NICTA

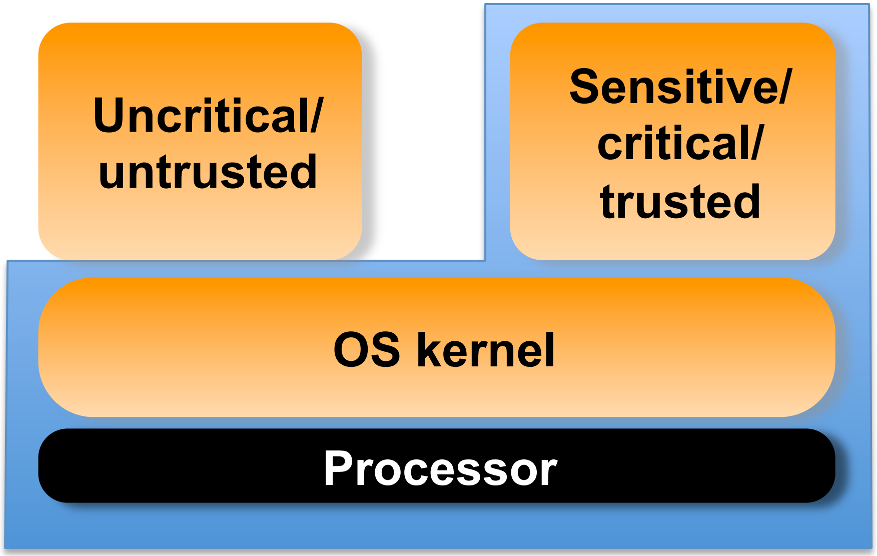
Leave nothing to chance: Building high-assurance software systems

Bernard Blackham
(on behalf of) Software Systems Research Group, NICTA

Present Systems are *NOT* Trustworthy!

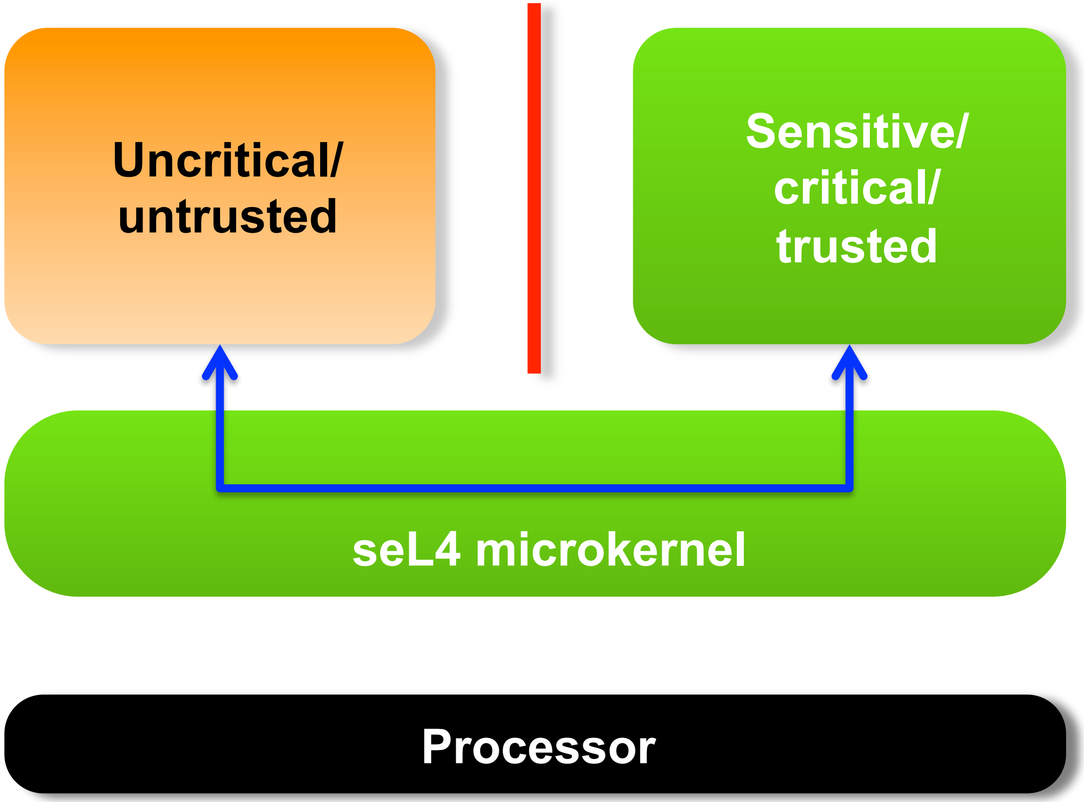
Trusted computing base is the weakest link

- The **trusted computing base** (TCB) of a system, is the set of all things which can, if at fault, potentially undermine the integrity/safety/security of a system



- Everything in the TCB must be trusted. But is it ***trustworthy***?
  - Make the TCB as small as possible – reduce risk
  - Just make sure there are no bugs…

Decomposition + Isolation = Sanity



What is seL4?

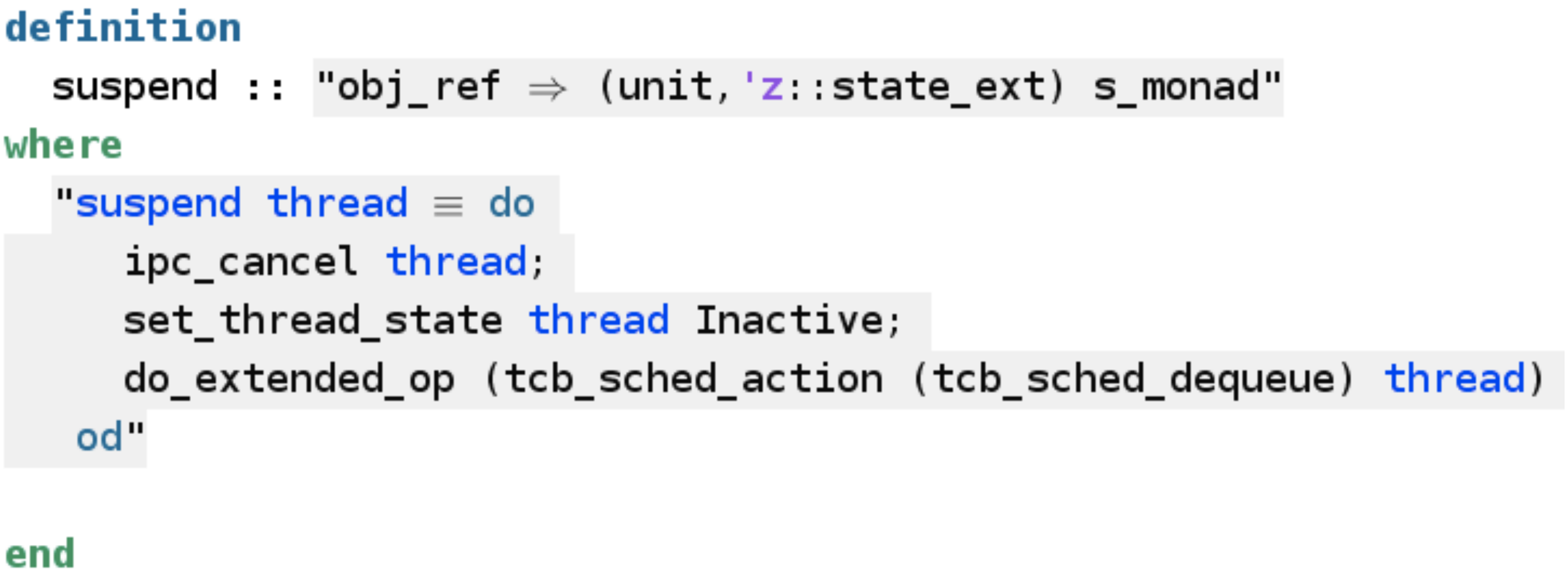
- seL4 is a high-performance general-purpose microkernel
  - Formal proof of correctness down to binary level
  - Developed for ARM and x86
  - 10k lines of code
  - 200,000 lines of proof
  - 0 bugs*

* Conditions apply

- Capabilities used for access control and privilege management
- Policy decisions live outside the kernel
- OS “personalities” built on top of seL4 API
- Timing guarantees provided by static analysis

Verified What?

- Every operation in seL4 is defined abstractly, e.g.:

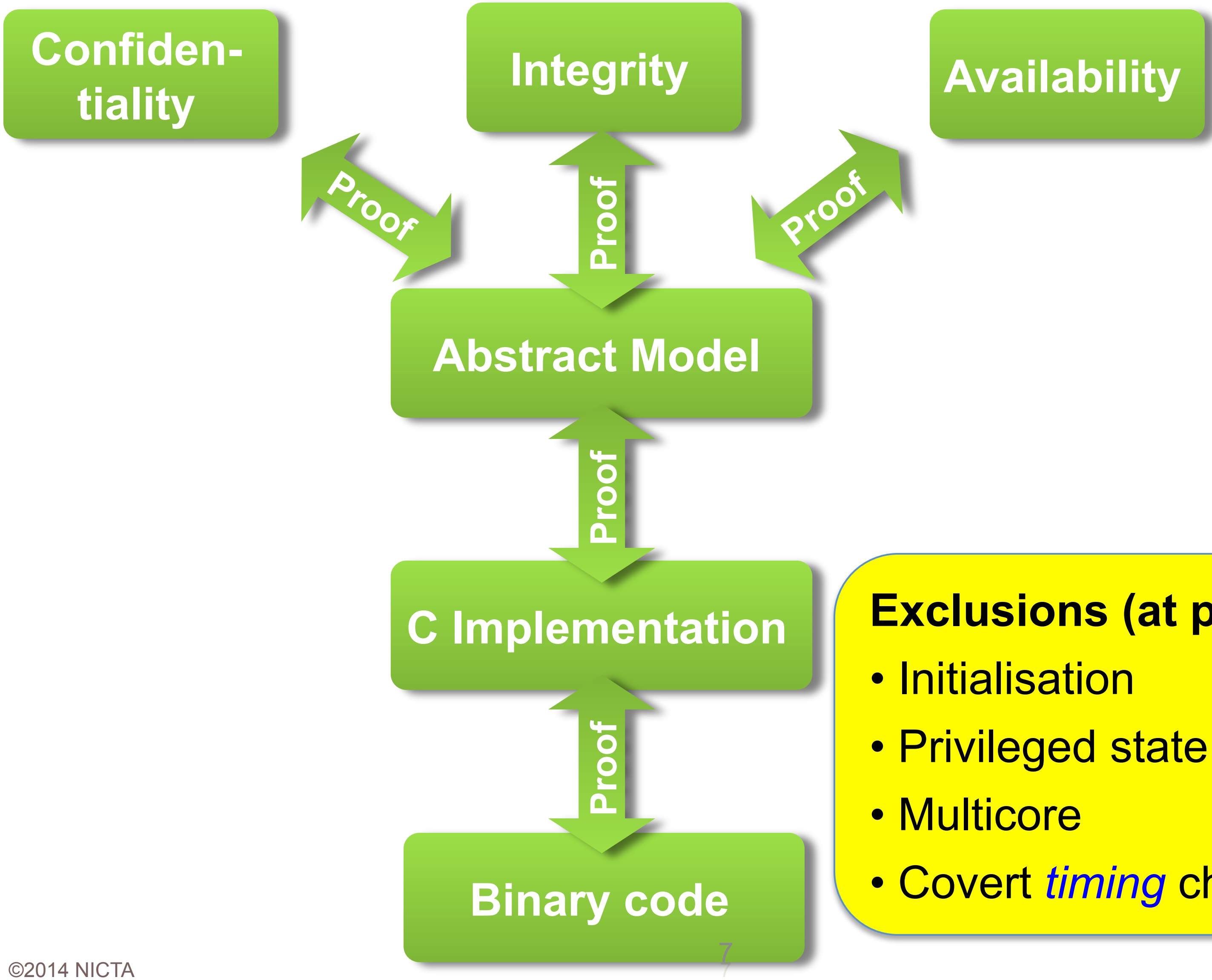


```
definition
  suspend :: "obj_ref ⇒ (unit,'z::state_ext) s_monad"
where
  "suspend thread ≡ do
     ipc_cancel thread;
     set_thread_state thread Inactive;
     do_extended_op (tcb_sched_action (tcb_sched_dequeue) thread)
   od"

end
```

- Refinement proof guarantees that the corresponding C code
  - is a correct implementation of the specification
  - will terminate
  - will not crash
  - will not access invalid memory

Mathematical *Proof* of Isolation



Exclusions (at present):

- Initialisation
- Privileged state & caches
- Multicore
- Covert *timing* channels

SMACCM: High-Assurance UAV

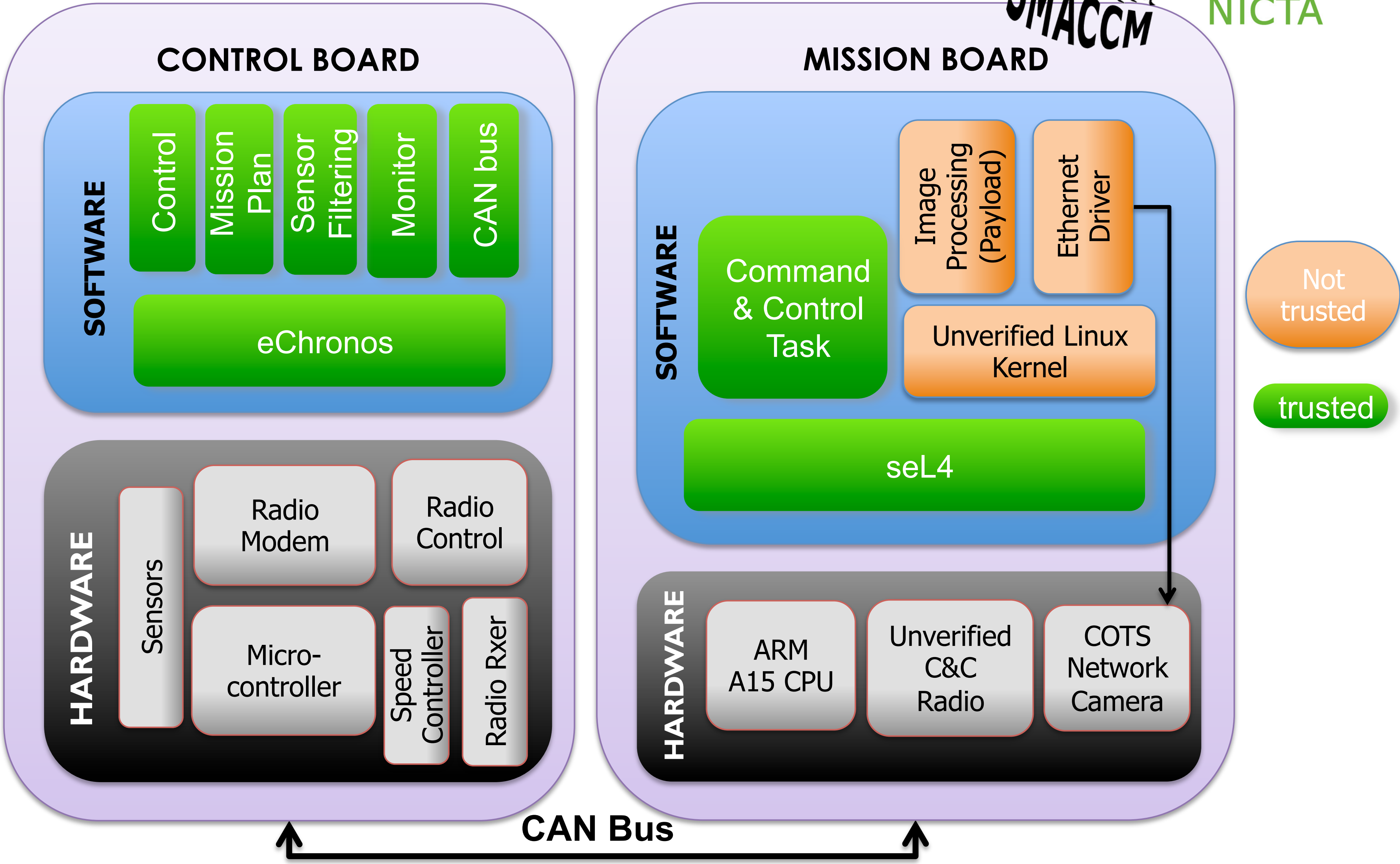
DARPA HACMS Program:

- Provable vehicle safety
- “Red Team” must not be able to divert vehicle

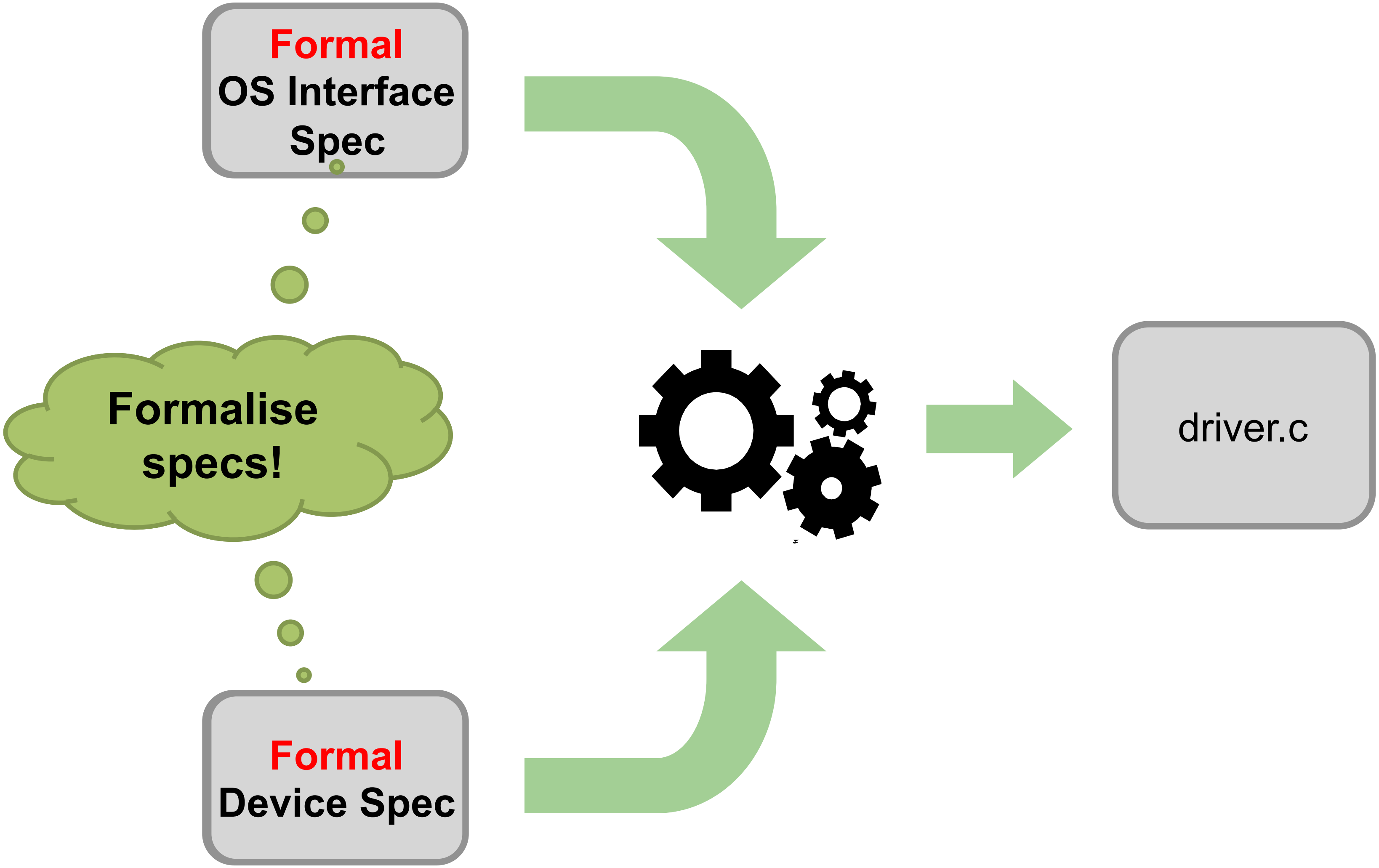
SMACCMcopter Research Vehicle

Unmanned Little Bird Deployment Vehicle

SMACCMcopter Architecture



Synthesis: Device Drivers



Formal OS Interface Spec

Formalise specs!

Formal Device Spec

driver.c

Actually works! (On Linux & seL4)

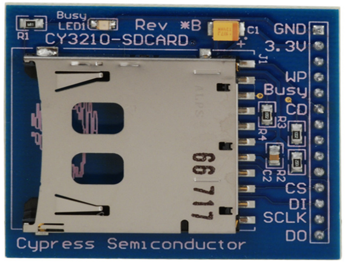
IDE disk controller

W5100 Eth shield

Intel PRO/1000 Ethernet

In progress:

- Extract device spec from device design work-flow
- Verified synthesis

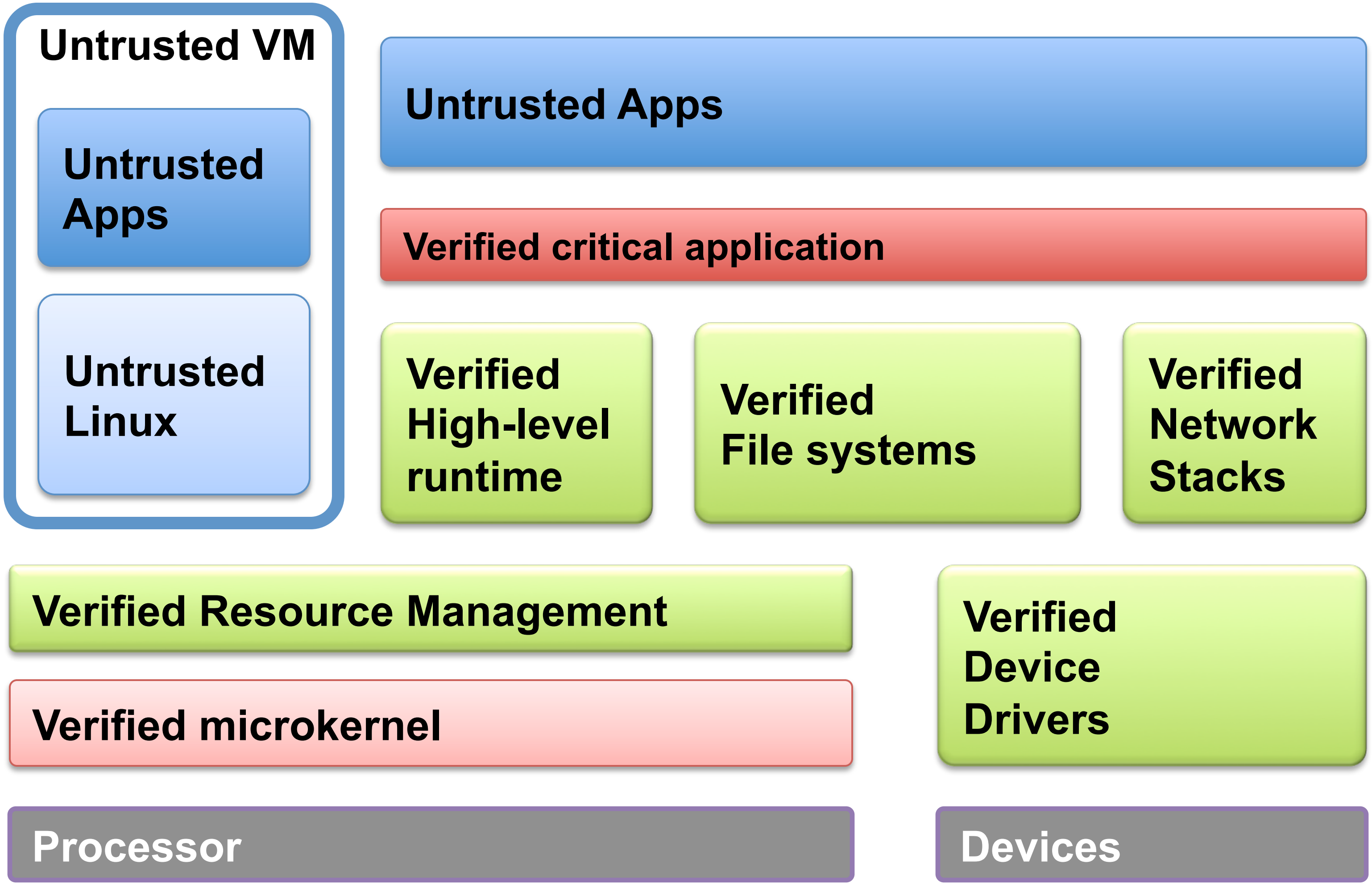


SD host controller

UART controller

Asix AX88772 USB-to-Eth adapter

Future: Full-Scale Trustworthy System



Lessons Learnt So Far

Formal methods are cost-effective

- Cost-effective for high assurance on small to moderate scale
- $200-400/LOC for 10kLOC

We think we can scale bigger and cheaper:

- Componentisation
  - verify components in isolation – enabled by seL4 guarantees
- Synthesis; code and proof co-generation
  - Abstraction: Domain-specific languages, and higher-level languages increase productivity

Check it out

seL4 is open source, on github! – see http://sel4.systems/

- C code
- Abstract model
- Proofs

http://trustworthy.systems/