UNSW
THE UNIVERSITY OF NEW SOUTH WALES

# COMP9242
# Advanced Operating Systems

## S2/2011 Week 11:
## Local Systems Research

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners
Australian National University
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW GOVERNMENT
State Government Victoria
THE UNIVERSITY OF MELBOURNE
THE UNIVERSITY OF SYDNEY
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
THE UNIVERSITY OF QUEENSLAND AUSTRALIA

# Copyright Notice

**These slides are distributed under the Creative Commons Attribution 3.0 License**

- You are free:
  - to share—to copy, distribute and transmit the work
  - to remix—to adapt the work
- under the following conditions:
  - **Attribution:** You must attribute the work (but not in any way that suggests that the author endorses you or your use of the work) as follows:

    "Courtesy of Gernot Heiser, UNSW/NICTA"

The complete license text can be found at
http://creativecommons.org/licenses/by/3.0/legalcode


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

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

# What's Next?

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Trust Without Trustworthiness

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Core Issue: Complexity

- Massive functionality ⇒ huge software stacks
  - Expensive recalls of CE devices

- Increasing usability requirements
  - Wearable or implanted medical devices
  - Patient-operated
  - GUIs next to life-critical functionality

- On-going integration of critical and entertainment functions
  - Automotive infotainment and engine control

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Our Vision: Trustworthy Systems

We will change industry's approach to the design and implementation of critical systems, resulting in true *trustworthiness*.

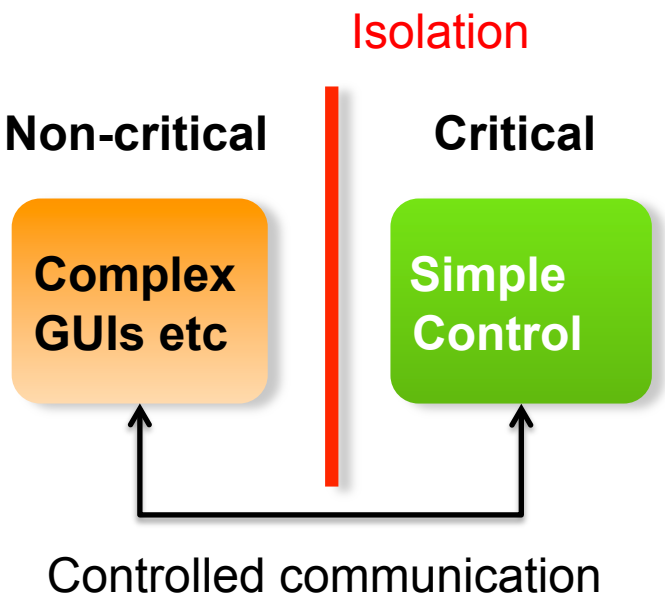
Trustworthy means *highly dependable*, with *hard guarantees* on security, safety or reliability.

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Dealing With Complexity

- Complexity of critical devices will continue to grow
  - Critical systems with millions of lines of code (LOC)
- We need to learn to ensure dependability despite complexity
  - Need to guarantee dependability
- Correctness guarantees for MLOCs unfeasible

- Key to solution: isolation
  - … with controlled communication



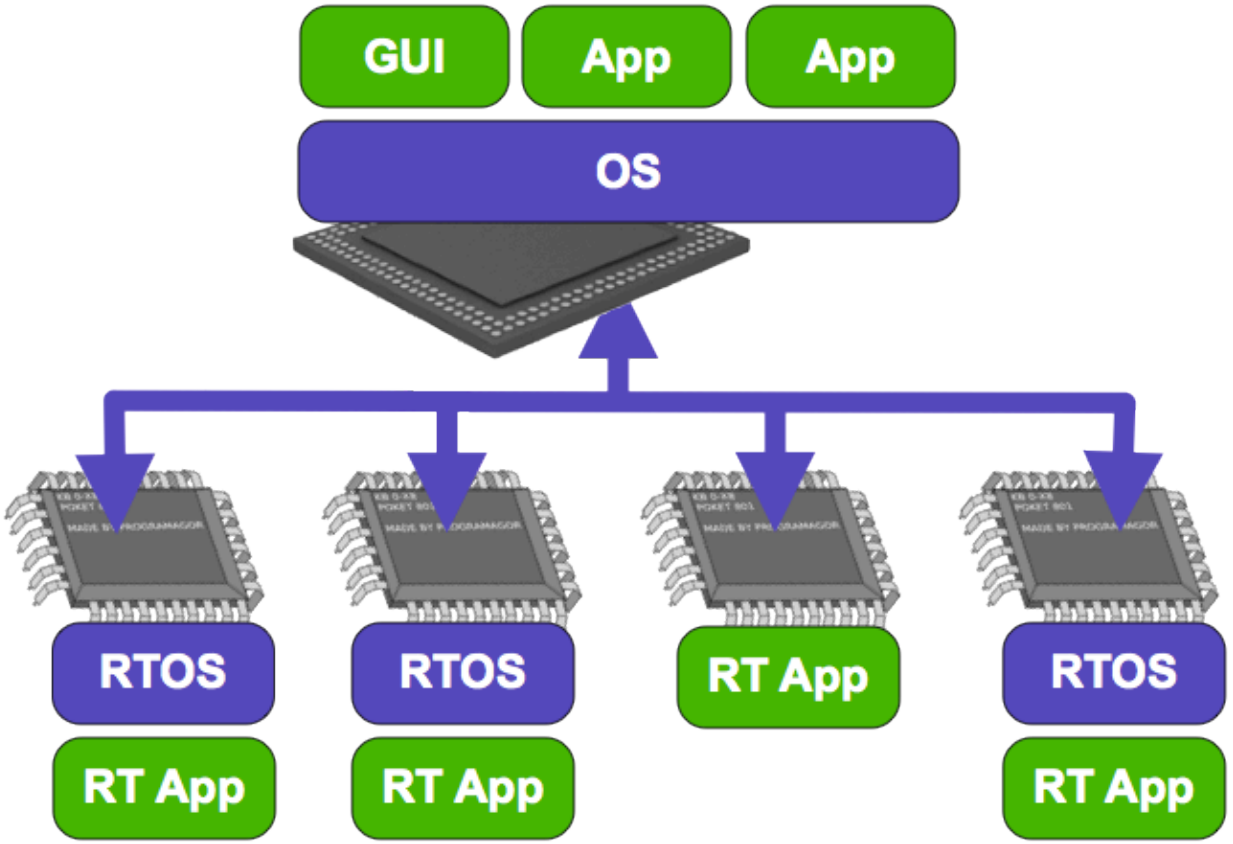


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Isolation: Physical




Dedicated CPUs for critical tasks

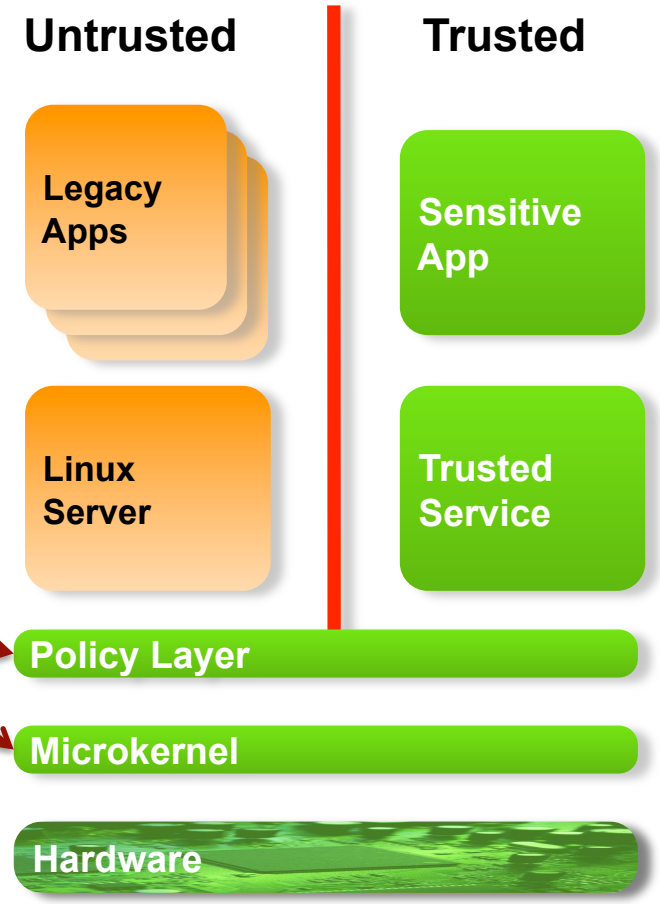



**Cost**: Space, costly interconnects, poor use of hardware


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Isolation: Logical

- Protect critical components by sandboxing complex components
- Provide tightly-controlled communication channels
- *Trustworthy microkernel* provides general mechanisms to enforce isolation
- *Policy layer* defines access rights
- Microkernel becomes core of *trusted computing base*
  - System trustworthiness only as good as microkernel!





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

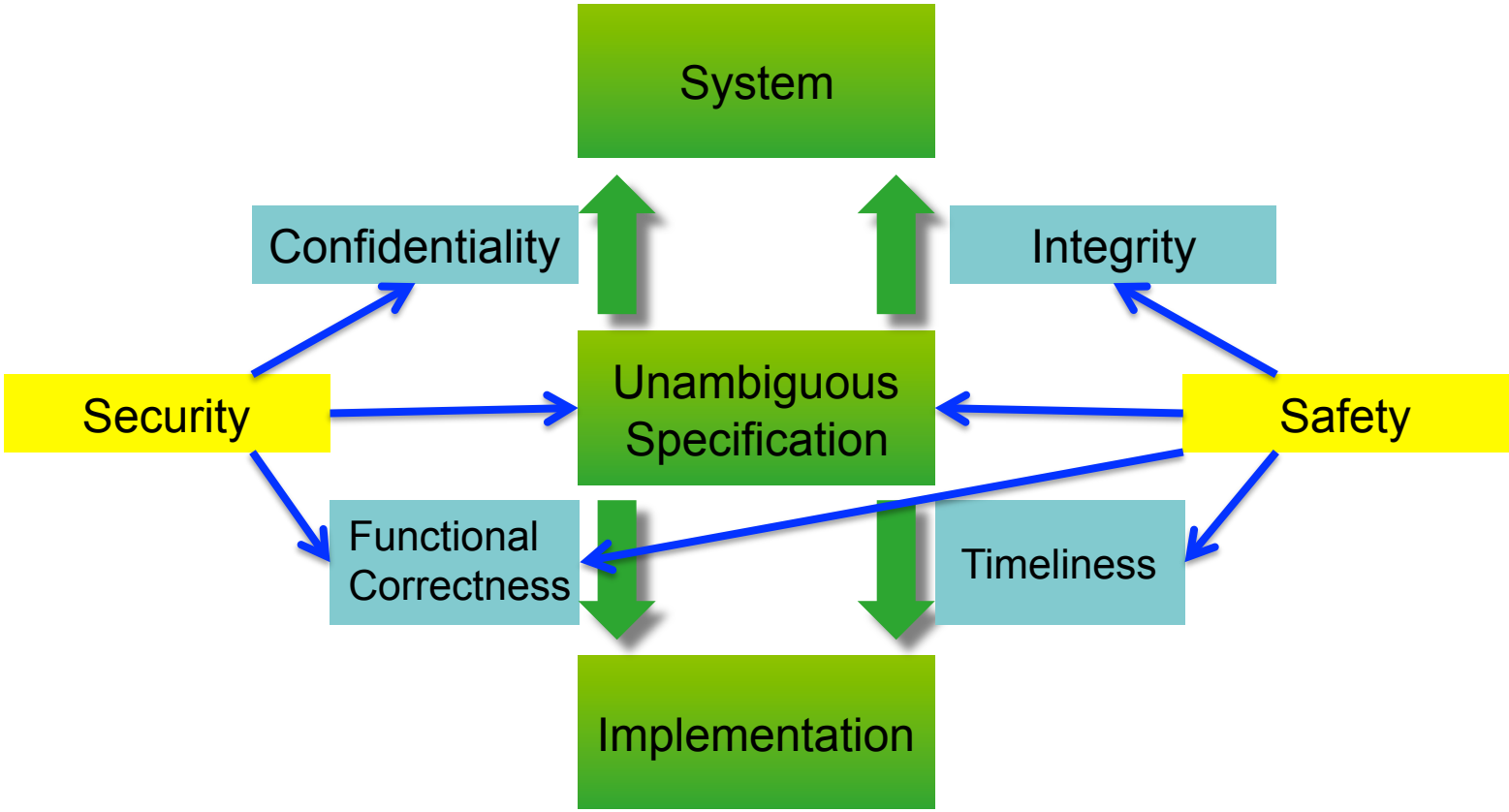
# Isolation Requirements

**To guarantee dependability, following must be guaranteed:**

- Isolation infrastructure impact must be specified
  - To allow reason about operation of isolated critical instances
- Isolation infrastructure must behave as specified
  - Functional correctness
  - Bounded and know worst-case latencies
- Isolation infrastructure must provide actual isolation
  - Integrity guarantees
  - Temporal isolation


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Dependability Requirements





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# NICTA Trustworthy Systems Agenda

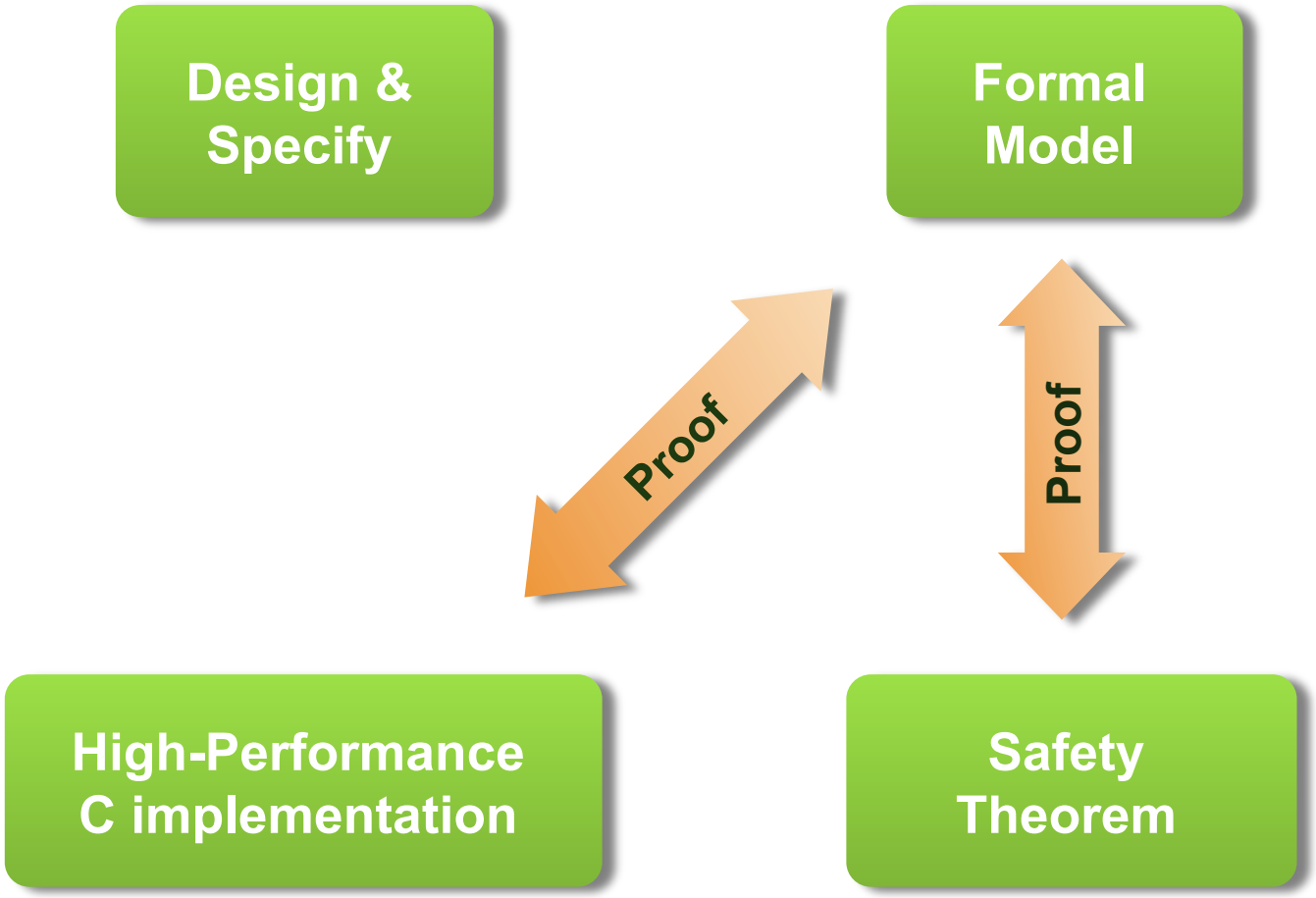
1. **Dependable microkernel (seL4) as a rock-solid base**
   - Formal specification of functionality
   - Proof of functional correctness of implementation
   - Proof of safety/security properties
   - Timeliness guarantees

2. **Lift microkernel guarantees to whole system**
   - Use kernel correctness and integrity to guarantee critical functionality
   - Ensure correctness of balance of trusted computing base
   - Prove dependability properties of complete system


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# seL4: Designing and Formalising

NICTA





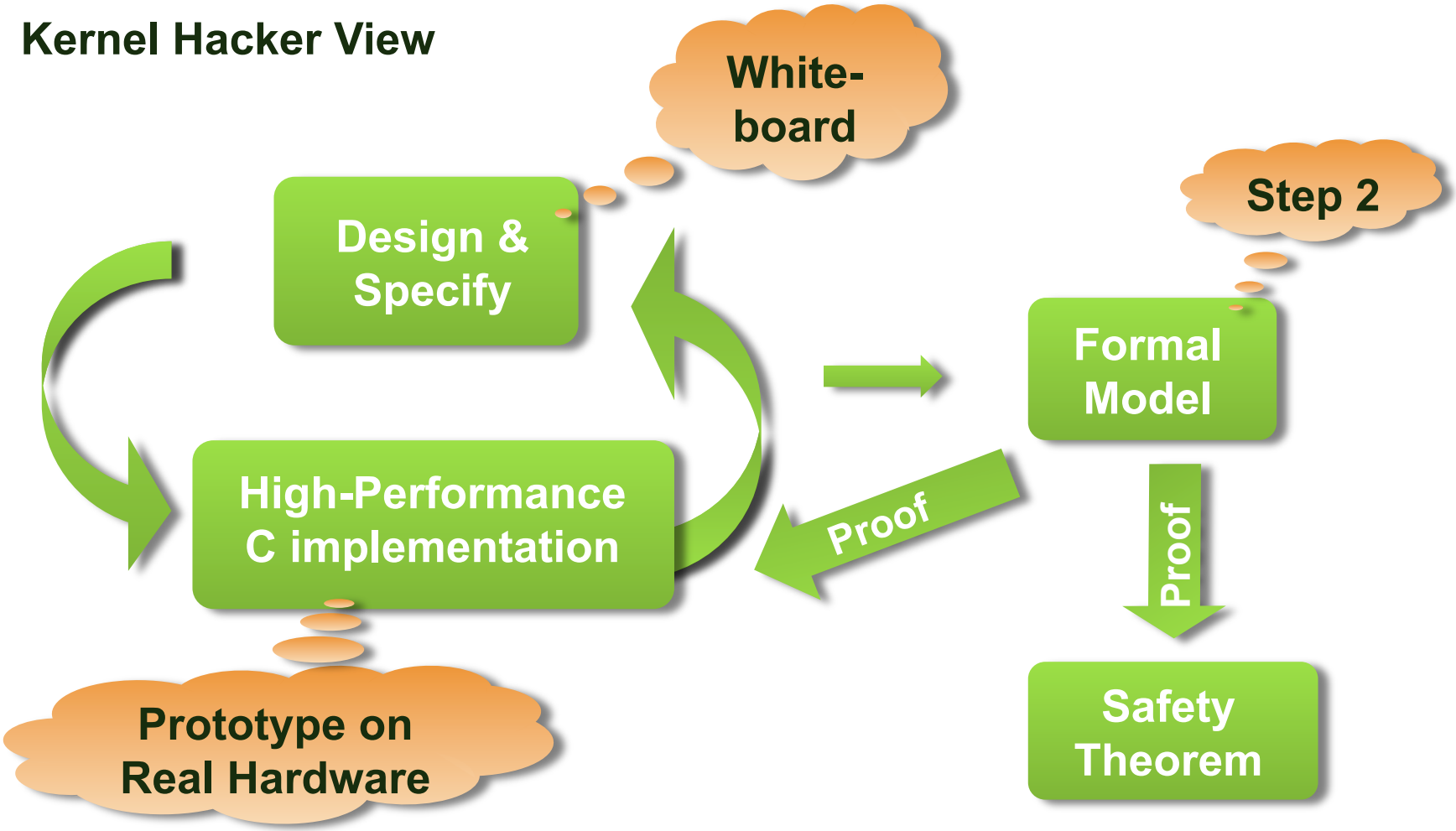
©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Two Mentalities

**Formal Methods Practitioners**
**vs**
**Kernel Developers**

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Standard Kernel Design



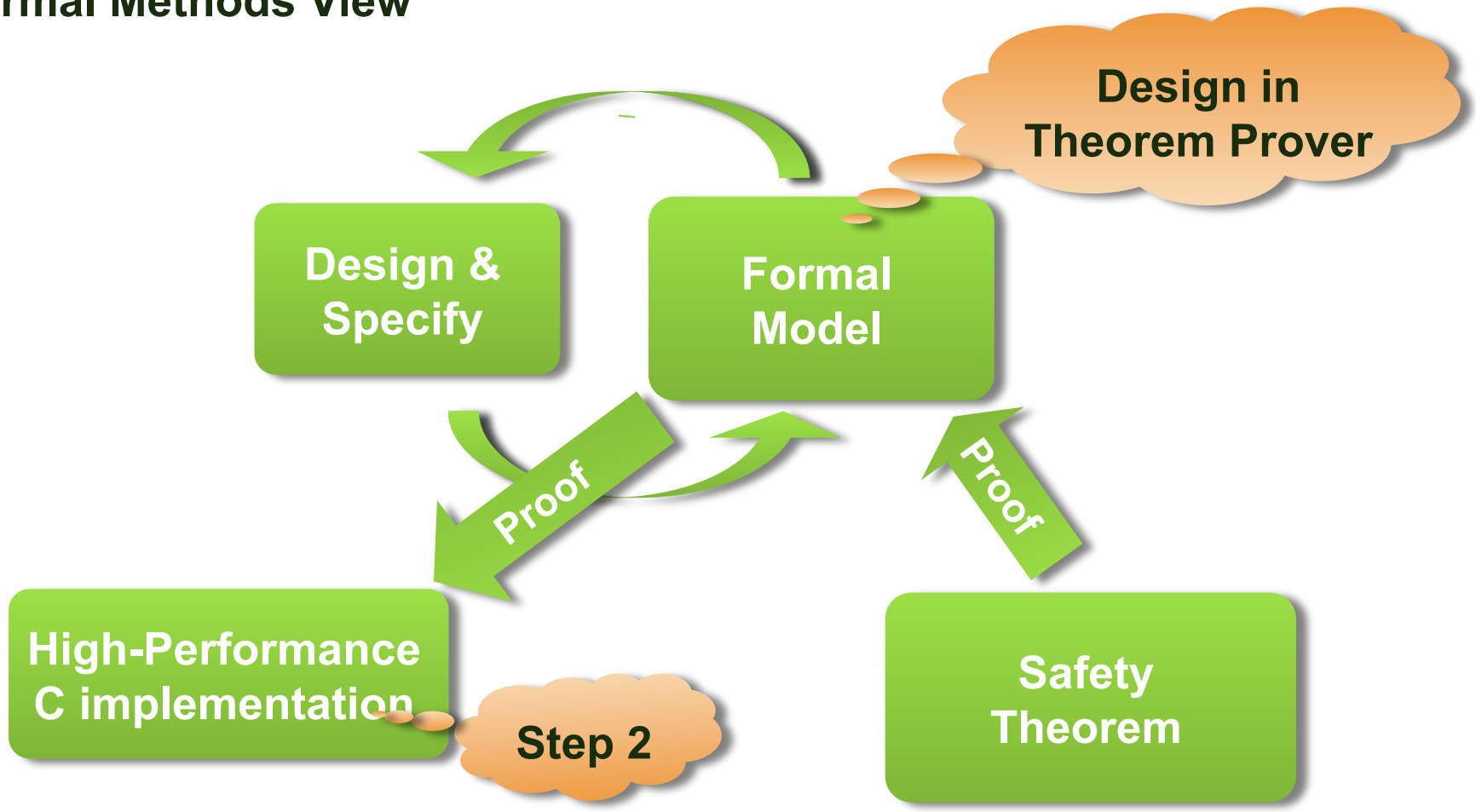


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Formal Design

NICTA

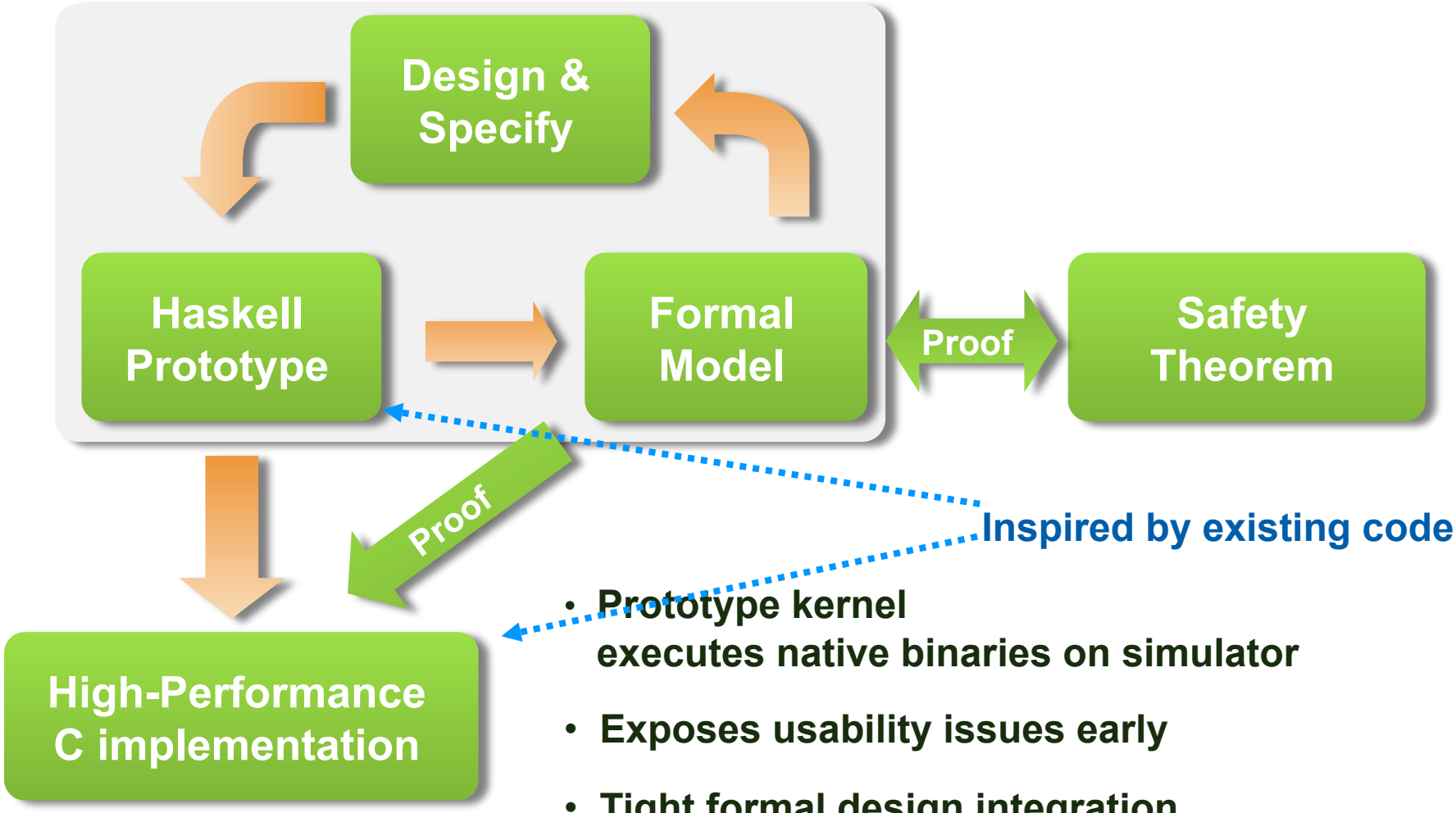
## Formal Methods View





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Iterative Design and Formalisation




- Prototype kernel executes native binaries on simulator
- Exposes usability issues early
- Tight formal design integration


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

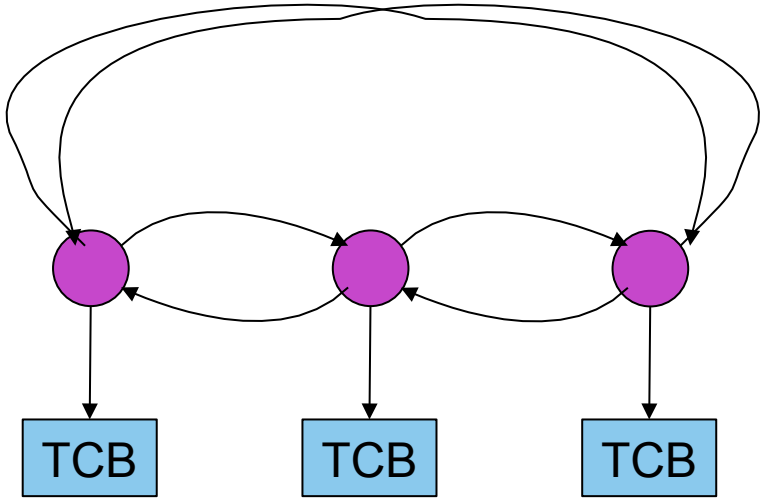
# Kernel Design for Verification

- Main objective: minimise complexity
  - global invariants must be proven for each state change
  - must prove pre- and post-conditions for statements/blocks
  - effort determined by complexity of conditions and state change
- … without sacrificing performance
- Affects design in many ways
  - global variables, side effects
  - kernel memory management
  - concurrency and non-determinism
  - I/O

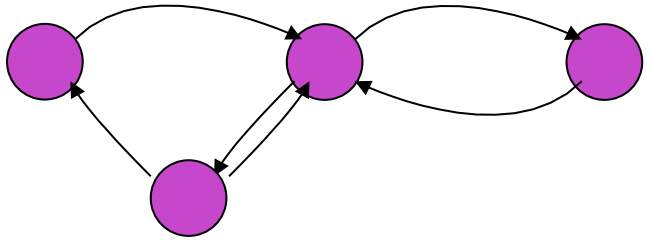

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Global Variables

- Not a difficulty per se, but potential source of complexity
- Eg: scheduler queue as doubly-linked list
  - Show that
    - all pointers are to valid nodes
    - front- and back-pointers are consistent
    - nodes point to TCBs



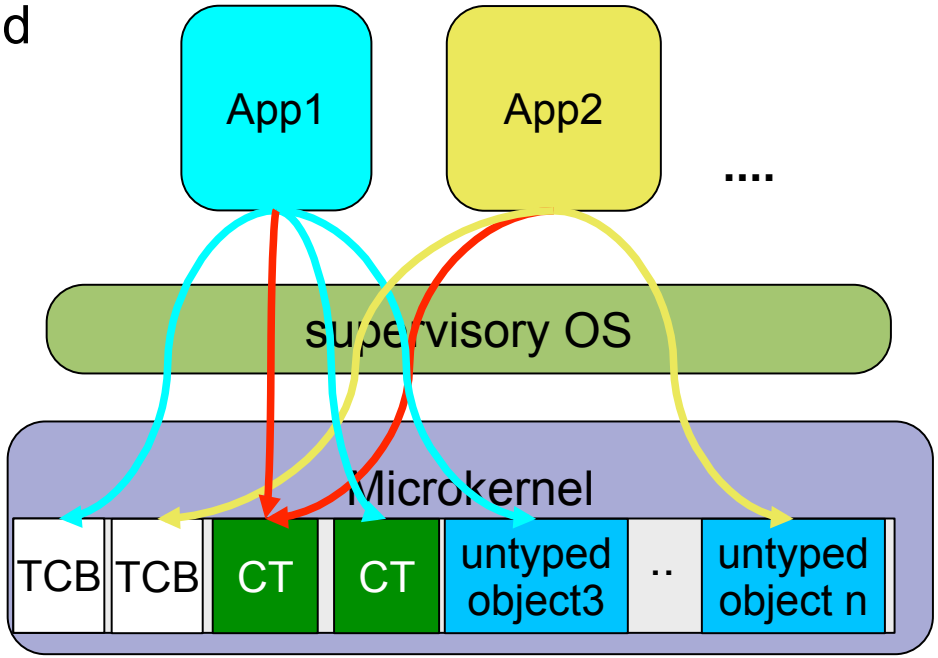

- Requires proof that any pointer operation maintains invariants
- Challenge is temporary violation
  - eg adding a node
  - Requires ensuring atomicity




©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Kernel Memory

- seL4 kernel memory management model pushes policy to userland
  - aids verification
  - need to ensure strict hierarchy
  - capability derivation tree
- Challenge is re-use
  - most difficult part of verification!
  - use derivation tree to detect all references
  - global data structure that requires invariants in all parts of the system





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Concurrency

- Proofs about concurrent programs are inherently hard!
- seL4 strictly limits concurrency to the bare minimum
  - Single processor
    - multicore via big kernel lock or multikernel approach
  - User-level device drivers
  - Non-preemptible, event-based
    - single kernel stack
  - Interrupt points to limit real-time latencies
    - poll interrupt status
    - insert new kernel event (ahead of user)
    - return to user boundary and re-enter kernel
    - allows maintaining all invariants


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Concurrency

**Preempting object destruction:**

- Keep one cap as zombie during object cleanup
  - only retained to reference partially cleaned-up object
  - stores state of cleanup, maintaining invariants
  - attempt by preemptor to remove zombie can just execute

**Exceptions in kernel:**

- Prevent memory exceptions
  - ensure kernel page tables are complete
  - map into every address space
- Disallow other exceptions
  - verification is its own friend ☺


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# I/O

- Mostly a non-issue
  - user-level drivers
    - IOMMU support for DMA security
  - non-preemptible kernel
- Exception is timer tick
  - essentially a source of interrupts
  - handled in-kernel as separate event
  - no real complication

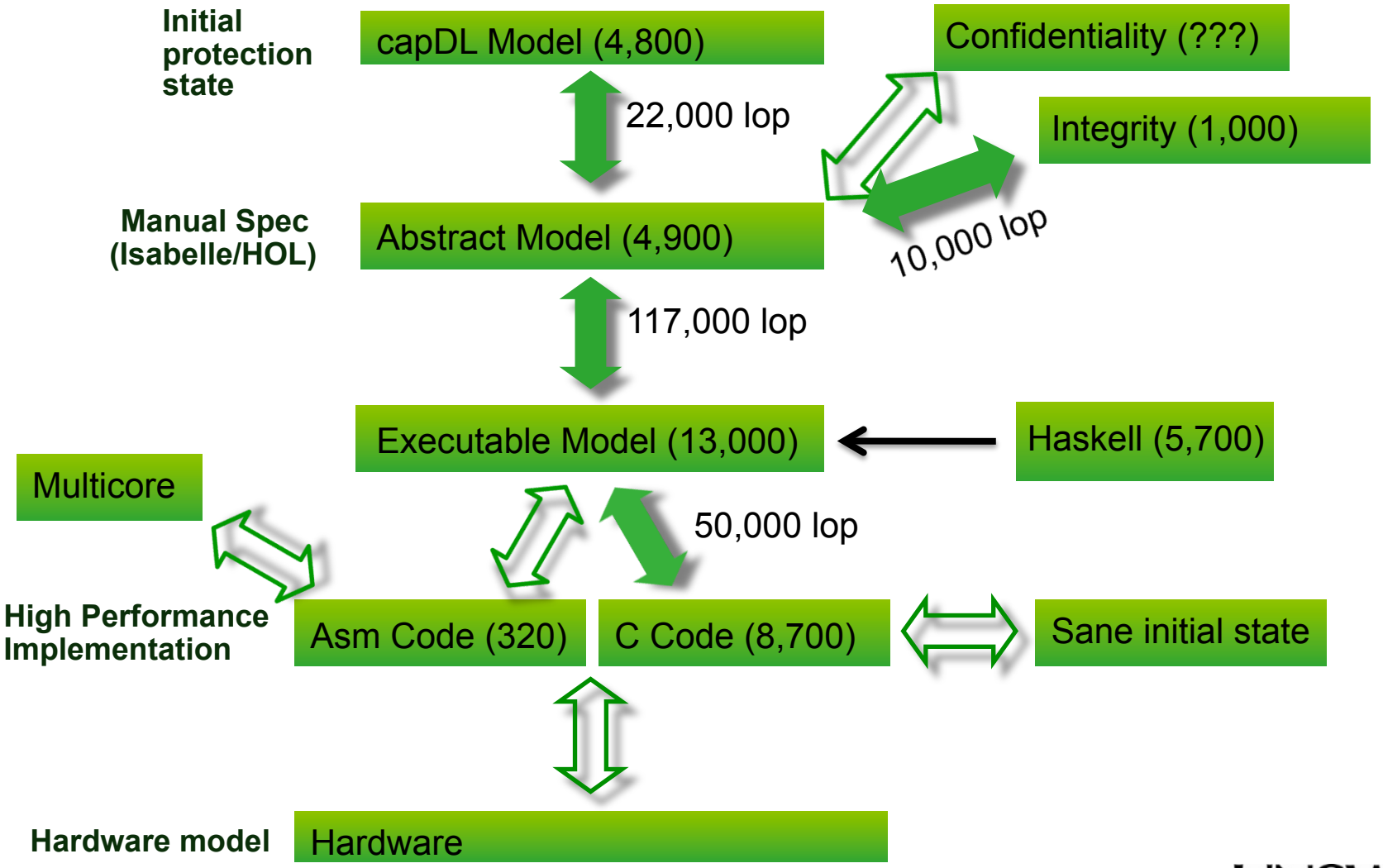

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Lessons for Kernel Design

- Need to reduce complexity forced simple and clean design
  - beneficial even with traditional validation
  - does not necessarily impact performance
- Some design decision beneficial for other reasons too
  - single kernel stack for memory footprint
  - interrupt handling by polling has performance advantages

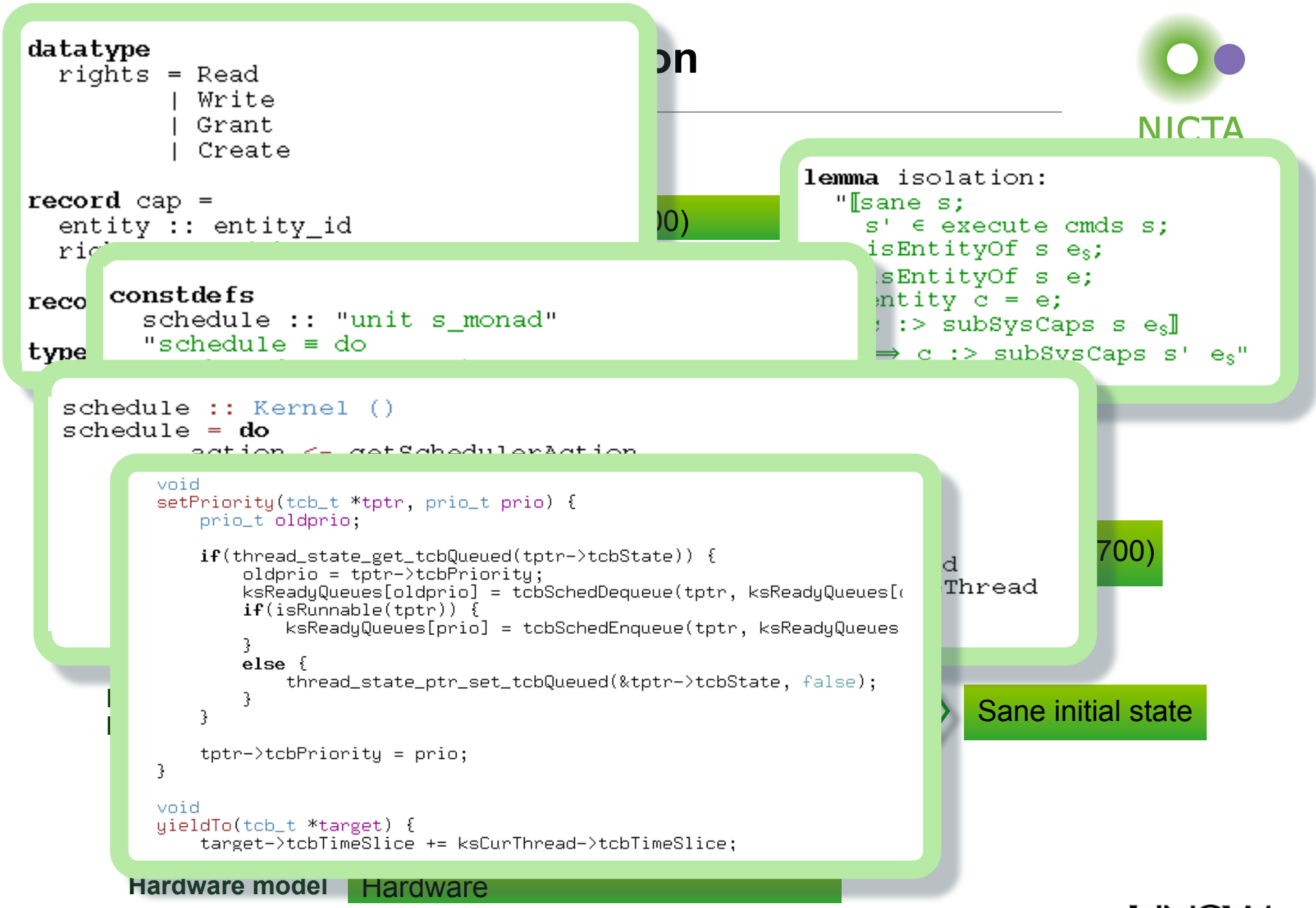

©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Kernel Functional Verification





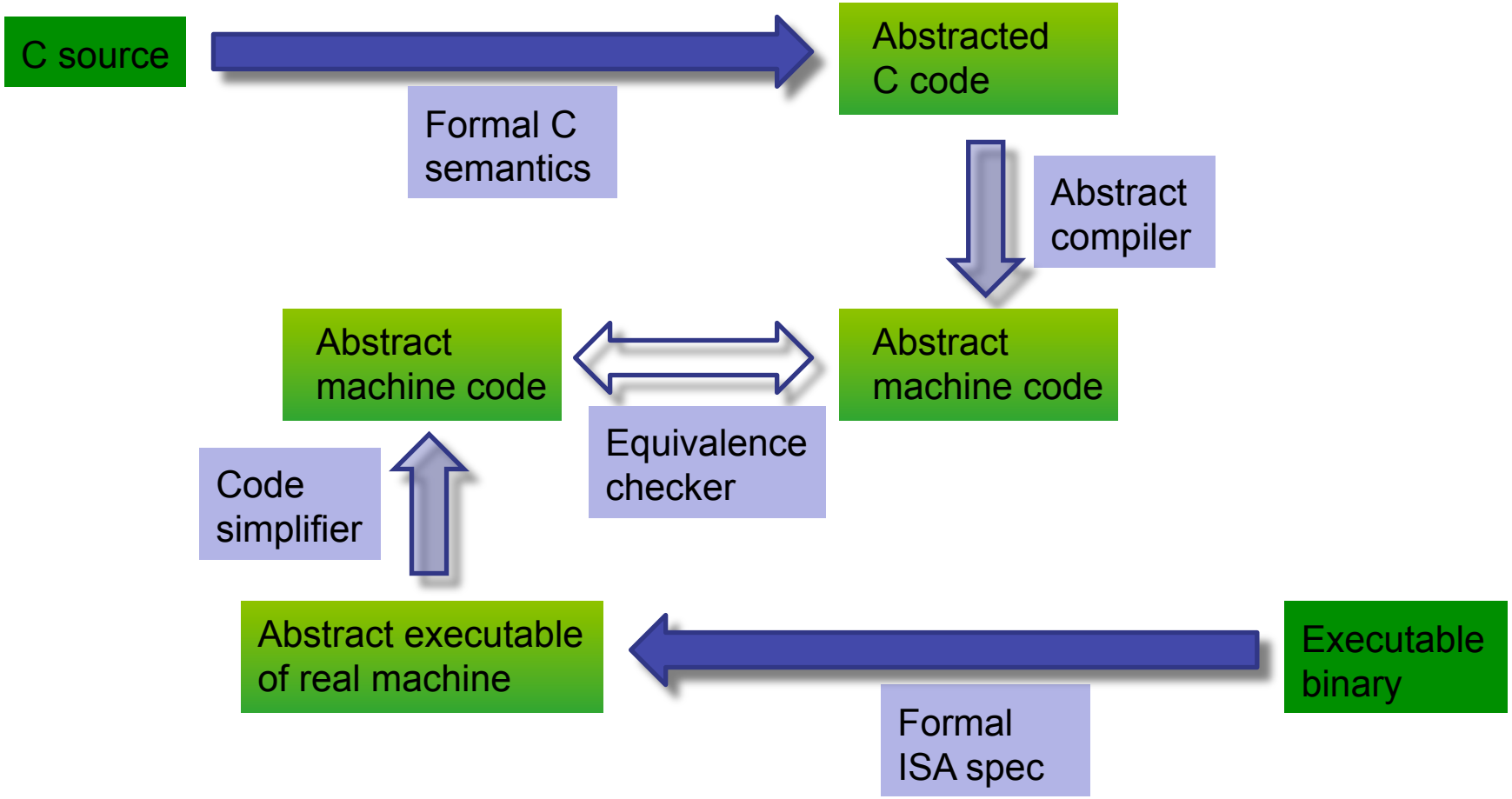
©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Binary Code Verification (In Progress)





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Formal Verification Summary

**Kinds of properties proved**

- Behaviour of C code is fully captured by abstract model
- Behaviour of C code is fully captured by executable model
  - Can prove many interesting properties on higher-level models
- Kernel never fails, behaviour is always well-defined
  - assertions never fail
  - will never de-reference null pointer
  - cannot be subverted by misformed input
- All syscalls terminate, reclaiming memory is safe, ...
- Well typed references, aligned objects, kernel always mapped…
- Access control is decidable

**Effort:**

- Average 6 people over 5.5 years
- Only 50–100% more than comparable (low-assurance) projects


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Verification vs Certification

## Common Criteria: Military-Strength Security

| Evaluation Level | Requirements | Functional Specification | Top Down Design | Imple-mentation | Cost |
|---|---|---|---|---|---|
| EAL1 | | Informal | | | |
| EAL2 | | Informal | Informal | | |
| EAL3 | | Informal | Informal | | |
| EAL4 | | Informal | Informal | Informal | |
| EAL5 | | Semi-formal | Semi-formal | Informal | |
| EAL6 | Formal | Semi-formal | Semi-formal | Informal | 1K/LoC |
| EAL7 | Formal | Formal | Formal | Informal | |
| seL4 | Formal | Formal | Formal | Formal | 0.6K/LoC |


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Kernel Worst-Case Execution Time

**Issues for WCET analysis of seL4:**

- Need knowledge of worst-case interrupt-latency
  - Longest non-preemptible path + IRQ delivery cost
  - seL4 runs with interrupts disabled
    - System calls in well-designed microkernel are short!
    - Strategic preemption points in long-running operations
    - Optimal average-case performance with reasonable worst-case
- Applications also need to know cost of system calls
  - Need WCET analysis of all possible code paths


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Kernel Worst-Case Execution Time

**Challenges for WCET analysis of OS kernels in general:**

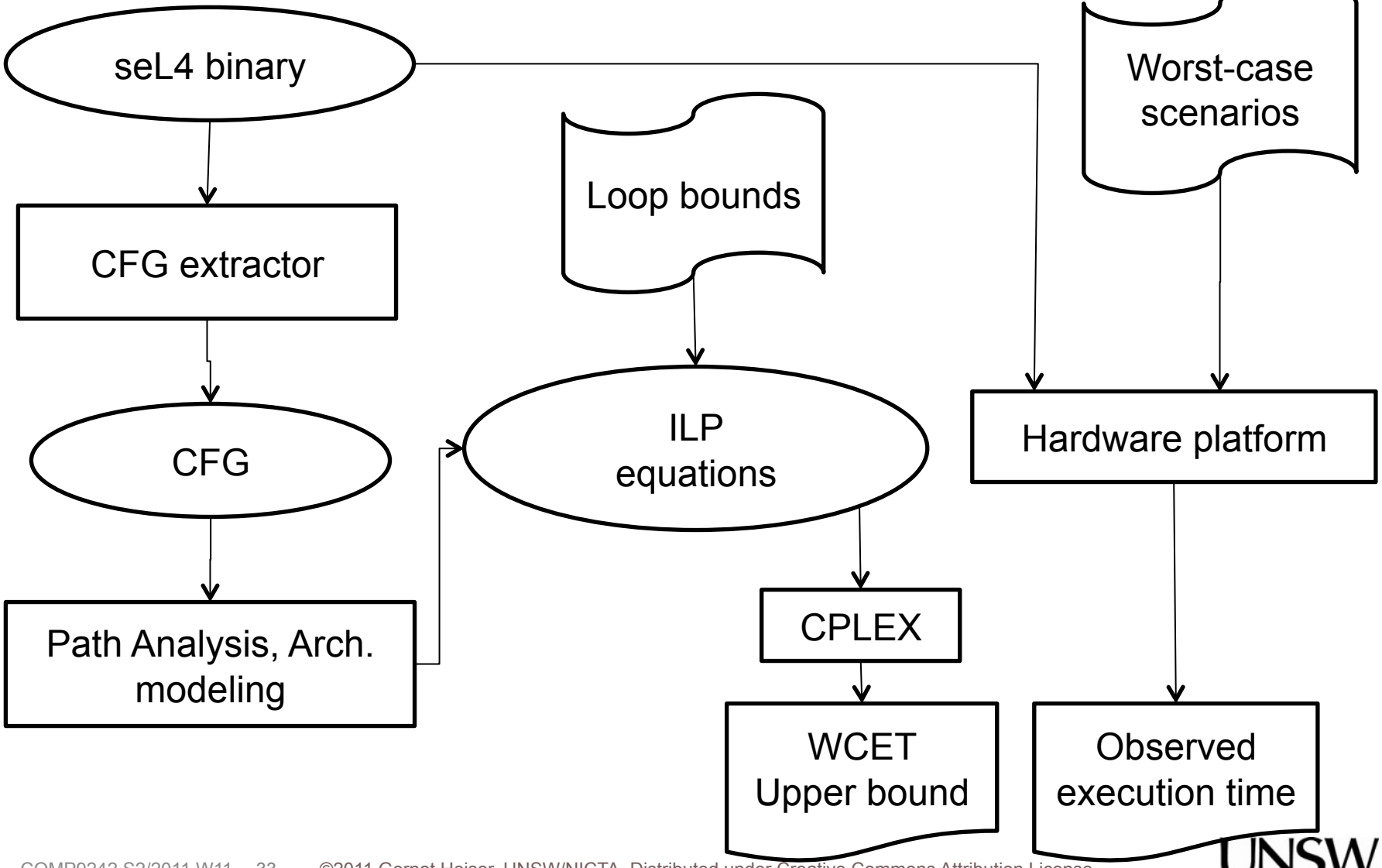
- Kernel code notoriously unstructured
- Low-level system-specific instructions
- Context-switching
- Assembly code

**seL4-specific advantages:**

- (Relatively) structured design (evolved from Haskell prototype)
- Event-based kernel (single kernel stack)
- Small (as far as operating systems go!)
- No function pointers in C
- Preemption points are explicit and preserve code structure
- Memory allocation performed in userspace


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# WCET analysis process





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

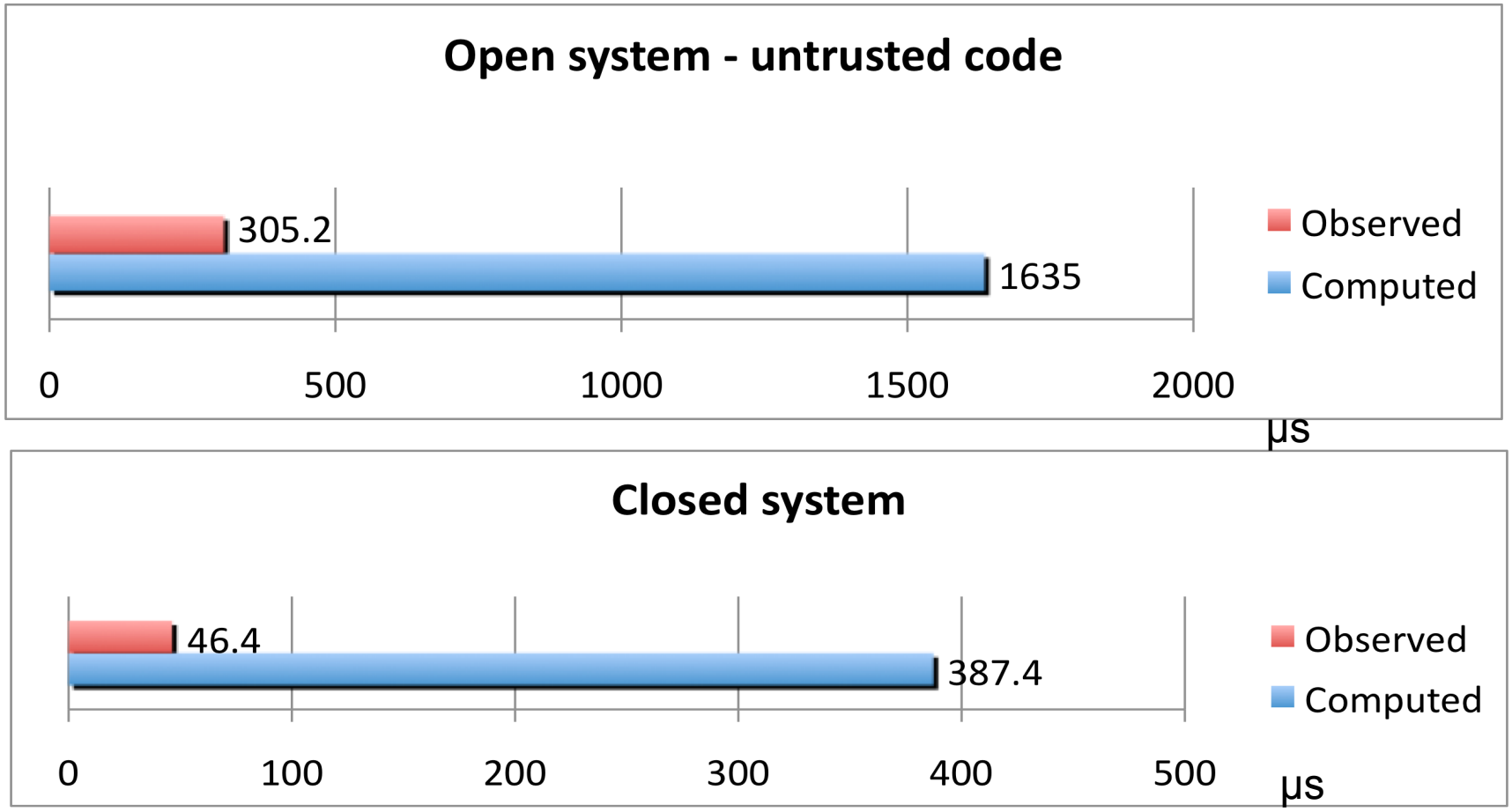
# Evaluation platform

- OMAP3-based BeagleBoard-xM
  - ARM Cortex-A8 @ 800 MHz
  - 128 MB memory
  - 32KB 4-way set-associative L1 instruction cache
    - random replacement ⇐ pessimistic model
  - Disabled L2 cache
    - Cache analysis does not (yet) scale
- Fairly accurate (but sound) model
  - dual-issue pipeline (simplified)
  - no branch prediction

Image Koen Kooi CC-SA 2.0


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Early Days…

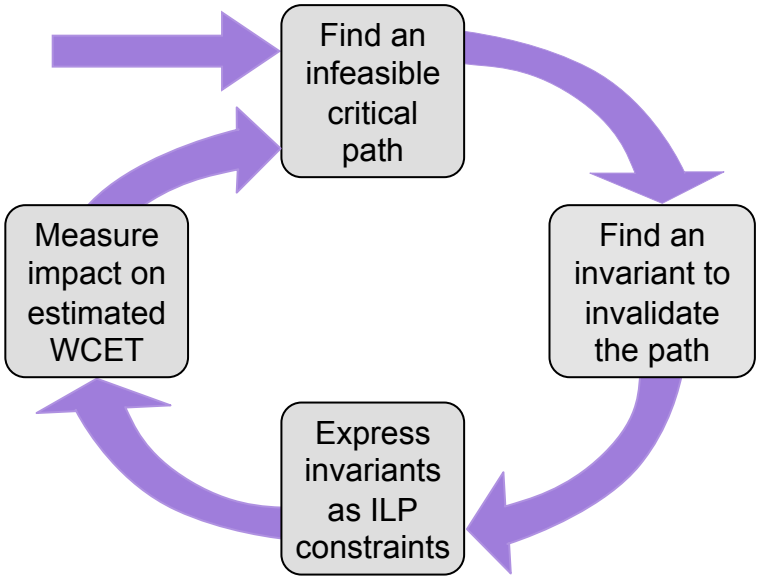




©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Improve WCET

- Analysis helps placing preemption points
  - Can optimise further
- Knowledge about seL4 can eliminate many paths
  - Invariants proved during verification
  - E.g. loop iteration counts, non-interference
  - Can easily prove new invariants
- Cache pinning
  - Big reduction in WCET
  - Eliminate cache pessimism
- Improved pipeline modelling
  - May have practical approach for complex pipelines
- Aim: IRQ WCET < 10 µs





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Phase Two: Full-System Guarantees

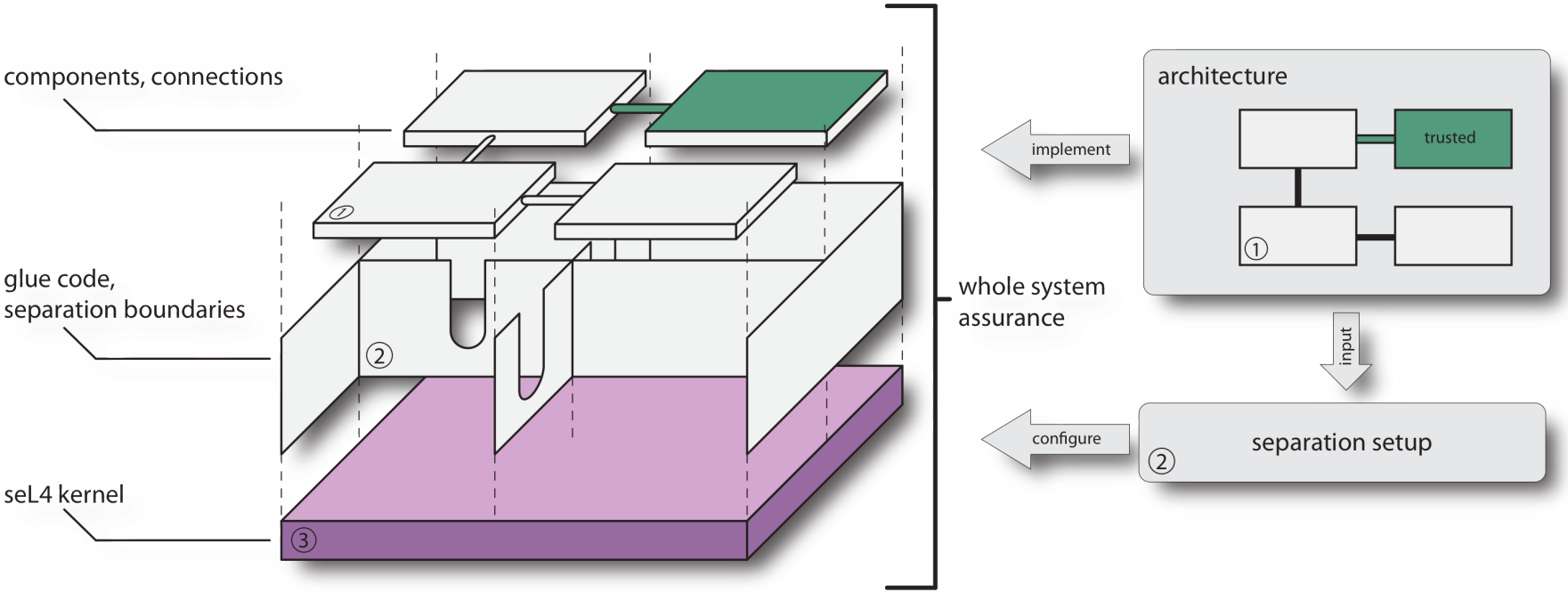
NICTA

- **Achieved:** Verification of microkernel (8,700 LOC)

- **Next step:** Guarantees for real-world systems (1,000,000 LOC)


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

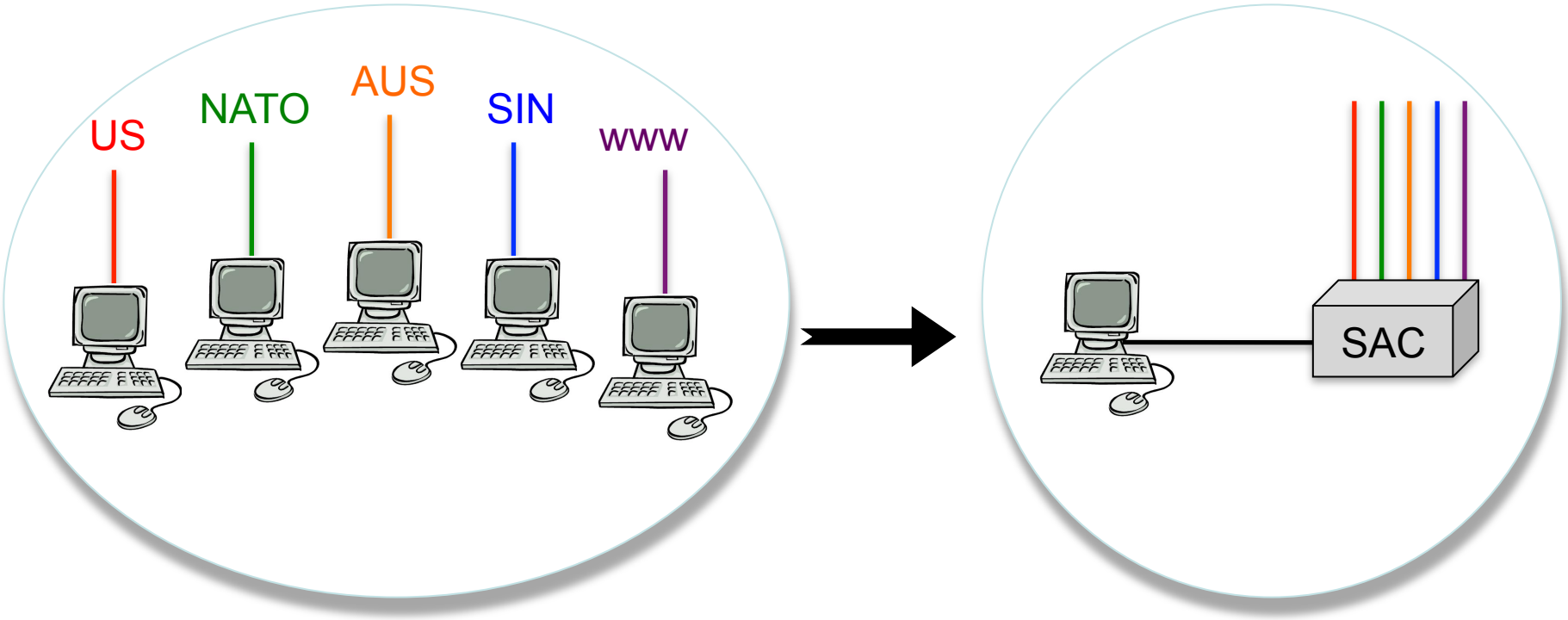
# Overview of Approach




- Build system with minimal TCB
- Formalize and prove security properties about architecture
- Prove correctness of trusted components
- Prove correctness of setup
- Prove temporal properties (isolation, WCET, …)
- Maintain performance


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

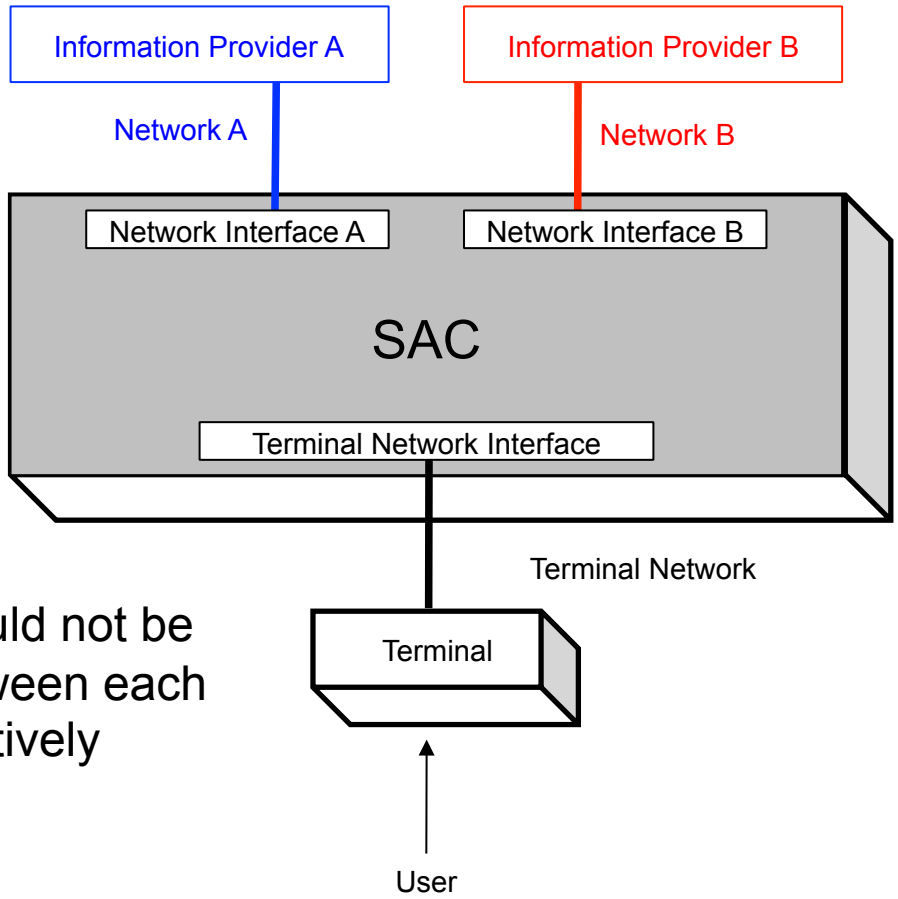
# Proof of Concept: Secure Access Controller





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

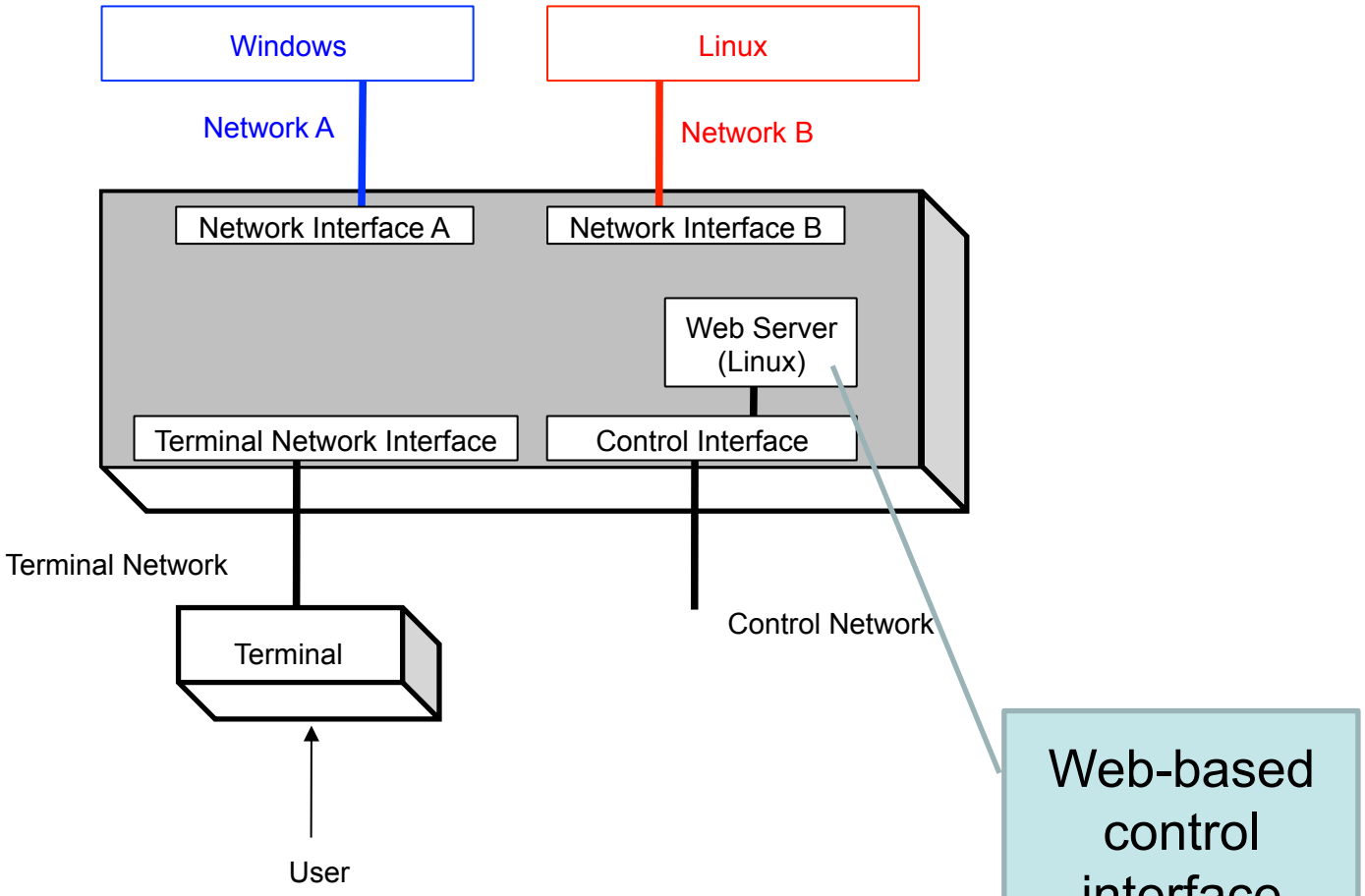
# SAC Aim




Providers A & B should not be able to leak info between each other even if they actively cooperate


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

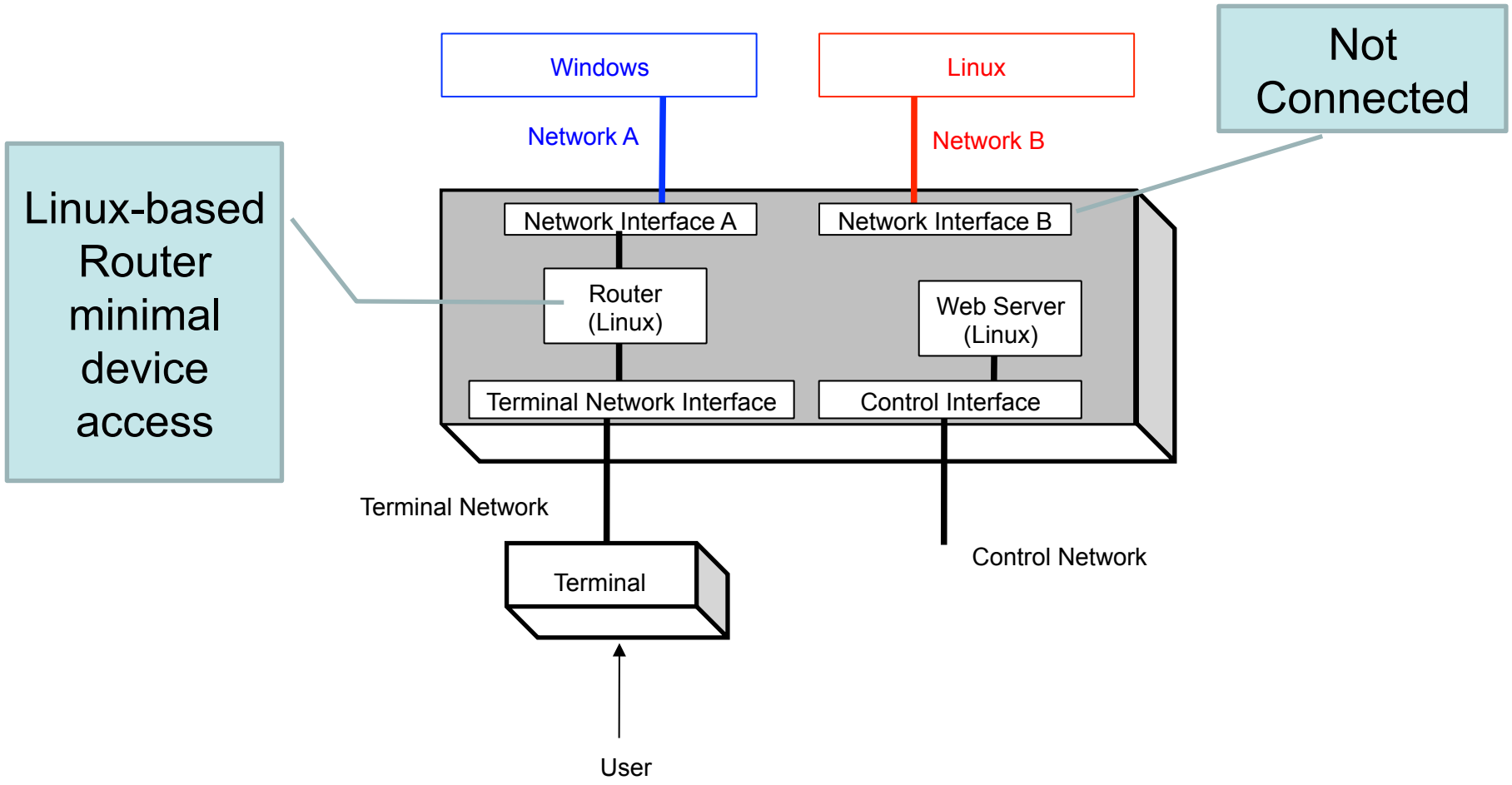
# Solution Overview





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Solution Overview





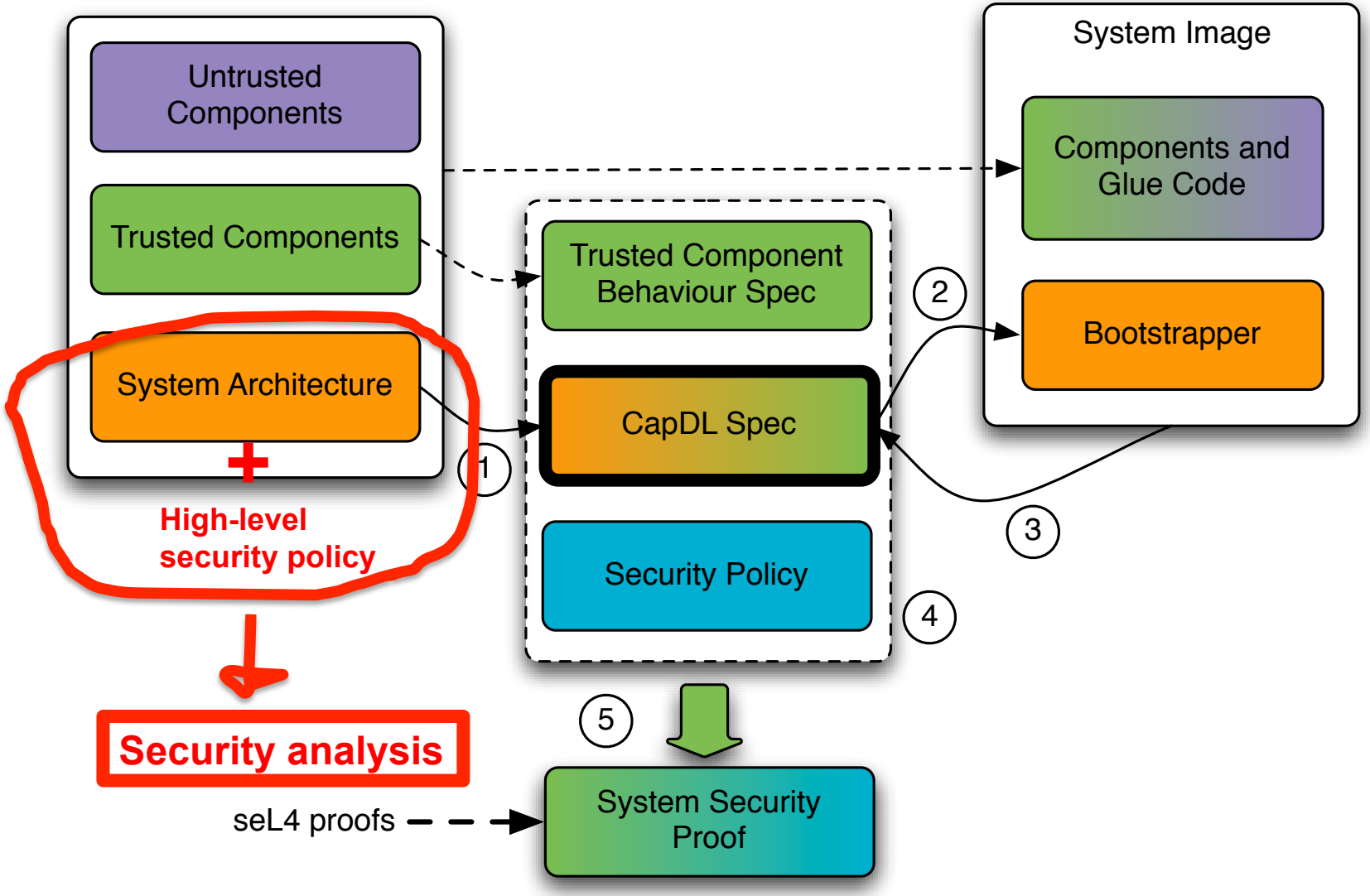
©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# SAC Significance

- Prototype of seL4-based security architecture
  - Demonstrates feasibility of seL4-based secure systems
    - incl minimal TCB
  - Demonstrates feasibility of proving relevant properties
  - Mostly hand-knitted
- Future:
  - High-level specification of architecture and properties
  - Automation of system generation
  - Automation of verifiation


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

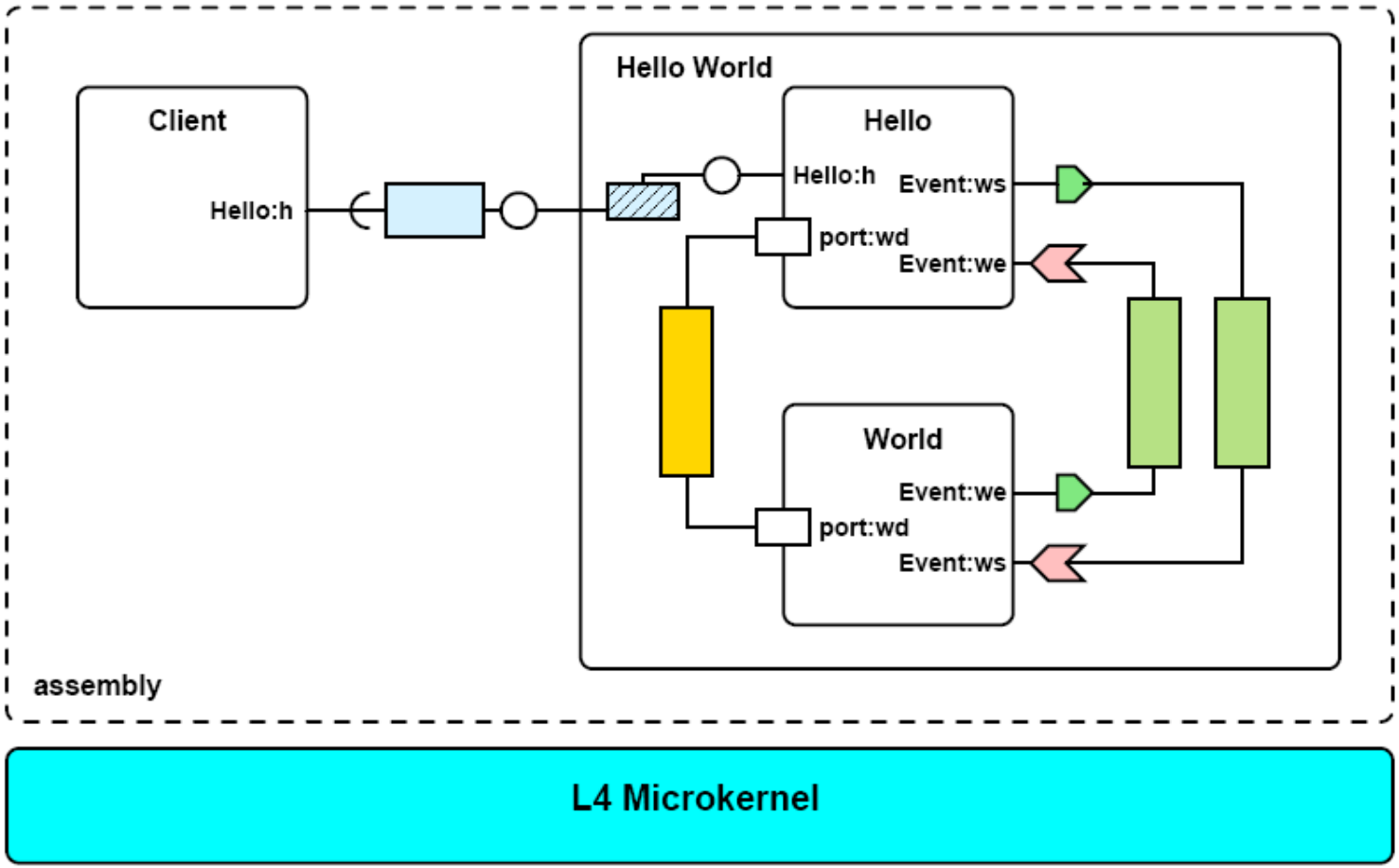
# Specifying Security Architecture





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Needed: Component Architecture (CAmkES)





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

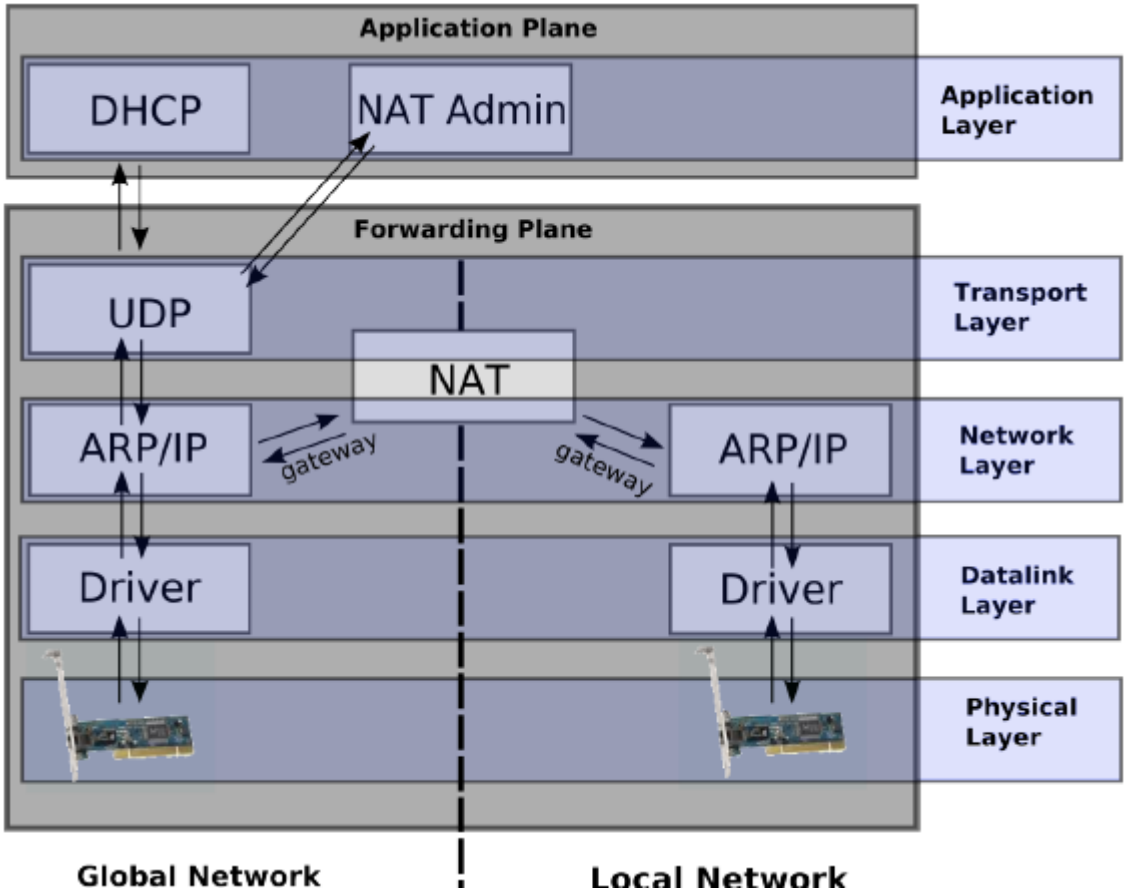
# Example System: Modular Access Router

**NAT to multiplex a single global IP address**

- Includes drivers, network stack,...

**Performance:**

- ARM11 (Freescale iMX31)
- RPC
  - 1 int arg, returning int
- CAmkES round trip:
  - 778 cycles
- raw seL4 round trip:
  - 698 cycles
    - not yet optimal!
- CAmkES overhead:
  - 80 cycles





©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Performance

- ARM11 (Freescale iMX31)
- RPC with 1 int arg, returning int
  - CAmkES round-trip: 778 cycles
  - raw seL4 round trip: 698 cycles
  - 80 cycles CAmkES overhead


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Other Bits

- Dependable device drivers
  - Event-driven drivers
  - Driver synthesis
    - exploring widening this to synthesising other system components
- Energy management
- Security
  - Information flow
  - Side channels and covert channels

**Summary:**

- Trustworthy systems are possible
  - … and we're the leaders in the field
- You can be part of it!


©2011 Gernot Heiser, UNSW/NICTA. Distributed under Creative Commons Attribution License

# Leaders – Really?

NICTA

Slashdot

Stories Recent Popular Searc

Slashdot is powered by your subm

Technology: World's Firs

Posted by Soulskill on Thursday Aug
from the wait-for-it dept.

An anonymous reader writes

> "Operating systems usually have
> and so forth are known by almos
> to prove that a particular OS ker
> formally verified, and as such it
> researchers used an executable
> the Isabelle theorem prover to ge
> matches the executable and the

Does it run Linux? "We're pleased to

New Scientist
Saturday 29/8/2009
Page: 21
Section: General News
Region: National
Type: Magazines Science / Technology
Size: 196.31 sq.cms.
Published: -----S-

# The ultimate way to keep your computer safe from harm

**FLAWS in the code, or "kernel", that sits at the heart of modern computers leave them prone to occasional malfunction and vulnerable to attack by worms and viruses. So the development of a secure general-purpose microkernel could pave the**

**just mathematics, and you can reason about them mathematically," says Klein.**

**His team formulated a model with more than 200,000 logical steps which allowed them to prove that the program would always behave as its**

kernelét alkotó 7 500 sornyi C forráskód helyességét igazolni egyedülálló teljesítmé
eredményeképpen pedig egy olyan megbízhatóságot kapnak a szoftvertől, amely e



UNSW THE UNIVERSITY OF NEW SOUTH WALES