Web ページ内のプログラム記述に対する例示を用いたアノテーションの付加に関する研究

2009SE099 加藤 大典　2009SE301 渡邉 大輔　2009SE321 吉田 匡志
指導教員：吉田 敦

1　はじめに

現在，インターネット上にソースコードを公開し，自由にそのソースコードの閲覧・拡張を行える「オープンソース」[1] が普及している．これに伴い，ソースコードやその一部の記述 (以下，コード記述とする) を Web 上で閲覧できる環境が提供されてきた [2]．プログラマはコード記述を閲覧する際，そのコード記述に対しアノテーションを行うことがある．それによってプログラマ自身がコード記述を再読するときや，他のプログラマが読むときに，予備知識を得ることができ，コード記述を理解する時間の短縮ができる．しかし，このようなアノテーションを Web 上のコード記述に対して行うことには問題がある．公開されているコード記述をダウンロードしアノテーションを行う方法ではコード記述が新版された際にアノテーションを引き継げず，またオープンソースのコード記述の記述量は膨大である場合があり，そのようなコード記述にはアノテーションの付加作業に大変な手間を要する．よって，Web 上のコード記述に対しアノテーションの付加ができ，コード記述の新版に対応でき，また，少ない時間でアノテーションの付加ができる環境の提供が望まれている．

アノテーション付加の支援に対し様々なツール [3, 4] が提供されている．しかし，それらのツールはエディタ環境下での動作を想定し提供しており，Web 上のコード記述に対するアノテーションは困難である．また，提供されているツールではアノテーション付加の際に対象コード記述の字句情報を参照するが，特定の関数宣言部の省略や特定の型を持つ変数を対象とした強調など，コード記述の構文情報を参照するアノテーションの付加が困難な場合がある．

本研究の目的として，これらの問題に対し，構文情報を用いて Web 上のコード記述に対しアノテーションが行える環境の構築を行う．

本研究で提案する環境では，コードリーディングを行うプログラマ（以下，ユーザとする）自身がコード記述に対し強調表現や省略表現などを用いて見栄えを操作・変更することで特定のコード記述に対しアノテーションを行える機能を提供する．このような表現で，コード記述に対し視覚的に情報を付加するものを視覚的アノテーションと呼称する．コード記述の記述量が膨大である場合に手間を要する問題には視覚的アノテーションの付加を直感的かつ簡潔な入力できる仕組みを提供することで解決する．その方法として，ユーザによるアノテーション方法の例示からコード記述の一部または全体に対する視覚的アノテーションの付加を考える．これは，対象のコード記述をもとにユーザが実際に書き換えた後の形を記述することで，ユーザがコード記述に対し行いたい視覚的アノテーションの作成・付加を簡潔に入力できるからである．また，例示として記述する視覚的アノテーションの表現としては見栄えを容易に操作・変更できる HTML タグを用いる．

2　関連研究・技術

先行技術 [3] は，コード記述に対し注釈記述の付加によるアノテーションを行う．また，字句解析による識別子の検索と記号表によって，注釈を付加した識別子に関連した識別子にも自動的に同一の注釈を付加する注釈伝搬という機能を提供する．アノテーションを行うために用いる情報としてコード記述の字句情報と記号表を利用する点で，字句情報だけでなく構文情報も利用する本研究と異なる．

先行研究 [5] は，コード記述の理解支援を目的として，Web 上のコード記述の閲覧機能をユーザが拡張できるよう，新たな機能の挿入を支援する．機能の挿入に Bookmarklet を用いることで Web 上のコード記述に対し，クライアントサイドからの機能の挿入・拡張作業の軽減化に対する有効性を確認している．理解支援機能の挿入方法として外部の JavaScript ファイルを呼び出す点で，理解支援機能の挿入方法としてユーザの例示によるアノテーションの付加とする本研究と異なる．

3　例示による視覚的アノテーションの付加

3.1　視覚的アノテーション付加の流れ

提案環境では，Web サイト上に記述されたコード記述を読みながら，アノテーションの付加を行える．アノテーションは HTML タグを用いて記述するので，言葉による注釈だけでなく文字の装飾も利用できる．

提案環境を用いてアノテーションを付加する作業の流れは次の通りである．ユーザはまず Bookmarklet を用いて提案環境を呼び出す．次にユーザはコード記述からアノテーションを付加したい箇所をマウスのドラッグにより選択する．その選択したコード記述が入った編集可能なテキストエリアが現われるので，ユーザは装飾に使いたい HTML タグを加えることでアノテーションを付加する．これにより，類似した箇所にも自動的にアノテーションが付加される．アノテーション付加後のコード記述は Web ブラウザ上でそのまま閲覧できる．これを実現するために，内部ではコード記述をアノテーションの付加された記述へ変換する変換ルールを自動生成し，抽出・解析された Web ページ内のコード記述に適用する．アノテーションが付加されるまでの流れを図 1 に示す．

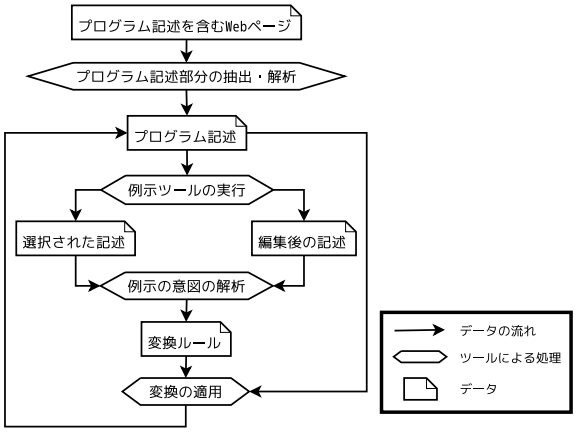

図 1 アノテーション付加の流れ

3.2 コード記述の解析方法と変換ルールの表現方法

本研究では，C 言語を対象とした解析・変換環境である TEBA [6] を用いてプログラムの解析と変換を行う．TEBA を用いると解析対象のコード記述が完全ではなく断片である場合でも解析が可能である．また，拡張により C 言語記述と HTML 記述が混在する記述を解析することが可能であり [7]，本研究ではこの方法を用いて Web サイト内のコード記述を解析する．解析結果は TEBA の属性付き字句系列として保持する．

TEBA はパターン変換によるコード記述の変換が可能である．TEBA のパターン変換は，コード記述を解析し属性付き字句系列として表現したものに対し，文字列に対する正規表現を用いて変換前の字句列 (Before 記述) として定義された記述とのマッチングを行い，マッチした箇所を変換後の字句列 (After 記述) として定義された記述に置き換える．一組の Before 記述と After 記述を「パターン変換記述」と呼ぶ．パターン変換記述では 「パターン変数」 を用いて字句の代わりに属性値を指定したマッチング条件も表現できる．パターン変換記述を用いることで，アノテーションを付加する箇所の記述を Before 記述，付加した書き換え後の記述を After 記述としてそれぞれ短い記述で表現できる．よって，アノテーションを付加する際には，ユーザが入力したタグを追加・削除するパターン変換記述を生成して適用する．

3.3 変換ルールの生成方法

例示の内容からパターン変換記述を生成するときには，ユーザが意図する類似した記述にも適用する必要があることから，可変と想定される箇所をパターン変数に置き換え一般化する必要がある．本研究では一般化する箇所を決定しパターン変数に置き換えることを抽象化と呼称する．

ユーザの意図と合うように置き換え箇所を決めることは難しい．図 2 の例では，ユーザが「int 型の変数にアノテーションを付加する」という意図で例を与えた場合を想定し，「変数名のみを一般化したパターン変換記述が生成する」場合を考えている．すべての字句の一般化を行うと，型名によらずアノテーションが付加され，ユーザの意図に沿わない．この場合，「変数名のみを一般化する」という情報を得ることができれば意図通りの抽象化が可能となる．

例示のみでは，ユーザの意図を解釈するための情報が不足していることから，例の与え方に最低限の制約を加えることで情報の不足を補う方法を採用する．

【ユーザによる例示】
選択された記述　編集後の記述
`int num` => `int <b>num</b>`

【すべての字句を一般化する抽象化の結果】
`型名 変数名` => **`型名 <b>変数名</b>`**

【変数名のみを一般化する抽象化の結果】
`int` **`変数名`** => `int <b>`**`変数名`**`</b>`

注: コード記述の中の日本語で記述された部分は
一般化された構文要素を表す．

図 2 抽象化の例

3.4 例示の制約

情報の不足を補うにあたり，補完方法として考えうる次の 3 つを考える．以降の節では，各方法について議論しどれを用いるべきか評価する．

- 方法 1: ユーザから詳細な例示を得る
- 方法 2: ユーザから設定情報を得る
- 方法 3: アノテーションの履歴から補う

3.4.1 ユーザから詳細な例示を得る方法

この方法ではユーザが与える例の中に補助的な情報を加える．例示の手順は基本的には 3.1 節のものと同様であるが，HTML タグに加えて，一般化を指定する特殊な記号やタグを付加する必要がある．特殊な記号やタグには各種類ごとに専用のフィルタを用意する．これにより，フィルタを追加・修正・削除することで，使用できる記号やタグをカスタマイズできる．加えられた特殊な記号やタグは一般化の際にフィルタにより取り除かれる．

3.4.2 ユーザから設定情報を得る方法

この方法では解釈方法を制約するために抽象化の方法を事前に設定する．ユーザは，提示される抽象化方法に対して使用の有無を設定する．各抽象化方法に対し対応するフィルタが用意されており，ユーザが指定したフィルタを繋げたフィルタ群に例示を通すことで抽象化の処理が行われる．フィルタを追加・修正・削除することで，抽象化の方法をカスタマイズできる．本研究で想定する抽象化方法の例を以下に示す．

- 変数名を一般化する
- 制御文内の実行文を一般化する
- 制御文の実行文を囲む括弧の有無を無視する

3.4.3 アノテーションの履歴から補う方法

この方法では方法 1 や方法 2 を用いたアノテーション付加の履歴から環境が自動的に抽象化の内容を決定する．環境によって決定された抽象化は常にユーザの意図と一致するとは限らないが，ユーザは抽象化の仕方を意識することなくアノテーションを付加できる．例示の手順は 3.1 節の通りである．

抽象化の方法は，過去の履歴の中で例示の字句系列との差異が最も少ないものを探索することで決定する．抽象化内容の記録は，方法 1 であれば記号やタグを追加する編集の前後の差分を抽象化することによって得られる変換記述であり，方法 2 であれば抽象化方法の選択の仕方である．

3.4.4 各方法の評価

各方法の「設定を行う主体」と「設定内容の詳細さ」を明らかにし，精度と直感性を評価する．評価結果は表 1 のようになる．この結果から，自動化より高い精度が求められる場合は方法 1 と方法 2 が有効であり，直感的な入力ができるよう自動化が優先されるなら方法 3 を用いるのが有効である．本研究では，処理の自動化よりユーザの意図を記述する方法を明らかにすることに焦点があるので、方法 1 と方法 2 のみを用いる．

表 1　各方法の評価

	方法 1	方法 2	方法 3
設定を行う主体	ユーザ	ユーザ	環境
抽象化の設定	詳細設定	粗い設定	自動設定
精度	高い	中程度	低い
直感性	低い	中程度	高い

4　タグ情報の管理

4.1　パターン変換時の問題点

本研究では，直感的な入力を実現するために，Web ブラウザ上に表示されるテキストをユーザに編集させ，HTML タグを含むページ記述は提示しない．しかし，アノテーション付加は HTML タグを含むページ記述に対して行うので，ユーザの例示から得られるパターン変換記述では，タグの情報が欠落が生じ適用できない．よって，ユーザの例示と元ページ記述を融合したパターン変換記述を生成する必要がある．

4.2　タグのユーザへの提示について

ユーザの例示と元の HTML 記述を融合したパターン変換記述を生成する最も簡単な方法は，ユーザに対してすべてのタグ情報を提示することである．しかし，Web ページ上に元々存在している HTML タグ（以下,「元タグ」とする）は Web ページのスタイル情報を定義しているので，パターンを生成する際には必要な情報であるが，例示を用いてアノテーションを付加する際には不要である．アノテーション付加に関係しない情報が増えることは，ユーザの直感的な入力の妨げとなる．また，ユーザが例示を行う際に，誤って元タグを消去した場合，Web ページの見栄えが崩れる問題が生じる．よって，直感的な入力を保つことと意図しない見栄えの崩れを避けることを実現するためには，ユーザに対してアノテーション付加の際に提示する情報を選択する必要がある．

4.3　隠蔽する情報と提示する情報

プログラムのパターン変換記述を生成するときに必要な情報は，次の 3 つである．

- コード記述
- 元タグ
- ユーザが追加したタグ

ユーザに提示すべき情報はコード記述とユーザが追加したタグである．アノテーションを付加する際に，ユーザは追加したタグを再編集することがあるので，元タグとは区別する必要がある．そこで，字句解析の段階で、元タグとユーザが追加したタグを別の種類の字句として扱うことで区別する．

4.4　パターン生成の方法

パターン変換記述生成の際に，ユーザがアノテーション付加の際の入力と変更された部分の元タグを含むコード記述の差分を求め，その差分から元タグとユーザが付加したアノテーションを含むパターンを生成する．しかし，元タグを復元してパターン生成を行うと，元タグとユーザの追加タグとの間で開始タグと終了タグの入れ子関係が不適切になる場合がある．そこで，元タグとユーザが追加したタグを入れ替えるフィルタ（以下,「タグの入れ替え規則」とする）を事前に定義し，ユーザの例示とコード記述から生成されたパターン変換記述に対して，タグの入れ替え規則を適用しタグの入れ子関係を修正する．

5　評価と考察

提案環境を実装し，オープンソースに対して適用したときに，意図通りにアノテーションができるか検証する．このとき，次の観点に着目し評価と考察を行う．

1. 制約に従ったアノテーション付加
2. タグの入れ子関係の修正

5.1　制約に従ったアノテーション付加

方法 1 と方法 2 をタグを含まないコード記述のテキストに対して適用して，実際にアノテーション付加をできるかを確認する．前提条件として，評価実験で扱う HTML タグは一種類とする．これは，タグを 1 種類について扱えれば，あらゆるアノテーションに対応できるからである．対象の構文要素，変数名，関数，型，制御文に対してアノテーションを付加した実験結果を表 2 に示す．

方法 1 では一部の制御文に対して，方法 2 では一部の

変数，型に対してユーザの意図したアノテーショになら なかった．このことから，必ずしも意図通りにならないが HTML タグが挿入できることは確認できた．意図が反映 できなかったものについては，方法 1 と方法 2 の間で相互 に補完できることから，方法 1 と方法 2 を組み合わせて使 用することで回避することができると考える．

表 2　各方法の実験結果

	変数	関数	型	制御文
方法 1	○	△	○	△
方法 2	△	△	△	○

5.2　タグの入れ子関係の修正

前節と同じコード記述にタグを含めたものを対象に，方法 1 を用いてアノテーション付加を行いタグの入れ子関係が適切に修正されるか実験をした．その結果，図 3 のように適切に補正されない事例が存在した．これは入れ替え規則の精度が悪いことが原因であった．また，図 4 の場合，ユーザ意図は変数 m も演算子 + と同じ色を付けたいが，変数 m を内側で囲っている元タグが優先されており，意図通りの色になっていないという，CSS の優先度の問題でユーザの意図するアノテーションを得られない問題も判明した．よって，入れ替え規則の精度の向上と，CSS の解析を行い，ユーザの意図通りにアノテーションが反映される方法を検討する必要がある．

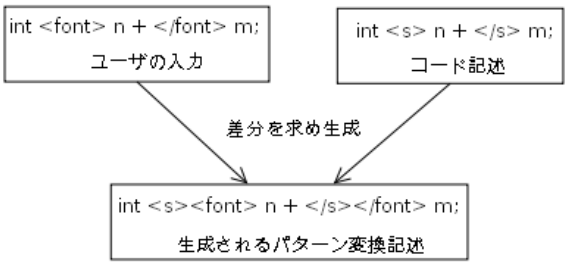

図 3　適切に補正されないパターン変換記述の例

6　おわりに

本研究ではユーザがコード記述に対しアノテーションの付加を行える環境を提供した．例示を用いて HTML タグによるアノテーションを行う仕組みを提案し，ユーザの意図通りに例示を解釈できる方法と，対象のコード記述に合わせて付加される HTML タグを最適化する方法を明確化することで，ユーザの要求に特化したアノテーションを直感的な操作で行うことが可能になった．Web 上の自然言語の記述に対しては，その記述に対応した解析が行えれば，本研究で提案した例示の手法を用いてアノテーションの付加を行える．今後の課題として，アノテーションを Web ページの URL に紐づけて保存する機能などの応用的な機能の導入を検討する必要がある．

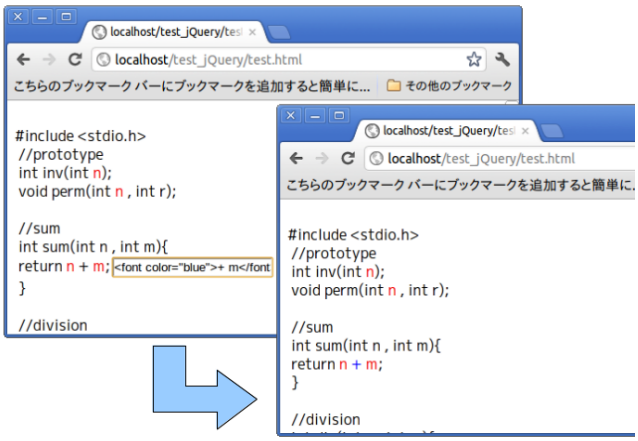

図 4　意図しないアノテーション付加の例

参考文献

[1] “Open Source Initiative, ” http://opensource.org/.

[2] “SourceForge, ” http://sourceforge.jp/.

[3] 悦田 翔悟, 田中 昌弘, 石尾 隆, 井上 克郎, “識別子に対する注釈付加ツール DocumentTag, ” 日本ソフトウェア科学会 FOSE2009, ソフトウェア工学の基礎 XVI, pp. 323-324.

[4] Tama Communications Corporation, “GNU GLOBAL source code tagging system, ” http://www.gnu.org/software/global/

[5] 森島敦子, 椎名優貴, 土屋陽平, “クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究 —プログラム記述への理解支援機能の挿入方法—, ” 南山大学 数理情報学部 情報通信学科 2011 年度卒業論文要旨集.

[6] 吉田 敦, 蜂巣 吉成, 沢田 篤史, 張漢明, 野呂 昌満, “属性付き字句系列に基づくソースコード書き換え支援環境, ” 情報処理学会論文誌, Vol.53, 7, 1832-1849, 2012.

[7] 松野秀泰, 森瑞歩, “クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究 —HTML 文書内のプログラム断片の抽出と解析—, ” 南山大学 数理情報学部 情報通信学科 2011 年度卒業論文要旨集.