Lucent CORBA Seminar 1999

Distributed Debugging API for ORBs and Services

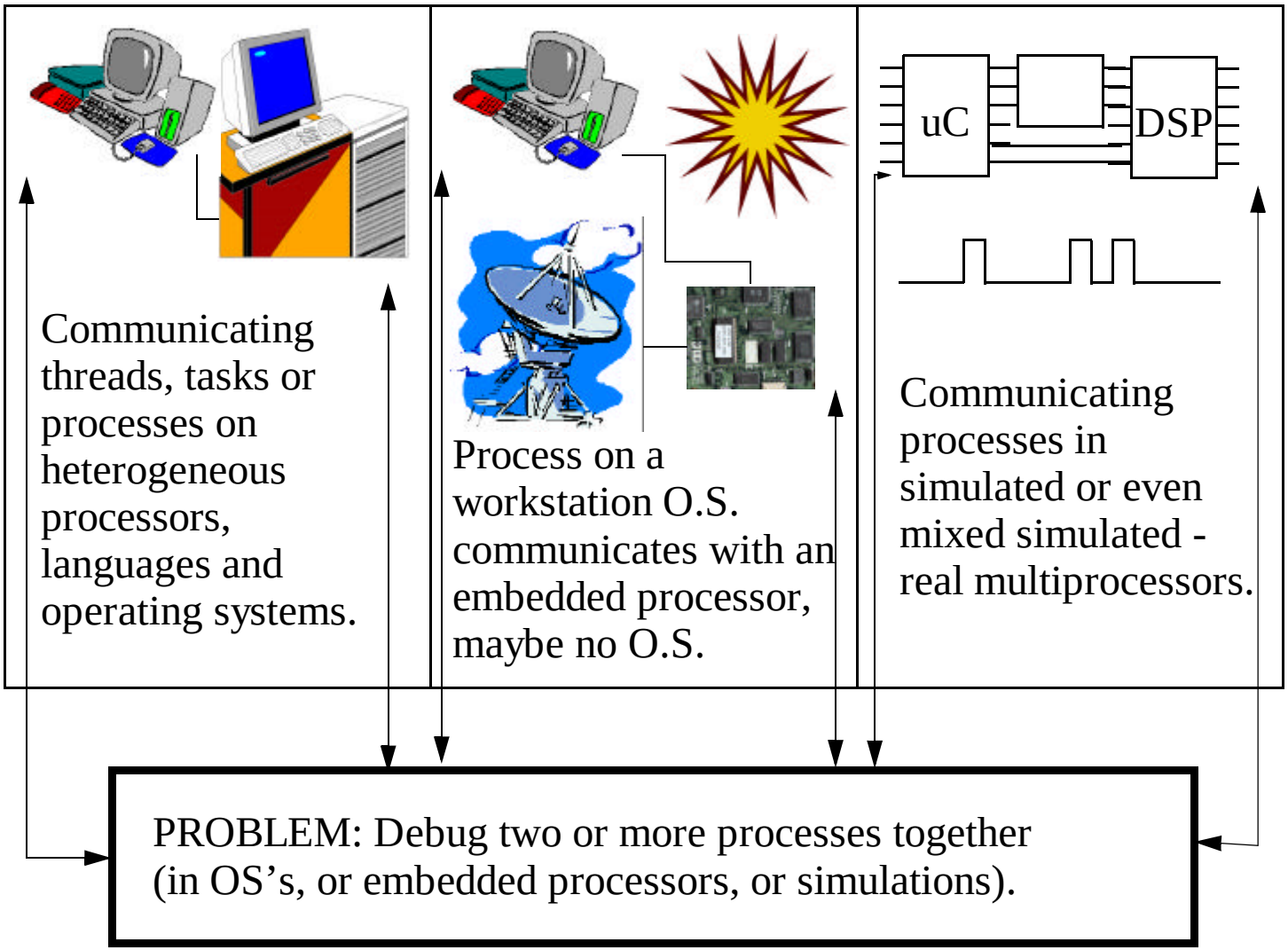
Request for Proposal, test/99-08-02

September 28, 1999

Dale Parson, Distinguished Member of Technical Staff

Bell Labs, Microelectronics Division

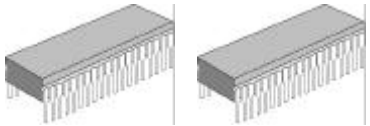
dparson@lucent.com

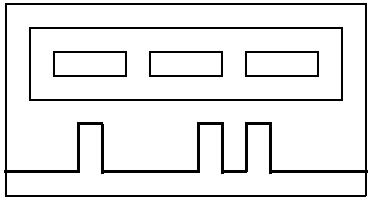

uC
DSP
Communicating threads, tasks or processes on heterogeneous processors, languages and operating systems.
Process on a workstation O.S. communicates with an embedded processor, maybe no O.S.
Communicating processes in simulated or even mixed simulated - real multiprocessors.
PROBLEM: Debug two or more processes together (in OS's, or embedded processors, or simulations).

Operating System Object contains one or more Virtual Processors as Sequential Threads

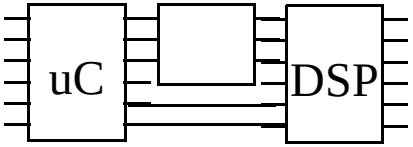
registers memory flags page table breakpoints profile	registers memory flags page table breakpoints profile
registers memory flags page table breakpoints profile	registers memory flags page table breakpoints profile

Hardware Debug Server Object runs on Interface Card Processor to provide access to Embedded Application Processors

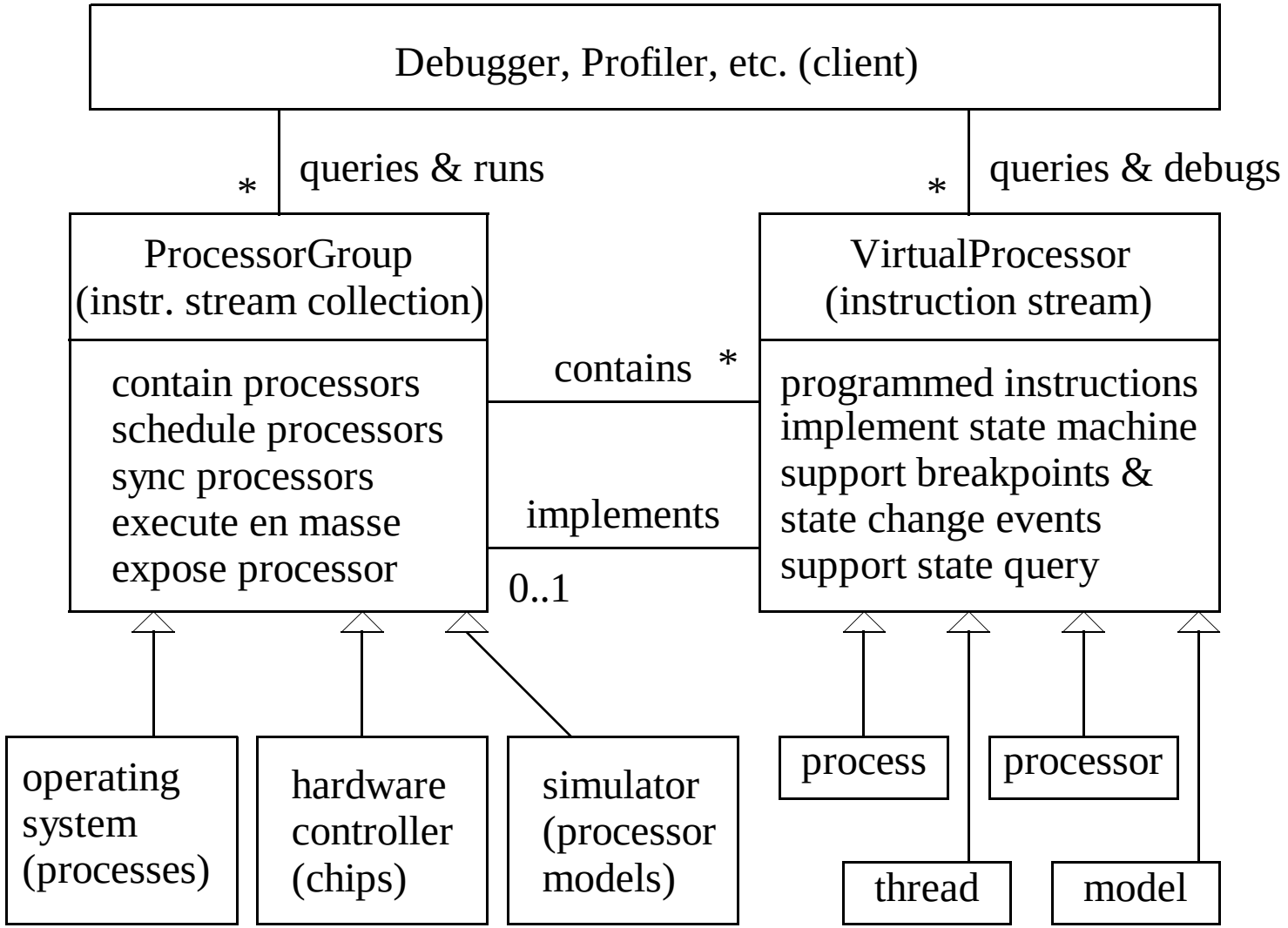




Circuit Simulator or Functional Simulator Object runs on Simulation Processor to provide access to Simulated Processor Models



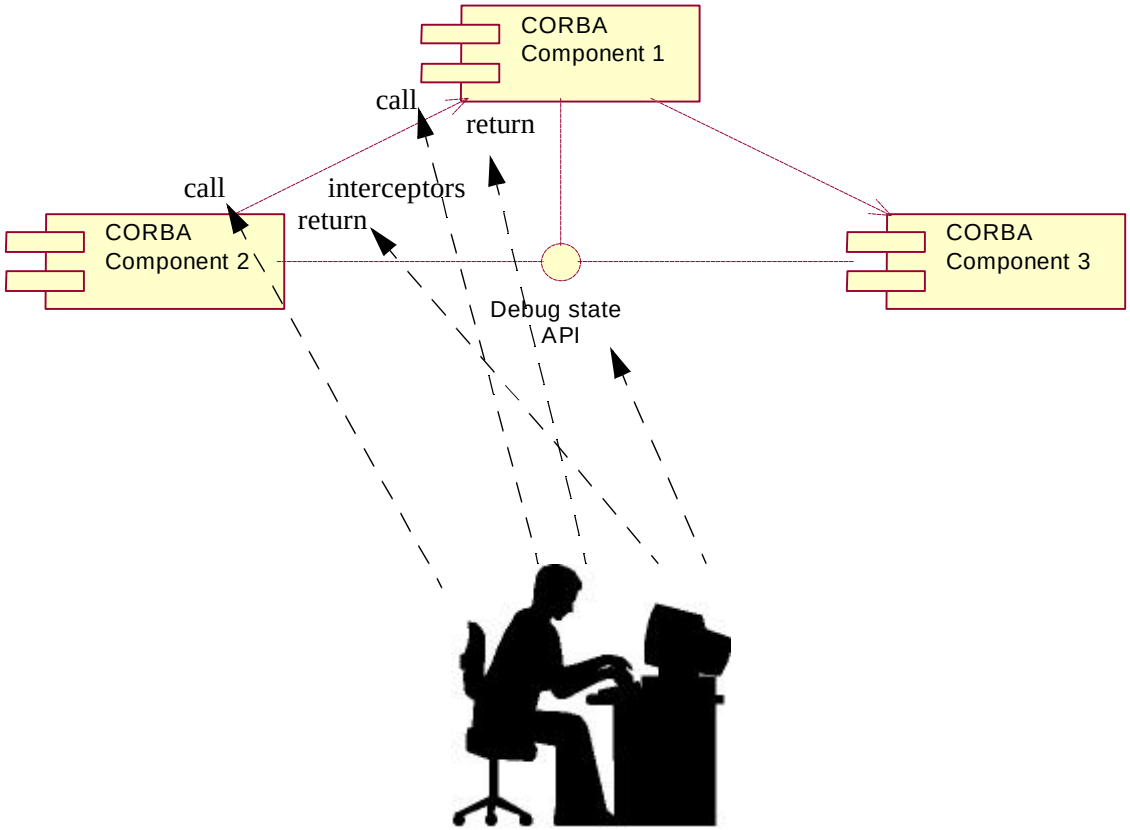
HIERARCHY: Persistent virtual processor servers (O.S., hardware interface card or simulator) provide access to multiple Virtual Processors..


Debugger, Profiler, etc. (client)
*
queries & runs
*
queries & debugs
ProcessorGroup
(instr. stream collection)
contain processors
schedule processors
sync processors
execute en masse
expose processor
contains
*
implements
0..1
VirtualProcessor
(instruction stream)
programmed instructions
implement state machine
support breakpoints &
state change events
support state query
operating
system
(processes)
hardware
controller
(chips)
simulator
(processor
models)
process
processor
thread
model

Four-part debugging / profiling API proposal:

- Black box CORBA component debugging
- Networked, heterogeneous source debugging
- Real-time, embedded system debugging
- Simulated processor system debugging

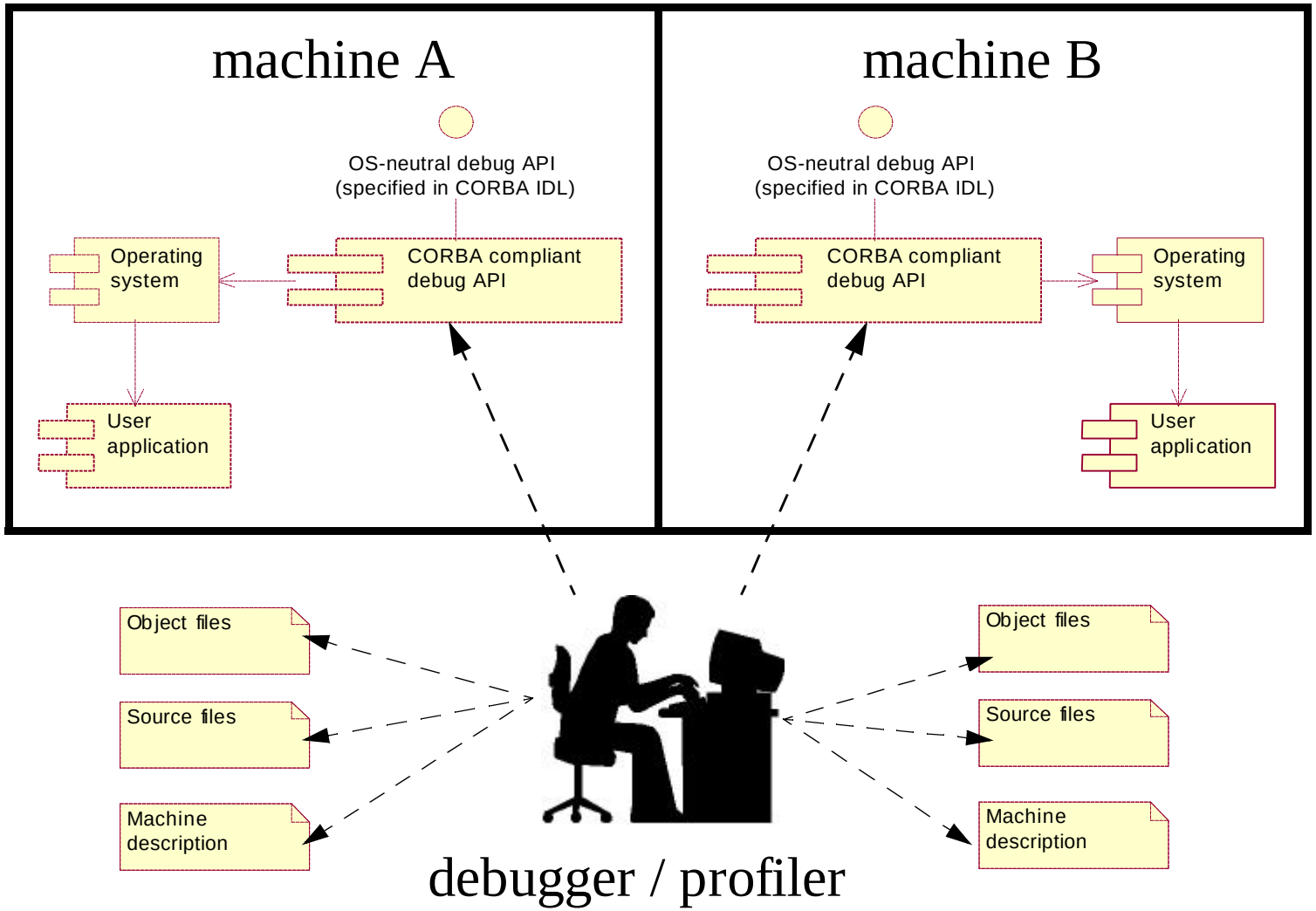
Black box CORBA component debugging



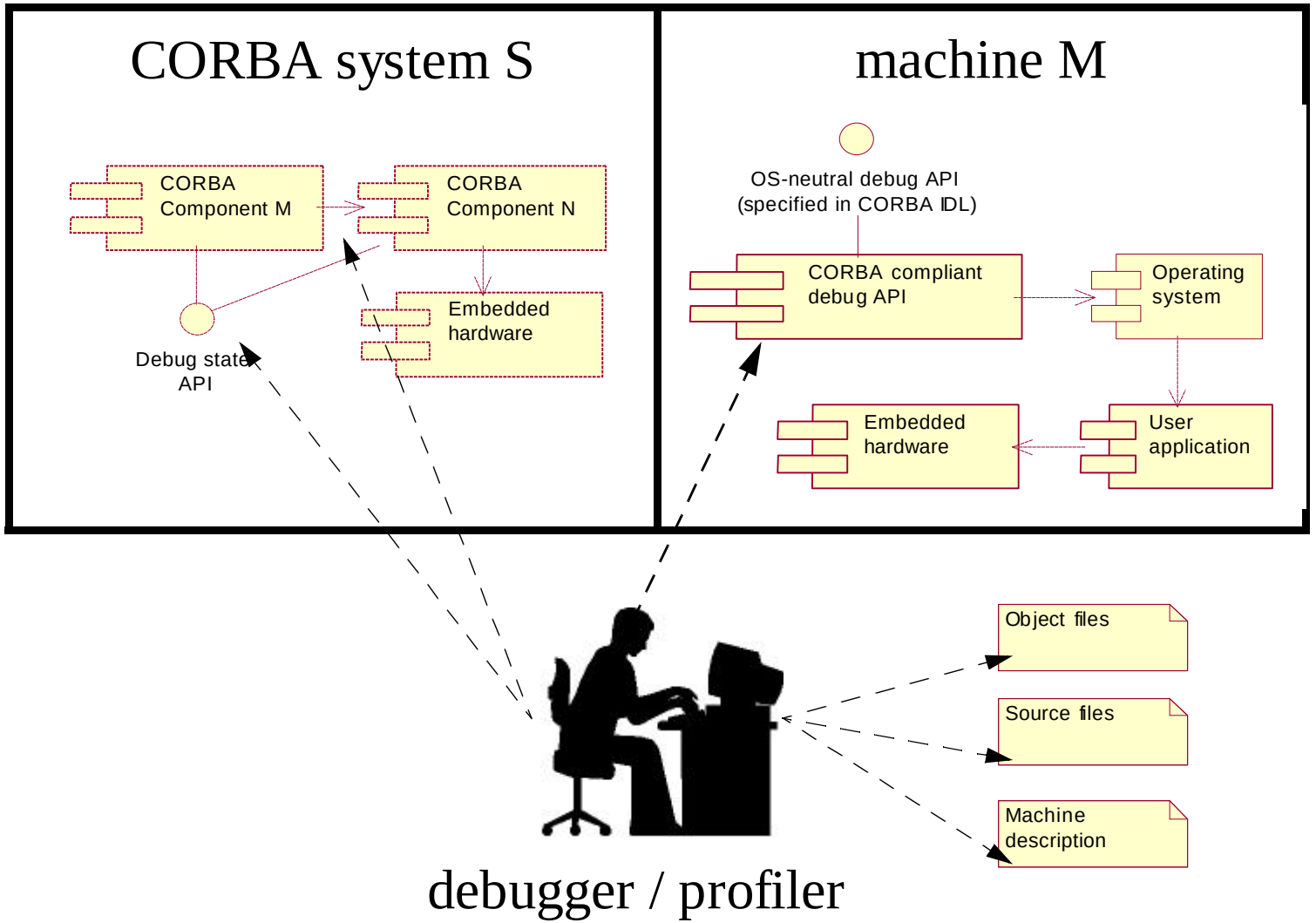
Black box CORBA component debugging (mandatory)

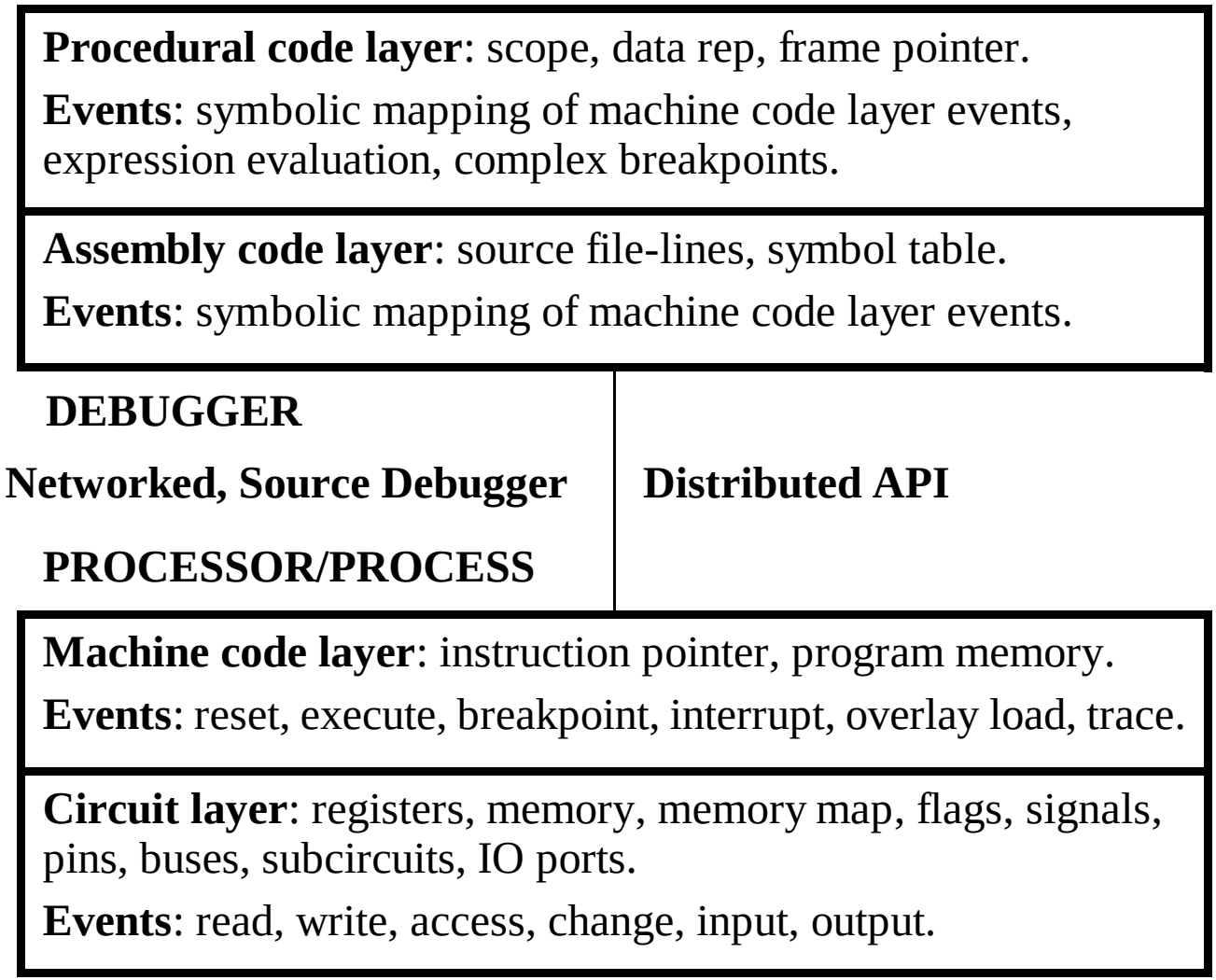
- Interface repository identifies operations, parameter types, return values
- Interceptors support inspection of client-side and server-side parameters and return values (orbos/98-09-11)
- Debugger can halt & restart server at intercepted points (black box breakpoints)
- Debug / profile state API, described via meta-data, available for debugger state inspection
- Interceptors can buffer information, raise exceptions on overflow

Networked, heterogeneous source debugging (optional)

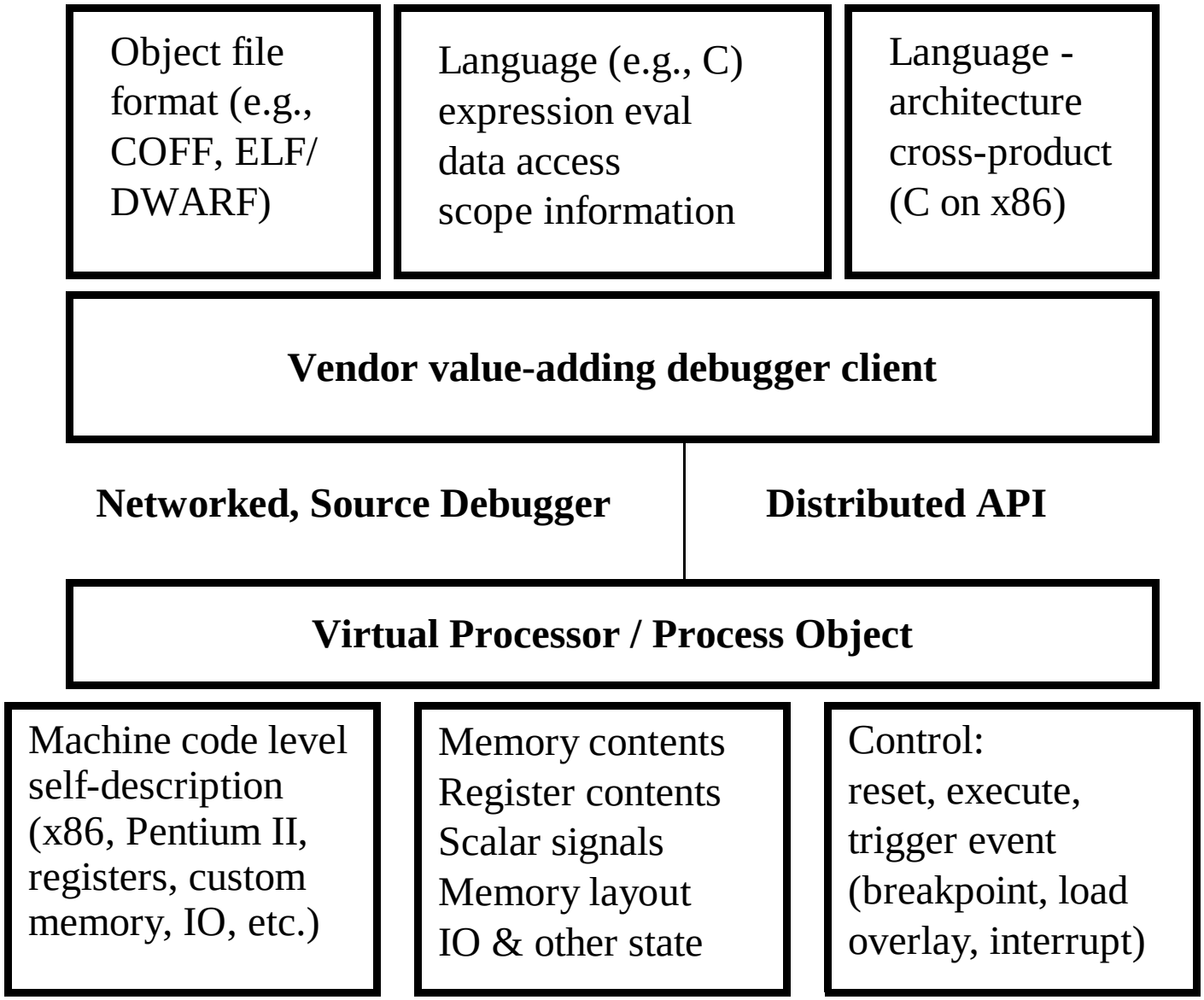


Real-time, embedded system debugging (optional)





Layers of Virtual Processors Available for Debugging


Object file format (e.g., COFF, ELF/ DWARF)
Language (e.g., C) expression eval data access scope information
Language - architecture cross-product (C on x86)
Vendor value-adding debugger client
Networked, Source Debugger
Distributed API
Virtual Processor / Process Object
Machine code level self-description (x86, Pentium II, registers, custom memory, IO, etc.)
Memory contents Register contents Scalar signals Memory layout IO & other state
Control: reset, execute, trigger event (breakpoint, load overlay, interrupt)

Issues for embedded systems

- Intra-processor and inter-processor signals
- Inter-processor subcircuits, IO objects, registers
- Pins and buses
- Timing — clock ticks, measurement, sync
- Execution & breakpoint control in wiring events
- Other asynchronous events
- Possible lack of an operating system

Issues for simulated systems

- Observation hooks exceeding physical systems
- Simulated time is important
- On-the-fly tuning of processor architecture
- Multi-valued bit states — 0, 1, U, Z, etc.

Options in the RFP

- Source code debugging
- Real-time or embedded systems
- Debugger can halt & restart one client thread at server intercepted points
- Call tracing through distributed CORBA system
- Meta-Object Facility support for automatic configuration of debugging tools, based on MOF model or meta-model

Other standardization efforts in the debugging, embedded system or processor CAD industry.

Open Microprocessor Systems Initiative (OMI)

- European consortium associated with Esprit
- Generic Debug Instrument Interface (GDI), 1/98
- Addresses embedded hardware and simulation
- Marginally addresses multiprocess debugging
- No support for multi-valued simulation logic
- Six companies involved — Kontron, Siemens, TASKING, IMEC, Syndesis, Synopsis
- www.omimo.be and www.tasking.com

JavaSoft's distributed Java™ debugger protocol (JPDA)

- http://java.sun.com/products/jpda/
- Java™ Virtual Machine Debugger Interface
- Java™ Debug Wire Protocol
- Java™ Debug Interface
- Single language support
- Single virtual machine support
- Distributed debugging
- Lowest level is Java class-object-method-frame
- No interface into "machine code" JVM

Distributed Debugging API status

- Revised draft RFP in San Jose in August
- Final revision, ORBOS and Architecture Board vote targeted for November
- OMG members (tools vendors) have around a year to respond to RFP with proposals
- Likelihood of source debugging & embedded system debugging unknown — major tools players are not fully engaged
- My guess is little or no source code debugging

Conclusions

- Programs will always have bugs & bottlenecks
- Programmers will need debuggers & profilers
- Lack of visibility of distributed program behavior puts an upper limit on the complexity of systems
- OMG should be a place to raise that limit
- Bell Labs / Lucent ME in conjunction with Lehigh University EECS, is supporting research into distributed debugging & profiling for heterogeneous distributed systems