

# Thinking Aloud about Confusing Code

## A Qualitative Investigation of Program Comprehension and Atoms of Confusion

---

Dan Gopstein, Anne-Laure Fayard

Sven Apel, Justin Cappos

*In Proceedings of the 28th ACM Joint Meeting on European  
Software Engineering Conference and Symposium on the  
Foundations of Software Engineering (ESEC/FSE 2020).  
Association for Computing Machinery, 605–616*

# Atom of Confusion

- 正しく理解するのが難しい最小単位のコード片 [11]
- 先行研究[11]で15種類定義されている
  - Reversed Subscripts `1["abc"] -> "abc"[1]`
  - Operator Precedence `0 && 1 || 2 -> (0 && 1) || 2`
  - etc...

# 問題点

---

- 先行研究[11]ではAoCを定義したが、説明と批評に重点が置かれていない
- AoCへの対処方法を構築するには不十分な状態である

# 目的・アプローチ

---

- 目的

- プログラマがAoCの誤解を引き起こす流れと理由を明らかにする

- アプローチ

- 思考発話法, インタビューを用いて質的分析を実施する

# コード片選定

---

- 被験者に見せるコード片を先行研究[11]のデータセットから選定
- 以下の構成で選定
  - 5つのAoC(C)
  - 1つの通常のコード片(NC)
  - 1つの仮説が設定されているが検証されていないAoC(HC)
  - 1つの単純化されたHC(HNC)

# データ収集

---

- コード片を見せてコード片の出力結果を考えて回答してもらう
- 回答は筆者と被験者と会話しながら行う（思考発話法）
- 被験者の回答についての理由をインタビューする

# 被験者の内容

- 被験者の内訳
  - 5人の学生
  - 4人のC++アプリケーション開発者
  - 5人のC++ライブラリ開発者

Table 1: Programmers recruited to participate in the study

| Subj. | Group         | Age | Preferred Language | C/C++ Proficiency | Years Progr. |
|-------|---------------|-----|--------------------|-------------------|--------------|
| 4168  | Student       | 22  | C++ / Python       | 4                 | 15           |
| 3316  | Student       | 22  | Go                 | 3                 | 4            |
| 4281  | Student       | 24  | C++                | 4                 | 7            |
| 9112  | Student       | 21  | JS / Kotlin        | 3                 | 6            |
| 3787  | Student       | 22  | Python             | 4                 | 7            |
| 6061  | C++ User      | 32  | C++                | 5                 | 27           |
| 8888  | C++ User      | 35  | C++                | 5                 | 29           |
| 4304  | C++ User      | 53  | C++                | 5                 | 39           |
| 1879  | C++ User      | 48  | C++                | 5                 | 34           |
| 1157  | C++ Librarian | 37  | C#                 | 6                 | 20           |
| 7640  | C++ Librarian | 29  | C++                | 6                 | 16           |
| 4642  | C++ Librarian | 53  | C++                | 6                 | 33           |
| 8697  | C++ Librarian | 22  | JS / C++           | 4                 | 5            |
| 1867  | C++ Librarian | 29  | C++                | 5                 | 9            |



Figure 1: Subjects' performance relative to C/C++ experience

# 誤解原因のカテゴリ分け

---

- 被験者の回答，会話，インタビューからコードの  
不理解の原因をカテゴリ分け

# 誤解の原因の種類

- 知識不足

- 言語についての知識が不足することで発生する誤解

```
int V1 = 013;
```

“I assume the int is going to be thirteen, because it doesn't have to represent that zero in memory ”

- 思い違い

- 誤った知識を持ったままコードの内容を解釈しようとすることで発生する誤解

```
if (++V1 || ++V2) ...
```

“This is pre-increment and uh, what I remember from what I studied is that, you know, before the statement is executed, this will be incremented”

# 誤解の原因の種類

- 言語の違い

- 普段使っていない言語を使う際に構文の違いによって発生する誤解

```
int V1 = 013;
```

“We have like different number base, but for int. I know Python has something like it, but in C, I’m not sure. [In Python, the 0 in front] means nothing”

- 注意不足

- 開発者の注意力が不足していることで発生する誤解

```
printf("%d %dn", V1, V2)
```

“V2 is going to equal 1 and V1 is going to equal 2 after that expression. And so it’s going to print out one, two”

# 想定外の誤解原因

- 先行研究[11]の想定とは異なる誤解が発生することが判明
- 右記のコードで以下のような想定外の誤解の仕方をする被験者がいた
  - Cでは0以外の値が**true**として評価されることを知らない
  - 変数でないと論理演算できないと思い込んでいる
- **AoCの最小限度が不十分である可能性がある**

```
int V1 = 0;
if (0 && 1 || 2) {
    V1 = 6;
} else {
    V1 = 3;
}
printf("%d¥n", V1);
```

# 誤解しながらの正答

- コードの内容を誤解しているにも関わらず、コードの出力結果を正答することがあることが判明
- `&&`と`||`の優先度は同じと答えた被験者がいた
  - `if (0 && 1 || 2) {...}`の場合は正答できるが、`if (2 || 1 && 0) {...}`の場合は誤答してしまう
- 正答の中にも誤解によるものが含まれており、誤解件数を網羅できない可能性がある

# 判明した問題点と議論

- 結果から先行研究のいくつかの仮説がいつもその通りになるとは限らないことがわかった
- 今後の**AoC**研究のために以下の話題について議論する
  - 分析過程でわかった新しい**AoC**候補
  - **AoC**モデルの限界
  - 先行研究[11]の妥当性への脅威
  - 定量的作業における定性的方法の役割

# 新しいAoC候補（一部）

- 未初期化変数

- 言語によっては宣言時に初期値を与えない場合に不定な値が入っている可能性があり，変数の状態がわからない

- 剰余演算

- 数字が大きくなると頭で即座に剰余の演算が難しくなり，条件文などで使うときにわかりにくくなる

# AoCモデルの限界

- AoCの概念はプログラムの理解プロセスの測定を単純化するために導入された
  - 精度を得るためにアトミック性を追求した
- 分析の中でAoCの概念の見直しをする必要がある以下の事例があることがわかった
  - 被験者ごとに解釈が大きく異なる場合がある
  - 最小化の仕方を見直す必要がある

# 妥当性への脅威

- printfによる混乱
  - printfで回答する必要があるが、printfの使い方を完全に覚えておくのは困難
- 変数名
  - 変数名によって理解度を左右する可能性があるため、被験者に見せる変数はV1, V2など単調なものにした
  - しかし、単調が故にある値がどの変数に格納されていたかわからなくなる事例があった

# 定量的作業における定性的方法の役割

---

- 先行研究[11]を始める前に本研究のような分析を行えば欠陥に気づくことができた
- プログラム理解実験を実施する際は、定量的な実験を行う前に定性的に検証することを勧める

# まとめ

---

- 先行研究[11]で定義したAoCをどのように誤解するかを被験者に対し思考発話法などを用いて実験
- 4種の誤解の原因を明らかにした
- 先行研究[11]のいくつかの問題点を洗い出した

# 所管

---

- アプローチの処理内容がわかりにくかった
- AoCを検出することでプログラマの誤解を防ぐことができるのではないかと思った
- 実際の開発でAoCが紛れ込んでいる事例はどれだけあるのか気になった