8,700 LoC
1 Microkernel
0 Bugs*

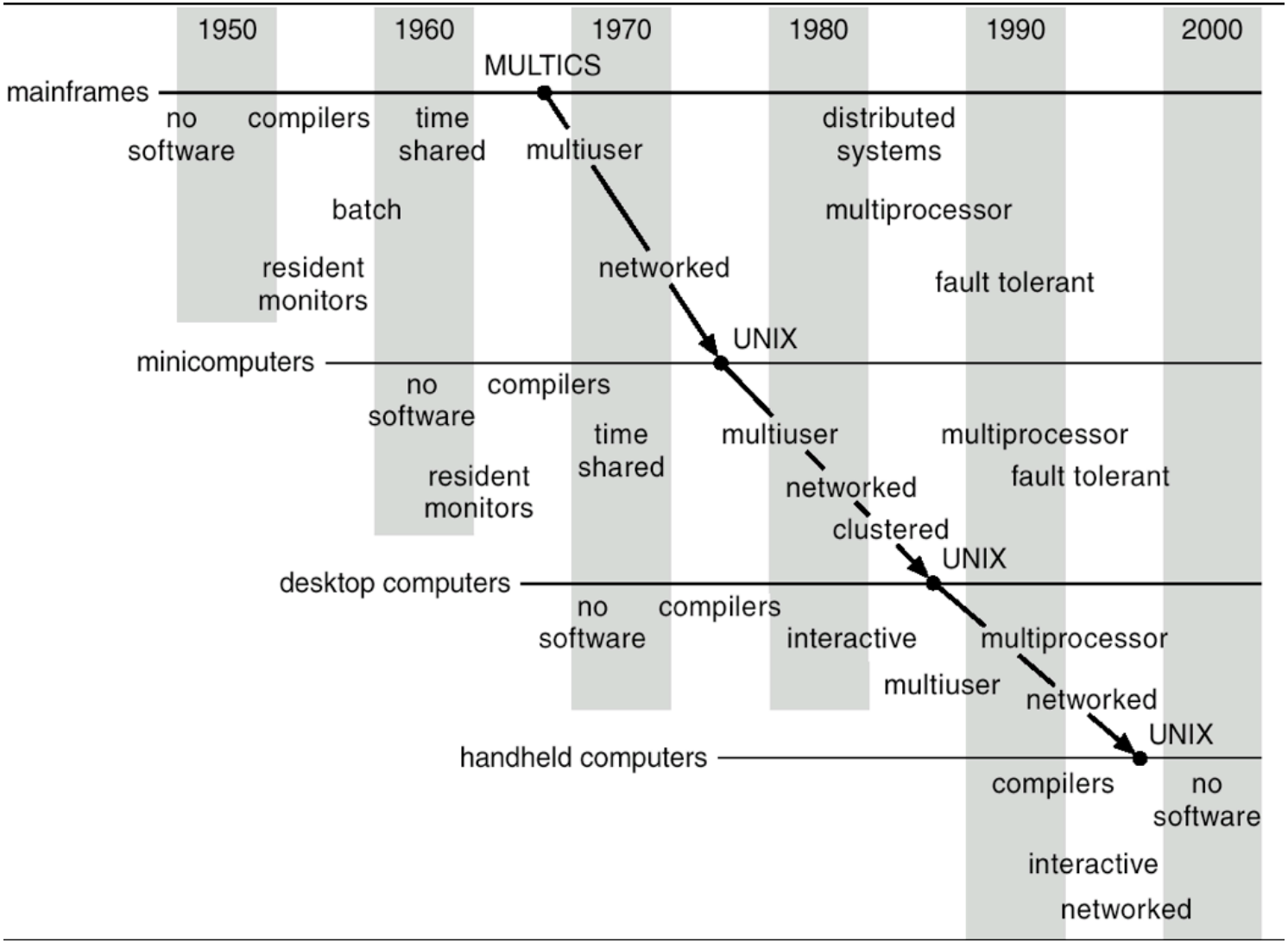
Gernot Heiser

John Lions Professor of Operating Systems, University of New South Wales
Leader, Trustworthy Embedded Systems, NICTA
CTO and Founder, Open Kernel Labs

*Conditions apply


Windows
An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000. It may be possible to continue normally.
* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue

System-Software Timeline



The Problem

Microkernel Approach

NICTA

Small trustworthy foundation

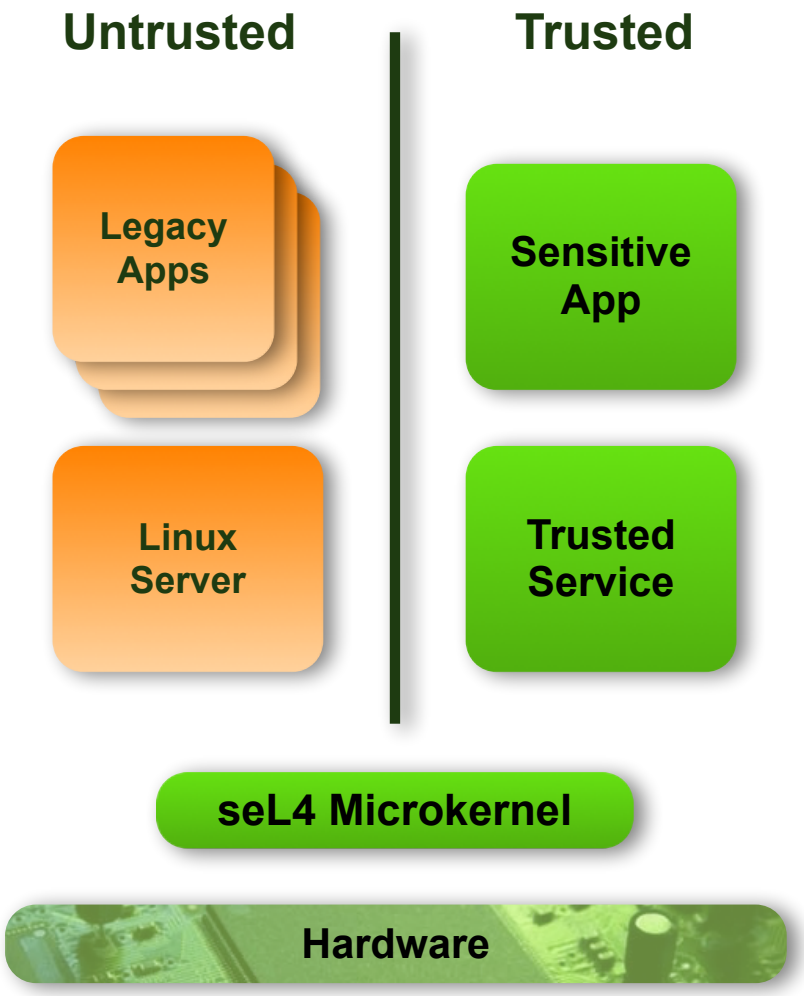
- Fault isolation
- Fault identification
- IP protection
- Modularity
- High assurance components in presence of other

Designed for verification

- small API

Designed for security

- novel kernel resource management



Aim: Suitable for Real-World Use

NICTA

Model: OKL4 microkernel

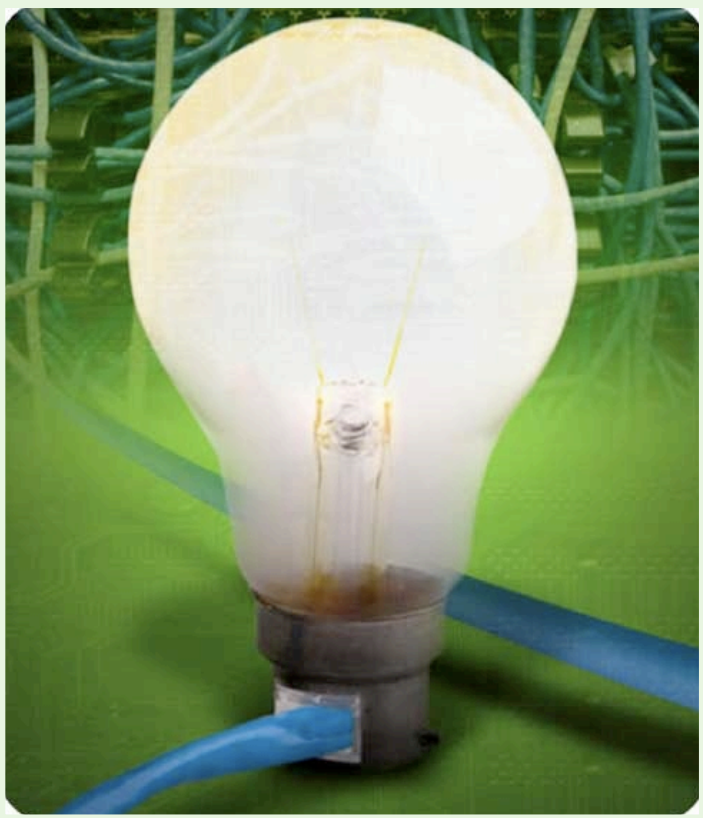
- resulting from L4-based research at NICTA/UNSW
- spun out to independent company Open Kernel Labs in 2006
- deployed in >300 M devices

Open Kernel Labs™
Be open. Be safe.

seL4 API based on L4:

- IPC
- Threads
- Virtual Memory
- IRQs, exception redirection
- Capabilities

The Proof



Functional Correctness

NICTA

What

Proof

How

Specification

```
definition
  schedule :: unit s_monad where
  schedule ≡ do
    threads ← allActiveTCBs;
    thread ← select threads;
    switch_to_thread thread
  od
  OR switch_to_idle_thread
```

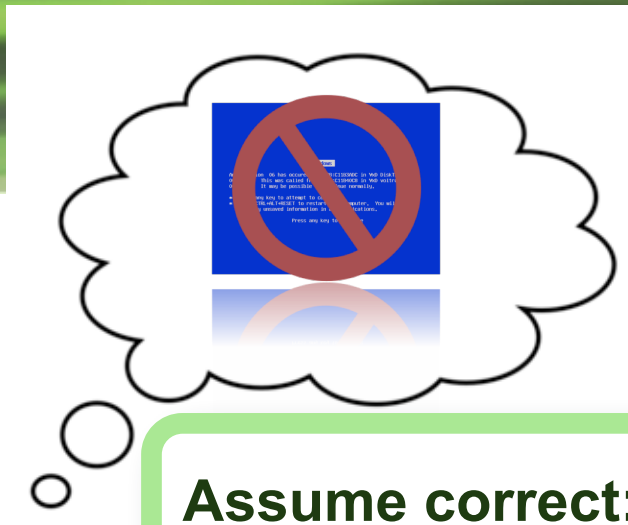
```
void
schedule(void) {
    switch ((word_t)ksSchedulerAction) {
        case (word_t)SchedulerAction_ResumeCurrentThread:
            break;

        case (word_t)SchedulerAction_ChooseNewThread:
            chooseThread();
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;

        default: /* SwitchToThread */
            switchToThread(ksSchedulerAction);
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;
    }
}

void
chooseThread(void) {
    prio_t prio;
    tcb_t *thread, *next;
```

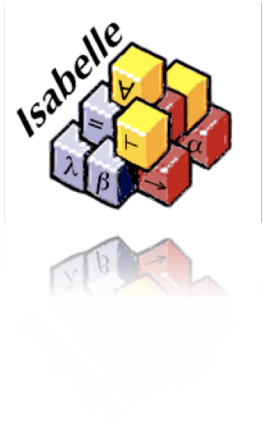
*conditions apply



NICTA

Assume correct:

- compiler + linker (wrt. C op-sem)
- assembly code (600 loc)
- hardware (ARMv6)
- cache and TLB management
- boot code (1,200 loc)



Specificati

Proof

Code

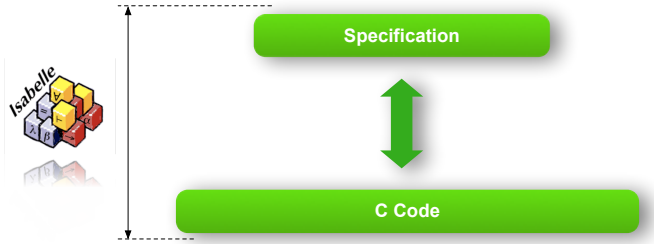
Assumptions

Implications

NICTA

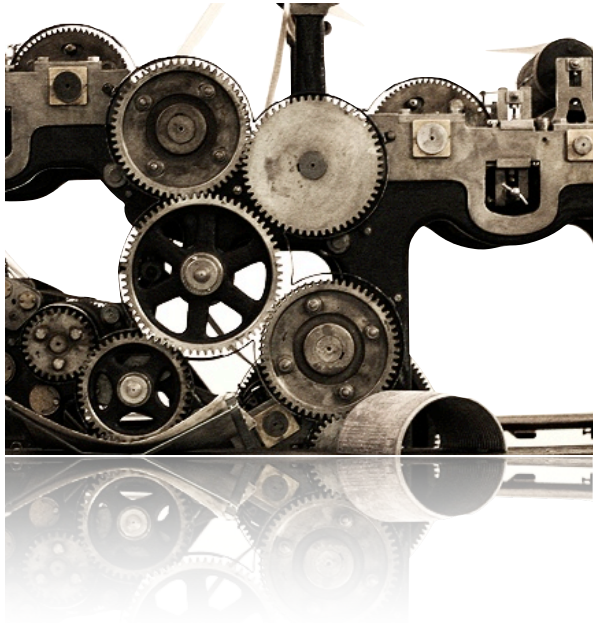
Execution always defined:

- no null pointer de-reference
- no buffer overflows
- no code injection
- no memory leaks/out of kernel memory
- no div by zero, no undefined shift
- no undefined execution
- no infinite loops/recursion

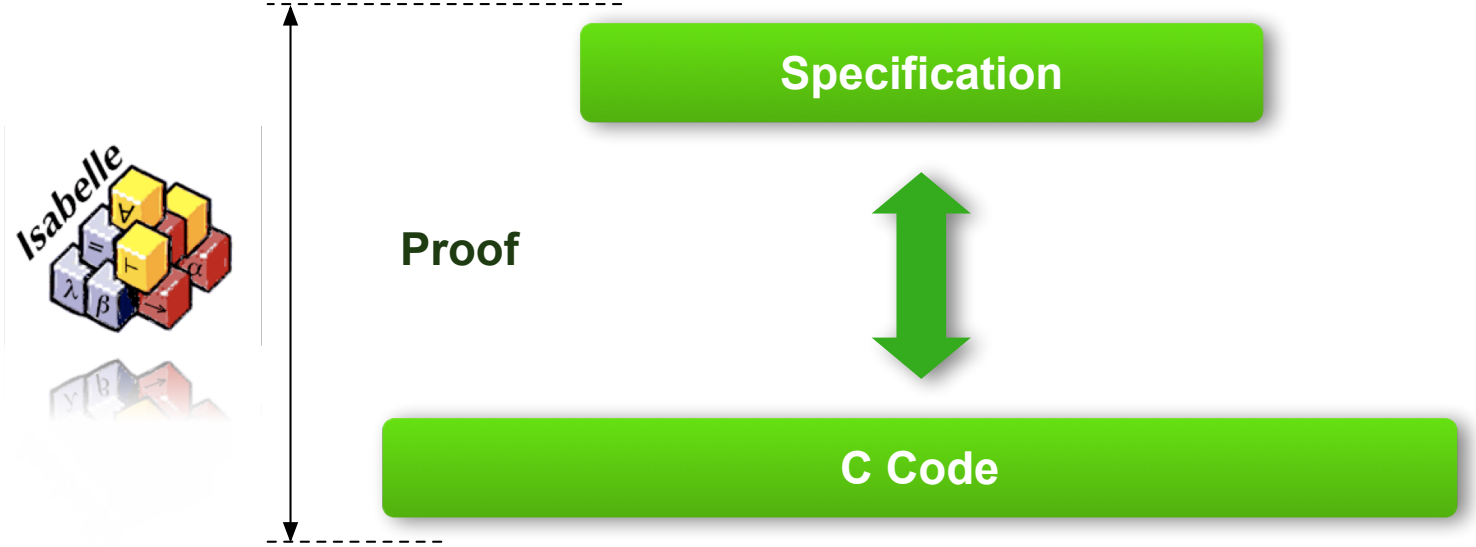


Not implied:

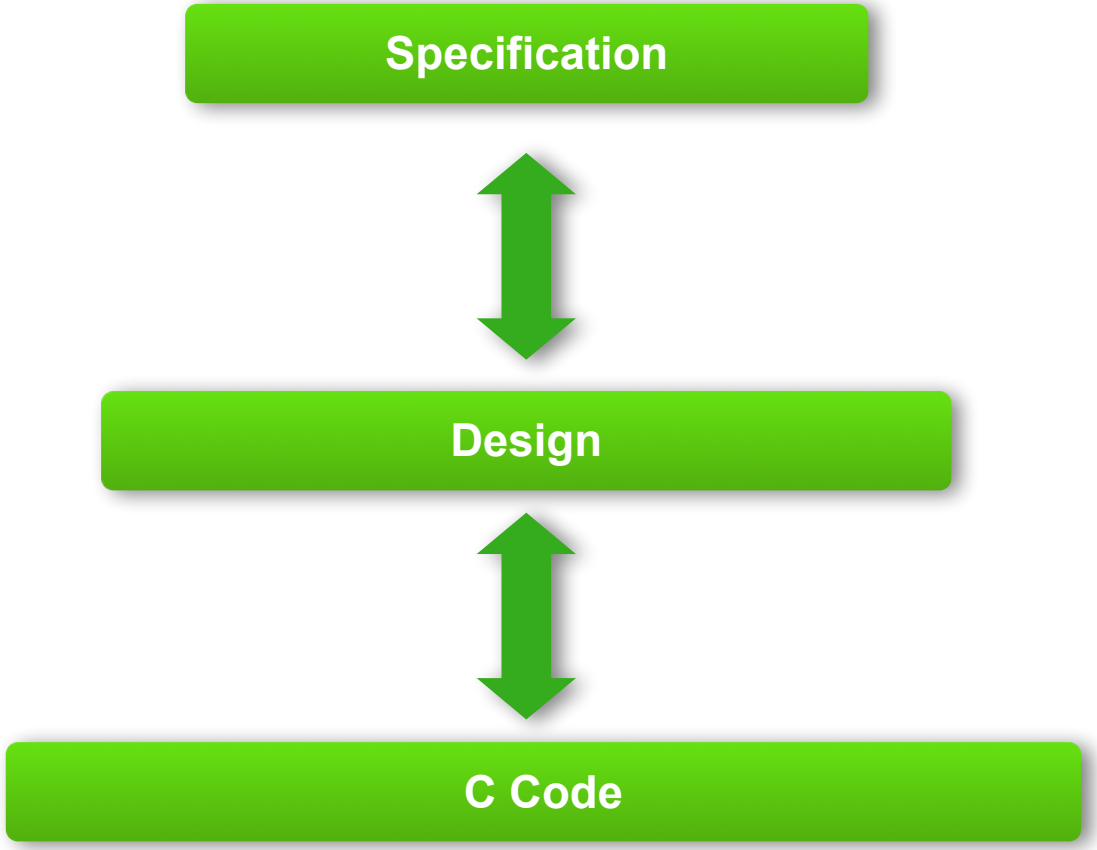
- “secure” (define secure)
- zero bugs from expectation to physical world
- covert channel analysis



Proof Architecture

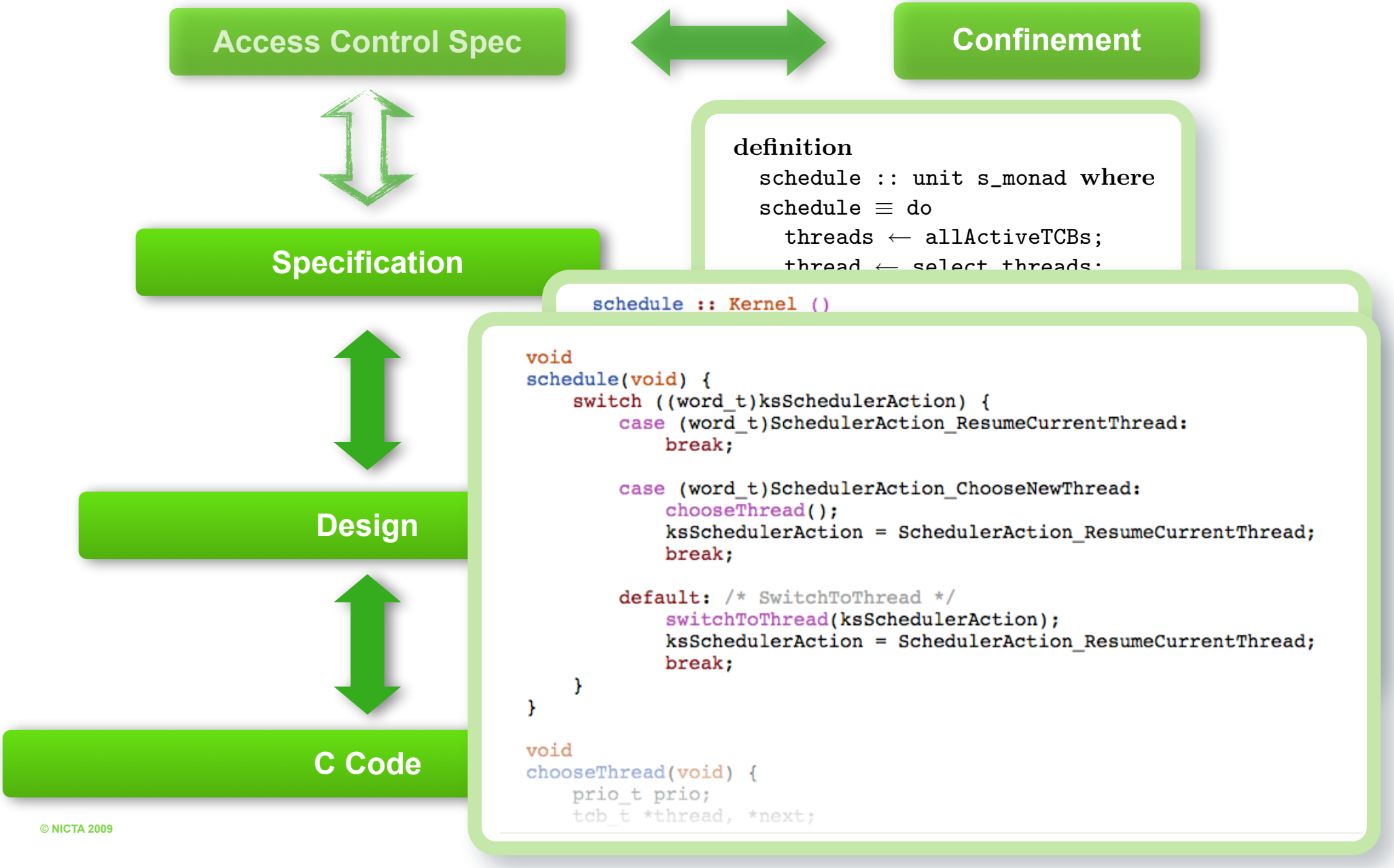


Proof Architecture



Proof Architecture

NICTA



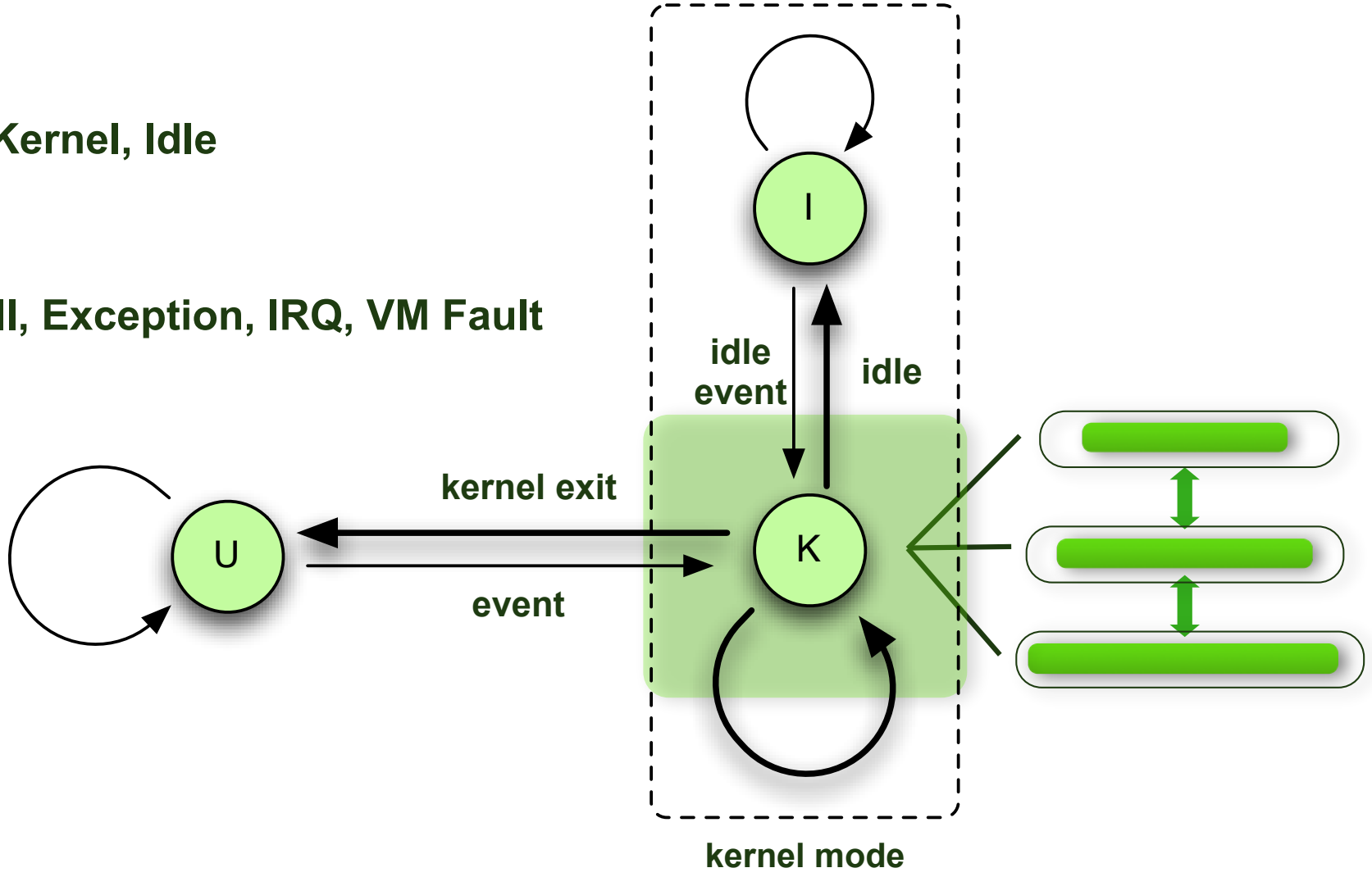
System Model

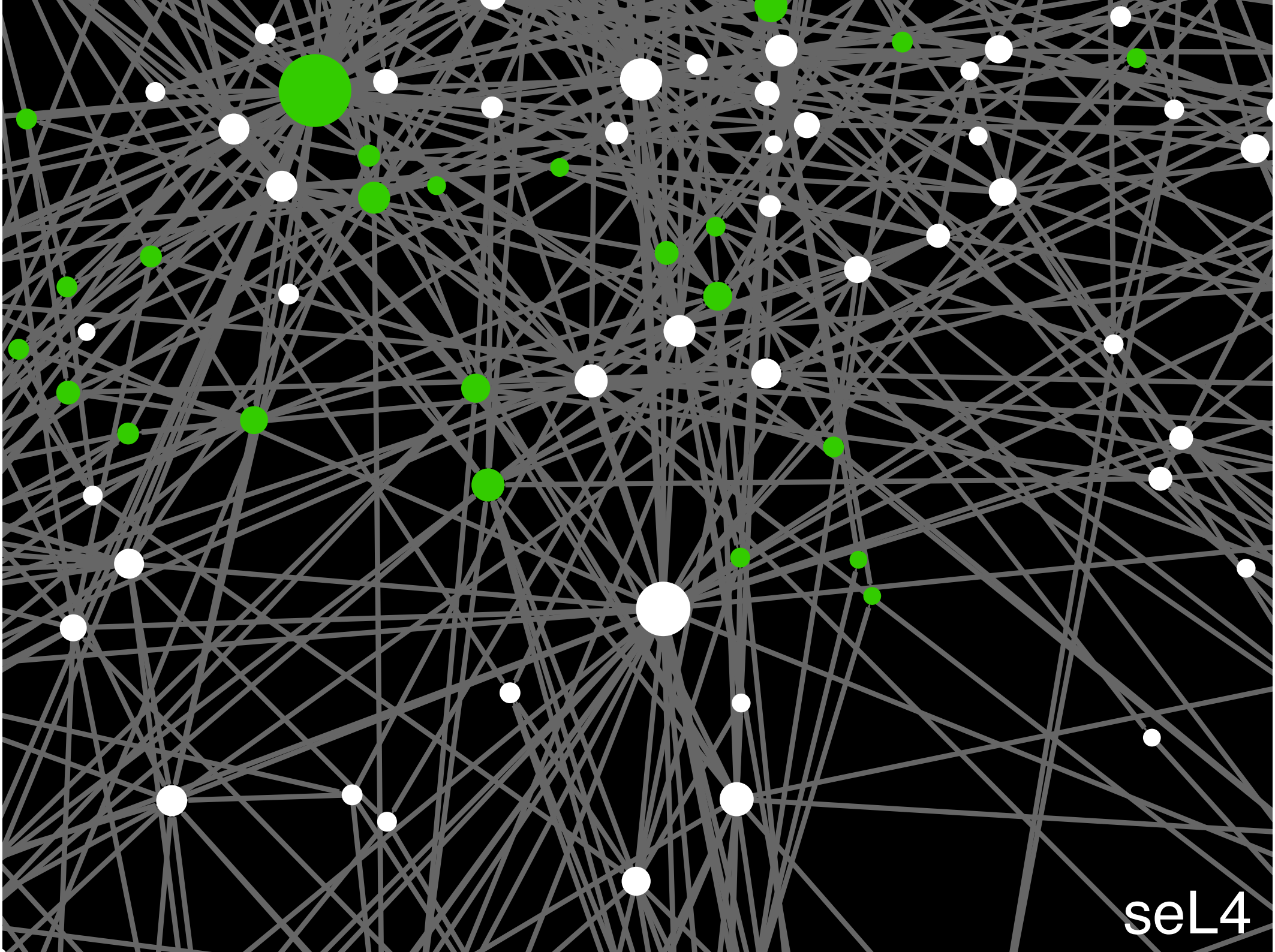
States:

User, Kernel, Idle

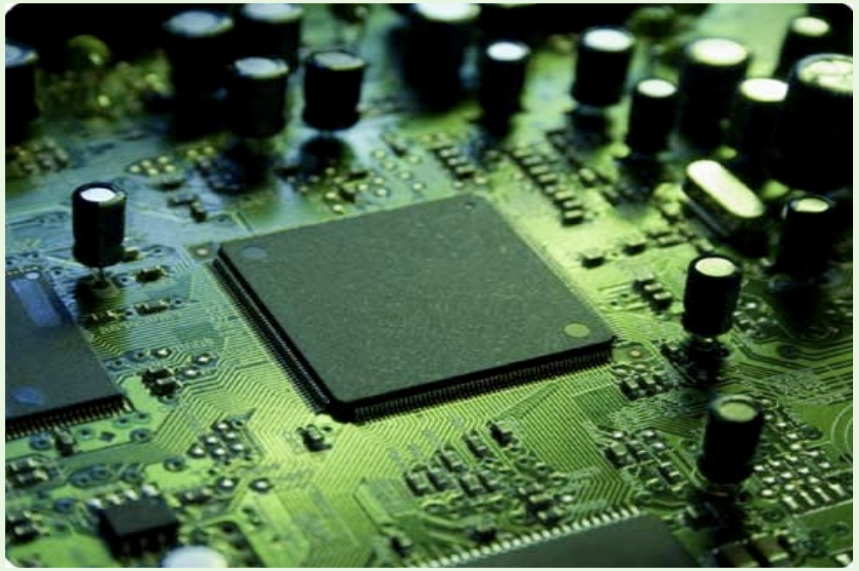
Events:

Syscall, Exception, IRQ, VM Fault




seL4

Kernel Design for Verification

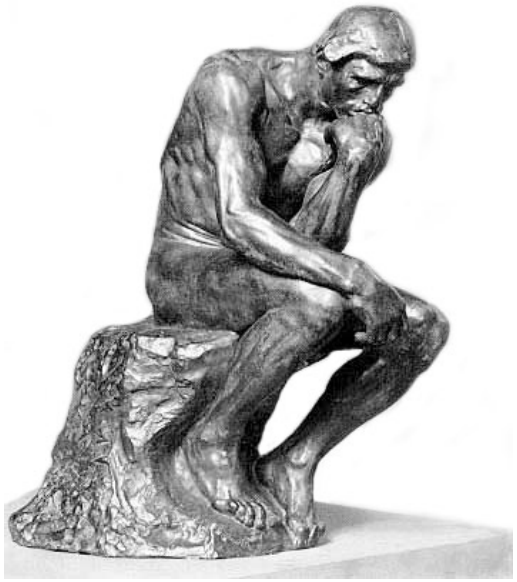


Two Teams

NICTA

Formal Methods Practitioners

Kernel Developers



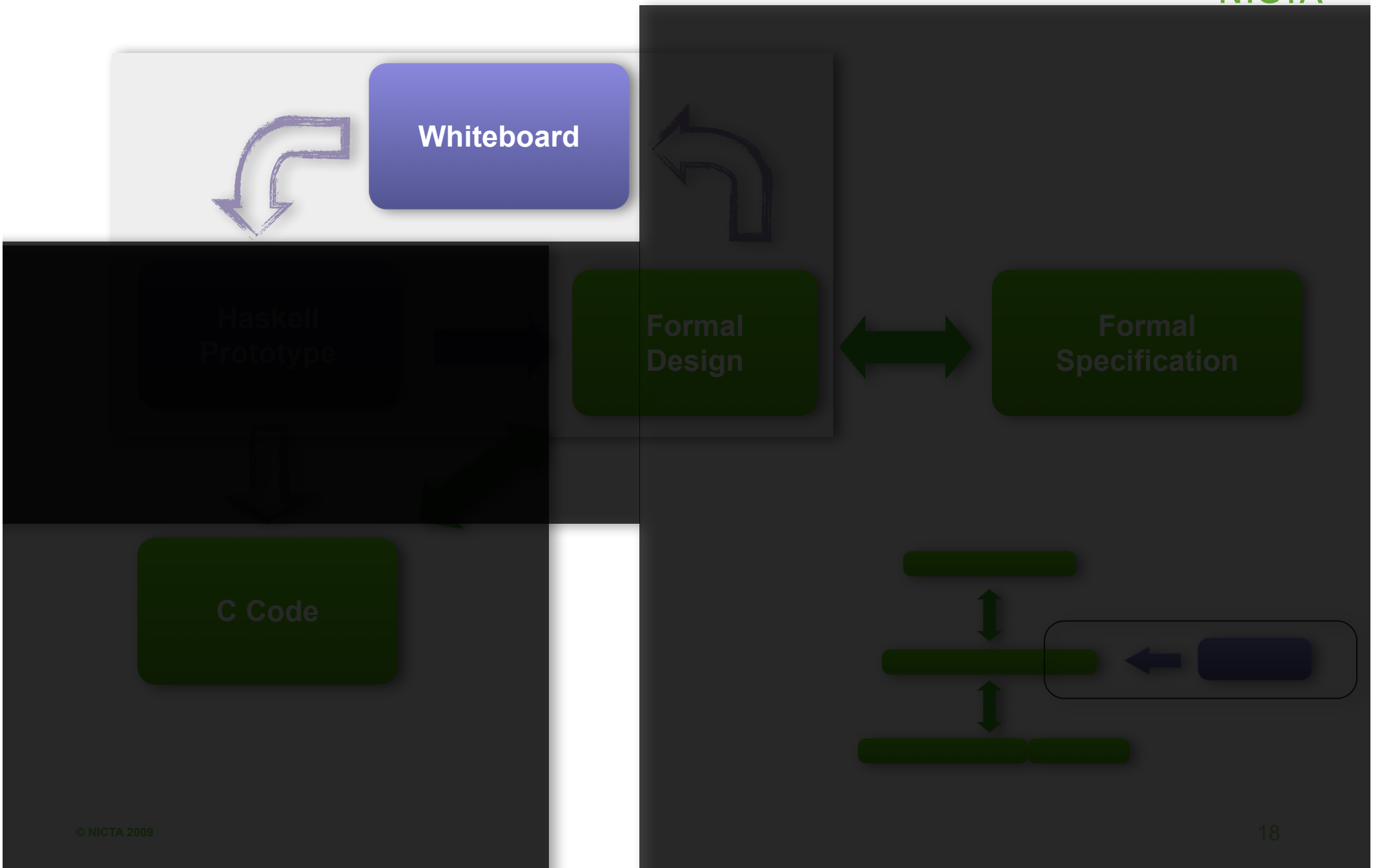
The Power of Abstraction

[Liskov 09]

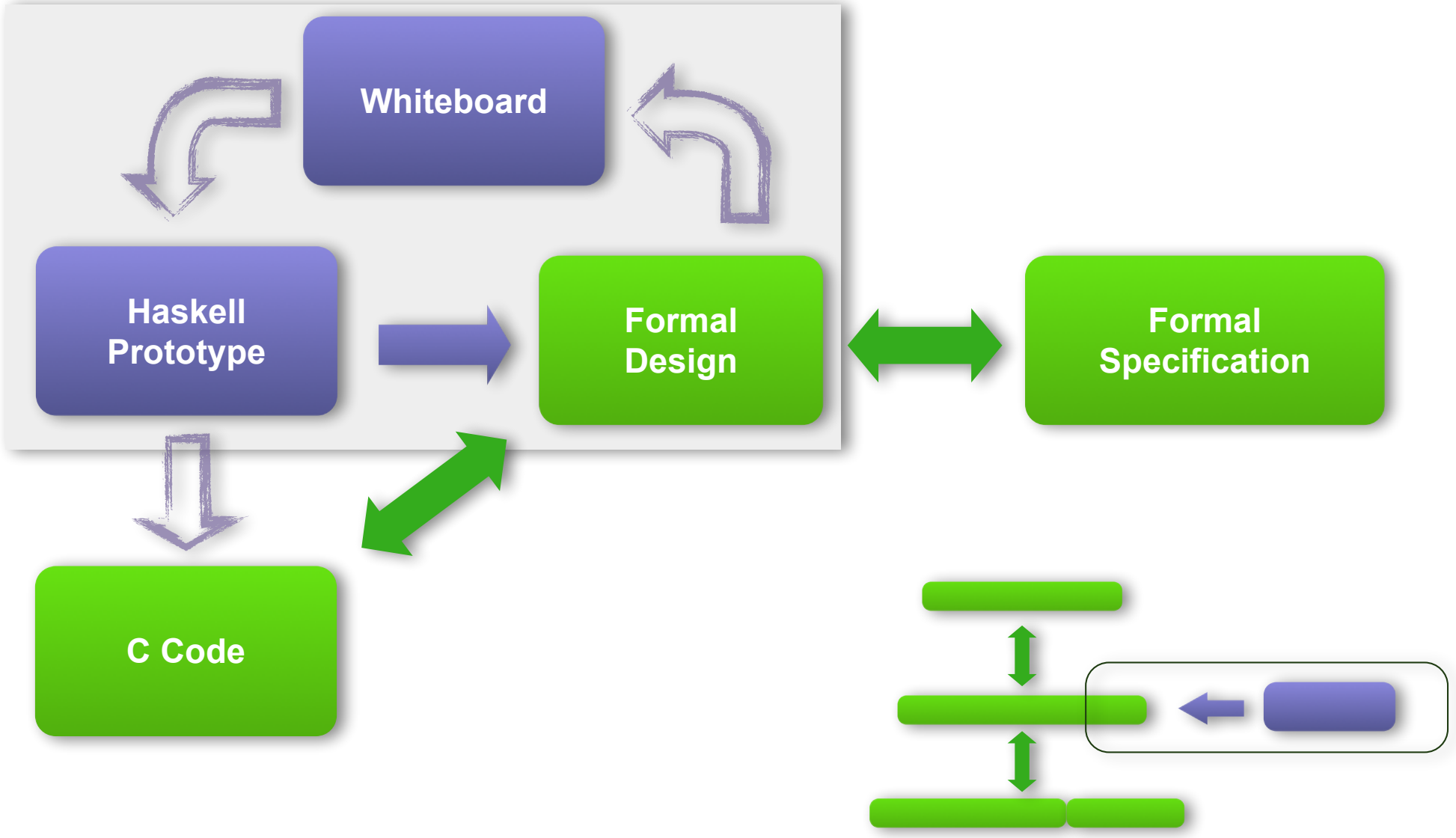
Exterminate All OS Abstractions!

[Engler 95]

Iterative Design and Formalisation



Iterative Design and Formalisation



Design for Verification

NICTA

Reducing Complexity

Hardware

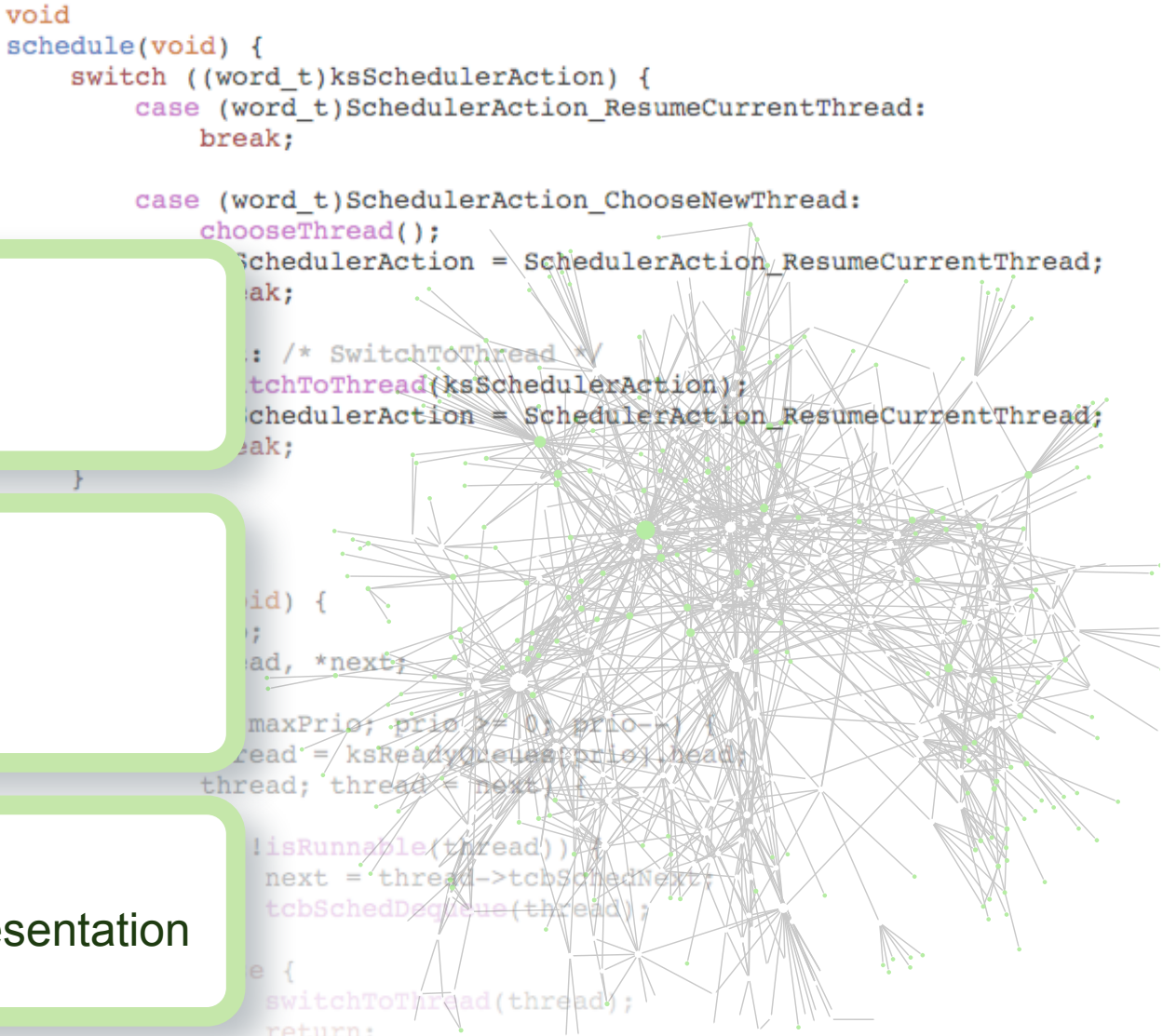
- drivers outside kernel

Concurrency

- event-based kernel
- limit preemption

Code

- derive from functional representation



C subset

NICTA

Everything from C standard

- **including**:
  - pointers, casts, pointer arithmetic
  - data types
  - structs, padding
  - pointers into structs
  - precise finite integer arithmetic

- **minus:**
  - goto, switch fall-through
  - reference to local variable
  - side-effects in expressions
  - function pointers (restricted)
  - unions

- **plus** compiler assumptions on:
  - data layout, encoding, endianess
-

Did you find any Bugs?

NICTA

Bugs found

during testing: 16

during verification:

- in C: 160
- in design: ~150
- in spec: ~150

460 bugs

Effort

Haskell design	2	py
First C impl.	2	weeks
Debugging/Testing	2	months
Kernel verification	12	py
Formal frameworks	10	py
Total	25	py

Comparison of approaches

Trad. engineering	4-6	py
Repeat verification	6	py

Cost

Common Criteria EAL6: **$87M**

L4.verified: **$6M**

Summary

NICTA

Formal proof all the way from spec to C

- **200 kLoC** handwritten, machine-checked proof, **10 k** theorems
- **~460** bugs (160 in C)
- Verification on **code**, **design**, and **spec**
- **Hard in the proof** → **Hard in the implementation**

Formal Code Verification up to 10 kLoC:

It works.
It's feasible.
It's cheaper.

(It's fun, too...)

Future Work

Towards real systems

- 1 MLoC
- real-time analysis
- power management

Remove limitations

- verify assembler code
- verify bootstrap code
- verify MMU operations
- multicore version
- verify x86 version
- temporal isolation
- information flow

The Team (Past and Present)

- June Andronick
- Timothy Bourke
- Andrew Boyton
- David Cock
- Jeremy Dawson
- Philip Derrin
- Dhammika Elkaduwe
- **Kevin Elphinstone**
  - *leader, kernel design*
- Kai Engelhardt
- David Greenaway
- Lukas Haenel
- Gernot Heiser
- **Gerwin Klein**
  - *leader, verification*
- Rafal Kolanski
- Jia Meng
- Catherine Menon
- Michael Norrish
- Thomas Sewell
- David Tsai
- Harvey Tuch
- Michael von Tessin
- Adam Walker
- Simon Winwood

Thank You