プログラミング初学者に向けた空文の修正支援環境

橋浦研究室　　118I050 大島天太

1　はじめに

C 言語や Java を学ぶ際にプログラミング初学者は，不注意などのミスにより意図しない動作をするプログラムを書いてしまうことがある [1]．加えて，不注意などで起こるミスを修正するのは初学者には困難である [2]．そのため, 修正箇所を発見して修正を行うまでに時間がかかることがある．

このようなミスを引き起こすものの 1 つに空文という何もしないという処理がある．この空文を if 文や for 文等の文末に付けてしまうと，意図しない動作を引き起こすことがある．このような余分なセミコロンは不注意によって発生するミスである [1]．空文によって発生したミスは，警告文などで出力されることは少なく，発見に困難がなってしまい初学者が修正するのに時間がかかる．

リスト 1: 空文を含むプログラム例

```
1        if(a<10);{
2         for(int i=a;i<10;i++){
3           a=a++;
4         }
5        }
```

2　空文の使用例

本研究では，空文が意図的に使用される場合を明らかにするために複数の OSS のソースコードを利用して予備調査を行った．

表 1: 予備調査の結果

#	対象	if 文の数	if 文の空文	for 文の数	for 文の空文	while 文の数	while 文の空文
1	Hadoop	2,353	0	189	19	28	17
2	Arrow	0	0	0	0	0	0
3	James	203	0	11	0	7	0
4	OpenNLP	335	0	26	0	7	1
5	Pivot	2	0	0	0	0	0
6	Maven	0	0	2	0	0	0

表から熟練したプログラマーが空文を意図的に使用することはあるがその頻度は非常に低いことが分かる．

3　関連研究

空文が関係している研究の一つに Altadmri ら [3] の研究がある. この研究でが学生が書いた Java のソースコードを収集し，学生がどのような欠陥を起こすのか，欠陥が起きる頻度と修正にかかる時間を調べている. この研究では空文による欠陥の修正には平均 401 秒かかっていることが判明している．

また掛下と小田 [1] の研究では，正規表現を用いた C 言語の落とし穴検出ツールを開発している. この研究の C 言語の落とし穴とはプログラマーが意図していない動作を引き起こす欠陥のことを指しているこの落とし穴の一つとして空文が含まれている．

4　研究目的

本研究では前述の問題を解決するためにプログラミング初学者を対象とした空文による欠陥の修正を支援するツールの開発を行う．ツールによって，修正に時間がかかる空文による欠陥をより早くかつ発見しやすいように支援することを目指す．研究にあたっては，以下の RQ を設定した．

- RQ: 開発したツールを使用した場合と使用しない場合で欠陥の修正率に差が出るか

5　提案手法

本研究では初学者がプログラミングを行う際に修正の支援を行えるツールを想定している．そのためツールに求められるのは以下の機能である．

1. IDE 環境で使用が可能

   初学者がプログラミングをする際に使用すると考えられるのは Visual Studio や Eclipse といった IDE 環境である．そのため，これらの環境を使用している状態で利用できるツールが望ましい．そのため，Visual Stadio や Eclipse で利用できるプラグインとして実装する．

2. 初学者が空文を発見しやすくなるような機能

   IDE 環境ではエラーが発生した際や構文に誤りがあるとき，エラーメッセージを表示したり，誤りがある箇所の強調表示などを行う機能がある．それらの機能を余分なセミコロンによる空文に対応させる必要がある．

6　実装

統合開発環境の一つである Eclipse [4] のプラグインとしてツールを実装する．Eclipse [4] のプラグインの一つである，ASTview [5] にソースコード上の空文を認識したときに Eclipse [4] のマーカー機能でメッセージを表示する機能を追加することで実装を行った．

ツールの対象は Java である．

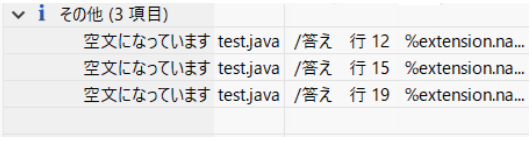

図 1: ツール使用時のマーカー表示

7　評価

日本工業大学の学生 12 名を対象にプラグインを使用しないグループ (統制群) とプラグインを使用するグループ (実験群) に分けて実験を行った．実験前に Eclipse [4] の使用方法と実験群にはプラグインの使用方法を教え, プログラムを 5 問修正してもらう. プロ

グラムは特定の欠陥が入った Java のソースコードである．欠陥を完全に修正し正しい出力結果を出力できた状態を修正できたと判定する．欠陥は先行研究 [3] から一部のカテゴリを抜粋して使用した. 以下の表 2 がそのカテゴリを示す.

表 2: 間違いのカテゴリ

#	カテゴリ
等号	代入演算子と比較演算子の混同
非対称	アンバランスな括弧, 中括弧, 角括弧, 引用符, またはこれらの異なる記号を互換的に使用する
論理演算子	無関係な意図的要素の作成短絡評価子 (&&および ||) と従来の論理演算子 (&および|) を混同
空文	if 文,for 文,while 文の後の不要な「;」*
区切り	for 文内の間違った区切り文字 (「;」ではなく「,」を使用)
括弧	if 文の条件を括弧ではなく中括弧内に挿入する
比較	>=,<=ではなく=>,=<を使用する
メソッド末	メソッドの末尾の「;」

統制群と実験群で修正率に差が出るかを調べるため, 実験の結果から修正できたプログラムの数と修正できた各カテゴリの数, 全てのカテゴリの合計修正数をフィッシャーの正確確率検定を用いて検定を行った.

表 3: プログラム 5 問の修正数

#	修正完了	修正未完	合計
統制群	13	17	30
実験群	30	0	30
合計	43	17	60

表 3 のプログラム全体の修正数には有意差が認められ ($\alpha = 0.05$, $p = 2.318 * 10^{-5}$), 統制群と比べて実験群のほうが修正率が高かったと言える.

表 4: カテゴリの修正率の検定結果

#	統制群		実験群		p 値
	修正完了	修正未完	修正完了	修正未完	α =0.05
等号	14	4	17	1	0.33766
非対称	12	0	12	0	1
論理演算子	4	2	4	2	1
空文	13	17	29	1	7.95496×10^{-6}
区切り	6	0	6	0	1
括弧	12	0	12	0	1
比較	15	9	24	0	0.00156*
メソッド末	12	0	12	0	1
全体	88	32	116	4	1.49823×10^{-7}

表 4 の空文の修正率にも有意差が認められ ($\alpha = 0.05$, $p = 3.093 \times 10^{-7}$), 統制群と比べて修正率に非常に差があると言え, 実験群の修正率が非常に高いと言える.

表 4 の比較の修正率にも有意差が認められ ($\alpha = 0.05$, $p = 0.00078$), 統制群と比べて修正率に差があり, 実験群のほうが修正率が高いが空文ほどの差はない.

表 4 のカテゴリ全体の修正率は有意差が認められ ($\alpha = 0.05$, $p = 3.977 \times 10^{-9}$), 統制群と比べて修正率に差があり, 実験群のほうがより多くのカテゴリを修正できていると言える．また, カテゴリごとの修正率は空文と比較以外のカテゴリに有意差が認められなかったため, 統制群と実験群の差は誤差の範囲内と言える. 以下の表 5 は習熟による影響を受けている可能性がある被験者の人数と割合を示している.

表 5: 習熟の影響者数

#	人数	割合
習熟の影響	9	0.75

表 5 から空文に有意差は認められているが, 習熟による影響は高いと考えられる．ただし, 統制群の空文の修正数と実験群の空文の修正数を比較すると, 習熟による影響だけでなくツールの使用の有無も修正率に影響を与えていると言える.

比較の修正率の差の検定結果からツールによる影響だけでなく被験者の能力差の可能性はあるが, 他のカテゴリ結果を踏まえるとツールの使用の有無のほうが影響していると考えられる.

空文と比較では全体修正率に与えている影響の差を考えると空文の修正率のほうがより強い影響を与えているため, ツールの使用の有無が修正率に差を生んでおり, 本研究で作成したツールは空文の修正支援に有効であると考えられる.

8 まとめ

本研究では, 初学者がプログラムの修正を行う際に修正が困難と言える空文の修正を支援するため, 空文に対してメッセージを表示する機能を持つ Eclipse プラグインの開発を行った. 空文の修正率に有意な差が見られたが習熟による影響があった. しかし, 両群の空文の修正数の差を考えるとツールの使用の有無の効果があったと言える結果になった.

今後の課題として, 今回の手法では標本数が少ないためより多くの標本で修正率に差が表れるか確かめること, また被験者にプラグインを使わずプログラムを修正してもらい, その後にプラグインを使用して同一のプログラムを修正してもらうことで空文の修正率に影響があるかを確かめることが挙げられる.

参考文献

[1] 掛下哲郎，小田まり子，“正規表現による c プログラムの落とし穴検出ツール,” 情報処理学会研究報告ソフトウェア工学，Vol.1992，No.59，pp.43–150，1992.

[2] 岡本雅子，喜多一，“プログラミングの「写経型学習」における初心者のつまずきの類型化とその考察,” 滋賀大学教育学部附属教育実践総合センター紀要，Vol.22，pp.49–53，2014.

[3] Amjad Altadmri, Neil C. C. Brown, “37 million compilations:investigating novice programming mistakes in large-scale student data,” Proceedings of the 46th ACM Technical Symposium on Computer Science Education, Vol.15，pp.522–527, 2015.

[4] Eclipse Foundation, “Eclipse,” `https://www.eclipse.org/`. accessed 2023/07/13

[5] Eclipse JDT UI team, “Astview,” `https://marketplace.eclipse.org/content/astview#details`. accessed 2025/11/27