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

Testing and Verification of Operating Systems and Information Security Issues

Prof. Alexander K. Petrenko,
petrenko@ispras.ru

12th TAROT Summer School on Software Testing, Verification & Validation

Paris, July, 2016

Institute for System Programming

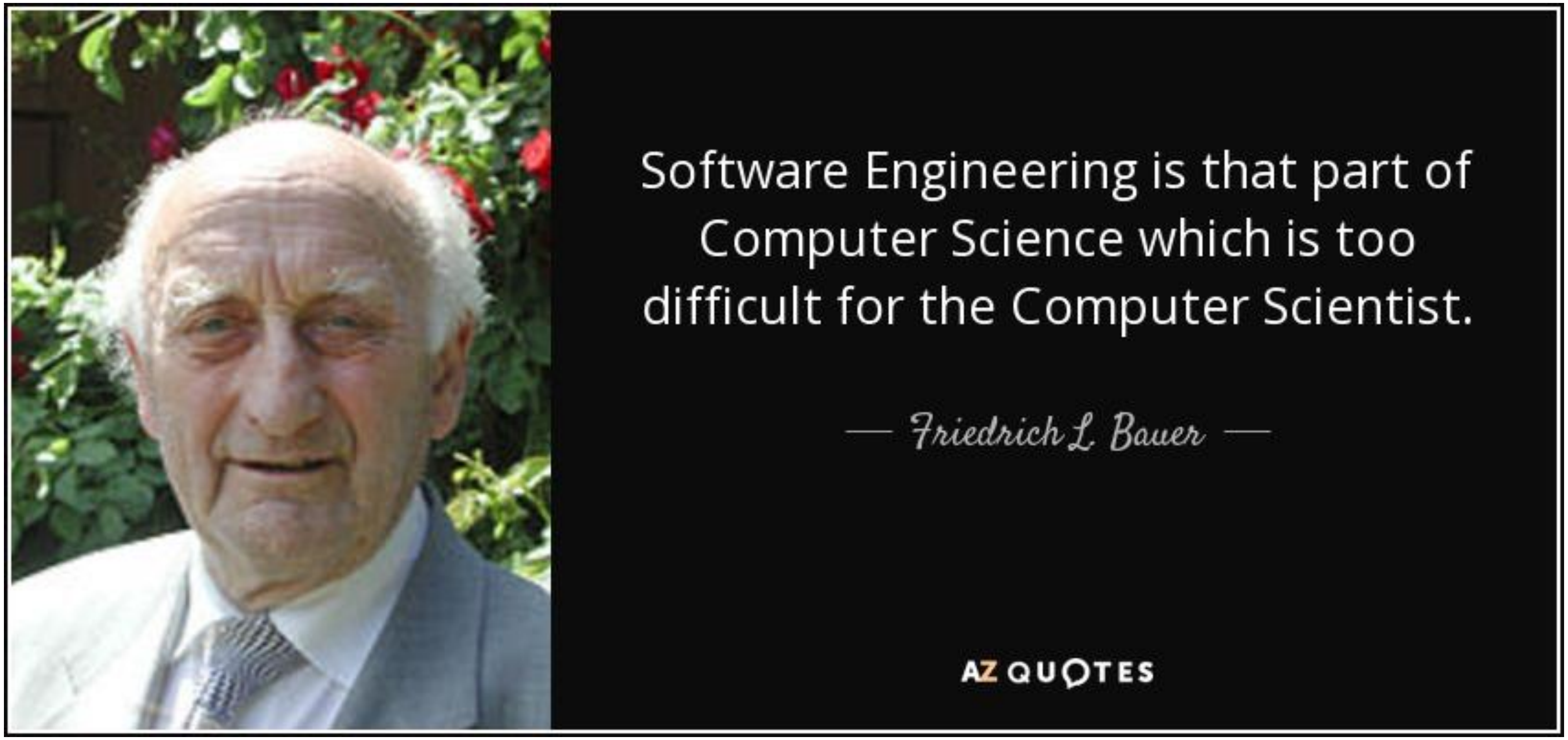
- ISP RAS belongs to the Division of Mathematical Sciences of the RAS.
- The Institute employs more than 200 highly qualified researchers and software engineers, including 12 doctors of science and 45 philosophy doctors.
- Many employees of the Institute also work as professors in leading Moscow universities.

SAMSUNG
intel
Microsoft
NVIDIA CUDA RESEARCH CENTER
DELL
Google
IBM
Klocwork
hp
NORTEL
Synchro
Explore options. Manage solutions.
THE LINUX FOUNDATION
INRIA
ETSI
ГосНИИ АС
Научно-производственное объединение
РусБИТех
Открытое акционерное общество
Билайн
живи на яркой стороне

Software Engineering Department

- SE Department staff:
  - over 40 researchers and engineers, including 3 Doctors of Sc. and 13 Ph.D.
- Major partners and customers
  - Foreign partners: Microsoft Research, Intel Labs, Nokia, Google, ETRI, EADS Telecom, University of Passau, Fraunhofer FOKUS
  - Russian partners: NIISI RAS, GosNIIAS, VimpelCom, MCST(Elbrus)
  - International organizations: ISO/JTC 1, ETSI, The Linux Foundation

ISPRAS Research Model = ?

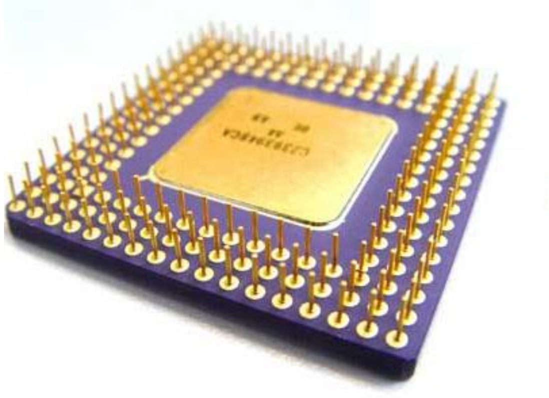


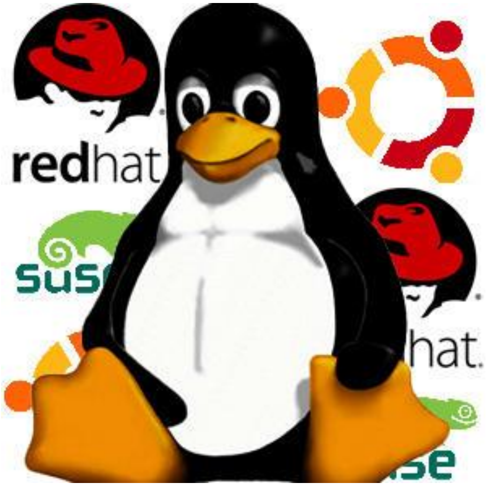
Software Engineering is that part of Computer Science which is too difficult for the Computer Scientist.

— Friedrich L. Bauer —

ISPRAS Research Model = Industrial Research

Application Domains





SE Department R&D Domains

- Verification techniques and tools (testing, software model checking, deductive verification)
- Trusted operating systems (Linux family, ARINC-653 Real-Time OS)
- Tool chains for critical software life cycle support
  - Requirements management tools
  - System modeling (AADL), simulation, risk analysis
  - Cyber-physical system integration (avionics)
- Telecom and operating systems API/ABI standards
- Hardware designs testing
- Model Based Testing foundations

Agenda

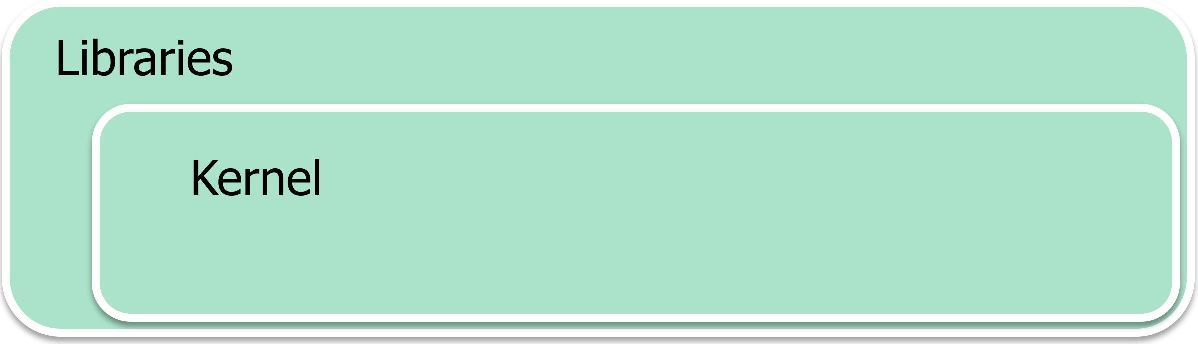
1. What is the “Operating System”?
2. Spectrum of OS testing and verification methods
3. State of the Art and ISPRAS’s experience
4. Information security specifics and OS verification

OS Verification Challenge

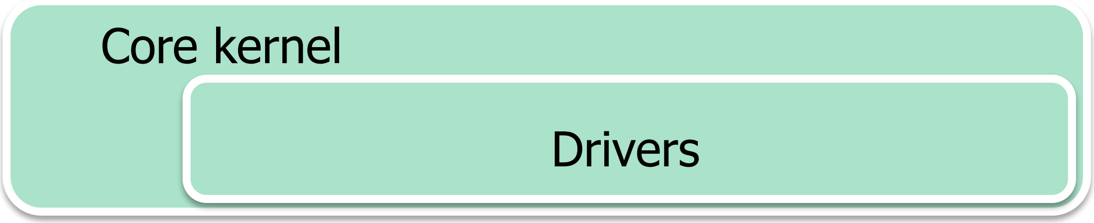
- Operating System is a base of software platform. Reliability and security of OS is ultimate prerequisite of information technologies quality
- Critical software/systems need certification. OS certification is necessary part of certification process
- IT domains requiring reliable, secure, trusted OSs:
  - Servers and work stations
  - Data centers
  - Avionics, other computing intensive systems
  - Mobile devices
  - SCADA, etc.

OS Architecture

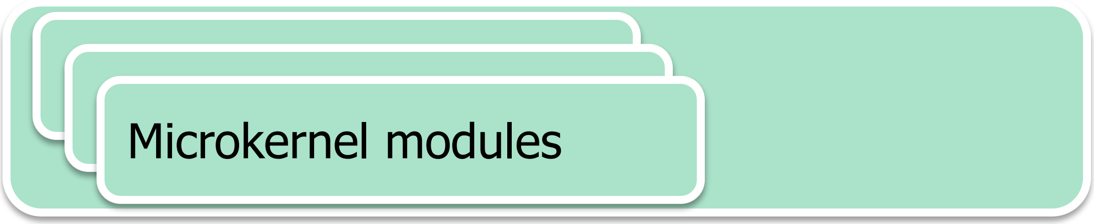
- Libraries + Kernel



- Monolithic Kernel

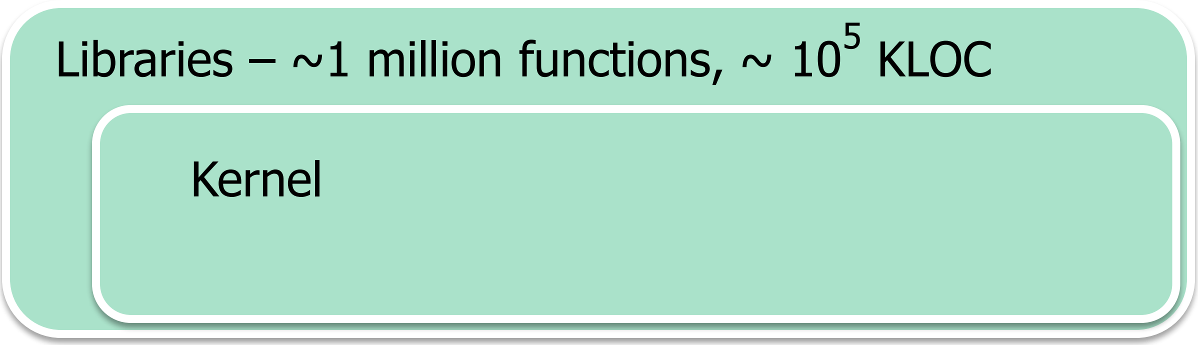


- Microkernel



OS Architecture. Scale

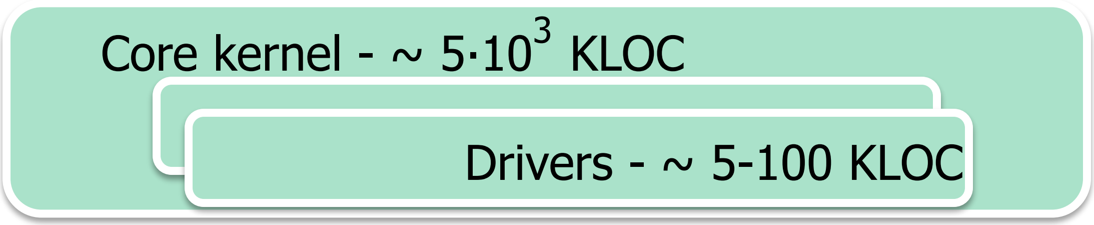
- Libraries + Kernel



Libraries – ~1 million functions, ~ 10^5 KLOC

Kernel

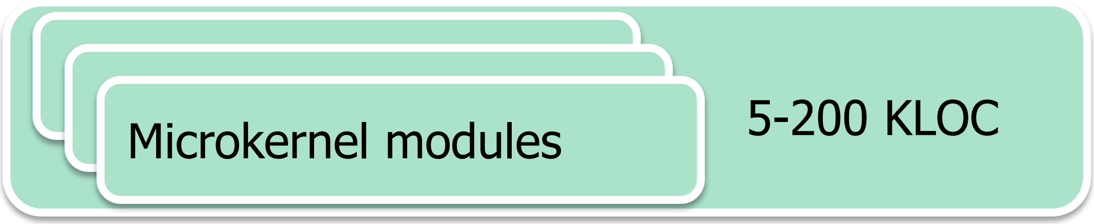
- Monolithic Kernel



Core kernel - ~ $5 \cdot 10^3$ KLOC

Drivers - ~ 5-100 KLOC

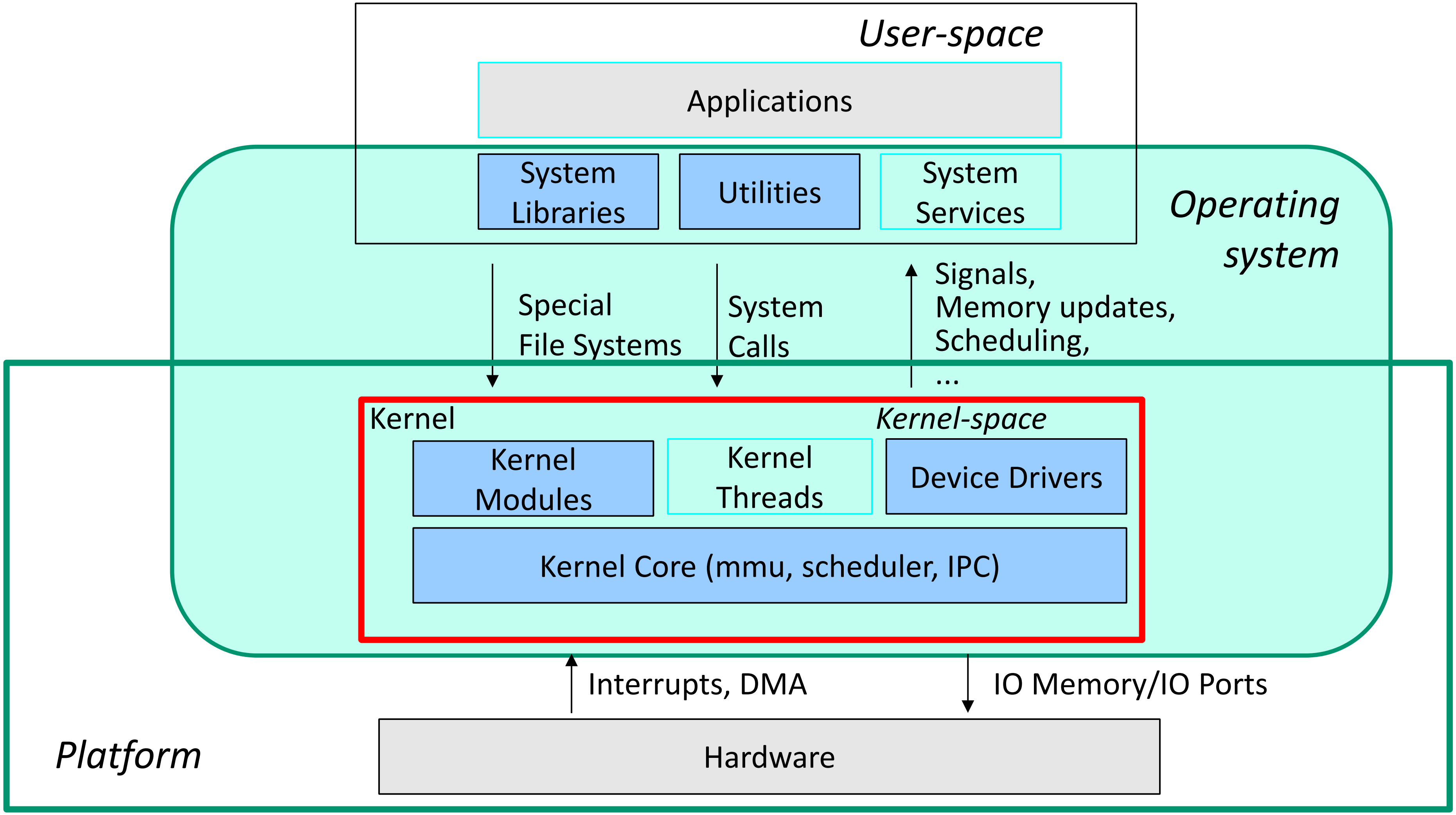
- Microkernel



Microkernel modules

5-200 KLOC

Operating Systems Structure



Spectrum of Testing/Verification Approaches

- Testing (dynamic analysis, monitoring, run-time verification, fault injection)
- Static analysis (*lightweight analysis*, software model checking)
- Static/dynamic analysis (DART, concolic testing)
- Deductive verification

Spectrum of Testing/Verification Approaches vs. Verification Aspects

- Testing (dynamic analysis, monitoring, run-time verification)
- Static analysis (*lightweight analysis*, software model checking)
- Static/dynamic analysis (DART, concolic testing)
- Deductive verification

Testing/Verification aspects:

- Functionality / Conformance / Reliability / Security / . . .
- *Usability testing*
- *Performance modeling and testing*
- *. . .*

Static Analysis

Dynamic Analysis

Static Analysis	Dynamic Analysis
+ All paths at once	− One path only

Static Analysis	**Dynamic Analysis**
+ 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 Analysis	Dynamic Analysis
+ 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 Analysis	Dynamic Analysis
+ 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

State of the Art. Methods and Tools. Testing

- 3 views on OS :
  - OS as API for applications
  - OS is an OS kernel
  - OS is a part of software/hardware platform
- OS as API for applications
  - **Problems**
  - Huge set of APIs (over 1 million functions)
  - Lack of specifications (poor quality of specifications)

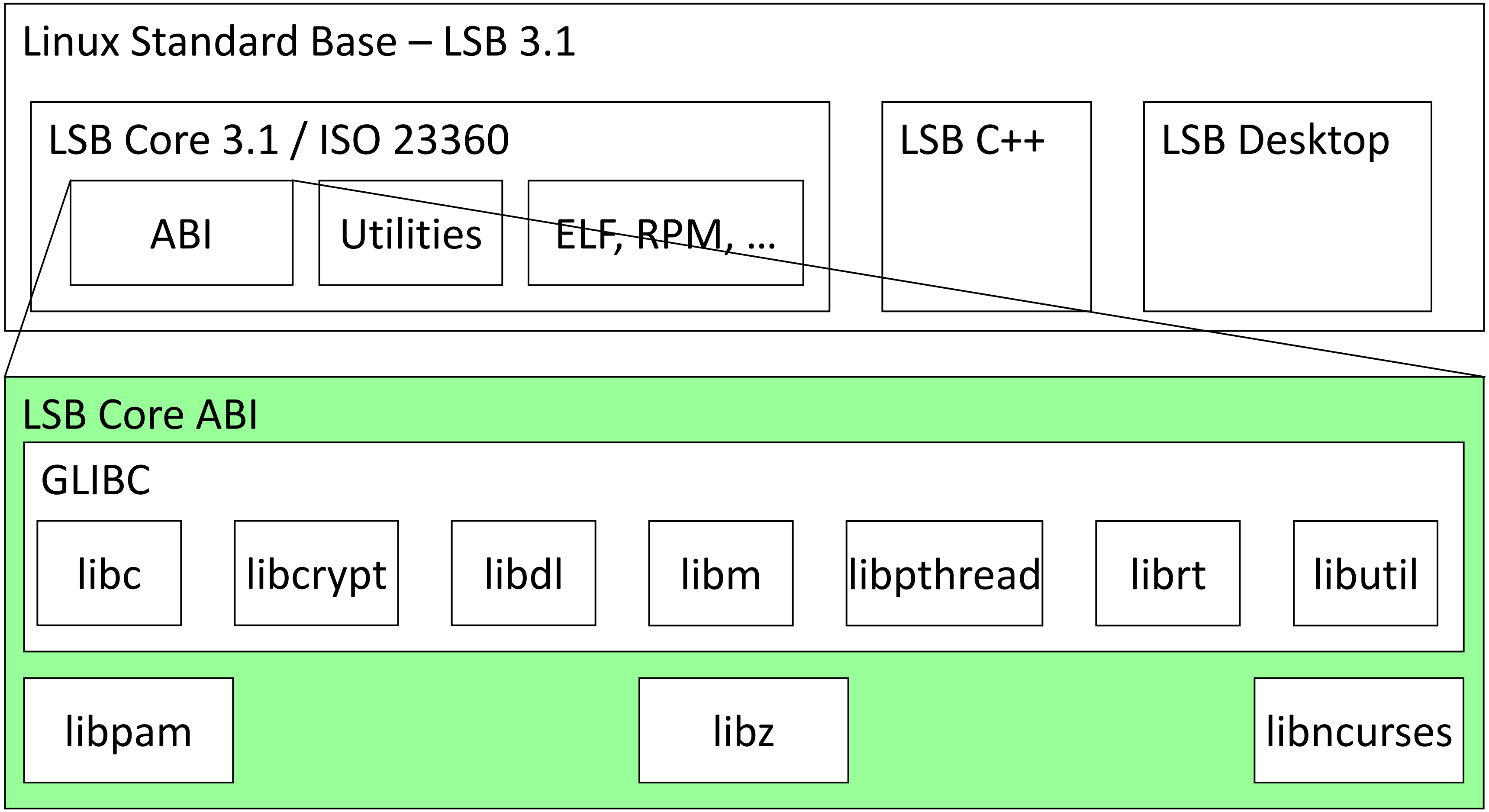
State of the Art.
Methods and Tools. Testing

- 3 views on OS:
  - OS as API for applications
  - OS is an OS kernel
  - OS is a part of software/hardware platform
- OS as API for applications.
  - **Problems**
  - Huge set of APIs (over 1 million functions)
  - Lack of specifications (poor quality of specifications)
  - **Methods**
  - Traditional (handmade) test suites
  - Specification/model based testing
  - **Specification based testing tools**
  - ADLT (Sun Microsystem, 1993)
  - ***KVEST (Nortel, ISPRAS, 1994-1999)***
  - ***UniTESK/CTESK (ISPRAS, 2000-2007***
  - SpecExplorer (Microsoft, 2004-2009)

OLVER – Model Based Testing of Linux Basic Libraries(*)

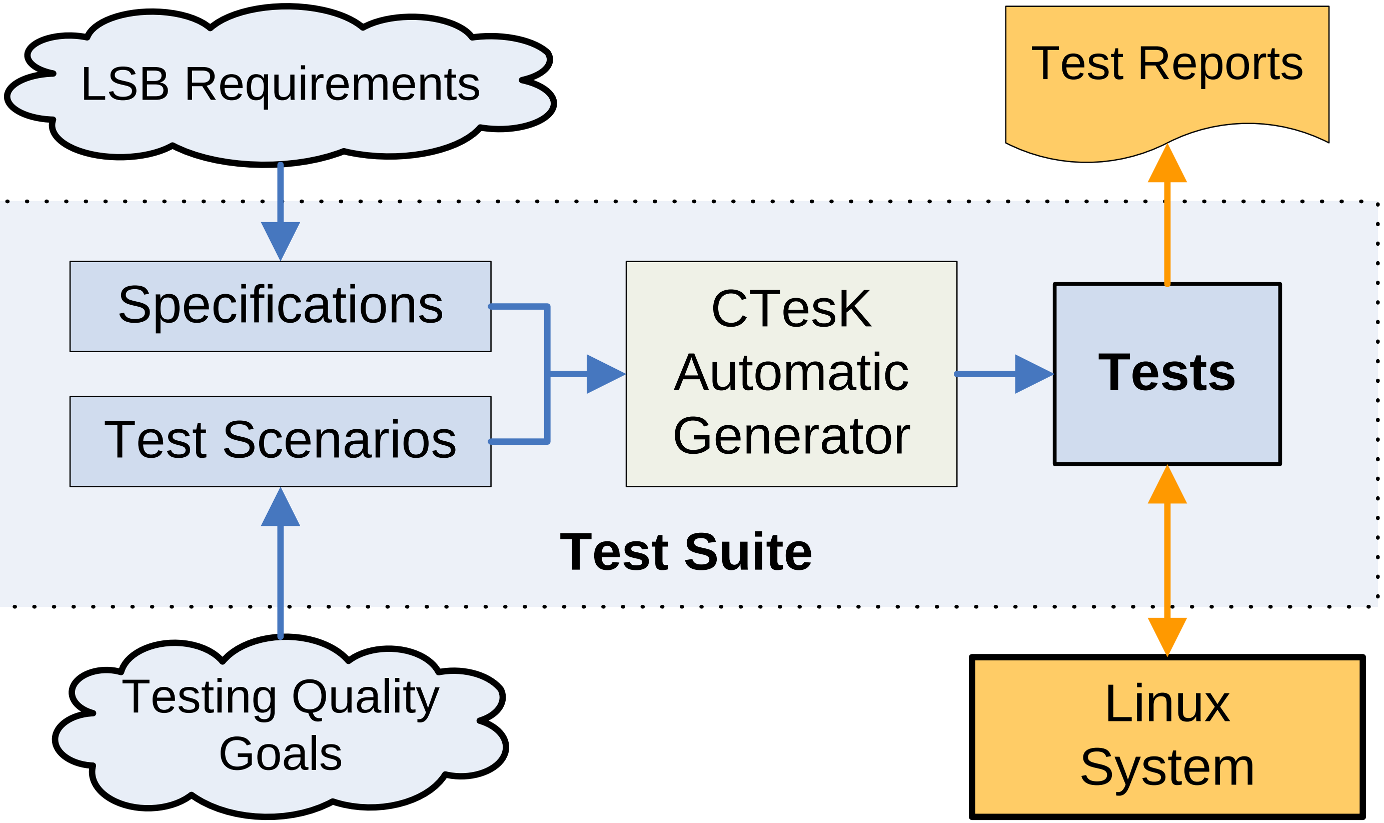
(*) The project was supported by Russian Ministry of Education and Science and by The Linux Foundation

OLVER: Open Linux VERification

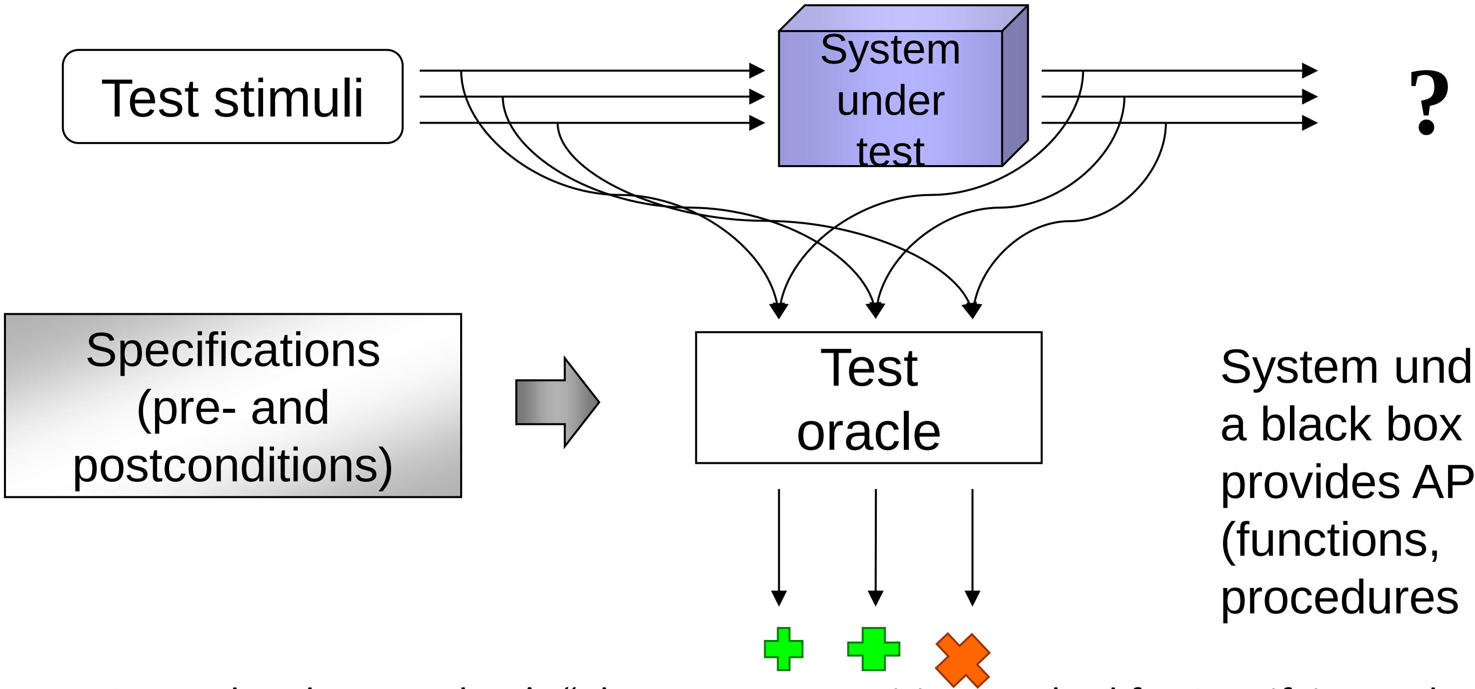


More 1500 interfaces

OLVER Process



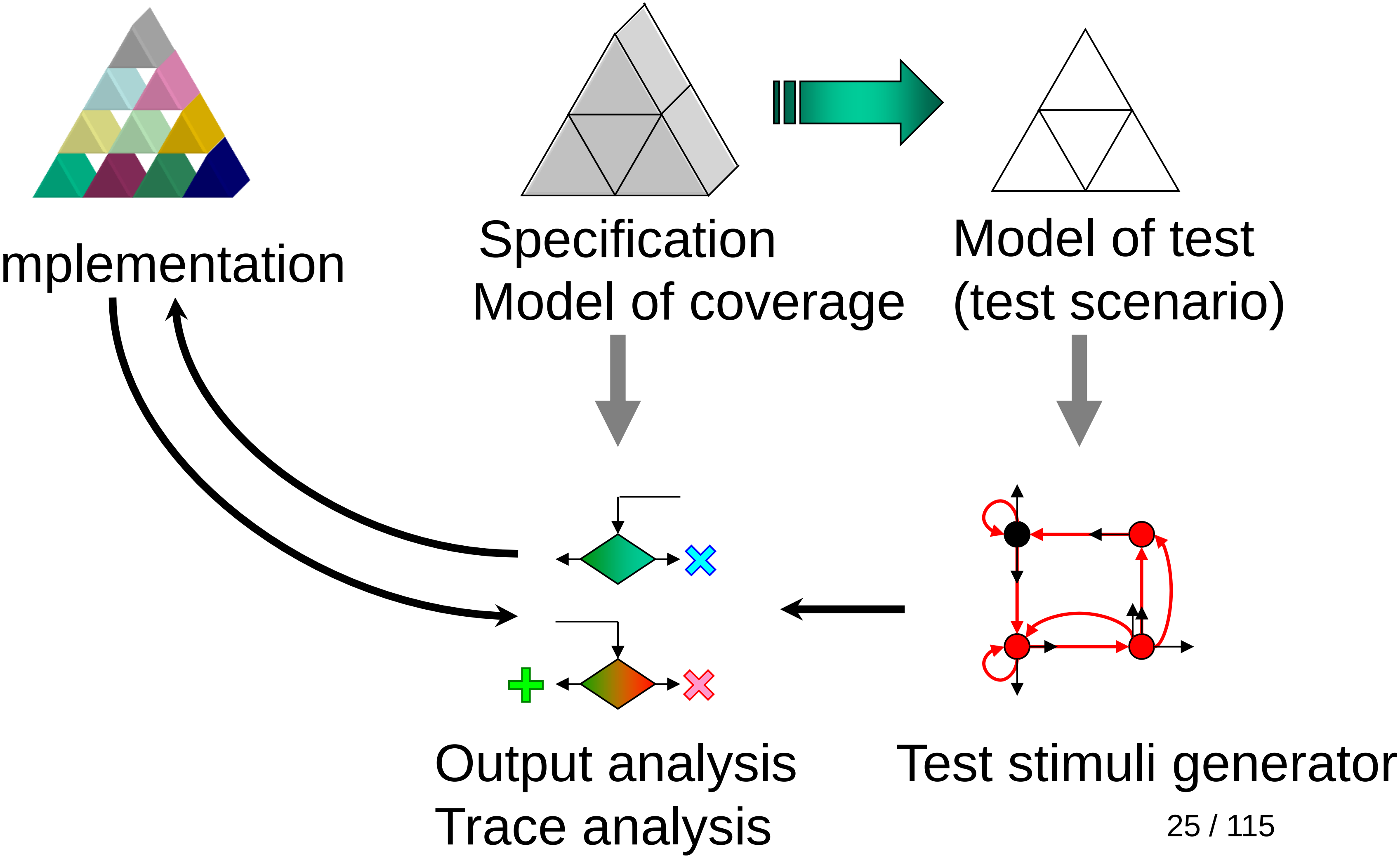
Technology: KVEST (1999)/UniTESK (2002)
Test Oracles



System under test is a black box that provides API (functions, procedures etc.)

- T. J. Ostrand and M. J. Balcer’s “The Category-Partition Method for Specifying and Generating Functional Tests” (in CACM, 31(6):676–686, June 1988).
- I.Burdonov, A.Kossatchev, A.Petrenko, D.Galter. KVEST: Automated Generation of Test Suites from Formal Specifications. Proceedings of Formal Method Congress, Toulouse, France, 1999, LNCS, No. 1708.
- I.Bourdonov, A.Kossatchev, V.Kuliamin, and A.Petrenko. UniTesK Test Suite Architecture. Proc. of FME 2002. LNCS 2391.

KVEST/UniTesK Workflow



Implementation

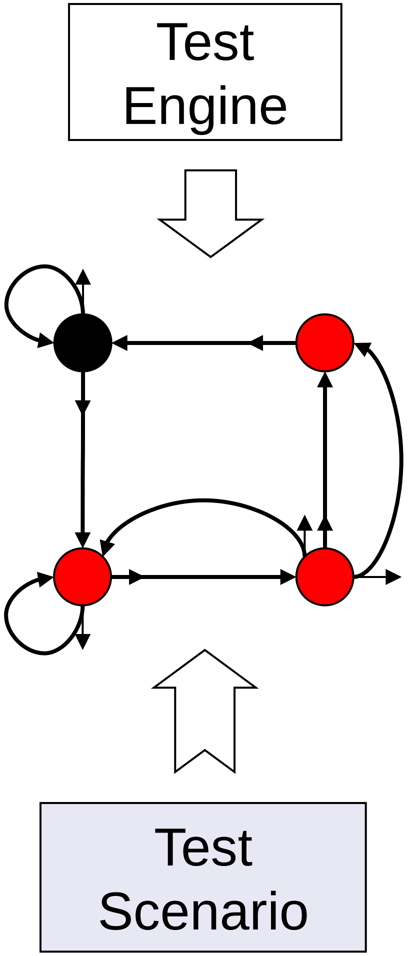
Specification
Model of coverage

Model of test
(test scenario)

Output analysis
Trace analysis

Test stimuli generator

UniTESK Test Scenario Model



Test Engine

Test Scenario

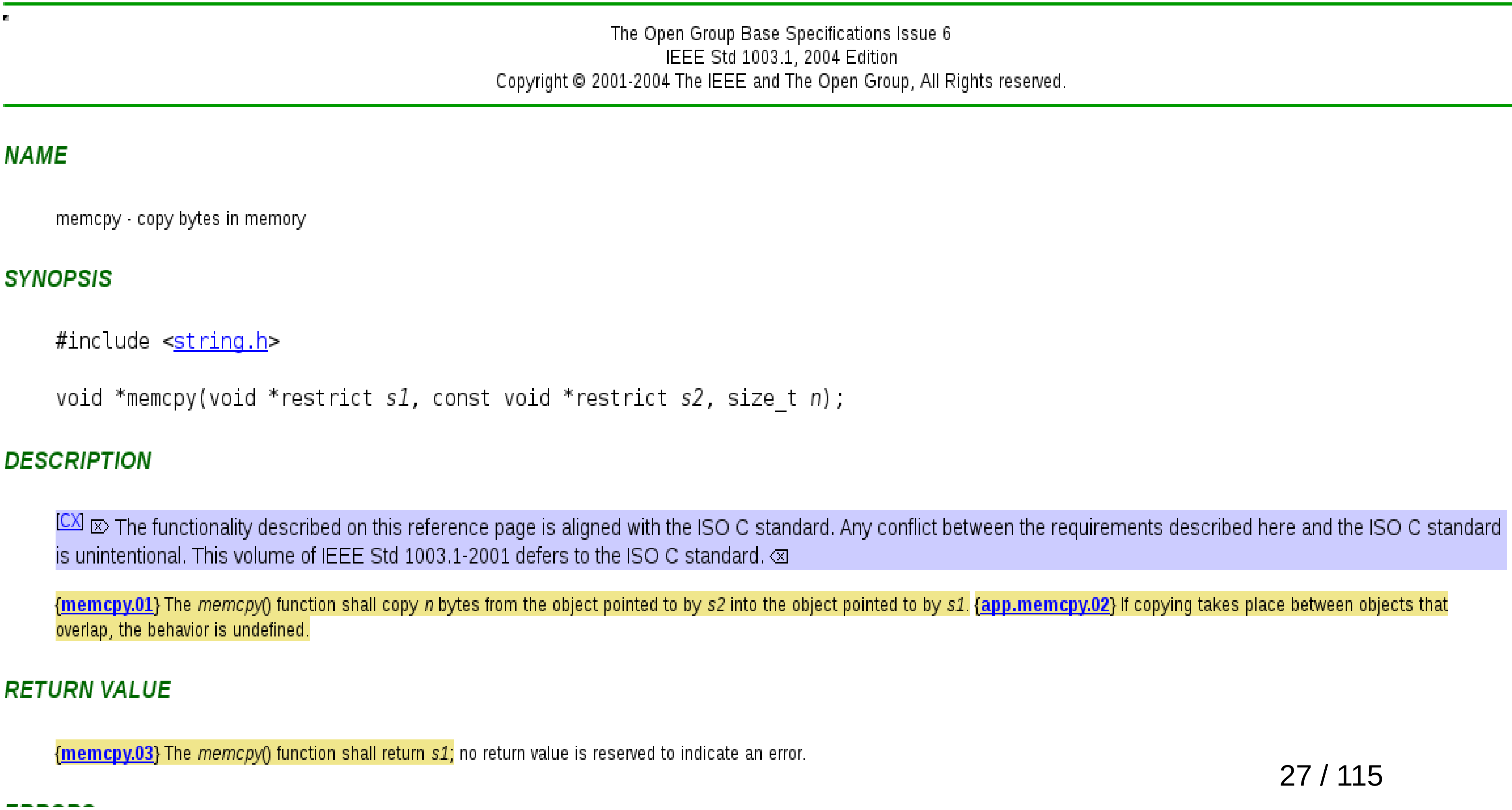
So called “Implicit automata” or
EFSM derived during on-the-fly test scenario execution.

Implicit automata is an ADT with
2 operations:

- recognise_node_ID () -> ({new, visited} x ID)
- next_call (next_input_stimulus) -> (…)

The test engine step by step builds/explores all nodes (states) and all available function calls (transitions).

Requirements Catalogue



The Open Group Base Specifications Issue 6
IEEE Std 1003.1, 2004 Edition
Copyright © 2001-2004 The IEEE and The Open Group, All Rights reserved.

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;
    }
}
```

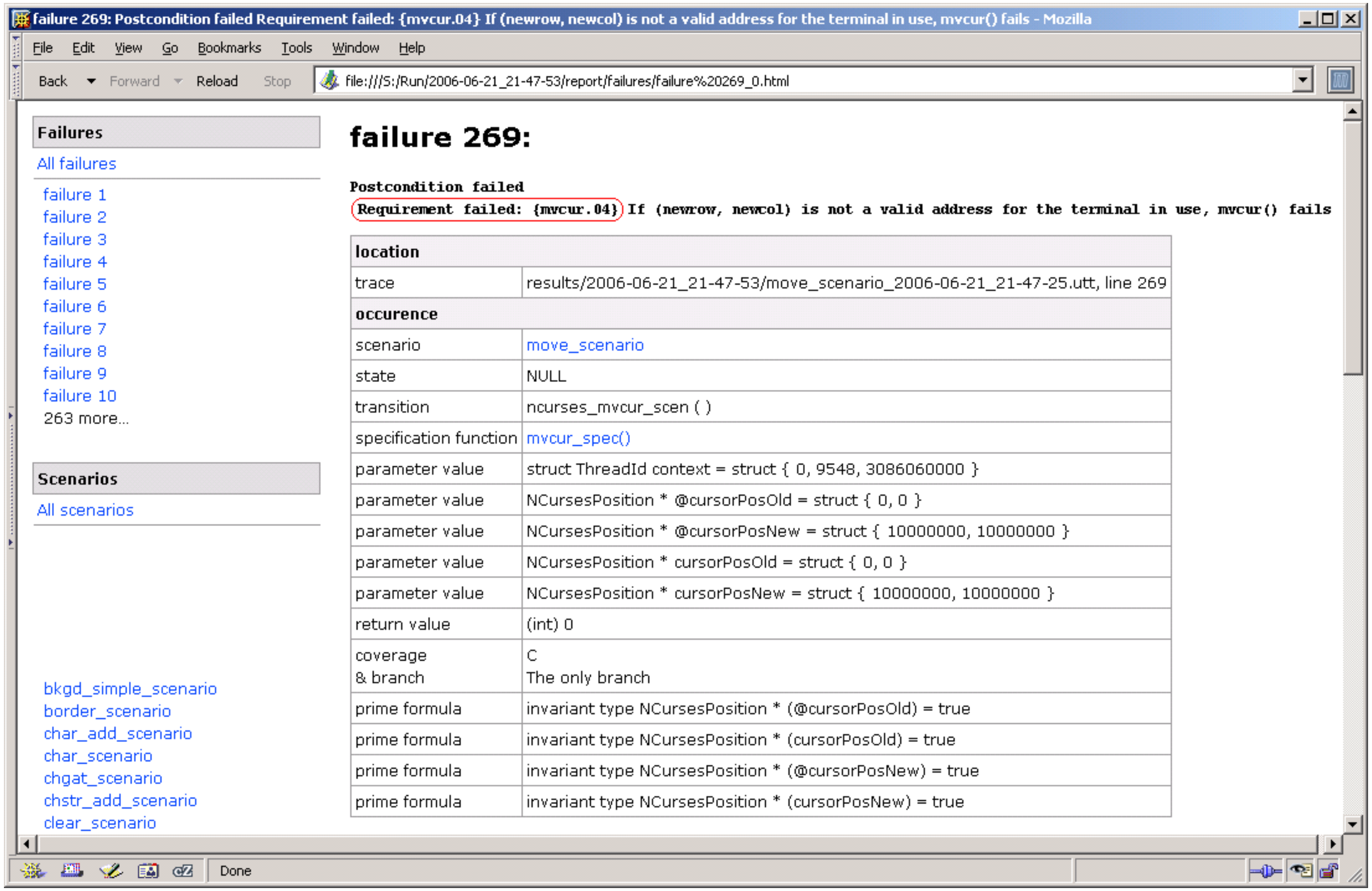
memcpy() precondition

```
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;
    }
```

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;
    }
```

Requirements Traceability


failure 269: Postcondition failed Requirement failed: {mvcur.04} If (newrow, newcol) is not a valid address for the terminal in use, mvcur() fails - Mozilla
File
Edit
View
Go
Bookmarks
Tools
Window
Help
Back
Forward
Reload
Stop
file:///S:/Run/2006-06-21_21-47-53/report/failures/failure%20269_0.html
Done

Failures

All failures

- failure 1
- failure 2
- failure 3
- failure 4
- failure 5
- failure 6
- failure 7
- failure 8
- failure 9
- failure 10
- 263 more...

Scenarios

All scenarios

- bkgd_simple_scenario
- border_scenario
- char_add_scenario
- char_scenario
- chgat_scenario
- chstr_add_scenario
- clear_scenario

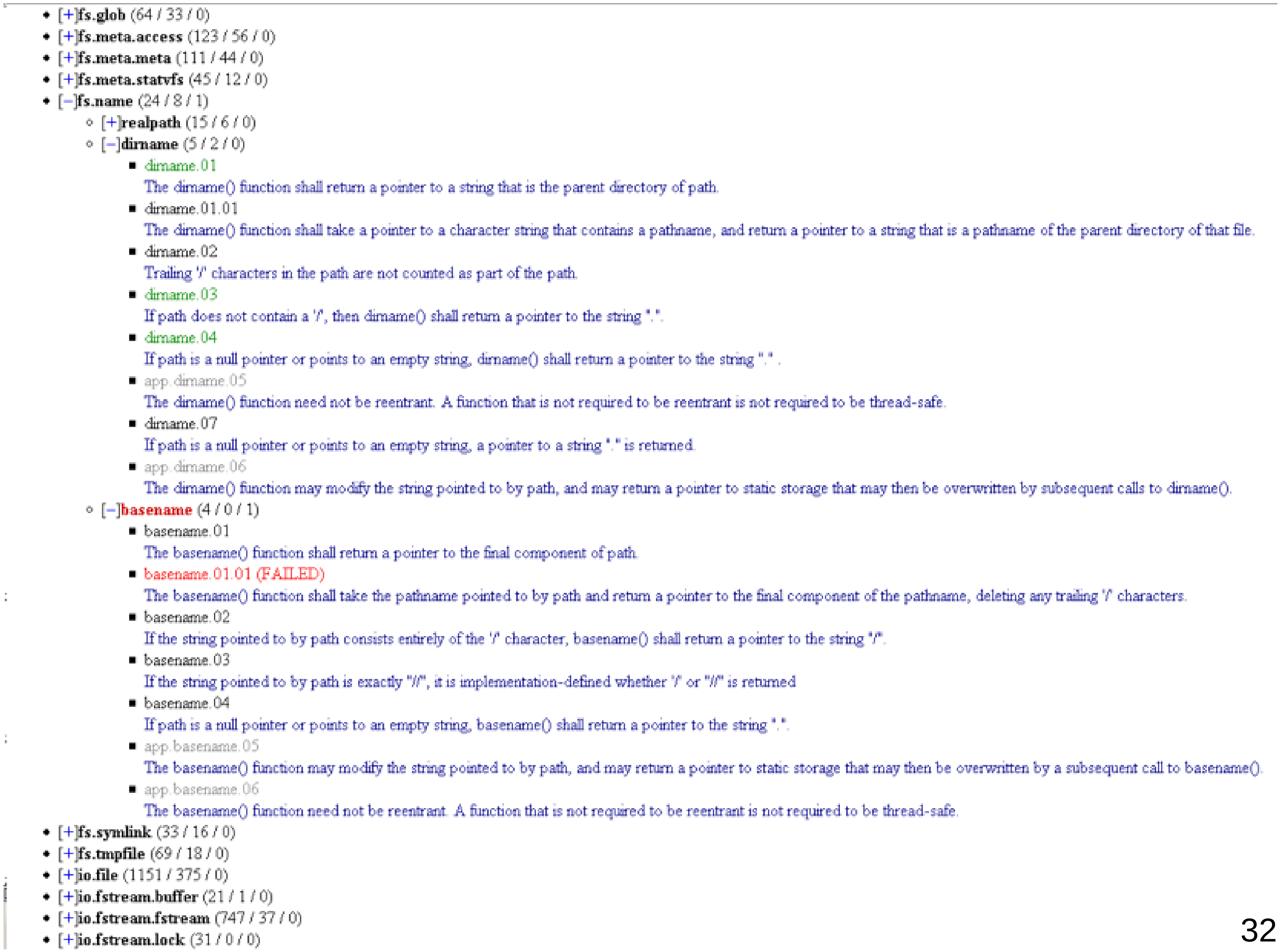
failure 269:

Postcondition failed
Requirement failed: {mvcur.04} If (newrow, newcol) is not a valid address for the terminal in use, mvcur() fails

location	
trace	results/2006-06-21_21-47-53/move_scenario_2006-06-21_21-47-25.utt, line 269
occurence	
scenario	move_scenario
state	NULL
transition	ncurses_mvcur_scen ( )
specification function	mvcur_spec()
parameter value	struct ThreadId context = struct { 0, 9548, 3086060000 }
parameter value	NCursesPosition * @cursorPosOld = struct { 0, 0 }
parameter value	NCursesPosition * @cursorPosNew = struct { 10000000, 10000000 }
parameter value	NCursesPosition * cursorPosOld = struct { 0, 0 }
parameter value	NCursesPosition * cursorPosNew = struct { 10000000, 10000000 }
return value	(int) 0
coverage & branch	C The only branch
prime formula	invariant type NCursesPosition * (@cursorPosOld) = true
prime formula	invariant type NCursesPosition * (cursorPosOld) = true
prime formula	invariant type NCursesPosition * (@cursorPosNew) = true
prime formula	invariant type NCursesPosition * (cursorPosNew) = true

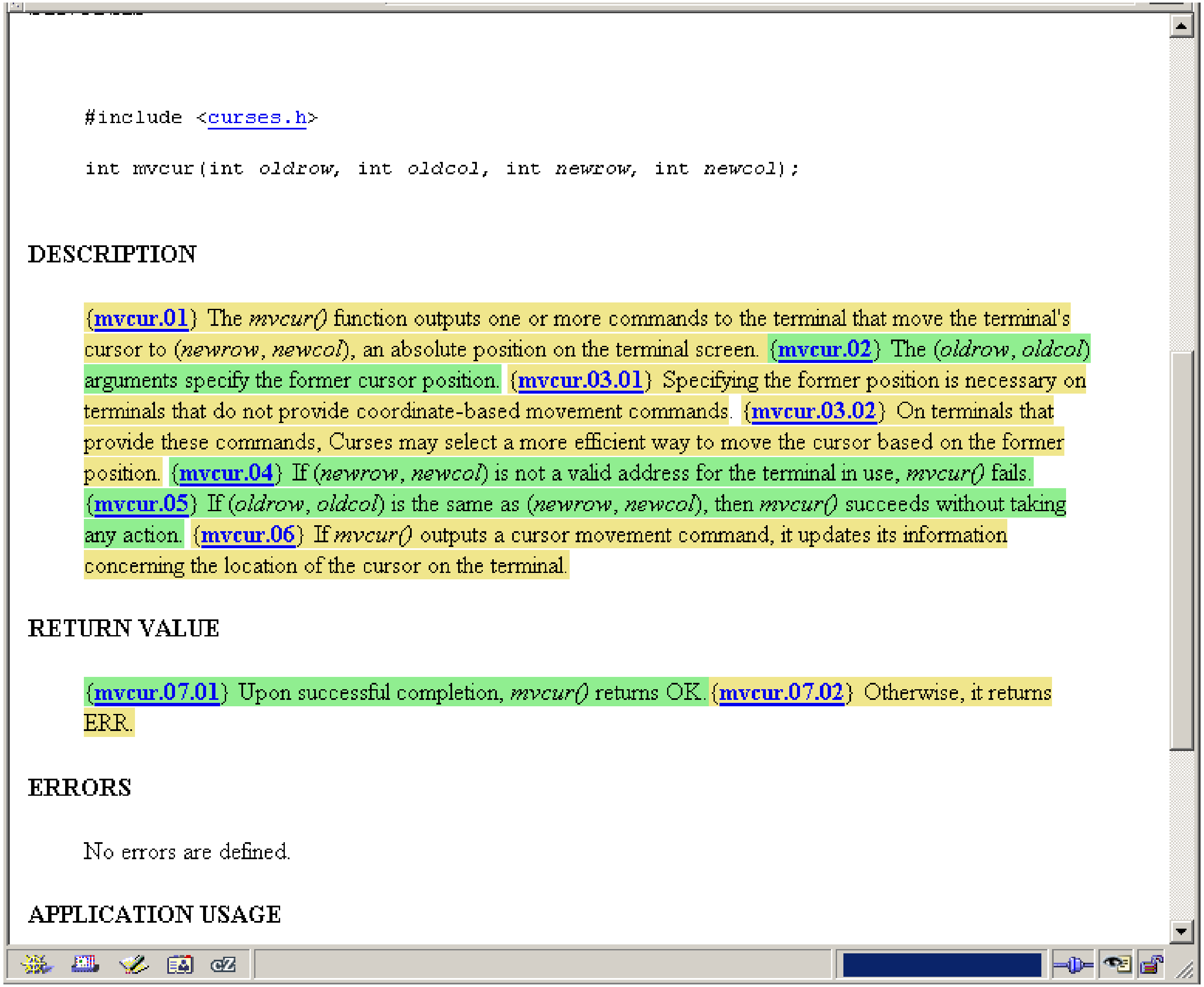
Failure report: requirement {mvcur.04} failed

Requirements Coverage Report



- [+]**fs.glob** (64 / 33 / 0)
- [+]**fs.meta.access** (123 / 56 / 0)
- [+]**fs.meta.meta** (111 / 44 / 0)
- [+]**fs.meta.statvfs** (45 / 12 / 0)
- [−]**fs.name** (24 / 8 / 1)
  - [+]**realpath** (15 / 6 / 0)
  - [−]**dirname** (5 / 2 / 0)
    - dirname.01
      The dirname() function shall return a pointer to a string that is the parent directory of path.
    - dirname.01.01
      The dirname() function shall take a pointer to a character string that contains a pathname, and return a pointer to a string that is a pathname of the parent directory of that file.
    - dirname.02
      Trailing '/' characters in the path are not counted as part of the path.
    - dirname.03
      If path does not contain a '/', then dirname() shall return a pointer to the string ".".
    - dirname.04
      If path is a null pointer or points to an empty string, dirname() shall return a pointer to the string "." .
    - app.dirname.05
      The dirname() function need not be reentrant. A function that is not required to be reentrant is not required to be thread-safe.
    - dirname.07
      If path is a null pointer or points to an empty string, a pointer to a string "." is returned.
    - app.dirname.06
      The dirname() function may modify the string pointed to by path, and may return a pointer to static storage that may then be overwritten by subsequent calls to dirname().
  - [−]**basename** (4 / 0 / 1)
    - basename.01
      The basename() function shall return a pointer to the final component of path.
    - basename.01.01 (FAILED)
      The basename() function shall take the pathname pointed to by path and return a pointer to the final component of the pathname, deleting any trailing '/' characters.
    - basename.02
      If the string pointed to by path consists entirely of the '/' character, basename() shall return a pointer to the string "/".
    - basename.03
      If the string pointed to by path is exactly "//", it is implementation-defined whether '/' or "//" is returned
    - basename.04
      If path is a null pointer or points to an empty string, basename() shall return a pointer to the string ".".
    - app.basename.05
      The basename() function may modify the string pointed to by path, and may return a pointer to static storage that may then be overwritten by a subsequent call to basename().
    - app.basename.06
      The basename() function need not be reentrant. A function that is not required to be reentrant is not required to be thread-safe.
- [+]**fs.symlink** (33 / 16 / 0)
- [+]**fs.tmpfile** (69 / 18 / 0)
- [+]**io.file** (1151 / 375 / 0)
- [+]**io.fstream.buffer** (21 / 1 / 0)
- [+]**io.fstream.fstream** (747 / 37 / 0)
- [+]**io.fstream.lock** (31 / 0 / 0)

Requirements Coverage Report (2)



```
#include <curses.h>

int mvcur(int oldrow, int oldcol, int newrow, int newcol);
```

DESCRIPTION

{mvcur.01} The *mvcur()* function outputs one or more commands to the terminal that move the terminal's cursor to (*newrow*, *newcol*), an absolute position on the terminal screen. {mvcur.02} The (*oldrow*, *oldcol*) arguments specify the former cursor position. {mvcur.03.01} Specifying the former position is necessary on terminals that do not provide coordinate-based movement commands. {mvcur.03.02} On terminals that provide these commands, Curses may select a more efficient way to move the cursor based on the former position. {mvcur.04} If (*newrow*, *newcol*) is not a valid address for the terminal in use, *mvcur()* fails. {mvcur.05} If (*oldrow*, *oldcol*) is the same as (*newrow*, *newcol*), then *mvcur()* succeeds without taking any action. {mvcur.06} If *mvcur()* outputs a cursor movement command, it updates its information concerning the location of the cursor on the terminal.

RETURN VALUE

{mvcur.07.01} Upon successful completion, *mvcur()* returns OK. {mvcur.07.02} Otherwise, it returns ERR.

ERRORS

No errors are defined.

APPLICATION USAGE

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 Conclusion

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

OLVER Conclusion

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

OLVER Conclusion

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

OLVER Conclusion

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

Configuration Testing
Product Line Testing

State of the Art.
Methods and Tools. Testing

- 3 views on OS:
  - OS as API for applications
  - OS is an OS kernel
  - OS is a part of software/hardware platform
- OS is a part of software/hardware platform
  - **Problems**
  - Huge number of configurations
  - Unavailable hardware devices and lack of devices models
  - **Methods**
  - Ad-hoc ≡ proprietary know-how
  - Systematical reduction of target configurations

ISP RAS *V.V. Kuliamin. Combinatorial generation of software-based OS configurations. The Proceedings of ISP RAS], 2012.*

  - **Tools**
  - No commercial or popular tool
  - **Testing quality**
  - Not available

Linux Product Line Verification

- University of Waterloo
  - Y. Xiong, A. Hubaux, S. She, and K. Czarnecki, “Generating range fixes for software configuration,” in Proc. of ICSE, 2012.
- University of Passau
  - Sven Apel, Alexander von Rhein, Philipp Wendler, Armin Größlinger, and Dirk Beyer. Strategies for Product-Line Verification: Case Studies and Experiments. In *Proc. of ICSE*, 2013.

OS Kernel Testing/Verification

State of the Art.
Methods and Tools. Testing

- 3 views on OS:
  - OS as API for applications
  - OS is an OS kernel
  - OS is a part of software/hardware platform
- OS is a kernel
  - **Problems**
  - Event driven multithreading systems
  - Lack of specifications (poor quality of specifications, Microsoft Windows is an exclusion)
  - **Methods**
  - Run-time verification
  - Fault simulation

ISP RAS ***Linux Kernel Testing (KEDR):*** ***http://code.google.com/p/kedr***

  - **Tools**
  - No commercial or popular tool applicable in kernel mode
  - **Testing quality**
  - Average test coverage lower 20%

Run-Time Verification

Sanitizer Tools Family.

Google research group of Konstantin Serebryany(*)

Run-time verification and compile-time code instrumentation.

Tools:

- MemorySanitizer: fast detector of uninitialized memory use in C++
- AddressSanitizer: A Fast Address Sanity Checker
- Dynamic Race Detection with LLVM Compiler
- ThreadSanitizer – data race detection
- KernelThreadSanitizer – data races in Linux Kernel

(*) http://research.google.com/pubs/KonstantinSerebryany.html

Robustness Testing

Fault Handling Code

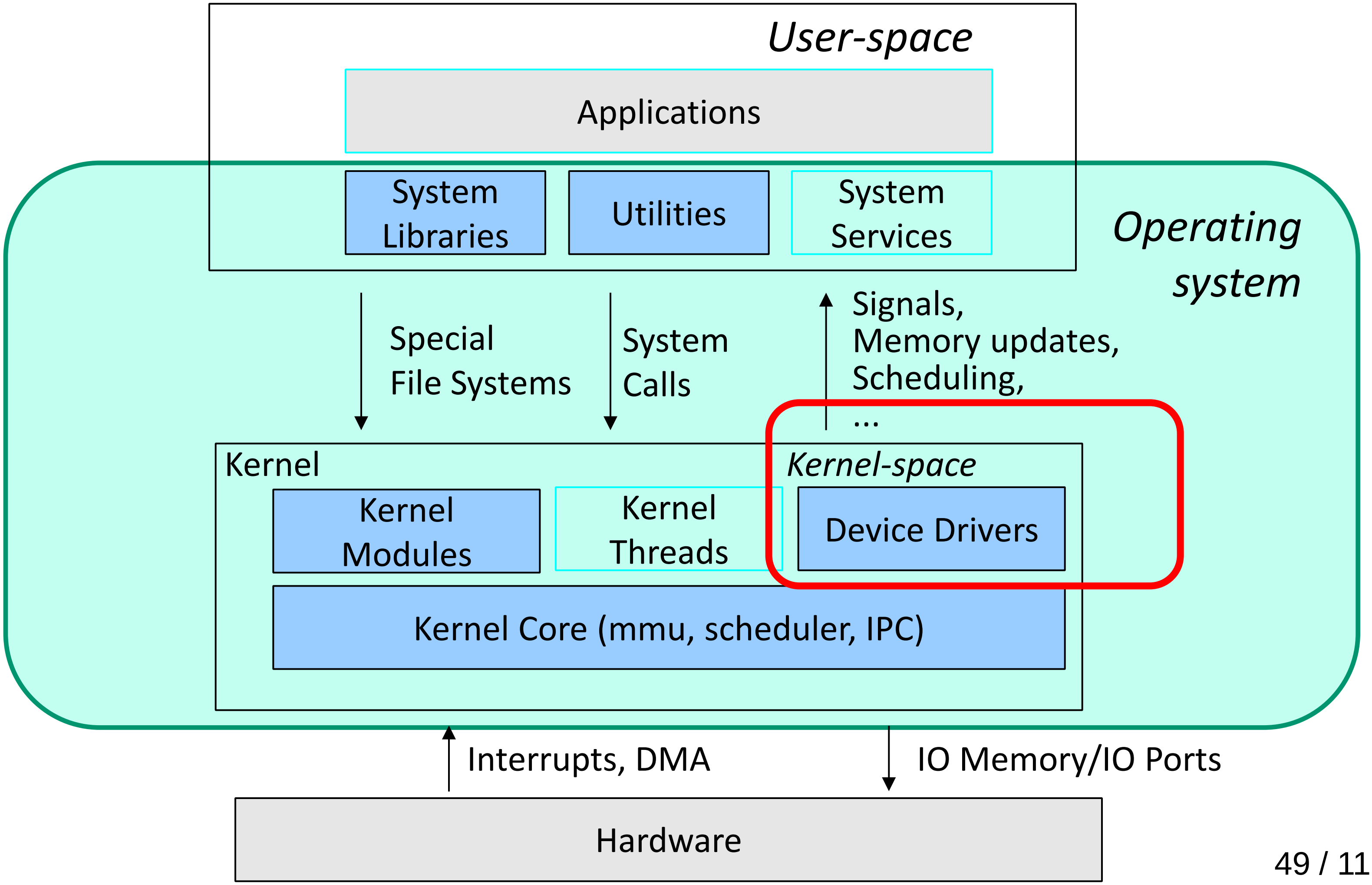
- Is not so fun
- Is really hard to keep all details in mind
- Practically is not tested
- Is hard to test even if you want to
- **Bugs seldom(never) occurs**

  => low pressure to care

Why do we care?

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

Operating Systems Structure



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

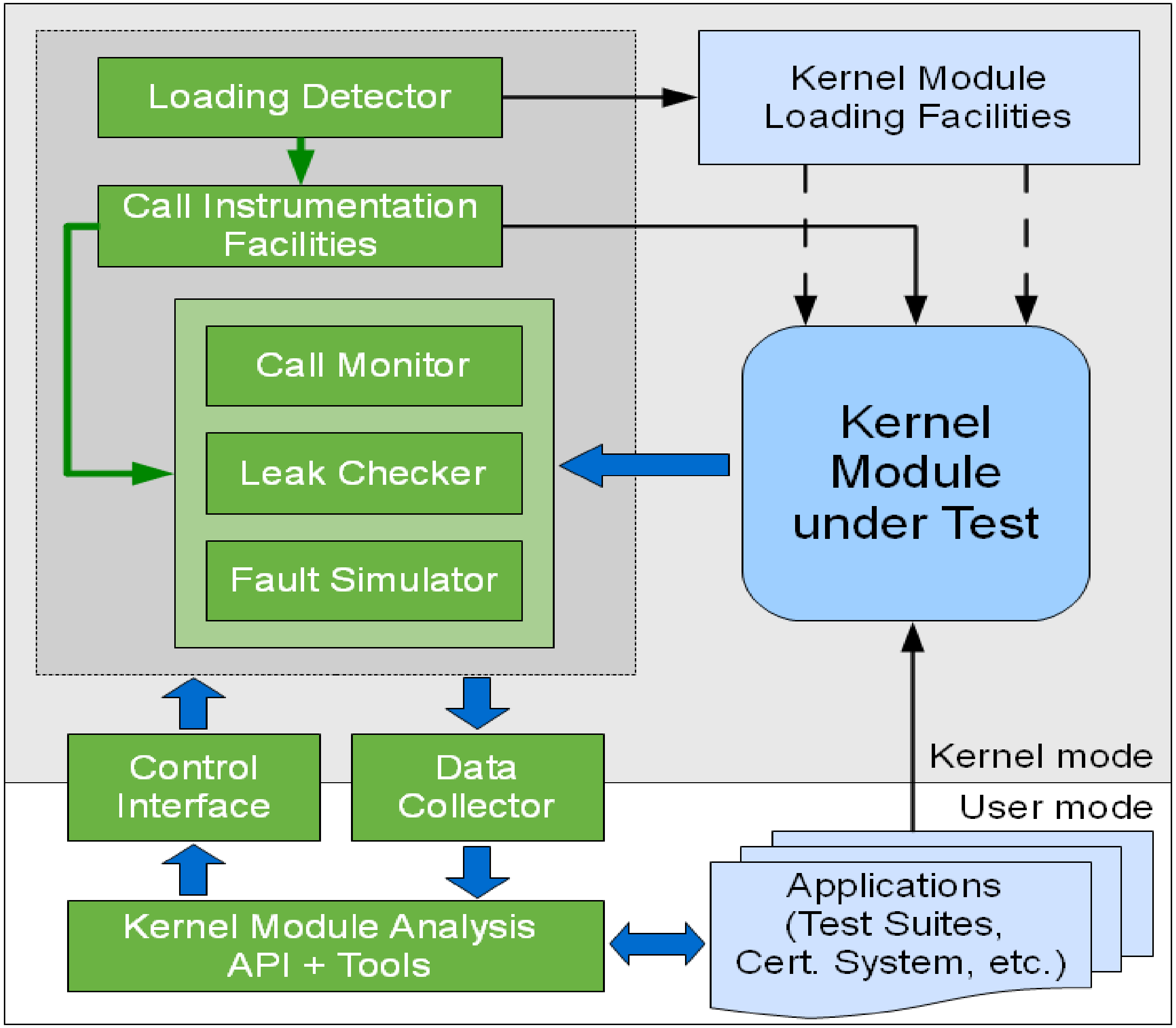
- Hypothesis:
  - Existing tests lead to more-or-less deterministic control flow in kernel code
- Idea:
  - Execute existing tests and collect all potential fault points in kernel code
  - **Systematically** enumerate the points and inject faults there

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

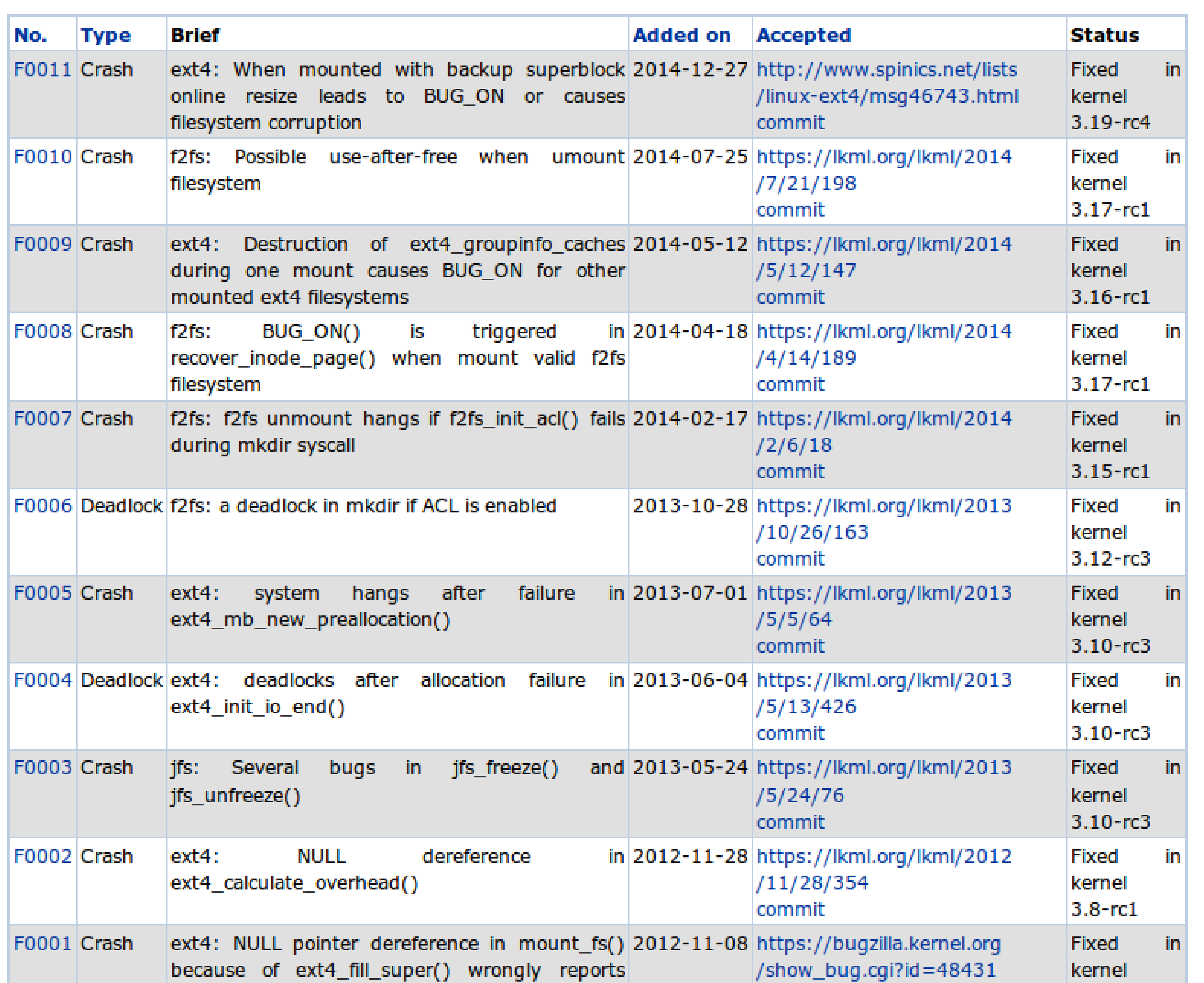

Loading Detector
Kernel Module Loading Facilities
Call Instrumentation Facilities
Call Monitor
Leak Checker
Fault Simulator
Kernel Module under Test
Kernel mode
User mode
Control Interface
Data Collector
Kernel Module Analysis API + Tools
Applications (Test Suites, Cert. System, etc.)

http://linuxtesting.org/project/kedr

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

Concolic Testing

Concolic Testing

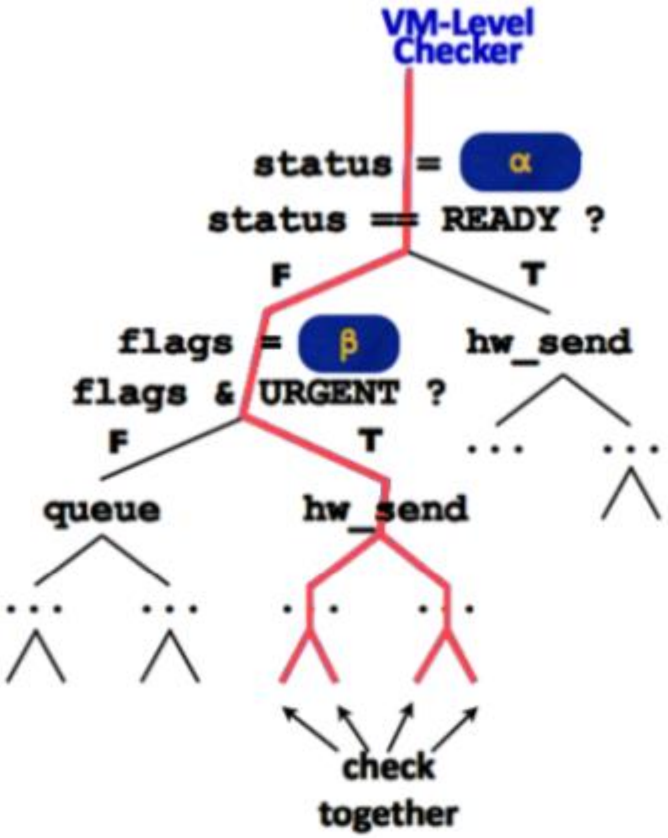
- Concolic = Symbolic + Concrete
  - SUT runs in concrete and in symbolic modes
  - Symbolic execution is used to collect conditions and branches of the current path
  - Collected data is used to generate new input data to cover more execution paths

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 for Kernel Testing

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



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

Testing Aspects

	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH	KStrider
Monitoring Aspects				-	-	+	+-	+	+-
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									+

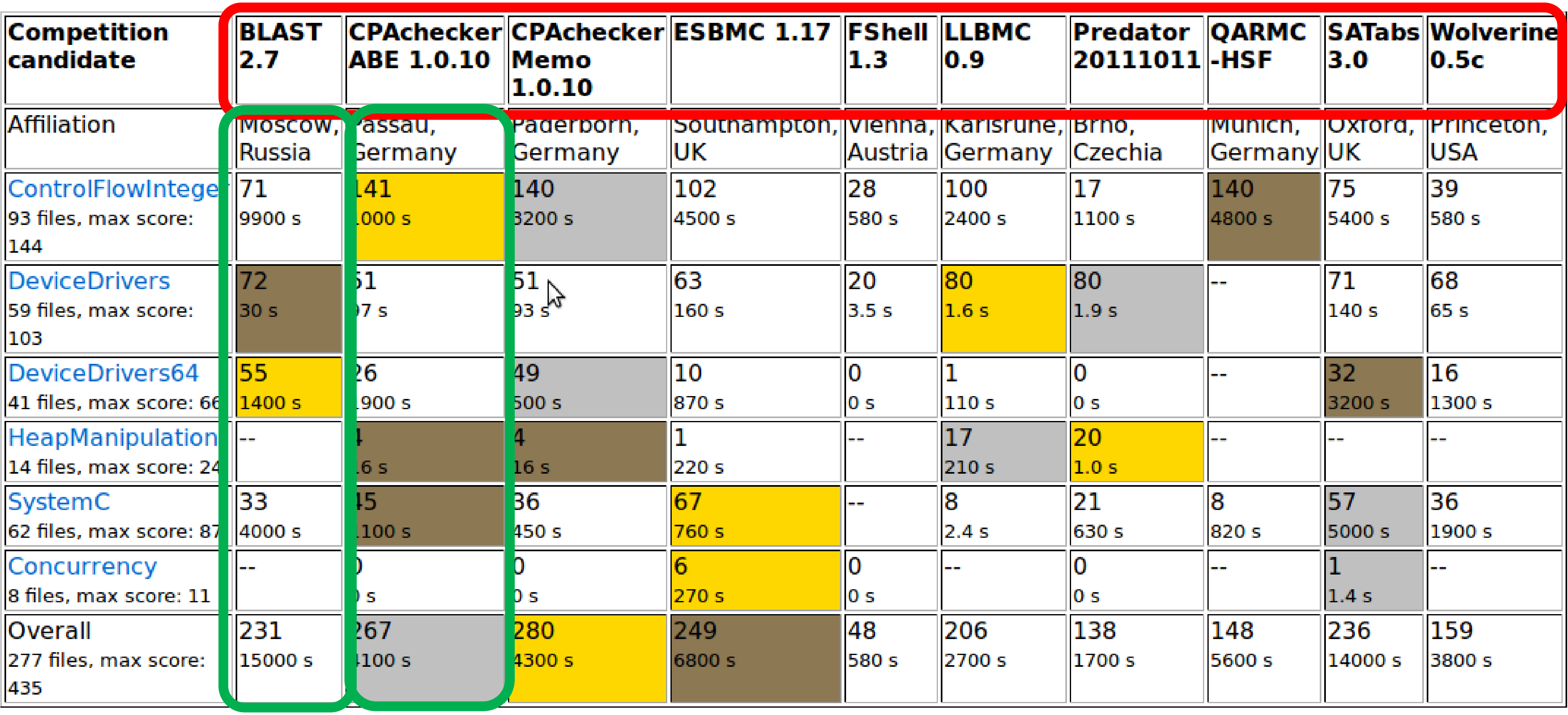
	T2C	OLVER	Autotest	Cfg	FI	KEDR-LC	S2E	RH	KStrider
Active Aspects				+-	+	-	+	+	-
Target Test Situations Set				cfgs				Specific	
requirements coverage	+	+							
class equivalence coverage		+							
model coverage (SUT/reqs)		+							
source code coverage					almost		+		
Test Situations Setup/Set Gen									
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							+		

Software Model Checking

State of the Art. Methods and Tools. Software Model Checking

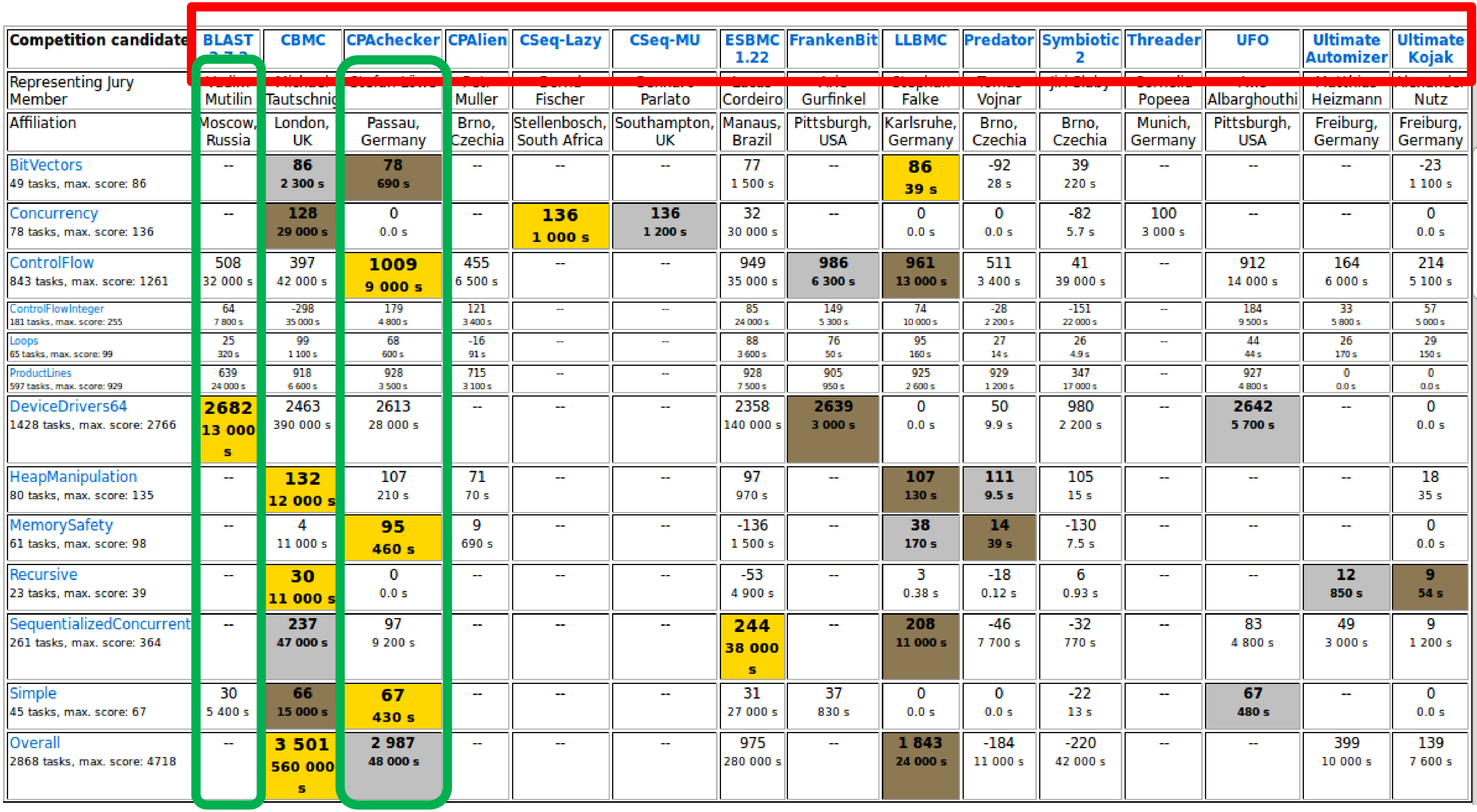
- Approaches:
  - Counterexample guided abstraction refinement (CEGAR) - Edmund Clarke et al.
  - Configurable Program Analysis – Dirk Beyer
  - Abstract interpretation - Patrick Cousot and Radhia Cousot
  - Bounded Model Checking – BMC – Edmund Clarke et al.
- Gold practices
  - Microsoft Research (SLAM)
  - ***LDV – Linux Driver Verification***
- Problems
  - Lack of specs
  - Limitations on size and complexity of modules (no more 30-100KLine)
- Tools
  - Many but no commercial or popular tool
- Verification quality

SVCOMP‘2012 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	assau, Germany	Paderborn, Germany	Southampton, UK	Vienna, Austria	Karlsruhe, Germany	Brno, Czechia	Munich, Germany	Oxford, UK	Princeton, USA
ControlFlowIntege 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

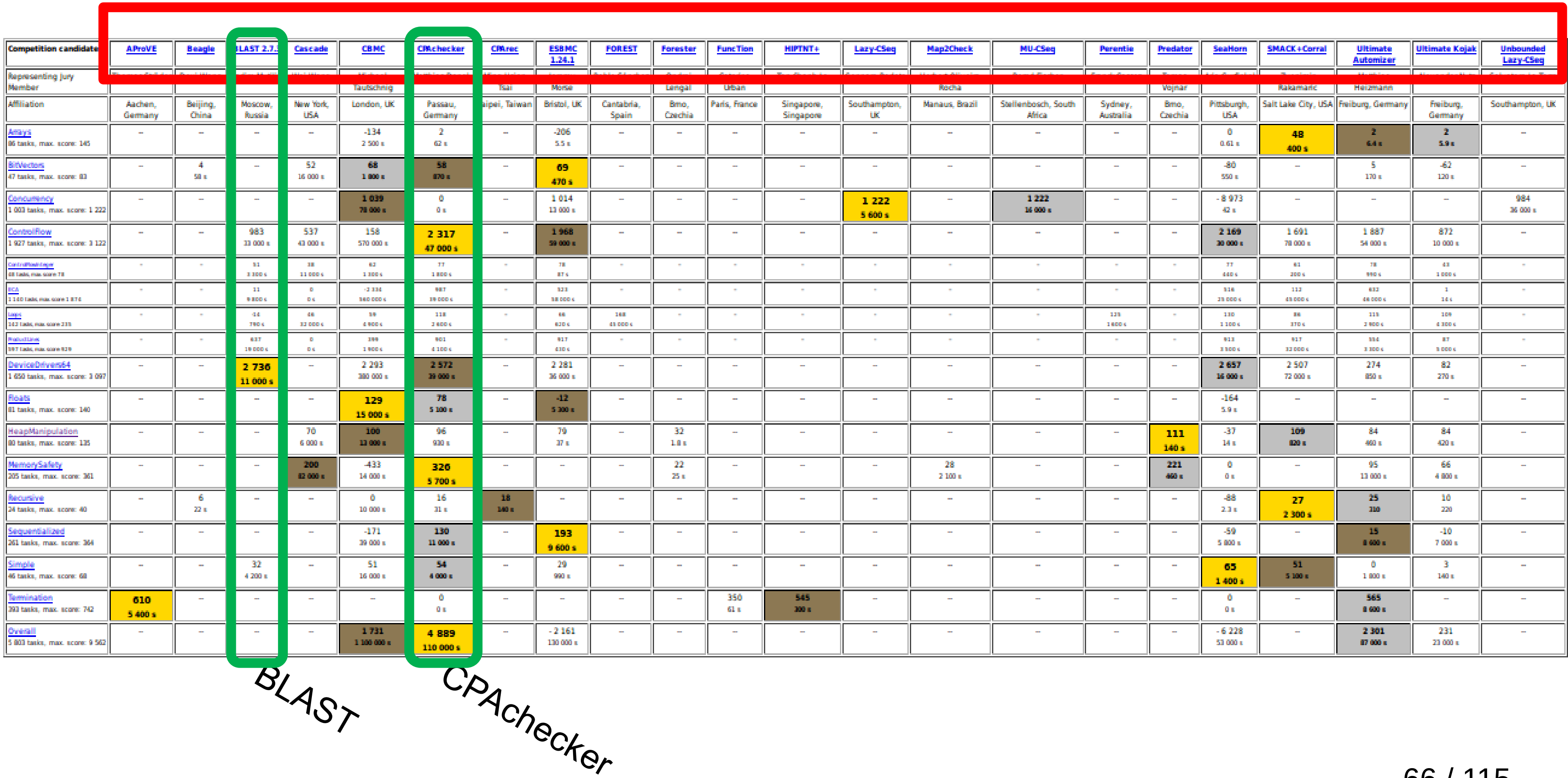
SVCOMP‘2014 Results



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

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

SVCOMP‘2015 Results



Competition candidate	AProVE	Beagle	BLAST 2.7.3	Cascade	CBMC	CPAchecker	CPArec	ESBMC 1.24.1	FOREST	Forester	FuncTion	HIPTNT+	Lazy-CSeg	Map2Check	MU-CSeg	Perentie	Predator	SeaHorn	SMACK+Corral	Ultimate Automizer	Ultimate Kojak	Unbounded Lazy-CSeg
Representing jury Member					Tautschnig		Tsai	Morse		Lengal	Urban			Rocha			Vojnar		Rakamaric	Heizmann		
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 78 000 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	–
ControlFlowInteger 48 tasks, max score 78	–	–	51 3 300 s	38 11 000 s	62 1 300 s	77 1 800 s	–	78 87 s	–	–	–	–	–	–	–	–	–	77 440 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 19 000 s	–	523 58 000 s	–	–	–	–	–	–	–	–	–	516 25 000 s	112 45 000 s	632 46 000 s	1 14 s	–
Loops 142 tasks, max score 235	–	–	-14 790 s	46 32 000 s	59 4 900 s	118 2 600 s	–	66 620 s	168 41 000 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 929	–	–	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

LDV: Linux Driver Verification

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

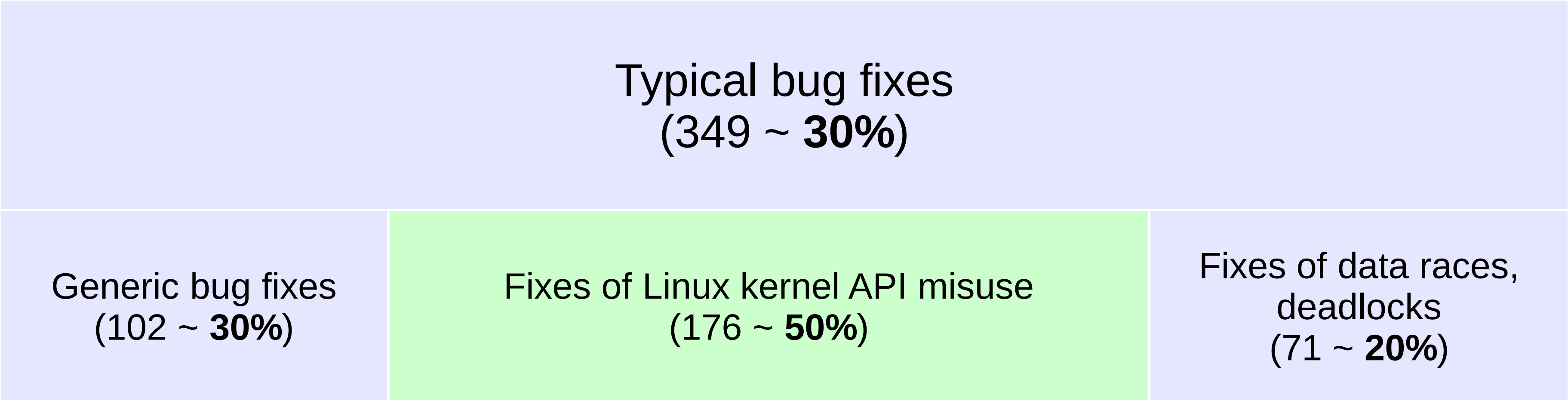
Commit Analysis

- 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%**)

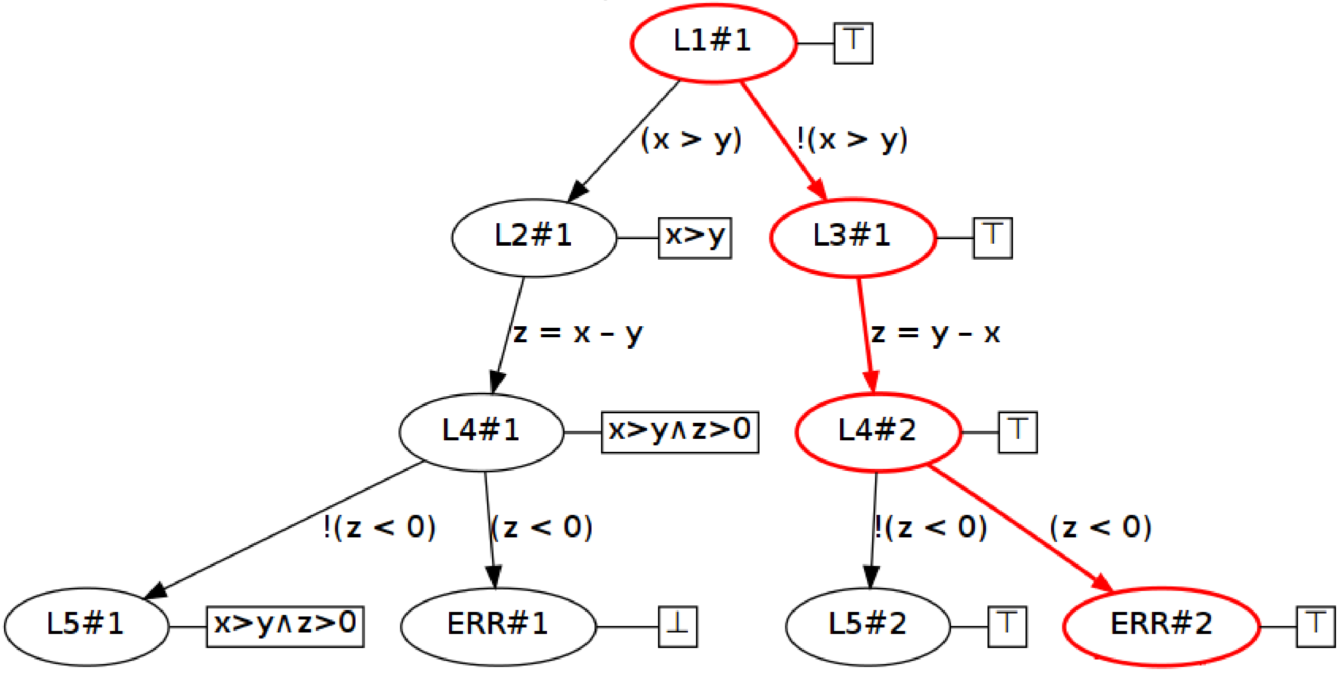
Fixes of data races, deadlocks
(71 ~ **20%**)

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

entry point



error location

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);
```

Callback interface procedures registration

No explicit calls to init/exit procedures

Driver Environment Model

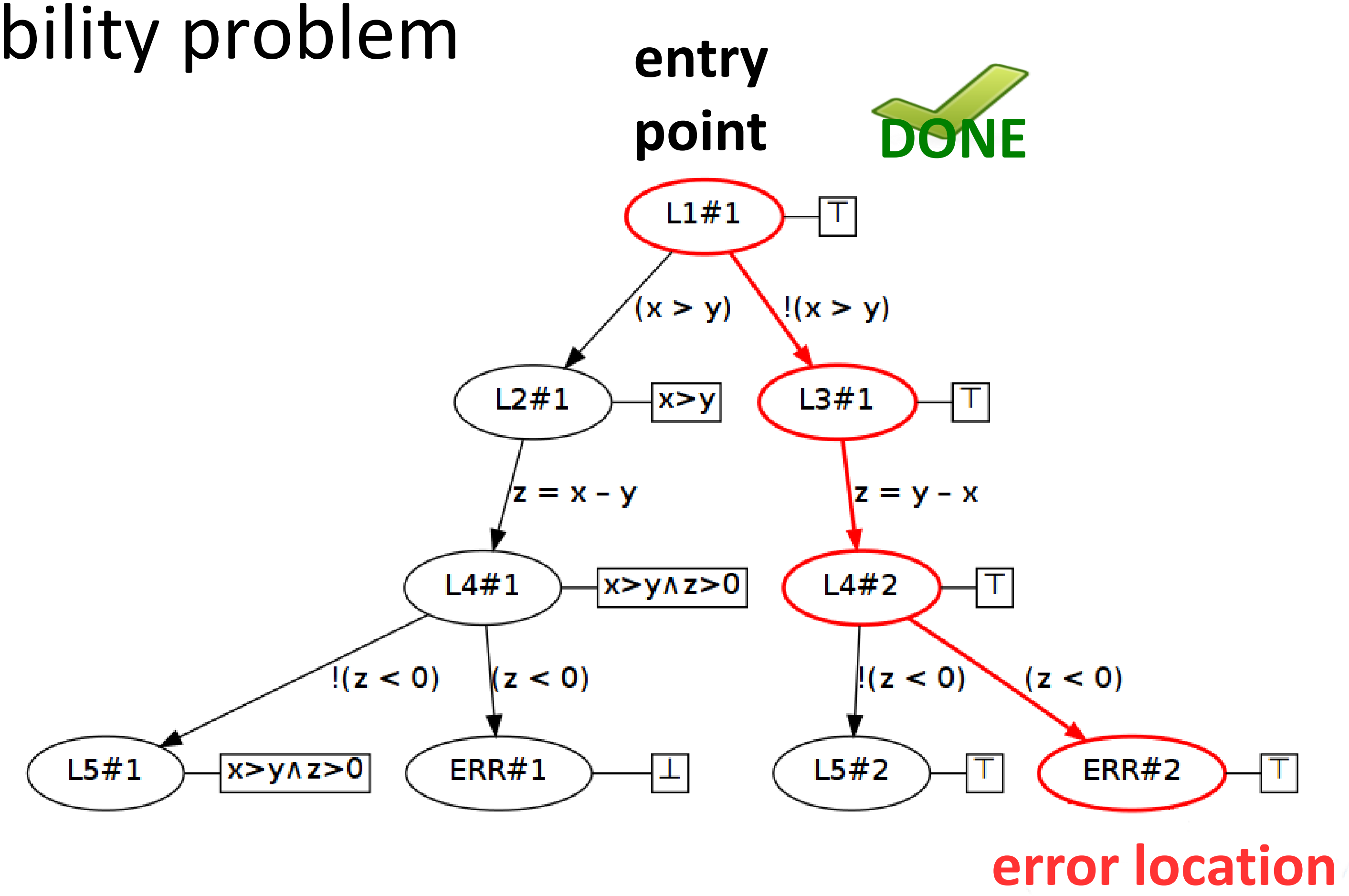
```
int main(int argc,char* argv[])
{
  usbpn_init()
  for(;;) {
    switch(*) {
     case 0: usbpn_probe(*,*,*);break;
     case 1: usbpn_open(*,*);break;
     ...
    }
  }
  usbpn_exit();
}
```

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



Instrumentation

- int f(int y)
- {
- struct urb *x;
- x = usb_alloc_urb(0,GFP_KERNEL);
- ...
- usb_free_urb(x);
- return y;
- }

➡

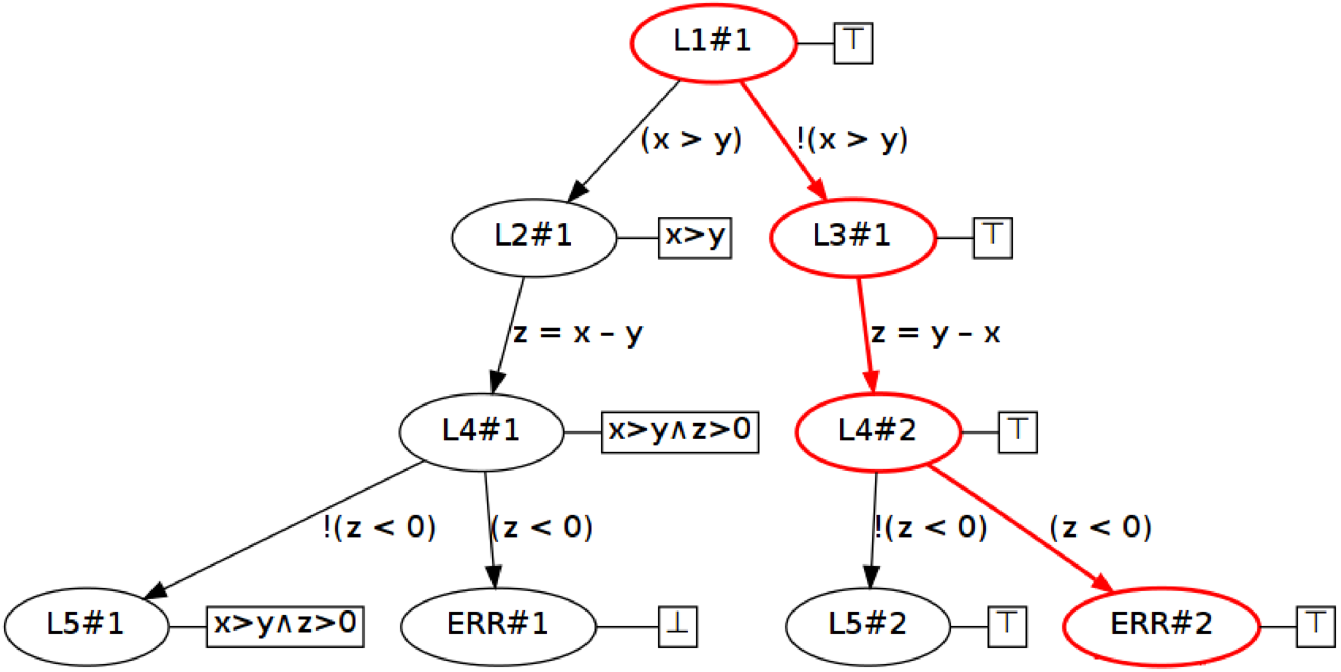
```
set URBS = empty;

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));
```

Model Checking and Linux Kernel

- Reachability problem

entry point

DONE

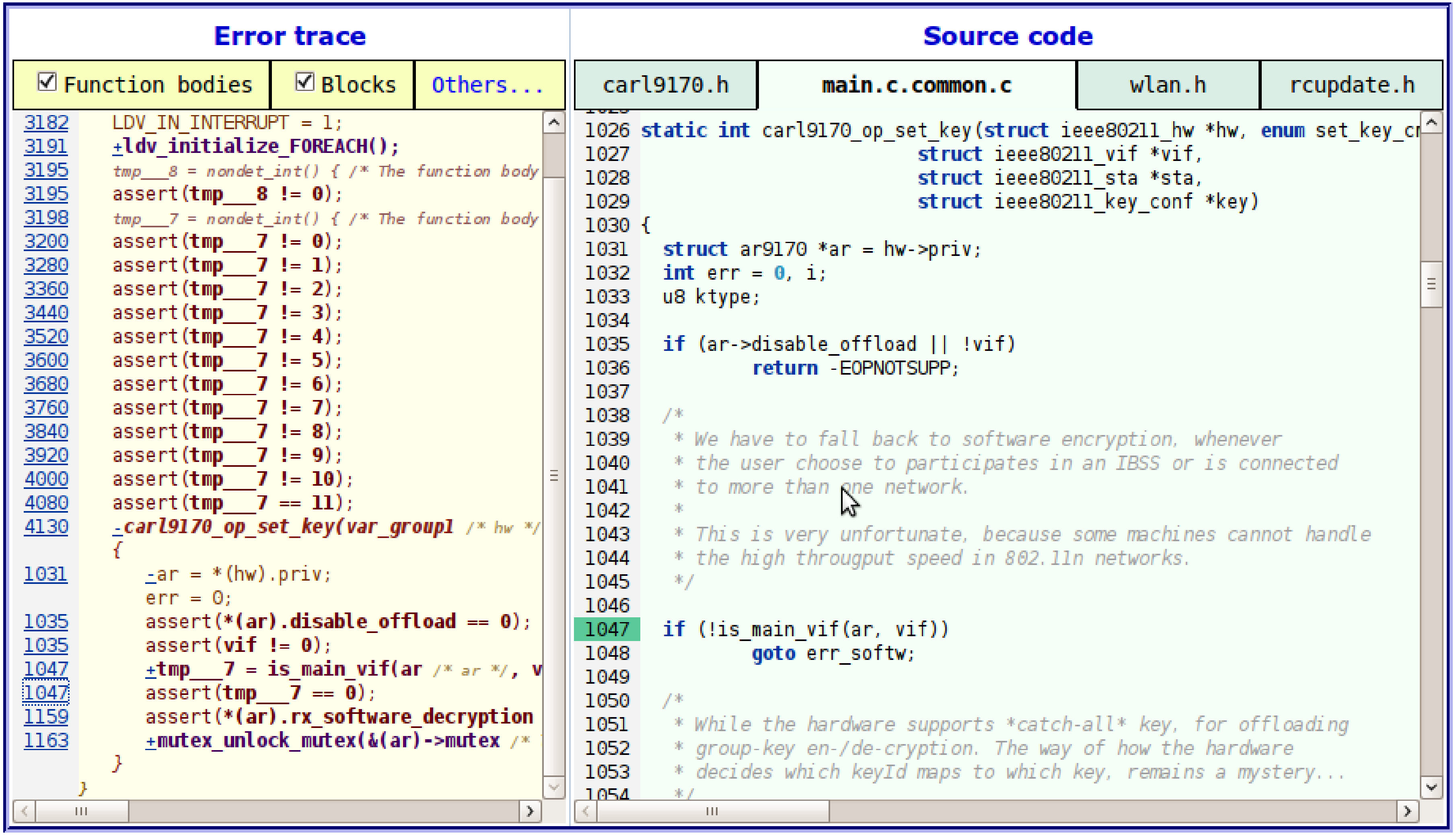


error location

DONE

Error Trace Visualizer

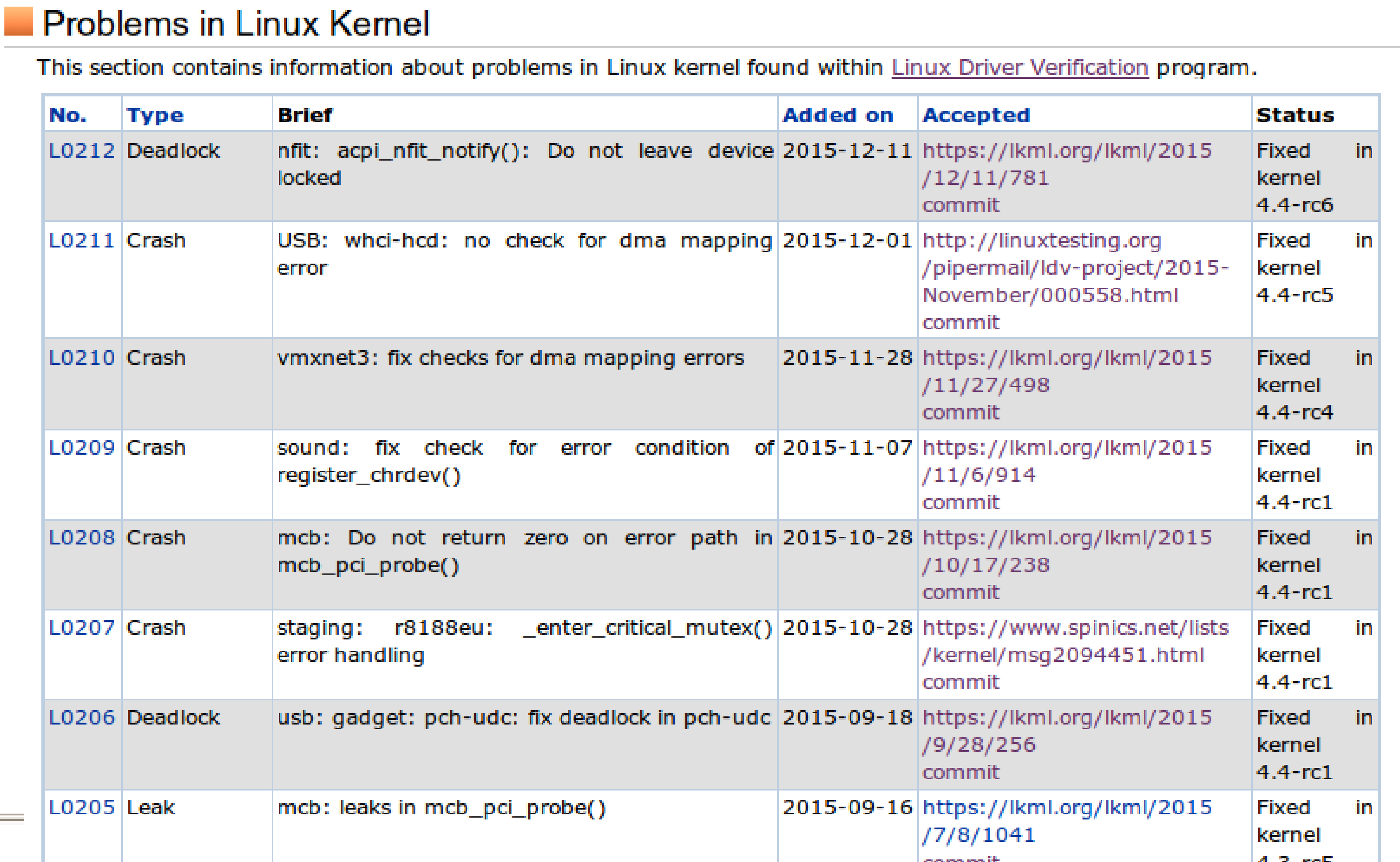
Rule: Mutex lock/unlock



```
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 /*
        }
    }
```

```
static int carl9170_op_set_key(struct ieee80211_hw *hw, enum set_key_cr
                               struct ieee80211_vif *vif,
                               struct ieee80211_sta *sta,
                               struct ieee80211_key_conf *key)
{
	struct ar9170 *ar = hw->priv;
	int err = 0, i;
	u8 ktype;

	if (ar->disable_offload || !vif)
		return -EOPNOTSUPP;

	/*
	 * We have to fall back to software encryption, whenever
	 * the user choose to participates in an IBSS or is connected
	 * to more than one network.
	 *
	 * This is very unfortunate, because some machines cannot handle
	 * the high througput speed in 802.11n networks.
	 */

	if (!is_main_vif(ar, vif))
		goto err_softw;

	/*
	 * While the hardware supports *catch-all* key, for offloading
	 * group-key en-/de-cryption. The way of how the hardware
	 * decides which keyId maps to which key, remains a mystery...
	 */
```

Bugs Found (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 commit	Fixed in kernel 4.3-rc5

Deductive Verification

State of the Art. Methods and Tools. Deductive Verification

- Approaches:
  - Design and verify an ideal “perfect” OS
  - Verify a critical component of real-life OS
- Gold practices
  - **L4 Kernel Verification**
    - Gerwin Klein. Operating System Verification — An Overview. 2009
  - **seL4**
    - Gerwin Klein, June Andronick, Kevin Elphinstone, Gernot Heiser, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt. seL4: Formal Verification of an Operating-System Kernel
  - **Verisoft OS**
    - HillebrandMA, PaulWJ. On the architecture of system verification environments. 2008.
  - **Verisoft + Microsoft Research – Pike OS, Hyper-V verification**
    - C. Baumann, B.Beckert, et al. Ingredients of Operating System Correctness. Lessons Learned in the Formal Verification of PikeOS
- Problems
  - Tools limitations and lack of module specifications, no frozen interfaces in Linux Kernel
- Tools
  - Many but no commercial or common used tool

Astraver Project

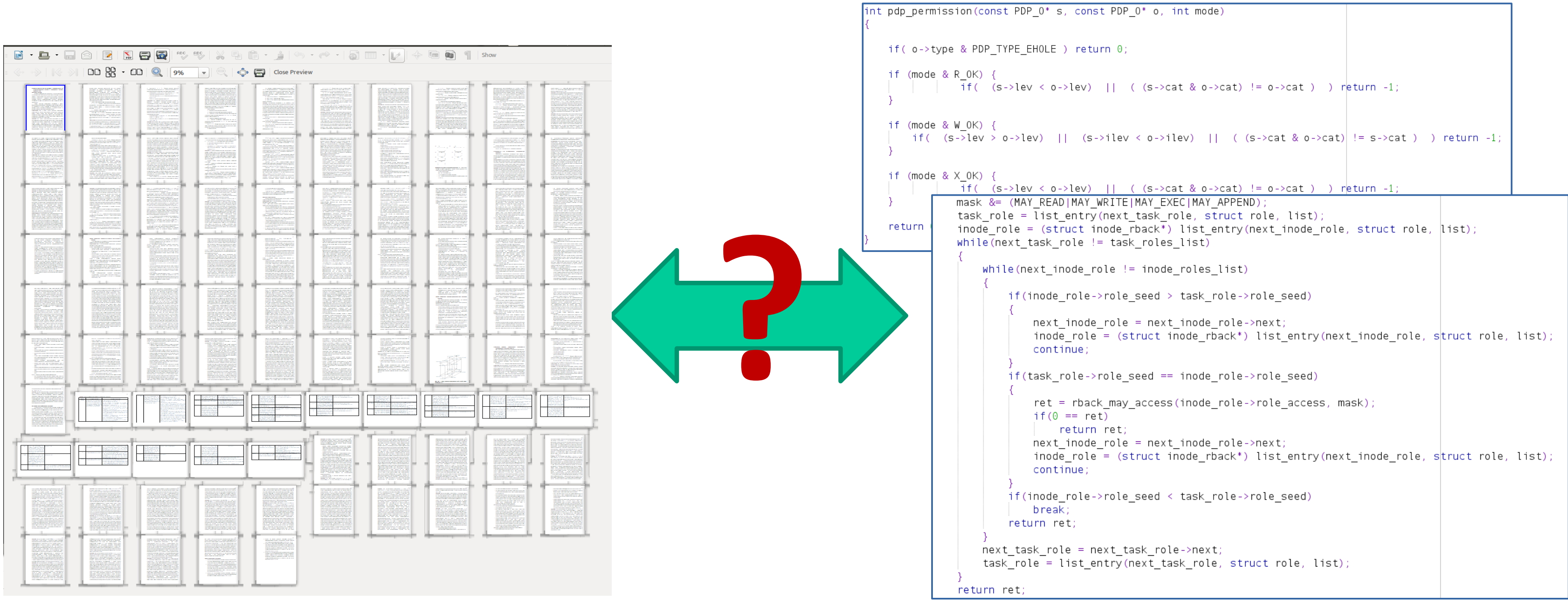
- Deductive Verification of Linux Security Module
  - Joint project with NPO RusBITech
  - Formal security model MROSL-DP
- Assumptions
  - Linux kernel core conforms with its specifications
    - It is not target to prove
- Code under verification
  - Code is hardware independent
  - Verification unfriendly

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

LSM Verification Project

LSM stands for Linux Security Module



```
int pdp_permission(const PDP_0* s, const PDP_0* o, int mode)
{

    if( o->type & PDP_TYPE_EHOLE ) return 0;

    if (mode & R_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    if (mode & W_OK) {
        if(  (s->lev > o->lev)  ||  (s->ilev < o->ilev)  ||  ( (s->cat & o->cat) != s->cat )  ) return -1;
    }

    if (mode & X_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    return
}
```

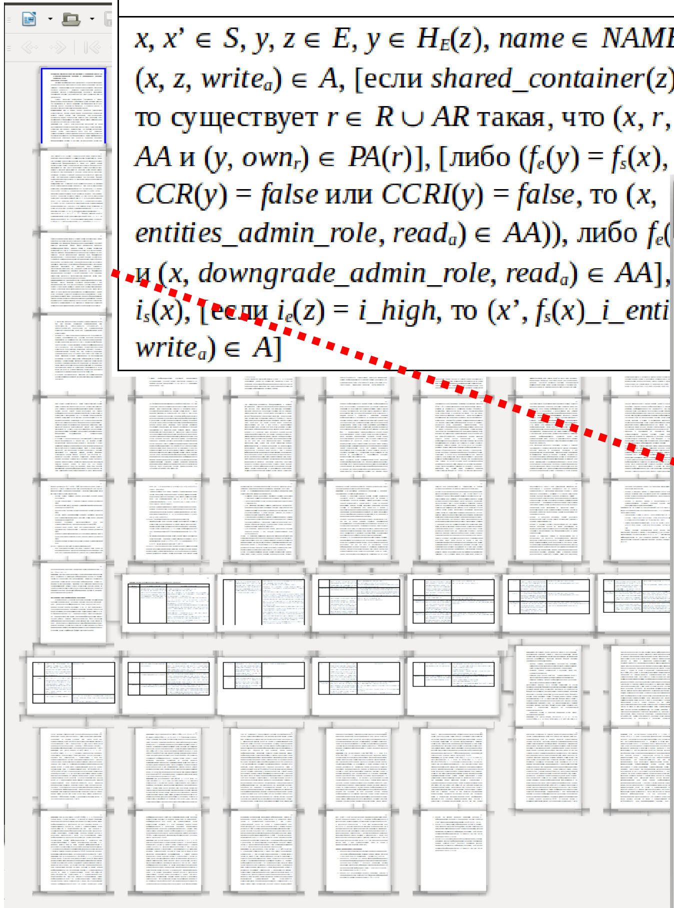
```
    mask &= (MAY_READ|MAY_WRITE|MAY_EXEC|MAY_APPEND);
    task_role = list_entry(next_task_role, struct role, list);
    inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
    while(next_task_role != task_roles_list)
    {
        while(next_inode_role != inode_roles_list)
        {
            if(inode_role->role_seed > task_role->role_seed)
            {
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(task_role->role_seed == inode_role->role_seed)
            {
                ret = rback_may_access(inode_role->role_access, mask);
                if(0 == ret)
                    return ret;
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(inode_role->role_seed < task_role->role_seed)
                break;
            return ret;
        }
        next_task_role = next_task_role->next;
        task_role = list_entry(next_task_role, struct role, list);
    }
    return ret;
```

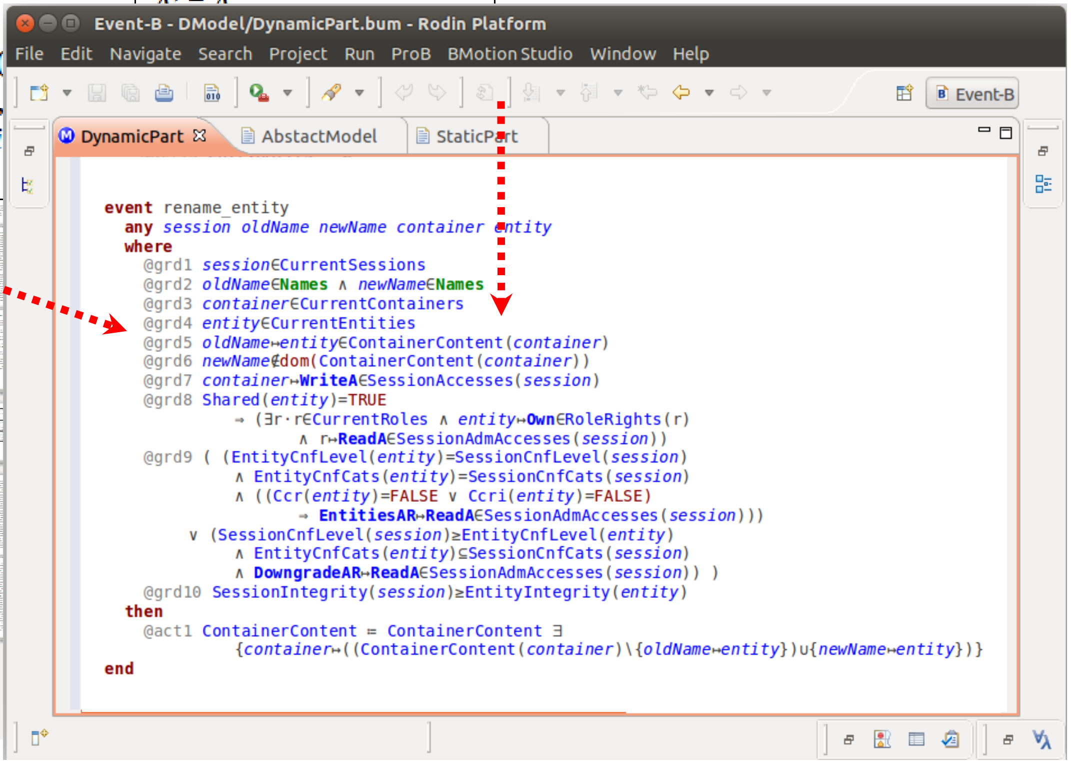
Security requirements in math notation (MROSL DP model integrates of RBAC, MIC and, MAC)

Implementation of LSM in Linux kernel

From Rigorous to Formal Security Model Requirements

rename_entity(x, x', y, name, z)	
$x, x' \in S, y, z \in E, y \in H_E(z), name \in NAME \setminus \{“”\}$, $(x, z, write_a) \in A$, [если $shared\_container(z) = true$, то существует $r \in R \cup AR$ такая, что $(x, r, read_a) \in AA$ и $(y, own_r) \in PA(r)$], [либо $(f_e(y) = f_s(x)$, если $CCR(y) = false$ или $CCRI(y) = false$, то $(x,$ $entities\_admin\_role, read_a) \in AA))$, либо $f_e($ и $(x, downgrade\_admin\_role, read_a) \in AA]$, $i_s(x)$, [если $i_e(z) = i\_high$, то $(x', f_s(x)\_i\_enti$ $write_a) \in A]$	$S' = S$, $E' = E$, $APA' = APA$, $PA' = PA$, $A' = A$




Event-B - DModel/DynamicPart.bum - Rodin Platform
File Edit Navigate Search Project Run ProB BMotion Studio Window Help
Event-B
DynamicPart
AbstactModel
StaticPart

```
event rename_entity
  any session oldName newName container entity
  where
    @grd1 session∈CurrentSessions
    @grd2 oldName∈Names ∧ newName∈Names
    @grd3 container∈CurrentContainers
    @grd4 entity∈CurrentEntities
    @grd5 oldName↦entity∈ContainerContent(container)
    @grd6 newName∉dom(ContainerContent(container))
    @grd7 container↦WriteA∈SessionAccesses(session)
    @grd8 Shared(entity)=TRUE
              ⇒ (∃r·r∈CurrentRoles ∧ entity↦Own∈RoleRights(r)
                    ∧ r↦ReadA∈SessionAdmAccesses(session))
    @grd9 ( (EntityCnfLevel(entity)=SessionCnfLevel(session)
              ∧ EntityCnfCats(entity)=SessionCnfCats(session)
              ∧ ((Ccr(entity)=FALSE ∨ Ccri(entity)=FALSE)
                    ⇒ EntitiesAR↦ReadA∈SessionAdmAccesses(session)))
          ∨ (SessionCnfLevel(session)≥EntityCnfLevel(entity)
              ∧ EntityCnfCats(entity)⊆SessionCnfCats(session)
              ∧ DowngradeAR↦ReadA∈SessionAdmAccesses(session)) )
    @grd10 SessionIntegrity(session)≥EntityIntegrity(entity)
  then
    @act1 ContainerContent ≔ ContainerContent ∃
              {container↦((ContainerContent(container)\{oldName↦entity})∪{newName↦entity})}
end
```

Example: *access_write*(*x*, *x’*, *y*) vs. Implementation

$x, x' \in S$,
$y \in E \cup R \cup AR$,

существует $r \in R \cup AR$: $(x, r, read_a) \in AA$,

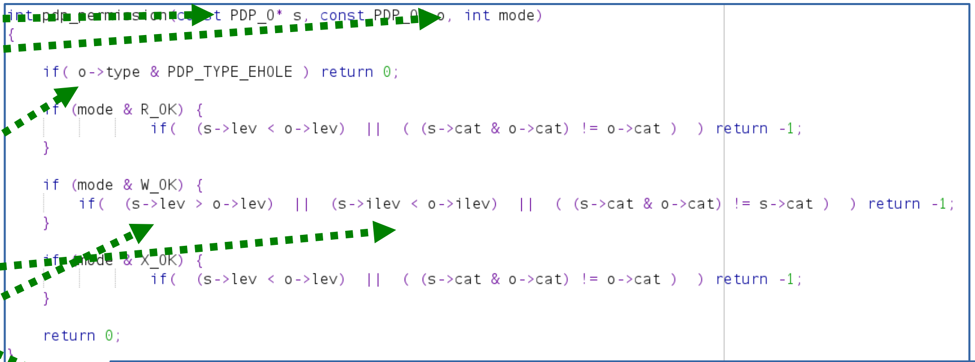
[если $y \in E$, то

$i_e(y) \le i_s(x)$

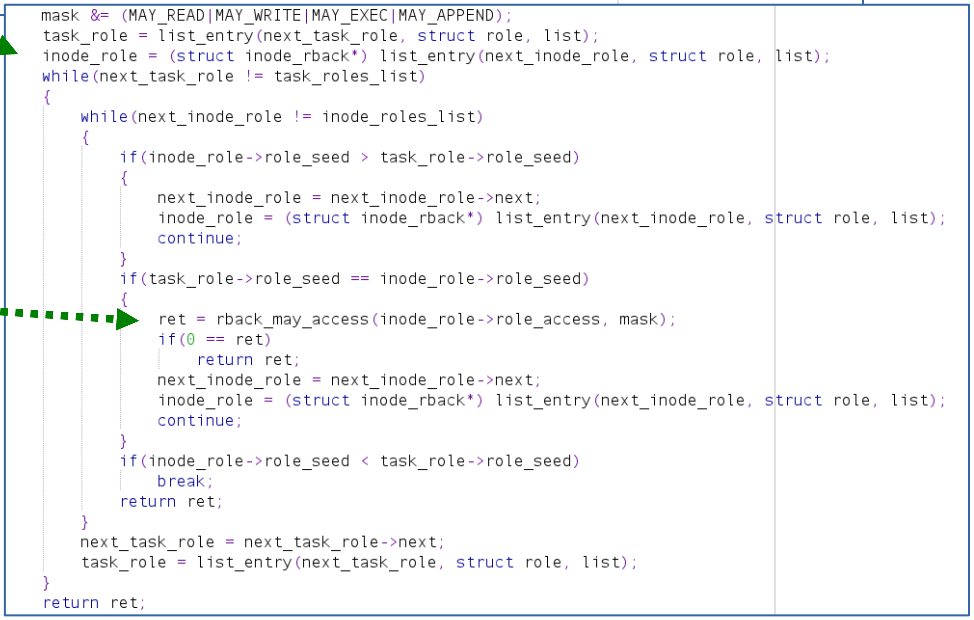
и (либо ($execute\_container(x, y) = true$

и, если $y \in E\_HOLE$, то $f_s(x) \le f_e(y)$,
иначе $f_e(y) = f_s(x)$),
либо $(x, downgrade\_admin\_role, read_a) \in AA$),

и $(y, write_r) \in PA(r)$],
[если $y \in R \cup AR$, то $(y, write_r) \in APA(r)$,
$i_r(y) \le i_s(x)$, $Constraint_{AA}(AA') = true$,
(для $e \in \,]y[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$), (либо $f_r(y) = f_s(x)$,
либо $(x, downgrade\_admin\_role, read_a) \in AA$)],
[если ($y \in E$ и $i_e(y) = i\_high$) или
($y \in R \cup AR$ и $i_r(y) = i\_high$),
то $(x', f_s(x)\_i\_entity, write_a) \in A$]



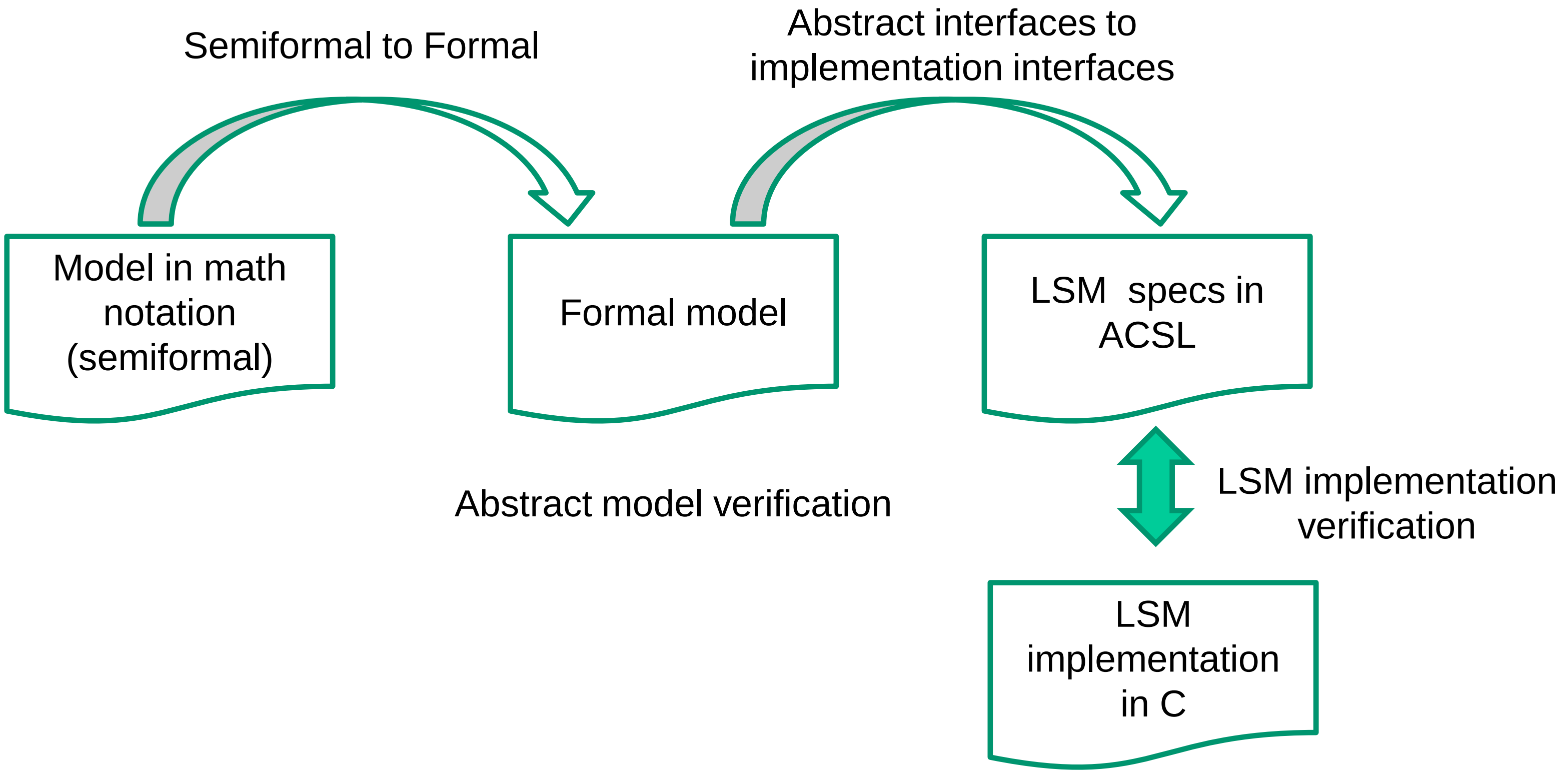
```
int pdp_permission(const PDP_0* s, const PDP_0* o, int mode)
{

    if( o->type & PDP_TYPE_EHOLE ) return 0;

    if (mode & R_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    if (mode & W_OK) {
        if(  (s->lev > o->lev)  ||  (s->ilev < o->ilev)  ||  ( (s->cat & o->cat) != s->cat )  ) return -1;
    }

    if (mode & X_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    return 0;
}
```



```
    mask &= (MAY_READ|MAY_WRITE|MAY_EXEC|MAY_APPEND);
    task_role = list_entry(next_task_role, struct role, list);
    inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
    while(next_task_role != task_roles_list)
    {
        while(next_inode_role != inode_roles_list)
        {
            if(inode_role->role_seed > task_role->role_seed)
            {
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(task_role->role_seed == inode_role->role_seed)
            {
                ret = rback_may_access(inode_role->role_access, mask);
                if(0 == ret)
                    return ret;
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(inode_role->role_seed < task_role->role_seed)
                break;
            return ret;
        }
        next_task_role = next_task_role->next;
        task_role = list_entry(next_task_role, struct role, list);
    }
    return ret;
```

LSM Verification Project

LSM stands for Linux Security Module



Semiformal to Formal

Abstract interfaces to implementation interfaces

Model in math notation (semiformal)

Formal model

LSM specs in ACSL

Abstract model verification

LSM implementation verification

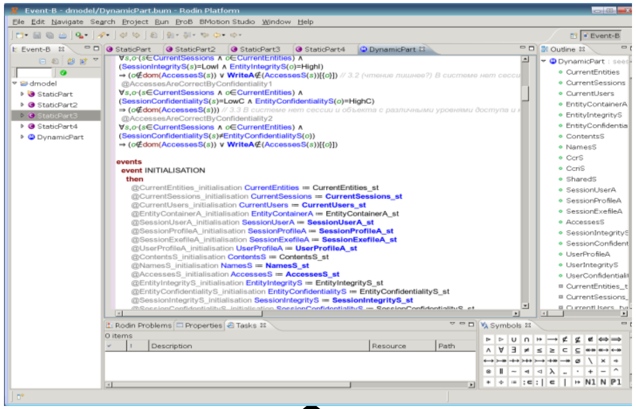
LSM implementation in C

Verification Tool Chain

MROSL-DP model in math notation

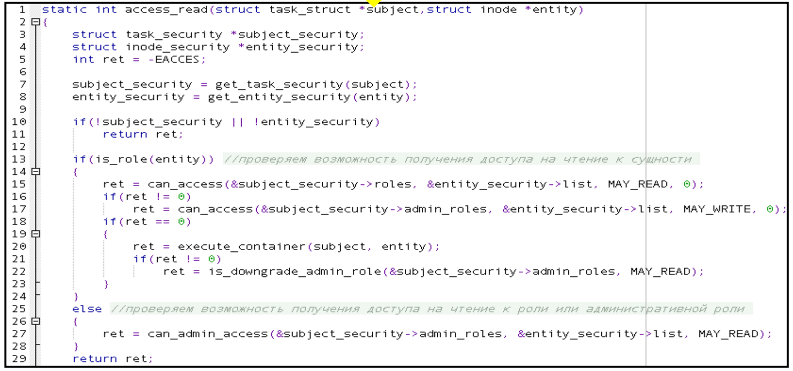
access_read(x, x′, y)	
$x, x' \in S, y \in E \cup R \cup AR$, существует $r \in R \cup AR$: $(x, r, read_a) \in AA$, [если $y \in E$, то $(y, read_r) \in PA(r)$ и либо ($execute\_container(x, y) = true$ и $f_e(y) \le f_s(x)$), либо $(x, downgrade\_admin\_role, read_a) \in AA$], [если $y \in R \cup AR$, то $(y, read_r) \in APA(r)$, $i_r(y) \le i_s(x)$, $Constraint_{AA}(AA') = true$, (для $e \in]y[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$), (либо $f_r(y) \le f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$)], [если $y \in R \cup AR$ и $i_r(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]	$S' = S$, $E' = E$, $APA' = APA$, $PA' = PA$, $user' = user$, $H_E' = H_E$, $F' = F$, если $y \in E$, то [$A' = A \cup \{(x, y, read_a)\}$, $AA' = AA$], если $y \in R \cup AR$, то [$AA' = AA \cup \{(x, y, read_a)\}$, $A' = A$]

Deductive verification of MROSL-DP model



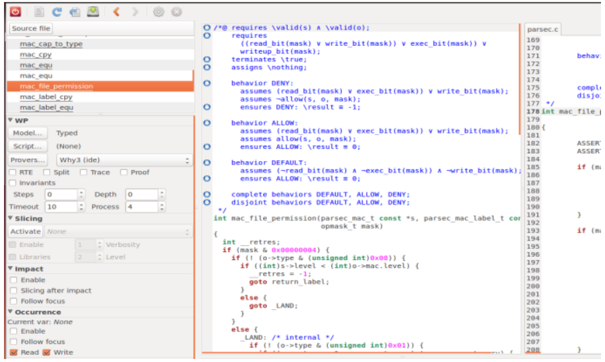
```
∀s,o·(s∈CurrentSessions ∧ o∈CurrentEntities) ∧
(SessionIntegrityS(s)=LowI ∧ EntityIntegrityS(o)=HighI)
⇒ (o∉dom(AccessesS(s)) ∨ WriteA∉(AccessesS(s))[{o}]) // 3.2 (чтение лишнее?) В системе нет сесси
 @AccessesAreCorrectByConfidentiality1
∀s,o·(s∈CurrentSessions ∧ o∈CurrentEntities) ∧
(SessionConfidentialityS(s)=LowC ∧ EntityConfidentialityS(o)=HighC)
⇒ (o∉dom(AccessesS(s))) // 3.3 В системе нет сессии и объекта с различными уровнями доступа и к
 @AccessesAreCorrectByConfidentiality2
∀s,o·(s∈CurrentSessions ∧ o∈CurrentEntities) ∧
(SessionConfidentialityS(s)≠EntityConfidentialityS(o))
⇒ (o∉dom(AccessesS(s)) ∨ WriteA∉(AccessesS(s))[{o}])

events
 event INITIALISATION
  then
   @CurrentEntities_initialisation CurrentEntities ≔ CurrentEntities_st
   @CurrentSessions_initialisation CurrentSessions ≔ CurrentSessions_st
   @CurrentUsers_initialisation CurrentUsers ≔ CurrentUsers_st
   @EntityContainerA_initialisation EntityContainerA ≔ EntityContainerA_st
   @SessionUserA_initialisation SessionUserA ≔ SessionUserA_st
   @SessionProfileA_initialisation SessionProfileA ≔ SessionProfileA_st
   @SessionExefileA_initialisation SessionExefileA ≔ SessionExefileA_st
   @UserProfileA_initialisation UserProfileA ≔ UserProfileA_st
   @ContentsS_initialisation ContentsS ≔ ContentsS_st
   @NamesS_initialisation NamesS ≔ NamesS_st
   @AccessesS_initialisation AccessesS ≔ AccessesS_st
   @EntityIntegrityS_initialisation EntityIntegrityS ≔ EntityIntegrityS_st
   @EntityConfidentialityS_initialisation EntityConfidentialityS ≔ EntityConfidentialityS_st
   @SessionIntegrityS_initialisation SessionIntegrityS ≔ SessionIntegrityS_st
```

Rodin (Event-B)



```c
static int access_read(struct task_struct *subject,struct inode *entity)
{
    struct task_security *subject_security;
    struct inode_security *entity_security;
    int ret = -EACCES;

    subject_security = get_task_security(subject);
    entity_security = get_entity_security(entity);

    if(!subject_security || !entity_security)
        return ret;

    if(is_role(entity)) //проверяем возможность получения доступа на чтение к сущности
    {
        ret = can_access(&subject_security->roles, &entity_security->list, MAY_READ, 0);
        if(ret != 0)
            ret = can_access(&subject_security->admin_roles, &entity_security->list, MAY_WRITE, 0);
        if(ret == 0)
        {
            ret = execute_container(subject, entity);
            if(ret != 0)
                ret = is_downgrade_admin_role(&subject_security->admin_roles, MAY_READ);
        }
    }
    else //проверяем возможность получения доступа на чтение к роли или административной роли
    {
        ret = can_admin_access(&subject_security->admin_roles, &entity_security->list, MAY_READ);
    }
    return ret;
```

Part of LSM in Astra Linux



▾ WP
Model... Typed
Script... (None)
Provers... Why3 (ide)
☐ RTE ☐ Split ☐ Trace ☐ Proof
☐ Invariants
Steps 0 Depth 0
Timeout 10 Process 4
▾ Slicing
Activate None
☐ Enable 1 Verbosity
☐ Libraries 2 Level
▾ Impact
☐ Enable
☐ Slicing after impact
☐ Follow focus
▾ Occurrence
Current var: *None*
☐ Enable
☐ Follow focus
☑ Read ☑ Write

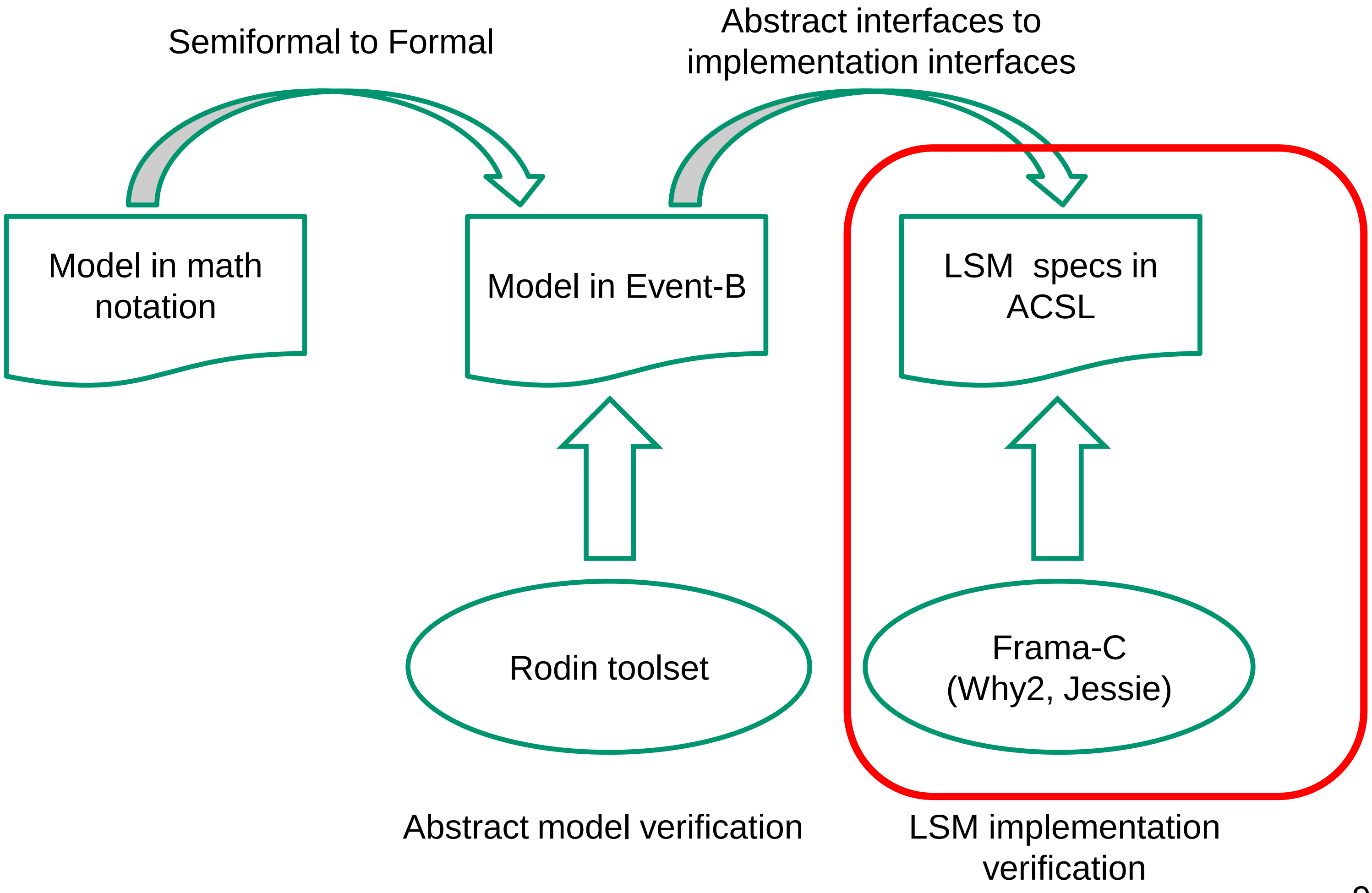
```c
/*@ requires \valid(s) ∧ \valid(o);
    requires
      ((read_bit(mask) ∨ write_bit(mask)) ∨ exec_bit(mask)) ∨
      writeup_bit(mask);
    terminates \true;
    assigns \nothing;

    behavior DENY:
      assumes (read_bit(mask) ∨ exec_bit(mask)) ∨ write_bit(mask);
      assumes ¬allow(s, o, mask);
      ensures DENY: \result ≡ -1;

    behavior ALLOW:
      assumes (read_bit(mask) ∨ exec_bit(mask)) ∨ write_bit(mask);
      assumes allow(s, o, mask);
      ensures ALLOW: \result ≡ 0;

    behavior DEFAULT:
      assumes (¬read_bit(mask) ∧ ¬exec_bit(mask)) ∧ ¬write_bit(mask);
      ensures ALLOW: \result ≡ 0;

    complete behaviors DEFAULT, ALLOW, DENY;
    disjoint behaviors DEFAULT, ALLOW, DENY;
 */
int mac_file_permission(parsec_mac_t const *s, parsec_mac_label_t cor
                        opmask_t mask)
{
  int __retres;
  if (mask & 0x00000004) {
    if (! (o->type & (unsigned int)0x08)) {
      if ((int)s->level < (int)o->mac.level) {
        __retres = -1;
        goto return_label;
      }
      else {
        goto _LAND;
      }
    }
    else {
      _LAND: /* internal */
        if (! (o->type & (unsigned int)0x01)) {
```

Frama-C, Why3

Deductive verification LSM in Astra Linux

LSM Verification Project

LSM stands for Linux Security Module

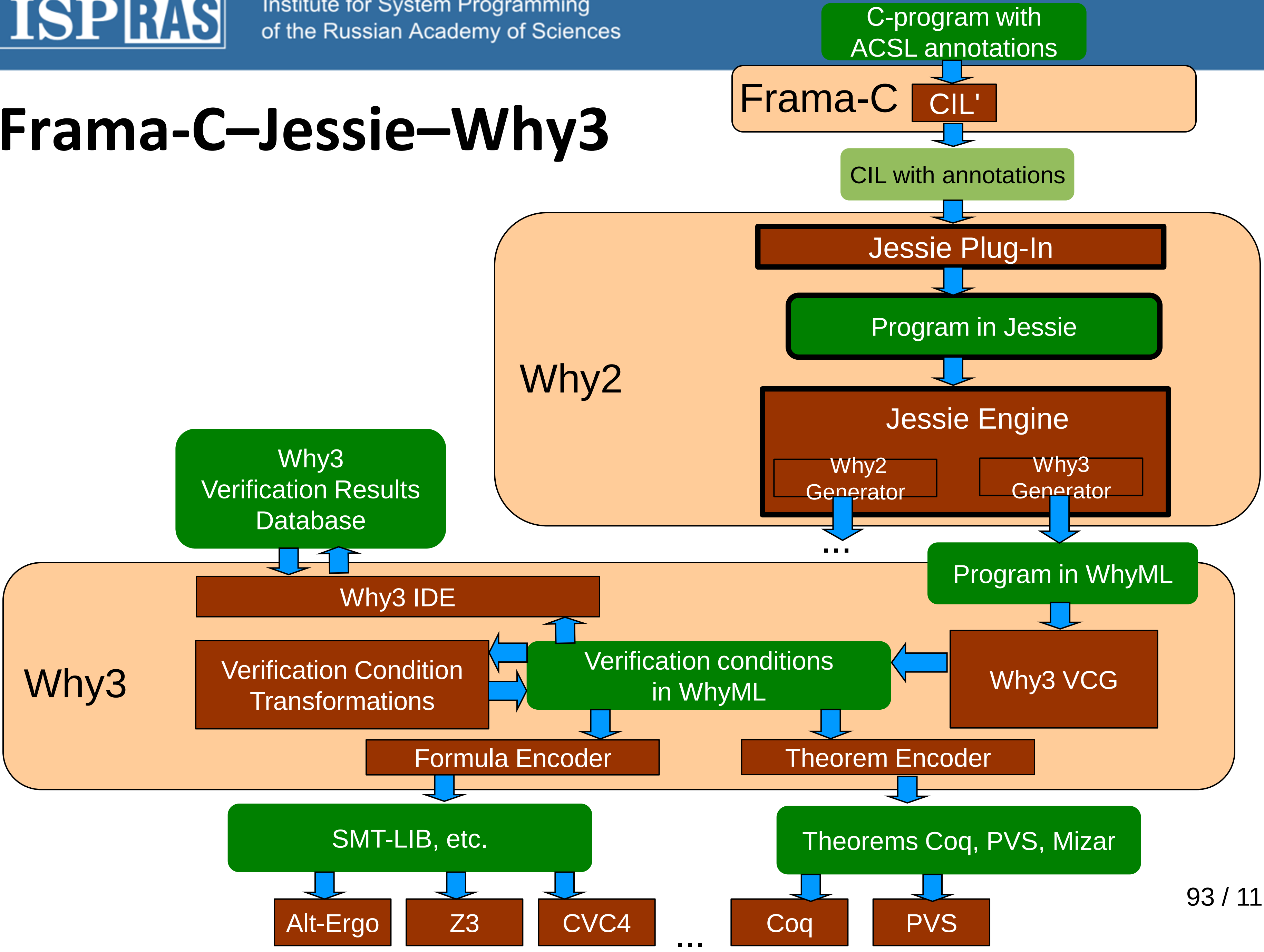


Deductive Verification in C (*)

	Open source	Memory model	Already applied for OS low-level code verification	Usability
VCC	−	+	+	−
Why3	+	+	−	+
Frama-C WP	+	∓	−	+
VeriFast	−	+	∓	−
C-to-Isabelle	+	+	+	±

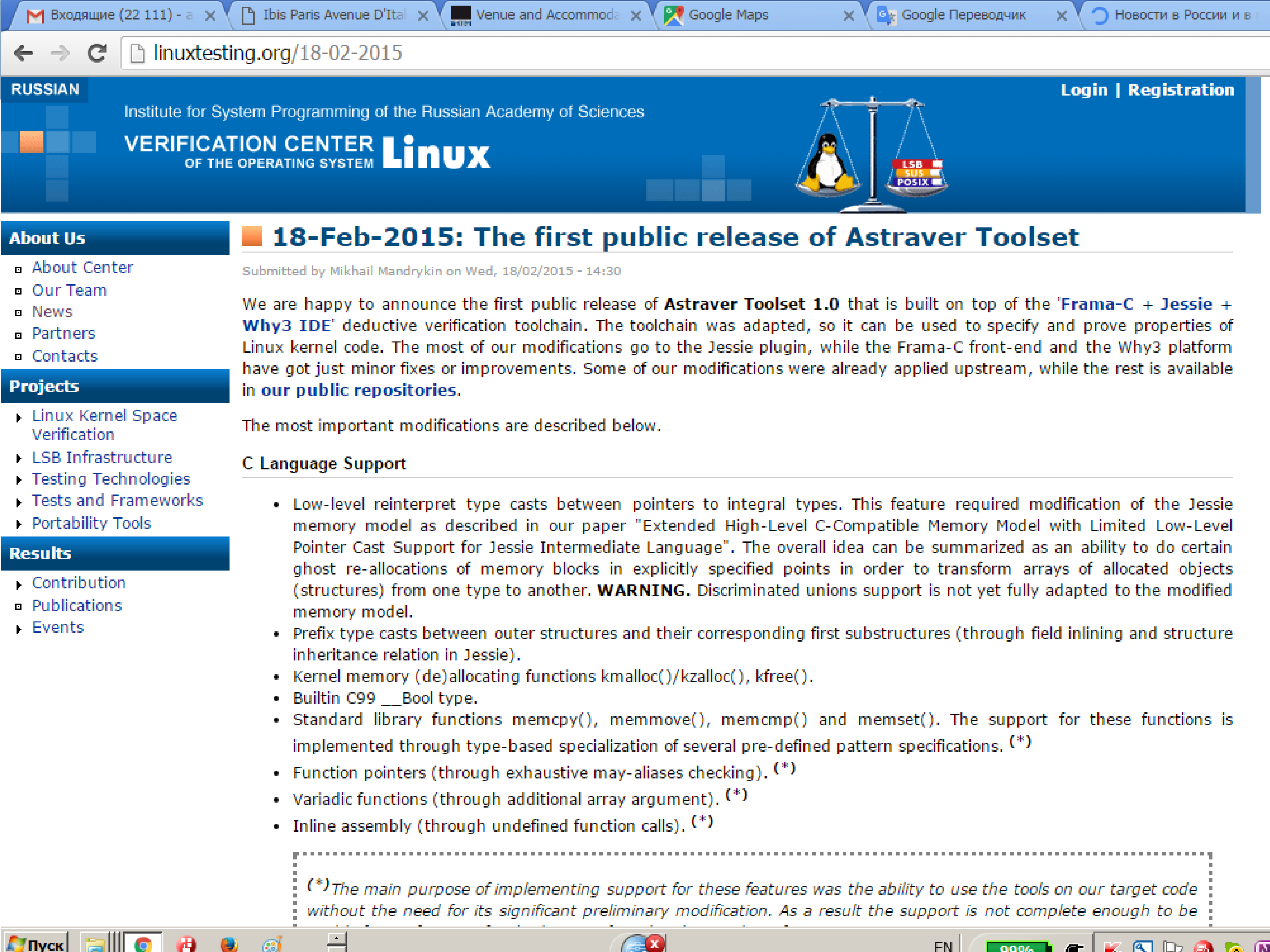
(*) The research on deductive verification tools development was carried out with funding from the Ministry of Education and Science of Russia(the project unique identifier is RFMEFI60414X0051)

Frama-C–Jessie–Why3



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



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

Submitted by Mikhail Mandrykin on Wed, 18/02/2015 - 14:30

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.

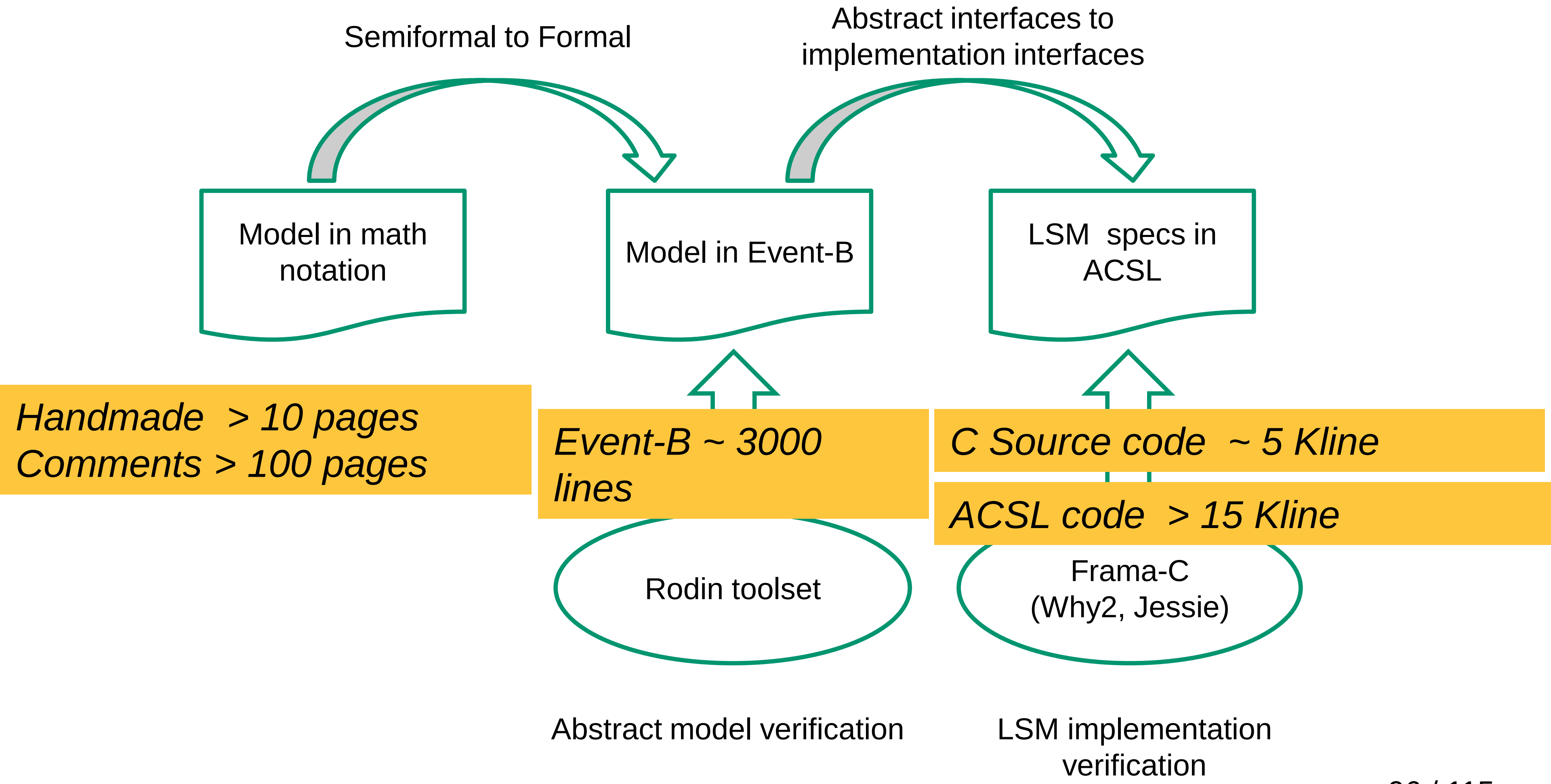
C Language Support

- 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

LSM Verification Project

LSM stands for Linux Security Module



Semiformal to Formal

Abstract interfaces to implementation interfaces

Model in math notation

Model in Event-B

LSM specs in ACSL

Handmade > 10 pages
Comments > 100 pages

Event-B ~ 3000 lines

C Source code ~ 5 Kline

ACSL code > 15 Kline

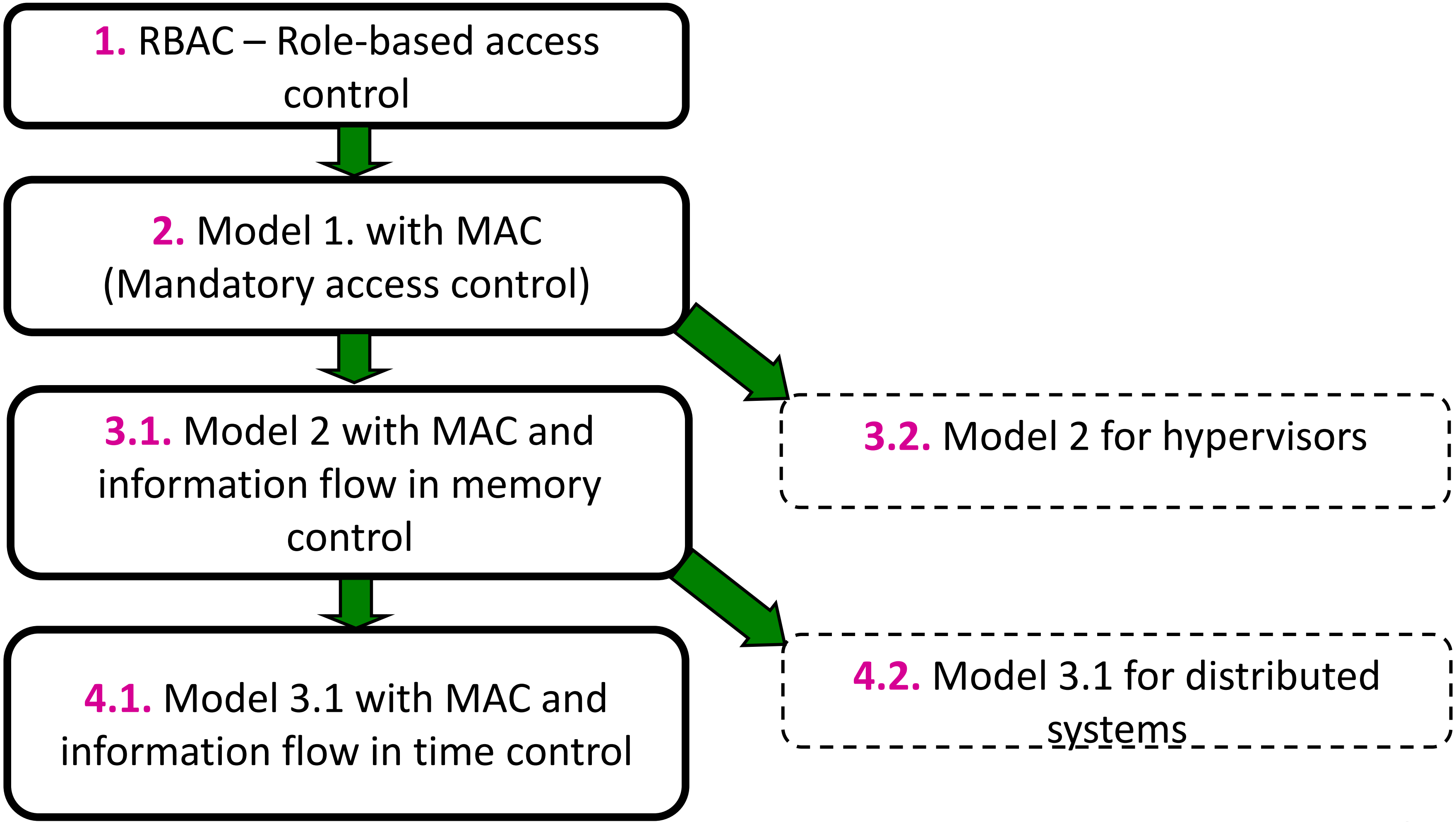
Rodin toolset

Frama-C (Why2, Jessie)

Abstract model verification

LSM implementation verification

Hierarchical MROSL DP Model (decomposition of Event-B model)



1. RBAC – Role-based access control

2. Model 1. with MAC (Mandatory access control)

3.1. Model 2 with MAC and information flow in memory control

3.2. Model 2 for hypervisors

4.1. Model 3.1 with MAC and information flow in time control

4.2. Model 3.1 for distributed systems

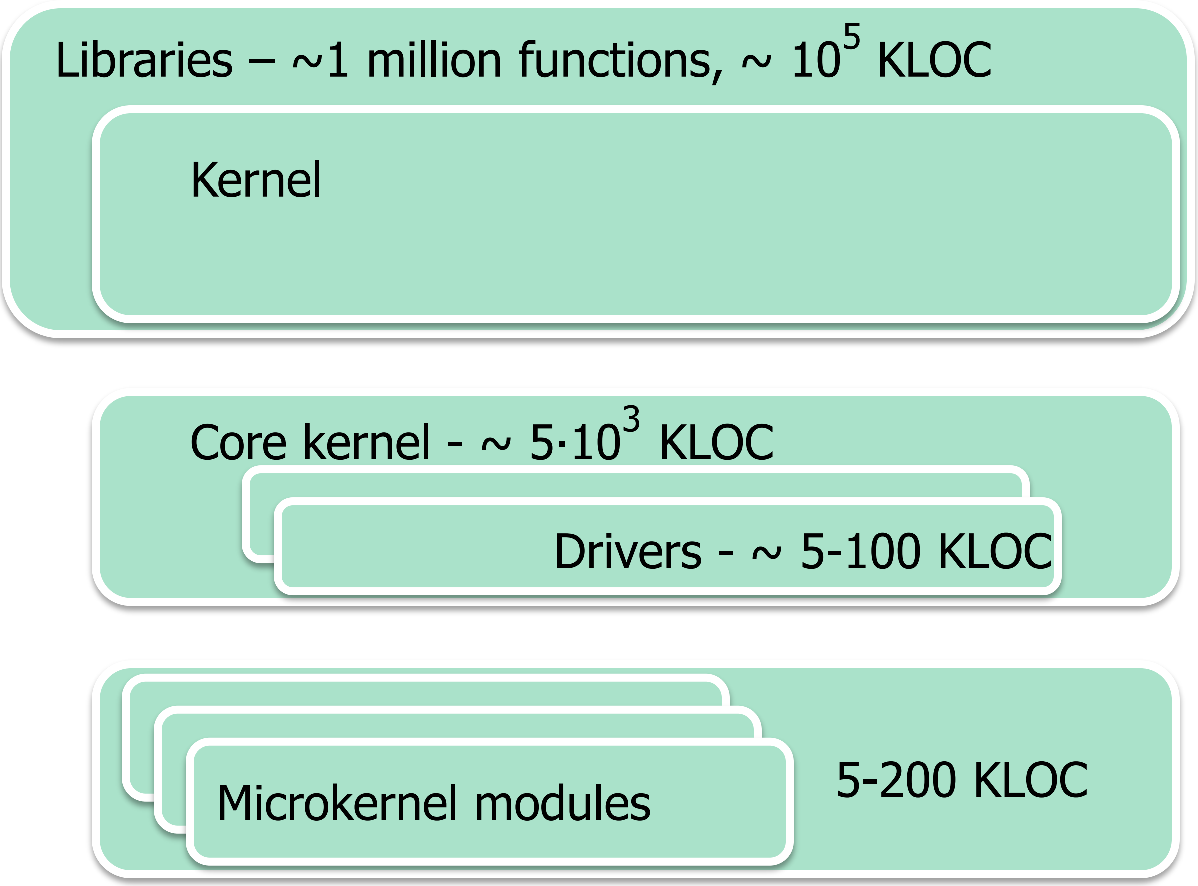
LSM Verification Conclusion

- InfoSec requirements are essentially non-functional, they are not decomposed as the functional requirements and
- the direct correspondence between the formal security model entities implementation entities of such a complex system as the operating system (?) can not be built
- What to do?

Final Discussion

OS Scale

- Libraries + Kernel
- Monolithic Kernel
- Microkernel



Libraries – ~1 million functions, ~ 10^5 KLOC

Kernel

Core kernel - ~ $5{\cdot}10^3$ KLOC

Drivers - ~ 5-100 KLOC

Microkernel modules

5-200 KLOC

OS Scale - Verification Approaches

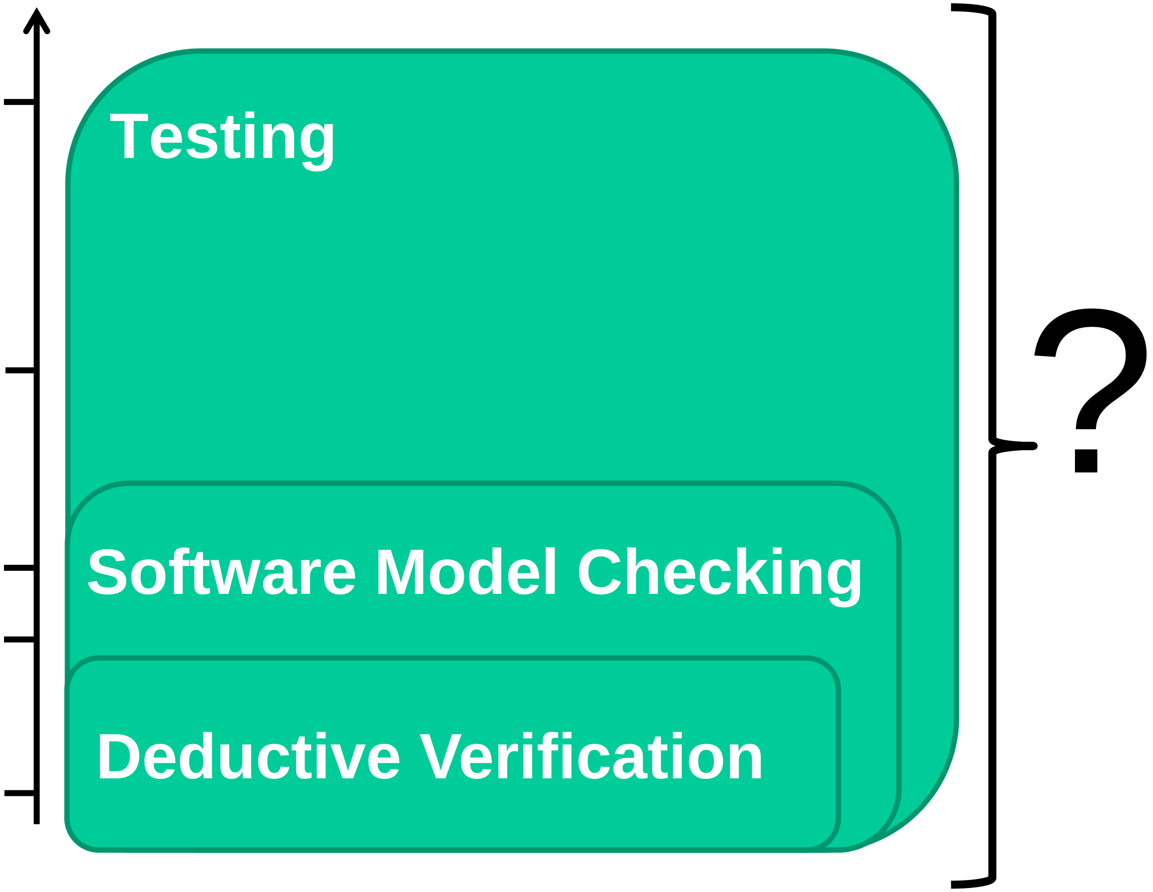
Libraries + Kernel ~ 10^6 KLOC

Monolithic Kernel ~ 10^4 KLOC
(Linux, Windows)

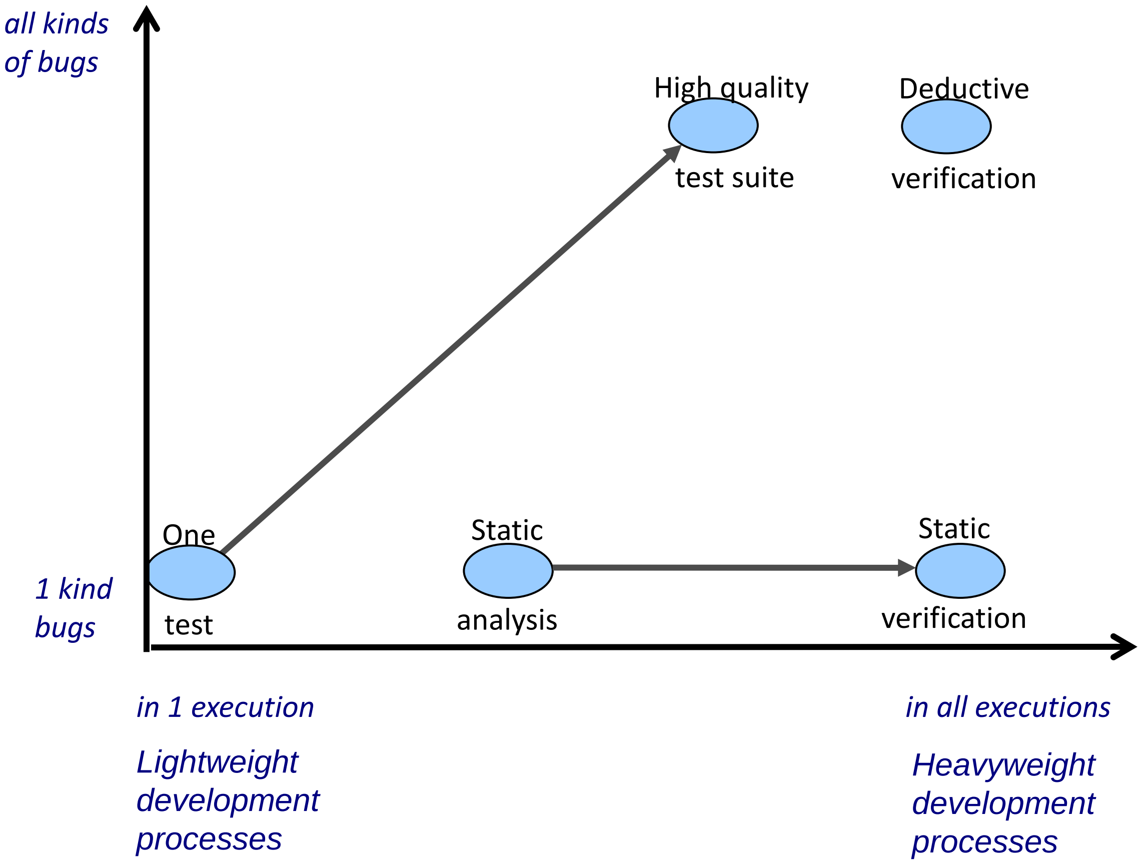
Hypervisors (Hyper-V)< 300 KLOC

Drivers/modules < 100 KLOC

Microkernel < 10 KLOC
(L4, PikeOS)

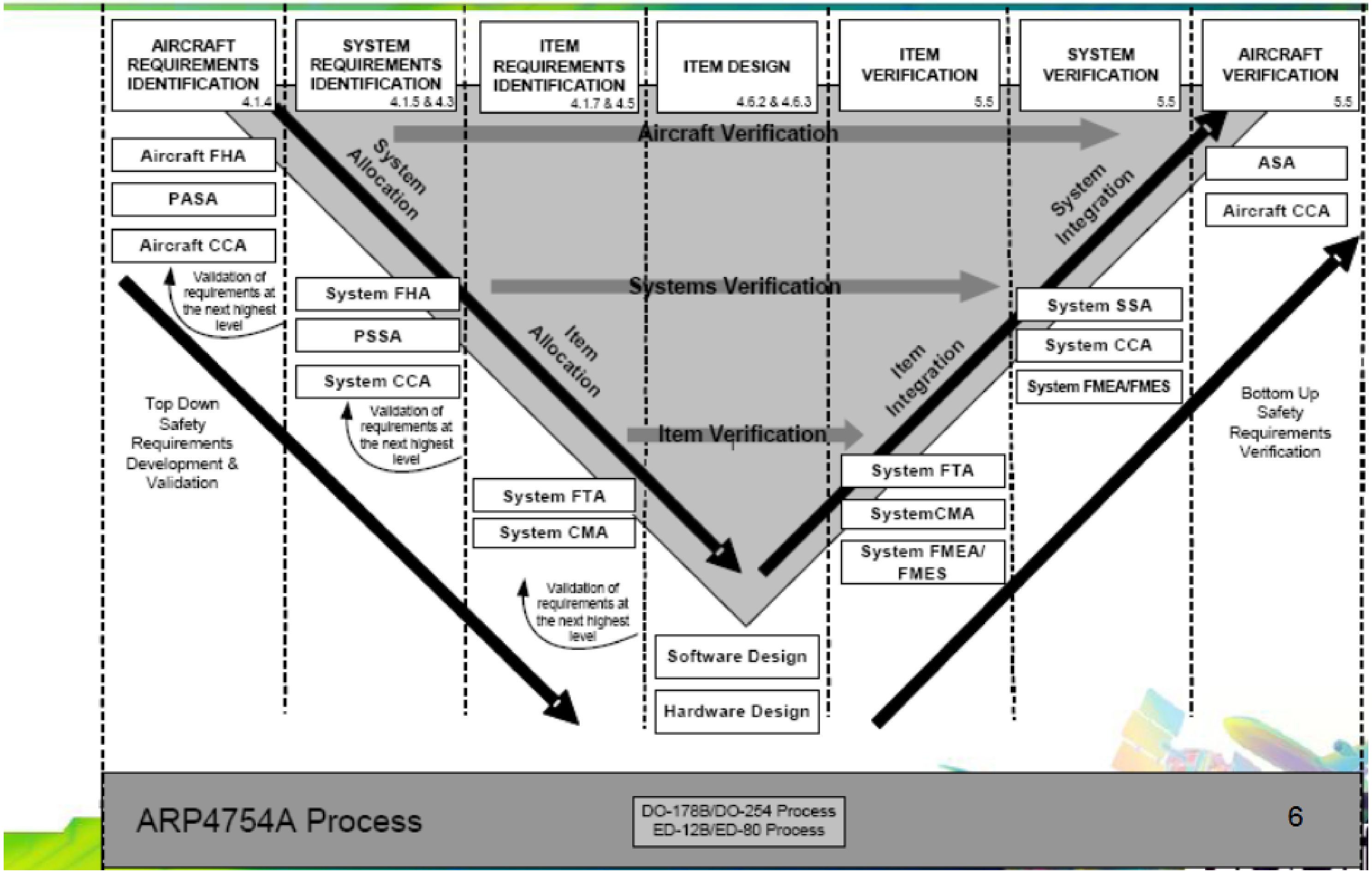


Verification Approaches and Development Processes



What is “Heavyweight processes”?

ARP 4754A: Interactions of Requirements, Safety, and Development



AIRCRAFT REQUIREMENTS IDENTIFICATION 4.1.4

SYSTEM REQUIREMENTS IDENTIFICATION 4.1.5 & 4.3

ITEM REQUIREMENTS IDENTIFICATION 4.1.7 & 4.5

ITEM DESIGN 4.6.2 & 4.6.3

ITEM VERIFICATION 5.5

SYSTEM VERIFICATION 5.5

AIRCRAFT VERIFICATION 5.5

Aircraft Verification

Systems Verification

Item Verification

System Allocation

Item Allocation

Item Integration

System Integration

Aircraft FHA

PASA

Aircraft CCA

Validation of requirements at the next highest level

Top Down Safety Requirements Development & Validation

System FHA

PSSA

System CCA

Validation of requirements at the next highest level

System FTA

System CMA

Validation of requirements at the next highest level

Software Design

Hardware Design

System FTA

SystemCMA

System FMEA/ FMES

System SSA

System CCA

System FMEA/FMES

ASA

Aircraft CCA

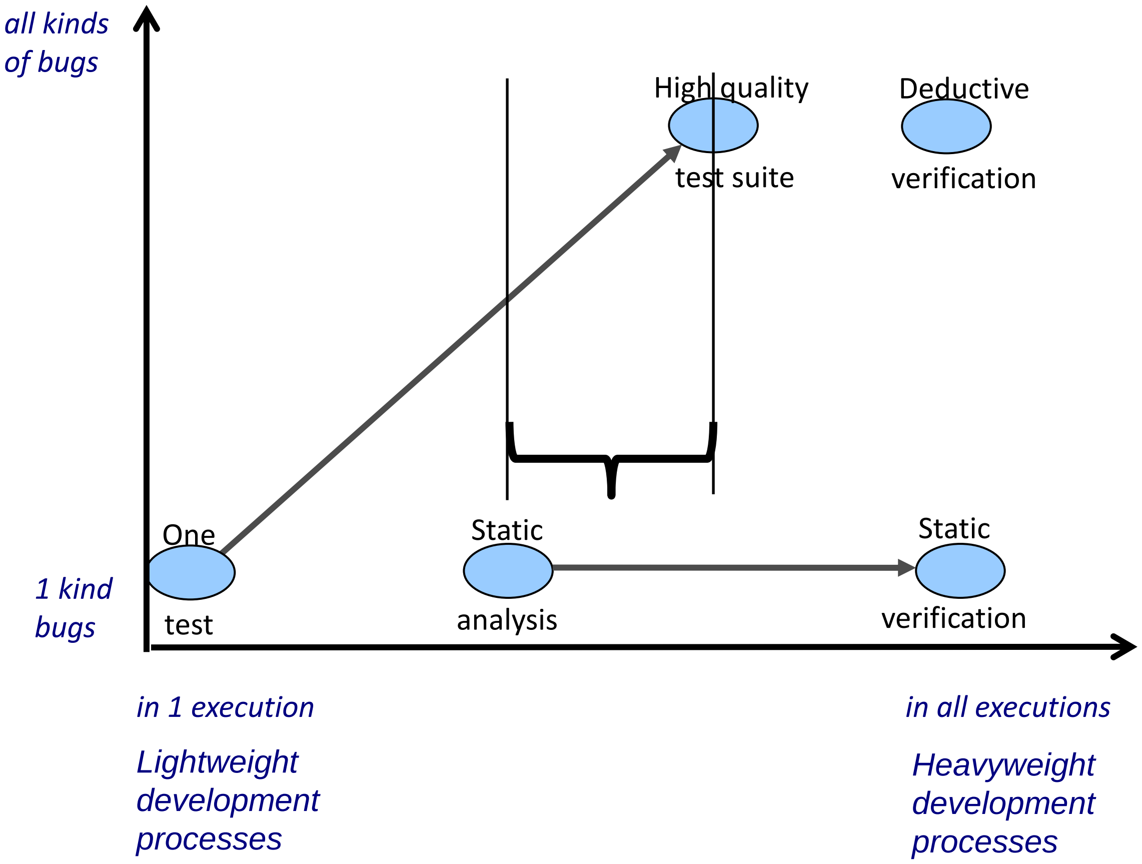
Bottom Up Safety Requirements Verification

ARP4754A Process

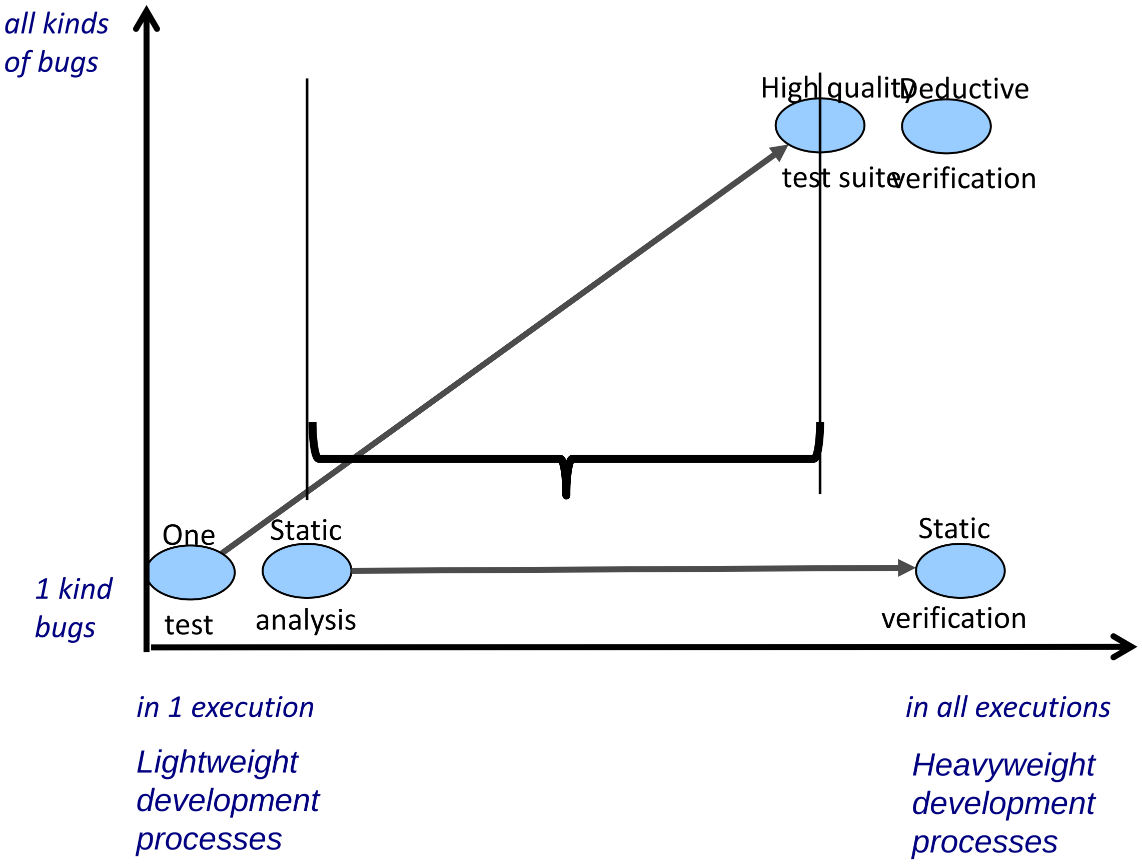
DO-178B/DO-254 Process
ED-12B/ED-80 Process

6

Verification Approaches and Development Processes



Verification Approaches and Development Processes



Conclusion on Practical Verification

Trivial conclusions:

- No silver bullet
- We are seeing remarkable progress in the use of formal and other sophisticated software analysis techniques.

Other ones:

- However deep testing and verification require a deep knowledge of the system under analysis and it is not clear how such a situation may change in the near future
- The axiom that testing should be done by an independent testers group in the case of very complex systems is not valid.

Frederick P. Brooks Jr.

Conclusion on Practical Verification

- Dines Bjørner : Each development team must include at least one mathematicion
- In practice, Intel and Microsoft have integrated development team and testers
- seL4 & PikeOS verification experience shows that such projects joint designers and mathematicians-verifiers.

Dines Bjørner

Conclusion OS Information Security

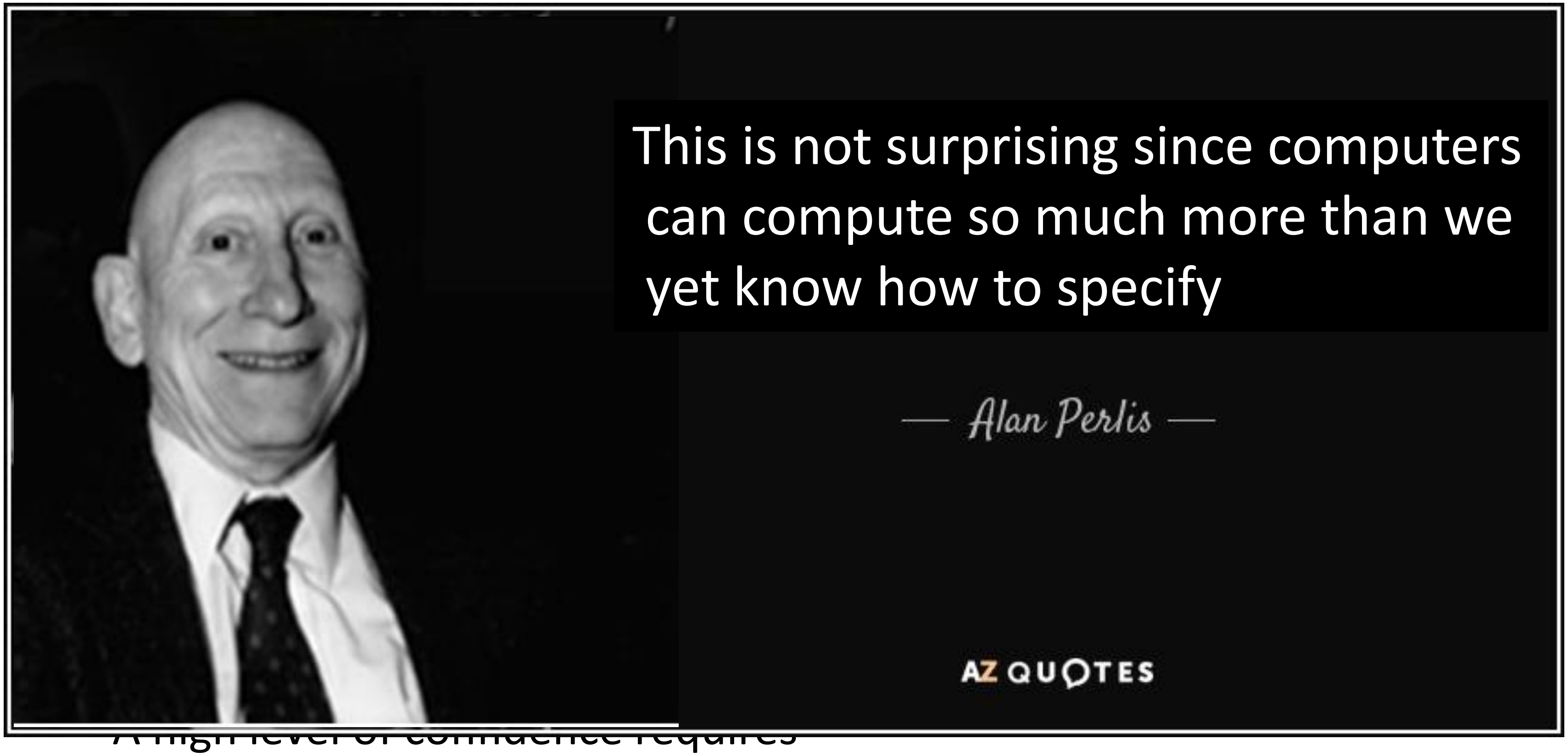
Trivial conclusion:

- Safety & security strongly intersect, one without the other can not be provided
- Deep verification easier to perform for a small and simple OS than for large and complex one.

Other ones:

- Programmers try to ensure safety without linking the design decisions with security issues - to some extent it is possible.
- But sometimes we can not follow this way, for example, we can not pass certification process.
- A high level of confidence requires heavyweight processes, in particular, careful work with the requirements specification - this is the most difficult moment - pointed out by Alan Perlis

Conclusion OS Information Security



This is not surprising since computers can compute so much more than we yet know how to specify

— Alan Perlis —

A high level of confidence requires heavyweight processes, in particular, careful work with the requirements specification - this is the most difficult moment - pointed out by Alan Perlis

Conclusion OS Information Security

- We have to establish the problem of conformance of security model with protection mechanisms of a trusted operating system informally (or formally in part).
- Shura-Bura noted that the transition from the informal to the formal is essentially informal.
- This thesis leads to the conclusion that in addition to the verification tasks we have establish and solve the validation task.
- Open problem: How to combine and reuse the techniques, tools, and verification&validation artifacts?

M.P.Shura-Bura

Acknowledgements:

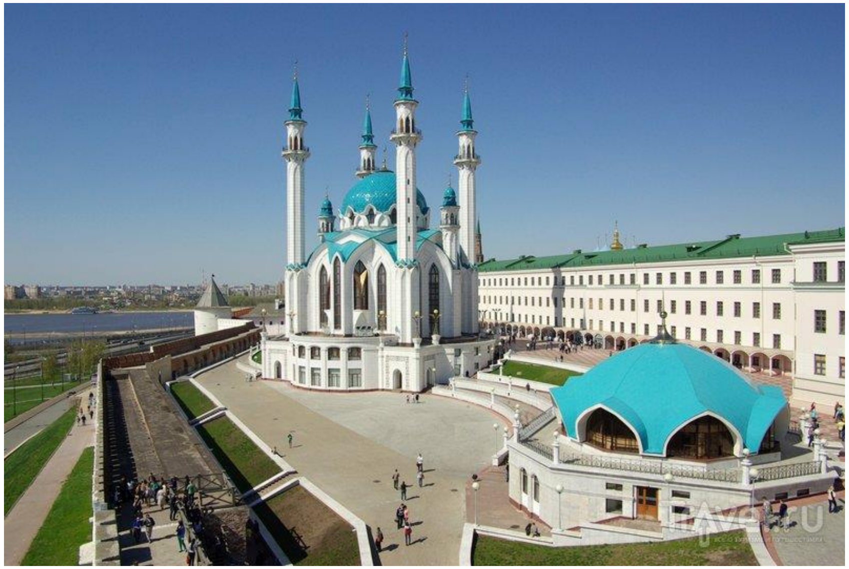
- TAROT organizers
- Antoine Rollet
- Alexey Khoroshilov, Victor Kuliamin, Petr Devyanin
- Sponsors and industrial partners

Merci!

Welcome to SYRCoSE-2017 in Kazan(Innopolis)

May 29-31, 2017

http://syrcose.ispras.ru



Read More . . .

  - http://www.ispras.ru/groups/se/
- Publications
  - http://www.ispras.ru/groups/se/publications.php
- Open projects: UniTESK, OLVER, LDV, BLAST, CPAchecker, MASIW, Requality, Frama-C/Why3/Jessie
  - http://unitesk.ru
  - http://forge.ispras.ru
  - http://hardware.ispras.ru
  - http://linuxtesting.org
  - http://www.linuxbase.org/navigator/commons/welcome.php
  - http://www.ispras.ru/technologies/
  - http://sdat.ispras.ru/
  - http://syrcose.ispras.ru/
  - http://www.isprasopen.ru/en/conf.html

Merci!