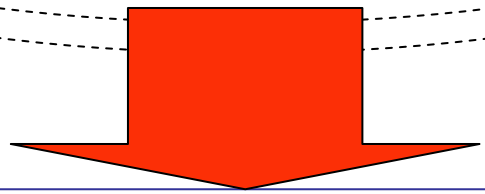
JaSST 2007

複数言語でのシステム開発におけるソース解析の効用

日本アイ・ビー・エム㈱
蔭山泰之

プログラム読解の必要性

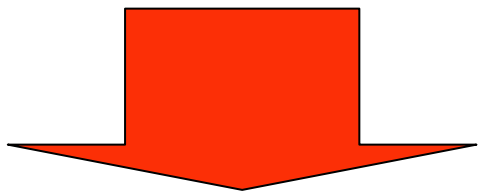
- システムの複雑化
- システムの大規模化



問題判別のため，仕様変更，
機能拡張などのためにソース
コードを読む必要性は
増してきている

リファクタリング

プログラムは一度書いて，テストして動いてしまえばそれで終わりというのではなく，繰り返し何度も手が加えられるものである



ソースコードは，いつでも手を加えることができるように，コンパクトにまとめられていなければならないし，いつでも読みやすいかたちに整えられていなければならない

ソース解析ツール

ソースコードを読み解くための解析ツール

→最新のJavaでは充実

→過去のプログラミング言語では？？？

複数言語によるシステム開発

オンライン系：Java

バッチ系：C

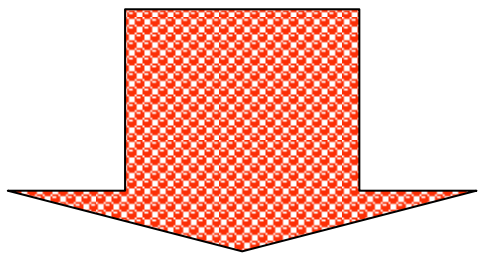
メインフレームI/F：PL/I

データ移行：PL/SQL

ソースコード解析ツール

独自開発のCプログラム解析ツール

ソースコードのLOCやNCSS，関数のサイズなどを計算し，関数呼び出しツリーを出力するための簡単な機能のみをもつツール



開発プロジェクトに合わせて機能拡張

複数言語（C,Java,PL/I,PL/SQL)対応

解析ツールの出力

- ソースコードの計測情報

C

	Byte	LOC	Byte/loc	No comment	Comment	Comment rate	Func num	Avg size	Min size	Max size
leader.h	5716	131	43.6	76	55	42.0%	0	0.0	0	0
Main.c	3183	91	35.0	46	45	49.5%	1	55.0	55	55
DBAccessDA.pc	9308	257	36.2	141	116	45.1%	4	40.8	28	57
FileAccessDA.c	6208	185	33.6	94	91	49.2%	3	35.3	25	48
Process.c	21076	633	33.3	394	239	37.8%	7	63.6	43	77
	45491	1297	35.1	751	546	42.1%	15	51.3	25	77

Java

	Byte	LOC	Byte/loc	No comment	Comment	Comment rate	Method Nm	Avg size	Min size	Max size
gic.java	11367	380	29.9	221	159	41.8%	6	43.3	26	98
troller.java	13076	419	31.2	262	157	37.5%	7	41.0	5	84
taDelivery.java	13087	416	31.5	210	206	49.5%	8	25.3	11	40
	37530	1215	30.9	693	522	43.0%	21	35.7	5	98

PL/I

	Byte	LOC	Byte/loc	No comment	Comment	Comment rate	Proc num	Avg size	Min size	Max size
060.PLI	48973	966	50.7	753	213	22.0%	10	126.7	6	953
	48973	966	50.7	753	213	22.0%	10	126.7	6	953

PL/SQL

	Byte	LOC	Byte/loc	No comment	Comment	Comment rate	Proc num	Avg size	Min size	Max size
6.sql	51240	1294	39.6	1058	236	18.2%	8	132.9	72	237
UTIL.sql	16250	527	30.8	246	281	53.3%	15	12.3	5	33
SCN.sql	34539	1080	32.0	804	276	25.6%	8	107.1	26	244
	102029	2901	35.2	2108	793	27.3%	31	67.9	5	244

• ヒストグラム

【HISTOGRAM OF FUNCTION SIZE】

Func size range	Num	Percent	
0- 9 lines	23	11.0%	***********************
10- 19 lines	39	29.7%	***************************************
20- 29 lines	41	49.3%	***
30- 39 lines	25	61.2%	*************************
40- 49 lines	17	69.4%	*****************
50- 59 lines	18	78.0%	******************
60- 69 lines	7	81.3%	*******
70- 79 lines	12	87.1%	************
80- 89 lines	5	89.5%	*****
90- 99 lines	5	91.9%	*****
100-109 lines	2	92.8%	**
110-119 lines	1	93.3%	*
120-129 lines	4	95.2%	****
130-139 lines	3	96.7%	***
140-149 lines	1	97.1%	*
150-159 lines	1	97.6%	*
160-169 lines	1	98.1%	*
170-179 lines	0	98.1%	
180-189 lines	3	99.5%	***
190-199 lines	0	99.5%	
200-209 lines	0	99.5%	
210-219 lines	0	99.5%	
220-229 lines	1	100.0%	*

たとえば関数サイズ

• クロスリファレンス

【FUNCTION TREE】

```
LEVEL GAUGE
───── 0───── 1───── 2───── 3───── 4───── 5───── 6───── 7───── 8───── 9─────10─────

■─ main( ) global [in <mtomain.c> at 170 |size:59| called=0 depth=0]
      ├── print response time( ) global [in <mtodbug.c> at 964 |size:
      │     ├── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      │     ├── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      │     └── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      ├── messageprint( ) global [in <mtoerrm.c> at 134 |size:59| cal
      │     ├── output notify message( ) static [in <mtoerrm.c> at 21
      │     └── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      ├── messageprint( ) global [in <mtoerrm.c> at 134 |size:59| cal
      │     ├── output notify message( ) static [in <mtoerrm.c> at 21
      │     └── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      ├── messageprint( ) global [in <mtoerrm.c> at 134 |size:59| cal
      │     ├── output notify message( ) static [in <mtoerrm.c> at 21
      │     └── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
      ├── messageprint( ) global [in <mtoerrm.c> at 134 |size:59| cal
      │     ├── output notify message( ) static [in <mtoerrm.c> at 21
      │     └── dbg print( ) global [in <mtodbug.c> at 67 |size:14| c
```

呼びだしツリー

実行制御語からテストケース生成へ

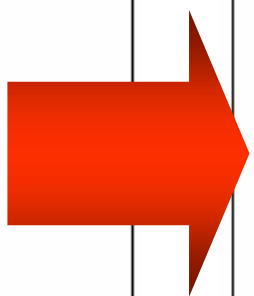
ソースコード

```
static D001_RC func( const char *key, const int x )
{
    int i;

    if ( 0 == strcmp( key, KEY_STRING ) ) {①

        if ( x > 0 ) {②

            for ( i = 0 ; i < MAX ; i++ ) {③④
                :
            }
        } else {⑤
            :
            :
        }
    } else if ( key[0] == NULL_STRING ) {⑥⑦
            :
    }
    return( D001_SUCCESS );
}
```



生成UTケース 分岐網羅ケース

No.	ファイル名	関数名	分岐	条件文
1	xxx.c	func	if	(0 == strcmp(key, KEY_STRING))
2	xxx.c	func	if	(x > 0)
3	xxx.c	func	for	(i = 0 ; i < MAX ; i++)
4	xxx.c	func	added for-end	
5	xxx.c	func	else	
6	xxx.c	func	else if	(key[0] == NULL_STRING)
7	xxx.c	func	added else	

複数言語での統一的な単体テスト項目

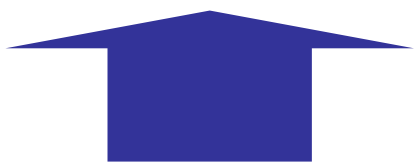
コードインスペクション

設計に慣れていないと・・・

→処理手順をそのまま書き下ろす．コピー・ペースト

→関数やメソッドなどが一般的に長くなりがち

部品としての関数やクラス，メソッドを設計してそれを組合せるようにしてプログラムを構築するようアドバイス



この目的のために解析ツールを活用

関数分割によるテスト工数の削減

改善前

【HISTOGRAM OF FUNCTION SIZE】

Func size range	Num	Percent	
0- 9 lines	0	0.0%	
10- 19 lines	0	0.0%	
20- 29 lines	3	10.7%	***
30- 39 lines	0	10.7%	
40- 49 lines	6	32.1%	******
50- 59 lines	6	53.6%	******
60- 69 lines	1	57.1%	*
70- 79 lines	2	64.3%	**
80- 89 lines	1	67.9%	*
90- 99 lines	0	67.9%	
100-109 lines	1	71.4%	*
110-119 lines	0	71.4%	
120-129 lines	4	85.7%	****
130-139 lines	0	85.7%	
140-149 lines	1	89.3%	*
150-159 lines	1	92.9%	*
160-169 lines	0	92.9%	
170-179 lines	0	92.9%	
180-189 lines	0	92.9%	
190-199 lines	0	92.9%	
200-209 lines	0	92.9%	
210-219 lines	0	92.9%	
220-229 lines	0	92.9%	
230-239 lines	0	92.9%	
240-249 lines	0	92.9%	
250-259 lines	0	92.9%	
260-269 lines	0	92.9%	
270-279 lines	0	92.9%	
280-289 lines	0	92.9%	
290-299 lines	0	92.9%	
300-309 lines	0	92.9%	
310-319 lines	0	92.9%	
320-329 lines	0	92.9%	
330-339 lines	0	92.9%	
340-349 lines	0	92.9%	
350-359 lines	0	92.9%	
360-369 lines	0	92.9%	
370-379 lines	0	92.9%	
380-389 lines	0	92.9%	
390-399 lines	0	92.9%	
400-409 lines	0	92.9%	
410-419 lines	0	92.9%	
420-429 lines	0	92.9%	
430-439 lines	0	92.9%	
440-449 lines	0	92.9%	
450-459 lines	0	92.9%	
460-469 lines	0	92.9%	
470-479 lines	0	92.9%	
480-489 lines	0	92.9%	
490-499 lines	0	92.9%	
500-509 lines	0	92.9%	
510-519 lines	1	96.4%	*
520-529 lines	0	96.4%	
530-539 lines	0	96.4%	
540-549 lines	0	96.4%	
550-559 lines	0	96.4%	
560-569 lines	0	96.4%	
570-579 lines	0	96.4%	
580-589 lines	0	96.4%	
590-599 lines	0	96.4%	
600-609 lines	0	96.4%	

テストケース削減

改善後

【HISTOGRAM OF FUNCTION SIZE】

Func size range	Num	Percent	
0- 9 lines	0	0.0%	
10- 19 lines	3	3.8%	***
20- 29 lines	35	48.7%	**********
30- 39 lines	11	62.8%	**********
40- 49 lines	13	79.5%	**********
50- 59 lines	6	87.2%	******
60- 69 lines	1	88.5%	*
70- 79 lines	4	93.6%	****
80- 89 lines	0	93.6%	
90- 99 lines	0	93.6%	
100-109 lines	0	93.6%	
110-119 lines	1	94.9%	*
120-129 lines	0	94.9%	
130-139 lines	0	94.9%	
140-149 lines	0	94.9%	
150-159 lines	0	94.9%	
160-169 lines	0	94.9%	
170-179 lines	0	94.9%	
180-189 lines	1	96.2%	*
190-199 lines	2	98.7%	**
200-209 lines	0	98.7%	
210-219 lines	0	98.7%	
220-229 lines	0	98.7%	
230-239 lines	0	98.7%	
240-249 lines	1	100.0%	*

改善前

[TOTAL BRANCH COVERAGE CASES: 337]

[NO COMMENT SOURCE STEPS : 3168]

[CASES PER KILO NCSS : 106.4]

改善後

[TOTAL BRANCH COVERAGE CASES: 274]

[NO COMMENT SOURCE STEPS : 3656]

[CASES PER KILO NCSS : 74.9]

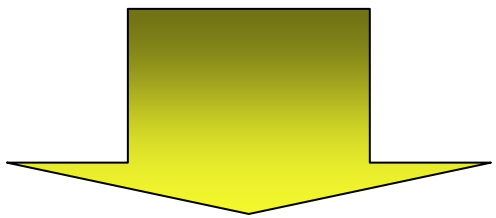
複数言語での思想の共有

- Javaの習慣：

  クラス単位、メソッド単位にコンパクトにまとめる

- 過去のプログラミングでは・・・？

  同じ考え方を共有するための標準が必要



解析ツールによる統一的な計測情報の出力

比較検討を可能にする

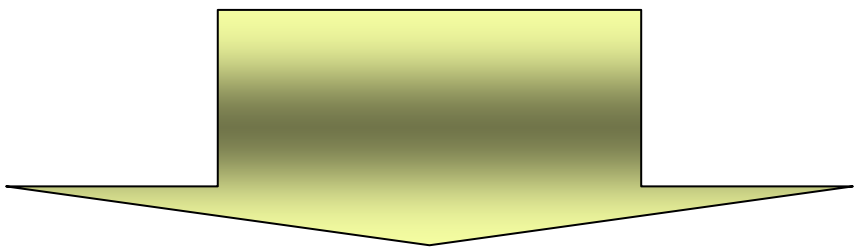
テストの統一的管理

- 予想に基づくテスト管理

NCSS1キロステップあたり単体テスト平均ケース数	
C	107.2
Java	122.8
PL/I	92.3
PL/SQL	130.4

比較検討

予想外



重点管理対象を見出す

ソースコードの改善へ向けて

- **その第一歩**

プログラマに自分が書いたプログラムの姿を見せる
保守性、再利用性、可読性から見た姿を評価する
→ソースコードの計測情報からこれらの点を評価する

- **その第二歩**

よいプログラムの条件は言語に依存しない
プログラミングにとって本質的な部分を評価する
→複数言語のあいだで本質的な部分を比較評価する