A Framework for Simulating Heterogeneous Virtual Processors

32nd Annual Simulation Symposium, System Simulation session

April 12, 1999

Dale Parson, Paul Beatty, John Glossner and Bryan Schlieder

Bell Labs Innovations for Lucent Technologies

dparson@lucent.com

Design Patterns Generate Behavior across Media

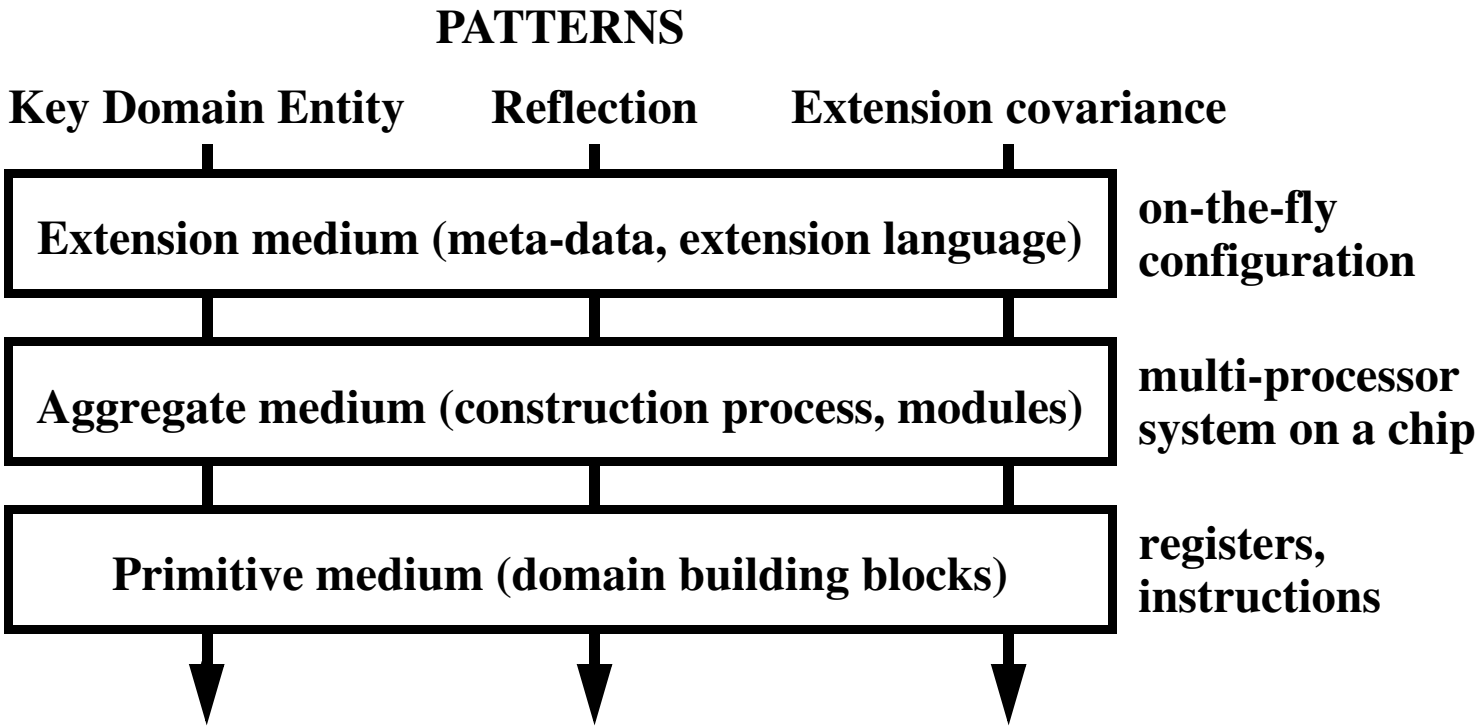
> The question is whether it is possible to write down rules or patterns for architecture—and software and art—so that ordinary people can follow the rules or patterns and, by the nature of the patterns and using only the abilities of ordinary people, beauty is generated. If this happens, then the rules or patterns are generative, which is a rare quality.
>
> — Richard Gabriel, *Patterns of Software*

LUxWORKS *luxdbg* simulator / debugger applies three design patterns across system layers:

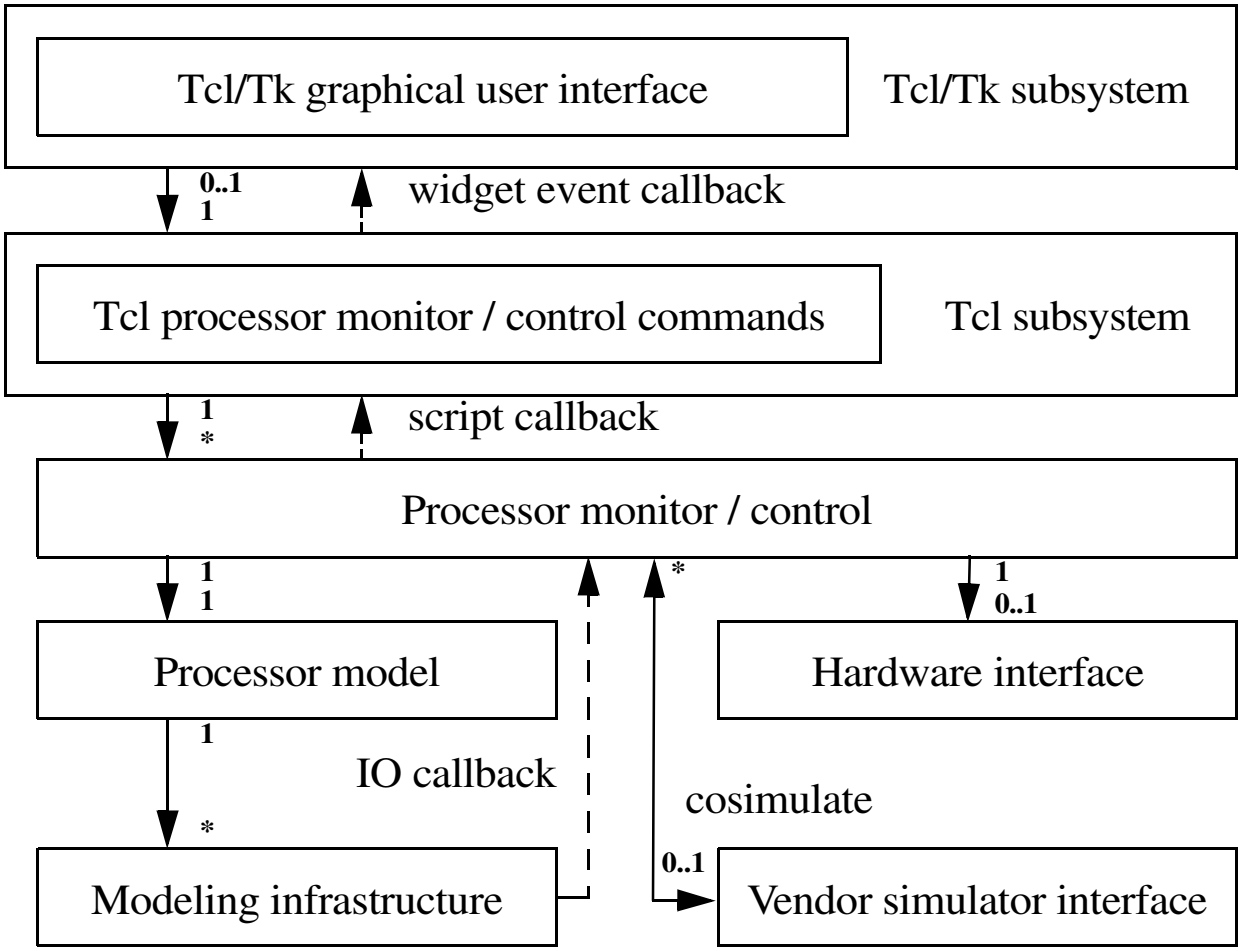
- Build and extend abstract virtual processors.
- Build reflective entities.
- Build a covariant extensible system.

Universal Design Patterns constitute a *lingua franca*

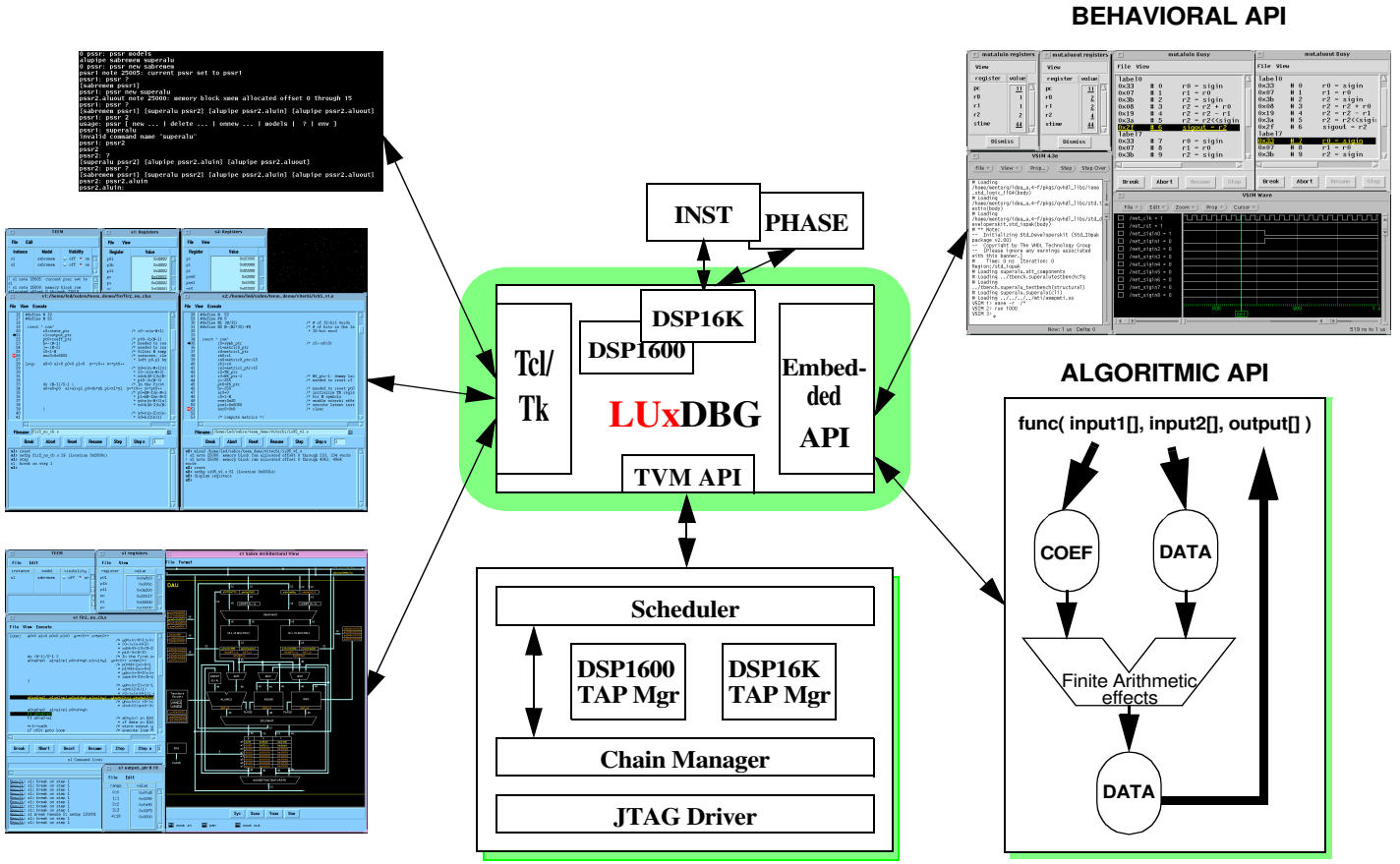
"I can go anywhere in the system and recognize what is going on."
— a processor designer using LUxWORKS framework.



Layers of the LUxWORKS Framework

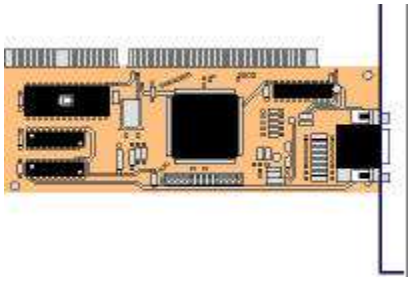


Modules of the LUxWORKS Framework

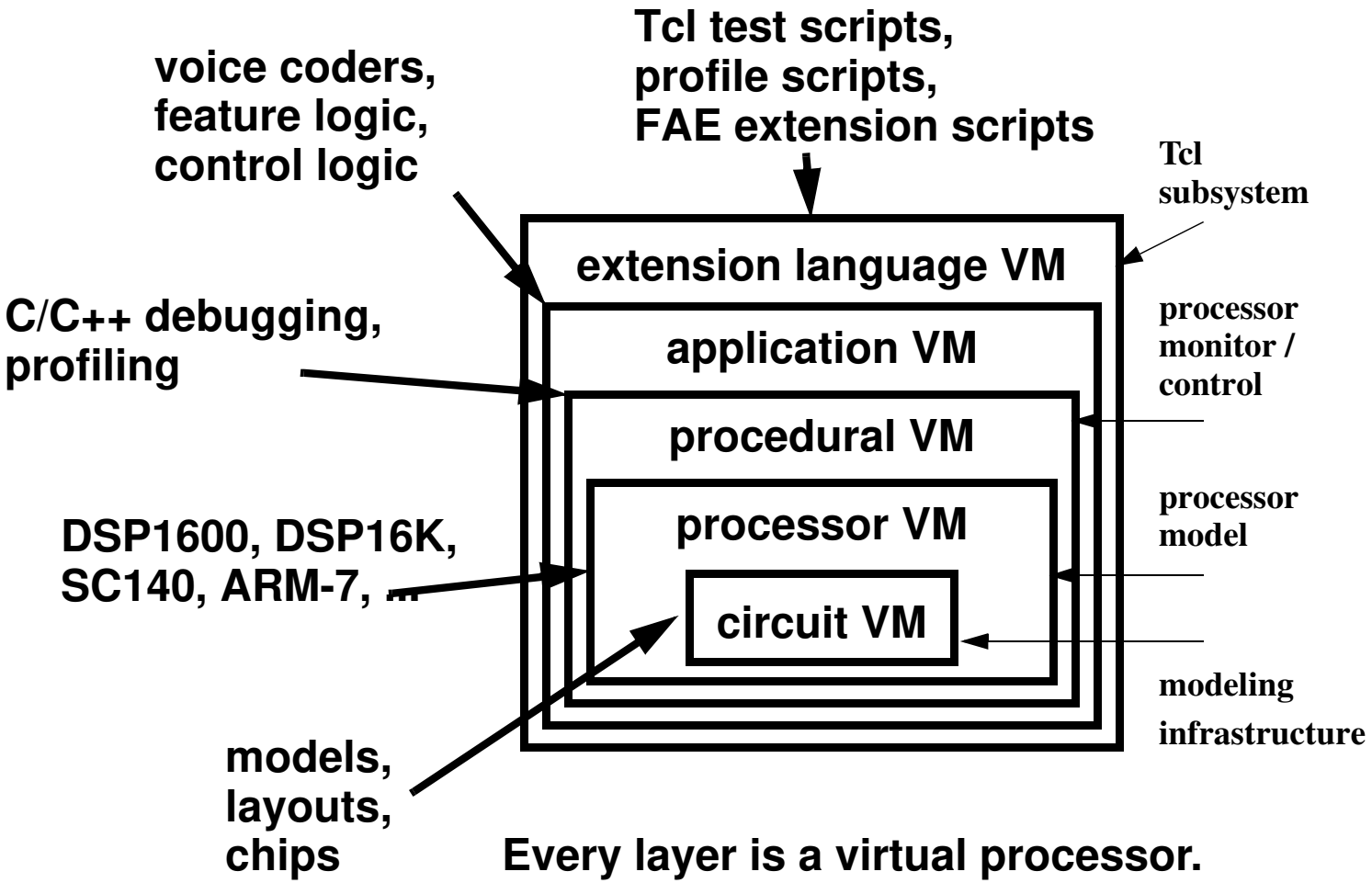


PATTERN #1: Build and extend abstract virtual processors.

```
for (tap = 0 ; tap < numtap ; tap++) {
        accum += sample[tap]
                * coefficient[(tap+offset) % numtap];
}
```



PATTERN #1: Build and extend abstract virtual processors.

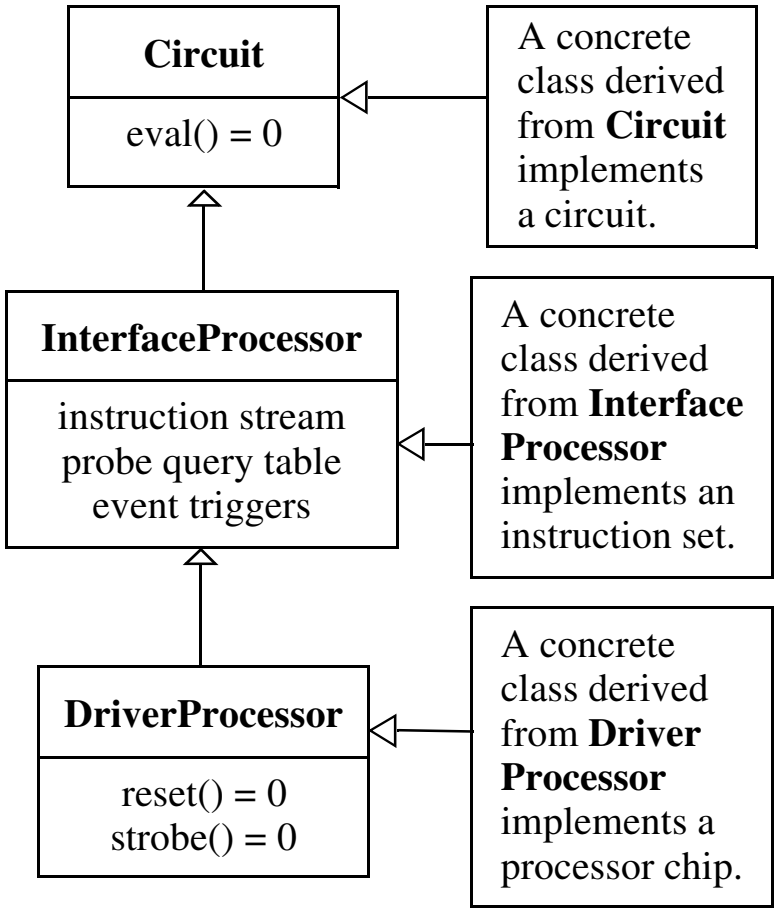


What can one do with an Abstract Virtual Processor?

Update output and state from input and state

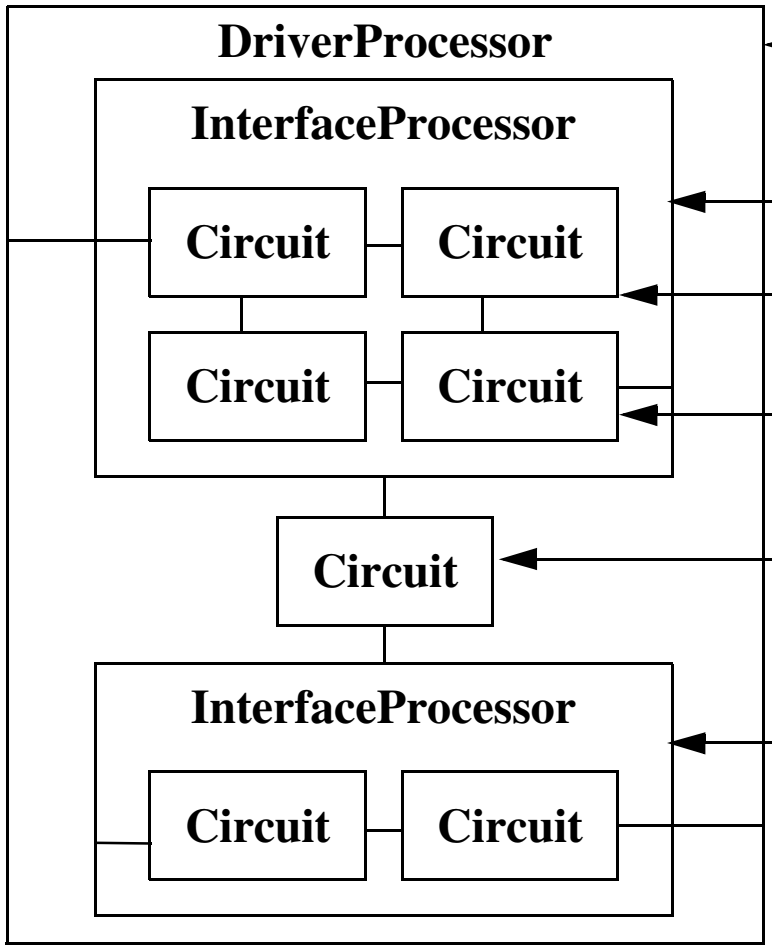
Run, stop, trigger breakpoint, fire exception, program

Reset, synchronize, connect to world



How can one combine Abstract Virtual Processors?

Connect them at construction time.



ARM / DSP16K System in a chip

ARM

control

datapath

shared memory

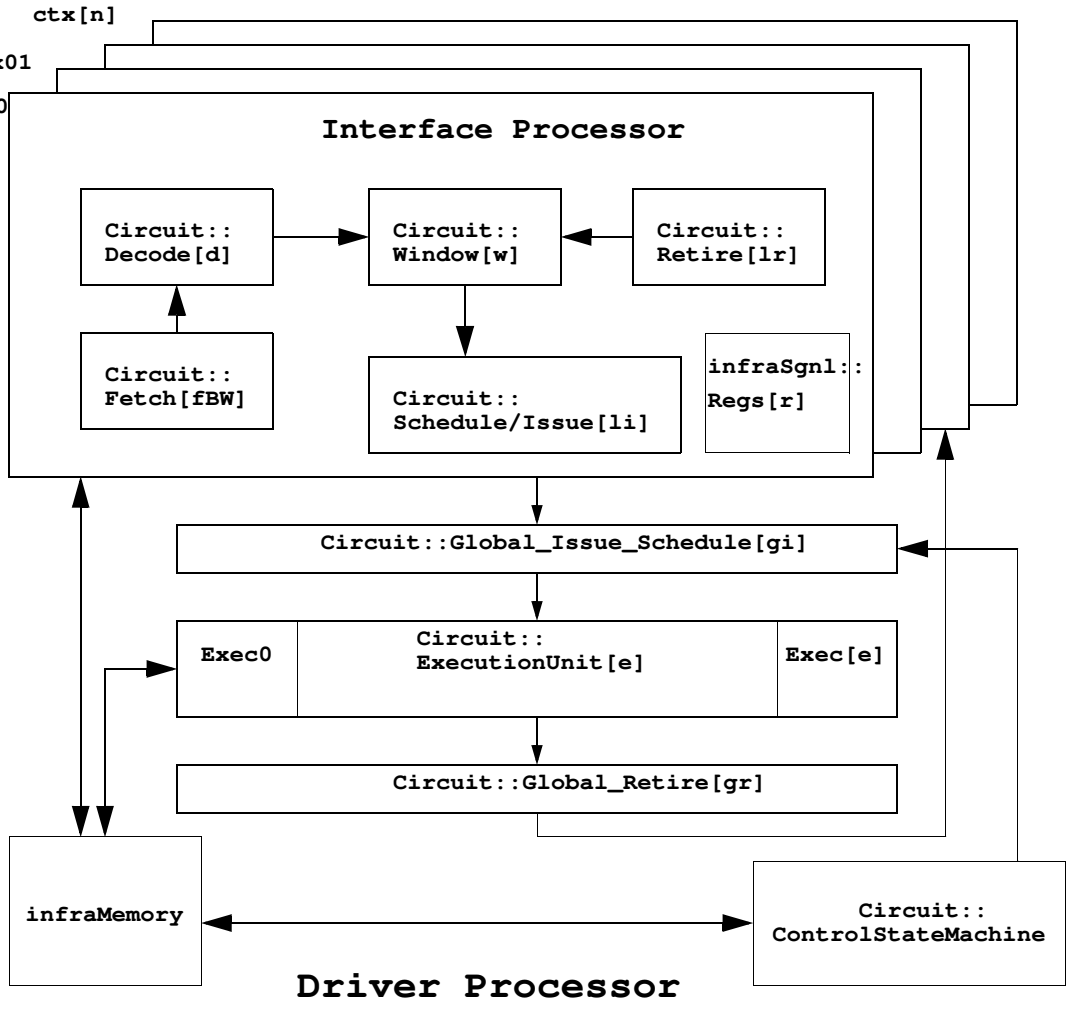
DSP16K

Abstract Virtual Processors for architectural prototypes

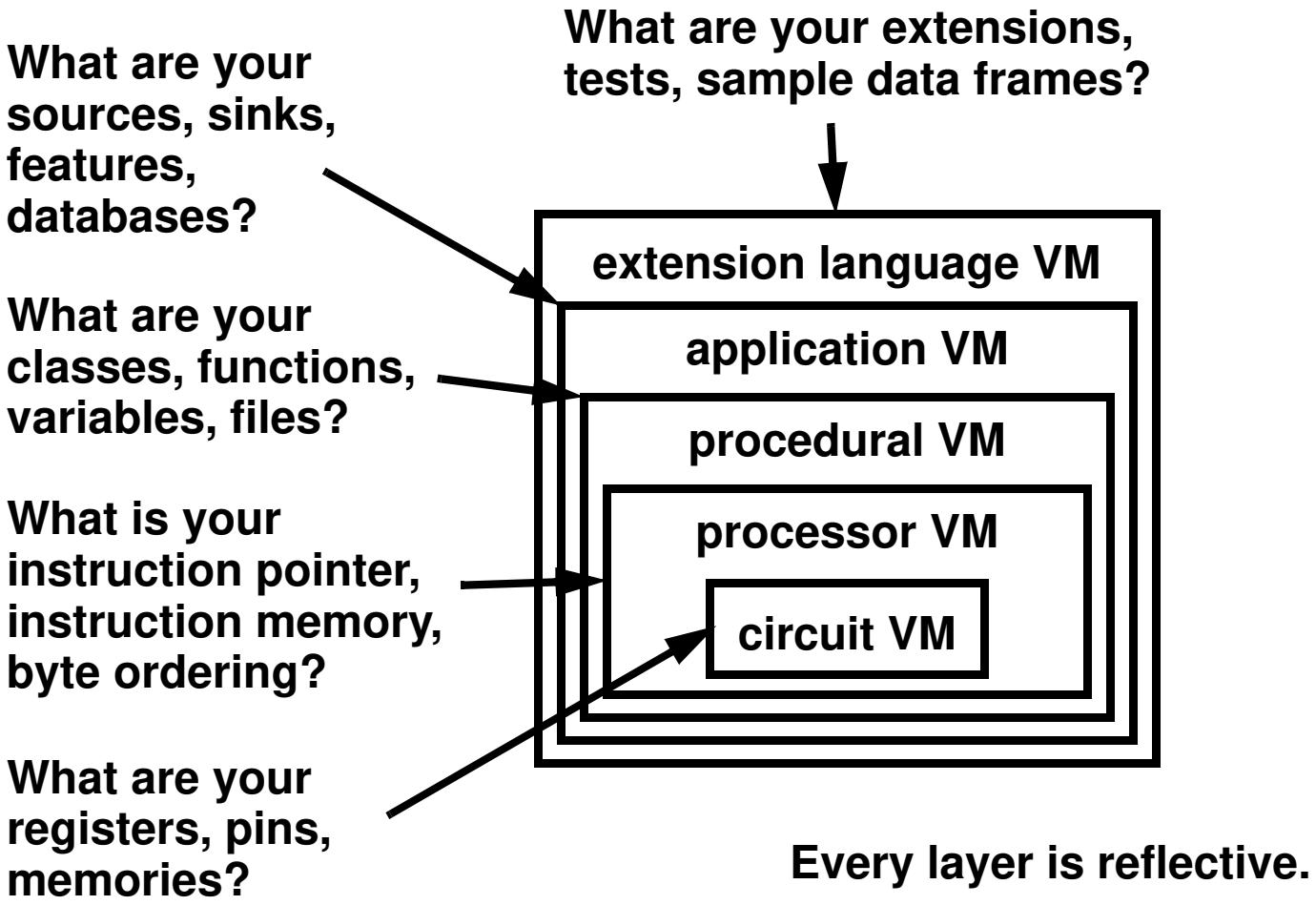
Run, stop, trigger breakpoint, fire exception, program

Update output and state from input and state

Reset, synchronize, connect to world

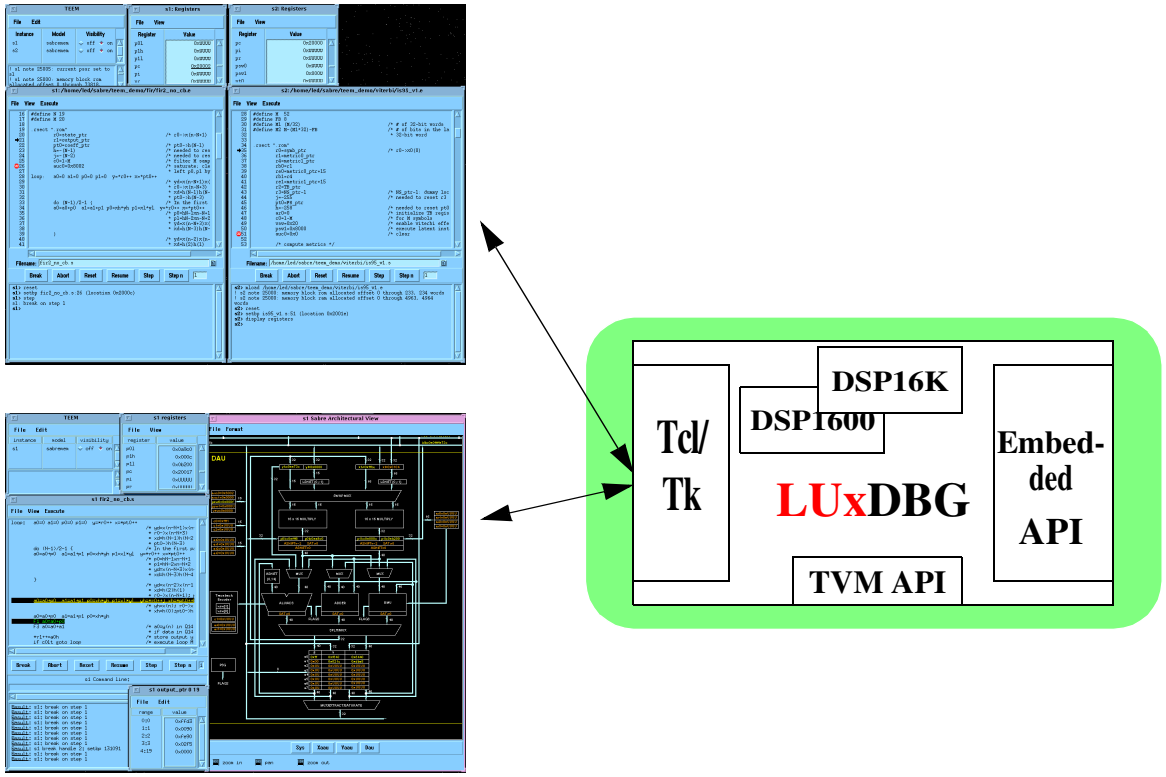


PATTERN #2: Build reflective entities.



Every layer is reflective.

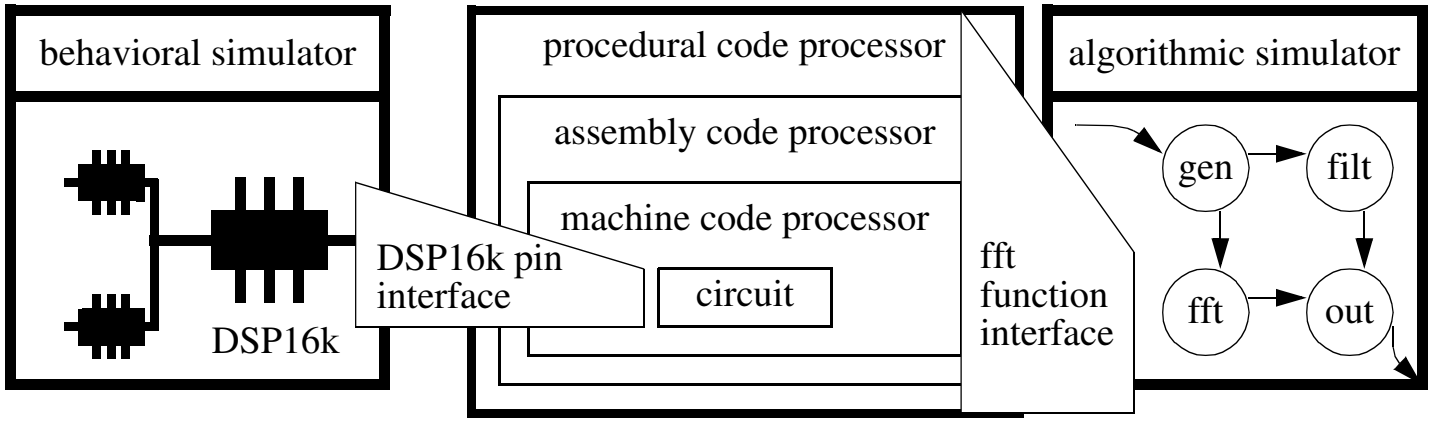
Reflective processors yield configurable tools.



GUI uses relational and hierarchical mega-widgets.
GUI configures itself to processors & programs at run time.

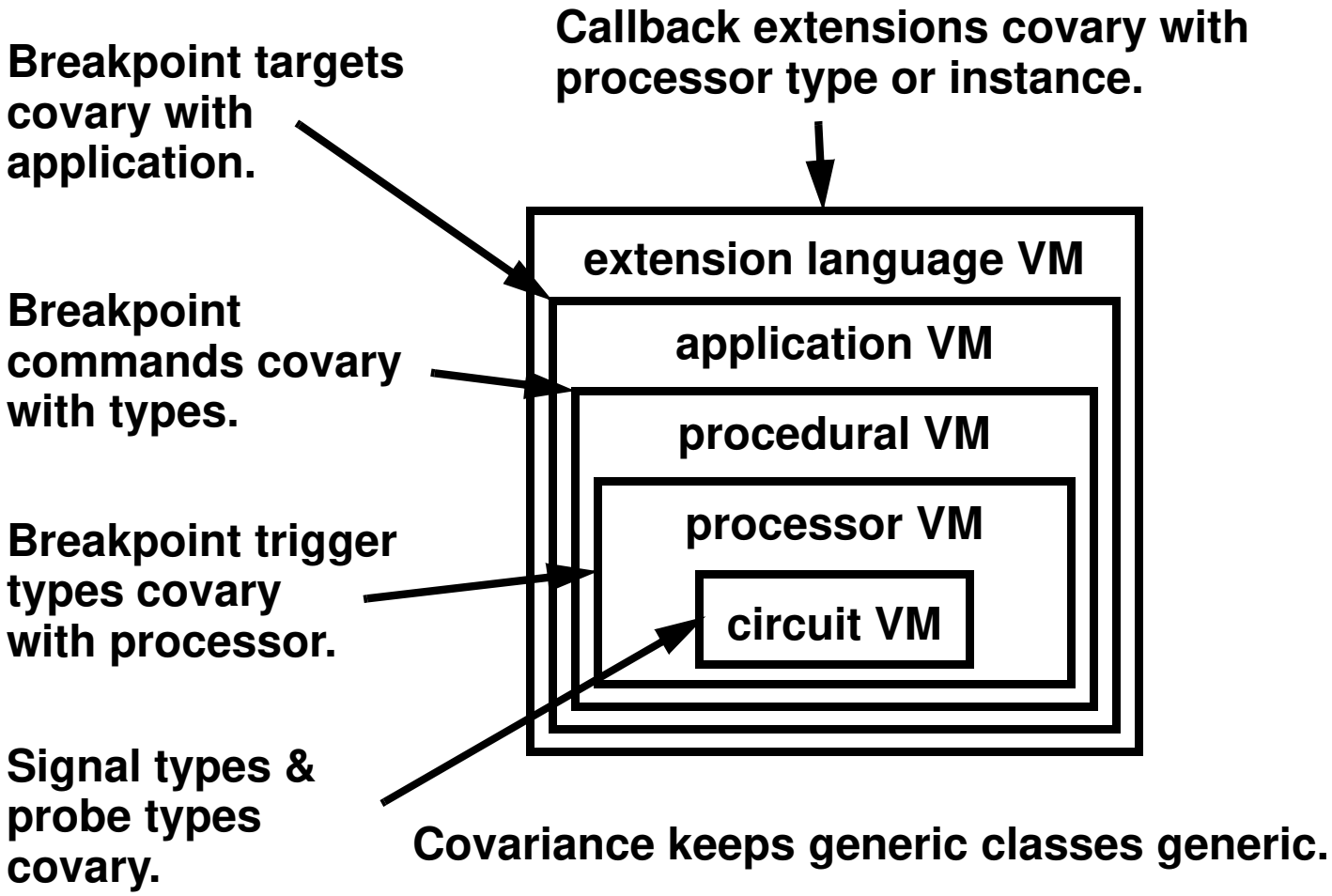
Reflective processors yield configurable bridges to tools.

Behavioral simulator bridge queries simulator & LUxWORKS model for pin & clock timing bindings.

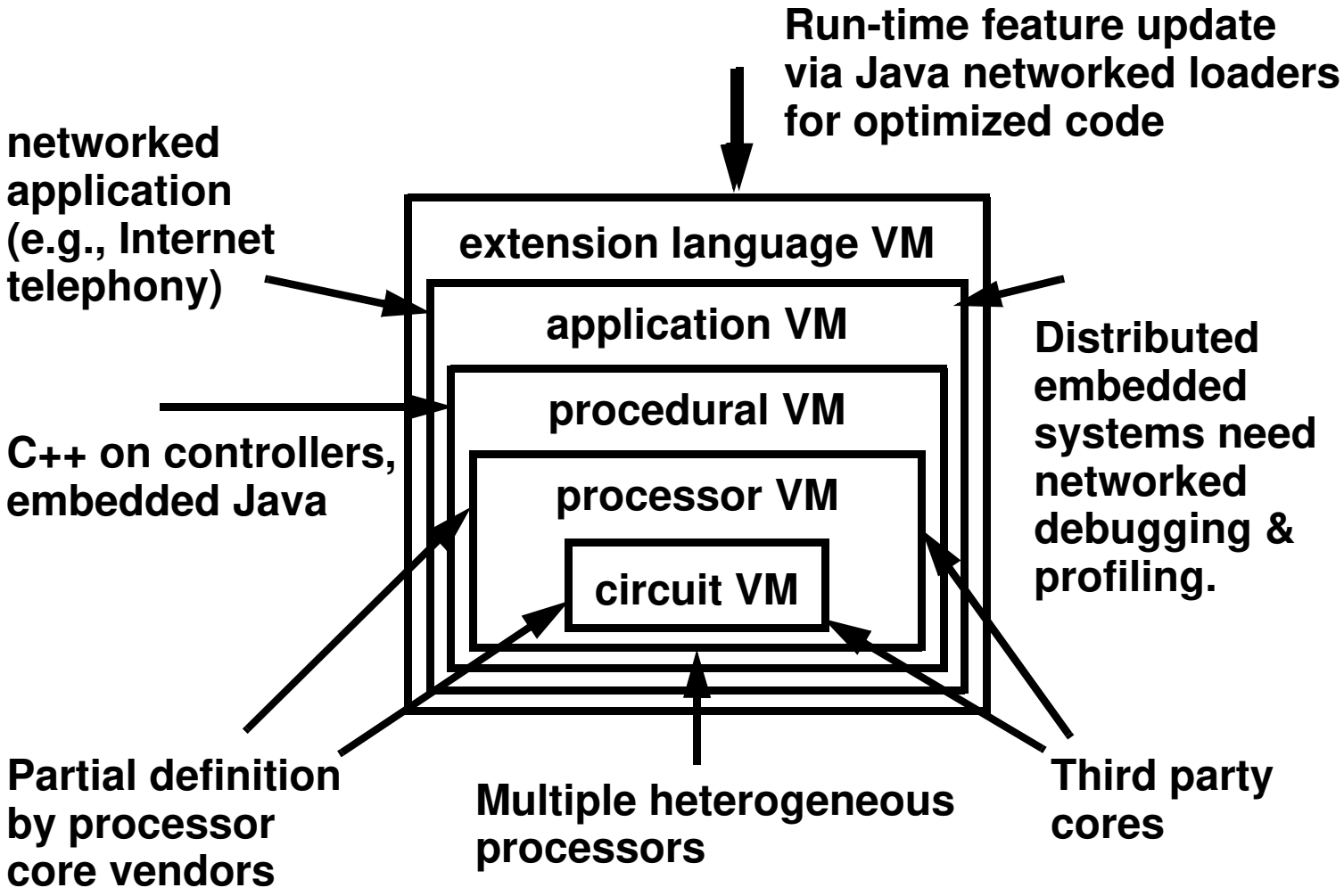


Algorithmic simulator bridge queries simulator & LUxWORKS symbol table for function & parameter bindings.

PATTERN #3: Build a covariant extensible system.



Forces driving simulation & debugging architectural evolution



Conclusions

- Virtual Processor pattern gives universal ability to trigger events, reset, execute, halt execution and deliver break events to clients.
- Universal inheritance from Virtual Processor base classes — Circuit and Interface Processor — supports on-the-fly composition of processors into higher-order system simulations.
- Universal reflection allows tools and tool bridges to configure themselves to processors and programs at execution time.
- Processor-oriented covariance allows tool capabilities to vary with process-specific support without perturbing generic tool code.
- Current direction is unification of simulation, embedded hardware execution & real-time operating system task execution.
- Increase in support for networked simulation & debugging.