Dependable Systems made by FIRST
BOSS: Real time Operating System in Space

FIRST

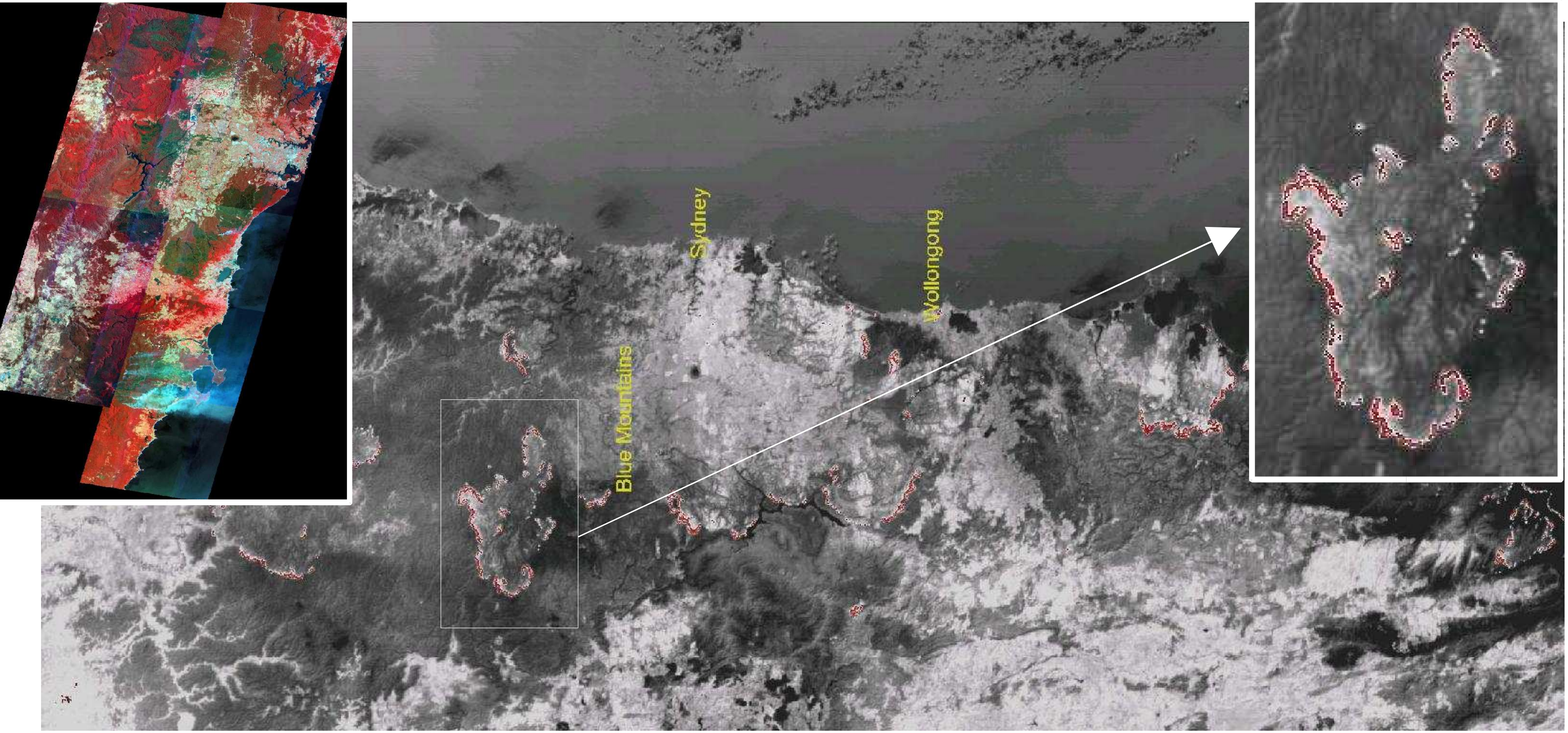
Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Sergio Montenegro
FhG FIRST
www.first.fhg.de/~sergio
sergio@first.fhg.de

FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

BIRD Aufnahmen

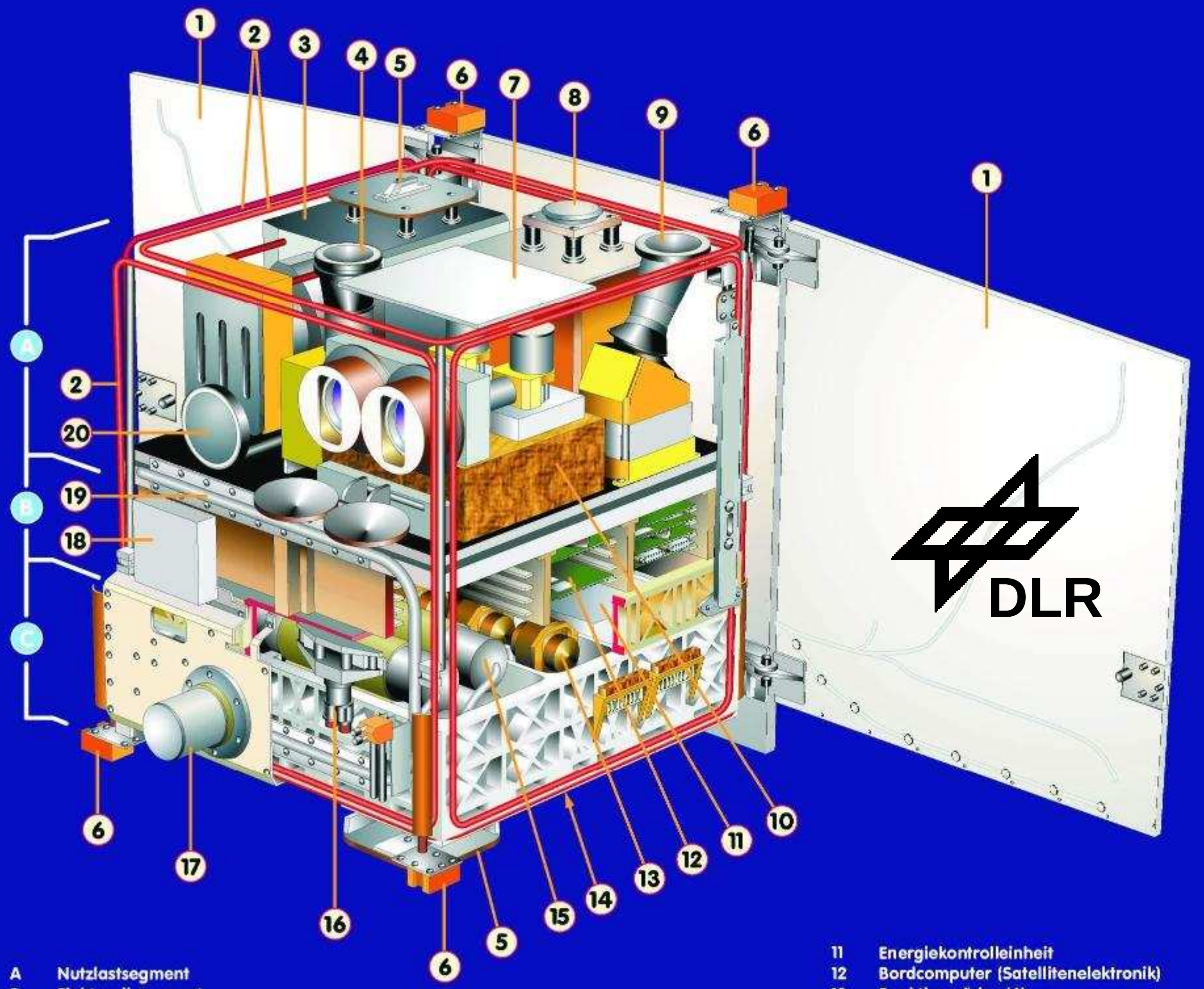


DLR

FIRST

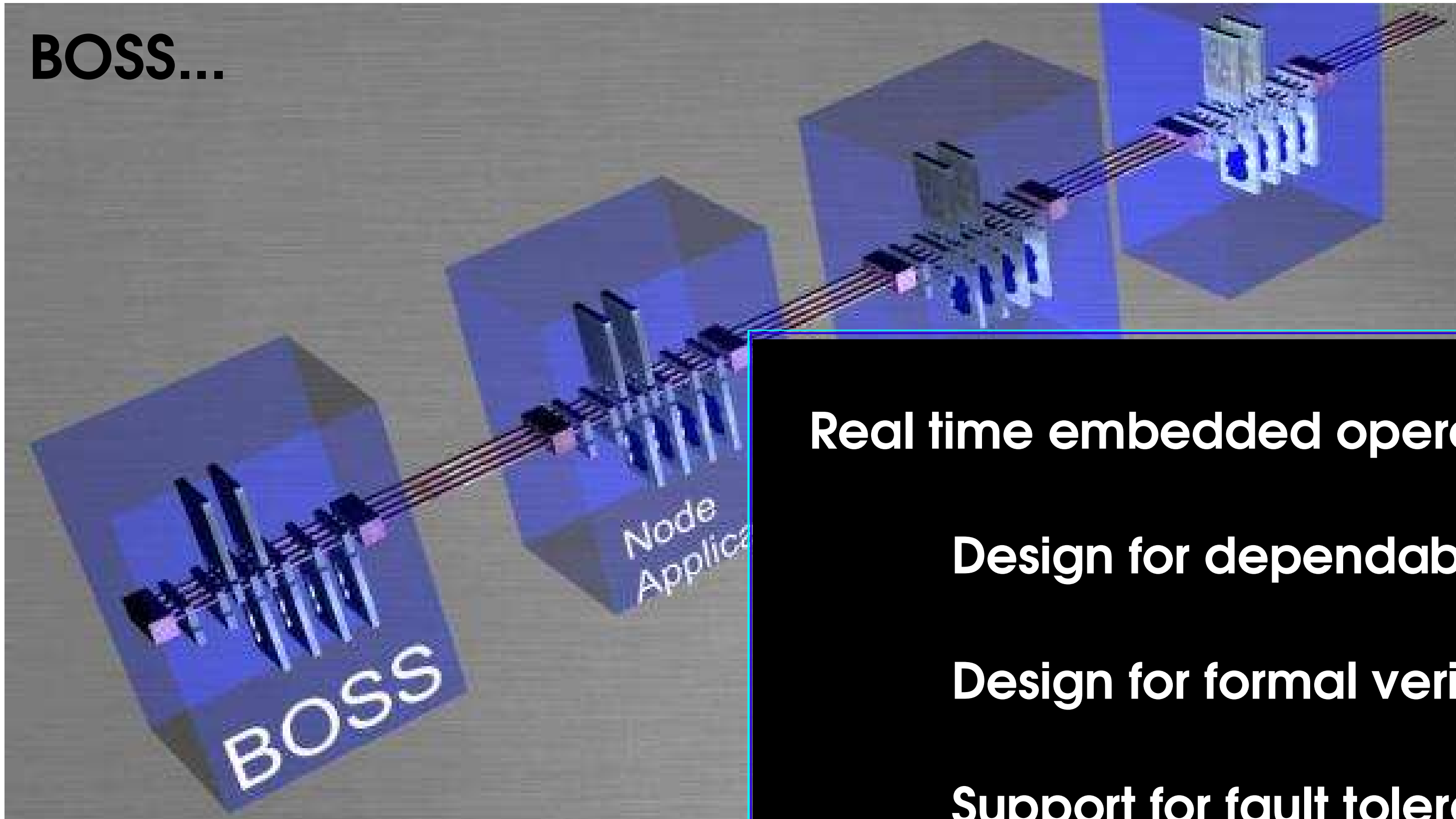
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

BIRD-Satellit (BIRD : Bi-spectral InfraRed Detection)



A	Nutzlastsegment
B	Elektroniksegment
C	Dienstsegment
1	Solarzellenfläche
2	Magnetspulen (6)
3	Weitwinkel-Stereokamera WAOSS-B
4	Sternsensor-1
5	S-Band-Halbrundstrahlantenne
6	Sonnensensor
7	Infrarotsystemradiator
8	GPS-Antenne
9	Sternsensor-2
10	Zweikanal-Infrarotsensorsystem
11	Energiekontrolleinheit
12	Bordcomputer (Satellitenelektronik)
13	Reaktionsräder (4)
14	Satellitenradiator
15	2 x 4 NiH_2 -Zellen (12 Ah)
16	Solarzellenaufklappzünder
17	S-Band-Richtstrahlantenne
18	S-Band-Elektronik (Sender)
19	Wärmerohr
20	Matrixkamera

BOSS...



Real time embedded operating system

Design for dependability

Design for formal verification

Support for fault tolerance

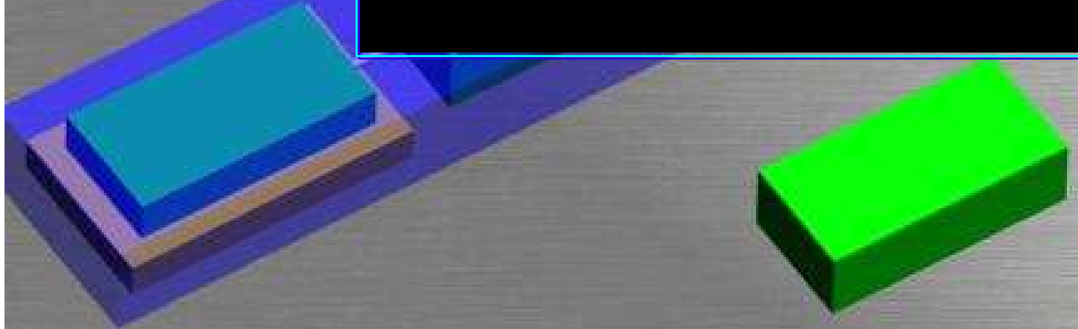
Fast, small,

.... and .... Open Source!

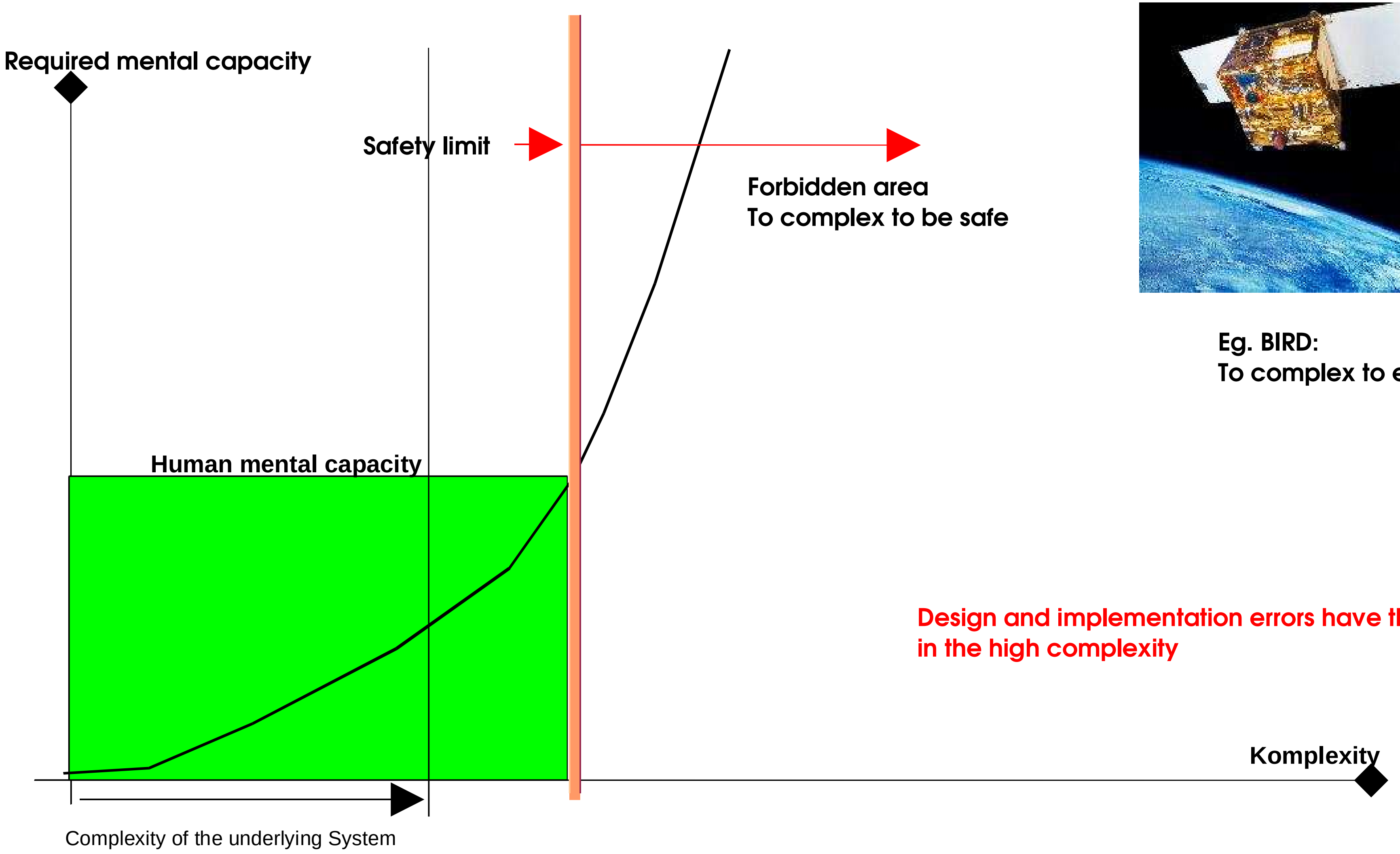
BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. component technology to handle complexity (not to create complexity)

-> + Formal verification



Complexity destroys safety



Eg. BIRD:
To complex to exist?

Design and implementation errors have their roots in the high complexity

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Simple ->Formal Verification

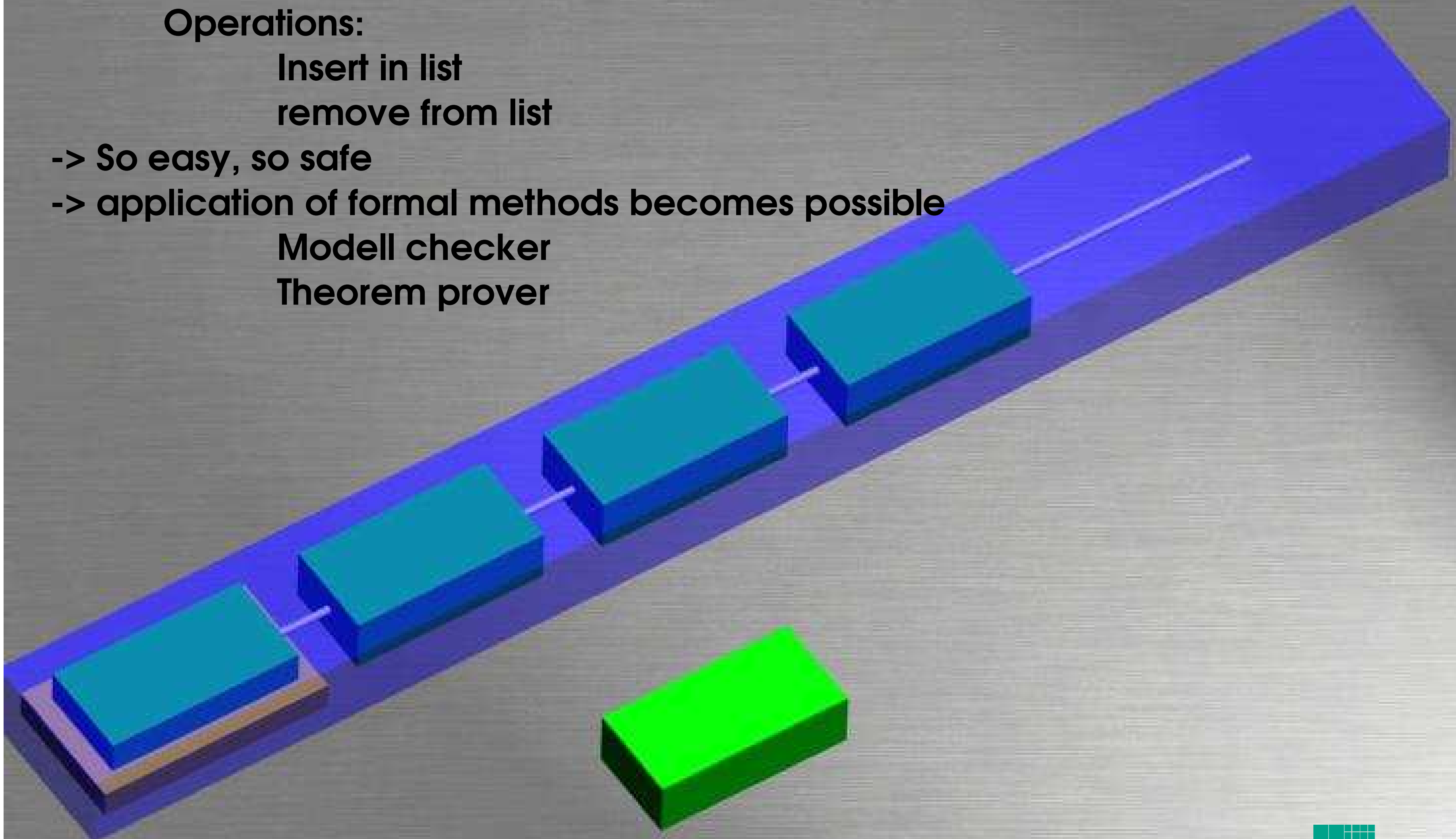
BOSS basic functions (for every thing): lists

- Operations:
  - Insert in list
  - remove from list

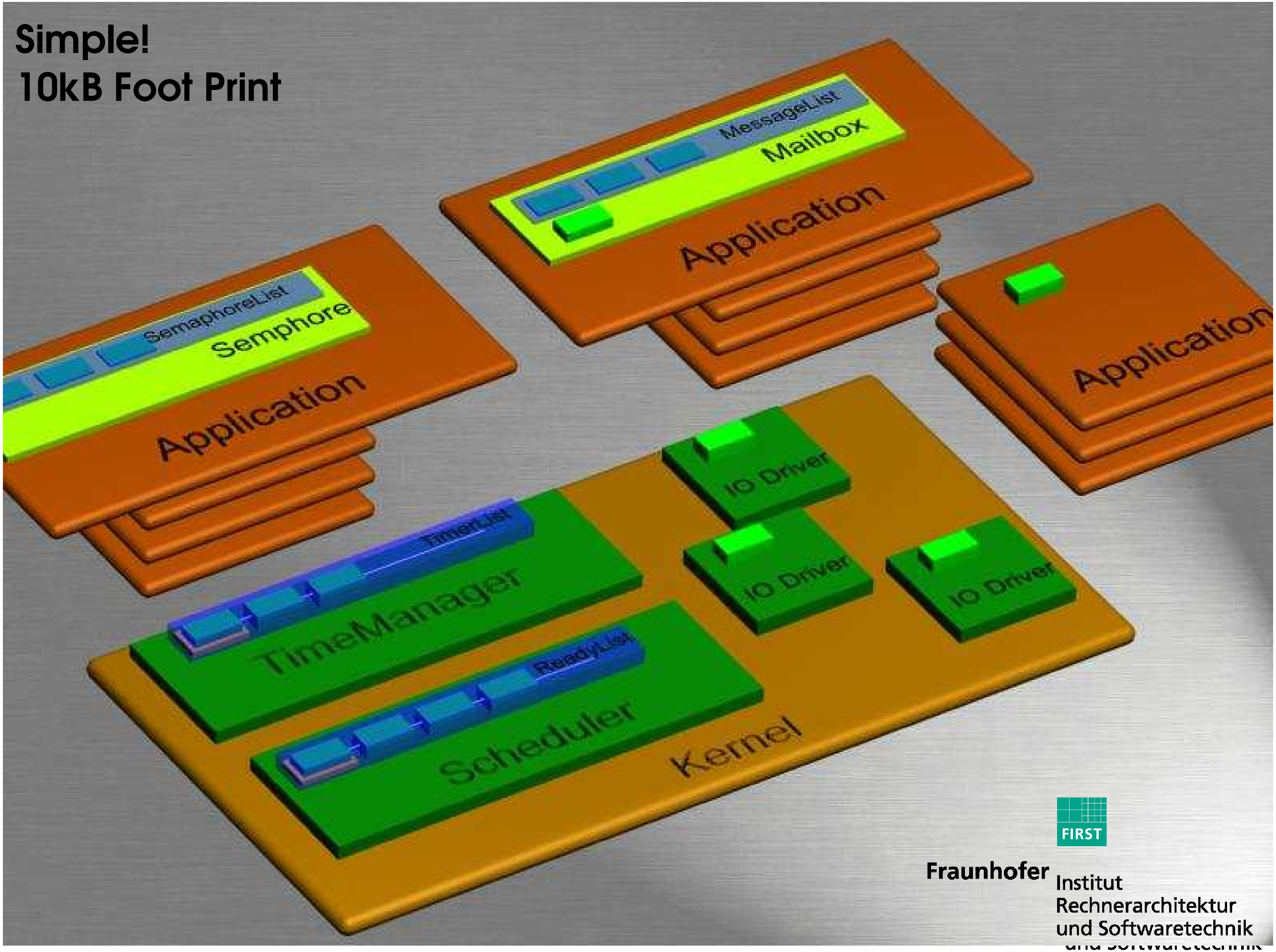
-> So easy, so safe

-> application of formal methods becomes possible

- Modell checker
- Theorem prover



Simple!
10kB Foot Print



FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

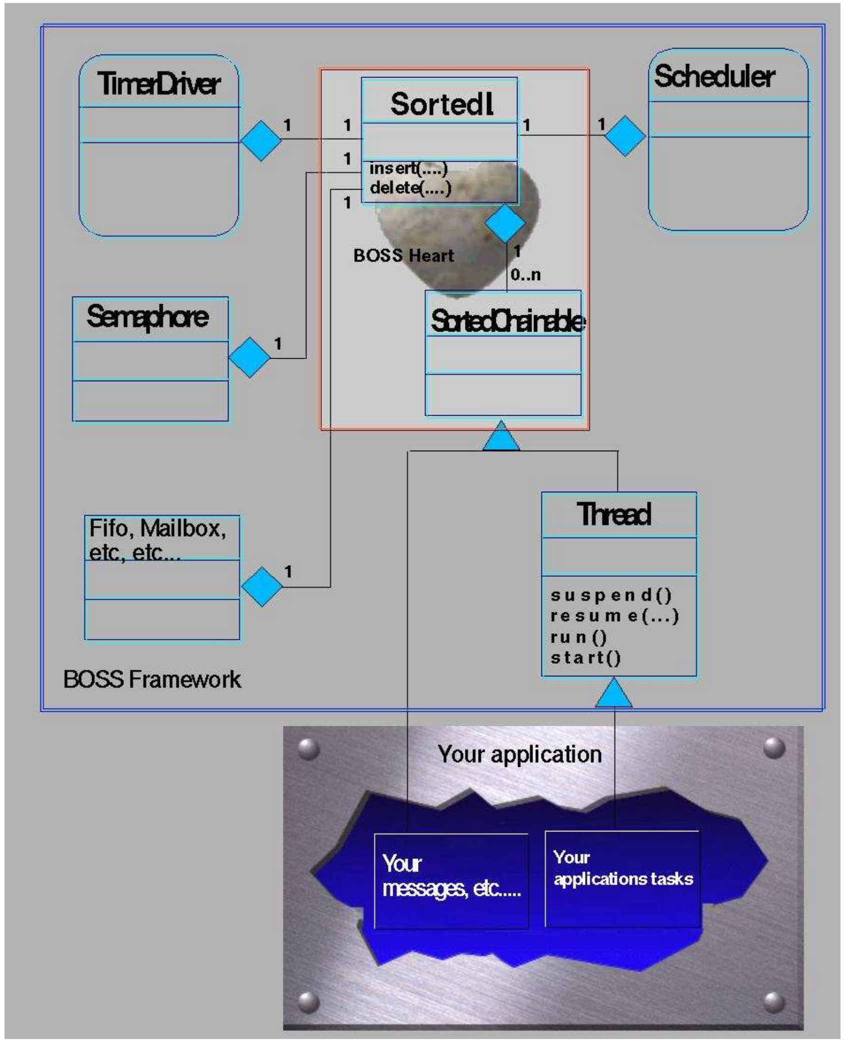
und Softwaretechnik

BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. component technology to handle complexity
   (not to create complexity)

-> + Formal verification

BOSS Framework



```
External Thread xx;
class TestThread: public Thread {                // active object
        void run () {
                while(1) {
                                {.... do something }
                                yield();
                                {.... do something }
                                suspend();
                                {.... do something }
                                suspendFor(1000);
                                resume(xx);
                }
        }
};
/** Another example: **/
Semaphore monitor;
class OtherTestThread : public Thread {
                void run () {
                                TimeControl timeControl;            //To implement time loops
                                timeControl.startAt(5000); // Time point for the first time
                                timeControl.every(100);             // Cyclus time
                                while(1) {
                                                timeControl.wait();
                                                monitor.enter();        // protected area,
                                                {.... do something }
                                                monitor.leave();
                                }
                }
};
/** Create 6 threads or applications ***/
TestThread              a, b, xx;
OtherTestThread  x, y, z;
```

OS Framework:

modern software technology / engineering

Design for real time safety critical applications

cost effective

Fraunhofer FIRST
Institut
Rechnerarchitektur
und Softwaretechnik

BOSS... designed for dependability

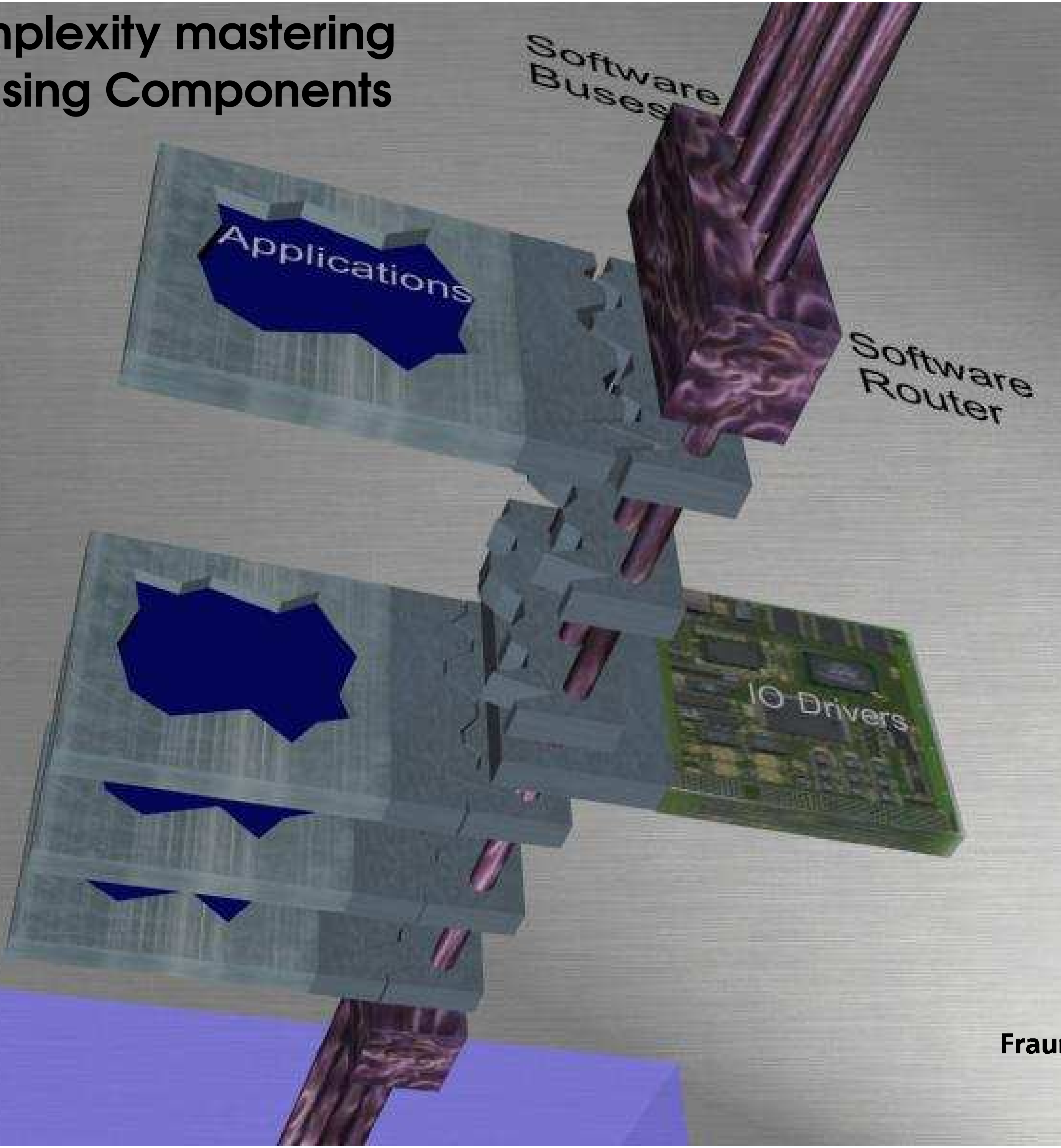
1. Irreducible complexity
2. Framework technology to reduce complexity
3. component technology to handle complexity
   (not to create complexity)

-> + Formal verification

FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Complexity mastering by using Components

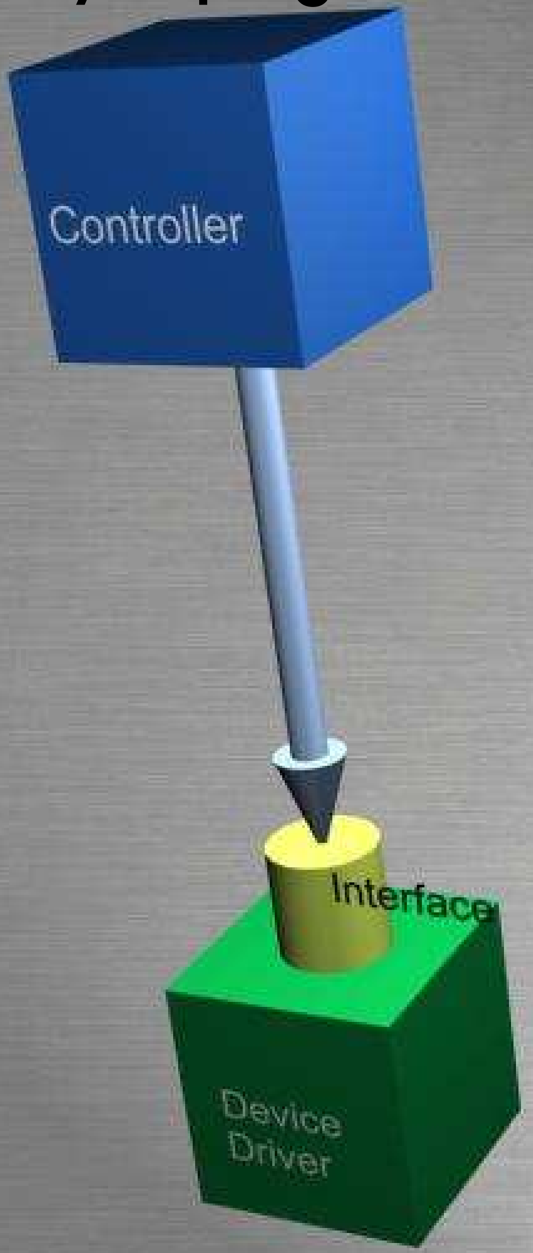


Build the System by plugging applications as components

Communication by using Software buses and routers

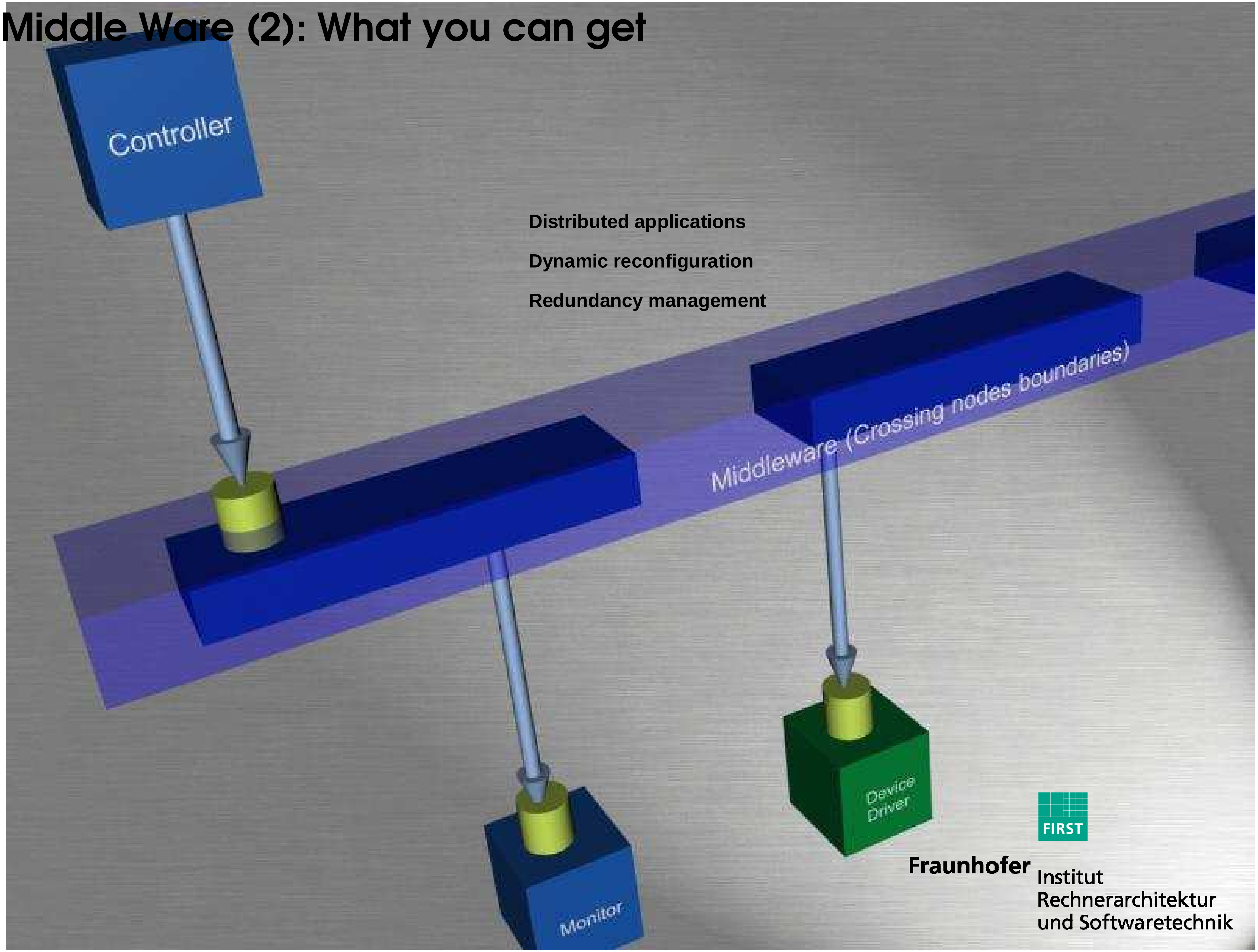
Fraunhofer FIRST Institut
Rechnerarchitektur
und Softwaretechnik

Middle Ware (1): What you program



Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

Middle Ware (2): What you can get



Distributed applications

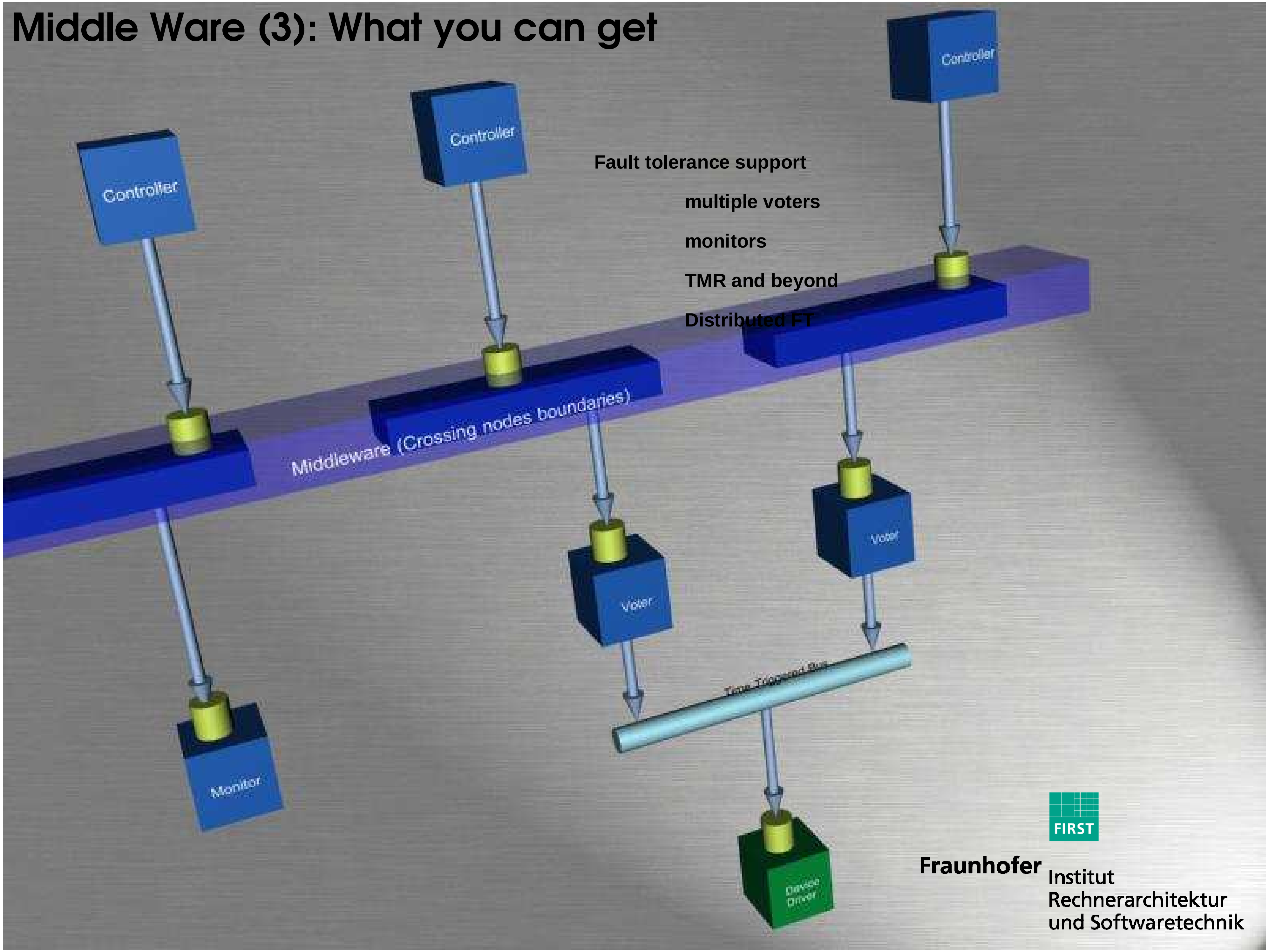
Dynamic reconfiguration

Redundancy management

FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

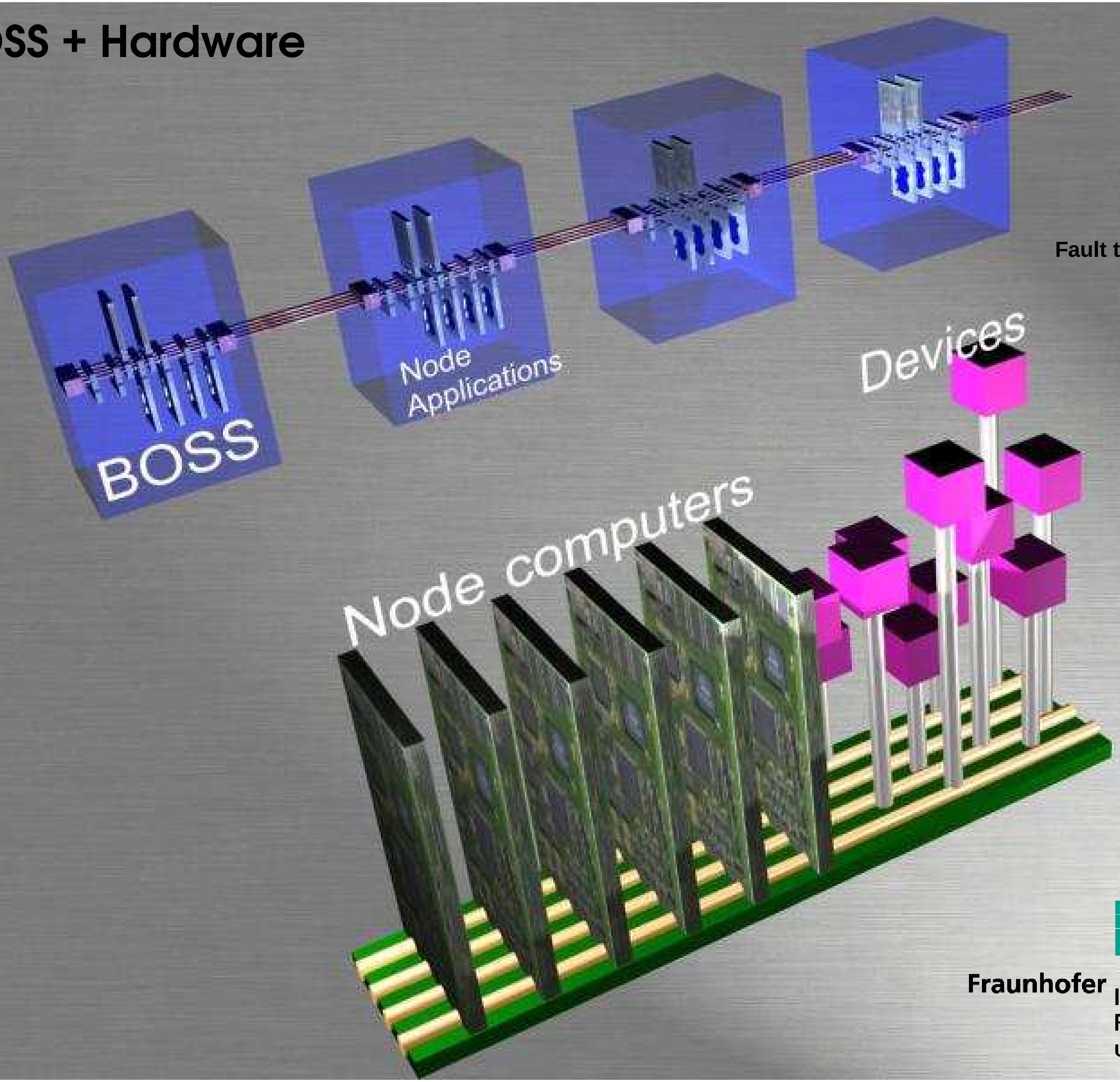
Middle Ware (3): What you can get



Fault tolerance support

- multiple voters
- monitors
- TMR and beyond
- Distributed FT

BOSS + Hardware



Fault tolerance support

multiple voters

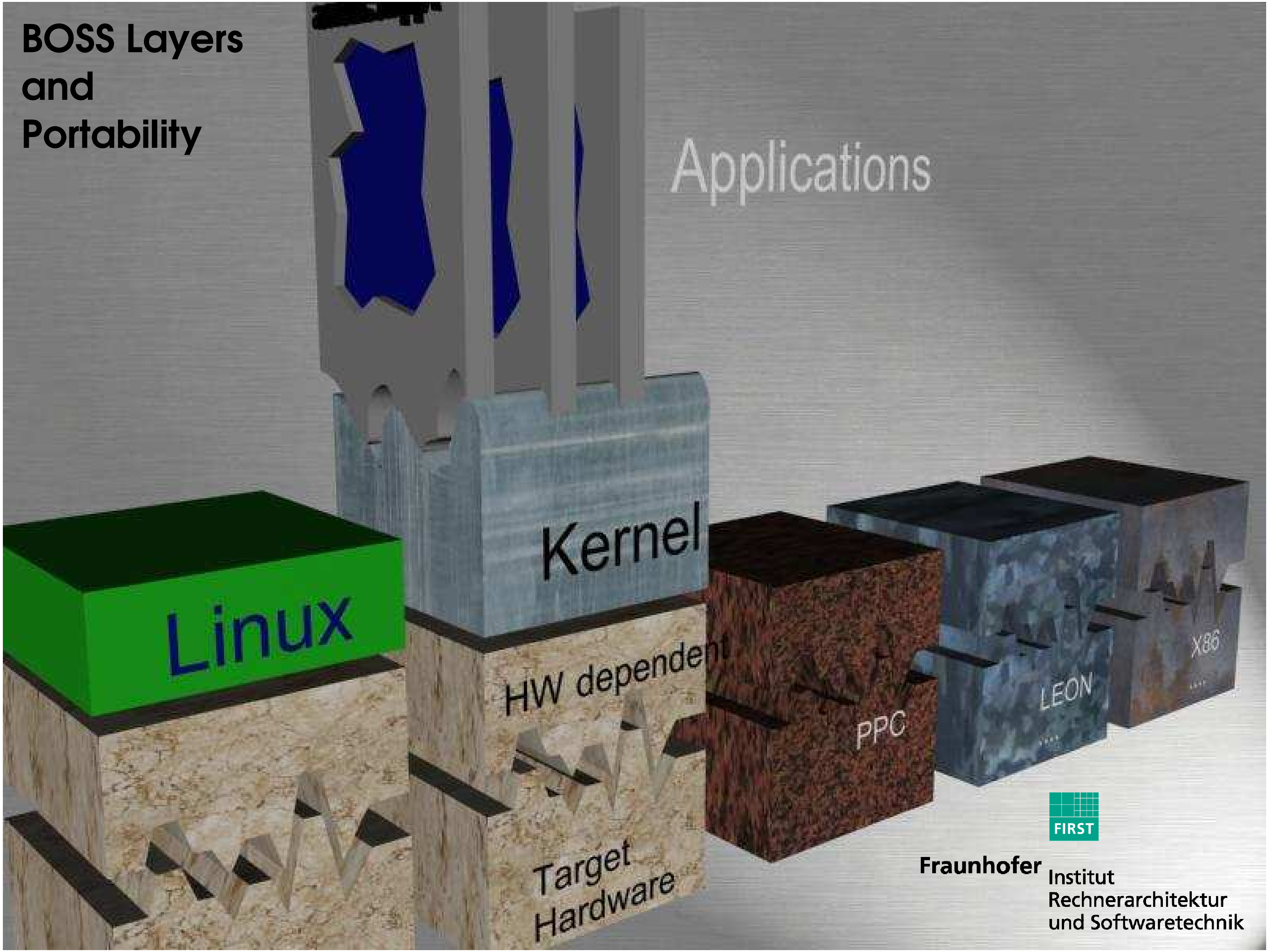
monitors

TMR and beyond

Distributed FT

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

BOSS Layers and Portability



FIRST

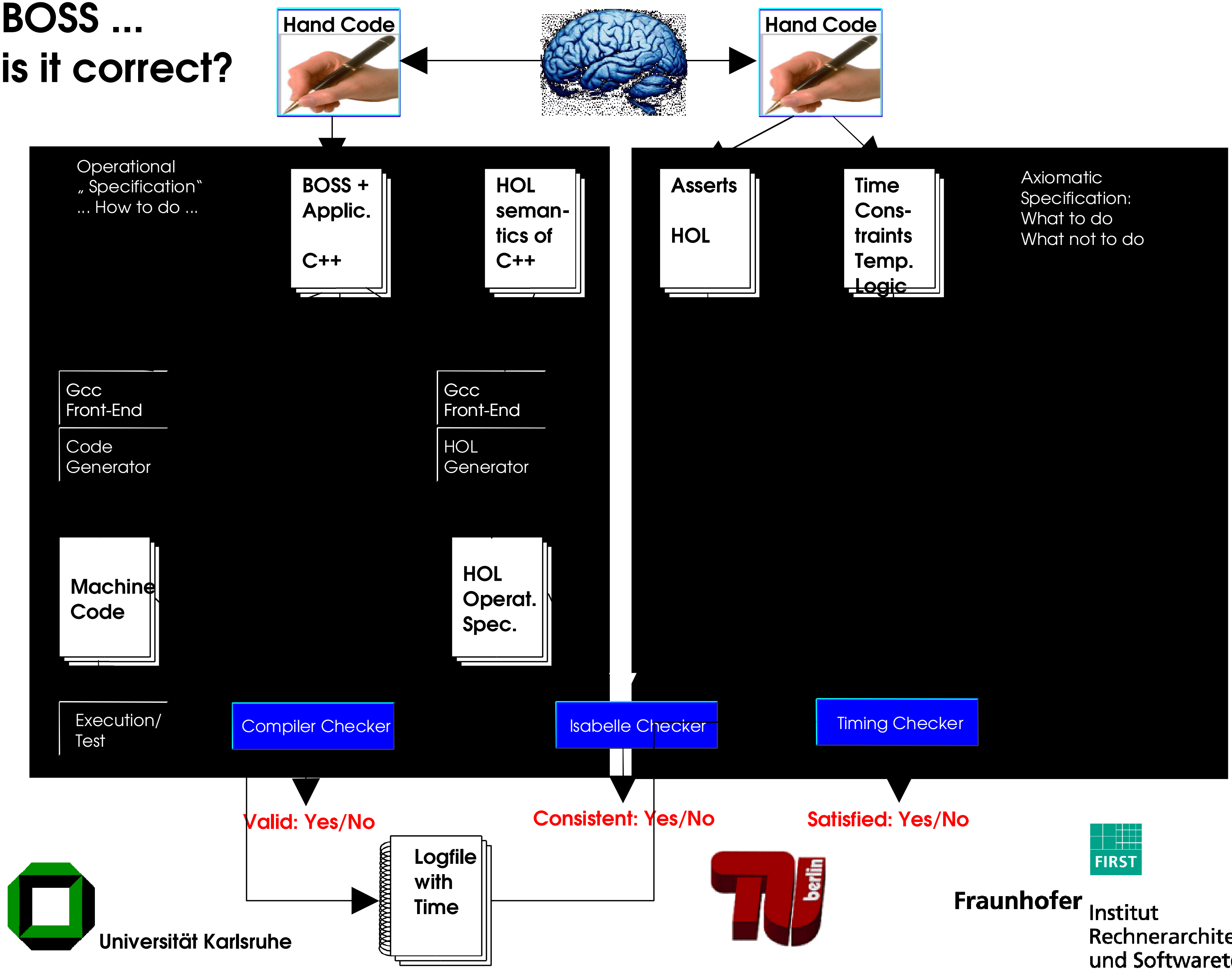
Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. component technology to handle complexity
   (not to create complexity)

-> + Formal verification

BOSS ...
is it correct?



Hand Code

Hand Code

Operational
„ Specification"
... How to do ...

BOSS +
Applic.
C++

HOL
seman-
tics of
C++

Gcc
Front-End

Code
Generator

Gcc
Front-End

HOL
Generator

Machine
Code

HOL
Operat.
Spec.

Execution/
Test

Compiler Checker

Isabelle Checker

Asserts
HOL

Time
Cons-
traints
Temp.
Logic

Axiomatic
Specification:
What to do
What not to do

Timing Checker

Valid: Yes/No

Consistent: Yes/No

Satisfied: Yes/No

Logfile
with
Time

Universität Karlsruhe

berlin

FIRST

Fraunhofer

Institut
Rechnerarchitektur
und Softwaretechnik

Thank You

FIRST

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik