SubInResulting	0.151	○	SubInResulting
Singleton	Prototype	SimilarResult	0.149	△	SubInResulting
Mediator	Observer	SubInResulting	0.148	○	SubInResulting

○ . . . 同種の関連が原著で言及されている，△ . . . 異種の関連が原著で言及されている，× . . . 原著では言及されていない

化してしまう．これは，複合パターン間関連の存在判定を，基本パターン間関連の存在の組み合わせ，つまり，ブール値の組み合わせとしたのが原因であると考えられる．

今後の課題としては，提案手法を利用したパターン選択支援システムの構築と評価を行い，特に閾値の設定についての議論を行いたい．また，複合パターン間関連のグラフ表現の提案も検討している．

参 考 文 献

1) Cunningham & Cunningham, I.: Canonical Form. http://c2.com/cgi-bin/wiki?Canonical Form.
2) 鷲崎弘宜，深澤良彰：ソフトウェアパターン研究の現在と未来，情報処理学会第 141 回ソフトウェア工学研究会， Vol.2003, No.55 (2003).
3) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley (1994).
4) Zimmer, W.: Relationships between Design Patterns, *Pattern Languages of Program Design Vol.1*, Addison-Wesley, pp.345–364 (1995).
5) Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M.: *Pattern Oriented Software Architecture: A System of Patterns*, Wiley, New York (1996).
6) Noble, J.: Classifying relationships between object-oriented design patterns, *Proceedings of Australian Software Engineering Conference (ASWEC)*, IEEE CS Press (1998).
7) Volter, M.: Server-Side Components - A Pattern Language, *proceedings of EuroPLoP '2000* (2000).
8) Kubo, A., Washizaki, H., Takasu, A. and Fukazawa, Y.: Extracting Relations among Embedded Software Design Patterns, *Journal of Design & Process Science*, Vol. 9, No. 3 (2005).
9) Salton, G. and McGill, M. J.: *Introduction to Modern Information Retrival*, McGraw-Hill Inc., New York (1983).
10) Paice, C.D.: Another Stemmer, *SIGIR Forum*, Vol.24, No.3, pp.56–61 (1990).
11) Huston, V.: Huston Design Patterns. http://home.earthlink.net/huston2/dp/patterns.html.
12) Rising, L.: *The Pattern Almanac 2000*, Addison-Wesley (2000).
13) Cunningham &Cunningham, I.: Portland Pattern Repository. http://c2.com/ppr/.
14) Cunningham, W. et al.: PatternShare. http://patternshare.org/.
15) Borchers, J. O.: A Pattern Approach to Interaction Design, *AI and Society Journal of Human-Centred Systems and Machine Intelligence*, Vol.15, No.4, pp.359–376 (2001).
16) 久保淳人，鷲崎弘宜，深澤良彰：ソフトウェア設計パターンの抽象度測定法，第 14 回ソフトウェア工学の基礎ワークショップ (FOSE2006) (2006).