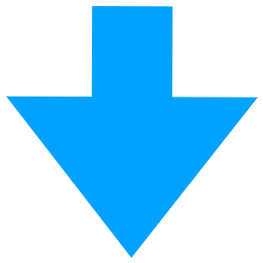
組み込みソフトウェアにおける
コードクローン出現に関する考察

京都工芸繊維大学　大学院
ソフトウェア工学研究室　若林 奎人

研究背景と目的

- コードクローンはソフトウェアの保守性を下げる.
- 組み込みソフトウェアではハードウェアの多様化に伴いコードクローンが発生している可能性がある.
- Arduinoはハードウェア，ソフトウェア共にオープンソースであるため，コードクローンが多く発生していると考えた.



CCFinderを用いてコードクローンに関する知見を得る[1]

[1]Toshihiro Kamiya, Shinji Kusumoto, and Kasturo Inoue, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code”, IEEE Transaction on Software Engineering, pp.1-17, 2002

研究設問

RQ1: Arduino環境においてどの程度の割合で
コードクローンが存在するか.

RQ2: ソースコードのどのような部分にコードクローンが
存在するか.

RQ3: 組み込みソフトウェア特有のコードクローンは
存在するか.

コードクローン

- ソースファイル中で類似したコード部分を表す.
- コードクローンはソフトウェアの保守性を下げる.

```
1 static inline void emac_enable_receive(Emac* p_emac, uint8_t uc_enable)
2   {
3     if (uc_enable) {
4       p_emac->EMAC_NCR |= EMAC_NCR_RE;
5     } else {
6       p_emac->EMAC_NCR &= ~EMAC_NCR_RE;
7     }
8   }
```

```
1 static inline void emac_enable_transmit(Emac* p_emac, uint8_t uc_enable)
2   {
3     if (uc_enable) {
4       p_emac->EMAC_NCR |= EMAC_NCR_TE;
5     } else {
6       p_emac->EMAC_NCR &= ~EMAC_NCR_TE;
7     }
8   }
```

実験準備

CCFinder

- 大規模なソフトウェアのソースコードにも適用できる.
- あらゆるプログラミング言語に対応している.

メトリクス名	略称	内容
Number of File	FILE	入力するファイル数
Length	LEN	コードクローンの文字数
Line of Code	LOC	入力されたソースファイル群の合計行数
Colen Line of Code	CLOC	コードクローンの合計行数
Population of Clone	POP	コードクローンの出現回数
CVR % LOC	CVR % LOC	合計行数のうちのコードクローンの行数の割合

実験準備

Arduino

- ソフトウェアがオープンソースであり、ハードウェアを安価で入手できる組み込みソフトウェア向けプラットフォーム.

- 派生ハードウェアが多数存在する.

実験準備

Google Big Query

- Google が提供するビッグデータ解析ツールである.データベースの操作には SQL文を使用できる.

GitHub

- Gitによりバージョン管理されたリポジトリをホスティングする.
- リポジトリのデータベースが存在し、Google Big Queryを用いて取得できる.

実験準備

データセット

- Google Big Queryを用いてGitHubからArduinoのソフトウェアを取得する.
- 2016年1月から5月の間でウォッチ数が10以上であるものを条件とし，170個のソフトウェアを入手した.

実験手順

<RQ1: Arduino環境においてどの程度の割合でコードクローンが存在するか>

- GitHubから入手した170個のArduinoソフトウェアに CCFinderを適用し，CVR % LOCを得る.

関連研究[2]の結果から，gcc，JDK，LinuxのCVR % LOCの値と比較する.

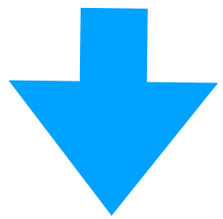
- コンパイル機能を持つESP8266のArduinoライブラリと，JavaのコンパイラであるApache AntのCVR % LOCの値を比較する.

[2]Toshihiro Kamiya, Shinji Kusumoto, and Kasturo Inoue, "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code”, IEEE Transaction on Software Engineering, pp.1-17, 2002

実験手順

<RQ2: ソースコードのどのような部分にコードクローンが存在するか>

- RQ1で得た170個のArduinoソフトウェアに含まれる
  CVR % LOCが大きいコードクローンを順に30個
  選出する.



ソースコードのどのような部分にコードクローンが発生しているかを調べる.

実験手順

<RQ3: 組み込みソフトウェア特有のコードクローンは存在するか>

- RQ2で得たコードクローンのソースファイルを観察することで，組み込みソフトウェア開発経験者の手で分類を行い，組み込みソフトウェア特有のコードクローンの存在を調べる.

実験結果

<RQ1: Arduino環境においてどの程度の割合でコードクローンが存在するか>

- Arduinoソフトウェアそれぞれの CVR % LOC は平均21.5%であった.
- 関連研究[3]の実験結果から，小規模なソフトウェアの CVR % LOCは平均0.38%である.

	FILE	LOC	CVR % LOC
Arduino Projects	25,882	6,595,226	62,3
ESP8266 Library	1,464	316,462	37,8
Apache Ant	1,272	272,359	26,6
gcc	221	460,000	8,7
JDK	1,877	570,000	29,0
Linux	6,497	4,365,124	22,7

[3]R. Al　Ekram, C. Kapser, R. Holt, and M. Godfrey, "Cloning by accident: an empirical study of source code cloning across software systems”, International Symposium on Empirical Software Engineering, pp.17-18, 2005

実験結果

<RQ2: ソースコードのどのような部分にコードクローンが存在するか>

クローンの種類	LEN(コードクローンの文字数)	POP(コードクローンの出現回数)
組み込み関数によるSIMD命令の羅列	2,626	46
レジスタ処理	1,511	20
コンパイラによる構文解析文	1,081	11
ツールで自動生成されたコード	3,184	15

実験結果

<RQ3: 組み込みソフトウェア特有のコードクローンは存在するか>

- 13個の組み込みソフトウェア特有のコードクローンが選出された.

	LEN	POP
コアの種類	4,037	10
ディスプレイの種類	2,205	13
ファームウェアの動作モードの種類	3,115	10

まとめ

RQ1: 170個のソフトウェア全体で62.3%，1つのソフトウェア中に平均で21.5%コードクローンが存在し，比較対象よりも多く発生していることを確認した.

RQ2: SIMD命令の羅列や，レジスタの処理，自動生成によるコードなどにコードクローンが多く検出されることを確認した.

RQ3: コアの種類，ディスプレイの種類，ファームウェアの動作モードの種類によって生まれるコードクローンを組み込みソフトウェア特有のコードクローンであると判定した.

課題: Arduino以外の組み込みソフトウェアにも同じ結果が得られるか，コードクローンの対処

コードクローン例


```
1 __attribute__( ( always_inline ) ) static __INLINE uint32_t __SADD8(uint32_t op1, uint32_t op2)
2   {
3     uint32_t result;
4
5     __ASM volatile ("sadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
6     return(result);
7   }
```

```
1 __attribute__( ( always_inline ) ) static __INLINE uint32_t __QADD8(uint32_t op1, uint32_t op2)
2   {
3     uint32_t result;
4
5     __ASM volatile ("qadd8 %0, %1, %2" : "=r" (result) : "r" (op1), "r" (op2) );
6     return(result);
7   }
```

SIMD命令文

コードクローン例


```
GALGAS_arxmlMetaClasslist GALGAS_arxmlMetaClasslist::add_operation (const GALGAS_arxmlMetaClasslist &
inOperand,C_Compiler COMMA_UNUSED_LOCATION_ARGS) const
{
  GALGAS_arxmlMetaClasslist result ;
  if (isValid () && inOperand.isValid ()) {
    result = *this ;
    result.appendList (inOperand) ;
  }
  return result ;
}
```

```
GALGAS_locationList GALGAS_locationList::add_operation (const GALGAS_locationList & inOperand,C_Compiler
COMMA_UNUSED_LOCATION_ARGS) const
{
  GALGAS_locationList result ;
  if (isValid () && inOperand.isValid ()) {
    result = *this ;
    result.appendList (inOperand) ;
  }
  return result ;
}
```

構文解析文

コードクローン例


```
virtual property BridgeRT::E_ACCESS_TYPE Access
{
    BridgeRT::E_ACCESS_TYPE get()
    {
        return this->access;
    }

    void set(BridgeRT::E_ACCESS_TYPE accessType)
    {
        this->access = accessType;
    }
}
```

```
virtual property BridgeRT::SignalBehavior COVBehavior
{
    BridgeRT::SignalBehavior get() { return this->covBehavior; }

    void set(BridgeRT::SignalBehavior behavior)
    {
        this->covBehavior = behavior;
    }
}
```

自動生成されたコード

コードクローン例


```
1 String::String(const String & value)
2   {
3     init();
4     *this = value;
5   }
```

```
1 String::String(const __FlashStringHelper *pstr)
2   {
3     init();
4     *this = pstr;
5   }
```

コアのライブラリ

コードクローン例


```
1 void Adafruit_GFX::drawCircle(int16_t x0, int16_t y0, int16_t r,
2 uint16_t color) {
3   int16_t f = 1 - r;
4   int16_t ddF_x = 1;
5   int16_t ddF_y = -2 * r;
6   int16_t x = 0;
7   int16_t y = r;
8
9   drawPixel(x0, y0+r, color);
10  drawPixel(x0, y0-r, color);
11  drawPixel(x0+r, y0, color);
12  drawPixel(x0-r, y0, color);
```


```
1 void Adafruit_GFX::drawCircle(int16_t x0, int16_t y0, int16_t r,
2 uint16_t color) {
3   int16_t f = 1 - r;
4   int16_t ddF_x = 1;
5   int16_t ddF_y = -2 * r;
6   int16_t x = 0;
7   int16_t y = r;
8
9   drawPixel(x0, y0+r, color);
10  drawPixel(x0, y0-r, color);
11  drawPixel(x0+r, y0, color);
12  drawPixel(x0-r, y0, color);
```


ディスプレイのライブラリ

コードクローン例


```
1 unsigned char hex_nibble(unsigned char data) {
2     data = toupper(data);
3     return (data >= 'A' ? data - 'A' + 10 : data - '0') & 0xf;
4   }
5
6   unsigned char parse_hex() {
7     unsigned char data;
8     while (Serial.available() < 2)
9       ;
10    data = hex_nibble(Serial.read()) << 4;
11    data |= hex_nibble(Serial.read());
12
13    return data;
14  }
```


```
1 unsigned char hex_nibble(unsigned char data) {
2     data = toupper(data);
3     return (data >= 'A' ? data - 'A' + 10 : data - '0') & 0xf;
4   }
5
6   unsigned char parse_hex() {
7     unsigned char data;
8     while (Serial.available() < 2)
9       ;
10    data = hex_nibble(Serial.read()) << 4;
11    data |= hex_nibble(Serial.read());
12
13    return data;
14  }
```

コードクローン例


```
 1 oid parseStatusText_v3_2(int32_t severity, String text)
 2 {
 3   uint16_t textId = 0;
 4
 5   // Texts with textId = 0 will be ignored
 6
 7   // motors.pde
 8        if(text == "ARMING MOTORS")                                  textId = 1;
 9   else if(text == "PreArm: RC not calibrated")                      textId = 2;
10   else if(text.startsWith("PreArm: Baro not healthy"))              textId = 3;
```

```
 1 oid parseStatusText_v3_3(int32_t severity, String text)
 2 {
 3   uint16_t textId = 0;
 4
 5   // Texts with textId = 0 will be ignored
 6
 7   if(text == "")                                                   textId = 0;
 8
 9   //APMrover2/GCS_Mavlink.cpp
10   else if(text == "Initialising APM...")                           textId = 1;
11   else if(text == "Unsupported preflight calibration")             textId = 2;
12   else if(text == "command received:")                             textId = 3;
```

通信プロトコルのバージョン

LEN: 905, POP: 17