

제 10회 공개 SW 개발자 대회

41팀 SWQ&R - 스펙트럼 기반 통합 테스트 플랫폼



2019/2/21

jeonghodot@skku.edu

김정호

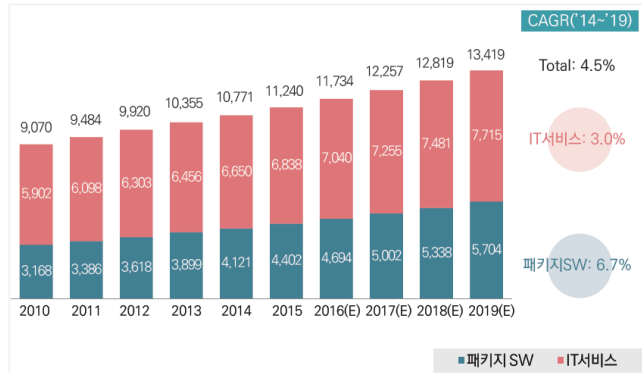


OSS World Challenge 2016
제 10회 공개SW개발자대회

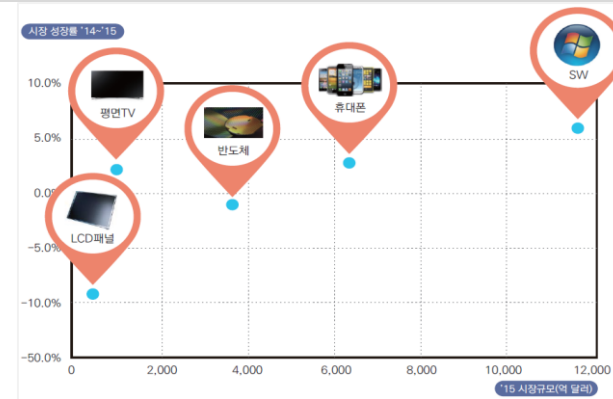


- 1. 개발 배경
- 2. REDLine
- 3. 특징점 및 활용 방안
- 4. 기대 효과
- 5. 유효성 확인
- 6. 시연 영상

1. 개발 배경



- 세계 SW 시장 규모 (단위: 억 달러) -



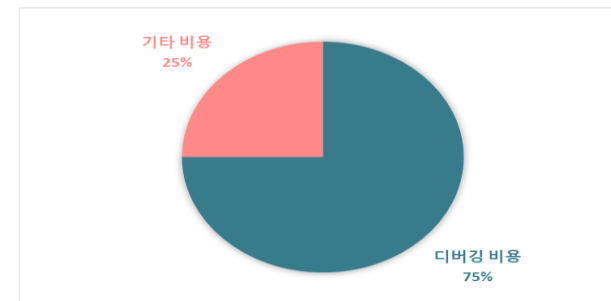
- 산업별 시장 성장률 (단위: 억 달러) -

• 소프트웨어 대한 의존도

- 4차 산업혁명 도래
- 디바이스의 기능 및 정보의 다양화

• 고신뢰 소프트웨어

- 고신뢰 소프트웨어를 생산하기 위한 각별한 노력이 요구 되고 있는 추세
- 고품질의 소프트웨어는 개발 기술(요구 공학, 설계, 구현 등)에 의한 영향력 이상으로, 개발된 결과물에 대한 디버깅 기술이나 성능이 특히 중요
 - ✓ 전체 소프트웨어 개발 비용의 50-75%를 디버깅 작업이 차지



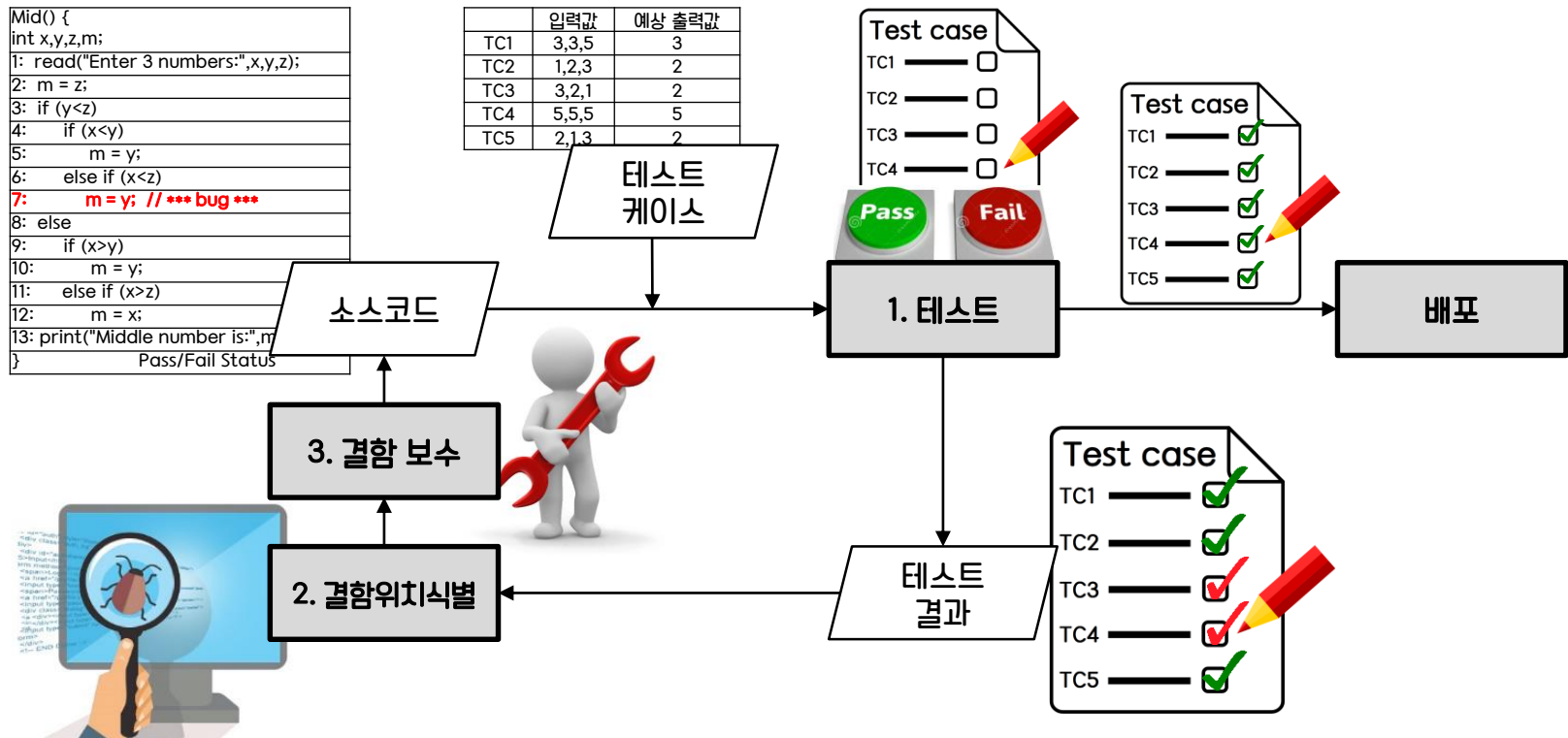
- 전체 소프트웨어 개발 비용 -

1. 개발 배경

▪ 디버깅

• 소프트웨어를 테스트로 오류가 있는지를 검사하고 이를 기반으로 결함의 위치를 찾아내 수정하는 작업

- 1. 테스트 - 테스트케이스를 실행시켜 오류가 있는지를 시험하는 것
- 2. 결함위치식별 - 오류 발생의 원인인 결함의 위치를 찾는 것
- 3. 결함보수 - 결함을 수정할 대안을 찾고 평가를 통해 올바른 소스코드로 수정하는 것

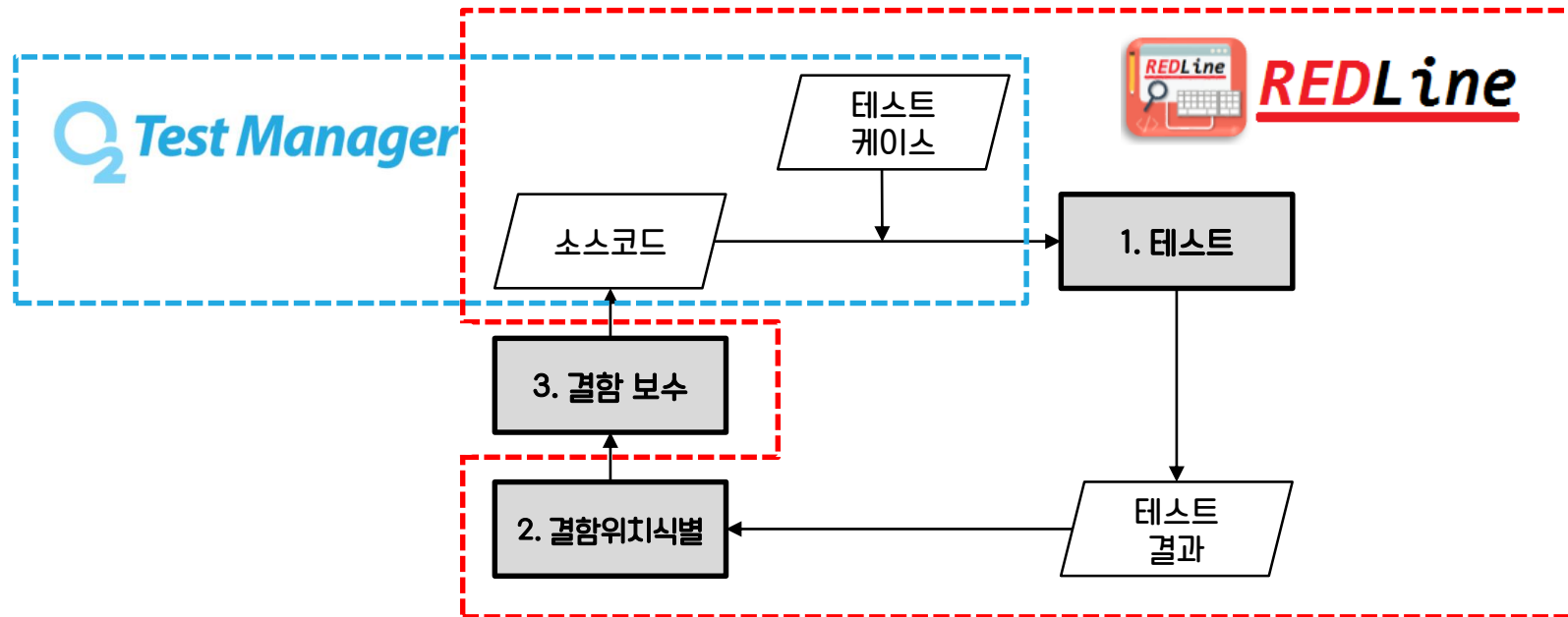


1. 개발 배경

▪ 디버깅

• 소프트웨어를 테스트로 오류가 있는지를 검사하고 이를 기반으로 결함의 위치를 찾아내 수정하는 작업

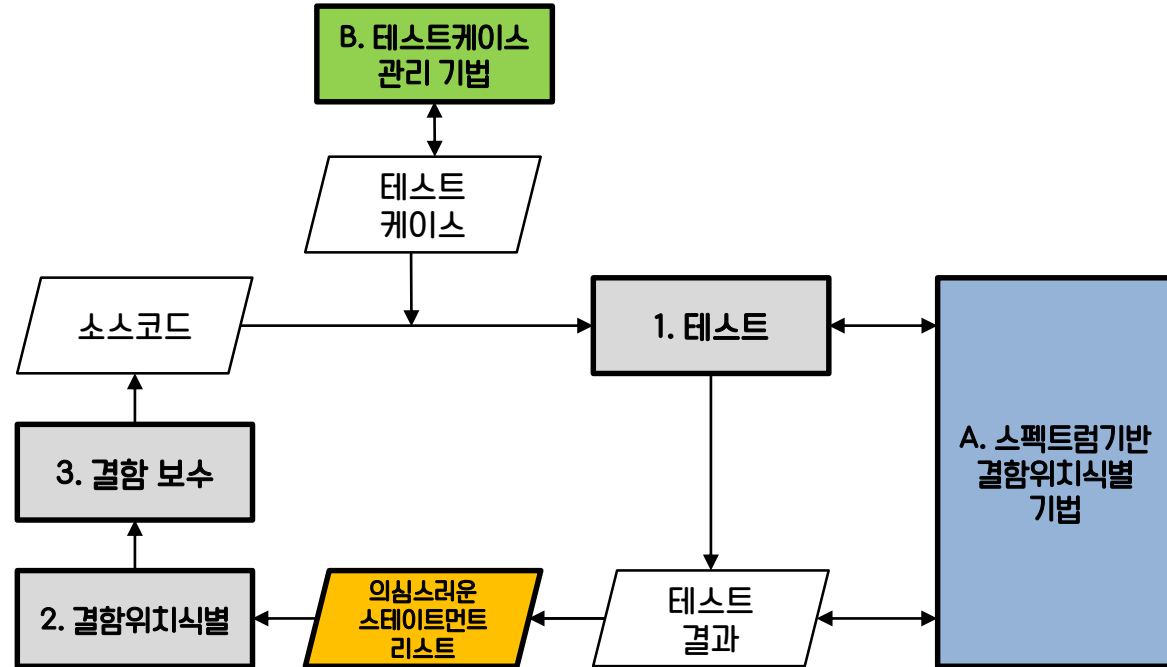
- 1. 테스트 - 테스트케이스를 실행시켜 오류가 있는지를 시험하는 것
- 2. 결함위치식별 - 오류 발생의 원인인 결함의 위치를 찾는 것
- 3. 결함보수 - 결함을 수정할 대안을 찾고 평가를 통해 올바른 소스코드로 수정하는 것



2. REDLine

▪ 아키텍처

- A. 스펙트럼 기반 결함 위치 식별 기법
- B. 테스트케이스 관리 기법



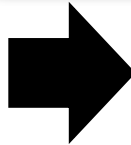
의심스러운 스테이트먼트 리스트

- 스테이트먼트의 의심스러운 정도에 따라 컬러링

```

[C:\SESS\B0\Programs\printtokens2\versions.alt\versions.orig\w9\print_tokens2.c] - UltraEdit
File Edit Search Insert Project View Format Column Macro Scripting Advanced Window Help [An update is available.]
[Icons]
+ print_tokens2.c x
203 int print_token(tok)
204 {
205     int type;
206     type=token_type(tok);
207     if(type==error)
208     {
209         fprintf(stdout, "error, \"%s\".\n", tok);
210     }
211     if(type==keyword)
212     {
213         fprintf(stdout, "keyword, \"%s\".\n", tok);
214     }
215     if(type==spec_symbol) print_spec_symbol(tok);
216     if(type==identifier)
217     {
218         fprintf(stdout, "identifier, \"%s\".\n", tok);
219     }
220     if(type==num_constant)
221     {
222         fprintf(stdout, "numeric, %s.\n", tok);
223     }
224     if(type==str_constant)
225     {
226         fprintf(stdout, "string, %s.\n", tok);
227     }
228     if(type==char_constant)
229     {
230         tok=tok+1;
231         fprintf(stdout, "character, \"%s\".\n", tok);
232     }
233     if(type==end)
234     {
235         fprintf(stdout, "eof.\n");
236     }
237 }
238
239 int is_eof_token(tok)
240 {
241     token tok;
242     if (*tok==EOF)
243         return(TRUE);
244     else
245         return(FALSE);
246 }
247
248 static int is_comment(ident)
249 {
250     token ident;
251     if ((*ident) == 59) /* the char is 59 */
252         return(TRUE);
253     else
254         return(FALSE);
255 }
256
257 static int is_keyword(str)
258 {
259     token str;
260     if (!strcmp(str, "and") || !strcmp(str, "or") || !strcmp(str, "if") ||
261         !strcmp(str, "xor") || !strcmp(str, "lambda") || !strcmp(str, ">"))
262         return(TRUE);
263     else
264         return(FALSE);
265 }
266
267 static int is_char_constant(str)
268 {
269     token str;
270     if ((*str) == '#' && isalpha(*(str+1)))
271         return(TRUE);
272     else
273         return(FALSE);
274 }
275
276 static int is_num_constant(str)
277 {
278     token str;
279     int i=1;
280     if (isdigit(*str))
281     {
282         while (*str+i != '\0') /* until meet token end sign */
283             i++;
284     }
285 }

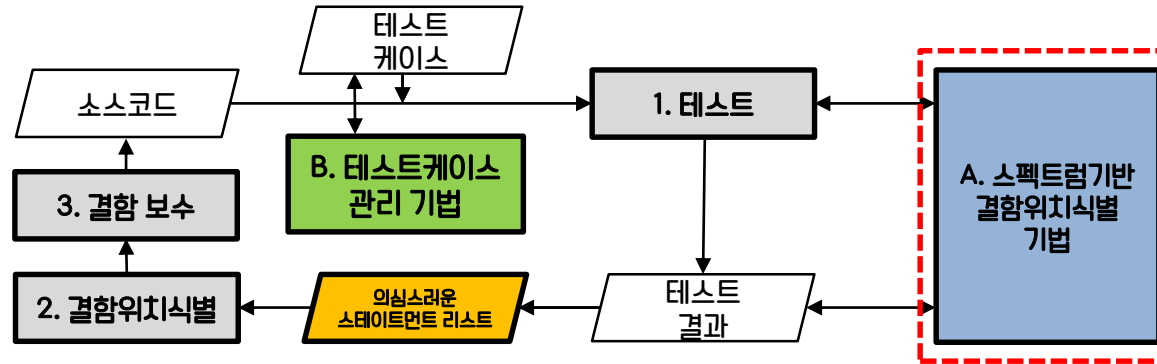
```



Line	Source Code	TC0001	TC0002	TC0003	...	TC4071	Tarantula	Rank
1	(tok)	:	:	:	:	:	:	:
2								45
3								45
4		1	1	1	...	1	0.5	43
5		1	1	1	...	1	0.5	43
6	(tok);	1	1	1	...	1	0.5	43
7		1	1	1	...	0	0.60522	34
8								45
9	ut, "error, \"%s\".\n", tok);	1	1	1	...	1	0.5	43
10		1	1	0	...	1	0.74813	25
11)							45
12	ut, "keyword, \"%s\".\n", tok);	1	1	1	...	1	0.5	43
13		1	1	1	...	1	0.5	43
14	ymbol)print_spec_symbol(tok);	1	1	1	...	1	0.78956	22
15	ier)							45
16	ut, "identifier, \"%s\".\n", tok);	1	1	1	...	1	0.5	43
17		1	1	1	...	1	0.89587	10
18	stant)							45
19	ut, "numeric, %s.\n", tok);	1	1	1	...	1	0.5	43
20		1	1	1	...	0	0.9156	8
21								45
22	stant)							45
23	ut, "string, %s.\n", tok);	1	1	1	...	1	0.5	43
24		1	1	1	...	1	0.5	43
25		1	1	1	...	0	0.9156	8
26								45
27	341 return(FALSE);	1	1	1	...	1	0.5	43
28	342 }	1	1	1	...	1	0.5	43
29	343							45
30	349 static int is_num_constant(str)							45
31	350 token str;							45
32	351 {	1	1	1	...	1	0.5	43
33	352 int i=1;	1	1	1	...	1	0.5	43
34	353							45
35	354 if (isdigit(*str))	1	1	1	...	1	0.5	43
36	355 {							45
37	356 while (*(str+i) != '\0') /* until meet token	1	1	1	...	0	0.9891	2
38	357 {							45
39	358 if(isdigit(*(str+i)))	1	1	0	...	0	0.99008	1
40	359 i++;	1	1	0	...	0	0.99008	1
41	360 else							45
42	361 return(FALSE);							45
43	362 }							45
44	363 return(TRUE);							45
45	364 }							45
46	365 else							45
47	366 return(FALSE); /* other return							45
48	367 }							45
49	368							45
50	374 static int is_str_constant(str)	1	1	1	...	1	0.5	43
51	375 token str;	1	1	1	...	1	0.5	43
52	376 {							45
53	377 int i=1;							45
54	378							45
55	379 if (*str =='')	1	1	1	...	1	0.5	43
56	380 { while (*(str+i) != '\0') /*	1	0	1	...	0	0.91392	9
57	381 { if(*(str+i) =='')	1	0	1	...	0	0.91392	9
58	382 return(TRUE);	1	0	1	...	0	0.9156	8
59	383 else	1	0	1	...	0	0.9156	8
60	384 i++;							45
61	385 }							45
62	/* end MB							45
63	:	:	:	:	:	:	:	:
64								45
65								45
66								45
67								45

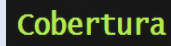
2. REDLine

■ A. 스펙트럼 기반 결함 위치 식별 기법



Mid() {	
int x,y,z,m;	
1: read("Enter 3 numbers:",x,y,z);	
2: m = z;	
3: if (y<z)	
4: if (x<y)	
5: m = y;	
6: else if (x<z)	
7: m = y; // *** bug ***	
8: else	
9: if (x>y)	
10: m = y;	
11: else if (x>z)	
12: m = x;	
13: print("Middle number is:",m);	
}	Pass/Fail Status

커버리지 추출 기술

[illegible]

스펙트럼기반 결함위치식별

Statistical analysis

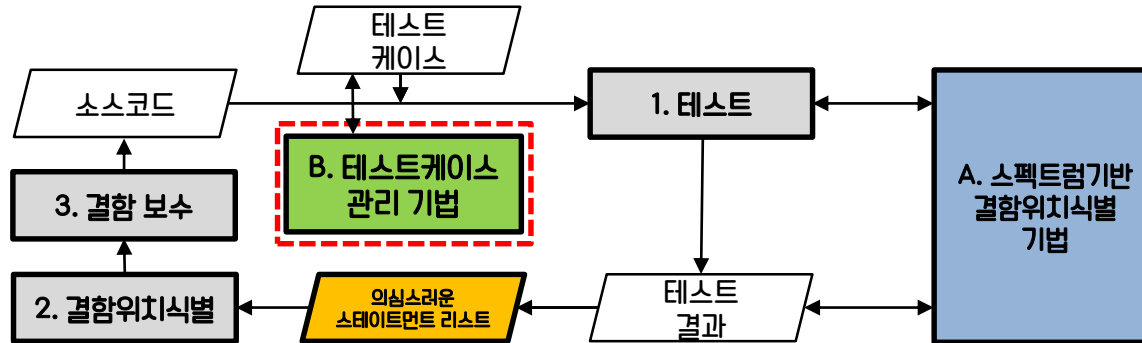
(Similarity & Distance coefficient)

1. TARANTULA, 2. AMPLE, 3. JACCARD, 4.
DICE, 5. CZEKANOWSKI, 6. 3WIACIARD, 7.
NELANDI, 8. SOKALANDSNEATH, 1, 9.
SOKALANDMICHER, 10.
SOKALANDSNEATH2, 11.
ROGERANDTANIMOTO, 12. FAITH, 13.
GOWERANDLEGENDRE, 14. INTERSECTION,
15. INNERPRODUCT, 16. RUSSELLANDRAO,
17. HAMMING, 18. EUCLID, 19.
SQUARED EUCLID, 20. CANBERRA, 21.
MANHATTAN, 22. MEAN MANHATTAN, 23.
CITYBLOCK, 24. MINKOWSK, 25. VARI, 26.
SIZEDIFFERENCE, 27. SHAPEDIFFERENCE,
28. PATTERNDIFFERENCE, 29.
LANCANDWILLIAMS, 30. BRAYANDCURTIS,
31. HELLINGER, 32. CHORD, 33. COSINE,
34. GILBERTANDWELLS, 35. OCHIAI1, 36.
FORBES1, 37. FOSSUM, 38. SORGENFREI, 39.
MOUNTFORD, 40. OTSUKA, 41.
MCCONNAUGHEY, 42. TARWID, 43.
KULCZYNSK2, 44. DRIVERANDKROEBER, 45.
JOHNSON, 46. DENNIS, 47. SIMPSON, 48.
BRAUNANDBANQUET, 49.
FAGERANDMCGOWAN, 50. FORBES2, 51.
SOKALANDSNEATH4, 52. GOWER, 53.
PEARSON1, 54. PEARSON2, 55. PEARSON3,
56. PEARSONANDHERON1, 57.
PEARSONANDHERON2, 58.
SOKALANDSNEATH3, 59.
SOKALANDSNEATH5, 60. COLE, 61. STILES,
62. OCHIAI2, 63. YULEQ, 64. YULEW, 65,
KULCZYNSKI1, 66. TANIMOTO, 67.
DISPERSON, 68. HAMANN, 69. MICHAEL,
70. GOODMANANDKRUSKAL, 71.
ANDERBERG, 72.
BARONI_URBANLANDUSER1, 73.
BARONI_URBANLANDUSER2, 74. PEIRCE, 75
EYRAUD, 76. Naish

Tarantula	Ranking	AMPLE	Ranking	Ochiai	Ranking	Jaccard	Ranking
0.5	4	0	9	0.41	5	0.17	4
0.5	4	0	9	0.41	5	0.17	4
0.5	4	0	9	0.41	5	0.17	4
0.63	3	0.4	3	0.5	3	0.25	3
0	8	0.2	6	0	8	0	8
0.71	2	0.6	2	0.58	2	0.34	2
0.83	1	0.8	1	0.71	1	0.5	1
0	8	0.4	3	0	8	0	8
0	8	0.4	3	0	8	0	8
0	8	0.2	6	0	8	0	8
0	8	0.2	6	0	8	0	8
0	8	0	9	0	8	0	8
0.5	4	0	9	0.48	4	0.17	4

2. REDLine

▪ B. 테스트케이스 관리 기법



	TC1	TC2	TC3	TC4	TC5	TC6
Mid() { int x,y,z,m;	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3
1: read("Enter 3 numbers:",x,y,z);	✓	✓	✓	✓	✓	✓
2: m = z;	✓	✓	✓	✓	✓	✓
3: if (y<z)	✓	✓	✓	✓	✓	✓
4: if (x<y)	✓	✓			✓	✓
5: m = y;		✓				
6: else if (x<z)	✓				✓	✓
7: m = y; // *** bug ***	✓					✓
8: else			✓	✓		
9: if (x>y)			✓	✓		
10: m = y;						
11: else if (x>z)			✓	✓		
12: m = x;						
13: print("Middle number is:",m);	✓	✓	✓	✓	✓	✓
Pass/Fail Status	Pass	Pass	Pass	Pass	Pass	Fail

테스트케이스 관리

ㄱ. 테스트케이스 중복 제거
동일한 트레이스와 테스트 결과를 갖는 테스트케이스를 제거

ㄴ. 테스트케이스 선택
수정된 스테이트먼트를 커버한 테스트케이스만 선택

ㄷ. 테스트케이스 재구성
테스트케이스를 Pass 와 Fail로 분류하여 경험적으로 재구성

ㄹ. 테스트케이스 우선순위화
개별 테스트케이스의 우선순위를 부여, 재정렬

	TC6	TC3	TC5	TC1	TC2
2,1,3	✓	✓	✓	✓	✓
3,2,1	✓	✓	✓	✓	✓
5,3,4	✓	✓	✓	✓	✓
3,3,5	✓	✓	✓	✓	✓
1,2,3	✓	✓	✓	✓	✓
Fail	✓	✓	✓	✓	✓

Tarantula	Ranking	AMPLE	Ranking	Ochiai	Ranking
0.5	4	0	9	0.41	5
0.5	4	0	9	0.41	5
0.5	4	0	9	0.41	5
0.63	3	0.4	3	0.5	3
0	8	0.2	6	0	8
0.71	2	0.6	2	0.58	2
0.83	1	0.8	1	0.71	1
0	8	0.4	3	0	8
0	8	0.4	3	0	8
0	8	0.2	6	0	8
0	8	0.2	6	0	8
0	8	0	9	0	8
0.5	4	0	9	0.48	4

Warning

3. 특징점 및 활용 방안

• 특징점

– 오픈소스로 발전 용이

- ✓ Github로 개발 (<https://github.com/jeonghodot/SWQR>)
 - » Apache License Version 2.0
- ✓ 문서화
 - » 요구사항명세서 (<http://naver.me/GCRKhJWG>)
 - » 설계명세서 (<http://naver.me/x3Lf9car>)



– 확장성

- ✓ 언어 지원 (Java, Python, Perl 등)
- ✓ 결함위치식별 기법 추가
- ✓ 테스트케이스 관리 기법 추가



• 활용 방안

– 오픈소스 소프트웨어의 신뢰도 평가 기준

- ✓ 기존 평가 기준(가독성, 규격, 커뮤니티, 라이선스 등) + 신뢰도

– 형상 관리 도구와 연동

- ✓ Github, Sourceforge



– 개발 도구와 연동

- ✓ Eclipse, Visual studio, Vi editor



4. 기대 효과

• 디버깅 비용 감소

- 소프트웨어의 스테이트먼트 단위로 결함위치식별이 가능하면 개발자가 결함의 원인 규명 및 해결 방안을 찾는 작업 부하 줄임 (비전문가도 쉽게 적용 가능)



• 고품질 고신뢰 소프트웨어의 생산성 극대화

- 통합 테스트 플랫폼의 활용으로 개발 중인 소프트웨어의 빠른 제품화가 가능하고 경쟁 상대의 서비스 및 제품보다 빠른 시장 선점 가능

• 소프트웨어산업의 질적 수준 향상

- 개발한 기술 및 지원 도구를 산업계에 제공함으로써 기업 내 소프트웨어 품질, 생산성 및 업무 효율성 향상에 기여



5. 유효성 확인

- 실험 대상

- 7개의 오픈소스 프로그램 (<http://sir.unl.edu/portal>)

- 평가 척도

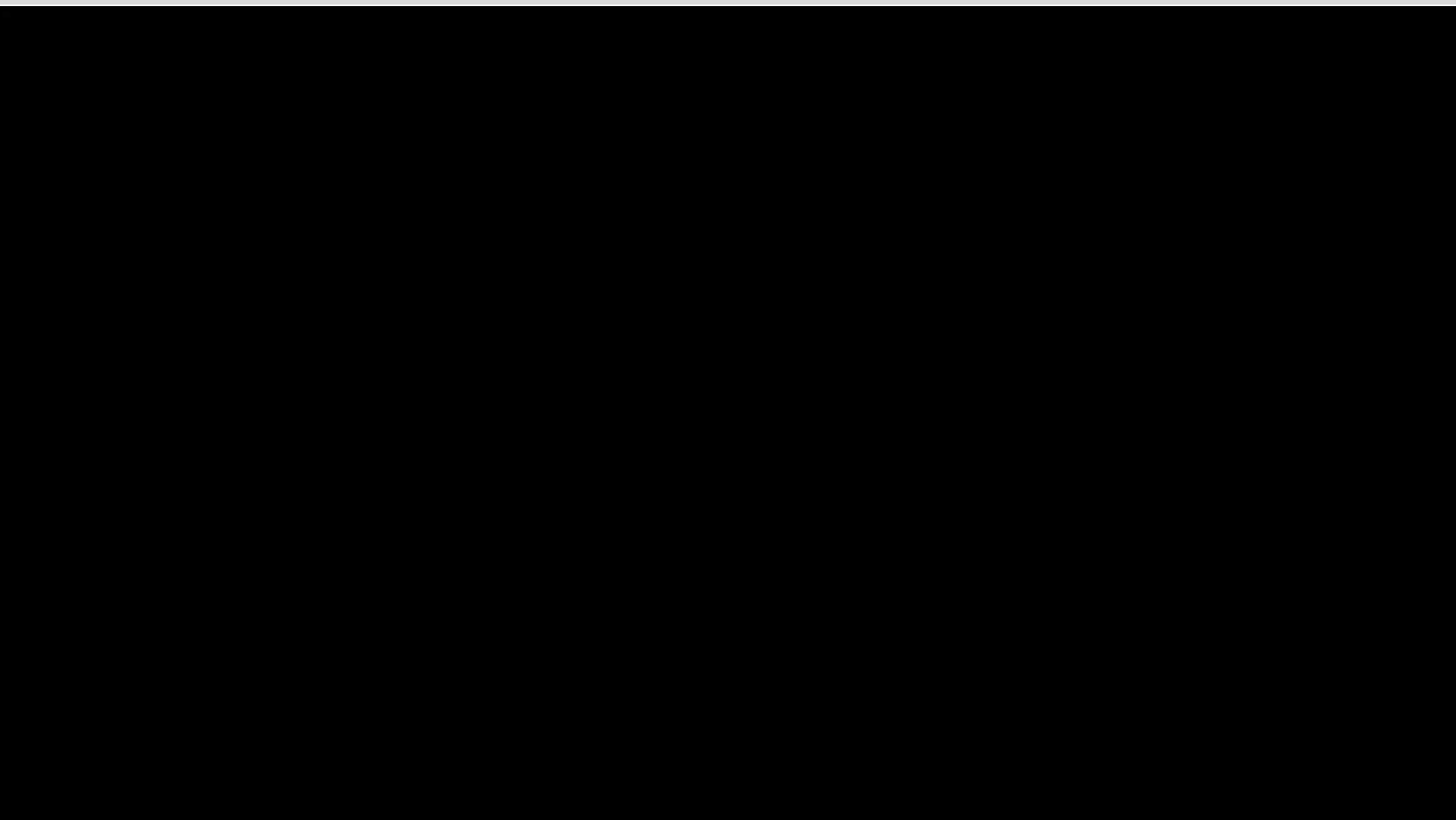
- $EXAM\ score = \frac{\text{실제 결함 위치에서의 순위}}{\text{전체 소스코드의 라인 수}} \times 100$

- 실험 결과

- 전체 프로그램의 평균 2.8%만 검사하면 실제 결함위치식별 가능 (Tarantula 기준)

Program	Version	LoC	Test case	Description	Average of EXAM score		
					Tarantula	AMPLE	Jaccard
printtokens	7	565	4140	lexical analyzer	1.5244	0.4962	1.0635
printtokens2	10	510	4071	lexical analyzer	3.6873	4.4510	4.4503
replace	32	563	5542	pattern recognition	2.5535	2.7669	2.3250
schedule	9	412	2650	priority scheduler	0.8330	2.7970	0.8330
schedule2	10	307	2680	priority scheduler	4.8648	6.8478	4.6843
tcas	41	173	1578	altitude separation	3.0875	3.7913	3.0444
totinfo	23	406	1054	information measure	3.0167	2.9894	2.9211
TOTAL	132	2936	21715		2.7953	3.4485	2.7602

6. 시연 영상



Q&A

Thanks