推薦論文● FOSE2005

重要度算出に基づくソフトウェアパターン検索システム

中山 弘之　久保 淳人　鷲崎 弘宜　深澤 良彰

現在 Web 上などで，電子化された大量のソフトウェアパターン (以降，パターンと略記) 文書が公開されている．しかしながらパターン文書に特化した検索システムは存在しないため，パターン利用者は汎用的な検索システムなどを利用し手動で必要なパターンを検索しなければならない．汎用的な検索システムを用いてパターンを検索する場合，検索結果にパターン文書以外の文書が含まれる．また，大量の検索結果が得られた場合，無作為に必要なパターンを探すのは非効率的である．我々は，同問題に対する解決法としてパターン検索システムを提案する．提案システムは，パターン文書を収集したリポジトリに対しパターン間の関係によるパターン文書への重要度の付加と並び替えおよびキーワード検索を実行する．提案システムの有効性を検証するため，Web 上に公開されている 131 個のパターン文書に対し，提案システムによる検索実験を行った．実験の結果，提案システムを利用することによりパターンを効率的に検討・学習することが可能となることを確認した．

There are many catalogs of software patterns available on Web and literature; however, conventional document search systems are not appropriate for retrieving objective patterns from a number of collected patterns because these systems are not specific to software patterns (e.g., these systems simply present many pattern documents as a result of searching with a user's query). To solve this problem, we propose a search system for software patterns. Our system is based on inter-pattern link analysis and link-based importance calculation technique, called "pattern rank". As a result of experimental evaluations for 131 pattern documents, it is found that our system is useful for retrieving patterns that are related to users' intension/requirement by calculating an appropriate importance for each pattern.

1　はじめに

ソフトウェアパターン (以降，パターンと略記) とは，ソフトウェア開発の各局面において繰り返し現れる問題に対する考慮すべき制約を伴った解法/指針である [14] [20]．パターンを利用することにより，ソフトウェア開発における分析工程や設計工程で頻繁に生じる問題を効率的に解決することが可能である．また，パターンを学習することで，個々のパターンが扱う問題の背景に存在するソフトウェア開発手法への理解を深めることが可能である．例えばデザインパターンの学習によるオブジェクト指向設計の習得などが挙げられる．Gamma らによるデザインパターン [3] の発表以来，Hillside Group などのパターンコミュニティの活動により，パターンの数は増加し，現在，多くのパターン文書が文献や Web 上で電子文書として公開されている [9]．

パターン文書の増加に伴い，パターンに特化し，大量のパターン文書集合から利用者の意図や要求に合致し，参考となるパターンを効率よく探し出すことが可能な検索システムの必要性が増大している．パターンに特化した検索システムにより，ソフトウェア開発におけるパターンの利用やパターンの学習を効率よく行うことが可能になる．このとき，システムの構築に

A Search System for Software Patterns based on Pattern Rank.

Nakayama Hiroyuki, Atsuto Kubo, 早稲田大学大学院理工学研究科, Dept. of Computer Science, Waseda University.

Hironori Washizaki, 国立情報学研究所／総合研究大学院大学, National Institute of Informatics/The Graduate University for Advanced Studies.

Yoshiaki Fukazawa, 早稲田大学理工学部, Dept. of Science and Engineering, Waseda University.

コンピュータソフトウェア, Vol.16, No.5 (1999), pp.78–83.

[論文] xxxx 年 yy 月 zz 日受付.

おいては検索精度の問題と検索効率の問題が生じる．検索精度の問題は，パターンを記述した文書であるパターン文書のみを収集したパターンリポジトリを準備することで解決できる．これにより，検索結果としてパターン文書のみを得ることができるため，一般の検索システムを利用した場合の雑音を排除できる．

また，リポジトリ内のパターン文書数の増加に伴って検索結果のパターン文書数が増加し，大量の検索結果全てについて有用性を検討することが現実的でなくなり検索効率が悪くなる問題が生じる．

検索効率の問題の解決には，検索結果を絞り込み利用者の意図や状況に応じて順序付けや分類を行う仕組みが有効である．

検索効率の向上を望める既存の手法として，リポジトリへのパターン登録時にパターンのメタ情報 (例えば適用対象領域など) を人手で記録し，メタ情報を利用して分類/検索する手法が提案されている [15] [17]．しかしながら既存の検索手法では，検索結果として同一の特徴を有する大量のパターン集合が得られた場合に，利用/学習を検討すべきパターンを選択することは困難であるる．

検索結果が大量に得られる問題の例として，システム開発時に Graphical User Interface(GUI) のウィンドウ設計に利用可能なパターンを得たい場合を想定する．Portland Pattern Repository [1] において検索クエリ”window”を入力しタイトル検索を行った．検索結果として，リポジトリ内に 32029 個存在する文書のうち，名前に基づいて順序付けられた 99 個の文書を得た．しかしながら，名前による順序付けは，パターンの有用性という面から考えると無作為に並べられているのに等しく，また検索結果として得られた文書集合はパターン文書以外の文書 (例えばパターンに関する記事など) を含むため，利用者は必要なパターンを得るために，最大 99 個の文書を読まねばならない．

本稿では，HTML を用いて記述されたパターン文書からパターン間の参照情報を自動的に抽出し，抽出した参照情報に基づいてパターンの重要度を自動的に算出する手法を提案する．提案手法をパターンランク法と呼ぶ．パターンランク法は，重要度算出手法について Web ページの重要度算出手法 PageRank [11] やソフトウェア部品の重要度算出手法である ComponentRank [7] と類似する．しかしながら，パターンランク法は同一パターンの異なる文書表現や同一パターンカタログへの帰属などのパターン特有の情報を考慮する点が異なる．

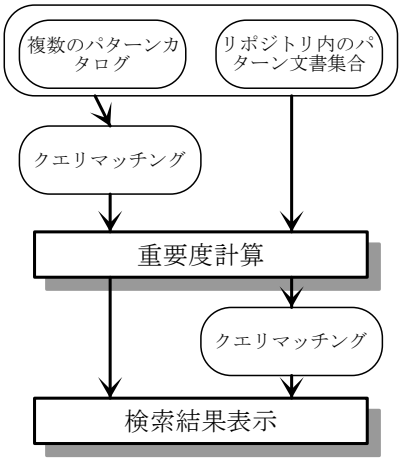

図 1　提案システムにおける処理の流れ

パターンランク法の応用として，利用者から与えられた検索クエリを含むパターン集合をパターンランク法の適用により得たパターン重要度で順序付けし，提示する検索システムを提案する．提案システムでは，検索クエリを持つパターン集合に対し，パターンランクによる重要度計算を行う「クエリマッチング後に重要度算出」と，リポジトリ内のパターン集合全体に対しパターンランク法による計算を行う「重要度算出後にクエリマッチング」の二つの処理を持つ．また検索対象となるパターン文書集合は，利用者が，複数のパターンカタログ，あるいは，リポジトリ内に存在する全てのパターン文書のどちらかを選択する (図 1)．

提案システムは，前述した規模の増大に伴う問題に対し，パターン文書のみを事前収集したリポジトリの構築により高い検索精度を実現する．さらに，パターンランク法を用いたパターンの重要度算出に基づく，パターン文書集合の順序付けにより，効果的な検索を実現する．

2 パターンの参照関係と検索システムの必要性

一般にパターンは，パターン名，前提条件/状況，問題，フォース (考慮すべき制約)，解法，関連パターンといった項目集合からなる形式 (パターン形式) に従って記述される [20]．パターンの記述例を図 2 に示す．図 2 では，デザインパターンの一種である `Singleton` パターン [3] が記述されている．

2.1 パターン間の参照関係

パターンは単独で成り立つものではなく，他の関係するパターンと共にパターンの体系を形成し，関係するパターンと協調して働くことにより，個々のパターンがもたらす解決集合よりも大きな解決をもたらす [13]．また，特徴により整理/分類された相互に関係するパターンの集合をパターンカタログと呼ぶ．

項目 Related Patterns(関連パターン) における記述内容から，`Singleton` パターンが `Abstract Factory` パターン [3] を参照していることがわかる．

2.2 パターン検索システムの必要性と問題

パターンを利用することにより，ソフトウェア開発における問題を効率よく解決することができる．また，パターンはソフトウェア開発において頻出する問題と解法を提示するため，プログラミングやオブジェクト指向設計といったソフトウェア開発手法の習得に役立つ [12]．

パターンを効率よく利用および学習するためには，利用者から与えられる検索クエリから推測して，大量のパターン集合から利用者の目的に合致する，あるいは参考となるパターンを提示する検索システムが不可欠である．パターン検索システムの構築にあたって解決すべき問題として，検索精度の問題と検索効率の問題の 2 つが挙げられる．検索精度の問題の解決にはパターン文書のみを収集したリポジトリを作成することで解決できる．検索効率に関する問題の解決には，リポジトリ内のパターン集合に対する事前の順序付けが有効である．これまでにいくつかのパターン検索システムが提案されている [1] [8] [15] [17] が，既存の検索システムはリポジトリを用いて検索精度の向上を図るものの検索結果の順序付けを行わないため，利用者が与える検索クエリに合致あるいは関係するパターンの候補が大量に存在する場合検索効率が低下する．そのため，リポジトリ内のパターン集合を何らかの指標に基づいて自動的に順序付ける処理が必要である．

パターンを順序付ける際，重要なパターンが上位になるようにパターン集合を並び替えるのが望ましい．パターンの利用の観点から見た重要なパターンとは，ソフトウェア開発に利用しやすいパターンであり，

- ソフトウェア開発において頻繁に利用されるパターン
- Web 上で頻繁に検索されるパターン
- パターン文書内で頻繁に参照されるパターン

の 3 つのパターンが挙げられる．ソフトウェア開発において頻繁に利用されるパターンに基づいてパターン集合を順序付ける方法は，パターンの再利用の観点からは非常に有効であるが，実現が非常に困難である．次に，Web 上で頻繁に検索されるパターンに基づいてパターン集合を順序付ける方法は，Web 上で利用される回数から重要度を求めるため上位に利用されやすいパターンがくるが，パターン文書の内容に基づいて順序付けするものではないためソフトウェア開発において利用可能なパターンが上位になるとは限らない．また，パターン文書内で頻繁に参照されるパターンに基づいて，パターン文書を順序付ける方法は，他のパターンに頻繁に参照されるより基となる汎用的なパターンを上位とみなすため，多くの利用者に必要とされやすい．

また，パターンの学習の観点から見れば，パターンの学習効率が重要である．つまり，背景にある開発手法の習得を目的としてパターンを学習する際に，既に学習したパターンが他のパターンを関連パターンとして参照していれば，被参照パターンを学ぶ際に効率良く複数のパターンを学習することができる．つまり，初めに学んだパターンが，より多くのパターンに参照されていれば，後のパターン学習をより効率的に行う事が可能となる．

そこで，本稿のパターン検索システムを，検索結果において多くのパターンから参照されるパターンの

Pattern Name	`Singleton`
Problem	The `Singleton` design pattern controls the number of objects created for one class. ...
Solution	Create a class with a static method that returns a single instance ...
Related Patterns	The `Singleton` design pattern can be used with the `Abstract Factory` design pattern to guarantee at most one factory class is created. ...

図 2 `Singleton` **パターン (抜粋)**

順に並び替える機能を有するものとする．パターンの効率の良い学習の支援ツールとなる．また，パターン文書内で頻繁に参照されるパターンに基づいてパターン文書を順序付ける機能を持つ．同機能によりパターンを利用する際に必要なパターンを効率よく利用するための有効な支援ツールとなる．

3 パターンの重要度に基づく検索システム

本章では，収集したパターン文書から抽出したパターン間の関係情報 (文書中で ”Related Patterns” として指定される参照情報と，パターンカタログへの帰属情報）に基づいて，各パターンの重要度を自動的に算出する手法を提案する．本稿では，提案手法をパターンランク法と呼ぶ．さらに，パターンランク法の適用で得られる重要度に基づいて，利用者から与えられる検索クエリを含むパターン集合を順序付けて提示する検索システムを提案する．提案システムでは，利用者が膨大な検索結果を得る場合においても，パターンランク法の適用により重要性の高いパターンから効率よく利用することが可能である．

重要性の高いパターンとして，ソフトウェア開発において頻繁に利用されるパターンと，Web 上で頻繁に検索されるパターンと，パターン文書内で頻繁に参照されるパターンの 3 つのパターンが考えられる．パターンランク法では，パターン文書内で頻繁に参照されるパターンを重要性の高いパターンとした．パターンランク法によるパターン文書集合の並び替えにより，多くの利用者に必要とされやすいパターンから検討することが可能となるが，必ずしもソフトウェア開発で頻繁に利用されるパターンが上位になるとは限らない．

また，既に学習したパターンが他のパターンを関連パターンとして参照していると，被参照パターンを学ぶのが容易になる．そのため，パターンランク法における重要度の高いパターンを多数のパターンから参照されているパターンであるとする．学習目的でパターンを検索する利用者は，提案システムが提示する重要度の高いパターンから学習することによりパターンを効率的に学習する事が可能となる．

3.1 パターンランク法

パターンランク法は，要素間の関係から各要素の重要度を算出する仕組みについて，Web ページの重要度算出手法 PageRank [11] やソフトウェア部品の重要度算出手法 ComponentRank [7] を参考にしている．PageRank および ComponentRank との違いとして以下が挙げられる．

- 同一パターンカタログへの帰属
- 同一パターンの異なる文書表現

PageRank は，”重要な Web ページから参照される Web ページの重要度は高い” という規則を持ち，ページ間リンクを参照関係として用いる．パターンランク法では，PageRank をパターンに特化させ，PageRank の Web ページをパターン文書，ページ間の参照関係をパターン間の参照関係とした．

単純に PageRank を適用するだけでは，

- 同一パターンカタログ内に存在するパターン同士が頻繁に同時に利用されるにも関わらず明示的な参照関係がない場合にパターン同士に関係が存在しないとして扱われる
- 同一のパターン文書が異なる Web サイトで公開されている場合に異なるパターンとして扱われる

などの問題が生じる．

パターンランク法ではこれらの問題を解決するため，パターン間の関係を以下の 3 種に分類した．また，パターン間の参照関係を以下の関連パターンの項

における参照関係と定義した．

- **関連パターン:** パターンの作者によりパターン文書内に記述された，他のパターンへの参照を指す．あるパターンが他のパターンを関連パターンとして指定する場合，両パターンが解法として与えるソフトウェア構造が類似している，一方のパターンが他方のパターンの派生型である (より特化した問題を扱う)，一方のパターンの解法内で他方のパターンを使用する，といった状況が挙げられる．パターンランク法では，このようなパターン間の関係があるとき関連パターンとして扱う．関連パターンの存在は，パターン形式における関連パターン (”Related Patterns”) の項目記述内に限らず，パターン文書全体において他のパターン名が出現するか否かから判定する．本稿では，関連パターンの関係を参照あるいは参照関係と定義する．
- **同一パターンカタログへの帰属:** 属するパターンカタログが同一である関係を指す．パターンカタログとは，互いに関係する複数のパターンをまとめて閲覧できるようにしたものである．組み込みソフトやリアルタイムソフトなどの対象領域，分析や設計などのソフトウェア開発における対象プロセスなどパターンの作者の意図する分野ごとにまとめられ，パターン言語により記述，収集，整理されている．同一パターンカタログ内のパターンはパターン間に明示的な関係が存在しない場合にも，しばしば複数同時に利用される．例えば，GoF デザインパターンカタログ [2] [5] の `Composite` パターンと `Proxy` パターンは，参照関係は存在しないが，頻繁に同時に利用される．パターンランク法では，同一のパターンカタログに含まれることをパターン間の関係として扱う．本稿では，同一パターンカタログへの帰属の関係を同一パターンカタログ帰属関係と定義する．
- **同一ソフトウェアプロジェクトにおける適用:** ソフトウェア開発においてしばしば同時に使われることがあるパターン同士の関係を指す．関連パターンの関係より弱い重要度を持たせる予定であるが，本稿では扱っていない．

また，パターンランク法ではパターンの重要度の規則を以下のように定義した．

- **規則 1:** より多くのパターンから参照されているパターンほど重要度は高い
- **規則 2:** 重要度の高いパターンから参照されているパターンほど重要度は高い

規則 1 により，複数のパターン文書集合の適切な適用順序や学習順序を各パターンの重要度に反映させ，重要度の高いパターンほど他のパターンの基となる汎用的なパターンとなる可能性や，他のパターンと共に用いられる可能性を高めることが可能である．例えば，GoF デザインパターンカタログにおいて `Factory Method` パターンは，同パターンの解法の実現手段として，`Template Method` パターンを参照している．ここで，両パターンを適用目的や学習目的から閲覧する際に，`Template Method` パターンを先に学んでいれば，`Factory Method` パターンを学ぶ際により理解しやすい．従って，`Template Method` パターンがより重要と考えることができる．

規則 2 により，参照元のパターンの重要度の高さを参照先のパターンの重要度の高さに反映させることが可能である．例えば，規則 2 を用いず単純に被参照数 (対象パターンを関連パターンとして参照するパターンの数) の多さから重要度を算出する場合に，重要度の低いパターンから参照されるパターン p_1 と，同数の重要度の高いパターンから参照されるパターン p_2 が，それぞれ同じ重要度を持つこととなり，p_1, p_2 の重要度の違いを適切に表現できない．ここで，規則 2 を適用して，パターン p_1 よりパターン p_2 の重要度を高いとみなす．

パターンランク法を用いた重要度算出の例を図 3 に示す．図 3 では，パターン p_1 がパターン p_2 及びパターン p_4 を，パターン p_3 がパターン p_4 をそれぞれ関連パターンとして参照している．また，パターン p_1 の重要度は 0.5，パターン p_3 の重要度は 0.1 である．パターンランク法では，まず参照辺の数と参照元のパターンの重要度から参照辺の重みを求める．例えば，パターン p_1 を参照元とする参照辺 e_{12} および e_{14} の重みは，参照辺が 2 つ存在するため $0.5/2 = 0.25$ となり重さが 0.25 となる．同様に，パターン p_3 を参

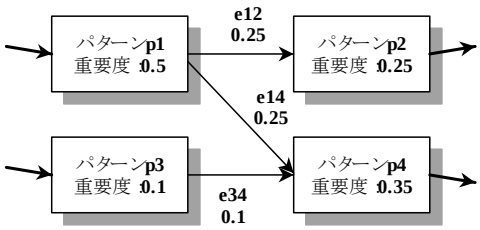

図 3　パターン間の参照に基づく重要度

照元とする参照辺 e_{34} は重さが 0.1 となる．このような，パターン重要度の参照辺の重みに対する分配の割合を配分率と呼ぶ．また，各パターンの重要度は自らを参照先とする参照辺の重みを合計して求める．例えば，パターン p_4 の重要度は $0.25 + 0.1 = 0.35$，パターン p_2 の重要度は 0.25 となる．

重要度の伝播の様子を図 4 に示す．パターンランク法では，まずパターン全体の重要度を 1 とし，各パターンに重要度を割り振る．図 4 では 1 をパターン総数で割った値 0.25 を初期値として与えている．次に，参照辺の重みを求める．参照辺の重みは，参照元のパターンの重要度を同じパターンを参照元とする参照辺の総数で割ることにより得る．図 4 では，`State` パターンは参照先を 2 つ持つため，参照辺の重さは $0.25/2 = 0.125$ となる．次に，参照辺からパターン重要度を求める．パターンの重要度は，パターンを参照している参照辺の重さを合計することにより得る．図 4 では，`FixedSizedBuffer` パターンを参照している辺が 2 つ存在する．そのため `FixedSizedBuffer` パターンの重要度は $0.25 + 0.125 = 0.375$ となる． 計算の過程において各パターンの重要度は一定の値に収束する．図 4 では，State パターンの重要度は 0.2814 に，`FixedSizedBuffer` パターンの重要度は 0.2832 に収束している．

しかしながら，参照を持たないパターンが存在した場合，参照を持たないパターンからは他のパターンへ重要度は伝播しないが参照を持たないパターンへは重要度が伝播するため，計算のたびにパターン全体の重みから参照を持たないパターンの重要度の分の重みが減衰する．そのため，参照を持たないパターンが存在した場合，計算を繰り返しても重要度は収束しない．

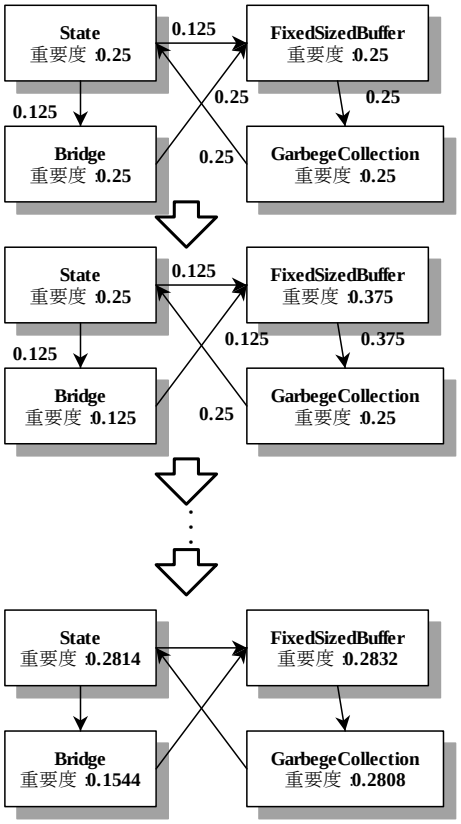

図 4　パターン重要度の変遷

この問題については，3. 1. 5 節において詳しく扱う．

3. 1. 1　パターンランク法の計算モデル

上述したパターンの重要度算出の方法を以下において数学的に定義する．

パターンの集合を，各パターンを表す頂点 p の集合 P および関連パターンへの参照を表す有向辺 e の集合 E から構成される有向グラフ $G = (P, E)$ として表す．また，各パターンの重要度を，対応する頂点 p の重み $w(p)$ として表す．ここで，グラフ上の全頂点の重みの合計を 1 とし ($\sum_{p \in G} w(p) = 1$)，さらに，常に $0 < w(p) \leq 1$ が成り立つと定義する．

頂点 p_i から頂点 p_j への辺 e_{ij} の重みを

$$w'(e_{ij}) = d_{ij} \cdot w(p_i)$$

と表す．d_{ij} を配分率と呼び，

$$\sum_i d_{ij} = 1$$

かつ

$$0 \leq d_{ij} \leq 1$$

を満たす．各頂点 p_i の重みは，p_i を終点とする辺 e_{*i} の集合 $IN(p_i)$ 内の各辺の重みの合計となる．具体的には，

$$w(p_i) = \sum_{e_{ki} \in IN(p_i)} w'(e_{ki})$$

と表すことができる．これに配分率の定義を合わせると，各頂点の重みを

$$w(p_i) = \sum_{e_{ki} \in IN(p_i)} d_{ki} \cdot w(p_k)$$

と表すことができる．

次に，頂点全体の重みを行列 W と表す．

$$W = \begin{pmatrix} w(p_1) \\ w(p_2) \\ \vdots \\ w(p_n) \end{pmatrix}$$

また，配分率全体を配分率行列 D とし，

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix}$$

と表す．W および D により，グラフ全体における重みの計算式は $W_{t+1} = D^t \cdot W_t$ と表される．D^t は配分率行列 D の転置行列，下付きの t は計算回数である．

上述の計算は線形代数において連立方程式の固有ベクトルを求める問題であり，反復法を用いて計算する．反復法では，初期ベクトル x_0 を適当にとり，$x_{(t+1)} = Ay_{(t)}$(ただし，$y_{(t)} = x_{(t)}/c_{(t)}$) として $t \to \infty$ としたとき，x は最大固有値を持つ固有ベクトルに収束し同時に c はその最大固有値に収束するを利用した計算方法である [21]．ここで，A は N 次正方行列である．ペロン・フロベニウスの定理から，推移確率行列の絶対値最大の固有値は 1 なので，$y_{(t)} = x_{(t)}$ となり $x_{(t+1)} = Ax_{(t)}$ となる．

パターンランク法では，反復法により $t \to \infty$ のとき一定の値に収束した行列 W の各頂点の重みをパターンの重要度として用いる．反復法の計算回数を無限大とするのは不可能であるため，閾値 s を設け $|\frac{w_t - w_{t-1}}{w_{t-1}}| \leq s$ を計算の終了条件とした．w_t は t 回目の計算における各パターン頂点の重さである．

配分率 d_{ij} は，同一パターンカタログ帰属関係および，参照関係を持たない独立のパターンを考慮して計算される．参照関係のみに基づいた配分率を d_{ij}^S とし，

$$d_{ij}^S = 1/d\_sum$$

とする．ここで d_sum は，パターン頂点 p_i を参照元とした参照辺の数である．また，パターンカタログ帰属関係を考慮した配分率全体の式ををを d_{ij}^C とし，同一パターンカタログ帰属関係にあるパターンへの参照辺の配分率を d_{ij}^c，そうでないパターンへの参照辺の配分率を $d_{ij}^{\neg c}$ とし，

$$d_{ij}^C = \begin{cases} d_{ij}^c(\text{同一パターンカタログに帰属}) \\ d_{ij}^{\neg c}(\text{パターンカタログが異なる}) \end{cases}$$

$$d_{ij}^c = d_{ij}^S + \grave{d}_{ij}^c$$

$$d_{ij}^{\neg c} = (1-c) \cdot d_{ij}^S$$

$$\grave{d}_{ij}^c = c \cdot \textstyle\sum_{k \in \{\neg c\}} d_{ik} / (c\_sum - 1)$$

とする．ここで，$\grave{d}_{ij}^c$ は擬似辺の配分率，c はパターンカタログ配分率，$\sum_{k \in \{\neg c\}} d_{ik}$ は同一パターンカタログ内に存在しないパターンへの配分率の合計，c_sum は同一パターンカタログ内のパターン数である．また，同一パターンについて考慮した配分率全体の式を d_{ij}^M とし，同一パターンを参照先にした参照辺の配分率を d_{ij}^{mi} とし，同一パターンを参照元にした参照辺の配分率を d_{ij}^{mo} とし，参照元も参照先も同一パターンを持たないパターンである参照辺の配分率を $d_{ij}^{\neg m}$ とし，

$$d_{ij}^M = \begin{cases} d_{ij}^{mo}(\text{参照元が同一パターン}) \\ d_{ij}^{mi}(\text{参照先が同一パターン}) \\ d_{ij}^{\neg m}(\text{参照先・元が非同一パターン}) \end{cases}$$

$$d_{ij}^{mo} = \textstyle\sum_{k \in \{m\}} d_{kj}^C / m\_sum$$

$$d_{ij}^{mi} = \textstyle\sum_{k \in \{m\}} d_{ik}^C$$

$$d_{ij}^{\neg m} = d_{ij}^C$$

とする．ここで，$\sum_{k \in \{m\}} d_{kj}^C$ は複数の同一パターン p_i を参照元とする参照辺の配分率の合計，$\sum_{k \in \{m\}} d_{ik}^C$ は複数の同一パターン p_j を参照先とする参照辺の配分率の合計，m_sum は同一パターン数である．また，最終的に配分率 d_{ij} は，

$$d_{ij} = \begin{cases} d_{ij}^{r}(p_i \text{ が参照を持つ場合}) \\ d_{ij}^{\neg r}(p_i \text{ が参照を持たない場合}) \end{cases}$$
$$d_{ij}^{r} = (1-r)\cdot d_{ij}^{M} + \grave{d}'_{ij}$$
$$d_{ij}^{\neg r} = (1-r)\cdot \grave{d}''_{ij} + \grave{d}'_{ij}$$
$$\grave{d}''_{ij} = 1/n$$
$$\grave{d}'_{ij} = r/n$$

となる．ここで，$\grave{d}'_{ij}$ はパターン全体への擬似的な参照辺の配分率，$\grave{d}''_{ij}$ は参照を持たないパターンからの擬似的な参照辺の配分率，r は修正配分率，n はパターンランク法の適応対象であるパターン集合内のパターン数である．パターンカタログ配分率 c および修正配分率 r の値は，実験により求める．式の各項について以下の節ごとに詳しい解説を行う．

3.1.2 パターンカタログ配分率

同一パターンカタログ内のパターン文書は，パターン間に明示的な関係が存在しない場合にも，しばしば複数同時に利用される．同一パターンカタログ内のパターン間には，明示的に示されていない同一パターンカタログ帰属関係が存在すると考えられるが，3.1.1 節で述べた参照に基づく計算方法では同一パターン帰属関係をパターン重要度へ影響させることはできない．

パターンランク法ではこの問題に対するアプローチとして，同一パターンカタログ内のパターン同士に擬似辺 $\grave{e}_{ij}^{c}$ を与え，擬似辺には通常の参照辺よりも弱い重みを与えている．擬似辺に与える重みは，パターンカタログ内のパターンからパターンカタログ外のパターンへの参照辺の重みから一定の割合を減ずることにより得ている (図 5)． 重みを減ずる割合をパターンカタログ配分率 c と定義し，同一パターンカタログ帰属関係を考慮した配分率全体を d_{ij}^{C}，同一パターンカタログ帰属関係にあるパターンへの参照辺の配分率を d_{ij}^{c}，パターンカタログ外のパターンへの参照辺の配分率を $d_{ij}^{\neg c}$ と定義する．同一パターンカタログ帰属関係を考慮した配分率の式を以下に示す．

p_i が p_j と同一パターン帰属関係にある場合の配分率

$$d_{ij}^{C} = d_{ij}^{c} = d_{ij}^{S} + \grave{d}_{ij}^{c}$$

擬似辺 $\grave{e}_{ij}^{c}$ の配分率

$$\grave{d}_{ij}^{c} = c\cdot \sum_{k\in\{\neg c\}} d_{ik}/(c\_sum - 1)$$

p_i が p_j と同一パターン帰属関係にない場合の配分率

$$d_{ij}^{C} = d_{ij}^{\neg c} = (1-c)\cdot d_{ij}^{S}$$

d_{ij}^{S} は参照関係のみに基づく計算式における配分率，$\grave{d}_{ij}^{c}$ は擬似辺 $\grave{e}_{ij}^{c}$ の配分率，c はパターンカタログ配分率，c_sum は同一パターンカタログ内のパターンの総数，$\sum_{k\in\{\neg c\}} d_{ik}$ は同一パターンカタログ内に存在しないパターンへの配分率の合計である．

p_i が p_j と同一パターン帰属関係にない場合の配分率 $d_{ij}^{\neg c}$ は，参照関係のみに基づく計算式における配分率 $d_{ij}^{S} = c\cdot d_{ij}^{S} + (1-c)\cdot d_{ij}^{S}$ から $c\cdot d_{ij}^{S}$ の分だけ減ずるため，$d_{ij}^{\neg c} = (1-c)\cdot d_{ij}^{S}$ となる．また，擬似辺 $\grave{e}_{ij}^{c}$ の配分率 $\grave{d}_{ij}^{c}$ は，p_i が p_j と同一パターン帰属関係にない場合の配分率から減じた配分率の合計 $c\cdot\sum_{k\in\{\neg c\}} d_{ik}$ を擬似辺の数で割ったものとなる．擬似辺は，同一カタログ内にある自分以外のパターンへ与えられるものなので，(c_sum-1) 個存在する．従って，擬似辺の配分率は $c\cdot\sum_{k\in\{\neg c\}} d_{ik}/(c\_sum-1)$ となる．p_i が p_j と同じパターンカタログに属する場合の配分率 d_{ij}^{c} は，参照関係のみに基づく計算式における配分率 d_{ij}^{S} に擬似辺 $\grave{e}_{ij}^{c}$ の配分率 $\grave{d}_{ij}^{c}$ を足したものなので，$d_{ij}^{c} = d_{ij}^{S} + \grave{d}_{ij}^{c} = d_{ij}^{S} + c\cdot\sum_{k\in\{\neg c\}} d_{ik}/(c\_sum-1)$ となる．

p_i が p_j と同じパターンカタログに属する場合の配分率 d_{ij}^{c} を，p_i が p_j と同一パターン帰属関係にない場合の配分率 $d_{ij}^{\neg c}$ から減じた重み $c\cdot\sum_{k\in\{\neg c\}} d_{ik}$ の分だけ増加するため全体の重みの合計は変化せず，条件 $\sum_i d_{ij} = 1$ は保証される．

同一パターンカタログ帰属関係を考慮した配分率の式の計算例として，図 5 のパターン p_1 からパターン p_4 への参照辺の重み $w(e_{14}^{c})$ について考える．パターンカタログ配分率を 0.25 とすると，参照辺の重みは式 $w(e_{ij}^{c}) = w(p_i)\cdot d_{ij}^{S} + w(\grave{e}_{ij}^{c})$ より，擬似辺の重み $w(\grave{e}_{14}^{c})$ となる．式 $\grave{e}_{14}^{c} = w(p_1)\cdot\grave{d}_{14}^{c}$ および $\grave{d}_{14}^{c} = 0.25\cdot 1/(3-1) = 0.125$ より擬似辺 $\grave{e}_{14}^{c}$ の重みは，$\grave{e}_{14}^{c} = 0.5\cdot 0.125 = 0.0625$ となる．また，パターン p_4 からパターン p_3 への参照辺の重み $w(e_{43}^{c})$ についても同様に考えると，式 $w(e_{43}^{c}) = w(p_4)\cdot d_{43}^{S} + w(\grave{e}_{43}^{c})$ となる．$w(p_4)\cdot d_{43}^{S} = 0.1\cdot 0.5 = 0.05$，また $\grave{d}_{43}^{c} = 0.25\cdot 0.5/(3-1) = 0.0625$ より式 $\grave{e}_{43}^{c} = w(p_4)\cdot\grave{d}_{43}^{c} = 0.1\cdot 0.0625 = 0.00625$，よって参照辺の

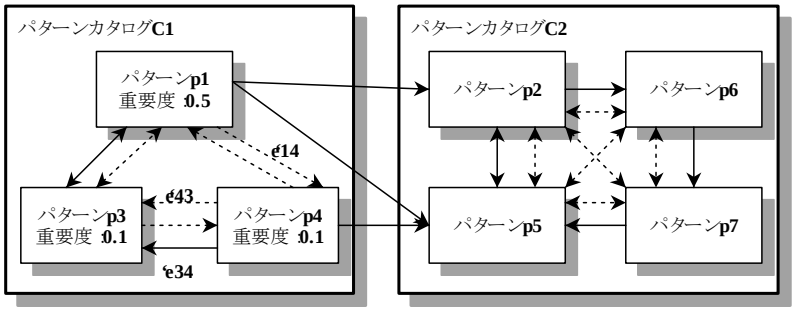

図 5 パターンカタログ配分率の扱い

表 1 パターンカタログ配分率を用いない重要度計算による結果

順位	パターン名	パターンカタログ名	重要度
1	パターン p_2	パターンカタログ C_2	0.3209
2	パターン p_5	パターンカタログ C_2	0.2703
3	パターン p_1	パターンカタログ C_1	0.1256
4	パターン p_6	パターンカタログ C_2	0.0993
5	パターン p_7	パターンカタログ C_2	0.0912
6	パターン p_3	パターンカタログ C_1	0.0710
7	パターン p_4	パターンカタログ C_1	0.0214

表 2 パターンカタログ配分率を用いた重要度計算による結果

順位	パターン名	パターンカタログ名	重要度
1	パターン p_5	パターンカタログ C_2	0.2867
2	パターン p_3	パターンカタログ C_2	0.2820
3	パターン p_7	パターンカタログ C_2	0.1415
4	パターン p_6	パターンカタログ C_2	0.1411
5	パターン p_1	パターンカタログ C_1	0.0689
6	パターン p_3	パターンカタログ C_1	0.0550
7	パターン p_4	パターンカタログ C_1	0.0243

重み $w(e^c_{43})$ は，$w(e^c_{43}) = 0.05 + 0.00625 = 0.05625$ となる．

パターン重要度の計算に同一パターンカタログ帰属関係を考慮した配分率の式を反映することにより，重要なパターンが多数存在するパターンカタログ内のパターンの重要度を全体的に底上げする効果がある．

例えば，図 5 においてパターンカタログ配分率を用いずに収束するまで重要度計算を行うと，各パターンの重要度は表 1 のようになる． また，図 5 においてパターンカタログ配分率を用いて (パターンカタログ配分率 c を 0.15，修正配分率 r を 0.15，閾値 s を 10^{-8} とし) 収束するまで重要度計算を行うと，各パターンの重要度は表 2 のようになる．

表 1 から，パターンカタログ C_2 の方がパターンカタログ C_1 より重要度が高いパターンが多く存在することが分かる．また，表 1，2 から，パターンカタログ配分率を用いることによりパターンカタログ C_2 に属すパターン p_6，パターン p_7 の順位が上がる．

参照関係のみに基づく計算式 d^S_{ij} における配分率行列内のそれぞれの配分率に対し，パターンカタログ配分率の計算を行ったものをパターンカタログ配分率行列と定義する．

3.1.3 文書表現が異なる同一パターンの考慮

著名なパターンは，同一パターンであるにも関わらず複数の異なる記述者により異なるパターン文書として記述および公開されていることがある．

パターンランク法では，異なる記述者により書かれた同一パターンのパターン頂点を併合した後に重要度を計算する (図 6)．同一パターンについて考慮した配分率全体を d^M_{ij}，同一パターンを参照先にした参照辺の配分率を d^{mi}_{ij}，同一パターンを参照元にした参照辺の配分率を d^{mo}_{ij}，参照元も参照先も同一パターンを持たないパターンである参照辺の配分率を $d^{\neg m}_{ij}$ と定義し，式を以下に示す．

複数の同一パターン p_i を参照元とする参照辺の配分率

$$d^M_{ij} = d^{mo}_{ij} = \sum_{k \in \{m\}} d^C_{kj} / m\_sum$$

複数の同一パターン p_j を参照する参照辺の配分率

$$d_{ij}^{M} = d_{ij}^{mi} = \sum_{k\in\{m\}} d_{ik}^{C}$$

参照元も参照先も同一パターンを持たないパターンである参照辺の配分率

$$d_{ij}^{M} = d_{ij}^{\neg m} = d_{ij}^{C}$$

$\sum_{k\in\{m\}} d_{kj}^{C}$ は複数の同一パターン p_i を参照元とする参照辺の配分率の合計，$\sum_{k\in\{m\}} d_{ik}^{C}$ は複数の同一パターン p_j を参照先とする参照辺の配分率の合計，m_sum は同一パターン数である．

同一パターンについて考慮した配分率の式の計算例として，図 6 のパターン p_3 の重要度について考える．図 6 では， パターン p_1 が，パターン p_3 を記述した文書を参照し，パターン p_2 が，パターン p_3 を記述した他の文書を参照している．パターンランク法では，まず同一のパターン p_3 および参照辺を併合する．パターン p_3 を参照する参照辺は 2 本になり重みがそれぞれ 0.5 と 0.1 となる．パターン p_3 の重要度はこれらの参照辺の重みの合計なので，$0.5 + 0.1 = 0.6$ となり重要度は 0.6 となる．

本稿における同一パターンはパターン文書の内容が同じであることを指すが，パターンランク法では同一パターンをパターン名が同じであるものとして扱っている．これは，パターン文書の内容が同一であることを自動的に判断するのが困難であるためである．

同一パターンの判断基準としてパターン名を使う利点に，パターン名がパターンを表現する一意的な名前を扱うもので他の既知な同一パターン名は別名として扱われるという点が挙げられる．また，同一パターンの判断基準としてパターン名を使う欠点に，同一パターンであるにも関わらずパターン名が異なるパターンが存在した場合に，異なるパターンとして扱ってしまう点と，異なるパターンであるにも関わらず同じパターン名を使っている場合に同一パターンとして扱ってしまう点が挙げられる．

同一パターンの判断基準におけるパターン名の利用は，実装が容易かつ現実的であり，パターン名の特性によるある程度の正確性を望むことが可能である．なお，本稿の実験において利用した 131 個のパターン文書のうち同一パターンは 23 組，パターン文書にして

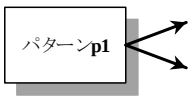

図 7　他のパターンから参照されないパターン

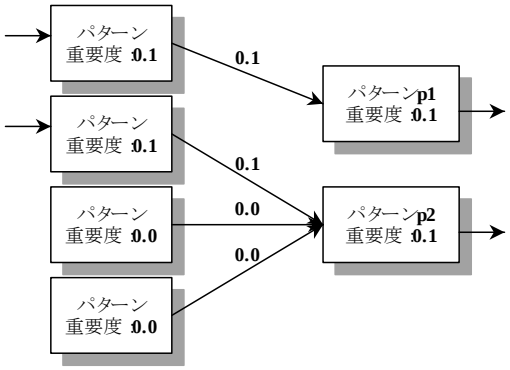

図 8　重要度が 0 なパターンの問題

46 個存在したが，全ての同一パターンのパターン名は同じであった．従って，パターン名を同一パターンの判断基準として用いる上での問題は生じなかった．

3.1.4　参照されないパターンの考慮

他のパターンから参照されないパターンは，そのパターンへの参照辺の重みが 0 になるため重要度が 0 となる．例えば，図 7 のようにパターン p_1 が他のパターンから参照されない場合，パターン p_1 の重要度はパターン p_1 を参照先とした参照辺の重みを合計して求めるため，重要度が 0 となる．

重要度が 0 のパターンが存在すると，同パターンを参照元とした参照辺の重みは 0 となる．例えば図 8 のようなパターン間関係が存在した場合，参照辺の重みの合計からパターン p_1 とパターン p_2 の重要度は 0.1 となる．しかしながら，パターン p_2 は重要度が 0 の 2 つのパターンから参照されているため，パターンランク法規則 1 の”より多くのパターンから参照されているパターンほど重要度は高い”に反する．

パターンランク法では同問題に対して，全てのパターン間に擬似辺 $\dot{e}'_{ij}$ を追加し，擬似辺に実際の参照辺よりも弱い重みを与える．擬似辺の重みは，パターン文書集合全体の重みから一定の割合を減ずることにより得る (図 9)． 擬似辺を追加することで，他の

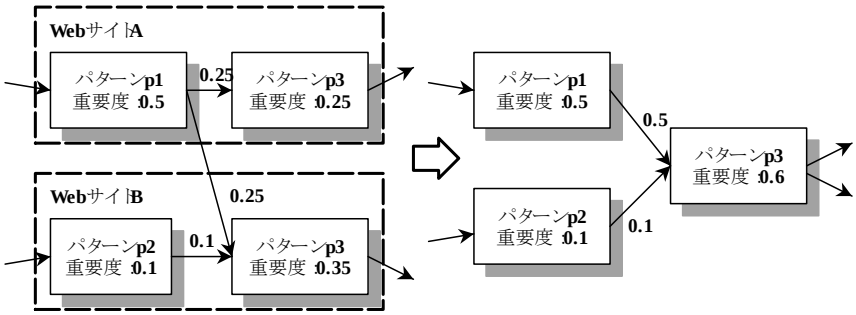

図 6 同一パターンの異なる文書表現の考慮

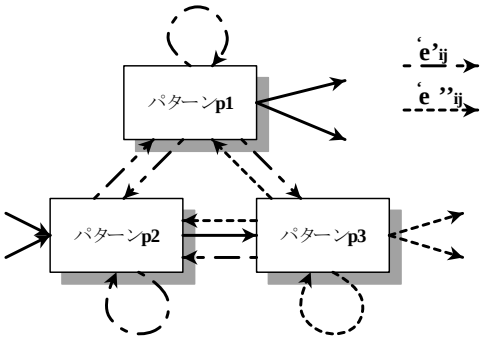

図 9 修正配分率の扱い

パターンに参照されないパターンは存在しなくなり，重要度が 0 のパターンは存在しなくなる．重みを減ずる割合を修正配分率 r と定義する．参照されないパターンおよび参照を持たないパターンを考慮した最終的な配分率全体を d_{ij} と定義する．参照されないパターンを考慮した配分率の式を以下に示す．

p_i が p_j を参照している場合の配分率

$$d_{ij} = (1-r)\cdot d_{ij}^{M} + \grave{d}'_{ij}$$

擬似辺 $\grave{e}'_{ij}$ の配分率

$$\grave{d}'_{ij} = r/n$$

n は計算対象のパターン文書集合内のパターンの総数，$\grave{d}'_{ij}$ は擬似辺 $\grave{e}'_{ij}$ の配分率，r は修正配分比率であり $0 \le r \le 1$ 満たす．

本来重要度が 0 となるべきパターンが重要度を持つことになるが，修正配分率 r の値を十分に小さくすれば重要度計算の繰り返しによりパターンの順位に影響しないほど小さな値に収束する．

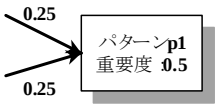

図 10 他のパターンを参照しないパターン

3.1.5 参照を持たないパターンの考慮

他のパターンへの参照を持たないパターンが存在する場合，同パターンを参照元とした参照辺の配分率は全て 0 になり配分率に関する制約 $\sum_i d_{ij} = 1$ を満たさなくなるため，3.1.1 節で述べたパターン全体の重要度計算は不可能になる．

例えば図 10 のようにパターン p_1 が他のパターンを参照しない場合，パターン p_1 は同パターンを参照先とした参照辺の重みを合計した重みを持つが，パターン p_1 を参照元とした参照辺の配分率は全て 0 であるため，重要度の計算ごとにパターン p_1 を参照先とした参照辺の重みを合計した分 (図 10 の例では 0.5) だけ全体の重みが減じる．

パターンランク法ではこの問題に対するアプローチとして，参照を持たないパターンから全てのパターンに対し擬似辺 $\grave{e}''_{ij}$ を追加している (図 9)．擬似辺の追加により，他のパターンを参照しないパターンは存在しなくなり，全体の重みが減じる問題を回避することが可能となる．他のパターンを参照しているパターンの参照辺の配分率を d_{ij}^{r}，他のパターンを参照していないパターンの参照辺の配分率を $d_{ij}^{\neg r}$ と定義し，参照を持たないパターンを考慮した配分率の式を以下に示す．

p_i が他のパターンを参照している場合の配分率

$$d^{r}_{ij} = (1-r)\cdot d^{M}_{ij} + \grave{d}'_{ij}$$

p_i が他のパターンを参照していない場合の配分率

$$d^{\neg r}_{ij} = (1-r)\cdot \grave{d}''_{ij} + \grave{d}'_{ij}$$

$\grave{d}''_{ij}$ は $\grave{e}''_{ij}$ の配分率である．

同式により本来関係のないパターン間に対し比較的強い関係を与えることになるが，全ての擬似辺に対して等しい重みを持たせることにより特定のパターンへの影響を回避し，他のパターンへ全く参照を持っていない状態を表現する．

参照されないパターンおよび参照を持たないパターンを考慮した計算は，3.1 節で述べた配分率行列に対して行う．ここで，行列内のそれぞれの配分率に対し修正配分率の計算を行ったものを修正配分率行列とする．

配分率行列内の配分率 d_{ij} は，参照関係から求めた配分率に対して，パターンカタログ配分率の処理，同一パターンの処理，修正配分率の処理を行って得る．よって配分率は，

$$d^{S}_{ij} = 1/d\_sum$$

$$\Downarrow$$

$$d^{C}_{ij} = \begin{cases} d^{c}_{ij}(\text{同一パターンカタログに帰属}) \\ d^{\neg c}_{ij}(\text{パターンカタログが異なる}) \end{cases}$$

$$d^{c}_{ij} = d^{S}_{ij} + \grave{d}^{c}_{ij}$$

$$d^{\neg c}_{ij} = (1-c)\cdot d^{S}_{ij}$$

$$\grave{d}^{c}_{ij} = c\cdot \textstyle\sum_{k\in\{\neg c\}} d_{ik}/(c\_sum - 1)$$

$$\Downarrow$$

$$d^{M}_{ij} = \begin{cases} d^{mo}_{ij}(\text{参照元が同一パターン}) \\ d^{mi}_{ij}(\text{参照先が同一パターン}) \\ d^{\neg m}_{ij}(\text{参照先・元が非同一パターン}) \end{cases}$$

$$d^{mo}_{ij} = \textstyle\sum_{k\in\{m\}} d^{C}_{kj}/m\_sum$$

$$d^{mi}_{ij} = \textstyle\sum_{k\in\{m\}} d^{C}_{ik}$$

$$d^{\neg m}_{ij} = d^{C}_{ij}$$

$$\Downarrow$$

$$d_{ij} = \begin{cases} d^{r}_{ij}(p_i \text{ が参照を持つ場合}) \\ d^{\neg r}_{ij}(p_i \text{ が参照を持たない場合}) \end{cases}$$

$$d^{r}_{ij} = (1-r)\cdot d^{M}_{ij} + \grave{d}'_{ij}$$

$$d^{\neg r}_{ij} = (1-r)\cdot \grave{d}''_{ij} + \grave{d}'_{ij}$$

$$\grave{d}''_{ij} = 1/n$$

$$\grave{d}'_{ij} = r/n$$

のように変遷する．

3.2 パターンランク法を用いた検索システム

パターンランク法を適用した提案システムの利用の流れを以下に示す (図 11)．流れの中の 5 および 7 では，「重要度算出後にクエリマッチング」するか，「クエリマッチング後に重要度算出」するかによって処理が分岐する．

1. 利用者は，検索対象として複数のパターンカタログを選択するか，あるいはリポジトリ内の全てのパターン文書を検索対象とするかを選択する．
2. 利用者は，選択したパターン文書集合に対し「重要度算出後にクエリマッチング」するか，「クエリマッチング後に重要度算出」するかを選択する．
3. 利用者は，必要とするパターンに関係のあると考えられる任意の文字列を検索クエリとしてシステムに入力する．
4. システムは，リポジトリ (収集済みのパターン文書集合) 内の全ての HTML 形式のパターン文書を解析し，パターン名，帰属するパターンカタログ名，関連パターン名などのパターン間の関係を抽出する．システムは，パターン文書内に同パターンとは異なるパターン p_x のパターン名文字列が出現する場合に，そのパターン p_x を参照パターンとして識別する．
5. システムは，利用者が「クエリマッチング後に重要度算出」する処理を選択していた場合，利用者から受け取った任意のキーワード列 (複数のキーワードが与えられた場合は全キーワード) を含むパターン文書を，リポジトリ内のパターン文書集合から検索/取得する．
6. システムは，与えられたパターン文書集合に対し，抽出したパターン間関係から配分率行列，パターンカタログ配分率行列，修正配分率行列を求め，重要度計算を行う．結果，各パターンの重要度を得る．
7. システムは，利用者がパターン文書集合に対し「重要度算出後にクエリマッチング」処理を選択していた場合，利用者から受け取った任意のキーワード列 (複数のキーワードが与えられた場合は全キーワード) を文書内に含むパターンを，重要

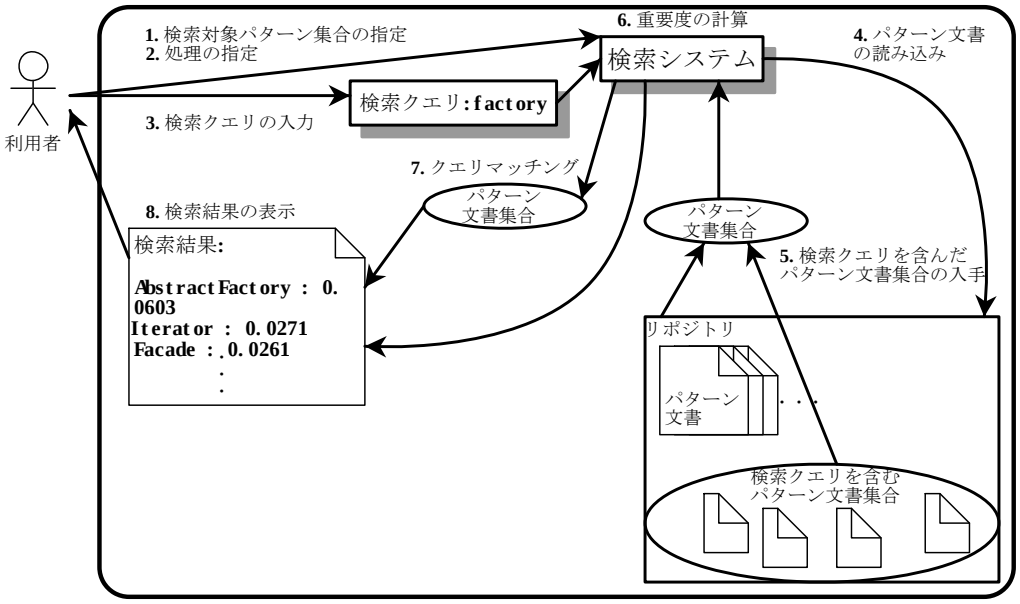

図 11 検索システムの全体図

度計算を行ったパターン文書集合から検索/取得する.

8. システムは，得られたパターン集合を重要度によって並び替え検索結果として利用者に提示する.

「重要度算出後にクエリマッチング」する場合，重要度の算出はリポジトリ内のパターン集合全体に対して行われる．対して「クエリマッチング後に重要度算出」する場合，検索クエリを含んだパターン集合を取得した後に重要度の算出を行うため，重要度の算出は検索クエリを含んだパターン集合に対してのみ行われる.

また,「重要度算出後にクエリマッチング」する場合，検索対象のパターン集合内の全てのパターンの参照辺が重要度に反映される (図 12). 「クエリマッチング後に重要度算出」を行う場合，検索クエリを含むパターン文書集合内での参照辺を用いた順位を得ることが可能であるが，検索対象のパターン集合内のパターンの参照辺のいくつかは重要度に反映されない．そのため，全体を対象とした重要度算出で重要度が高いパターンでも，低い重要度を示す場合がある (図 13).

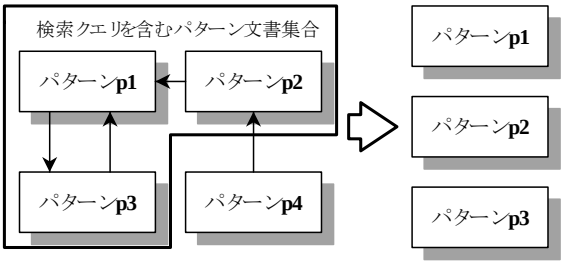

図 12 「重要度算出後にクエリマッチング」

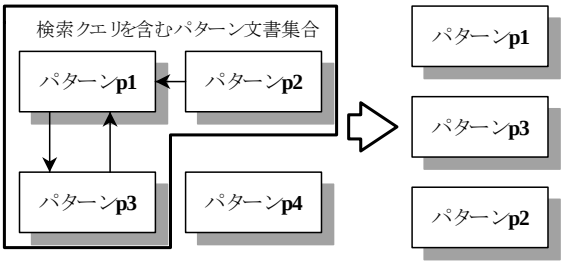

図 13 「クエリマッチング後に重要度算出」

提案システムを用いることで，利用者は検索クエリを文書内に含み，かつ，重要度によって順序付けられたパターン集合を検索結果として得る．そのため，利用者は検索意図に合致する，あるいは参考となるパ

ターンを効率よく発見できる．例えば図 11 で，利用者がオブジェクトの生成に関する問題を解決する必要があり，同問題の解決に参考となるパターンを検索するため検索クエリ”factory”を入力したとする．入力により，提案システムは検索結果として同クエリを文書内に含む Abstract Factory パターンなどのいくつかのパターンを，重要度によって順序付けて提示する．

提案システムに検索クエリ”factory”を入力した結果を図 14 に示す．図 14 から，検索クエリにマッチしたパターンが 29 個存在する事が分かる．また，第一位として得られた Abstract Factory パターンは，第二位でも得られているため，このパターンが異なる記述者によって異なる文書として公開されていることが分かる．パターンランク法では，これら異なる文書が同一のパターンを表していることをパターン名を用いて特定し，同一のパターン重要度 (0.06529) を提示した．画面には，パターン名にそのパターンが紹介されている Web ページへのリンクが，パターンカタログ名にそのパターンを紹介している Web サイトへのリンクが張られている．

また提案システムは，検索クエリの入力なしに，リポジトリ内の全パターンを重要度によって順序付けて提示することもできる．これにより，利用者はリポジトリ内のパターンを効率よく閲覧し学習することができる．

4 実験

提案システムを実装し，パラメータの検証実験およびシステムの有用性の評価を行った．

4.1 パラメータ検証実験

パラメータ検証実験は，パターンランク法の重要度計算に用いる各種パラメータの妥当な値を求めるための予備実験である．実験 1~3 の実施にあたり，パターン数 4 のテストセット 1 とパターン数 12 のテストセット 2 を用意した．

実験 1: 重要度計算の収束条件 $|\frac{w_t - w_{t-1}}{w_{t-1}}| \leq s$ の s 妥当な値を求める．

実験は，テストセット 1，2 およびリポジトリ内に存在する 4 つの Web サイト [2] [5] [6] [18] から収集したパターンカタログ内の 131 個のパターン文書集合のそれぞれについて s を 0.1 から順に 0.01，0.001$\cdots$ と変化させ各パターンの重要度を求めることにより行った．重要度の計算は，修正配分率 r を 0.15，パターンカタログ配分率 c を 0.15 とし，検索クエリは入力せず計算した．

実験の結果，テストセット 1 は s の値が 10^{-5} で，テストセット 2 と 131 個のパターン文書集合は s の値が 10^{-7} でパターンの重要度に変化が見られなくなった．

実験から，s の値は少なくとも 10^{-8} 以下に設定すれば重要度計算が収束することが分かった．

実験 2: 修正配分率 r の妥当な値をを求める．

実験は，テストセット 1，2 について，r の値を 0.0 から 1.0 まで変化させ各パターンの重要度を求めることにより行った．重要度の計算は，計算の終了条件 s を 10^{-8}，パターンカタログ配分率 c を 0.15 とし，検索クエリは入力せず計算した．

実験の結果，r の値が 0.2 より大きくなるにつれて各パターンの重要度に大きな変化が見られた．r の値は可能な限り重要度へ影響がでない値が望ましいため，重要度の計算において r の値は $r > 0.2$ を取るべきではないと言える．

実験から，r の値は $0.0 < r \leq 0.2$ の範囲の値を用いるべきであることが分かった．

実験 3: パターンカタログ配分率 c の適切な値を求める．

実験は，テストセット 1，2 について，パターンカタログ配分率 c の値を 0.0 から 1.0 まで変化させ各パターンの重要度を求めることにより行った．重要度の計算は，計算の終了条件 s を 10^{-8}，修正配分率 r を 0.15 とし，検索クエリは入力せず計算した．

実験の結果，c の値に伴い帰属するパターンカタログ単位でパターンの重要度が変化することが分かった (図 15)．

実験から，パターンカタログ配分率の値を変えることによりパターンカタログ単位で重要度を変化させることが可能であることを確認した．しかしながら，c の適切な値を得るには至らなかった．c の値は，パ

図 14　検索結果表示画面

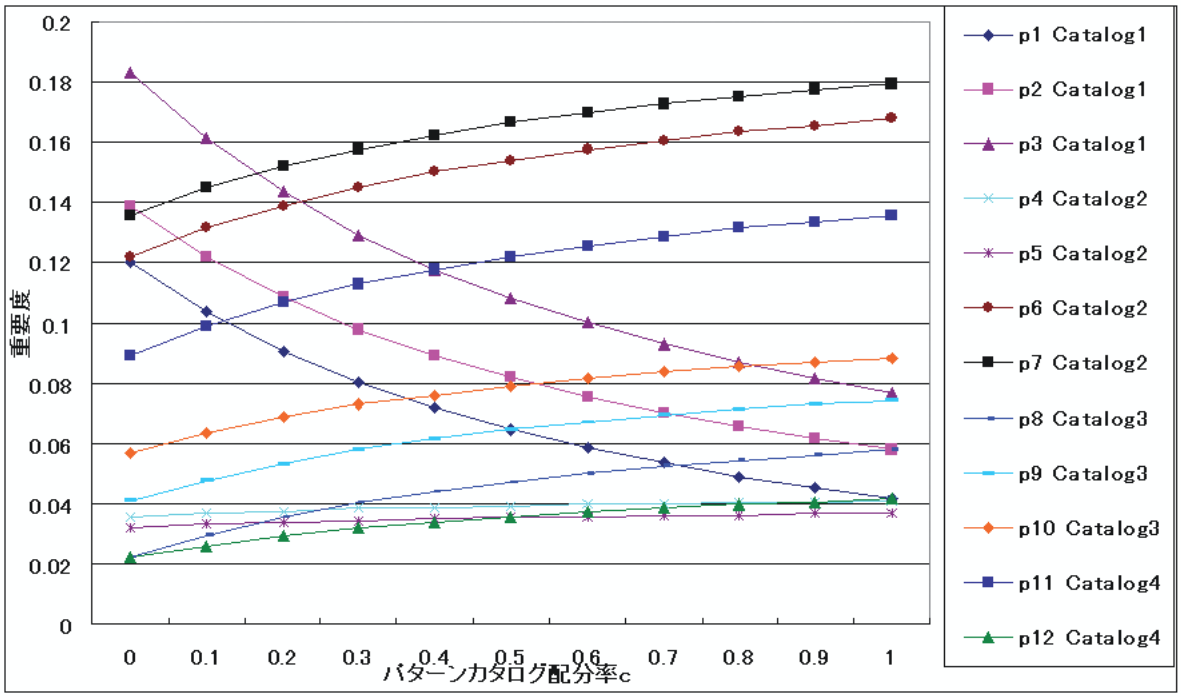

図 15　パターンカタログ配分率 c による重要度の変化

ターンカタログ帰属関係の重要度を指定するパラメータとして用いることができる．

4.2　システム有用性実験

実験 4 および実験 5 では，実在する Web サイトにあるパターン文書集合に対して，処理を変えて提案システムによる検索を行う．処理は，「重要度算出後に

クエリマッチング」,「クエリマッチング後に重要度算出」の二通り存在する．また，検索クエリに適合するパターン文書をあらかじめ人手で調査し，処理ごとに得られた検索結果の再現率および精度と，Portland Pattern Repository などの既存の検索システムにおける順序 (名前順) に並び替えた検索クエリを含むパターン文書集合の再現率と精度を求めた．適合する文書とは，利用者の検索要求に合っている文書である．再現率とは，適合する文書をどれだけ余すところなく得られたか示す割合であり，適合する文書の数を適合する文書の総数で割り求める．精度とは，適合しない文書をどれだけ得ずに済むか示す割合であり，適合する文書の数を得られた文書の数で割り求める [19]．また，検索クエリを入力してから検索結果が表示されるまでの時間 (プログラムを 10 回実行した平均のミリ秒) を各実験ごとに調べた．実験環境は Pentium4 3.20GHz 1.00GB RAM WINDOWS Home Edition である．

実験 4: パターン利用者がシステム開発時に GUI のウィンドウ設計に利用可能なパターンを得る場合を想定し，検索クエリ”window”を提案システムに入力した．

実験は，リポジトリ内に存在する 4 つの Web サイト [2] [5] [6] [18] から収集したパターンカタログ内の 131 個のパターン文書集合に対し，検索クエリ”`window`”を入力し行った．重要度の計算は，計算の終了条件 s を 10^{-8}，修正配分率 r を 0.15，パターンカタログ配分率 c を 0.15 とし,「重要度算出後にクエリマッチング」,「クエリマッチング後に重要度算出」のそれぞれの手法ごとに結果を求めた (表 4，5)．また，得られた結果の再現率と精度，検索クエリ”`window`”を含むパターン文書集合を名前順に並び替えた Portland Pattern Repository などの既存の検索システムにおける並び替えの再現率と精度を求めグラフ化した (図 17)．再現率は，GUI のウィンドウ設計に利用可能であると思われるが検索クエリ”`window`”を含まないパターン (`Command` パターン [2] [5]，`Extensibility` パターン [18]) を追加し求めた．実行時間は,「重要度算出後にクエリマッチング」が 2041 ミリ秒,「クエリマッチング後に重要度算出」が 156 ミリ秒だった．

表 4 において，`AbstractFactory` パターンや `Decorator` パターンなどの GUI ウインドウ設計において典型的なデザインパターンを上位に得た．例えば，`AbstractFactory` パターンは実行プラットフォーム環境に依存するウィンドウや関係する構成部品群の生成に利用できる．また，`Decorator` パターンは GUI 開発におけるスクロールバーやフレームといった視覚的な装飾を行うのに用いることができる．なお，`ReceiveProtocolHandler` パターンなどのネットワーク転送ウインドウに関するパターンも検索結果に出現しているが，被参照関係の少なさにより下位に位置している．`AbstractFactory` パターンは，`Bridge` パターンや `Builder` パターンなどの多数のパターンから参照されているため上位に得られた (図 16)．表 5 から,「クエリマッチング後に重要度算出」に順序を変更すると `AbstractFactory` パターンの順位が大きく下がった．これは `AbstractFactory` パターンを参照するパターンのほとんどが，検索クエリ”`window`”を含むパターン文書集合外に存在するためだと考えられる．

図 17 の再現率－精度グラフでは，各ポイントにおいて再現率と精度が高いほど性能が良いため，本実験による状況下では，名前順による並び替えより「重要度算出後にクエリマッチング」による並び替えの方が性能が良いことが分かる．名前順による並び替えと「クエリマッチング後に重要度算出」による並び替えの線は交差しているため，どちらが性能がよいかは一概に言えない．

実験結果から，提案システムは意味情報を持たない検索クエリを含んだ複数のパターンを，ある程度適切に順序付けて提示できると考えられる．検索結果に対しパターンランク法による順位付けを行わない場合，利用者は重要度の低い `TransmitProtocolHandler` パターンや `DllHell` パターンから検討してしまう可能性がある．提案システムを活用することで，パターン利用支援によるソフトウェア開発の効率化が期待される．

実験 5: 組込み開発におけるメモリ利用効率関係の技術の習得が必要な場合を想定し，検索クエリ”memory”を提案システムに入力しパターン検索を行った．

表 3 検索クエリ window を含むパターン文書集合 (名前順)

順位	パターン名	適合性	再現率	精度
1	AbstractFactory [2] [5] [18]	○	1	1
2	Decorator [2] [5]	○	0.3333	1
3	DllHell [18]	×	0.3333	0.6666
4	Observer [2] [5]	○	0.5	0.75
5	ReceiveProtocolHandler [6]	×	0.5	0.6
6	Singleton [2] [5] [18]	○	0.6666	0.6666
7	TransmitProtocolHandler [6]	×	0.6666	0.5714

表 4 検索クエリ window による結果 (重要度算出が先)

順位	パターン名	適合性	再現率	精度	重要度
1	AbstractFactory	○	0.1666	1	0.0610
2	Decorator	○	0.3333	1	0.0317
3	Singleton	○	0.5	1	0.0174
4	Observer	○	0.6666	1	0.0104
5	ReceiveProtocolHandler	×	0.6666	0.8	0.0051
5	TransmitProtocolHandler	×	0.6666	0.8	0.0051
6	DllHell	×	0.6666	0.5714	0.0014

表 5 検索クエリ window による結果 (クエリマッチングが先)

順位	パターン名	適合性	再現率	精度	重要度
1	Singleton [2] [5] [18]	○	0.1666	1	0.2378
2	ReceiveProtocolHandler [6]	×	0.1666	0.5	0.1626
2	TransmitProtocolHandler [6]	×	0.1666	0.5	0.1626
2	Decorator [2] [5]	○	0.3333	1	0.1626
2	Observer [2] [5]	○	0.3333	1	0.1626
3	AbstractFactory [2] [5] [18]	○	0.6666	0.6666	0.0873
4	DllHell [18]	×	0.6666	0.5714	0.0243

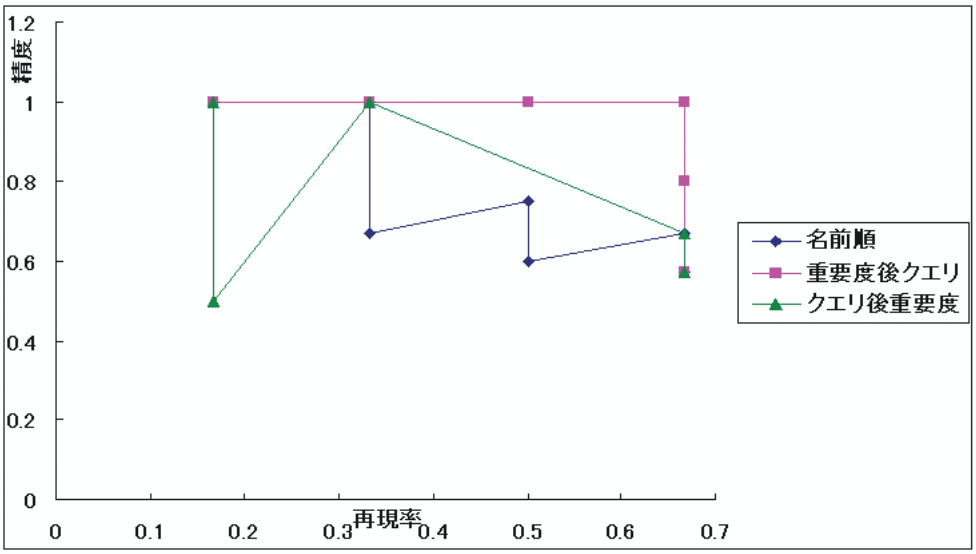

図 17 再現率－精度グラフ (window)

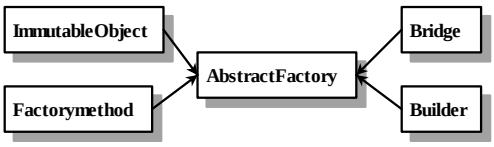

図 16 AbstractFactory **パターンの被参照関係**

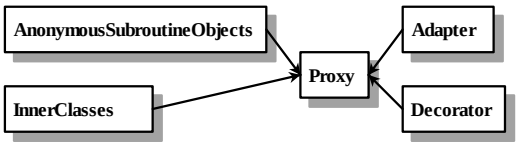

図 18 Proxy **パターンの被参照関係**

実験は，リポジトリ内に存在する 4 つの Web サイト [2] [5] [6] [18] から収集したパターンカタログ内の 131 個のパターン文書集合に対し，検索クエリ”memory”を入力し行った．重要度の計算は，計算の終了条件 s を 10^{-8}，修正配分率 r を 0.15，パターンカタログ配分率 c を 0.15 とし,「重要度算出後にクエリマッチング」,「クエリマッチング後に重要度算出」のそれぞれの手法ごとに結果を求めた (表 7，8)．得られた結果の再現率と精度，検索クエリ”memory”を含むパターン文書集合を名前順に並び替えた Portland Pattern Repository などの既存の検索システムにおける並び替えの再現率と精度を求めグラフ化した (図 19)．再現率は，組込み開発におけるメモリ利用効率の改善に有効であると思われるが検索クエリ”memory”を含まないパターン (ResourceManager パターン [6]) を追加し求めた．実行時間は,「重要度算出後にクエリマッチング」が 2031 ミリ秒,「クエリマッチング後に重要度算出」が 141 ミリ秒となった．

表 7 において，Proxy パターンや FlyWeignt パターンなどのメモリの効率的な利用に関するデザインパターンを上位に得た．例えば，Proxy パターンのうち Virtual Proxy では，メモリの節約のため必要になるまでオブジェクトを作成しない．また，FlyWeignt パターンでは，頻繁に使われる同種のオブジェクトを，メモリの効率を上げるため共有する．Proxy パターンは，Adapter パターンや Decorator パターンなどの多数のパターンから参照されているため，上位になった (図 18)．

表 8 から,「クエリマッチング後に重要度算出」に順序を変更すると Proxy パターンの順位が下がった．これは Proxy パターンを参照するパターンが，検索クエリ”memory”を含むパターン文書集合外に多く存在するためだと考えられる．

図 19 の再現率－精度グラフでは，各ポイントにおいて再現率と精度が高いほど性能が良いため，本実験による状況下では，名前順による並び替えより「重要度算出後にクエリマッチング」および「クエリマッチング後に重要度算出」による並び替えの方が性能が良いことが分かる．また,「重要度算出後にクエリマッチング」による並び替えより「クエリマッチング後に重要度算出」による並び替えの方が性能が良いことが分かる．

実験結果から，提案システムは利用者が与える検索クエリを含んだ複数のパターンを，ある程度適切に順序付けて提示できると考えられる．検索結果に対しパターンランク法による順位付けを行わない場合，利用者は重要度の低い AboutInheritance パターンや EvalToClosure パターンから習得してしまう可能性がある．従って提案システムの活用により，パターンの学習支援に基づく開発手法学習の効率化が期待できる．

5 関連研究

5.1 パターンの検索

Markus らは，扱う対象領域をセキュリティ設計に限定したパターン検索システムを提案した [15]．同検索システムでは，領域を限定することで領域特有の側面について検索条件を与えることが可能である．一方，我々の提案システムは扱う領域を限定せず，あらゆるパターンの扱いを可能である．

木梨らは，与えた要求分析記述を解析，適用可能なデザインパターンを提示するツールを提案している [8]．このツールでは，扱うデザインパターンについて人手により事前に要求分析記述断片との合致ルールを準備する必要がある．一方，我々の提案システムは，デザインパターンを含むあらゆるパターンを扱う

表 6 検索クエリ memory を含むパターン文書集合 (名前順)

順位	パターン名	適合性	再現率	精度
1	AboutInheritance[18]	×	0	0
2	ClassAsTypeCode[18]	×	0	0
3	EvalToClosure[18]	○	0.1	0.3333
4	FlyWeight[18] [5]	○	0.2	0.5
5	HighSpeedSerialPort [6]	○	0.3	0.6
6	ImmutableObject[18]	○	0.4	0.6666
7	IteratorInterface[18]	○	0.5	0.7142
8	Journaling[18]	×	0.5	0.625
9	LazyeValuation[18]	○	0.6	0.6666
10	Proxy[2] [5]	○	0.7	0.7
11	RaceCondition[18]	○	0.8	0.7272
12	SerialPort[6]	○	0.9	0.75

表 7 検索クエリ memory による結果 (重要度算出が先)

順位	パターン名	適合性	再現率	精度	重要度
1	Proxy	○	0.1	1	0.0171
2	FlyWeight	○	0.2	1	0.0121
3	IteratorInterface	○	0.3	1	0.0111
4	SerialPort	○	0.4	1	0.01
5	HighSpeedSerialPort	○	0.5	1	0.009
6	ImmutableObject	○	0.6	1	0.0085
7	ClassAsTypeCode	×	0.6	0.857	0.0052
8	RaceCondition	○	0.7	0.875	0.0048
9	Journaling	×	0.7	0.7777	0.0035
10	AboutInheritance	×	0.7	0.7	0.0033
11	LazyeValuation	○	0.8	0.7272	0.003
12	EvalToClosure	○	0.9	0.75	0.0014

表 8 検索クエリ memory による結果 (クエリマッチングが先)

順位	パターン名	適合性	再現率	精度	重要度
1	FlyWeight[18] [5]	○	0.1	1	0.1468
2	LazyeValuation[18]	○	0.2	1	0.1447
3	HighSpeedSerialport [6]	○	0.3	1	0.1114
3	SerialPort [6]	○	0.3	1	0.1114
3	Proxy[2] [5]	○	0.3	1	0.1114
4	ImmutableObject [18]	○	0.6	1	0.0844
5	IteratorInterface [18]	○	0.7	1	0.0782
6	Journaling [18]	×	0.7	0.875	0.029
6	RaceCondition [18]	○	0.8	1	0.029
7	AboutInheritance [18]	×	0.8	0.8	0.0167
7	ClassAsTypeCode [18]	×	0.8	0.8	0.0167
7	EvalToClosure [18]	○	0.9	0.9	0.0167

ことができ，パターン文書を収集する以外の人手による作業が不要である．

Web 上の代表的なパターンリポジトリ Portland Pattern Repository[1] は，パターンを検索クエリの入力による単純なキーワード検索の機能を持つが，検索結果をパターン名による順位付けで並び替えているため，パターンの有用性という面から考えると無作為に並べられているのに等しい．

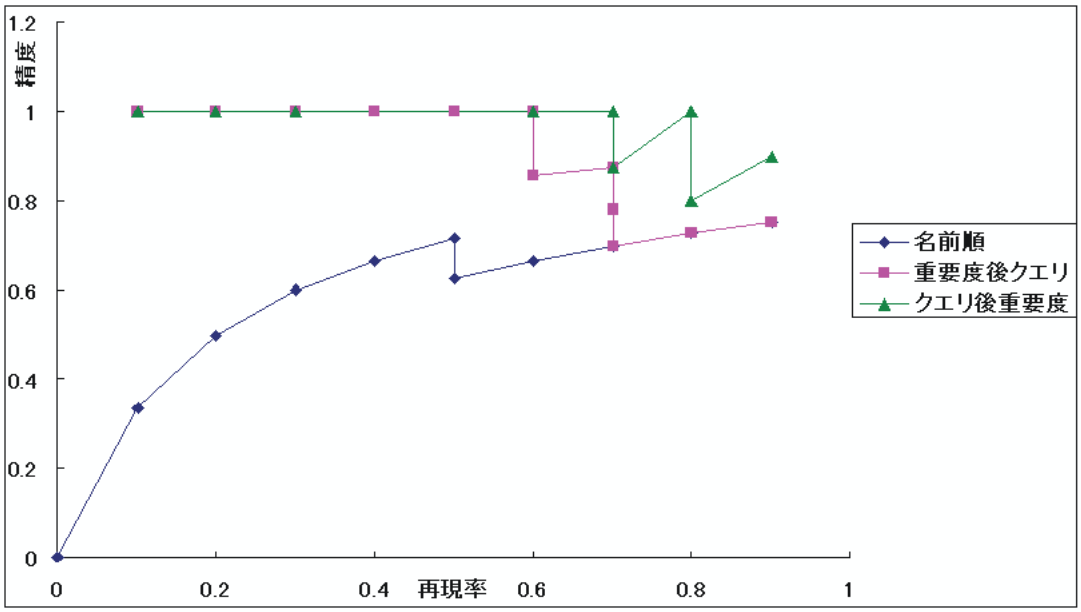

図 19　再現率－精度グラフ (memory)

5.2　関係に基づく重要度の算出

パターンランク法は，要素間の関係から各要素の重要度を算出する仕組みについて，Web ページの重要度算出手法 PageRank [11] やソフトウェア部品の重要度算出手法 ComponentRank [7] を参考にしている．PageRank は，重要な Web ページから参照される Web ページの重要度は高いという規則を持つ．ComponentRank は，よく利用される部品や重要な部品から利用される部品の重要度は高いという規則を持つ [7]．PageRank が Web ページ全般を，ComponentRank がソフトウェア/ソフトウェア部品をそれぞれ適用対象とするのに対して，パターンランク法はパターン文書を適用対象としており，同一パターンカタログへの帰属や同一パターンの異なる文書表現といったパターン特有の事柄を考慮している．

5.3　頻度に基づく重要度の算出

Vokac は幾つかの文献データベースを用いて，デザインパターンカタログ内の個々のデザインパターンの学術文献における出現頻度を調査している [16]．ただし，一般に学術文献は研究対象としてパターンを扱うため，パターンランク法の適用が目指すパターン利用支援とは目的が異なる．

Hahsler は多数のオープンソースソフトウェアにおけるデザインパターンの利用頻度を測定している [4]．測定方法は，バージョン管理システムにおけるログコメント内のデザインパターン名を解析するものである．このような，実際のソフトウェア開発におけるパターンの利用頻度は，パターンランク法におけるパターン間の関係に基づく重要度と異なるパターンの側面を表すと考えられる．具体的には，実際のソフトウェア開発におけるパターンの利用頻度にはパターンを利用する開発者の観点が反映され，パターンランク法におけるパターン間の関係に基づく重要度にはそのパターンを参照する側のパターンの作者の観点が反映されていると考えられる．将来において，両者を統合して活用する検索システムの実現を目指す．

6　おわりに

本稿では，収集したパターン集合についてパターン特有の関係に基づいて重要度を算出する手法 (パターンランク法) を提案し，同手法を検索結果の順序付けに用いるパターン検索システムを構築および提案した．パターンランク法の利用における各種パラメータ

の妥当な値を求める予備実験と，Web 上から収集したパターン文書集合を実験サンプルとした提案システムの有用性を検証する実験を行った．実験の結果，提案システムは，利用者が与える検索クエリを含む複数のパターンを，適用や学習の観点から見て適切に順序付けて提示することを確認した．提案システムの利用することにより，パターンの利用支援によるソフトウェア開発の効率化や，パターンの学習支援による開発手法学習の効率化が期待される．

また，提案システムはパターンを作成する立場 (作者) にとっても有用であると考えられる．具体的には，パターンの再利用機会とフィードバック機会の増大，および関係するパターンの効率的な検索を行える利点がある．

本稿では，ソフトウェアプロジェクトにおいて同時に使われることがあるパターン間関係を扱っていない．これは，ソフトウェアプロジェクトにパターンが利用されていることを自動的に発見する事が難しいためである．上述のパターン間関係を得る事ができれば，実際の利用情報を盛り込んだ重要度計算が可能となる．また，5.3 節で述べた具体的な利用頻度に基づく重要度や，利用者に検索される回数に基づく重要度も同時に考える必要がある．これらについても，パターン間関係に基づく重要度と統合する枠組みを今後提案したい．

また，提案手法を特定の問題領域に関する大規模なパターン集合に適用し，複数の被験者によるパターンの検索と具体的な検討および再利用を伴う実証実験を通じて，特定の問題領域における提案システムの実用性や有用性を詳細に検証したい．例えば現在，組込みシステム開発におけるパターン集合を対象として，重要度算出方法の有用性検証を進めている [10].

参 考 文 献

[1] Cunningham & Cunningham, I.: Portland Pattern Repository. http://c2.com/ppr/.

[2] DavidVanCamp.com: The Object-Oriented Pattern Digest. `http://patterndigest.com/`.

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

[4] Hahsler, M.: A quantitative study of the adoption of design patterns by open source software developers, Free/Open Source Software Development,Idea Group Publishing, 2004.

[5] Huston, V.: Huston Design Patterns. `http://home.earthlink.net/ huston2/dp/patterns.html`.

[6] Inc.EventHelix.com: Embedded Design Pattern Catalog. `http://www.eventhelix.com/RealtimeMantra/PatternCatalog/`.

[7] Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M., and Kusumoto, S.: Ranking Significance of Software Components Based on Use Relations, *IEEE Transactions on Software Engineering*, Vol. 31, No. 3, 2005, pp. 213–225.

[8] 木梨充高, 小飼敬, 上田賀一: デザインパターン利用促進のためのモデリング支援ツールの開発, 情報処理学会第 *144* 回ソフトウェア工学研究会 情報処理学会研究会報告, Vol. 2003, No. SIGSE-144, 2004.

[9] Kubo, A., Washizaki, H., Takasu, A., and Fukazawa, Y.: Analyzing Relations among Software Patterns based on Document Similarity, *Proc. of International Conference on Information Technology:Coding and Computing (ITCC2005)*, 2005.

[10] 中山弘之, 鷲崎弘宜, 久保淳人, 深澤良彰: 組込みシステム開発におけるソフトウェアパターンの重要度, 組込みソフトウェアシンポジウム *2005(ESS2005)*, 2005.

[11] Page, L., Brin, S., Motwani, R., and Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web, 1998. `http://dbpubs.stanford.edu:8090/pub/1999-66/`.

[12] Porter, R. and Calder, P.: A Pattern-Based Problem-Solving Process for Novice Programmers, *Proc. 5th Australasian Computing Eeducation Conference*, 2003.

[13] Riehle, D.: Composite Design Patterns, *Proc. 12th ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'97)*, 1997.

[14] Rising, L.: *The Patterns Handbook,Techniques, Strategies, and Applications*, Cambridge University Press, 1998.

[15] Schumacher, M.: Security engineering with patterns : origins, theoretical model, and new applications, *LNCS*, Vol. 2754, Springer-Verlag, 2003.

[16] Vokac, M.: Defect Frequency and Design Patterns: An Empirical Study of Industrial Code, *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, 2004.

[17] Vrsalovic, V.: Human Based Pattern Classification. `http://patterns.ing.puc.cl/`.

[18] Walters, S.: Perl Design Patterns. `http://perldesignpatterns.com/perldesignpatterns.html`.

[19] 徳永建伸: 情報検索と言語処理, 東京大学出版会, 1999.

[20] 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 情報処理学会研究会報告, Vol. 2003, No. 55 (SE-141), 2003.

[21] 馬場肇: Google の秘密 - PageRank 徹底解説.

http://www.kusastro.kyoto-u.ac.jp/ baba/wais /pagerank.html.