Verification of Operating System Components

A. KHOROSHILOV, V. KULIAMIN, A. PETRENKO
INSTITUTE FOR SYSTEM PROGRAMMING
RUSSIAN ACADEMY OF SCIENCES

Main Goal

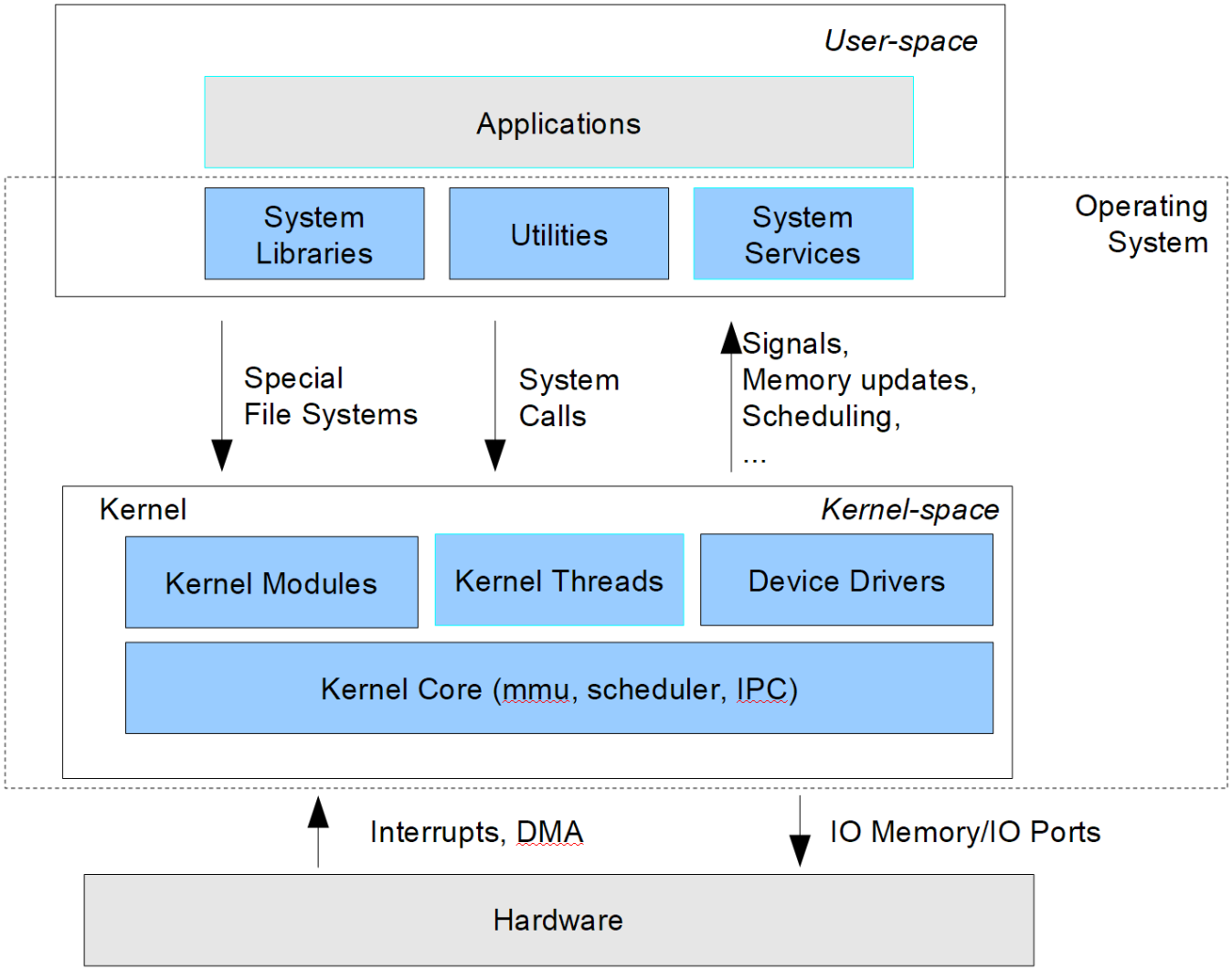
Verification of industrial operating system (OS)

- Industrial OS
  - actively used in some domain
  - developed and supported for a long period

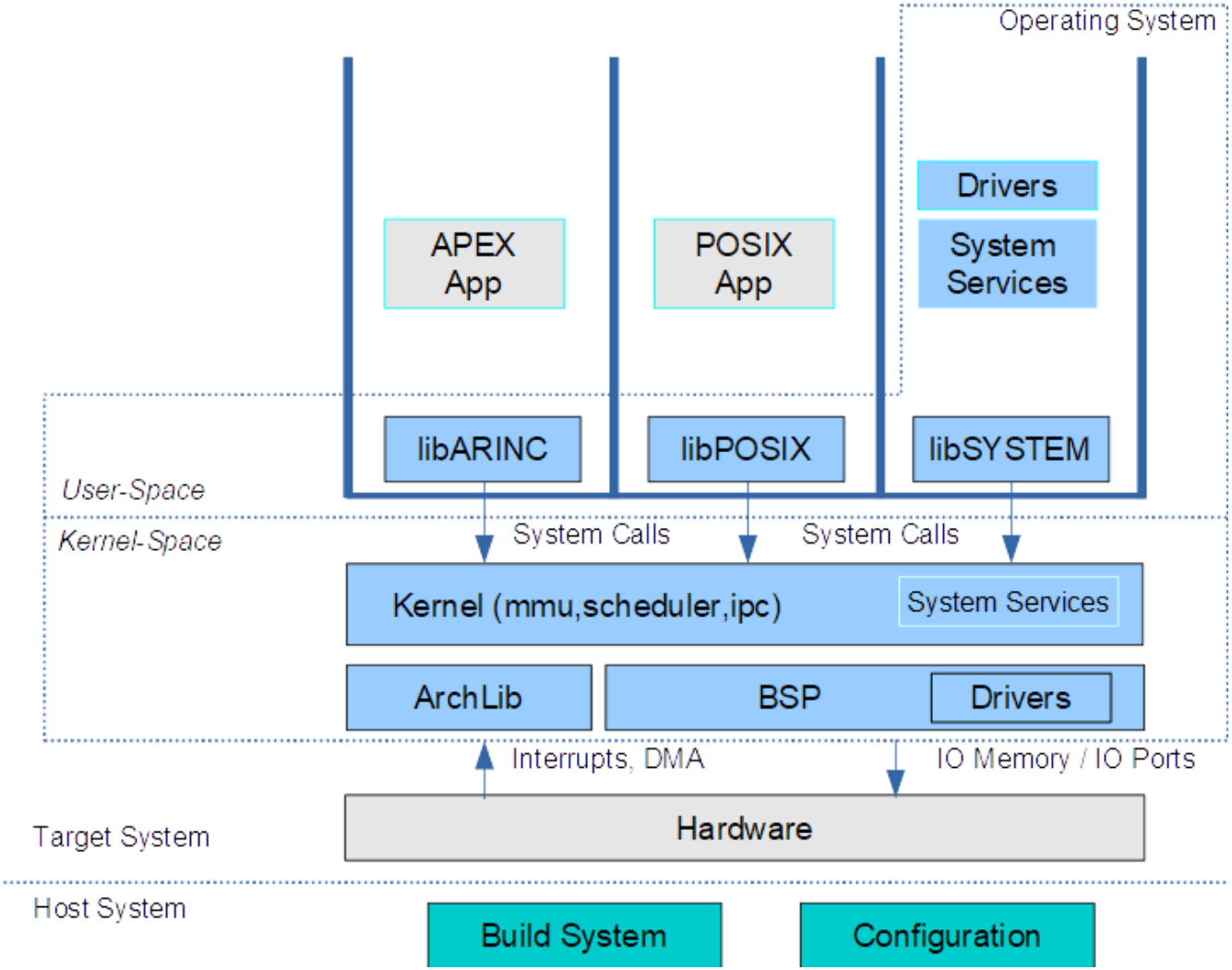
  - General-purpose (Linux, Windows, Android, MacOS, etc.)
  - Domain-specific (automotive, airplanes, energy generation, etc.)

- Verification
  - Strict guarantees that properties stated in requirements are implemented

Structure of General-purpose OS



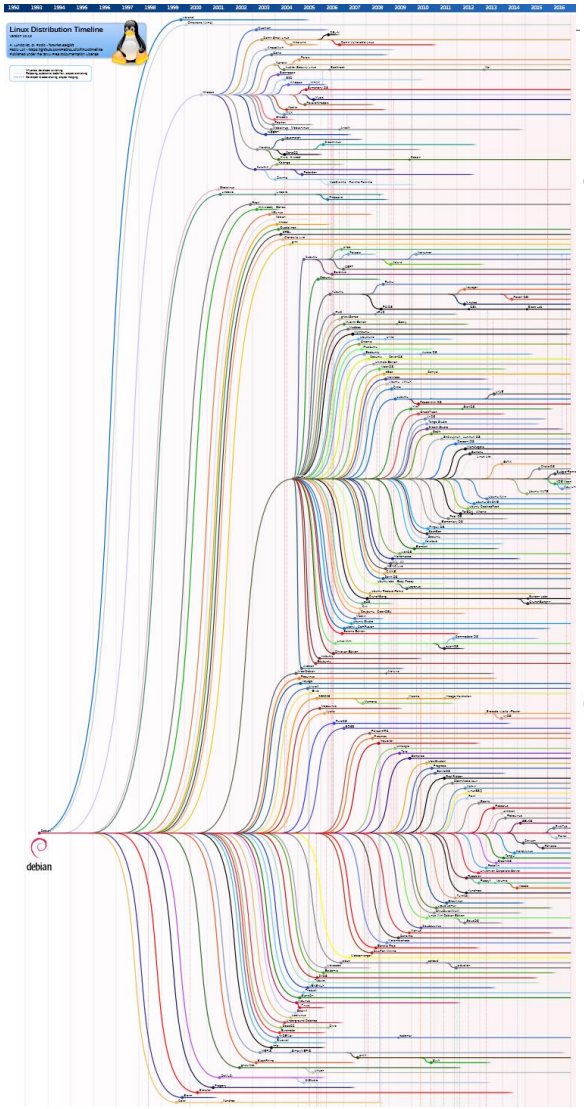
Structure of Real-time OS

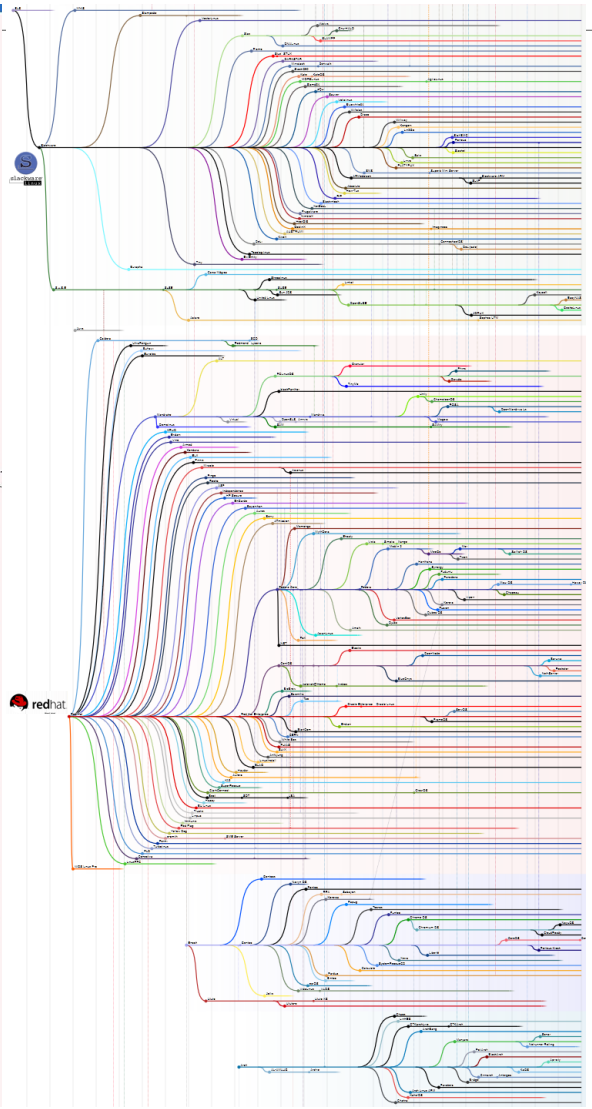


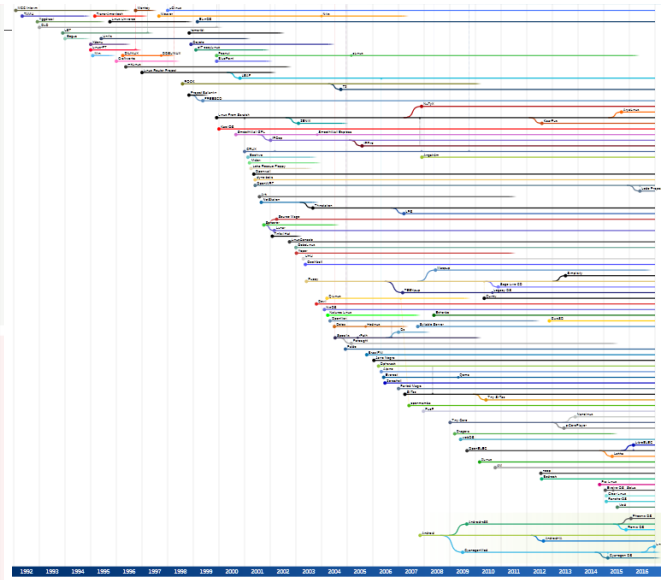
OS Code Size

Million lines of code								
1993	Windows NT 3.1	**~4.5**						
1994	Windows NT 3.5	**~7.5**						
1996	Windows NT 4.0	**~11.5**						
2000	Windows 2000	**>29**	Debian 2.2	**~57**				
2001	Windows XP	**45**			Linux kernel 2.4.2	**2.4**		
2002			Debian 3.0	**104**				
2003	Windows Server 2003	**50**			Linux kernel 2.6.0	**5.2**		
2005			Debian 3.1	**215**			Mac OS X 10.4	**86**
2009			Debian 5.0	**324**	Linux kernel 2.6.32	**12.6**	OpenSolaris (kernel)	**9.7**
2010					Linux kernel 2.6.35	**13.5**		
2012			Debian 7.0	**419**	Linux kernel 3.6	**15.9**		
2015					Linux kernel 4.2	**20.2**		

Linux Distributions







https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg

Verification of Industrial OS

Unreachable for now in any complete and accurate form

- Too large code
- Too many functions (eg., ~$7.2 \cdot 10^5$ Debian 7.0)
- Too complex behavior (too many states, multitasking, asynchrony, etc.)
- Too many variants
- Too many requirements (and almost all informal)
- Too diverse requirements (functionality, efficiency, fault tolerance, security, etc.)

What to do?

- Verify parts/modules/components separately
- Verify partial properties
- Combine different methods/techniques for more effectivity
- Iteratively enlarge verified parts of implementation and requirements

ISP RAS Projects 1

Dynamic analysis

- Testing
  - Based on formal specifications
    - 1994-1999 system libraries of switch OS by Nortel Networks KVEST
    - 2001-2005 protocols IPv6, Mobile IPv6, IPsec (for Windows) UniTESK
    - 2005-2008 system libraries of Linux (LSB Core, POSIX) UniTESK
    - 2006-2009 system libraries of Russian RTOS (POSIX, ARINC) UniTESK
  - W/o formal model of requirements
    - 2006-2009 system libraries of Linux (LSB) T2C, sanity testing
    - 2008-… system libraries of Russian RTOS (POSIX, ARINC) T2C
  - Testing + fault injection
    - 2014-2015 i/o system libraries KEDR
- Monitoring
  - 2009-… kernel modules, system libraries KEDR

Static analysis

Deductive verification

ISP RAS Projects 2

Dynamic analysis

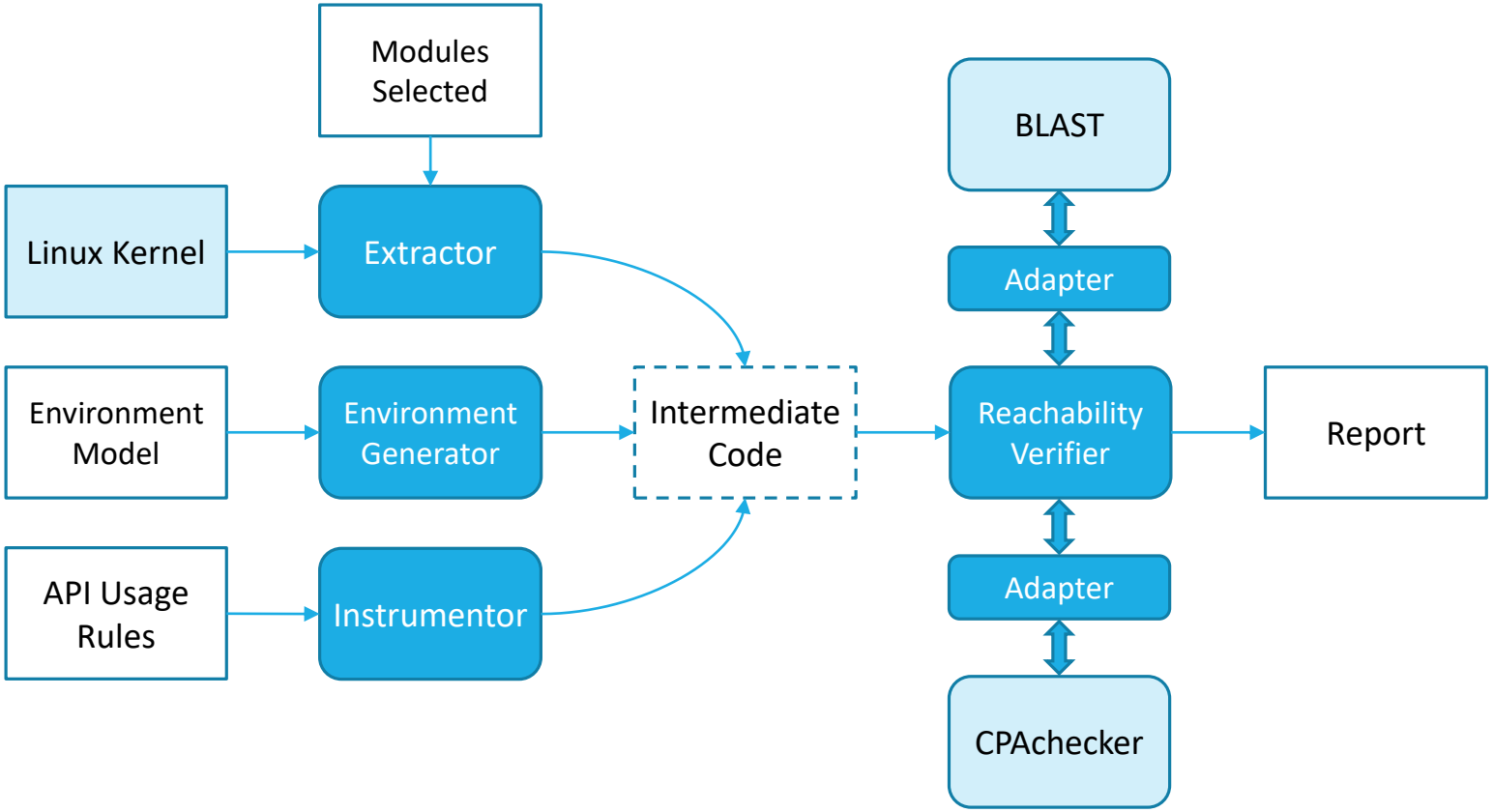
Static analysis

- Generic
  - 2010-… kernel modules against kernel core API usage rules Linux Driver Verification (LDV)
- Specific
  - 2014-… kernel code for race conditions

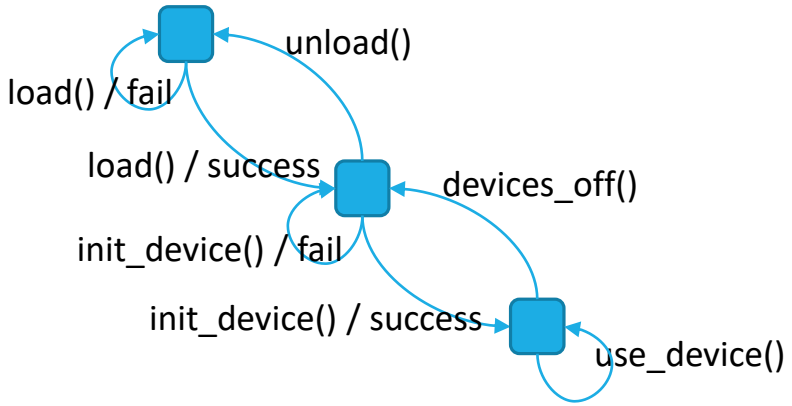
Deductive verification

- Security verification
  - 2014-… LSM code against MROSL-DP security model

Linux Driver Verification Tools



Environment Models

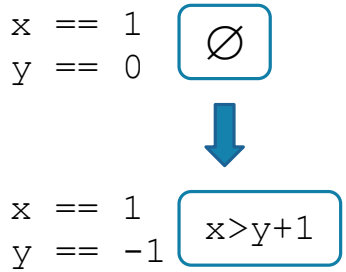


API Usage Rules – Aspect Contracts

```
pointcut USB_ALLOC_URB:
call( struct urb *usb_alloc_urb(int, gfp_t) )

pointcut USB_FREE_URB:
call( void usb_free_urb(struct urb *) )


around: USB_ALLOC_URB
{ return ldv_usb_alloc_urb(); }

around: USB_FREE_URB
{ ldv_usb_free_urb($arg1); }

before: USB_FREE_URB
{ ldv_assert(contains(URBS, $arg1)); }

after: MODULE_EXIT
{ ldv_assert(is_empty(URBS)); }
```

CEGAR Reachability Verification

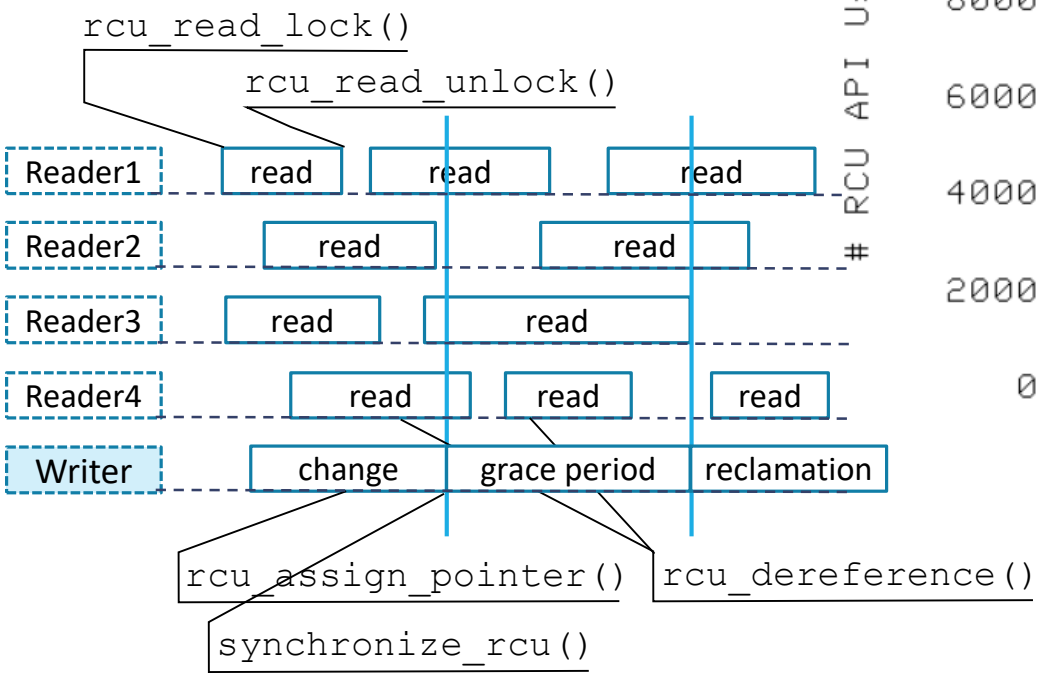
```
y++;
if(x > y)
{
  y =+ 2;
  if(x != y)
  {
    assert(x != 1);
  }
}

return y;
```

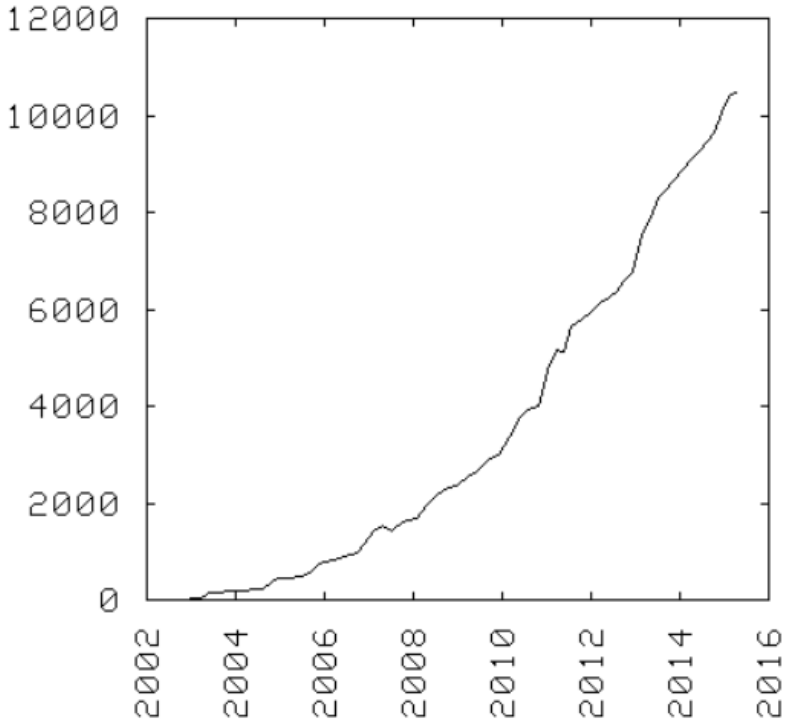


RCU Synchronization Mechanism

Read-Copy-Update

- 1993 DYNIX/ptx
- 2002 Linux kernel





Security Mechanisms

User and application level

- Security policies, security domains, users, user groups, roles, access rights

OS level (also DBMS, etc.)

- Security mechanisms
  - Discretionary access control (DAC)
  - Role-based access control (RBAC)
  - Mandatory access control (MAC)

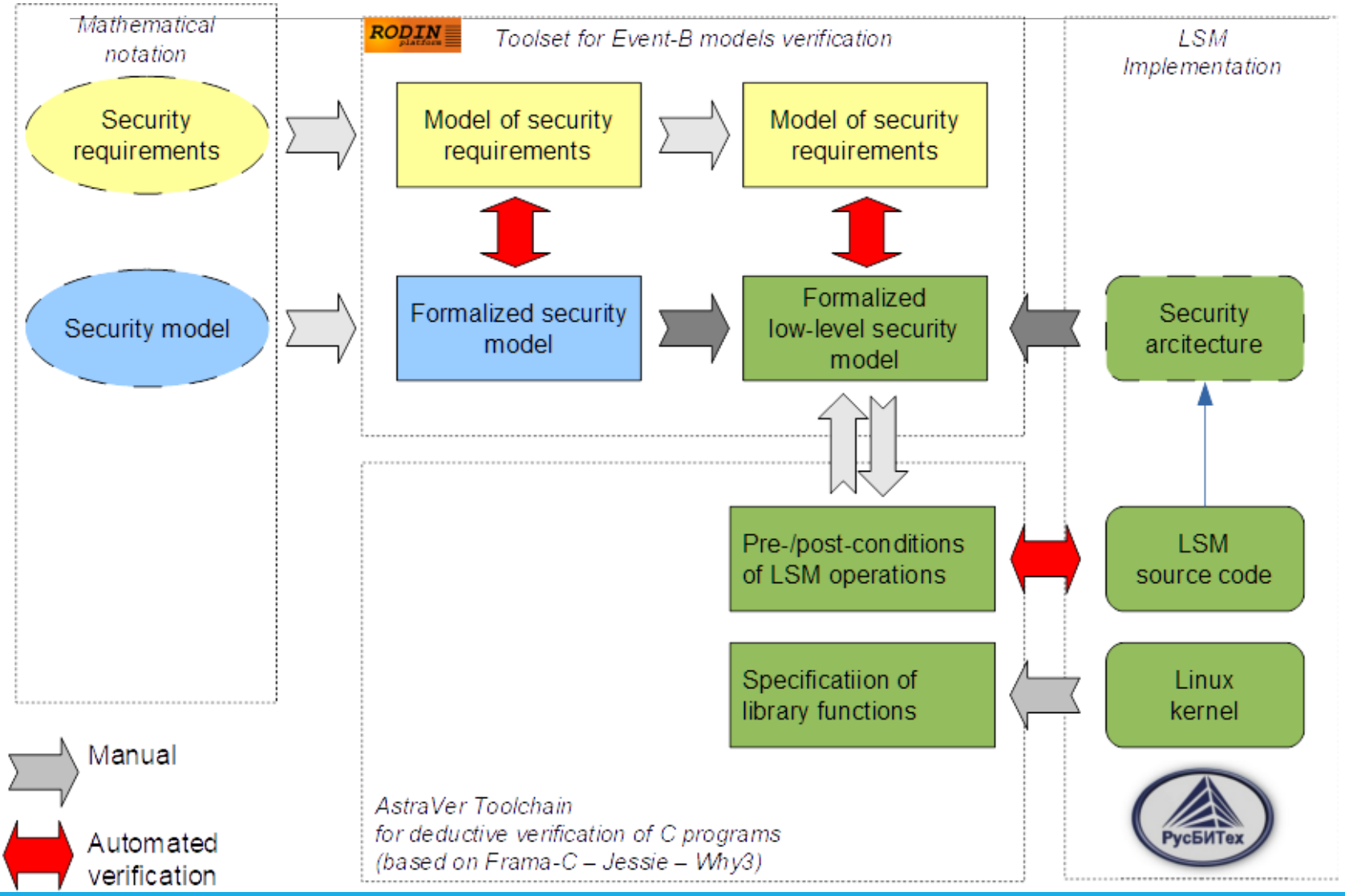
Confidentiality and integrity labels on objects
Confidentiality and integrity levels on subjects

$C(S) < C(O) \Rightarrow \neg(S \longrightarrow^{r} O)$
$C(S) \neq C(O) \Rightarrow \neg(S \longrightarrow^{w} O)$
$I(S) < I(O) \Rightarrow \neg(S \longrightarrow^{w} O)$

SELinux 1998

Based on Linux Security Module (LSM)

Deductive Verification of Security



Code Verification Tools - Problems

Memory model limitations

- Arithmetics with pointers to fields of structures
- Prefix structure casts
- Reinterpret casts

Integer model problems

Limited code support

- Functional pointers
- String literals

Scalability problems

Thank you!

Alexey Khoroshilov khoroshilov@ispras.ru
Victor Kuliamin kuliamin@ispras.ru
Alexander Petrenko petrenko@ispras.ru