SYRCoSE-2016
Krasnovidovo, Moscow Region
30 May 2016

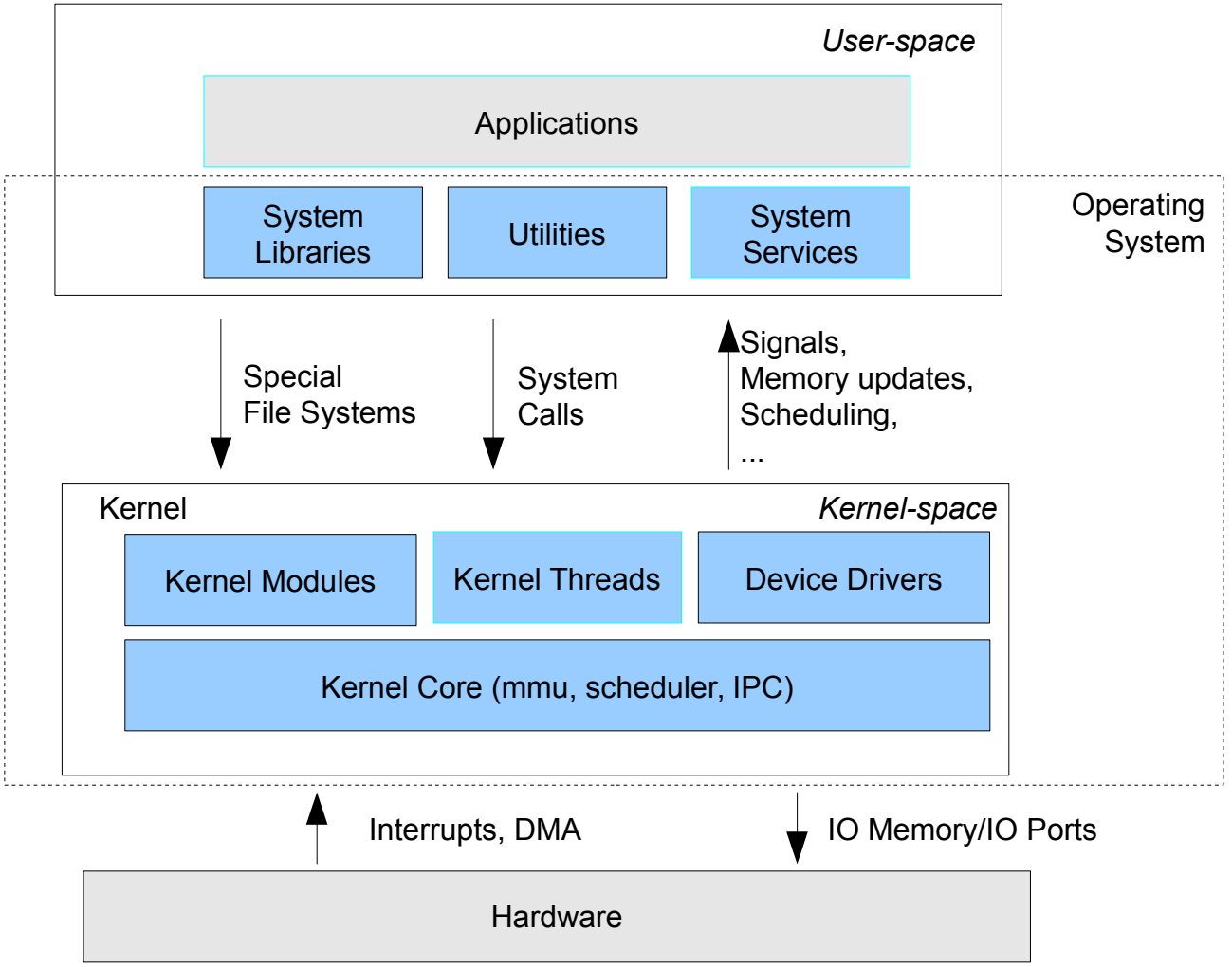
LinuxTesting.org

Verification of Operating Systems

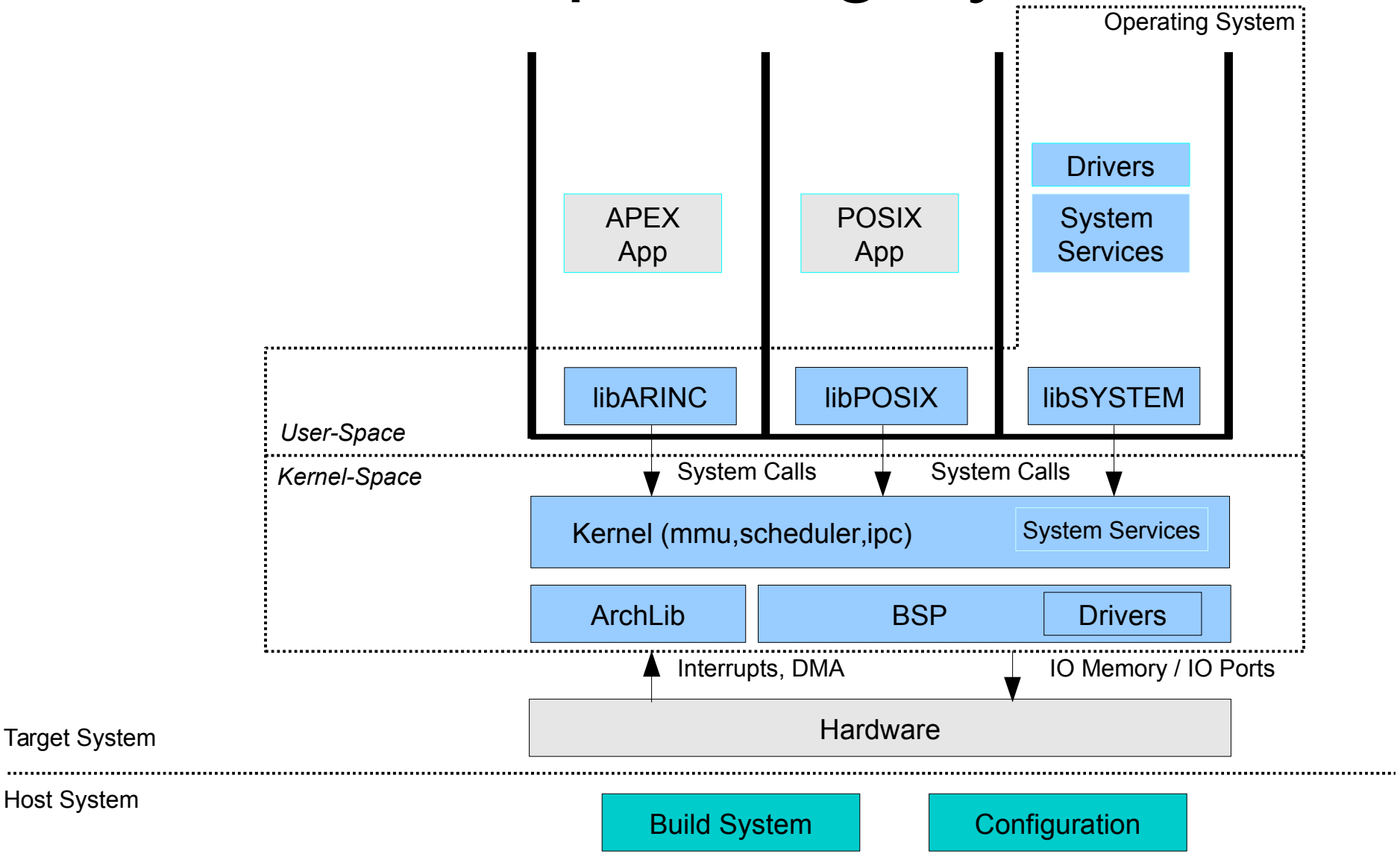
Alexey Khoroshilov
khoroshilov@ispras.ru

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Operating Systems



Embedded Operating Systems



Static Verification **Runtime Verification**

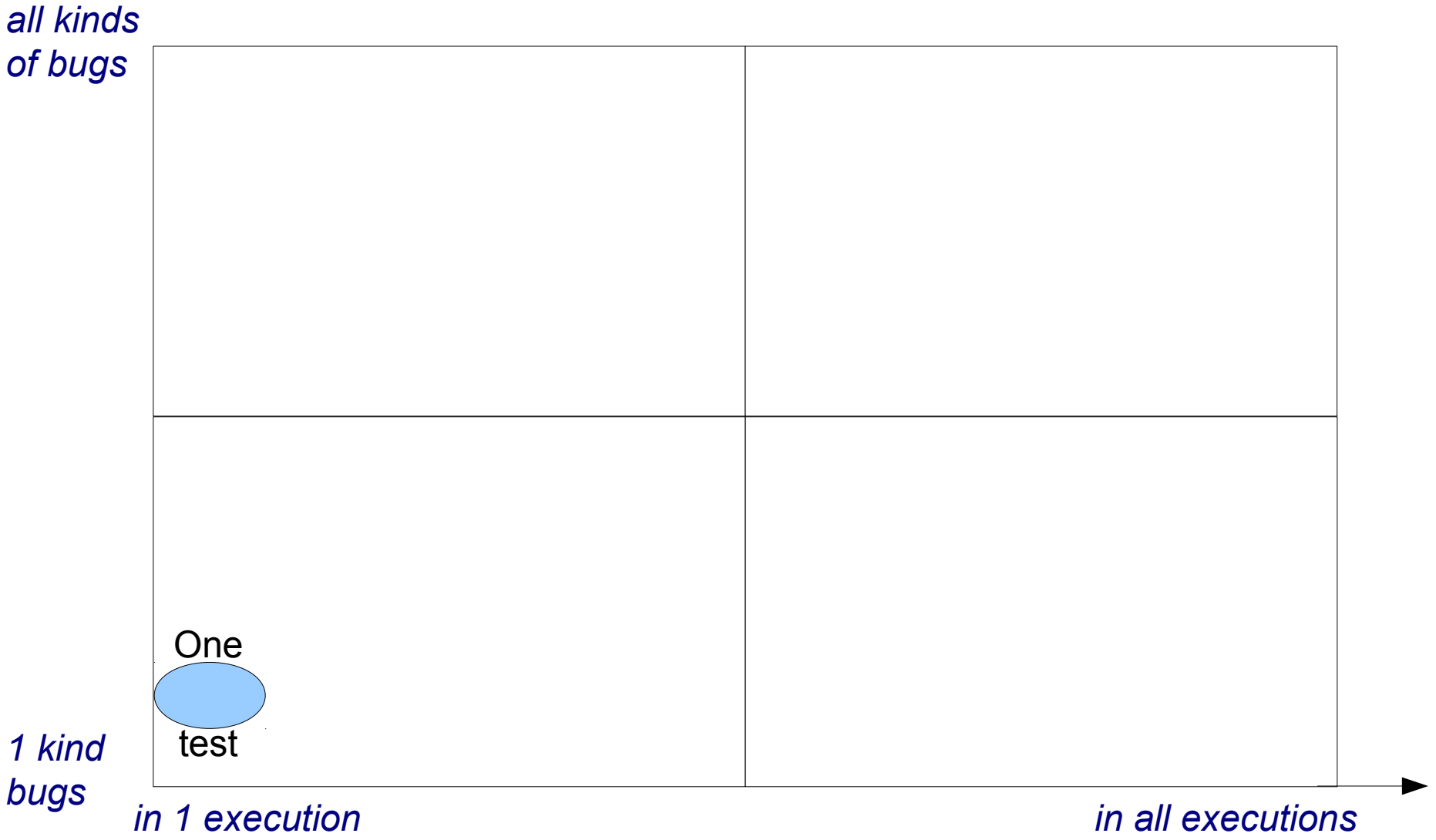
Static Verification	Runtime Verification
+ All paths at once	– One path only

Static Verification	Runtime Verification
+ All paths at once	– One path only
+ Hardware, test data and test environment **is not** required	– Hardware, test data and test environment **is** required

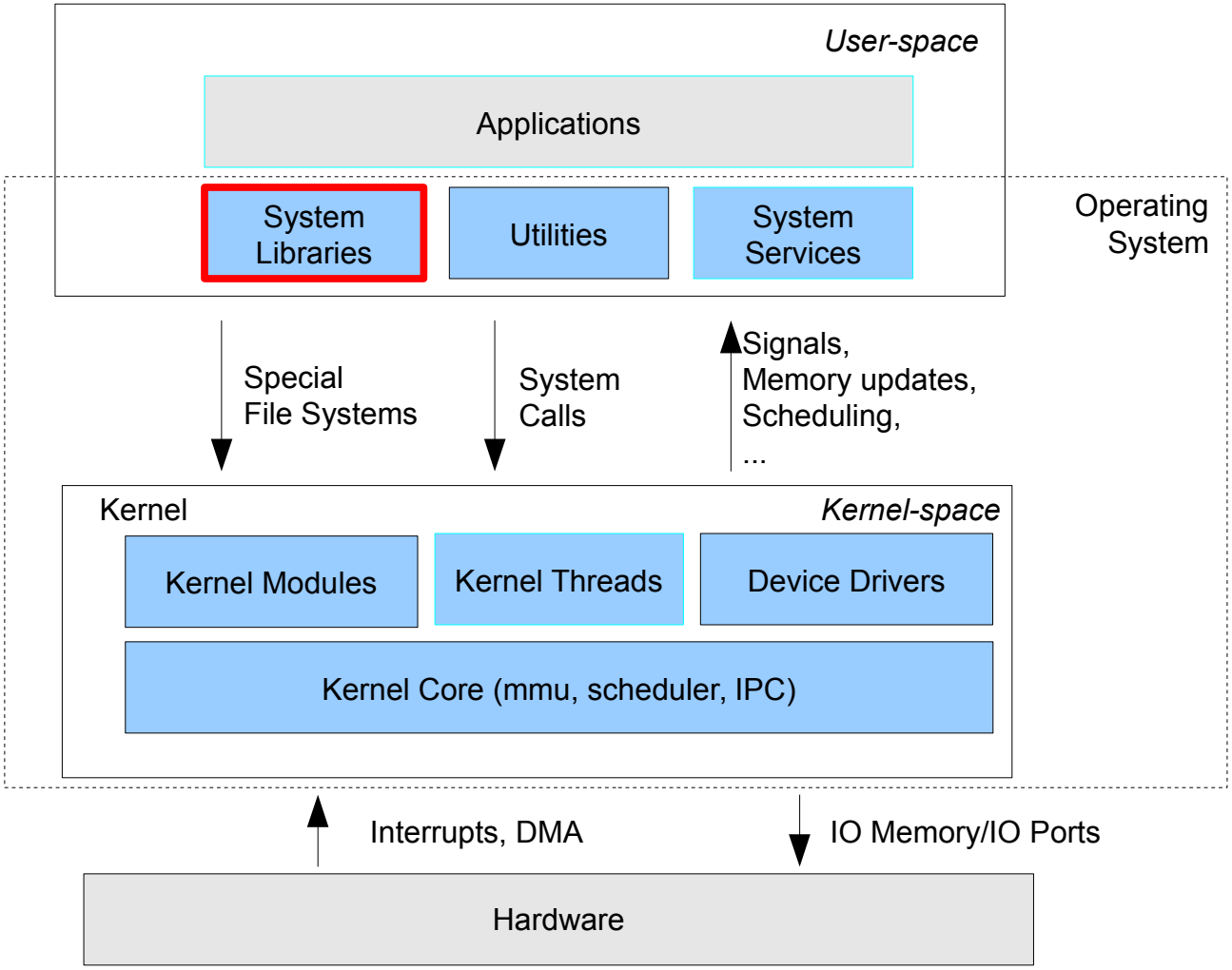
Static Verification	Runtime Verification
+ All paths at once	– One path only
+ Hardware, test data and test environment **is not** required	– Hardware, test data and test environment **is** required
– There are false positives	+ Almost no false positives

Static Verification	Runtime Verification
+ All paths at once	– One path only
+ Hardware, test data and test environment **is not** required	– Hardware, test data and test environment **is** required
– There are false positives	+ Almost no false positives
– Checks for predefined set of bugs only	+ The only way to show the code actually works

Verification Approaches



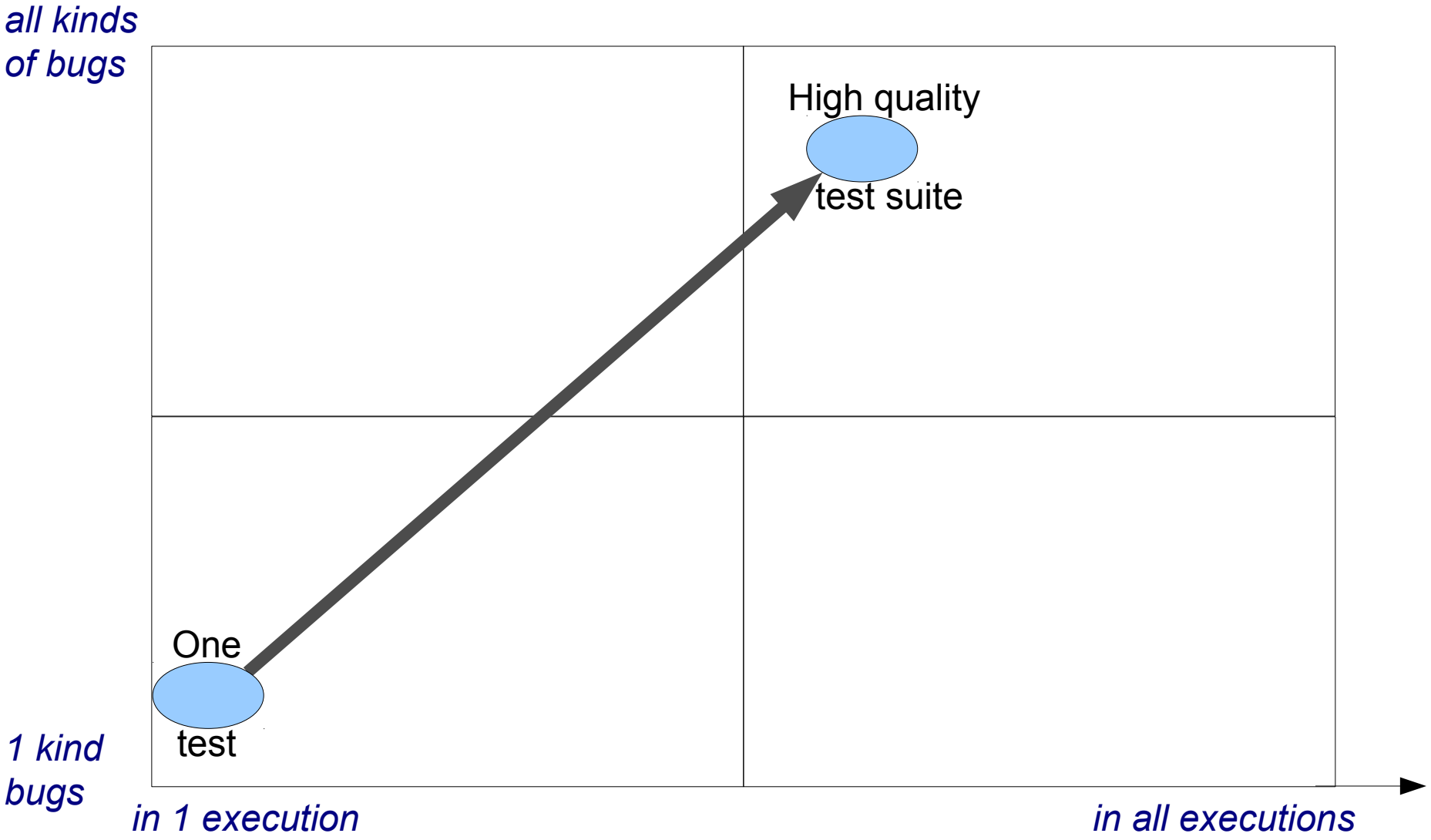
Operating Systems



Functional Testing

```
void test_memcpy(void)
{
    char *dst, *src;
    char *res;

    dst = malloc(16);
    assert(dst != NULL, "not enough memory");

    src = "0123465789";
    res = memcpy(dst, src, 10);
    assert(res == dst, "memcpy returns incorrect pointer %p, "
                       "while %p is expected", res, dst);
    assert(memcmp(src, dst, 10) == 0, "wrong result of copying");

    free(dst);
}
```

Verification Approaches



LinuxTesting.org

T2C – Template2C

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Requirements Catalogue

NAME

memcpy - copy bytes in memory

SYNOPSIS

```
#include <string.h>

void *memcpy(void *restrict s1, const void *restrict s2, size_t n);
```

DESCRIPTION

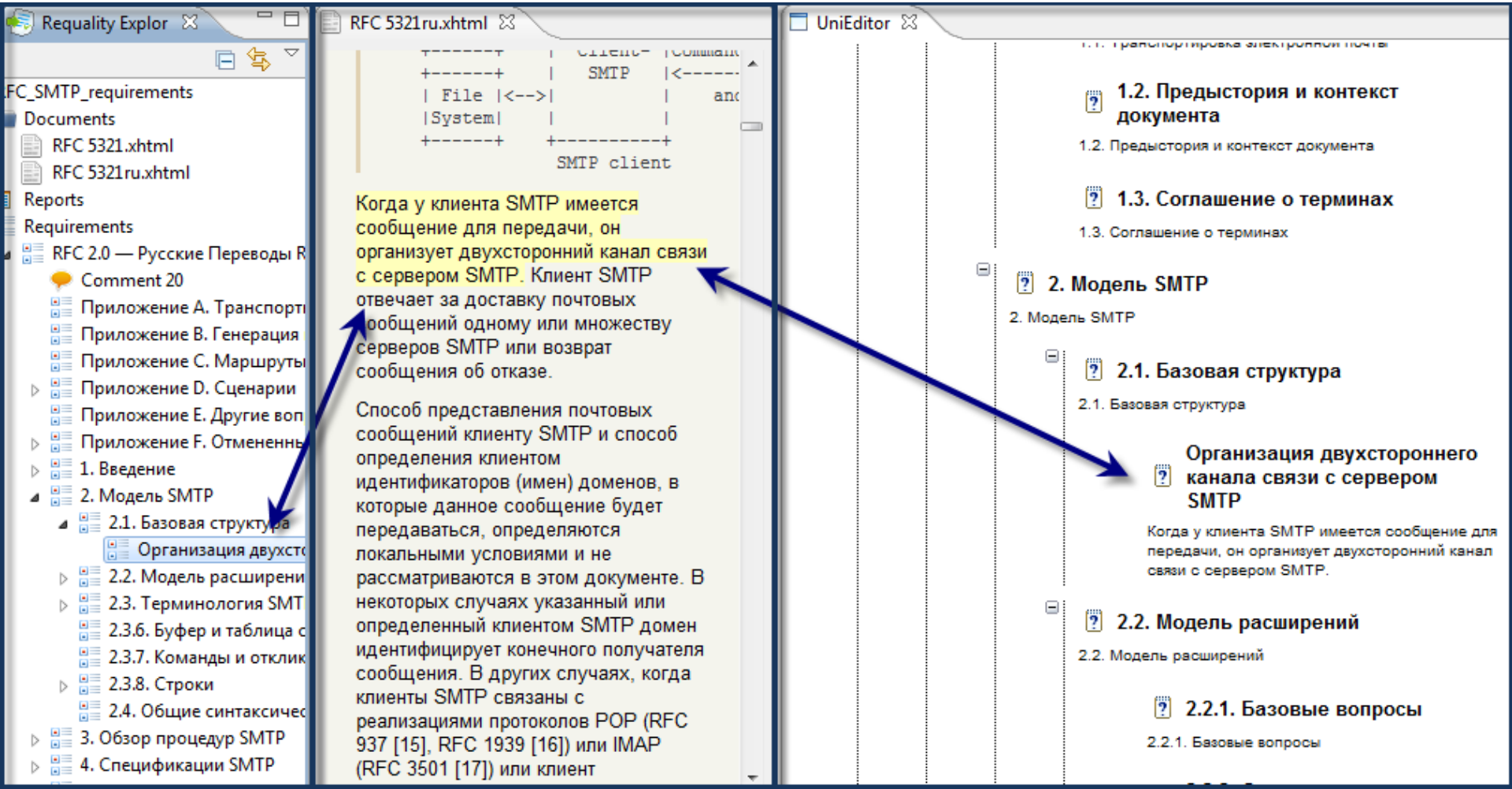
[CX] ⌦ The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of IEEE Std 1003.1-2001 defers to the ISO C standard. ⌫

{**memcpy.01**} The *memcpy*() function shall copy *n* bytes from the object pointed to by *s2* into the object pointed to by *s1*. {**app.memcpy.02**} If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

{**memcpy.03**} The *memcpy*() function shall return *s1*; no return value is reserved to indicate an error.

Requirements Catalogue:Requality



T2C test for memcpy

```
<CODE>
  char *res, *buffer = NULL;

  buffer = malloc( 200 );
  if (buffer == NULL) ABORT_TEST_PURPOSE("Not enough memory");

  // If copying takes place between objects that overlap, the behavior is undefined.
  REQ("app.memcpy.02", "", !are_buffers_overlapped(buffer+S1,buffer+S2,N));

  res = memcpy( buffer, buffer + 100, 100 );

  // The memcpy() function shall copy n bytes from the object pointed to by s2
  // into the object pointed to by s1.
  REQ("memcpy.01", "", buffer_compare( buffer, buffer + 100, 100 ) == 0 );

  // The memcpy() function shall return s1
  REQ("memcpy.03", "", res == buffer );


    if (buffer != NULL) free( buffer );
</CODE>
```

T2C test for memcpy

```
<CODE>
  char *res, *buffer = NULL;

  buffer = malloc( 200 );
  if (buffer == NULL) ABORT_TEST_PURPOSE("Not enough memory");

  // If copying takes place between objects that overlap, the behavior is undefined.
  REQ("app.memcpy.02", "", !are_buffers_overlapped(buffer+S1,buffer+S2,N));

  res = memcpy( buffer, buffer + 100, 100 );

  // The memcpy() function shall copy n bytes from the object pointed to by s2
  // into the object pointed to by s1.
  REQ("memcpy.01", "", buffer_compare( buffer, buffer + 100, 100 ) == 0 );

  // The memcpy() function shall return s1
  REQ("memcpy.03", "", res == buffer );
</CODE>
<FINALLY>
    if (buffer != NULL) free( buffer );
</FINALLY>
```

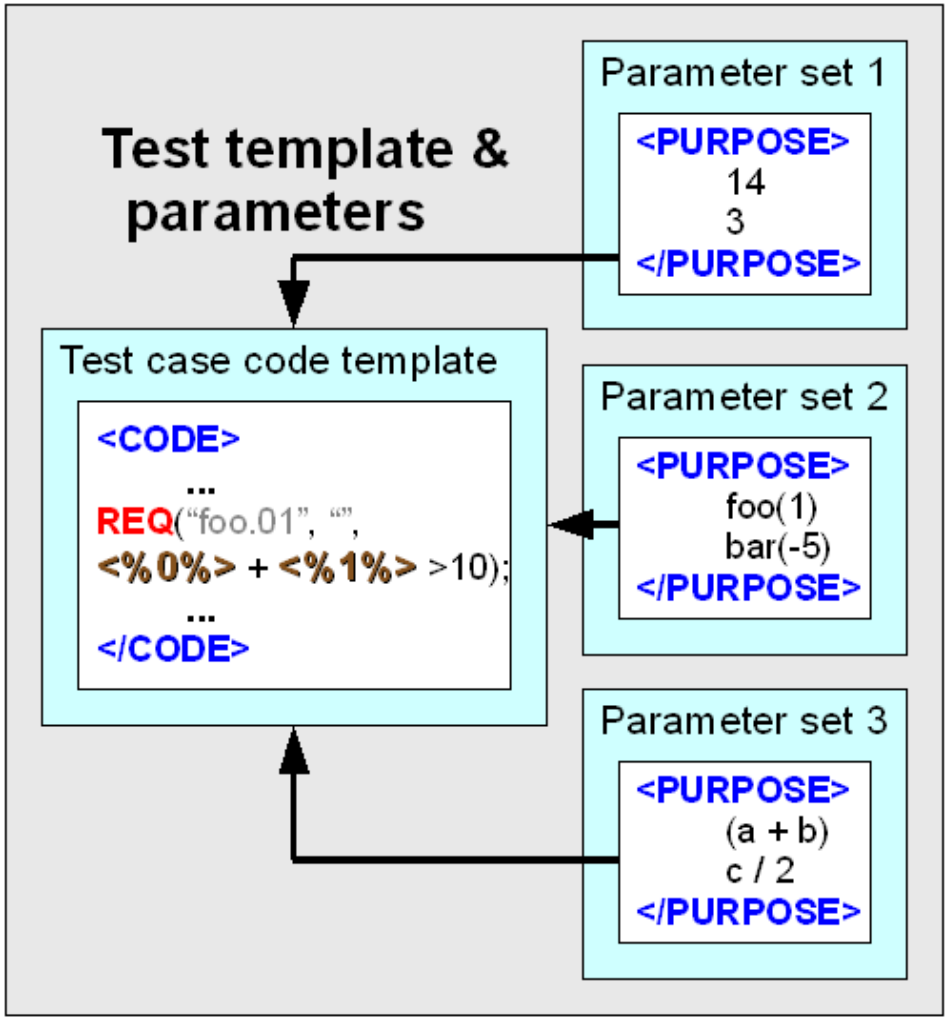
T2C test for memcpy

```
<CODE>
  char *res, *buffer = NULL;

  buffer = malloc( SIZE );
  if (buffer == NULL) ABORT_TEST_PURPOSE("Not enough memory");

  // If copying takes place between objects that overlap, the behavior is undefined.
  REQ("app.memcpy.02", "", !are_buffers_overlapped(buffer+S1,buffer+S2,N));

  res = memcpy( buffer + S1, buffer + S2, N );

  // The memcpy() function shall copy n bytes from the object pointed to by s2
  // into the object pointed to by s1.
  REQ("memcpy.01", "", buffer_compare( buffer + S1, buffer + S2, N ) == 0 );

  // The memcpy() function shall return s1
  REQ("memcpy.03", "", res == buffer + S1 );
</CODE>
<FINALLY>
    if (buffer != NULL) free( buffer );
</FINALLY>
```

```
<PURPOSE>
  1000
  0
  50
  50
</PURPOSE>
```

```
<PURPOSE>
  1000
  50
  0
  50
</PURPOSE>
```

Test Generation by Template



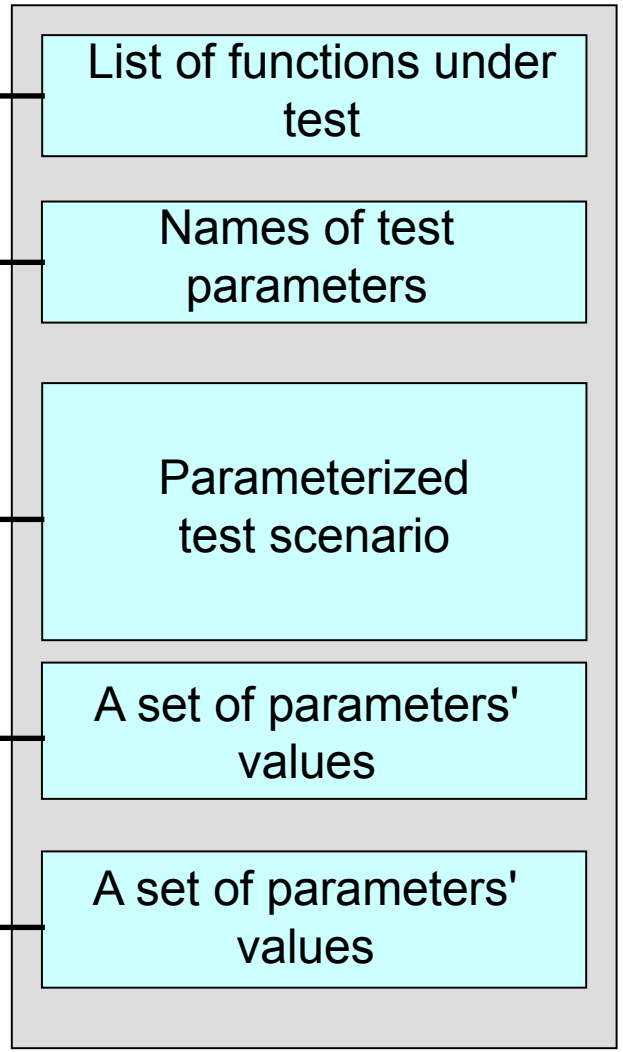

```
void test_purpose_1()
{
  ...
  REQ("foo.01", "",
  14 + 3 > 10);
  ...
}
```

```
void test_purpose_2()
{
  ...
  REQ("foo.01", "",
  foo(1) + bar(-5) > 10);
  ...
}
```

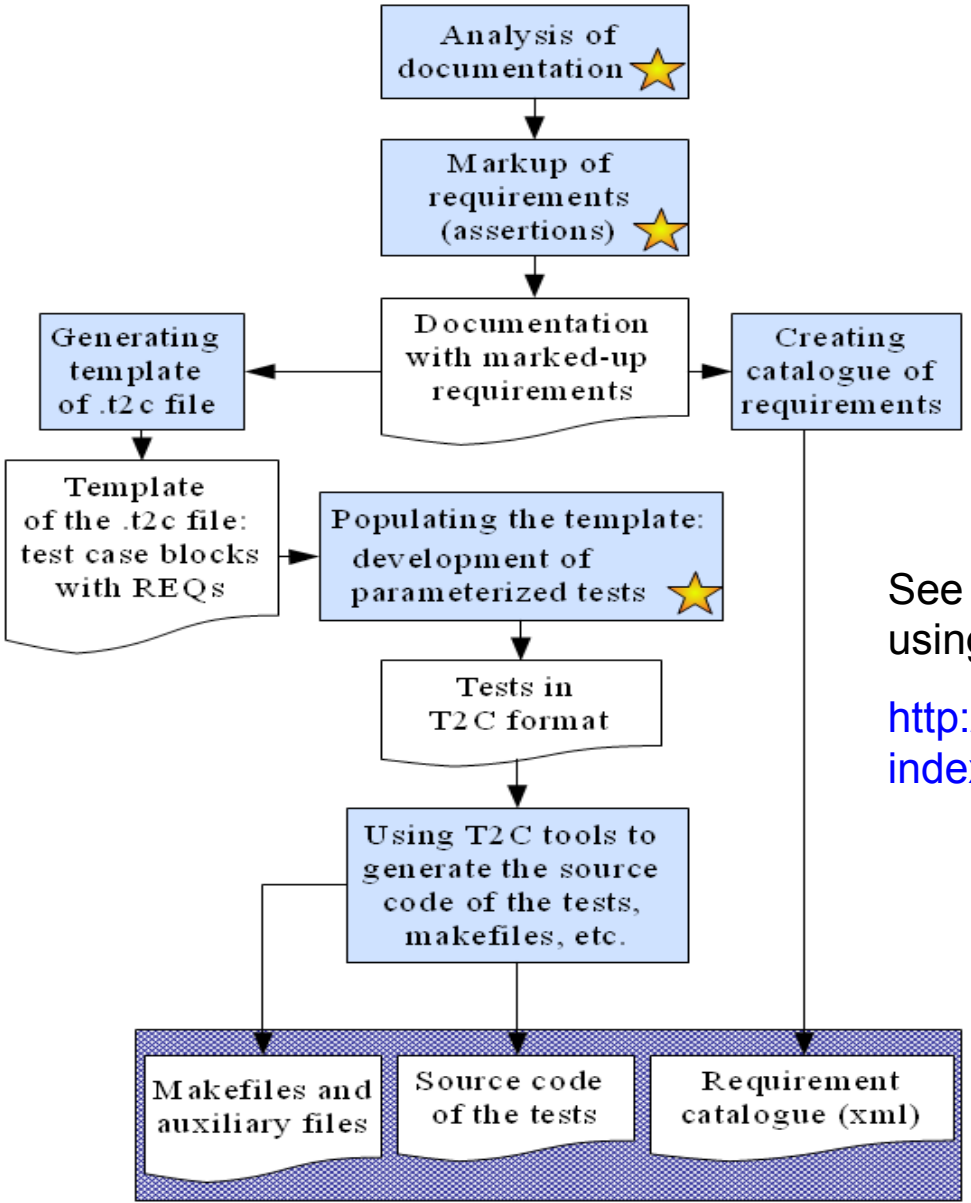
```
void test_purpose_3()
{
  ...
  REQ("foo.01", "",
  (a + b) + c / 2 > 10);
  ...
}
```

T2C Main Elements

```
<BLOCK>
    <TARGETS>
        g_array_remove_range
    </TARGETS>

    <DEFINE>
        #define TYPE <%0%>
        #define INDEX <%1%>
    </DEFINE>

    <CODE>
        ...
        REQ("g_array_remove_range.01", "",
        g_array_remove_range(ga, TYPE, INDEX) != old);
        ...
    </CODE>

    <PURPOSE>
        int
        6
    </PURPOSE>

    <PURPOSE>
        double
        999
    </PURPOSE>

</BLOCK>
```



T2C Test Development Process



See details about test development using T2C framework here:

http://ispras.linux-foundation.org/index.php/T2C_Framework

T2C Results – LSB Desktop

Target Library	Version	Interfaces (Tested of Total)	Requirements (Tested of Total)	Code Coverage (Lines of Code)	Bugs Found
libatk-1.0	1.19.6	222 of 222 (100%)	497 of 515 (96%)	-	11
libglib-2.0	2.14.0	832 of 847 (98%)	2290 of 2461 (93%)	12203 of 16263 (75.0%)	13
libgthread-2.0	2.14.0	2 of 2 (100%)	2 of 2 (100%)	149 of 211 (70.6%)	0
libgobject-2.0	2.16.0	313 of 314 (99%)	1014 of 1205 (84%)	5605 of 7000 (80.1%)	2
libgmodule-2.0	2.14.0	8 of 8 (100%)	17 of 21 (80%)	211 of 270 (78.1%)	2
libfonconfig	2.4.2	160 of 160 (100%)	213 of 272 (78%)	-	11
Total		**1537 of 1553 (99%)**	**4033 of 4476 (90%)**	**18168 of 23744 (76.5%)**	**39**

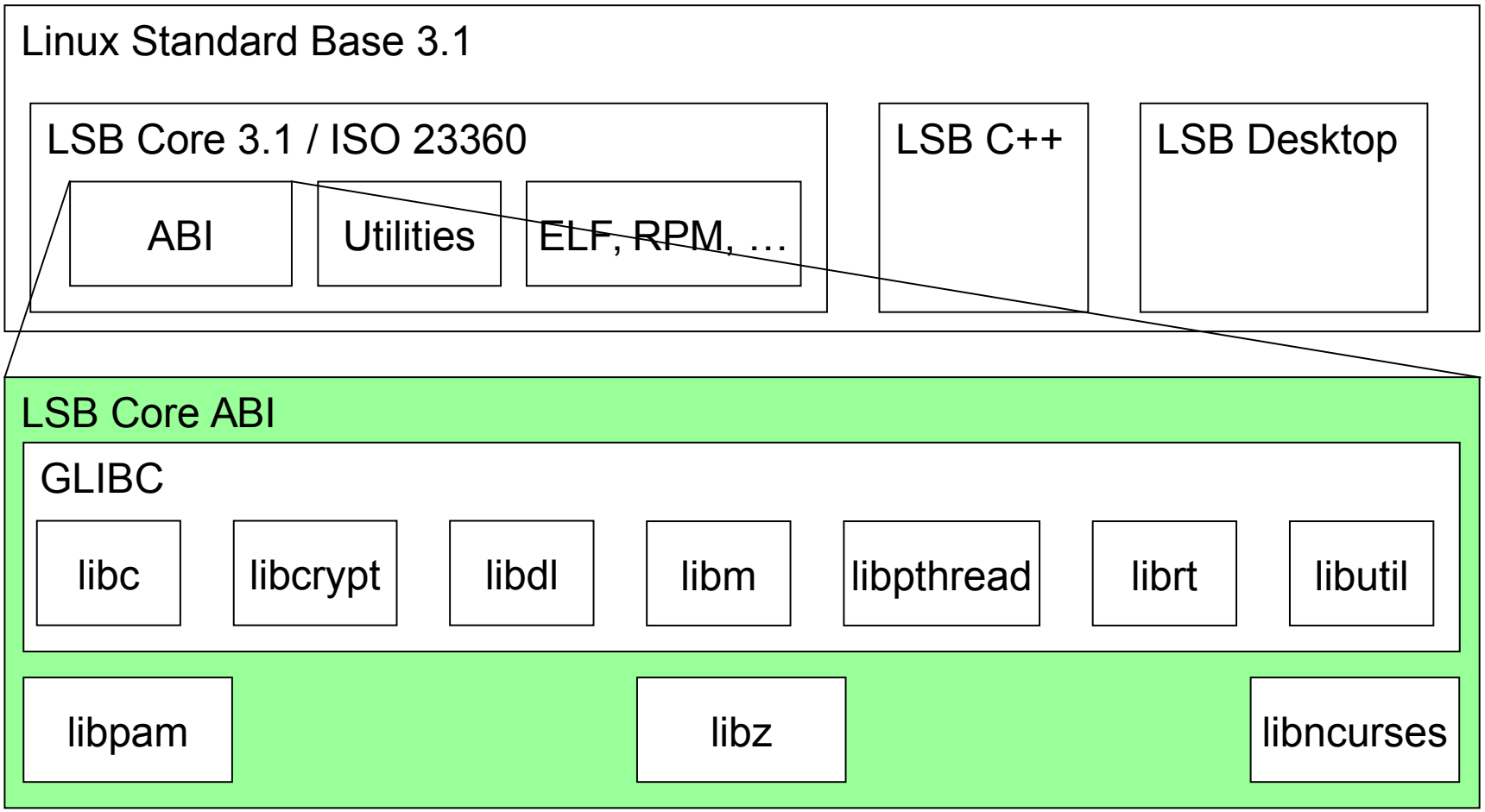
LinuxTesting.org

OLVER – Model Based Testing

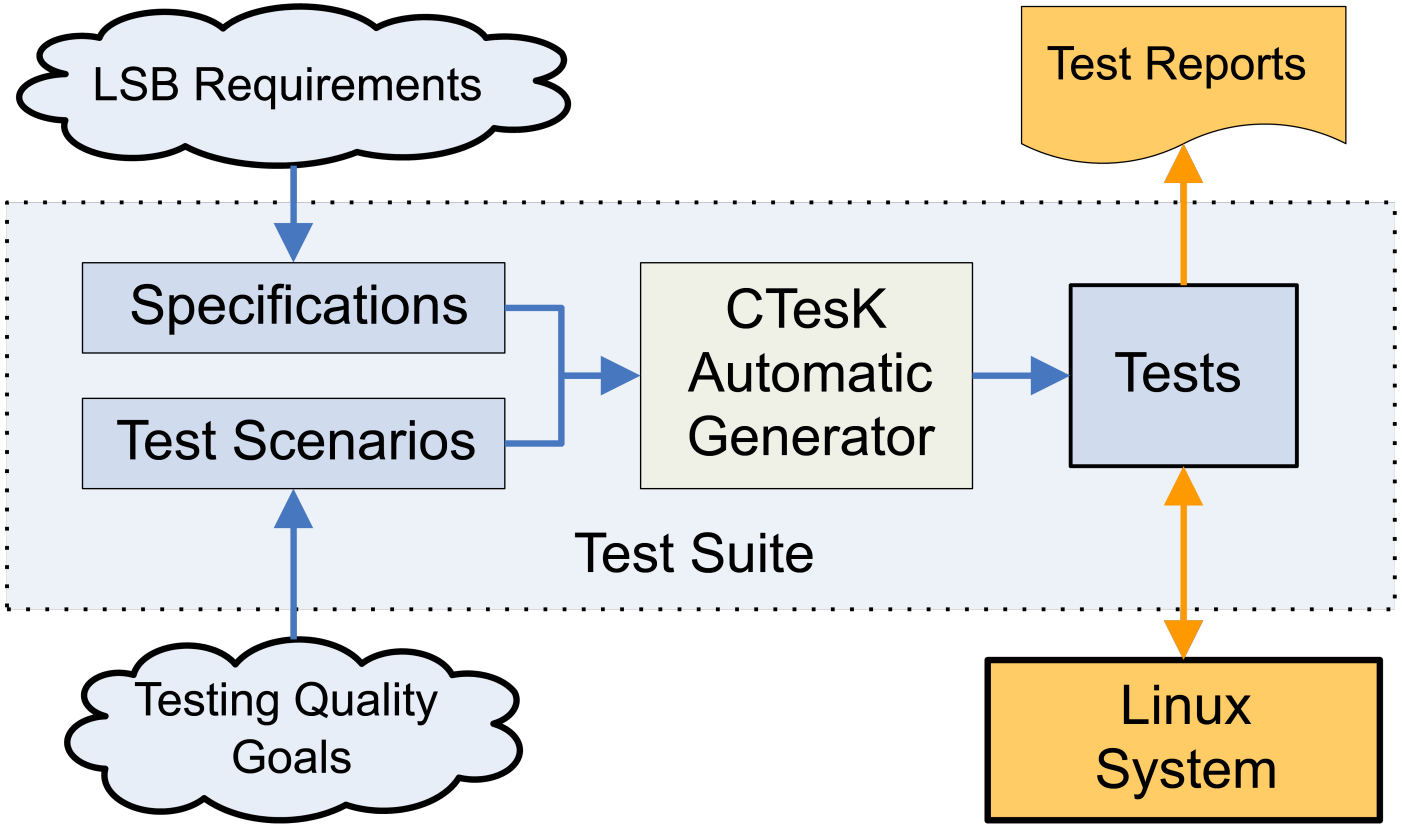
ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Open Linux VERification



OLVER Process



Requirements Catalogue

NAME

memcpy - copy bytes in memory

SYNOPSIS

```
#include <string.h>

void *memcpy(void *restrict s1, const void *restrict s2, size_t n);
```

DESCRIPTION

[CX] ⌦ The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of IEEE Std 1003.1-2001 defers to the ISO C standard. ⌫

{**memcpy.01**} The *memcpy*() function shall copy *n* bytes from the object pointed to by *s2* into the object pointed to by *s1*. {**app.memcpy.02**} If copying takes place between objects that overlap, the behavior is undefined.

RETURN VALUE

{**memcpy.03**} The *memcpy*() function shall return *s1*; no return value is reserved to indicate an error.

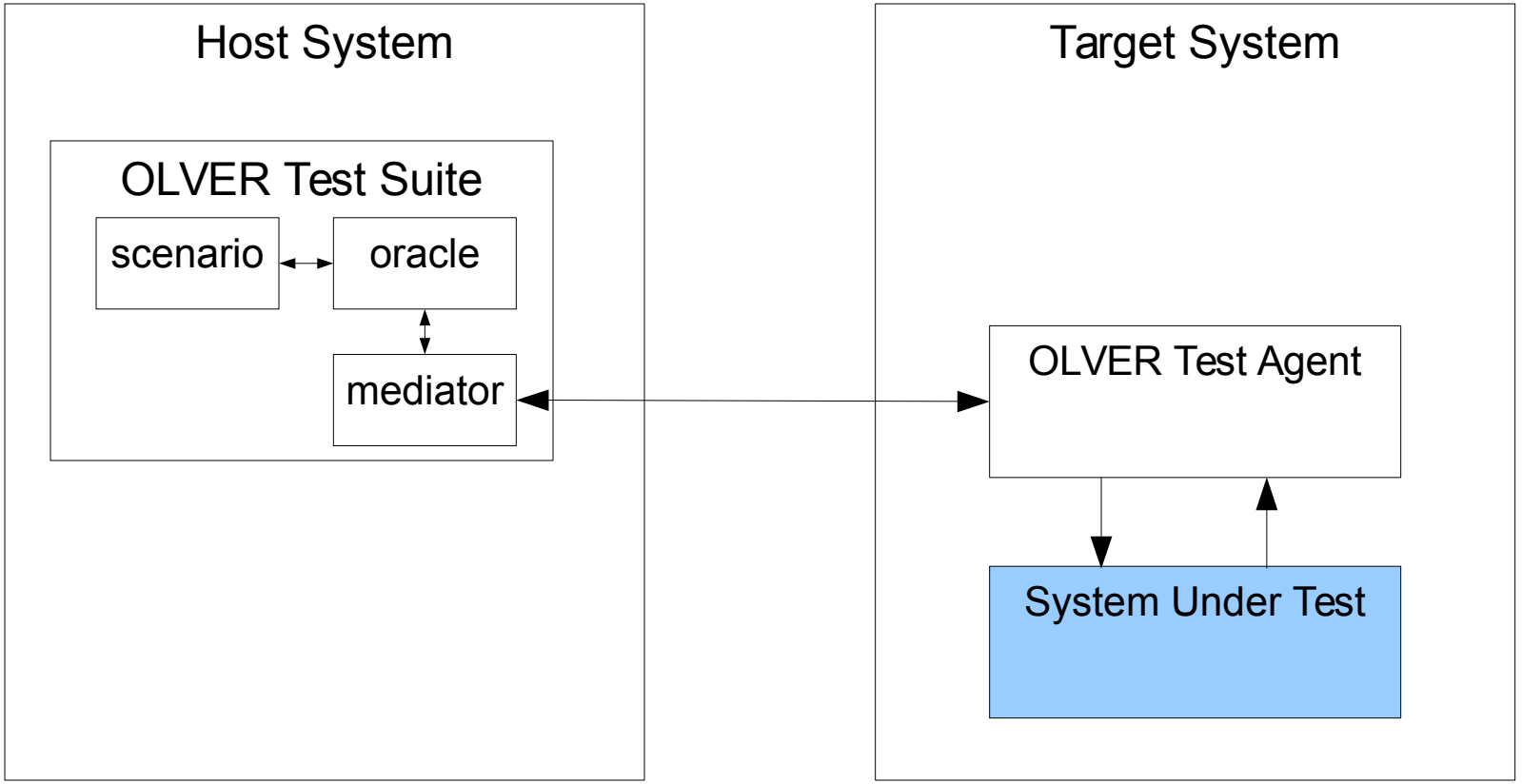
memcpy() specification template

```
{
    pre
    {
        // If copying takes place between objects that overlap, the behavior is undefined.
        REQ("app.memcpy.02", "Objects are not overlapped", TODO_REQ() );

        return true;
    }
    post
    {
        /*The memcpy() function shall copy n bytes from the object
          pointed to by s2 into the object pointed to by s1. */
        REQ("memcpy.01", "s1 contain n bytes from s2", TODO_REQ() );

        /* The memcpy() function shall return s1; */
        REQ("memcpy.03", "memcpy() function shall return s1", TODO_REQ() );

        return true;
    }
}
```

memcpy() precondition

```
specification
VoidTPtr memcpy_spec( CallContext context, VoidTPtr s1, VoidTPtr s2, SizeT n )
{
    pre
    {
        /* [Consistency of test suite] */
        REQ("", "Memory pointed to by s1 is available in the context",
            isValidPointer(context,s1) );
        REQ("", "Memory pointed to by s2 is available in the context",
            isValidPointer(context,s2) );

        /* [Implicit precondition] */
        REQ("", "Memory pointed to by s1 is enough", sizeWMemoryAvailable(s1) >= n );
        REQ("", "Memory pointed to by s2 is enough", sizeRMemoryAvailable(s2) >= n );

        // If copying takes place between objects that overlap, the behavior is undefined.
        REQ("app.memcpy.02", "Objects are not overlapped",
            !areObjectsOverlapped(s1,n,s2,n) );

        return true;
    }
```

OLVER Distributed Architecture



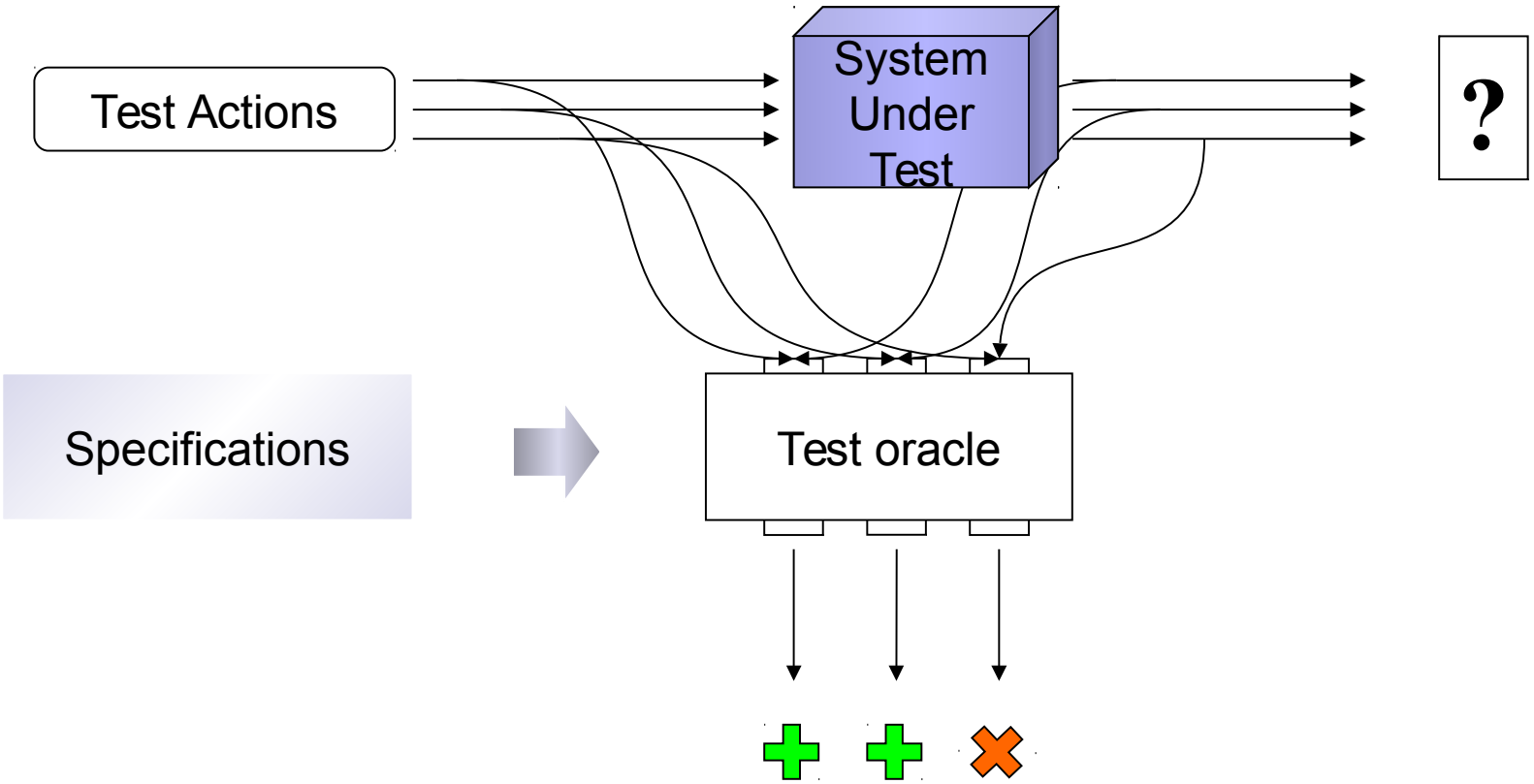
memcpy() precondition

```
specification
VoidTPtr memcpy_spec( CallContext context, VoidTPtr s1, VoidTPtr s2, SizeT n )
{
    pre
    {
        /* [Consistency of test suite] */
        REQ("", "Memory pointed to by s1 is available in the context",
            isValidPointer(context,s1) );
        REQ("", "Memory pointed to by s2 is available in the context",
            isValidPointer(context,s2) );

        /* [Implicit precondition] */
        REQ("", "Memory pointed to by s1 is enough", sizeWMemoryAvailable(s1) >= n );
        REQ("", "Memory pointed to by s2 is enough", sizeRMemoryAvailable(s2) >= n );

        // If copying takes place between objects that overlap, the behavior is undefined.
        REQ("app.memcpy.02", "Objects are not overlapped",
            !areObjectsOverlapped(s1,n,s2,n) );

        return true;
    }
```

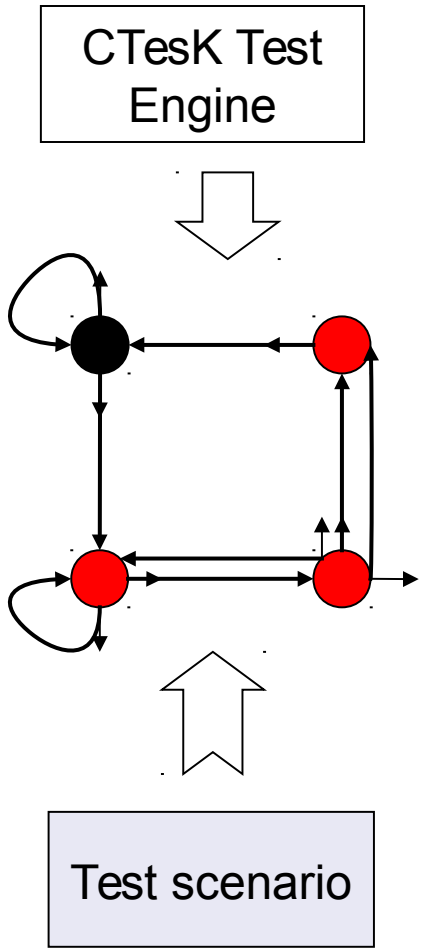
memcpy() postcondition

```
specification
VoidTPtr memcpy_spec( CallContext context, VoidTPtr s1, VoidTPtr s2, SizeT n ) {
    post
    {
        /*The memcpy() function shall copy n bytes from the object
          pointed to by s2 into the object pointed to by s1. */
        REQ("memcpy.01", "s1 contain n bytes from s2",
            equals(   readCByteArray_VoidTPtr(s1,n), @readCByteArray_VoidTPtr(s2,n) )
          );

        /* [The object pointed to by s2 shall not be changed] */
        REQ("", "s2 shall not be changed",
            equals(   readCByteArray_VoidTPtr(s2,n), @readCByteArray_VoidTPtr(s2,n) ));

        /* The memcpy() function shall return s1; */
        REQ("memcpy.03", "memcpy() function shall return s1",equals_VoidTPtr(memcpy_spec,s1) );

        /* [Other memory shall not be changed] */
        REQ("", "Other memory shall not be changed",
            equals(   readCByteArray_MemoryBlockExceptFor( getTopMemoryBlock(s1), s1, n ),
                    @readCByteArray_MemoryBlockExceptFor( getTopMemoryBlock(s1), s1, n )  ) );

        return true;
    }
```

Test Oracle Generation

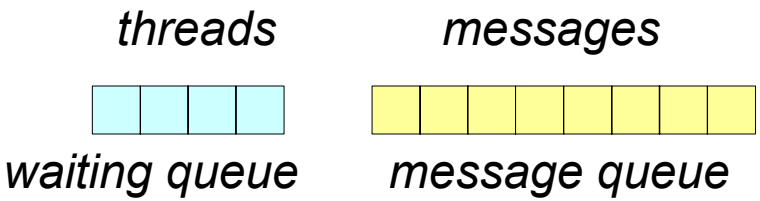


Test Scenarios Generation



- abstract state
- set of test actions
- Test Engine generates a sequence of actions
  - to ensure: each action is executed in each abstract state

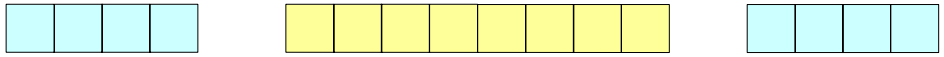
Bug Example - POSIX mq



- sending threads are blocked if queue is full
- receiving threads are blocked if queue is empty

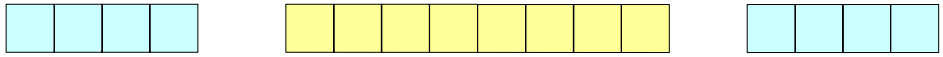
- Test scenario abstract state:
  - number of messages in the queue
  - number of threads waiting to send
  - number of threads waiting to receive

Bug Example - POSIX mq



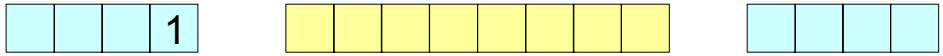
(0,0,0)

Bug Example - POSIX mq



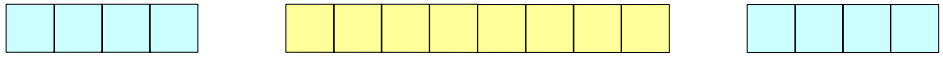
(0,0,0)

1. receiver blocks



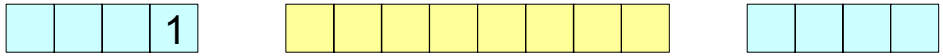
(1,0,0)

Bug Example - POSIX mq



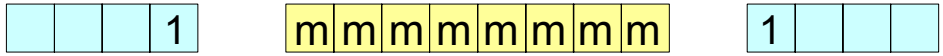
(0,0,0)

1. receiver blocks



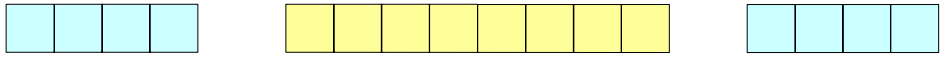
(1,0,0)

2. sender puts all messages and blocks



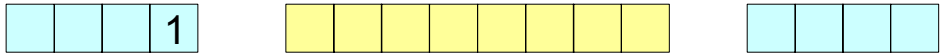
(1,N,1)

Bug Example - POSIX mq



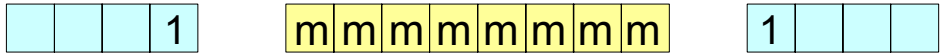
(0,0,0)

1. receiver blocks



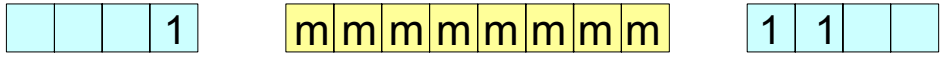
(1,0,0)

2. sender puts all messages and blocks



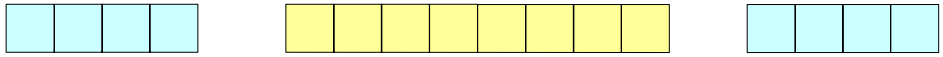
(1,N,1)

3. another high priority sender blocks



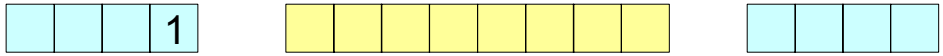
(1,N,2)

Bug Example - POSIX mq



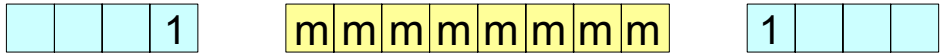
(0,0,0)

1. receiver blocks



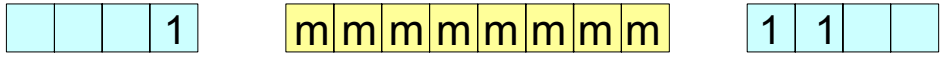
(1,0,0)

2. sender puts all messages and blocks



(1,N,1)

3. another high priority sender blocks

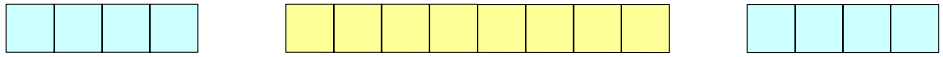


(1,N,2)

4. receiver receives all messages

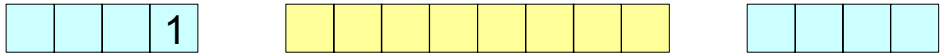
(0,0,1)

Bug Example - POSIX mq



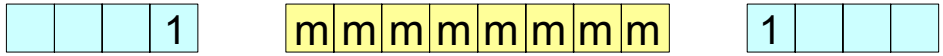
(0,0,0)

1. receiver blocks



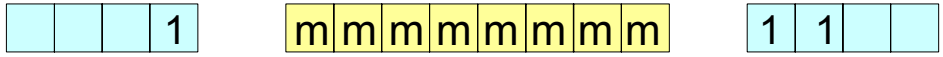
(1,0,0)

2. sender puts all messages and blocks



(1,N,1)

3. another high priority sender blocks



(1,N,2)

4. receiver receives all messages

(0,0,1)

5. 2nd sender never unblocked

OLVER Results

- Requirements catalogue built for LSB and POSIX
  - **1532** interfaces
  - **22663** elementary requirements
- **97 deficiencies** in specification reported
- Formal specifications and tests developed for
  - **1270 interface** (good quality)
  - + 260 interfaces (basic quality)
- **80+ bugs** reported in modern distributions
- OLVER is a part of the official LSB Certification test suite
  http://ispras.linuxfoundation.org

OLVER Conclusions

- model based testing allows to achieve better quality using less resources
- maintenance of MBT is cheaper

OLVER Conclusions

- model based testing allows to achieve better quality using less resources
  if you have advanced test engineers
- maintenance of MBT is cheaper
  if you have advanced test engineers

OLVER Conclusions

- model based testing allows to achieve better quality using less resources
  if you have advanced test engineers
- maintenance of MBT is cheaper
  if you have advanced test engineers
- traditional tests are more useful for typical test engineers and developers

OLVER Conclusions

- model based testing allows to achieve better quality using less resources
  if you have advanced test engineers
- maintenance of MBT is cheaper
  if you have advanced test engineers
- traditional tests are more useful for typical test engineers and developers
- so, long term efficiency is questionable
- but...

OLVER Conclusions

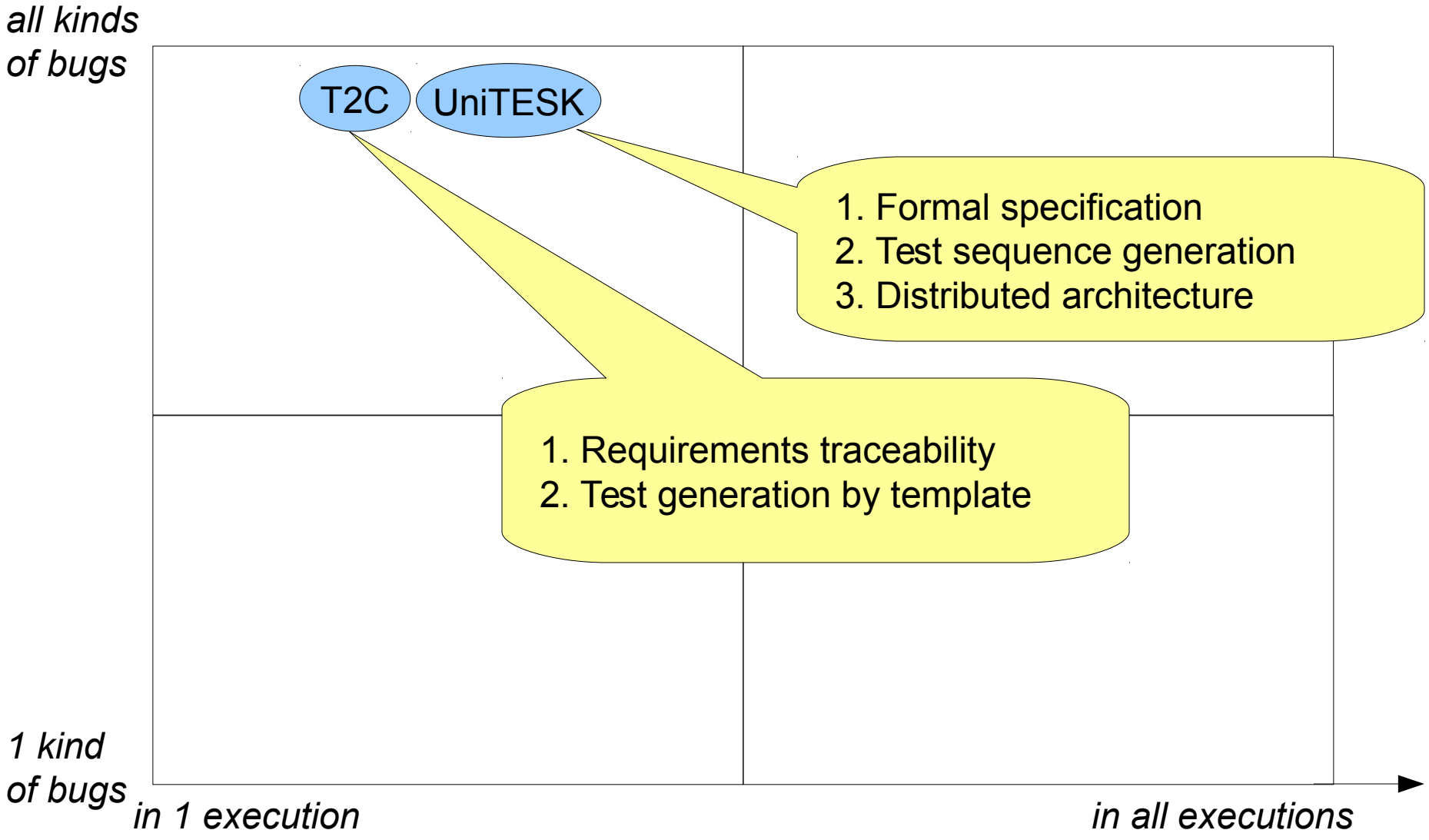
Subject **[lsb-bugs] [Bug 3285] New ncurses test failures** 11.05.2016 06:08

To lsb-bugs@lists.linuxfoundation.org

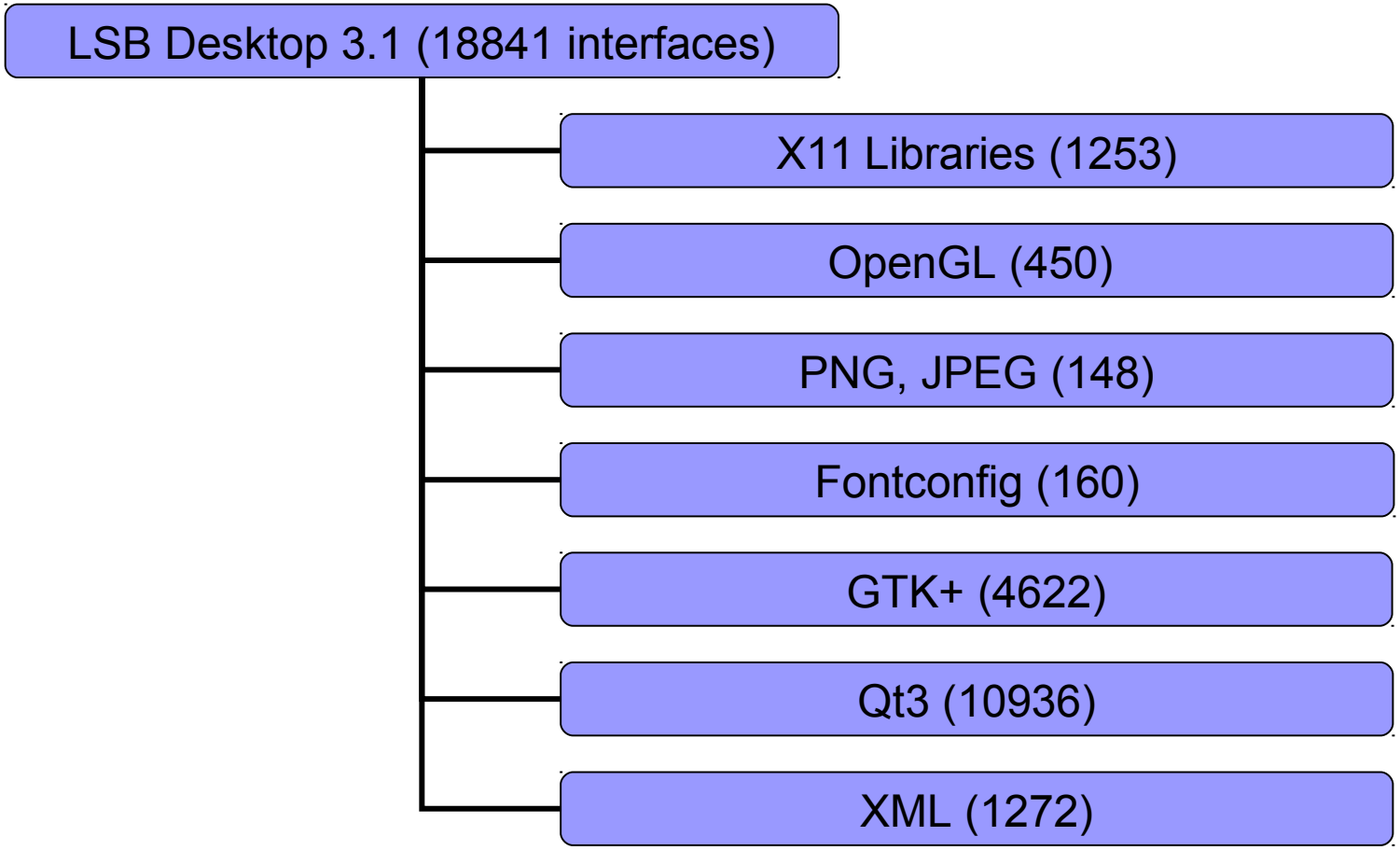
Comment # 13 on bug 3285 from zhpeng@linx-info.com

```
As the nature of the bug is that agent side pty state is uncertained, we have 2
solving methods. One is on agent side, the other one is on local model side.

Method one is described in the patch and previous comments.

Method tow is also Jeff's suggestion, I tried it but I found it is not
sufficient. The bug in test rw_case_a, rw_case_b and rw_canon can be fixed, but
rw_case_d can not because its program structure is different from the other 3.

In test rw_case_a, rw_case_b and rw_canon, there is a common scenario to set
the local termios, that is setattr_a_scen in rw_case_a, setattr_b_scen in
rw_case_b, tcsetattr_rw_scen in rw_canon.

But rw_case_d has not such a setattr_d_scen between open_d_scen and
read_d_scen. I added a setattr_d_scen following the other rw_case*, but the
test still failed. I gave up this way.

So I still advise to fix the bug in the agent side personally.

The code for fixing the bug on local model side is as followed, they are
suitable for test rw_case_a, rw_case_b and rw_canon separately.

Note that the csize and cread flag should be set explicitly.

---
/home/rocky/source/original/lsb-test-olver-core-4.1.4/src/model/io/term/tests/rw_case_
       2012-10-26 06:29:03.000000000 +0800
+++ rw_case_a_scenario.sec      2016-05-11 10:13:01.000000000 +0800
@@ -241,6 +241,15 @@
```

OS Verification



LSB Desktop


LSB Desktop 3.1 (18841 interfaces)
X11 Libraries (1253)
OpenGL (450)
PNG, JPEG (148)
Fontconfig (160)
GTK+ (4622)
Qt3 (10936)
XML (1272)

Some Statistics

	Release Date	System Calls	Libraries	Interfaces	Utilities
Debian 7.0	May 2013	~350	~1650	~ 720 000	~10 000
RTOS	Nov 2013	~200	-	~700	~80

LinuxTesting.org

API Sanity Autotest

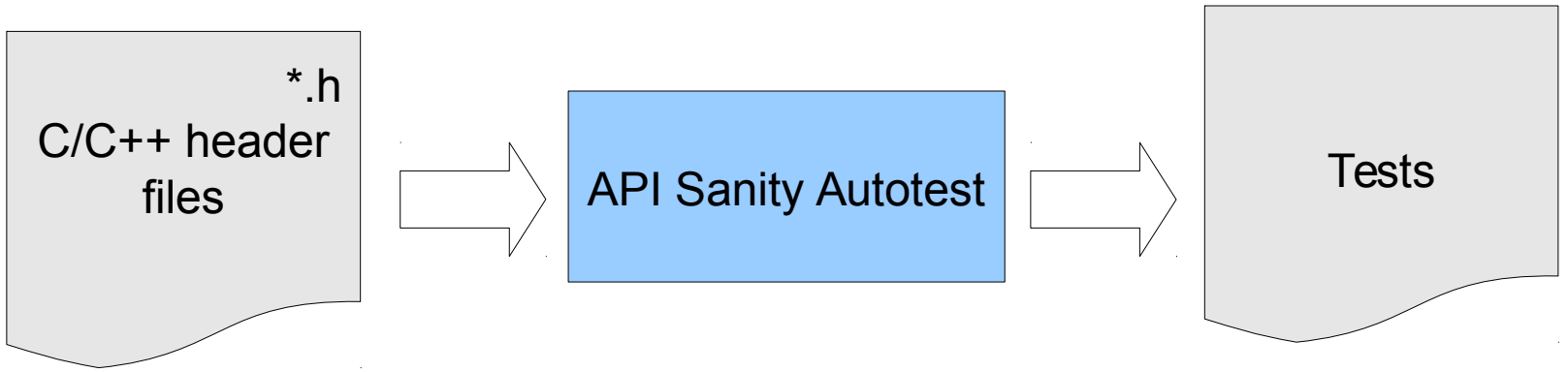
ISP RAS

Institute for System Programming of the Russian Academy of Sciences

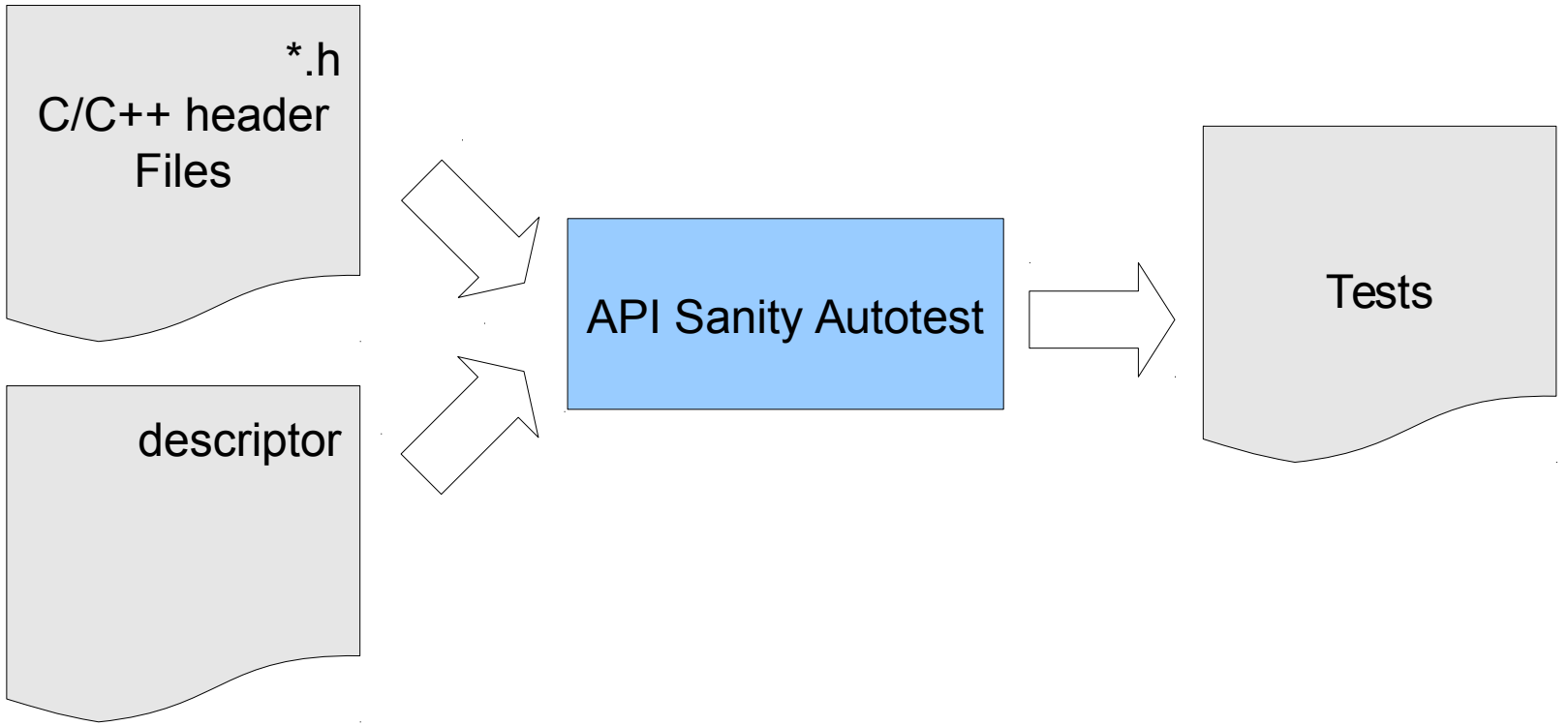
Smoke Testing

Smoke testing – checks only main use cases for basic requirements, i.e. the system under test is not broken and its results looks like correct.

API Sanity Autotest



API Sanity Autotest



Additional semantic information

- How to initialize library
- Hot to get a valid data of a particular type
- What is a valid argument of a particular function
- Which check can be done for returned types if it is of a particular type

Special Expressions

- $(type) – create an object of particular type

```
void create_QProxyModel(QProxyModel* Obj) {
     Obj->setSourceModel($(QItemModel*));
  }
```

- $[interface] – call the interface with some valid arguments

```
xmlListPtr create_filled_list() {
     xmlListPtr l = $[xmlListCreate];
     int num = 100;
     xmlListPushBack(l,&num);
     return l;
  }
```

Sources of Scalability

- Special expressions
- Extensive reuse

→ Minimal duplication of code

Results

Libraries	Number of interfaces
libqt-mt	9 792
libQtCore	2 066
libQtGui	7 439
libQtNetwork	406
libQtOpenGL	84
libQtSql	362
libQtSvg	66
libQtXml	380
libxml2	1 284
Total	**21 879**

Bugs Found (1)

git index : freetype/freetype2.git

The FreeType 2 library

summary refs log tree commit diff about

author	Werner Lemberg <wl@gnu.org>	2010-05-22 18:03:41 (GMT)
committer	Werner Lemberg <wl@gnu.org>	2010-05-22 18:03:41 (GMT)
commit	e30de299f28370ed5aa65755c6be69da58eefc72 (patch)	
tree	35355e4d7b42156baea7a21037d50f4c4ca5753c	
parent	09344385ee652cb8df33d35f07627c93e827f10d (diff)	
download	freetype2-e30de299f28370ed5aa65755c6be69da58eefc72.tar.gz	

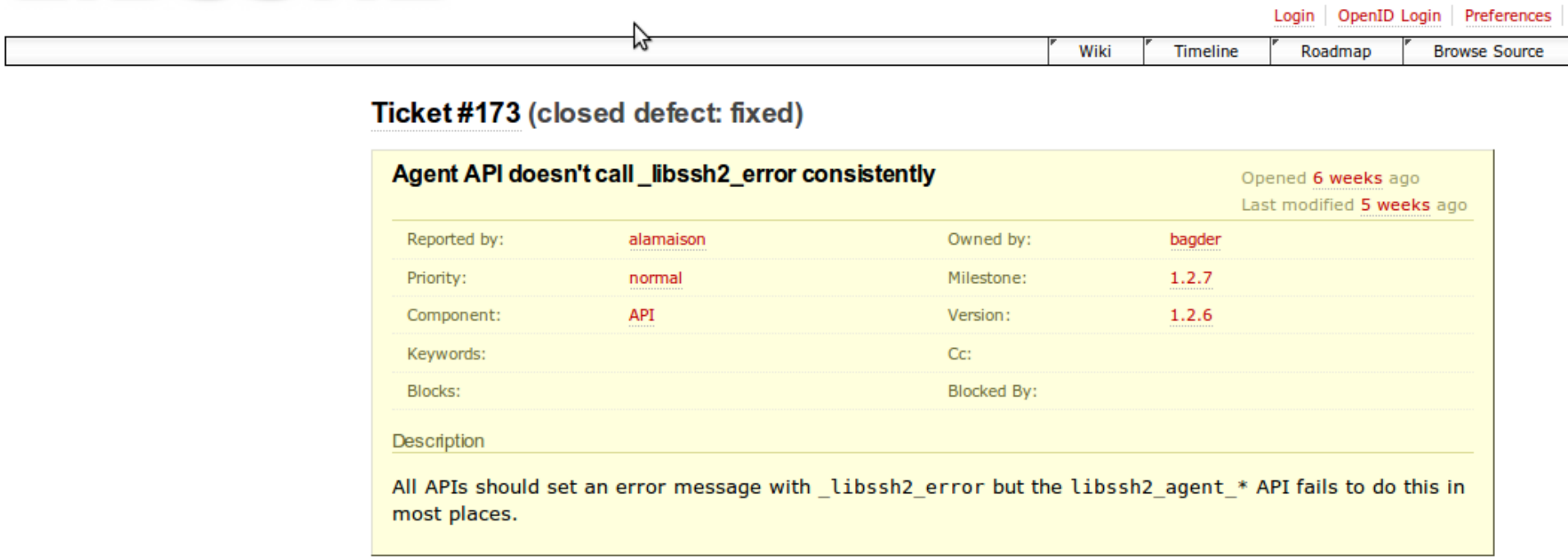
Fix various memory problems found by linuxtesting.org.

```
* src/base/ftgxval.c (FT_TrueTypeGX_Free, FT_ClassicKern_Free),
src/base/ftotval.c (FT_OpenType_Free), src/base/ftpfr.c
(ft_pfr_check): Check `face'.

* src/base/ftobjs.c (FT_Get_Charmap_Index): Check `charmap' and
`charmap->face'.
(FT_Render_Glyph): Check `slot->face'.
(FT_Get_SubGlyph_Info): Check `glyph->subglyphs'.

Improve API documentation.
```

http://git.savannah.gnu.org/cgit/freetype/freetype2.git/commit/?id=e30de299f28370ed5aa65755c6be69da58eefc72

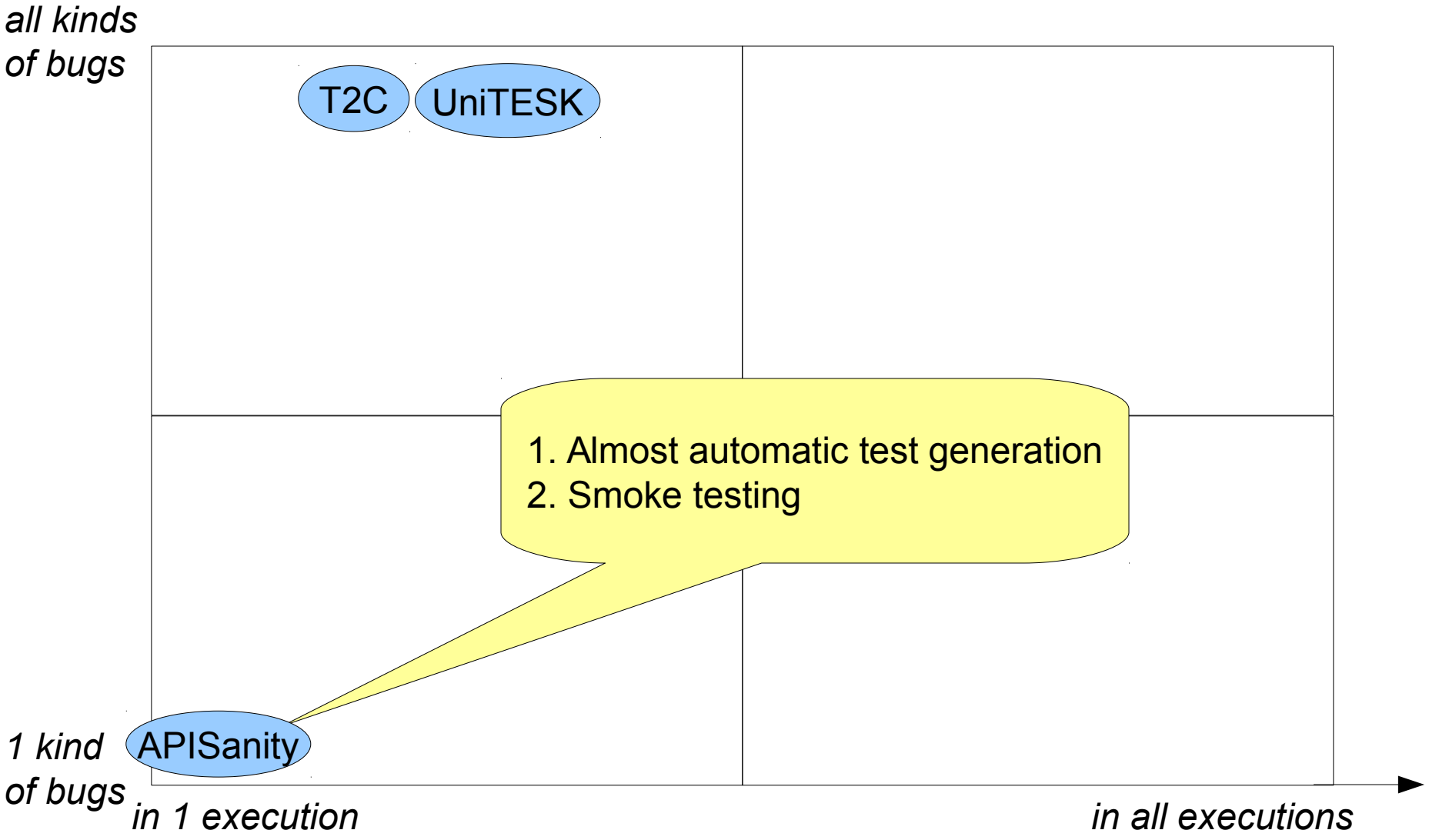
Bugs Found (1)

LIBSSH2



http://trac.libssh2.org/ticket/173

OS Verification



Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest
Kinds of Observable Events			
interface events	+	+	+
internal events			
Events Collection			
internal	+	+	+
external			
embedded			
Requirements Specification			
in-place (local, tabular)	+		+
formal model (pre/post+invariants,...)		+	
assertions/prohibited events	External	External	External
Events Analysis			
online	+	+	+
in-place	+		+
outside		+	
offline			

Test Aspects (2)

ISP RAS

Active Aspects	T2C	OLVER	Autotest
Target Test Situations Set			
requirements coverage	+	+	
class equivalence coverage		+	
model coverage (SUT/reqs)		+	
source code coverage			
Test Situations Setup/Set Gen			
passive			
fixed scenario	+		+
manual	+		
pre-generated			
coverage driven			
random			+-
adapting scenario		+	
coverage driven		+	
source code coverage			
model/... coverage		+	
random			
Test Actions			
application interface	+	+	+
HW interface			
internal actions			
inside			
outside			

LinuxTesting.org

Configuration Testing

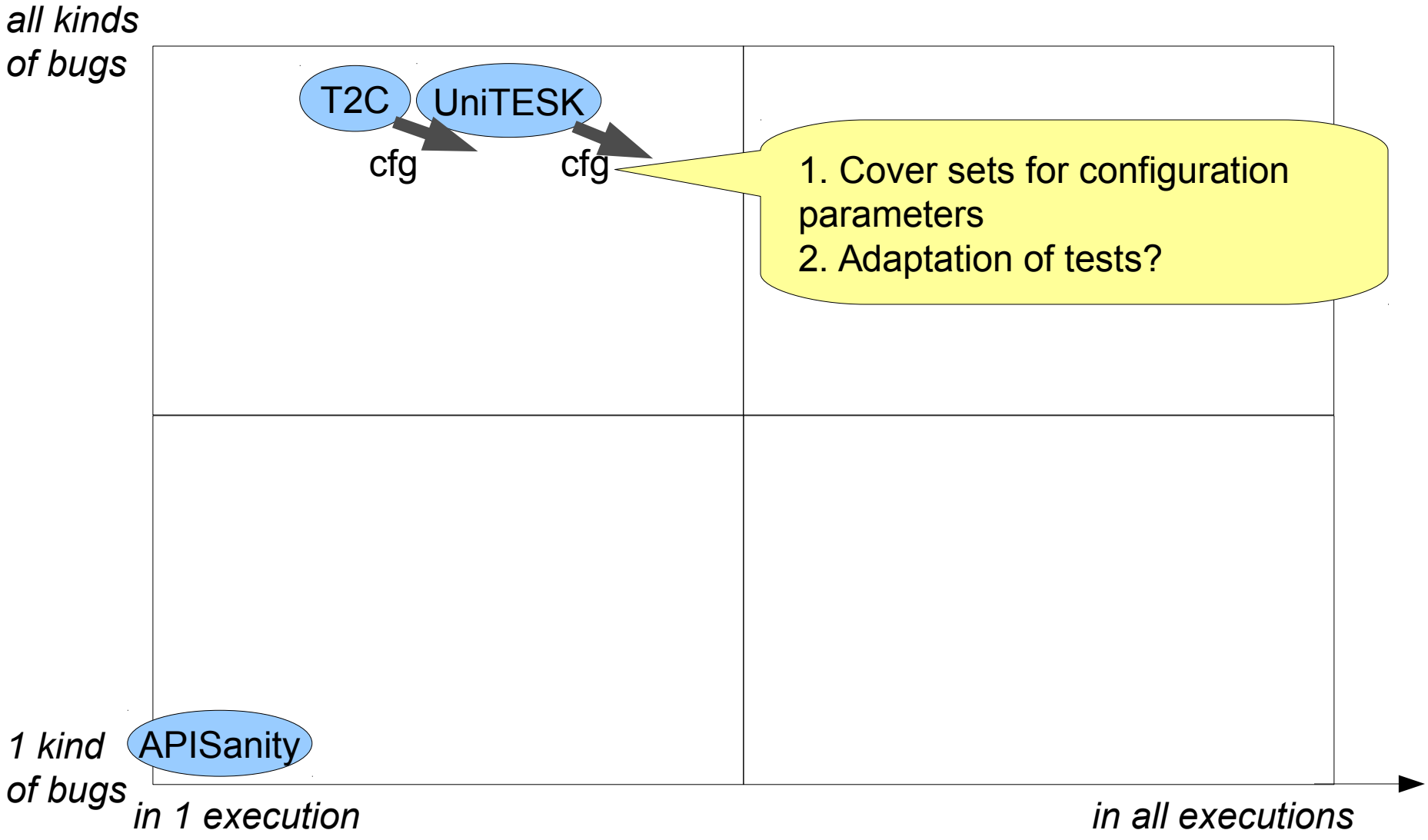
ISP RAS

Institute for System Programming of the Russian Academy of Sciences

№	Platform	NP	Params	C	N1	N2	T1	T2
1	i386/i386	228	$4^2 3^6 2^{220}$	172	25	92	0.01	0.31
2	i386/x86	215	$4^2 3^6 2^{207}$	170	23	72	0.01	0.26
3	mips/bt205	373	$17^2 15^1 8^8 7^2 6^1 5^3 4^{19} 3^{21} 2^{316}$	279	322	670	1.57	5.07
4	mips/ bt23-202	331	$17^2 15^1 13^4 9^4 7^2 5^3 4^7 3^{10} 2^{298}$	267	319	653	1.07	4.58
5	mips64/ bt128	548	$17^2 15^1 5^3 4^1 3^{15} 2^{526}$	500	289	556	0.57	7.88
6	mips64/ bt211	334	$17^2 15^1 13^4 9^4 7^2 5^3 4^5 3^{10} 2^{302}$	277	319	726	0.78	4.08
7	mips64/ bt206	334	$17^2 15^1 7^2 6^1 5^3 4^5 3^{18} 2^{302}$	278	289	651	0.58	2.99
8	mips64 /mpon	301	$17^2 15^1 5^3 4^1 3^{15} 2^{279}$	251	289	574	0.33	2.15
9	mips64/ vmips	241	$5^1 4^1 3^6 2^{233}$	203	27	89	0.01	0.42
10	mips64/ cprio64	563	$17^2 15^1 5^3 4^1 3^{15} 2^{541}$	511	289	564	0.58	6.70
11	komdiv64/ bt128	548	$17^2 15^1 5^3 4^1 3^{15} 2^{526}$	500	289	556	0.56	7.80
12	R4000/bt128	419	$17^2 15^1 5^3 4^2 3^{14} 2^{397}$	377	289	541	0.44	7.00
13	R4000/bt206	220	$17^2 15^1 14^{14} 7^2 6^1 5^3 4^6 3^{17} 2^{174}$	155	823	826	1.28	1.65
14	R4000/mpon	183	$17^2 15^1 5^3 4^2 3^{14} 2^{161}$	138	289	543	0.23	0.56
15	R4000/ vmips	113	$5^1 4^2 3^5 2^{105}$	80	28	62	0.01	0.07
16	R4000/ cprio64	434	$17^2 15^1 5^3 4^2 3^{14} 2^{412}$	388	289	549	0.46	5.59

V.V. Kuliamin. Combinatorial generation of operation system software configurations. Proceedings of the Institute for System Programming Volume 23. 2012 y. pp. 359-370.

OS Verification



Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest	Cfg
				-
Kinds of Observable Events				
interface events	+	+	+	
internal events				
Events Collection				
internal	+	+	+	
external				
embedded				
Requirements Specification				
in-place (local, tabular)	+		+	If
formal model (pre/post+invariants,...)		+		If
assertions/prohibited events	External	External	External	Co
Events Analysis				
online	+	+	+	
in-place	+		+	
outside		+		
offline				

Test Aspects (2)

ISP RAS

Active Aspects	T2C	OLVER	Autotest	Cfg
				+-
Target Test Situations Set				cfgs
requirements coverage	+	+		
class equivalence coverage		+		
model coverage (SUT/reqs)		+		
source code coverage				
Test Situations Setup/Set Gen				
passive				
fixed scenario	+		+	
manual	+			
pre-generated				
coverage driven				+-
random			+-	
adapting scenario		+		
coverage driven		+		
source code coverage				
model/... coverage		+		
random				
Test Actions				
application interface	+	+	+	
HW interface				
internal actions				
inside				
outside				

LinuxTesting.org

Robustness Testing

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Fault Handling Code

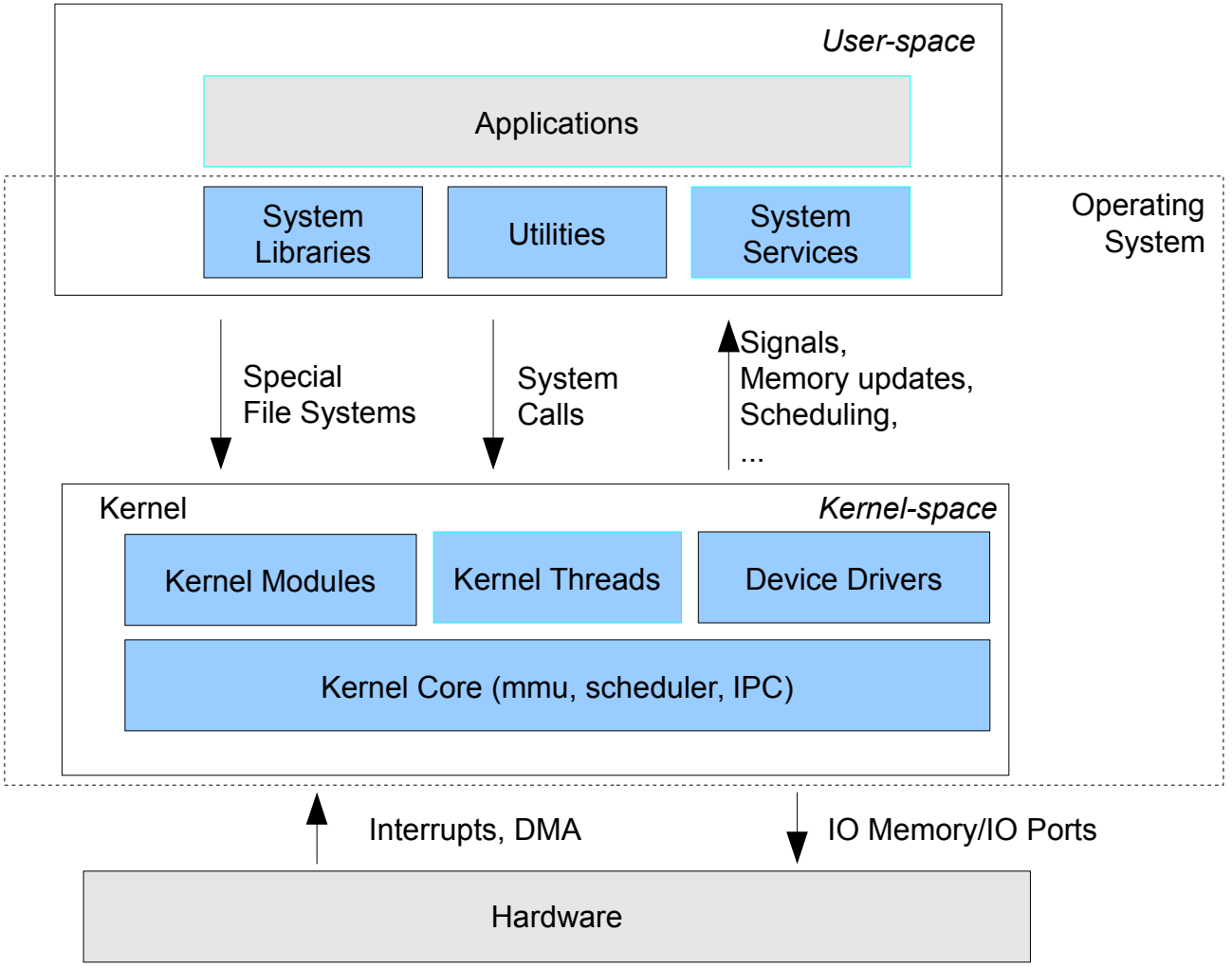
- Is not so fun
- Is really hard to keep all details in mind
- Practically is not tested
- Is hard to test even if you want to
- **Bugs seldom(never) occurs**

  => low pressure to care

Why do we care?

- It beats someone time to time
- Safety critical systems
- Certification authorities

Operating Systems



Run-Time Testing of Fault Handling

- Manually targeted test cases
  - \+ The highest quality
  - – Expensive to develop and to maintain
  - – Not scalable
- Random fault injection on top of existing tests
  - \+ Cheap
  - – Oracle problem
  - – No any guarantee
  - – When to finish?

Systematic Approach

- Hypothesis:
  - Existing test lead to more-or-less deterministic control flow in kernel code
- Idea:
  - Execute existing tests and collect all potential fault points in kernel code
  - Systematically enumerate the points and inject faults there

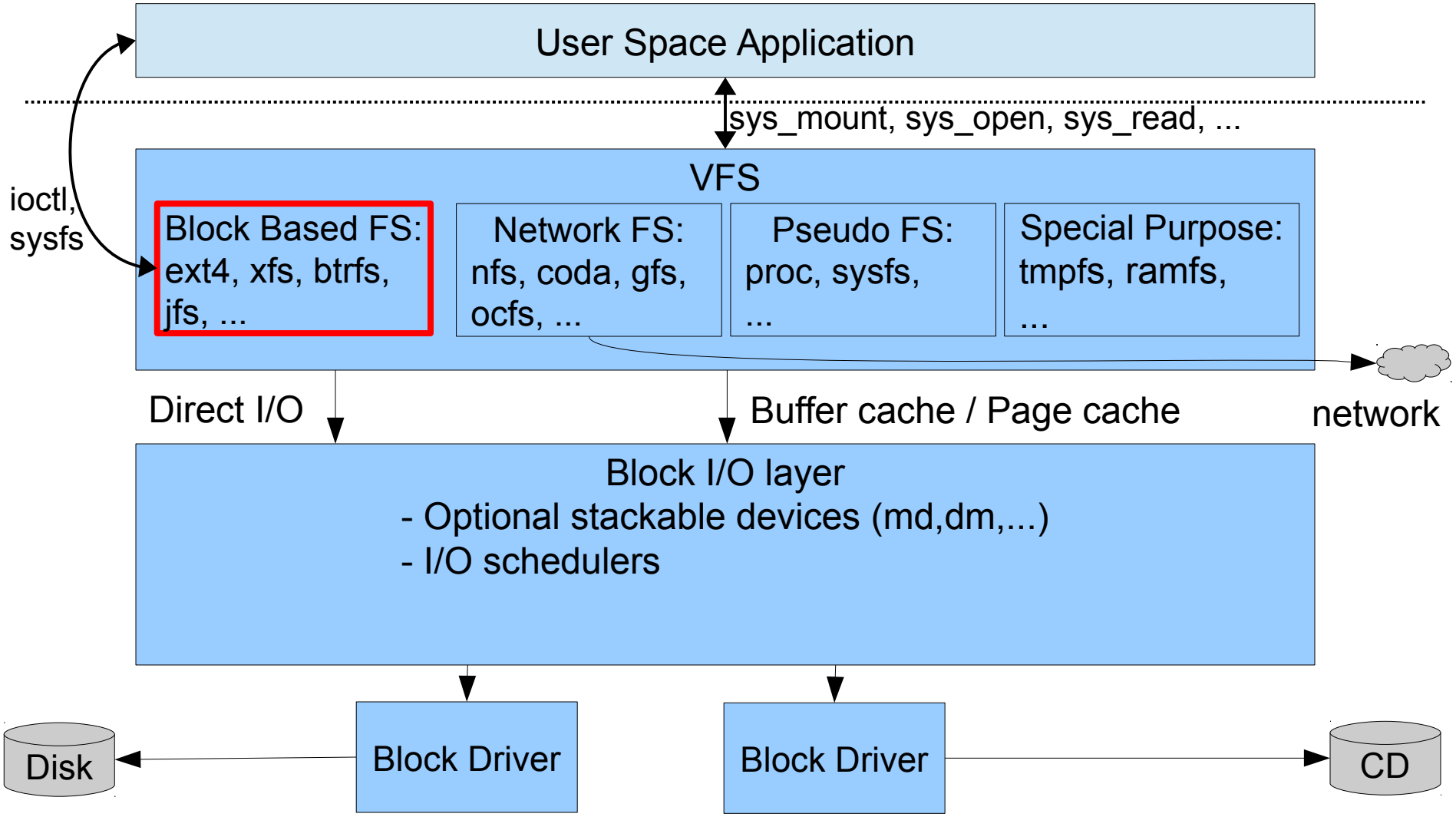
Experiments – Outline

- Target code
- Fault injection implementation
- Methodology
- Results

Experiments – Target

- Target code: file system drivers
- Reasons:
  - Failure handling is more important than in average
    - Potential data loss, etc.
  - Same tests for many drivers
  - It does not require specific hardware
  - Complex enough

Linux File System Layers



File System Drivers - Size

File System Driver	Size, LoC
JFS	18 KLOC
Ext4	37 KLoC *with jbd2*
XFS	69 KLoC
BTRFS	82 KLoC
F2FS	12 KLoC

File System Driver – VFS Interface

- file_system_type
- super_operations
- export_operations
- inode_operations
- file_operations
- vm_operations
- address_space_operations
- dquot_operations
- quotactl_ops
- dentry_operations

~100 interfaces in total

FS Driver – Userspace Interface

File System Driver	ioctl	sysfs
JFS	6	-
Ext4	14	13
XFS	48	-
BTRFS	57	-

FS Driver – Partition Options

File System Driver	mount options	mkfs options
JFS	12	6
Ext4	50	~30
XFS	37	~30
BTRFS	36	8

FS Driver – On-Disk State

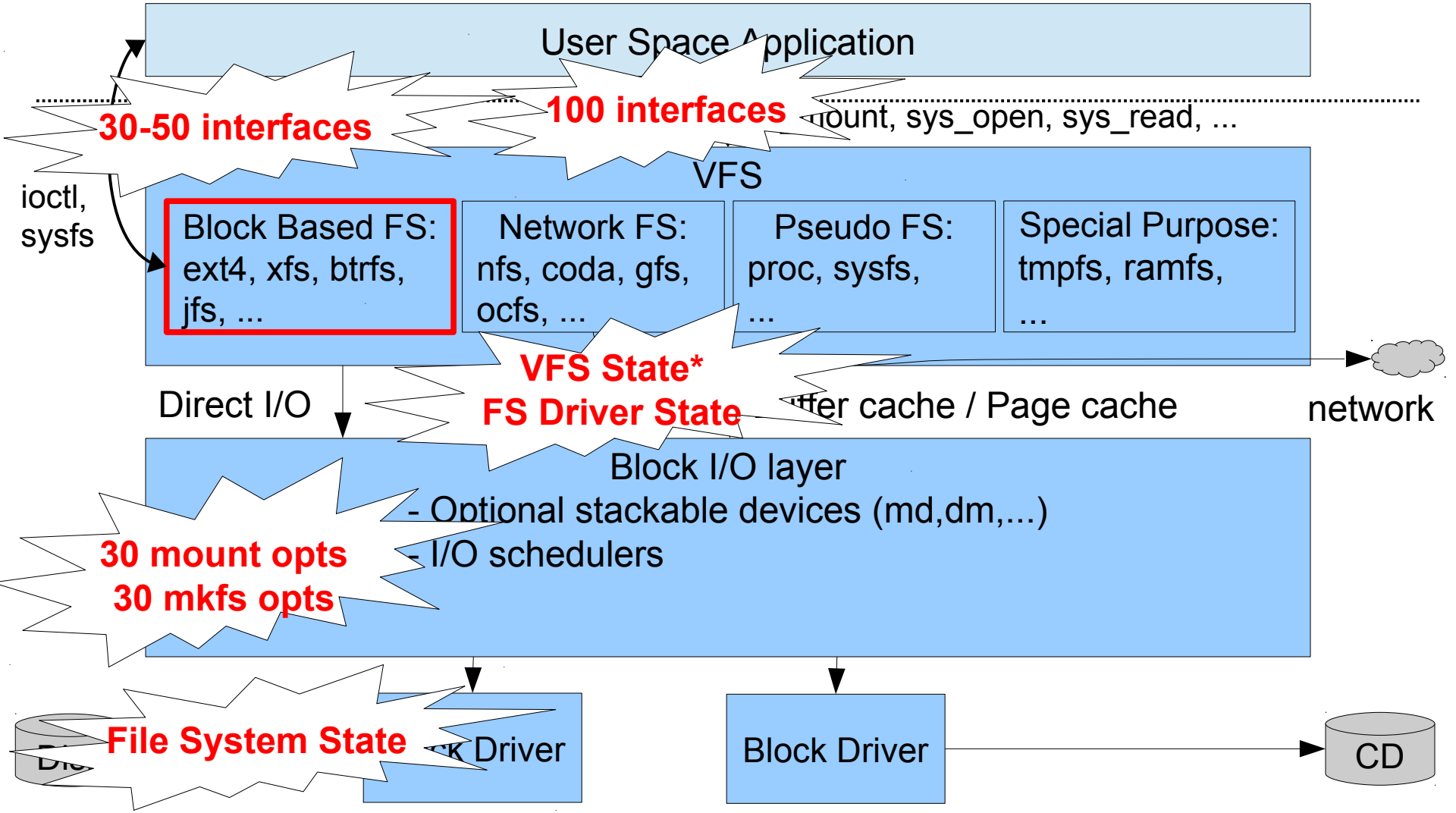
File System Hierarchy

* File Size
* File Attributes
* File Fragmentation
* File Content (holes,...)

FS Driver – In-Memory State

- Page Cache State
- Buffers State
- Delayed Allocation
- ...

Linux File System Layers



FS Driver – Fault Handling

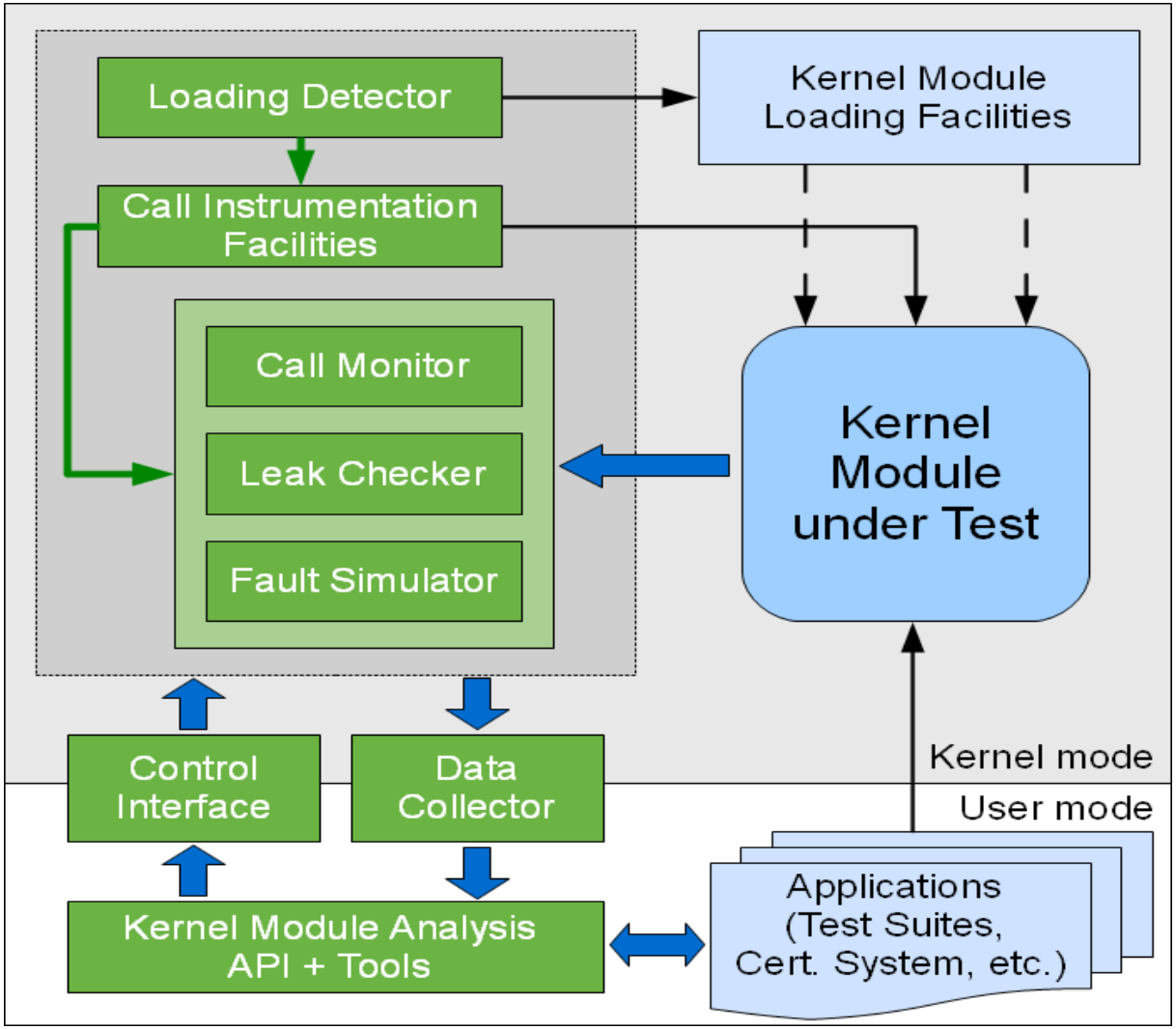
- Memory Allocation Failures
- Disk Space Allocation Failures
- Read/Write Operation Failures

Fault Injection - Implementation

- Based on KEDR framework*
  - intercept requests for memory allocation/bio requests
    - to collect information about potential fault points
    - to inject faults
  - also used to detect memory/resources leaks

(*) http://linuxtesting.org/project/kedr

KEDR Workflow



http://linuxtesting.org/project/kedr

Experiments – Oracle Problem

- Assertions in tests are disabled
- Kernel oops/bugs detection
- Kernel assertions, lockdep, memcheck, etc.
- Kernel sanitizers
- KEDR Leak Checker

Methodology – The Problem

- Source code coverage is used to measure results on fault injection
- If kernel crashes code, coverage results are unreliable

Methodology – The Problem

- Source code coverage is used to measure results on fault injection
- If kernel crashes code, coverage results are unreliable
- As a result
  - Ext4 was analyzed only
  - XFS, BTRF, JFS, F2FS, UbiFS, JFFS2 crashes and it is too labor and time consuming to collect reliable data

Experiment Results

Systematic vs. Random

	Increment *new lines*	**Time** *min*	**Cost** *second/line*
Xfstests without fault simulation	-	2	-
Xfstests+random(p=0.005,repeat=200)	411	182	27
Xfstests+random(p=0.01,repeat=200)	380	152	24
Xfstests+random(p=0.02,repeat=200)	373	116	19
Xfstests+random(p=0.05,repeat=200)	312	82	16
Xfstests+random(p=0.01,repeat=400)	451	350	47
Xfstests+stack filter	423	90	13
Xfstests+stackset filter	451	237	31

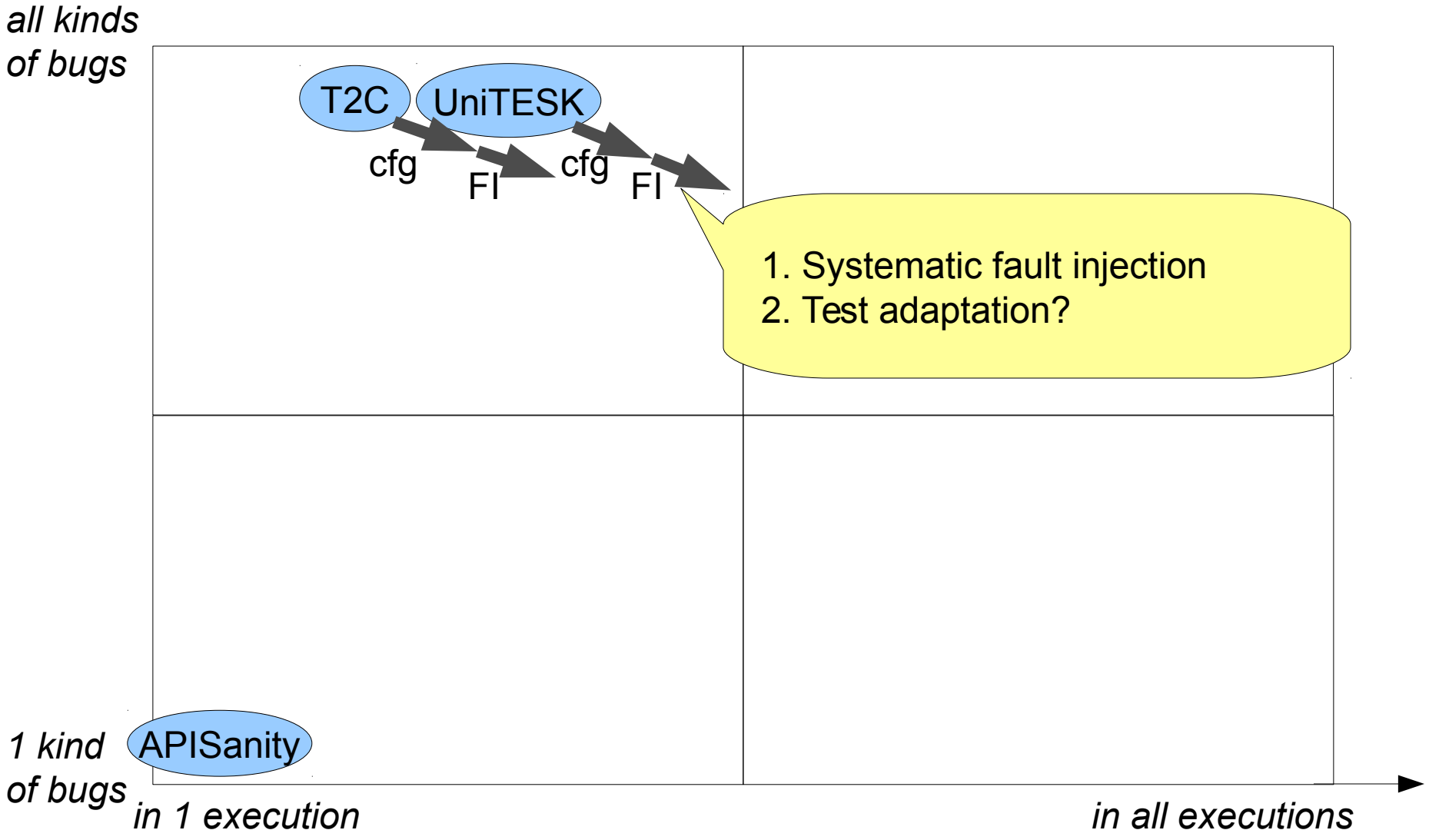
Systematic vs. Random

+ 2 times more cost effective
+ Repeatable results
– Requires more complex engine

+ Cover double faults
– Unpredictable
– Nondeterministic

No.	Type	Brief	Added on	Accepted	Status
F0011	Crash	ext4: When mounted with backup superblock online resize leads to BUG_ON or causes filesystem corruption	2014-12-27	http://www.spinics.net/lists/linux-ext4/msg46743.html commit	Fixed in kernel 3.19-rc4
F0010	Crash	f2fs: Possible use-after-free when umount filesystem	2014-07-25	https://lkml.org/lkml/2014/7/21/198 commit	Fixed in kernel 3.17-rc1
F0009	Crash	ext4: Destruction of ext4_groupinfo_caches during one mount causes BUG_ON for other mounted ext4 filesystems	2014-05-12	https://lkml.org/lkml/2014/5/12/147 commit	Fixed in kernel 3.16-rc1
F0008	Crash	f2fs: BUG_ON() is triggered in recover_inode_page() when mount valid f2fs filesystem	2014-04-18	https://lkml.org/lkml/2014/4/14/189 commit	Fixed in kernel 3.17-rc1
F0007	Crash	f2fs: f2fs unmount hangs if f2fs_init_acl() fails during mkdir syscall	2014-02-17	https://lkml.org/lkml/2014/2/6/18 commit	Fixed in kernel 3.15-rc1
F0006	Deadlock	f2fs: a deadlock in mkdir if ACL is enabled	2013-10-28	https://lkml.org/lkml/2013/10/26/163 commit	Fixed in kernel 3.12-rc3
F0005	Crash	ext4: system hangs after failure in ext4_mb_new_preallocation()	2013-07-01	https://lkml.org/lkml/2013/5/5/64 commit	Fixed in kernel 3.10-rc3
F0004	Deadlock	ext4: deadlocks after allocation failure in ext4_init_io_end()	2013-06-04	https://lkml.org/lkml/2013/5/13/426 commit	Fixed in kernel 3.10-rc3
F0003	Crash	jfs: Several bugs in jfs_freeze() and jfs_unfreeze()	2013-05-24	https://lkml.org/lkml/2013/5/24/76 commit	Fixed in kernel 3.10-rc3
F0002	Crash	ext4: NULL dereference in ext4_calculate_overhead()	2012-11-28	https://lkml.org/lkml/2012/11/28/354 commit	Fixed in kernel 3.8-rc1
F0001	Crash	ext4: NULL pointer dereference in mount_fs() because of ext4_fill_super() wrongly reports	2012-11-08	https://bugzilla.kernel.org/show_bug.cgi?id=48431	Fixed in kernel

OS Verification



Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC
				-	-	
Kinds of Observable Events						
interface events	+	+	+			
internal events						+
Events Collection						
internal	+	+	+			+
external						
embedded						
Requirements Specification						Specific
in-place (local, tabular)	+		+	If	Dis	
formal model (pre/post+invariants,...)		+		If	Co	
assertions/prohibited events	External	External	External	Co	Co	
Events Analysis						
online	+	+	+			
in-place	+		+			+
outside		+				
offline						

Test Aspects (2)

ISP RAS

Active Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC
				+-	+	-
Target Test Situations Set				cfgs		
requirements coverage	+	+				
class equivalence coverage		+				
model coverage (SUT/reqs)		+				
source code coverage					almost	
Test Situations Setup/Set Gen						
passive						
fixed scenario	+		+			
manual	+					
pre-generated						
coverage driven				+-		
random			+-			
adapting scenario		+				
coverage driven		+				
source code coverage					almost	
model/... coverage		+				
random					as option	
Test Actions						
application interface	+	+	+			
HW interface						
internal actions					+	
inside					+	
outside						

LinuxTesting.org

Concolic Testing (S2E)

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Concolic Testing

- Concolic = Symbolic + Concrete
  - SUT runs in concrete and in symbolic modes
  - Symbolic execution is used to collect conditions and branches of the current path
  - Collected data is used to generate new input data to cover more execution paths

Concolic Tools

Tool	Language	Platform	Constraint Solver
DART	C	NA	lp_solver
SMART	C	Linux	lp_solver
CUTE	C	Linux	lp_solver
CREST	**C**	**Linux**	**Yices**
EXE	C	Linux	STP
KLEE	**C (LLVM bitcode)**	**Linux**	**STP**
Rwset	C	Linux	STP
PathCrawler	C	NA	NA
SAGE	Machine code	Windows	Disolver

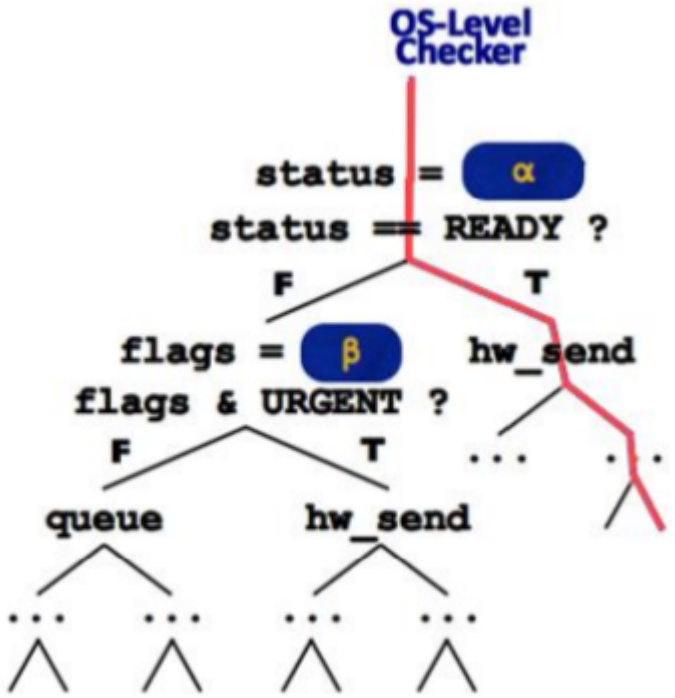
S2E

- based on KLEE
- uses patched Qemu
  - source code is not required
- supports plugins

(*) https://s2e.epfl.ch/

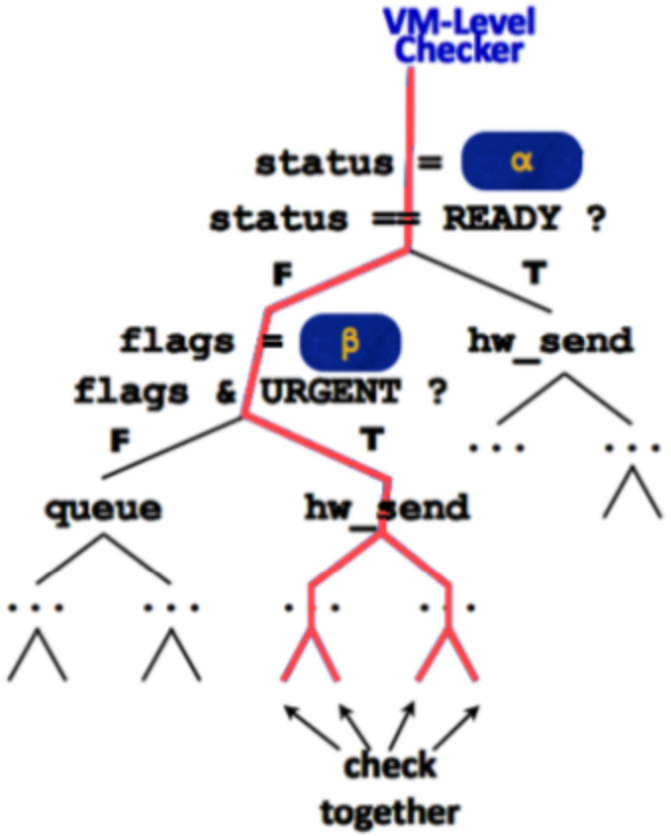
S2E – OS Level Checks

- lockdep
- asserts
- memleak
- sanitizers



S2E – VM Level Checks

- many paths
- requires plugins

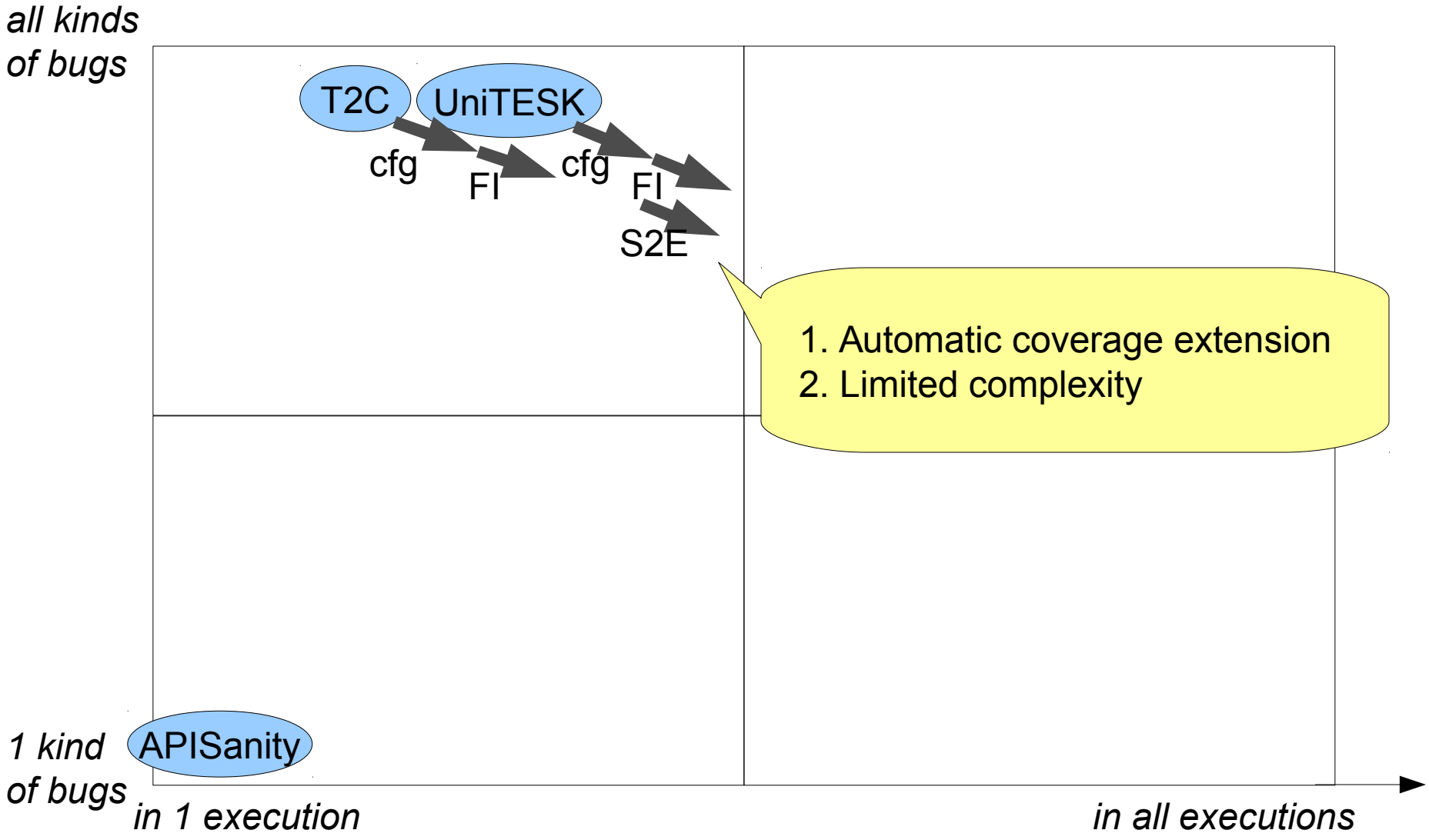


S2E for Linux File Systems

- Instrumentation FS driver with SystemTap
- Mark metadata readed from disk as symbolic

```
static inline void s2e_make_symbolic(void *buf, int size, char *name){
__asm__ __volatile__(
".byte 0x0f, 0x3f\n"
".byte 0x00, 0x03, 0x00, 0x00\n"
".byte 0x00, 0x00, 0x00, 0x00\n"
: : "a" (buf), "b" (size), "c" (name) );
}

probe vfs.read, vfs.write
{
s2e_make_symbolic(
&($file->f_path->dentry->d_inode->i_count),
4,
"i_count")
}
```

OS Verification



Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E
				-	-	+	+-
Kinds of Observable Events							
interface events	+	+	+				
internal events						+	+
Events Collection							
internal	+	+	+			+	
external							+
embedded							
Requirements Specification						Specific	Plugin
in-place (local, tabular)	+		+	If	Dis		Dis
formal model (pre/post+invariants,...)		+		If	Co		Co
assertions/prohibited events	External	External	External	Co	Co		Co
Events Analysis							
online	+	+	+				
in-place	+		+			+	
outside		+					
offline							

Test Aspects (2)

Active Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E
				+-	+	-	+
Target Test Situations Set				cfgs			
requirements coverage	+	+					
class equivalence coverage		+					
model coverage (SUT/reqs)		+					
source code coverage					almost		+
Test Situations Setup/Set Gen							
passive							
fixed scenario	+		+				
manual	+						
pre-generated							
coverage driven				+-			
random			+-				
adapting scenario		+					
coverage driven		+					
source code coverage					almost		+
model/... coverage		+					
random					as option		
Test Actions							
application interface	+	+	+				
HW interface							
internal actions					+		+
inside					+		
outside							+

LinuxTesting.org

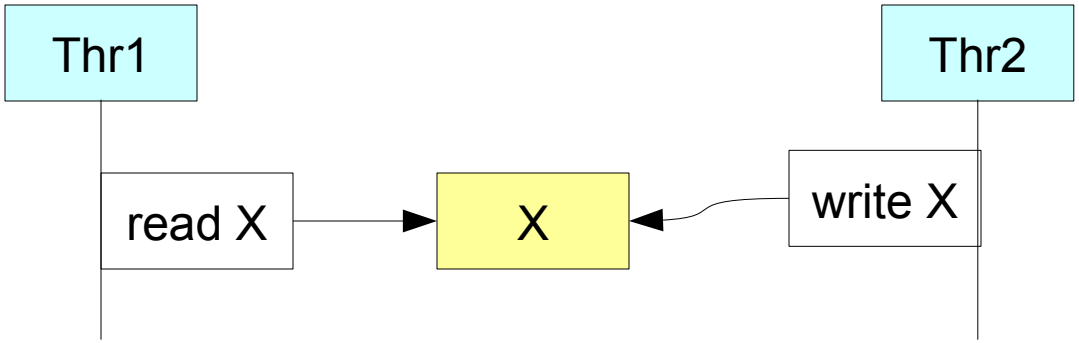
Race Hound

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

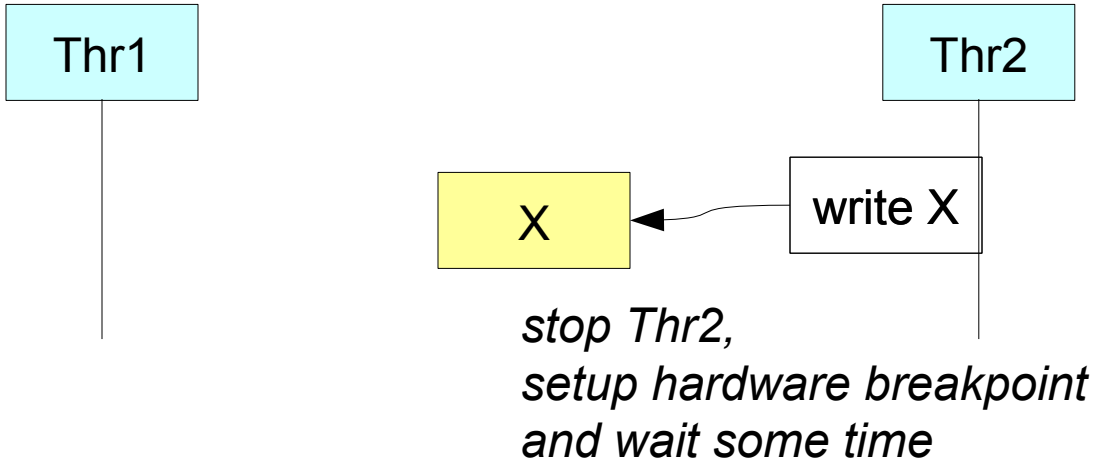
Data race

- Data race is a situation if two threads access one location in memory
  - at least one of them is a write
  - they are not both synchronization accesses
  - can be executed on a multiprocessor in such a way that two conflicting memory accesses are performed simultaneously



Race Hound

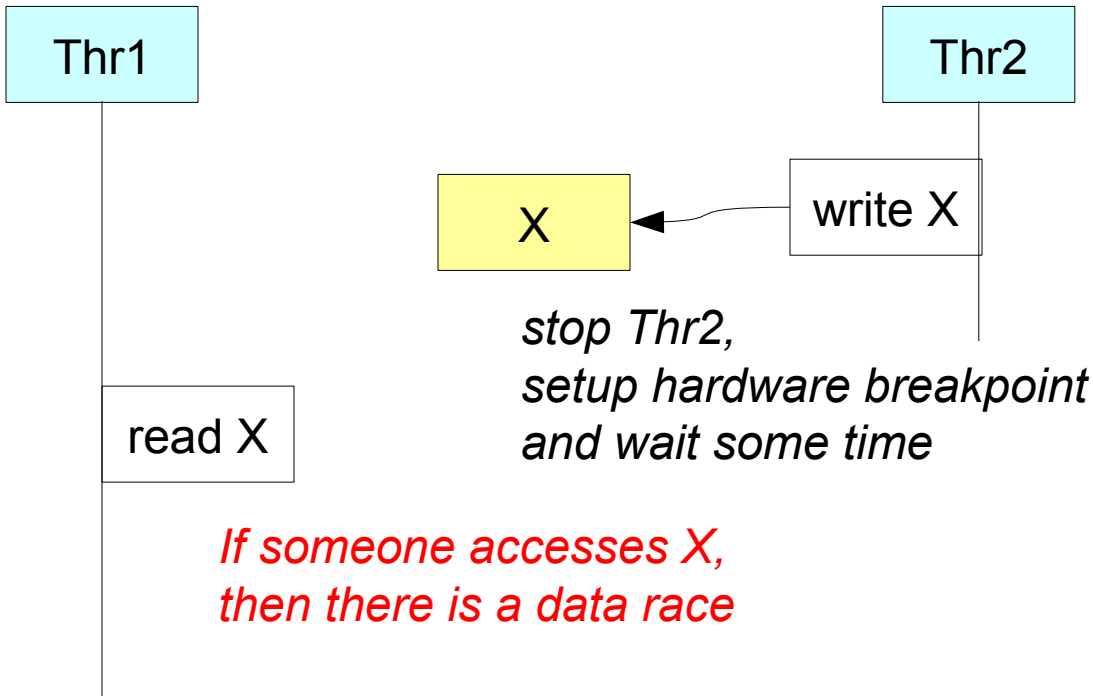
- idea close to DataCollider (Microsoft Research)



https://forge.ispras.ru/projects/race-hound

Race Hound

- idea close to DataCollider (Microsoft Research)



https://forge.ispras.ru/projects/race-hound

Race Hound Results

- works on x86/x86-64 multicore
- Linux kernel >= 3.2
- acknowledged 3 data race in Intel e1000 Linux device driver
- ~10 data race in total

Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH
				-	-	+	+-	+
Kinds of Observable Events								
interface events	+	+	+					
internal events						+	+	+
Events Collection								
internal	+	+	+			+		
external							+	
embedded								
Requirements Specification						Specific	Plugin	Specific
in-place (local, tabular)	+		+	If	Dis		Dis	
formal model (pre/post+invariants,...)		+		If	Co		Co	
assertions/prohibited events	External	External	External	Co	Co		Co	
Events Analysis								
online	+	+	+					
in-place	+		+			+		+
outside		+						
offline								

Test Aspects (1)

Monitoring Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH	KStrider
				-	-	+	+-	+	+-
Kinds of Observable Events									
interface events	+	+	+						
internal events						+	+	+	+
Events Collection									
internal	+	+	+			+			+
external							+		
embedded									
Requirements Specification						Specific	Plugin	Specific	Specific
in-place (local, tabular)	+		+	If	Dis		Dis		
formal model (pre/post+invariants,...)		+		If	Co		Co		
assertions/prohibited events	External	External	External	Co	Co		Co		
Events Analysis									
online	+	+	+						
in-place	+		+			+		+	
outside		+							
offline									+

Test Aspects (2)

ISP RAS

Active Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH
				+-	+	-	+	+
Target Test Situations Set				cfgs				Specific
requirements coverage	+	+						
class equivalence coverage		+						
model coverage (SUT/reqs)		+						
source code coverage					almost		+	
Test Situations Setup/Set Gen								
passive								+-
fixed scenario	+		+					
manual	+							
pre-generated								
coverage driven				+-				
random			+-					
adapting scenario		+						
coverage driven		+						
source code coverage					almost		+	
model/... coverage		+						
random					as option			
Test Actions								
application interface	+	+	+					
HW interface								
internal actions					+		+	+
inside					+			+
outside							+	

LinuxTesting.org

Kernel Strider

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Kernel Strider

- KEDR collects data on all memory access from target module and calls to synchronization primitives
  - annotation of Linux synchronization primitives
  - annotation of nonstandard happens-before dependencies in module code
- ThreadSanitizer in Offline mode is used to find data races

https://forge.ispras.ru/projects/kernel-strider

Test Aspects (1)

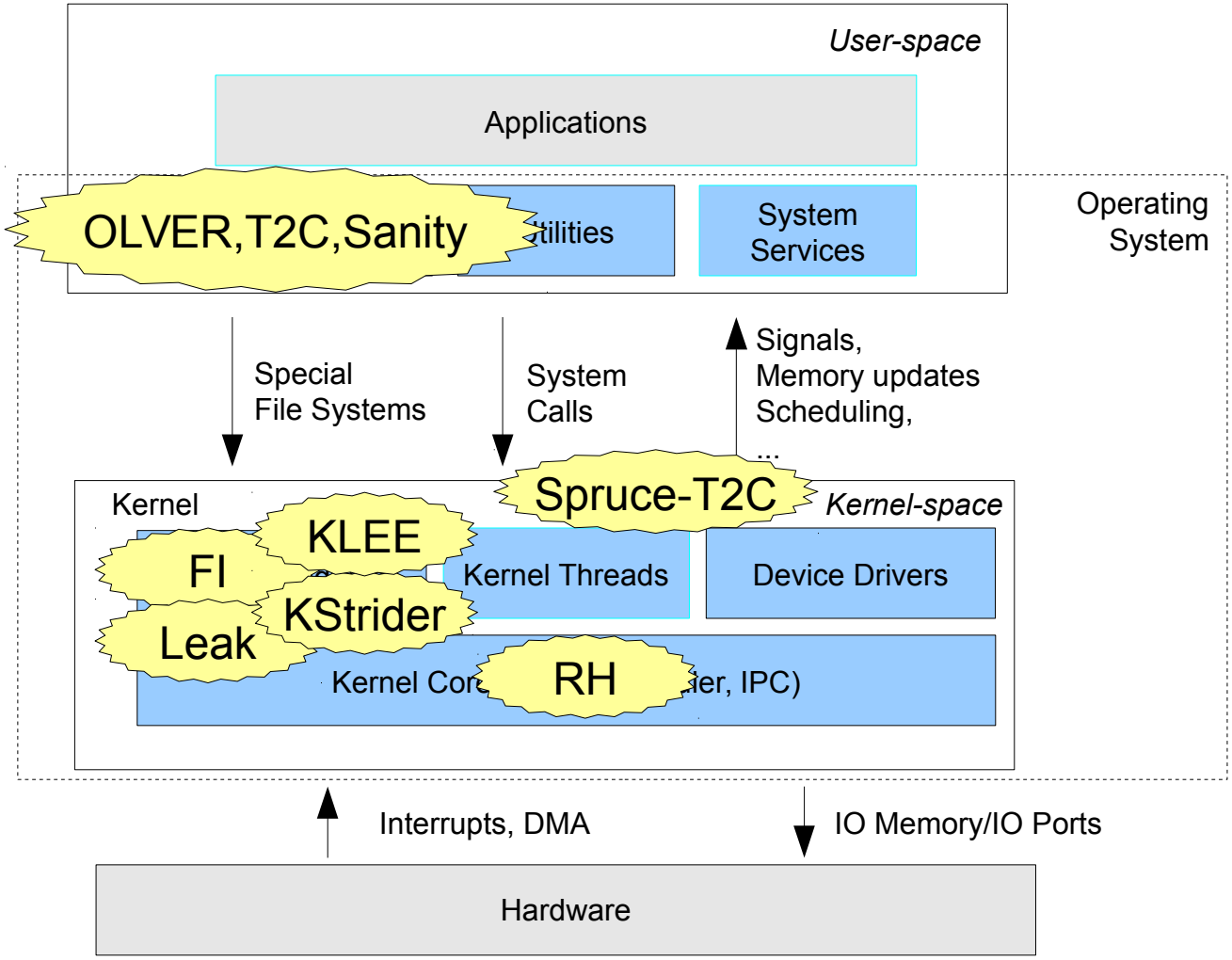
Monitoring Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH	KStrider
				-	-	+	+-	+	+-
Kinds of Observable Events									
interface events	+	+	+						
internal events						+	+	+	+
Events Collection									
internal	+	+	+			+			+
external							+		
embedded									
Requirements Specification						Specific	Plugin	Specific	Specific
in-place (local, tabular)	+		+	If	Dis		Dis		
formal model (pre/post+invariants,...)		+		If	Co		Co		
assertions/prohibited events	External	External	External	Co	Co		Co		
Events Analysis									
online	+	+	+						
in-place	+		+			+		+	
outside		+							
offline									+

Test Aspects (2)

ISP RAS

Active Aspects	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH	KStrider
				+-	+	-	+	+	-
Target Test Situations Set				cfgs				Specific	
requirements coverage	+	+							
class equivalence coverage		+							
model coverage (SUT/reqs)		+							
source code coverage					almost		+		
Test Situations Setup/Set Gen									
passive								+-	
fixed scenario	+		+						
manual	+								
pre-generated									
coverage driven				+-					
random			+-						
adapting scenario		+							
coverage driven		+							
source code coverage					almost		+		
model/... coverage		+							
random					as option				
Test Actions									
application interface	+	+	+						
HW interface									
internal actions					+		+	+	
inside					+			+	
outside							+		

Operating Systems



LinuxTesting.org

SVACE – Static Analysis

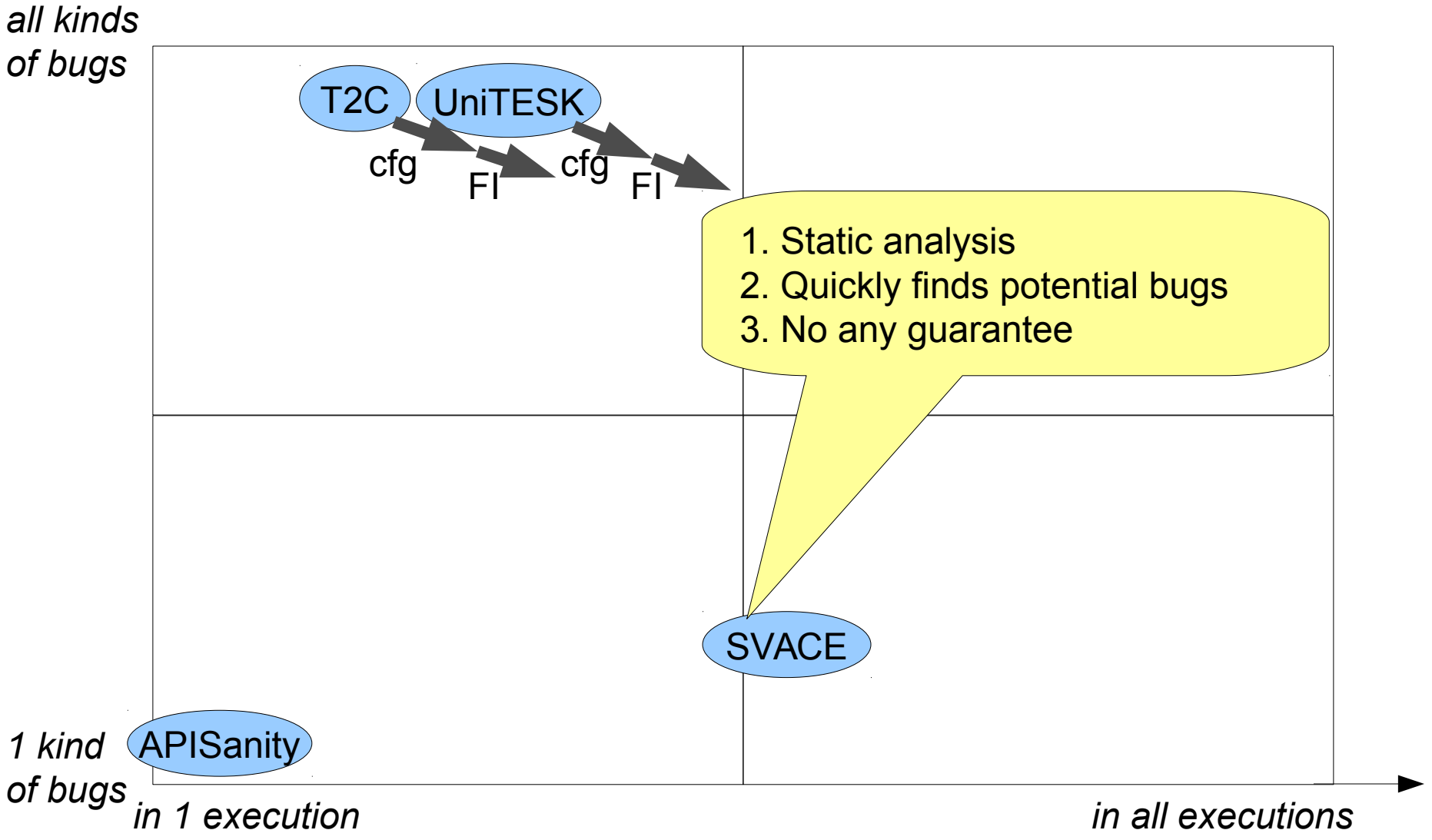
ISP RAS

Institute for System Programming of the Russian Academy of Sciences

SVACE

- Static analysis of C/C++/Java code, Linux/Windows
- 150+ kinds of defects
  - Buffer overflows, NULL-pointer dereferences
  - Memory management, tainted input
  - Concurrency issues
  - Lightweight analysis of semantic patterns
- Eclipse plugin or WebUI

OS Verification



LinuxTesting.org

Linux Driver Verification

ISP RAS

Institute for System Programming of the Russian Academy of Sciences

Commit Analysis(*)

- All patches in stable trees (2.6.35 – 3.0) for 1 year:
  - 26 Oct 2010 – 26 Oct 2011
- 3101 patches overall

(*) Khoroshilov A.V., Mutilin V.S., Novikov E.M. Analysis of typical faults in Linux operating system drivers. Proceedings of the Institute for System Programming of RAS, volume 22, 2012, pp. 349-374. (In Russian)
http://ispras.ru/ru/proceedings/docs/2012/22/isp_22_2012_349.pdf
Raw data: http://linuxtesting.org/downloads/ldv-commits-analysis-2012.zip

Commit Analysis

- All patches in stable trees (2.6.35 – 3.0) for 1 year:
  - 26 Oct 2010 – 26 Oct 2011
- 3101 patches overall

Unique commits to drivers (1503 ~ **50%**)	
Support of a new functionality (321 ~ **20%**)	Bug fixes (1182 ~ **80%**)

Commit Analysis

- All patches in stable trees (2.6.35 – 3.0) for 1 year:
  - 26 Oct 2010 – 26 Oct 2011
- 3101 patches overall

Typical bug fixes (349 ~ **30%**)		
Generic bug fixes (102 ~ **30%**)	Fixes of Linux kernel API misuse (176 ~ **50%**)	Fixes of data races, deadlocks (71 ~ **20%**)

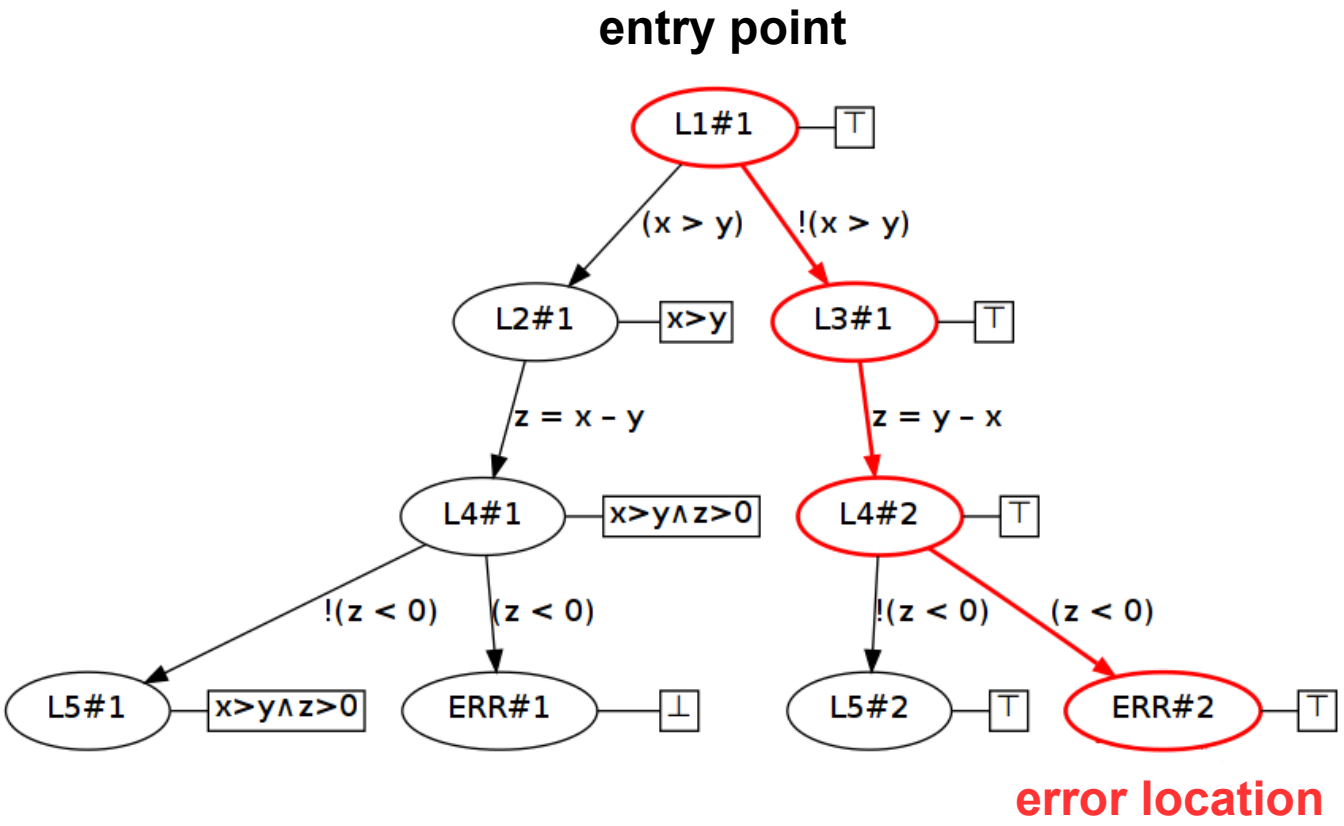
Taxonomy of Typical Bugs

ISP RAS

Rule classes	Types	Number of bug fixes	Percents	Cumulative total percents
Correct usage of the Linux kernel API (176 ~ 50%)	Alloc/free resources	32	~18%	~18%
	Check parameters	25	~14%	~32%
	Work in atomic context	19	~11%	~43%
	Uninitialized resources	17	~10%	~53%
	Synchronization primitives in one thread	12	~7%	~60%
	Style	10	~6%	~65%
	Network subsystem	10	~6%	~71%
	USB subsystem	9	~5%	~76%
	Check return values	7	~4%	~80%
	DMA subsystem	4	~2%	~82%
	Core driver model	4	~2%	~85%
	Miscellaneous	27	~15%	100%
Generic (102 ~ 30%)	NULL pointer dereferences	31	~30%	~30%
	Alloc/free memory	24	~24%	~54%
	Syntax	14	~14%	~68%
	Integer overflows	8	~8%	~76%
	Buffer overflows	8	~8%	~83%
	Uninitialized memory	6	~6%	~89%
	Miscellaneous	11	~11%	100%
Synchronization (71 ~ 20%)	Races	60	~85%	~85%
	Deadlocks	11	~15%	100%

Software Model Checking

- Reachability problem



Verification Tools World

```
int main(int argc,char* argv[])
{
  ...
 other_func(var);
  ...
}
```

```
void other_func(int v)
 {
    ...
    assert( x != NULL);
 }
```

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
    .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    dev->netdev_ops = &usbpn_ops;
    err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
    .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

No explicit calls to init/exit procedures

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
      .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
      dev->netdev_ops = &usbpn_ops;
      err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
      .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

Callback interface procedures registration

No explicit calls to init/exit procedures

Device Driver World

```
int usbpn_open(struct net_device *dev) { ... };
int usbpn_close(struct net_device *dev) { ... };
struct net_device_ops usbpn_ops = {
    .ndo_open = usbpn_open, .ndo_stop = usbpn_close
};
int usbpn_probe(struct usb_interface *intf, const struct usb_device_id *id){
    dev->netdev_ops = &usbpn_ops;
    err = register_netdev(dev);
}
void usbpn_disconnect(struct usb_interface *intf){...}

struct usb_driver usbpn_struct = {
    .probe = usbpn_probe, .disconnect = usbpn_disconnect,
};
int __init usbpn_init(void){ return usb_register(&usbpn_struct);}
void __exit usbpn_exit(void){usb_deregister(&usbpn_struct );}

module_init(usbpn_init);
module_exit(usbpn_exit);
```

Callback interface procedures registration

No explicit calls to init/exit procedures

Driver Environment Model

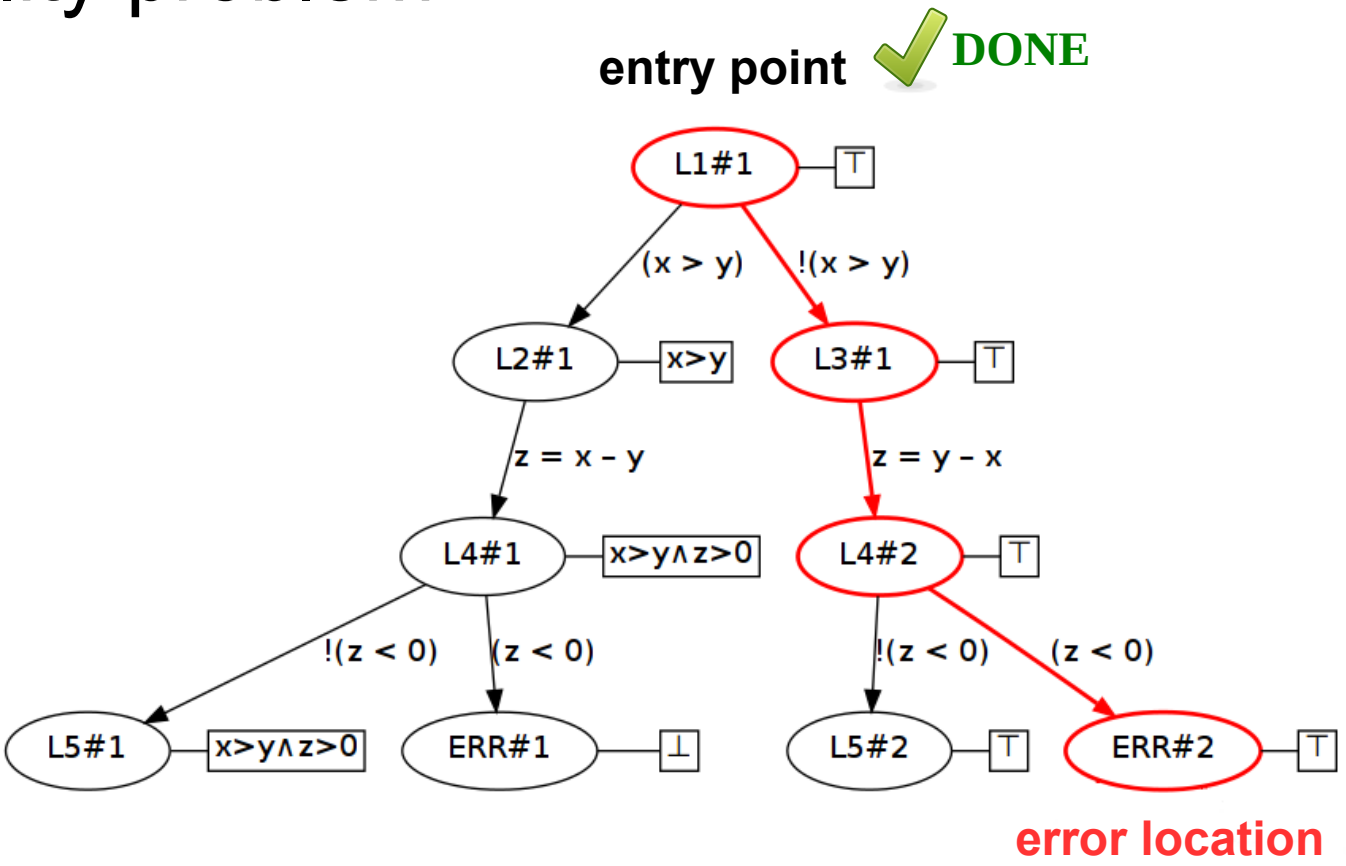
```
int main(int argc,char* argv[])
{
  usbpn_init()
  for(;;) {
    switch(*) {
     case 0: usbpn_probe(*,*,*);break;
     case 1: usbpn_open(*,*);break;
     ...
    }
  }
  usbpn_exit();
}
```

Driver Environment Model (2)

- Order limitation
  - open() after probe(), but before remove()
- Implicit limitations
  - read() only if open() succeed
- and it is specific for each class of drivers

Model Checking and Linux Kernel

- Reachability problem



Error Location?

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);
  ...
  usb_free_urb(x);

  return y;
}
```

Error Location?

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL); // allocate new URB
  ...
  usb_free_urb(x); // deallocate URB: assert(x is NULL or previously allocated URB)

  return y;
}

  …
  // after module exit: assert( all allocated URBs are deallocated)
```

Instrumentation

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);

  ...

  usb_free_urb(x);

  return y;
}
```

```
set URBS = empty;

int f(int y)
{
struct urb *x;

  x = usb_alloc_urb();
  add(URBS, urb);
  ...
  assert(contains(URBS, x));
  usb_free_urb(x);
  remove(URBS, urb);

  return y;
}
  …
  // after module exit
  assert(is_empty(URBS));
```

Aspect-Oriented Notation

```
// Model state: set of allocated URBs
set URBS = empty;

// Model functions
struct urn * ldv_usb_alloc_urb(void)
{
  void *urb;
  urb = ldv_alloc();
  if (urb) {
    add(URBS, urb);
  }
  return urb;
}

void ldv_usb_free_urb(struct urb *urb)
{
  if (urb) {
    remove(URBS, urb);
  }
}
```

```
// Pointcut declarations
pointcut USB_ALLOC_URB:
  call(struct urb *usb_alloc_urb(int, gfp_t));
pointcut USB_FREE_URB:
  call(void usb_free_urb(struct urb *));

// Update model state
around: USB_ALLOC_URB {
  return ldv_usb_alloc_urb();
}
around: USB_FREE_URB {
  ldv_usb_free_urb($arg1);
}

// Assertions
before: USB_FREE_URB {
  assert(contains(URBS, $arg1));
}
after: MODULE_EXIT {
  assert(is_empty(URBS));
}
```

Instrumentation

```
int f(int y)
{
struct urb *x;

  x = usb_alloc_urb(0,GFP_KERNEL);
  ...
  usb_free_urb(x);

  return y;
}
```

```
int f(int y)
{
struct urb *x;

  x = ldv_usb_alloc_urb();
  ...
  assert(contains(URBS, x));
  ldv_usb_free_urb(x);

  return y;
}


  …
  // after module exit
  assert(is_empty(URBS));
```

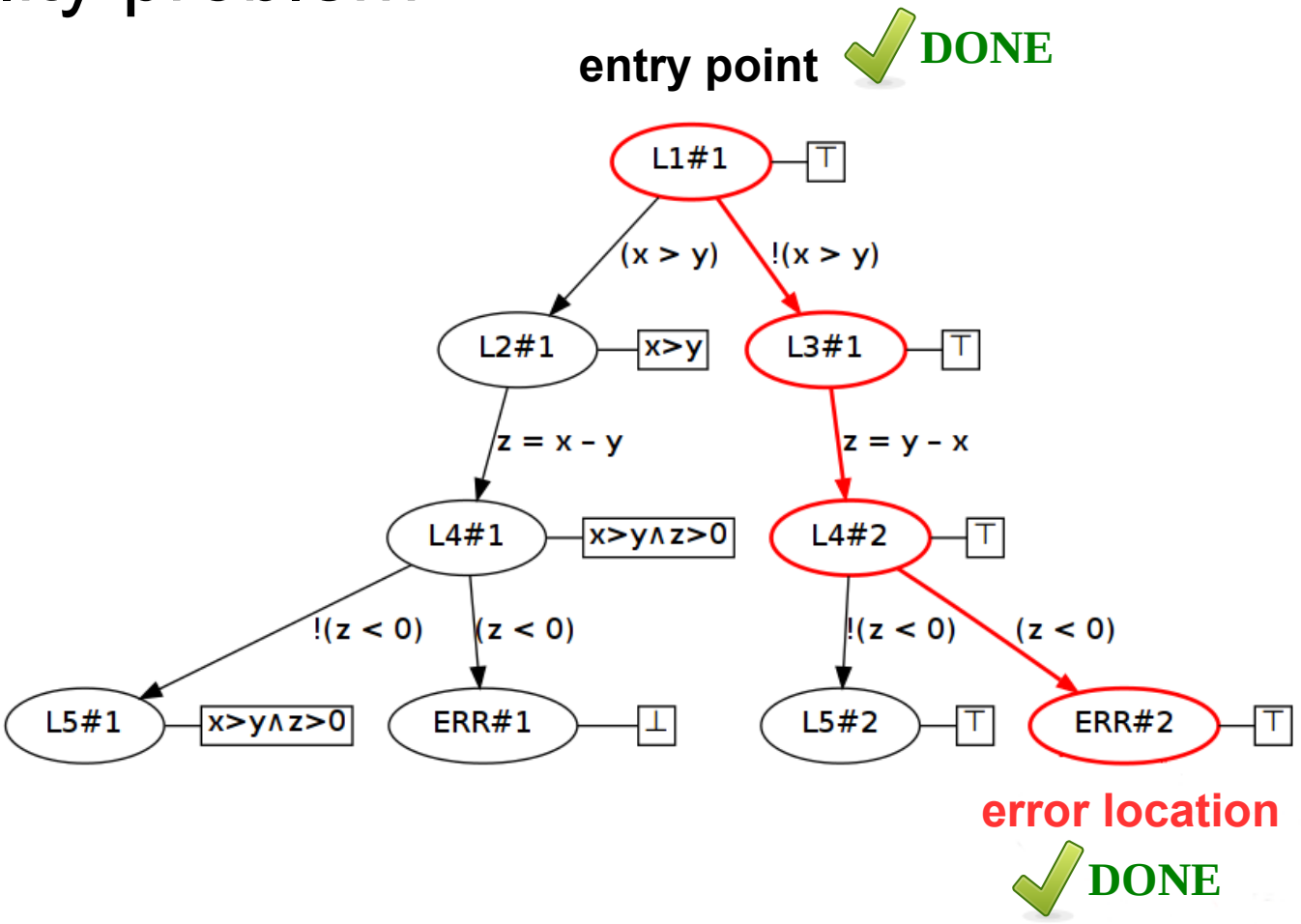
Rule Instrumentor

- **CIF** – C Instrumentation Framework
  - gcc-based aspect-oriented programming tool for C language
  - available under GPLv3:

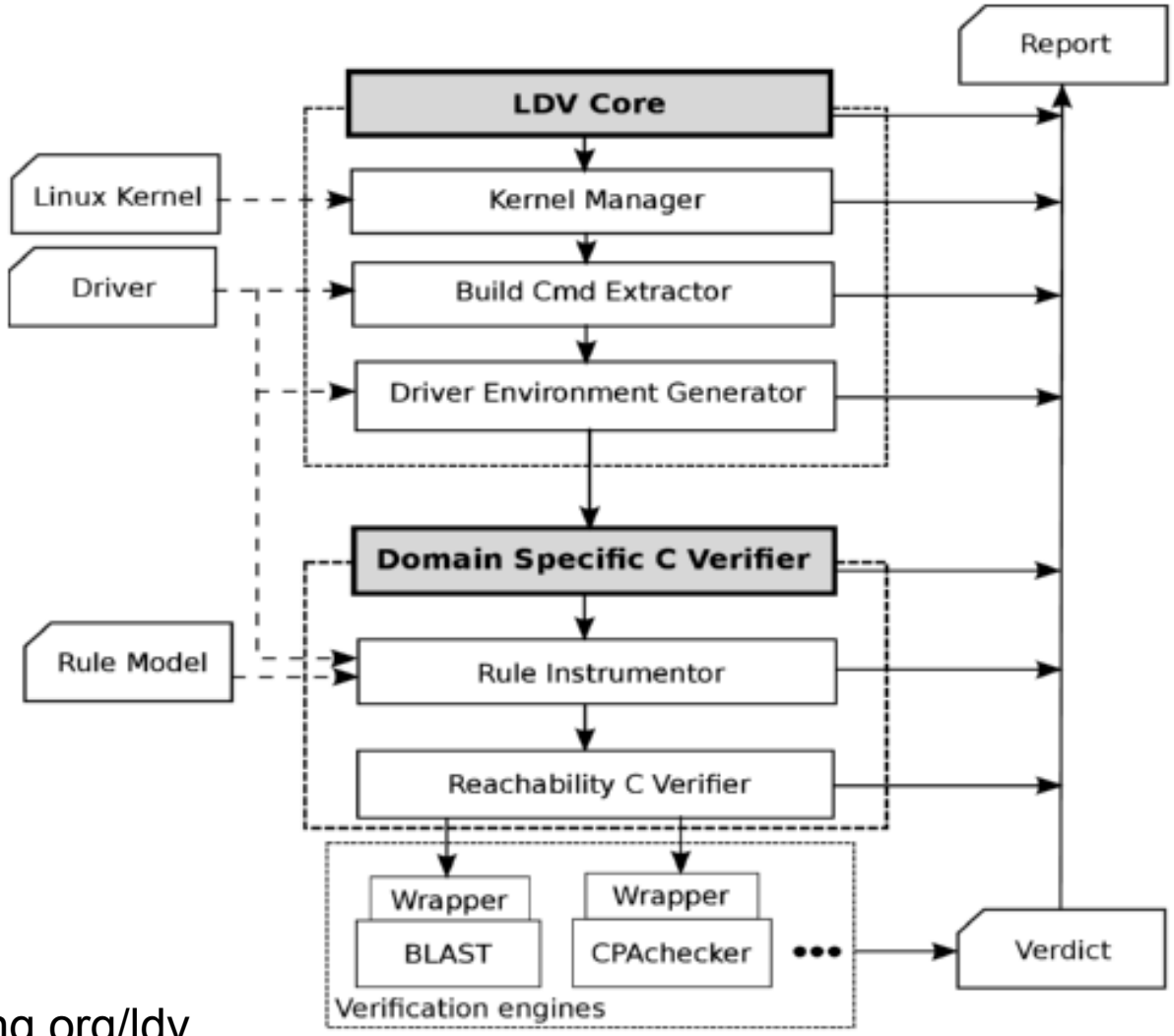
    http://forge.ispras.ru/projects/cif

Model Checking and Linux Kernel

- Reachability problem



Linux Driver Verification



http://linuxtesting.org/ldv

SVCOMP'12 Results

Competition candidate	BLAST 2.7	CPAchecker ABE 1.0.10	CPAchecker Memo 1.0.10	ESBMC 1.17	FShell 1.3	LLBMC 0.9	Predator 20111011	QARMC -HSF	SATabs 3.0	Wolverine 0.5c
Affiliation	Moscow, Russia	Passau, Germany	Paderborn, Germany	Southampton, UK	Vienna, Austria	Karlsruhe, Germany	Brno, Czechia	Munich, Germany	Oxford, UK	Princeton, USA
ControlFlowInteger 93 files, max score: 144	71 9900 s	141 1000 s	140 3200 s	102 4500 s	28 580 s	100 2400 s	17 1100 s	140 4800 s	75 5400 s	39 580 s
DeviceDrivers 59 files, max score: 103	72 30 s	51 97 s	51 93 s	63 160 s	20 3.5 s	80 1.6 s	80 1.9 s	--	71 140 s	68 65 s
DeviceDrivers64 41 files, max score: 66	55 1400 s	26 1900 s	49 500 s	10 870 s	0 0 s	1 110 s	0 0 s	--	32 3200 s	16 1300 s
HeapManipulation 14 files, max score: 24	--	4 16 s	4 16 s	1 220 s	--	17 210 s	20 1.0 s	--	--	--
SystemC 62 files, max score: 87	33 4000 s	45 1100 s	36 450 s	67 760 s	--	8 2.4 s	21 630 s	8 820 s	57 5000 s	36 1900 s
Concurrency 8 files, max score: 11	--	0 0 s	0 0 s	6 270 s	0 0 s	--	0 0 s	--	1 1.4 s	--
Overall 277 files, max score: 435	231 15000 s	267 4100 s	280 4300 s	249 6800 s	48 580 s	206 2700 s	138 1700 s	148 5600 s	236 14000 s	159 3800 s

SVCOMP'14 Results

Competition candidate	BLAST 2.7.2	CBMC	CPAchecker	CPAlien	CSeq-Lazy	CSeq-MU	ESBMC 1.22	FrankenBit	LLBMC	Predator	Symbiotic 2	Threader	UFO	Ultimate Automizer	Ultimate Kojak
Representing Jury Member	Vadim Mutilin	Michael Tautschnig	Stefan Löwe	Petr Muller	Bernd Fischer	Gennaro Parlato	Lucas Cordeiro	Arie Gurfinkel	Stephan Falke	Tomas Vojnar	Jiri Slaby	Corneliu Popeea	Aws Albarghouthi	Matthias Heizmann	Alexander Nutz
Affiliation	Moscow, Russia	London, UK	Passau, Germany	Brno, Czechia	Stellenbosch, South Africa	Southampton, UK	Manaus, Brazil	Pittsburgh, USA	Karlsruhe, Germany	Brno, Czechia	Brno, Czechia	Munich, Germany	Pittsburgh, USA	Freiburg, Germany	Freiburg, Germany
BitVectors 49 tasks, max. score: 86	--	**86** **2 300 s**	**78** **690 s**	--	--	--	77 1 500 s	--	**86** **39 s**	-92 28 s	39 220 s	--	--	--	-23 1 100 s
Concurrency 78 tasks, max. score: 136	--	**128** **29 000 s**	0 0.0 s	--	**136** **1 000 s**	**136** **1 200 s**	32 30 000 s	--	0 0.0 s	0 0.0 s	-82 5.7 s	100 3 000 s	--	--	0 0.0 s
ControlFlow 843 tasks, max. score: 1261	508 32 000 s	397 42 000 s	**1009** **9 000 s**	455 6 500 s	--	--	949 35 000 s	**986** **6 300 s**	**961** **13 000 s**	511 3 400 s	41 39 000 s	--	912 14 000 s	164 6 000 s	214 5 100 s
ControlFlowInteger 181 tasks, max. score: 255	64 7 800 s	-298 35 000 s	179 4 800 s	121 3 400 s	--	--	85 24 000 s	149 5 300 s	74 10 000 s	-28 2 200 s	-151 22 000 s	--	184 9 500 s	33 5 800 s	57 5 000 s
Loops 65 tasks, max. score: 99	25 320 s	99 1 100 s	68 600 s	-16 91 s	--	--	88 3 600 s	76 50 s	95 160 s	27 14 s	26 4.9 s	--	44 44 s	26 170 s	29 150 s
ProductLines 597 tasks, max. score: 929	639 24 000 s	918 6 600 s	928 3 500 s	715 3 100 s	--	--	928 7 500 s	905 950 s	925 2 600 s	929 1 200 s	347 17 000 s	--	927 4 800 s	0 0.0 s	0 0.0 s
DeviceDrivers64 1428 tasks, max. score: 2766	**2682** **13 000 s**	2463 390 000 s	2613 28 000 s	--	--	--	2358 140 000 s	**2639** **3 000 s**	0 0.0 s	50 9.9 s	980 2 200 s	--	**2642** **5 700 s**	--	0 0.0 s
HeapManipulation 80 tasks, max. score: 135	--	**132** **12 000 s**	107 210 s	71 70 s	--	--	97 970 s	--	**107** **130 s**	**111** **9.5 s**	105 15 s	--	--	--	18 35 s
MemorySafety 61 tasks, max. score: 98	--	4 11 000 s	**95** **460 s**	9 690 s	--	--	-136 1 500 s	--	**38** **170 s**	**14** **39 s**	-130 7.5 s	--	--	--	0 0.0 s
Recursive 23 tasks, max. score: 39	--	**30** **11 000 s**	0 0.0 s	--	--	--	-53 4 900 s	--	3 0.38 s	-18 0.12 s	6 0.93 s	--	--	**12** **850 s**	**9** **54 s**
SequentializedConcurrent 261 tasks, max. score: 364	--	**237** **47 000 s**	97 9 200 s	--	--	--	**244** **38 000 s**	--	**208** **11 000 s**	-46 7 700 s	-32 770 s	--	83 4 800 s	49 3 000 s	9 1 200 s
Simple 45 tasks, max. score: 67	30 5 400 s	**66** **15 000 s**	**67** **430 s**	--	--	--	31 27 000 s	37 830 s	0 0.0 s	0 0.0 s	-22 13 s	--	**67** **480 s**	--	0 0.0 s
Overall 2868 tasks, max. score: 4718	--	**3 501** **560 000 s**	**2 987** **48 000 s**	--	--	--	975 280 000 s	--	**1 843** **24 000 s**	-184 11 000 s	-220 42 000 s	--	--	399 10 000 s	139 7 600 s

SVCOMP'15 Results

Competition candidate	AProVE	Beagle	BLAST 2.7.3	Cascade	CBMC	CPAchecker	CPArec	ESBMC 1.24.1	FOREST	Forester	FunctIon	HIPTNT+	Lazy-CSeq	Map2Check	MU-CSeq	Perentie	Predator	SeaHorn	SMACK+Corral	Ultimate Automizer	Ultimate Kojak	Unbounded Lazy-CSeq
Representing Jury Member	Thomas Ströder	Dexi Wang	Vadim Mutilin	Wei Wang	Michael Tautschnig	Matthias Dangl	Ming-Hsien Tsai	Jeremy Morse	Pablo Sánchez	Ondrej Lengal	Caterina Urban	Ton-Chanh Le	Gennaro Parlato	Herbert Oliveira Rocha	Bernd Fischer	Franck Cassez	Tomas Vojnar	Arie Gurfinkel	Zvonimir Rakamarić	Matthias Heizmann	Alexander Nutz	Salvatore La Torre
Affiliation	Aachen, Germany	Beijing, China	Moscow, Russia	New York, USA	London, UK	Passau, Germany	Taipei, Taiwan	Bristol, UK	Cantabria, Spain	Brno, Czechia	Paris, France	Singapore, Singapore	Southampton, UK	Manaus, Brazil	Stellenbosch, South Africa	Sydney, Australia	Brno, Czechia	Pittsburgh, USA	Salt Lake City, USA	Freiburg, Germany	Freiburg, Germany	Southampton, UK
Arrays 86 tasks, max. score: 145	–	–	–	–	-134 2 500 s	2 62 s	–	-206 5.5 s	–	–	–	–	–	–	–	–	–	0 0.61 s	**48** **400 s**	**2** **6.4 s**	**2** **5.9 s**	–
BitVectors 47 tasks, max. score: 83	–	4 58 s	–	52 16 000 s	**68** **1 800 s**	**58** **870 s**	–	**69** **470 s**	–	–	–	–	–	–	–	–	–	-80 550 s	–	5 170 s	-62 120 s	–
Concurrency 1 003 tasks, max. score: 1 222	–	–	–	–	**1 039** **70 000 s**	0 0 s	–	1 014 13 000 s	–	–	–	–	**1 222** **5 600 s**	–	**1 222** **16 000 s**	–	–	- 8 973 42 s	–	–	–	984 36 000 s
ControlFlow 1 927 tasks, max. score: 3 122	–	–	983 33 000 s	537 43 000 s	158 570 000 s	**2 317** **47 000 s**	–	**1 968** **59 000 s**	–	–	–	–	–	–	–	–	–	**2 169** **30 000 s**	1 691 78 000 s	1 887 54 000 s	872 10 000 s	–
ControlFlow-Integer 68 tasks, max. score 78	–	–	51 3 300 s	38 11 000 s	62 1 300 s	77 1 800 s	–	78 87 s	–	–	–	–	–	–	–	–	–	77 640 s	61 200 s	78 990 s	43 1 000 s	–
ECA 1 140 tasks, max. score 1 874	–	–	11 9 800 s	0 0 s	-2 334 560 000 s	987 39 000 s	–	923 58 000 s	–	–	–	–	–	–	–	–	–	516 25 000 s	112 45 000 s	632 46 000 s	1 14 s	–
Heap 162 tasks, max. score 235	–	–	14 790 s	46 32 000 s	59 6 900 s	118 2 600 s	–	66 620 s	168 45 000 s	–	–	–	–	–	–	125 1 600 s	–	130 1 100 s	86 370 s	115 2 900 s	109 4 300 s	–
ProductLines 597 tasks, max. score 829	–	–	637 19 000 s	0 0 s	399 1 900 s	901 4 100 s	–	917 430 s	–	–	–	–	–	–	–	–	–	913 3 500 s	917 32 000 s	554 3 300 s	87 5 000 s	–
DeviceDrivers64 1 650 tasks, max. score: 3 097	–	–	**2 736** **11 000 s**	–	2 293 380 000 s	**2 572** **39 000 s**	–	2 281 36 000 s	–	–	–	–	–	–	–	–	–	**2 657** **16 000 s**	2 507 72 000 s	274 850 s	82 270 s	–
Floats 81 tasks, max. score: 140	–	–	–	–	**129** **15 000 s**	**78** **5 100 s**	–	**-12** **5 300 s**	–	–	–	–	–	–	–	–	–	-164 5.9 s	–	–	–	–
HeapManipulation 80 tasks, max. score: 135	–	–	–	70 6 000 s	**100** **13 000 s**	96 930 s	–	79 37 s	–	32 1.8 s	–	–	–	–	–	–	**111** **140 s**	-37 14 s	**109** **820 s**	84 460 s	84 420 s	–
MemorySafety 205 tasks, max. score: 361	–	–	–	**200** **82 000 s**	-433 14 000 s	**326** **5 700 s**	–	–	–	22 25 s	–	–	–	28 2 100 s	–	–	**221** **460 s**	0 0 s	–	95 13 000 s	66 4 800 s	–
Recursive 24 tasks, max. score: 40	–	6 22 s	–	–	0 10 000 s	16 31 s	**18** **140 s**	–	–	–	–	–	–	–	–	–	–	-88 2.3 s	**27** **2 300 s**	**25** **310**	10 220	–
Sequentialized 261 tasks, max. score: 364	–	–	–	–	-171 39 000 s	**130** **11 000 s**	–	**193** **9 600 s**	–	–	–	–	–	–	–	–	–	-59 5 800 s	–	**15** **8 600 s**	-10 7 000 s	–
Simple 46 tasks, max. score: 68	–	–	32 4 200 s	–	51 16 000 s	**54** **4 000 s**	–	29 990 s	–	–	–	–	–	–	–	–	–	**65** **1 400 s**	**51** **5 100 s**	0 1 800 s	3 140 s	–
Termination 393 tasks, max. score: 742	**610** **5 400 s**	–	–	–	–	0 0 s	–	–	–	–	350 61 s	**545** **300 s**	–	–	–	–	–	0 0 s	–	**565** **8 600 s**	–	–
Overall 5 803 tasks, max. score: 9 562	–	–	–	–	**1 731** **1 100 000 s**	**4 889** **110 000 s**	–	- 2 161 130 000 s	–	–	–	–	–	–	–	–	–	- 6 228 53 000 s	–	**2 301** **87 000 s**	231 23 000 s	–

BLAST

CPAchecker

Error Trace Visualizer

Rule: Mutex lock/unlock

Error trace

☑ Function bodies | ☑ Blocks | Others...

```
3182    LDV_IN_INTERRUPT = 1;
3191    +ldv_initialize_FOREACH();
3195    tmp___8 = nondet_int() { /* The function body
3195    assert(tmp___8 != 0);
3198    tmp___7 = nondet_int() { /* The function body
3200    assert(tmp___7 != 0);
3280    assert(tmp___7 != 1);
3360    assert(tmp___7 != 2);
3440    assert(tmp___7 != 3);
3520    assert(tmp___7 != 4);
3600    assert(tmp___7 != 5);
3680    assert(tmp___7 != 6);
3760    assert(tmp___7 != 7);
3840    assert(tmp___7 != 8);
3920    assert(tmp___7 != 9);
4000    assert(tmp___7 != 10);
4080    assert(tmp___7 == 11);
4130    -carl9170_op_set_key(var_group1 /* hw */
        {
1031        -ar = *(hw).priv;
            err = 0;
1035        assert(*(ar).disable_offload == 0);
1035        assert(vif != 0);
1047        +tmp___7 = is_main_vif(ar /* ar */, v
1047        assert(tmp___7 == 0);
1159        assert(*(ar).rx_software_decryption
1163        +mutex_unlock_mutex(&(ar)->mutex /*
        }
    }
```

Source code

carl9170.h | **main.c.common.c** | wlan.h | rcupdate.h

```
1026 static int carl9170_op_set_key(struct ieee80211_hw *hw, enum set_key_cr
1027                              struct ieee80211_vif *vif,
1028                              struct ieee80211_sta *sta,
1029                              struct ieee80211_key_conf *key)
1030 {
1031   struct ar9170 *ar = hw->priv;
1032   int err = 0, i;
1033   u8 ktype;
1034
1035   if (ar->disable_offload || !vif)
1036           return -EOPNOTSUPP;
1037
1038   /*
1039    * We have to fall back to software encryption, whenever
1040    * the user choose to participates in an IBSS or is connected
1041    * to more than one network.
1042    *
1043    * This is very unfortunate, because some machines cannot handle
1044    * the high througput speed in 802.11n networks.
1045    */
1046
1047   if (!is_main_vif(ar, vif))
1048           goto err_softw;
1049
1050   /*
1051    * While the hardware supports *catch-all* key, for offloading
1052    * group-key en-/de-cryption. The way of how the hardware
1053    * decides which keyId maps to which key, remains a mystery...
1054    */
```

Bugs Found

http://linuxtesting.org/results/ldv

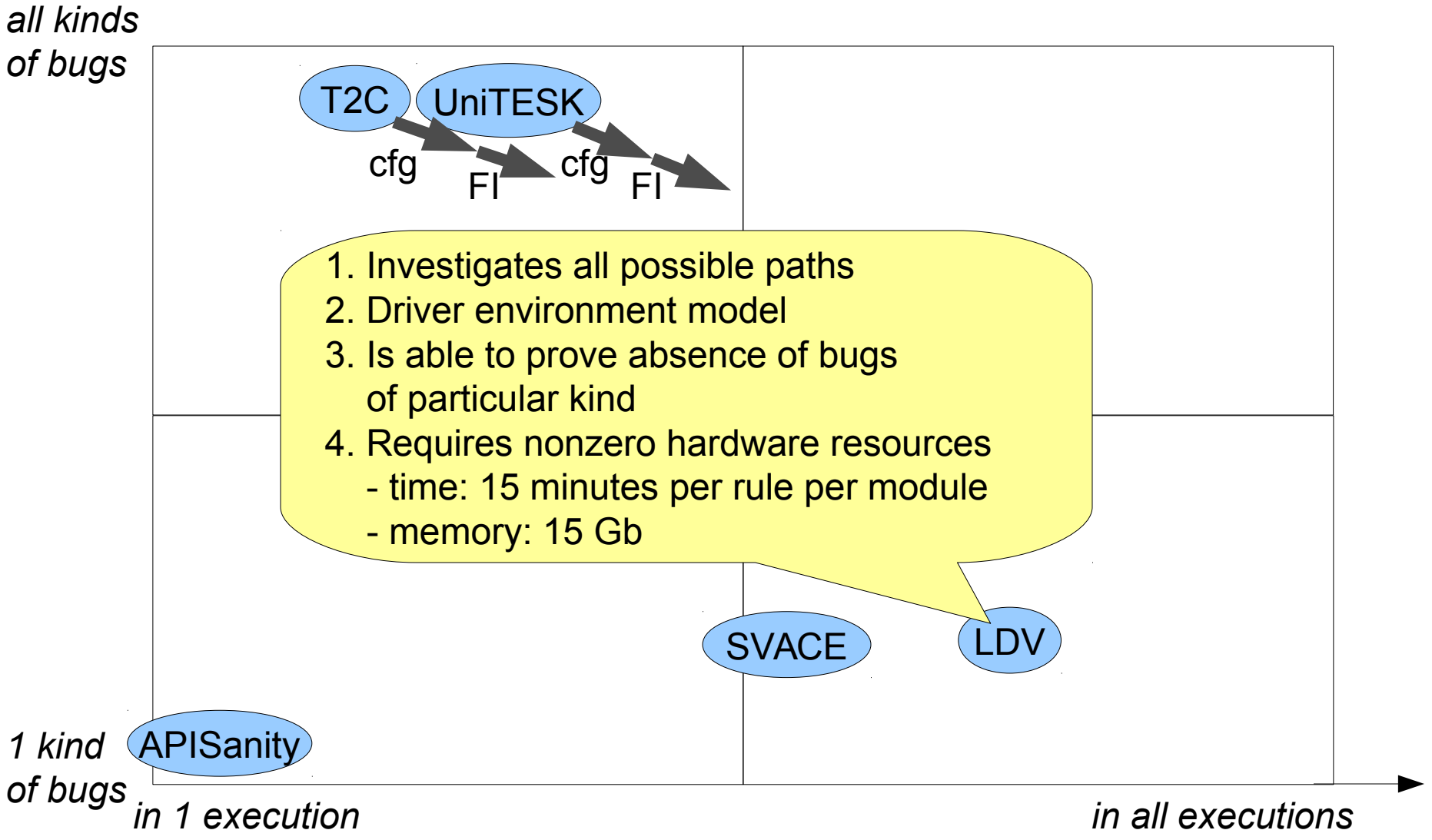
>230 patches already applied

Problems in Linux Kernel

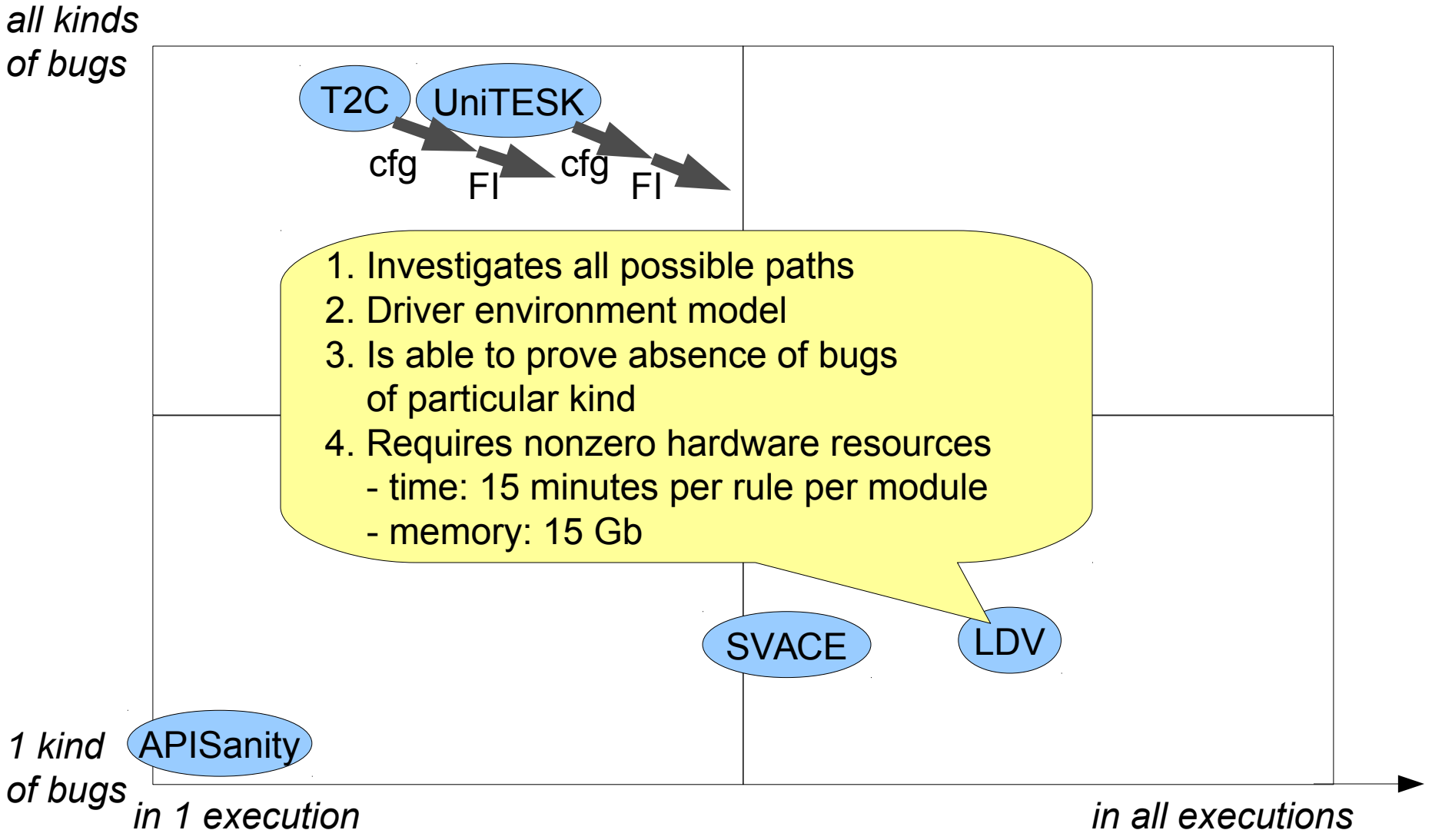
This section contains information about problems in Linux kernel found within Linux Driver Verification program.

No.	Type	Brief	Added on	Accepted	Status
L0212	Deadlock	nfit: acpi_nfit_notify(): Do not leave device locked	2015-12-11	https://lkml.org/lkml/2015/12/11/781 commit	Fixed in kernel 4.4-rc6
L0211	Crash	USB: whci-hcd: no check for dma mapping error	2015-12-01	http://linuxtesting.org/pipermail/ldv-project/2015-November/000558.html commit	Fixed in kernel 4.4-rc5
L0210	Crash	vmxnet3: fix checks for dma mapping errors	2015-11-28	https://lkml.org/lkml/2015/11/27/498 commit	Fixed in kernel 4.4-rc4
L0209	Crash	sound: fix check for error condition of register_chrdev()	2015-11-07	https://lkml.org/lkml/2015/11/6/914 commit	Fixed in kernel 4.4-rc1
L0208	Crash	mcb: Do not return zero on error path in mcb_pci_probe()	2015-10-28	https://lkml.org/lkml/2015/10/17/238 commit	Fixed in kernel 4.4-rc1
L0207	Crash	staging: r8188eu: _enter_critical_mutex() error handling	2015-10-28	https://www.spinics.net/lists/kernel/msg2094451.html commit	Fixed in kernel 4.4-rc1
L0206	Deadlock	usb: gadget: pch-udc: fix deadlock in pch-udc	2015-09-18	https://lkml.org/lkml/2015/9/28/256 commit	Fixed in kernel 4.4-rc1
L0205	Leak	mcb: leaks in mcb_pci_probe()	2015-09-16	https://lkml.org/lkml/2015/7/8/1041	Fixed in kernel

OS Verification



OS Verification



LinuxTesting.org

Deductive Verification

ISP RAS

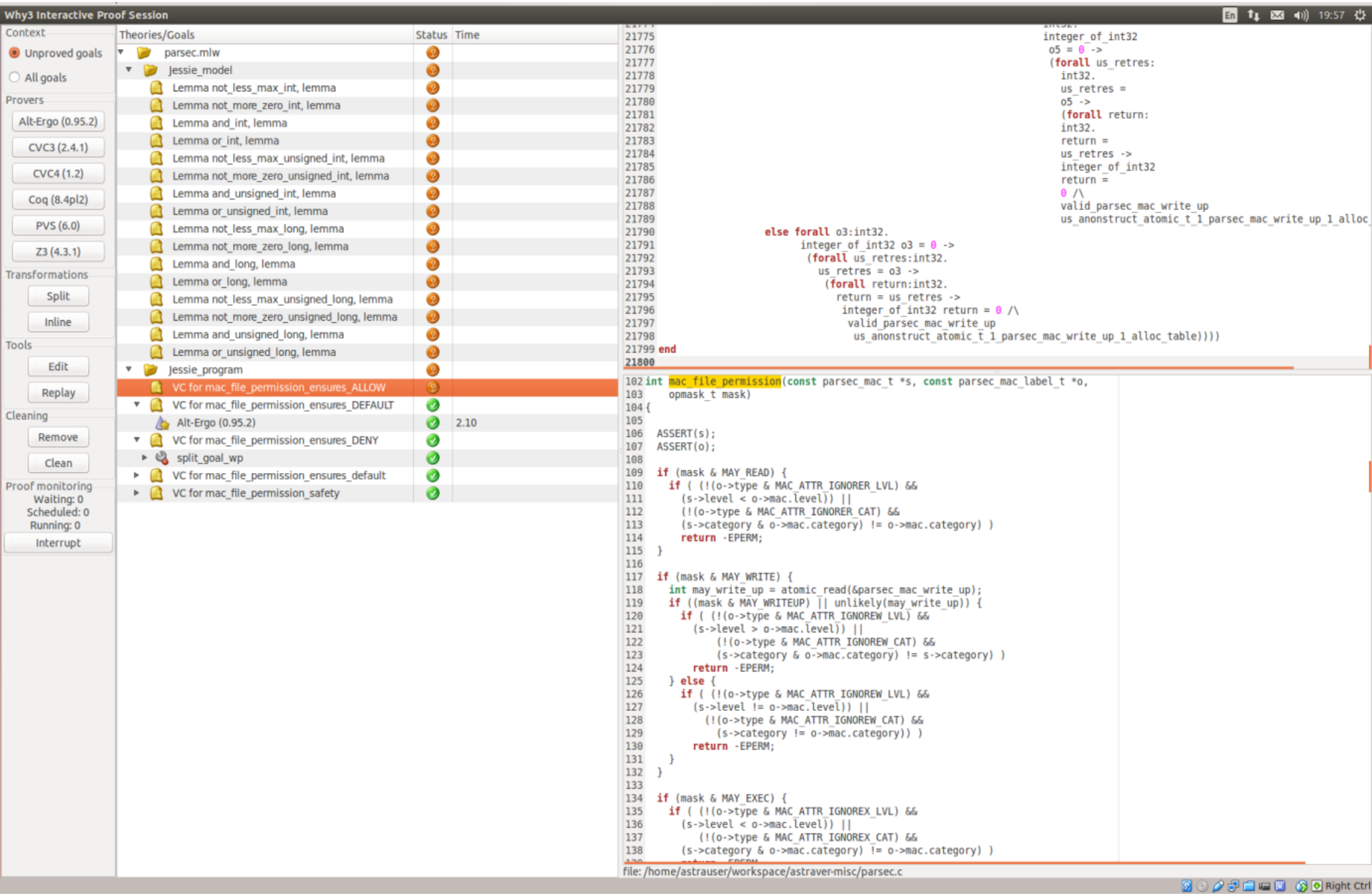
Institute for System Programming of the Russian Academy of Sciences

Deductive verification

ISP RAS

- 1947 — Alan Turing — Lecture to the London Mathematical Society
- 1970 — R. Floyd / C.A.R. Hoare Methods
- 2000 — Tools for deductive verification of programs in C, Java, C#
  - SunRise, ESC/Java, Frama-C, LOOP, Boogie/VCC
- 2010 — Industrial applications
  - Nuclear power (UK, France)
  - Avionics (Airbus, NASA, UK Air Traffic Control)
  - Operating systems
    (seL4, PikeOS, Hyper-V)

```
/*@ predicate is_divisor(int m, int n) { if (m) then (n % m == 0) else \false } */

/*@ predicate is_gcd(int z, int x1, int x2) {
  @         is_divisor(z,x1)
  @      && is_divisor(z,x2)
  @      && \forall int i; is_divisor(i,x1) && is_divisor(i,x2) => (i <= z) }
  @*/

/*@ logic int gcd(int m, int n) */
/*@ axiom gcd_def: \forall int m; \forall int n; is_gcd(gcd(m,n),m,n) */


/* The function returns the greatest common divisor of x1 and x2 */
/*@ requires (x1 > 0) && (x2 > 0)
  @ ensures is_gcd(\result,x1,x2)
  @*/
int nod(int x1, int x2)
{
int y1 = x1;
int y2 = x2;
int tmp = 0;

  if (y1 > y2) {
    y1 = x2;
    y2 = x1;
  }
  /*@
    @ invariant ((0 < y1 <= y2) && gcd(x1,x2) == gcd(y1,y2))
    @         || ((y1 == 0) && gcd(x1,x2) == y2)
    @ variant y1
    @*/
  while (y1 != 0) {
    tmp = y1;
    y1 = y2%y1;
    y2 = tmp;
  }
```


Why3 Interactive Proof Session
Context
Unproved goals
All goals
Provers
Alt-Ergo (0.95.2)
CVC3 (2.4.1)
CVC4 (1.2)
Coq (8.4pl2)
PVS (6.0)
Z3 (4.3.1)
Transformations
Split
Inline
Tools
Edit
Replay
Cleaning
Remove
Clean
Proof monitoring
Waiting: 0
Scheduled: 0
Running: 0
Interrupt
Theories/Goals
Status
Time
parsec.mlw
Jessie_model
Lemma not_less_max_int, lemma
Lemma not_more_zero_int, lemma
Lemma and_int, lemma
Lemma or_int, lemma
Lemma not_less_max_unsigned_int, lemma
Lemma not_more_zero_unsigned_int, lemma
Lemma and_unsigned_int, lemma
Lemma or_unsigned_int, lemma
Lemma not_less_max_long, lemma
Lemma not_more_zero_long, lemma
Lemma and_long, lemma
Lemma or_long, lemma
Lemma not_less_max_unsigned_long, lemma
Lemma not_more_zero_unsigned_long, lemma
Lemma and_unsigned_long, lemma
Lemma or_unsigned_long, lemma
Jessie_program
VC for mac_file_permission_ensures_ALLOW
VC for mac_file_permission_ensures_DEFAULT
Alt-Ergo (0.95.2)
2.10
VC for mac_file_permission_ensures_DENY
split_goal_wp
VC for mac_file_permission_ensures_default
VC for mac_file_permission_safety
21775 integer_of_int32
21776 o5 = 0 ->
21777 (forall us_retres:
21778 int32.
21779 us_retres =
21780 o5 ->
21781 (forall return:
21782 int32.
21783 return =
21784 us_retres ->
21785 integer_of_int32
21786 return =
21787 0 /\
21788 valid_parsec_mac_write_up
21789 us_anonstruct_atomic_t_1_parsec_mac_write_up_1_alloc
21790 else forall o3:int32.
21791 integer_of_int32 o3 = 0 ->
21792 (forall us_retres:int32.
21793 us_retres = o3 ->
21794 (forall return:int32.
21795 return = us_retres ->
21796 integer_of_int32 return = 0 /\
21797 valid_parsec_mac_write_up
21798 us_anonstruct_atomic_t_1_parsec_mac_write_up_1_alloc_table))))
21799 end
21800
102 int mac_file_permission(const parsec_mac_t *s, const parsec_mac_label_t *o,
103 opmask_t mask)
104 {
105
106 ASSERT(s);
107 ASSERT(o);
108
109 if (mask & MAY_READ) {
110 if ((!(o->type & MAC_ATTR_IGNORER_LVL) &&
111 (s->level < o->mac.level)) ||
112 (!(o->type & MAC_ATTR_IGNORER_CAT) &&
113 (s->category & o->mac.category) != o->mac.category))
114 return -EPERM;
115 }
116
117 if (mask & MAY_WRITE) {
118 int may_write_up = atomic_read(&parsec_mac_write_up);
119 if ((mask & MAY_WRITEUP) || unlikely(may_write_up)) {
120 if ((!(o->type & MAC_ATTR_IGNOREW_LVL) &&
121 (s->level > o->mac.level)) ||
122 (!(o->type & MAC_ATTR_IGNOREW_CAT) &&
123 (s->category & o->mac.category) != s->category))
124 return -EPERM;
125 } else {
126 if ((!(o->type & MAC_ATTR_IGNOREW_LVL) &&
127 (s->level != o->mac.level)) ||
128 (!(o->type & MAC_ATTR_IGNOREW_CAT) &&
129 (s->category != o->mac.category)))
130 return -EPERM;
131 }
132 }
133
134 if (mask & MAY_EXEC) {
135 if ((!(o->type & MAC_ATTR_IGNOREX_LVL) &&
136 (s->level < o->mac.level)) ||
137 (!(o->type & MAC_ATTR_IGNOREX_CAT) &&
138 (s->category & o->mac.category) != o->mac.category))
file: /home/astrauser/workspace/astraver-misc/parsec.c
Right Ctrl

Astraver Project

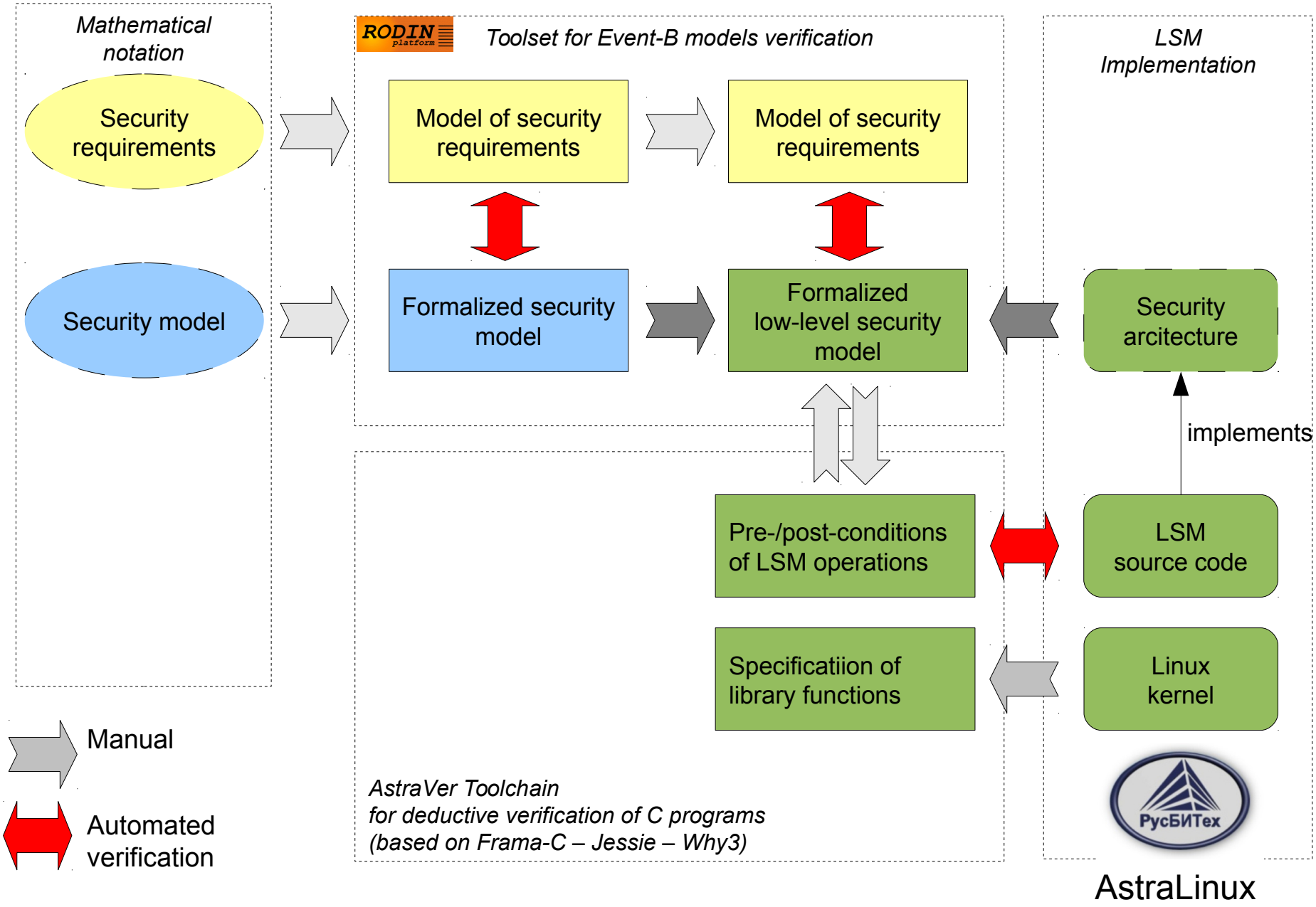
- Deductive Verification of Linux Security Module
  - Joint project with NPO RusBITech
  - Formal security model MROSL-DP
- Assumptions
  - Linux kernel core conforms with its specifications
    - It is not target to prove
- Target code
  - Code is hardware independent
  - Verification unfriendly
  - Out of control

MROSL DP

- Operating system access control model
  - Hierarchical Role-Based Access Control (RBAC)
  - Mandatory Access Control (MAC)
  - Mandatory Integrity Control (MIC)
- Implemented as Linux Security Module (LSM) for Astra Linux
- ~150 pages in mathematical notation

AstraVer Project

ISP RAS



Problems with the tools

- Memory model limitations
  - Arithmetics with pointers to fields of structures (container_of)
  - Prefix structure casts
  - Reinterpret casts
- Integer model problems
- Limited code support
  - Functional pointers
  - String literals
- Scalability problems
- Usability problems

Institute for System Programming of the Russian Academy of Sciences

VERIFICATION CENTER OF THE OPERATING SYSTEM Linux

About Us

- About Center
- Our Team
- News
- Partners
- Contacts

Projects

- Linux Kernel Space Verification
- LSB Infrastructure
- Testing Technologies
- Tests and Frameworks
- Portability Tools

Results

- Contribution
- Publications
- Events

khoroshilov

- My account
- User list
- Create content
- Feed aggregator
- Administer
- Log out

18-Feb-2015: The first public release of Astraver Toolset

View | Edit | Track | Translate

Submitted by Mikhail Mandrykin on Wed, 18/02/2015 - 18:30

We are happy to announce the first public release of **Astraver Toolset 1.0** that is built on top of the '**Frama-C** + **Jessie** + **Why3 IDE**' deductive verification toolchain. The toolchain was adapted, so it can be used to specify and prove properties of Linux kernel code. The most of our modifications go to the Jessie plugin, while the Frama-C front-end and the Why3 platform have got just minor fixes or improvements. Some of our modifications were already applied upstream, while the rest is available in **our public repositories**.

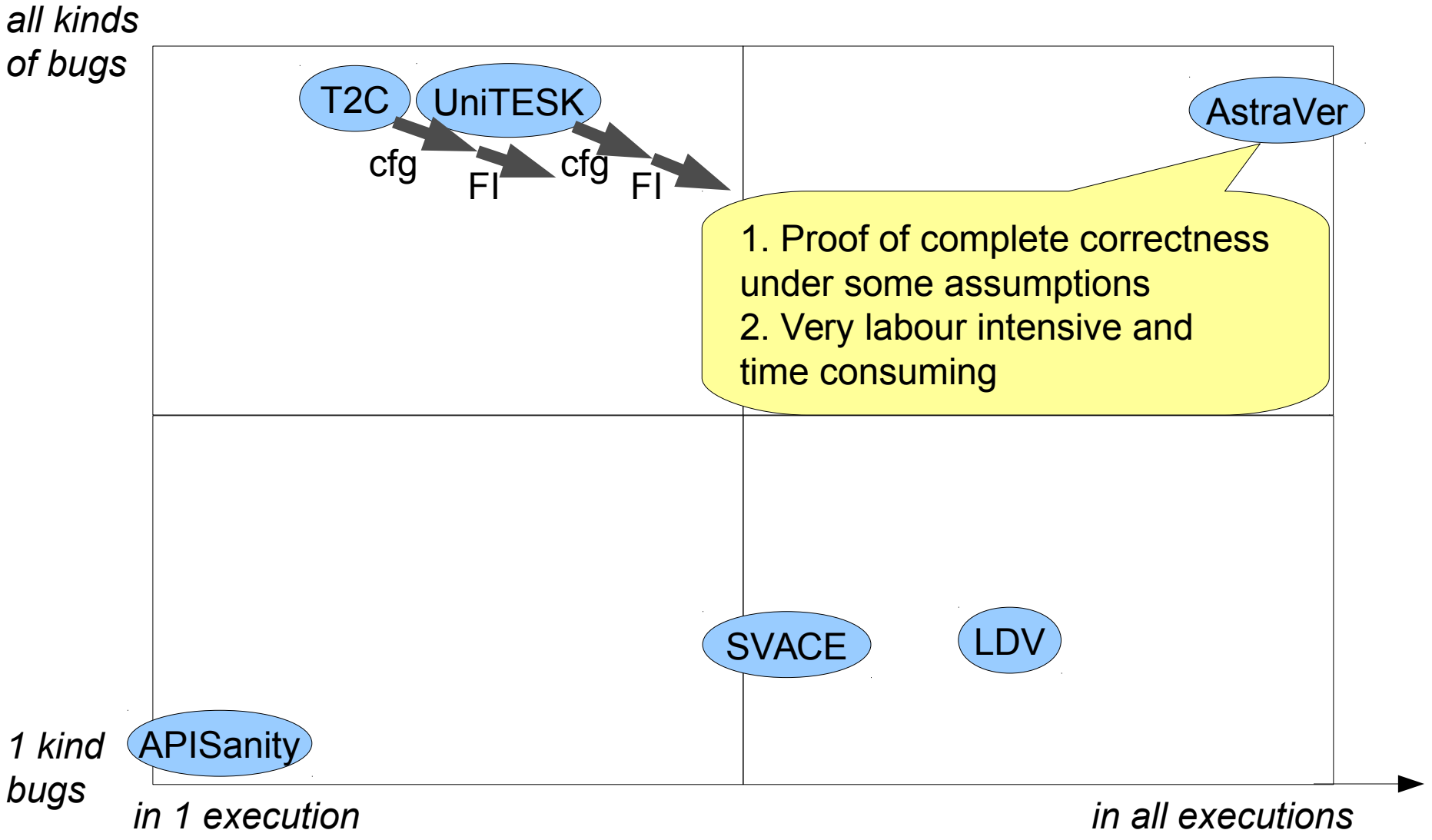
The most important modifications are described below.

C Language Support

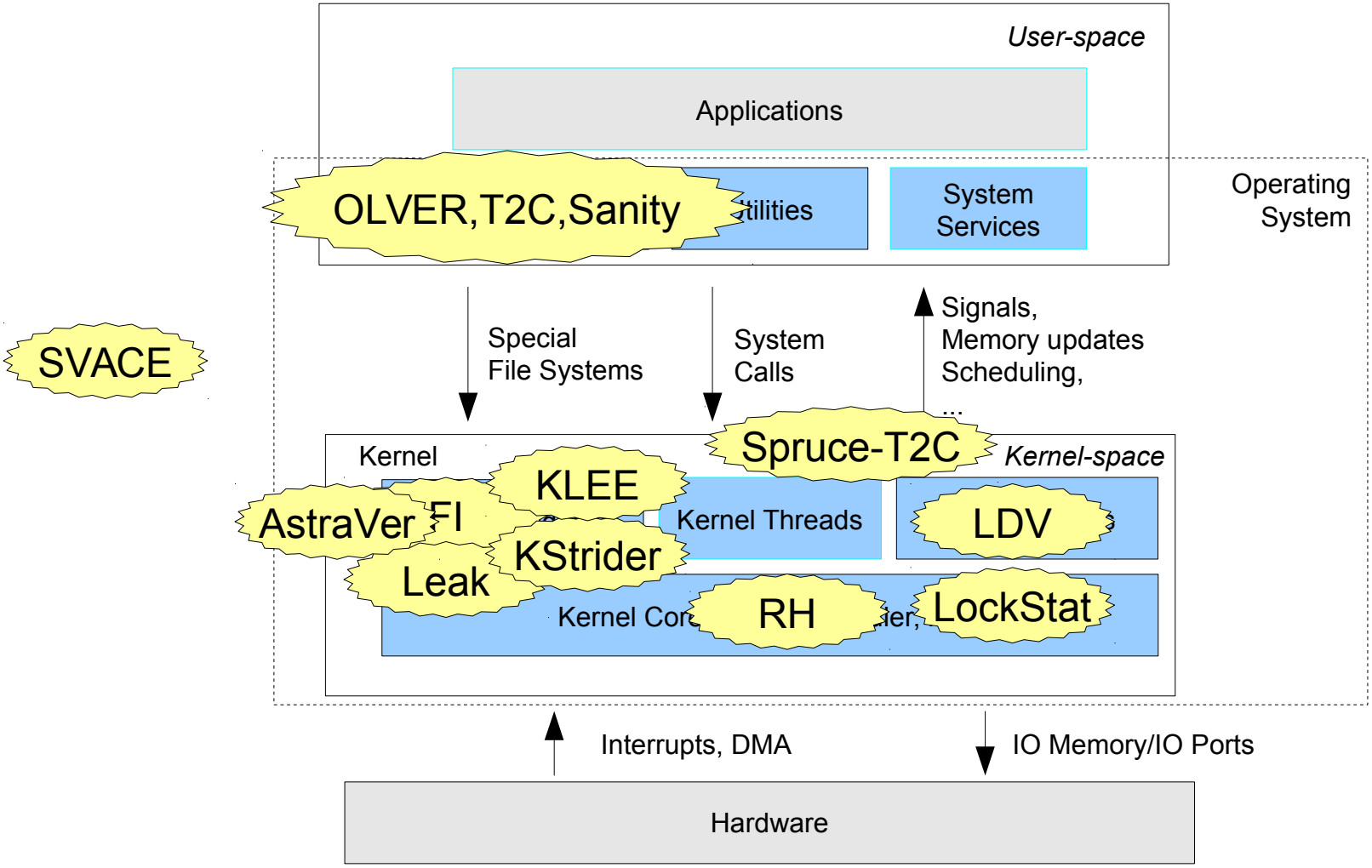
- Low-level reinterpret type casts between pointers to integral types. This feature required modification of the Jessie memory model as described in our paper "Extended High-Level C-Compatible Memory Model with Limited Low-Level Pointer Cast Support for Jessie Intermediate Language". The overall idea can be summarized as an ability to do certain ghost re-allocations of memory blocks in explicitly specified points in order to transform arrays of allocated objects (structures) from one type to another. **WARNING.** Discriminated unions support is not yet fully adapted to the modified memory model.
- Prefix type casts between outer structures and their corresponding first substructures (through field inlining and structure inheritance relation in Jessie).
- Kernel memory (de)allocating functions kmalloc()/kzalloc(), kfree().
- Builtin C99 __Bool type.
- Standard library functions memcpy(), memmove(), memcmp() and memset(). The support for these functions is implemented through type-based specialization of several pre-defined pattern specifications. **(*)**
- Function pointers (through exhaustive may-aliases checking). **(*)**
- Variadic functions (through additional array argument). **(*)**
- Inline assembly (through undefined function calls). **(*)**

> ***(*)***_The main purpose of implementing support for these features was the ability to use the tools on our target code without the need for its significant preliminary modification. As a result the support is not complete enough to be usable for verification of code that significantly relies on these features. For instance:_

OS Verification



Operating Systems



Conclusions

- Test Execution System
- Benchmarking

Thank you!

Alexey Khoroshilov
khoroshilov@ispras.ru
http://linuxtesting.org/

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Math

ISP RAS
Institute for System Programming of the Russian Academy of Sciences

Test Results: Details