ACME: An Interchange Language for Architecture Representation

David S. Wile
Research Professor
University of Southern California
Information Sciences Institute
wile@isi.edu

Community Consensus on:

- **Criteria and tools for defining and evaluating SGS architectures**
- **Role of software architecture in programs - impact on interoperability and standardization**
- **Approaches for representing software architectures for SGSs**
- **Implementation of SGS components/systems from architectures or frameworks**
- **Software technologies for support of architecture (e.g., object-oriented technologies, middleware)**
- **Mapping commercial products to reference software architectures**

Studies on Software Architectures

- Satellite Control Network (SCN)
- Milstar
- Global Positioning System (GPS)
- Defense Meteorological Satellite Program (DMSP)
- Phillips Laboratory Space Research Program
- Space Test and Evaluation (TE)
- Classified programs
- National Aeronautics and Space Administration (NASA)
- National Oceanic and Atmospheric Administration (NOAA)
- Commercial spacecraft programs
- Commercial ground system product suppliers.

Claims

- **Necessary to represent architectures formally to enhance communication among the GSAW stake-holders**
- **Necessary to exchange architecture information among a variety of representations used by the stakeholders**
- **May need dynamic mechanisms to mutate one architecture into another**

Architecture Design

- **Current practice:**
  - **ad hoc**
  - **informal**
  - **picture-based**
- **Therefore,**
  - **Poorly understood by developers**
  - **Designs cannot be analyzed for consistency or completeness**
  - **Constraints are not enforced during system evolution**
  - **No tools to help designers with their tasks**
- **Hence, we need *formal* architecture description languages**

Uses for ADL Specifications

- **Confluent terminology**
- **Structural specification for readers**
- **Application-independent analyses**
  - **connectors are connected:**
    - » **all top level inputs are inputs to some subcomponent**
    - » **all top level outputs are outputs from some subcomponent**
  - **contexts imposed on the same name are consistent**
  - **instantiations have the same number of input and output arguments as the generic**
  - **parts referenced by instances must be defined by generics**

Uses for ADL Specifications

- **Application-dependent analyses:**
  - **Interaction protocols**
  - **Bandwidths and latencies**
  - **Locations of resources**
  - **Anticipated dimensions of evolution**
- **Simulation**
- **Animation**
- **Instantiation to produce application code**
- **Traversal mechanisms for system programmers**
  - **apply to each**
  - **filter**
  - **choose subarchitecture**

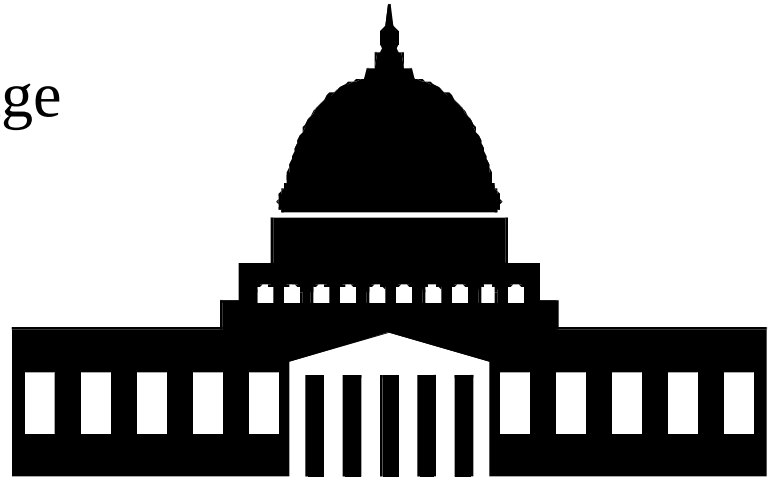
Formal Architecture Description Languages

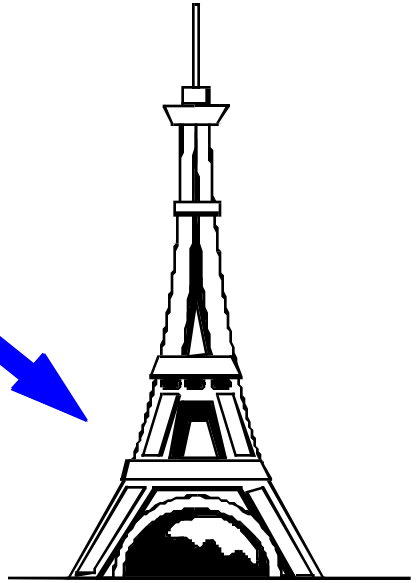
- **Aesop (Garlan at CMU): styles**
- **Adage (Coglionese at FSD): avionics navigation**
- **Meta-H (Vestal at Honeywell): real-time control**
- **C2 (Taylor at UCI): user interfaces**
- **Rapide (Luckham at Stanford): simulation and analysis**
- **SADL (Moriconi at SRI): refinement**
- **UniCon (Shaw at CMU): heterogenous styles**
- **Wright (Garlan at CMU): analysis of interactions between components**
- **Darwin (Kramer at Imperial): dynamic architectures**

ADL Proliferation

- **Plus side:**
  - **Exploring different facets of the overall problem**
  - **Tools developed for such exploration**
- **Minus side:**
  - **Stand-alone, stovepipe systems**
  - **Cannot combine with others**
  - **Must reimplement:**
    - » **graphical tools**
    - » **persistent stores for designs**
    - » **domain-independent forms of analysis**
  - **Investment to come on-board with a new application is heavy**

ACME:
An Architecture Interchange Language





ACME:
an Architecture Exchange Language

David Garlan (CMU)
Robert Monroe (CMU)
David Wile (ISI)

Goals for ACME

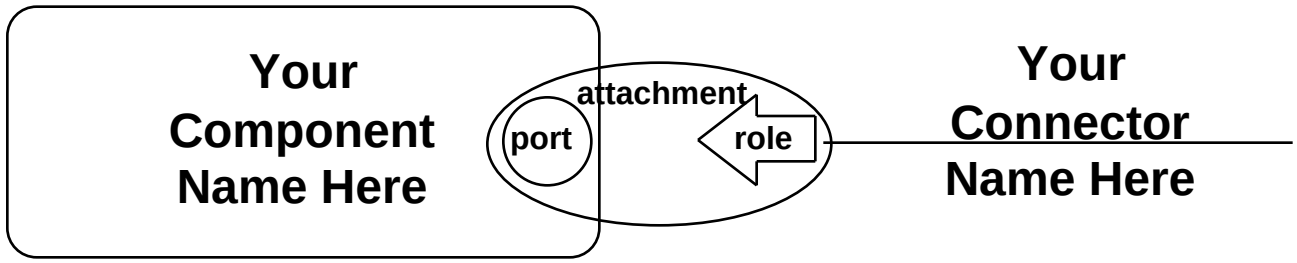
- **Interchange format for architectural development tools and environments**
  - **n * m problem -> m + n**
  - **tools**
    - » **graphical interface tools**
    - » **animation**
    - » **analysis for deadlock, well-formedness**
    - » **architecture style-specific tools**
- **Underlying representation for developing new tools for analyzing and visualizing architectures**
- **Foundation for developing new, domain-specific ADLS**

Goals for ACME

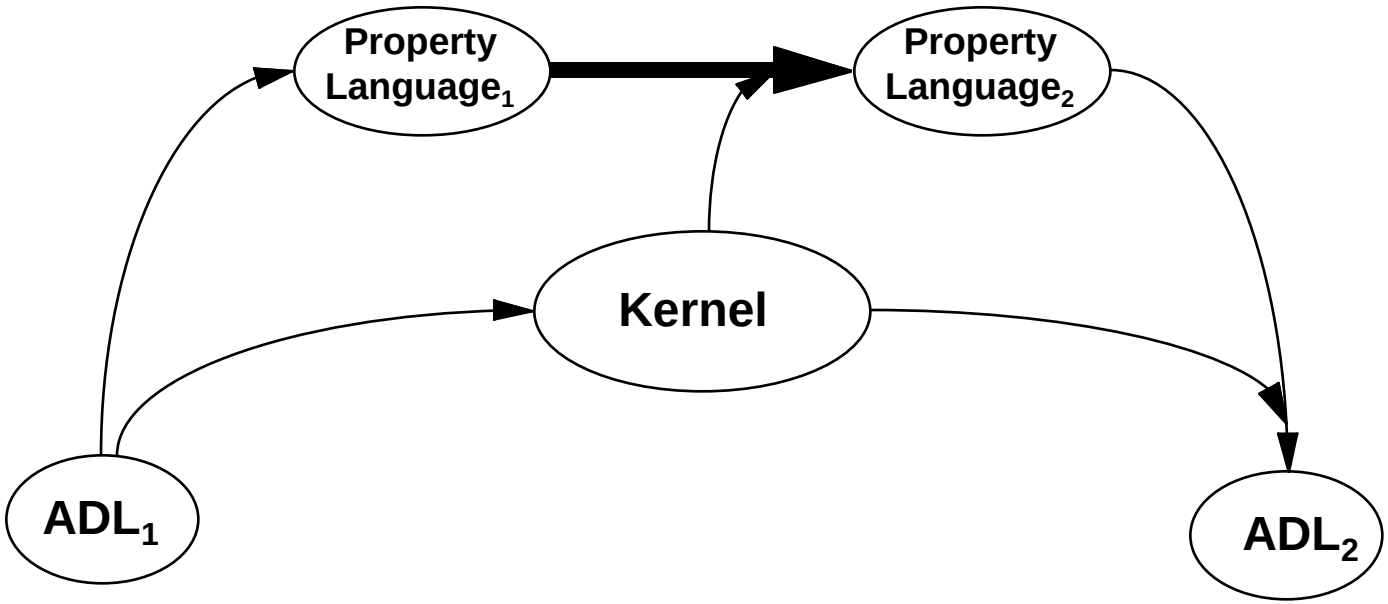
- **Vehicle for creating conventions: concensus building**
  - **Semantic foundations**
    - » **refinement**
    - » **event-based**
    - » **temporal logic**
  - **Architecture families**
    - » **Architecture evolution**
    - » **Dynamic architectures**
- **Expressive descriptions that are easy for humans to read and write**

ACME Kernel

- **Components, with ports**
- **Connectors, with roles**
- **Attachments of particular ports to particular roles**
- **Aggregates: collections of components, connectors and attachments**
- **Properties of any of above**



Translation between ADLs

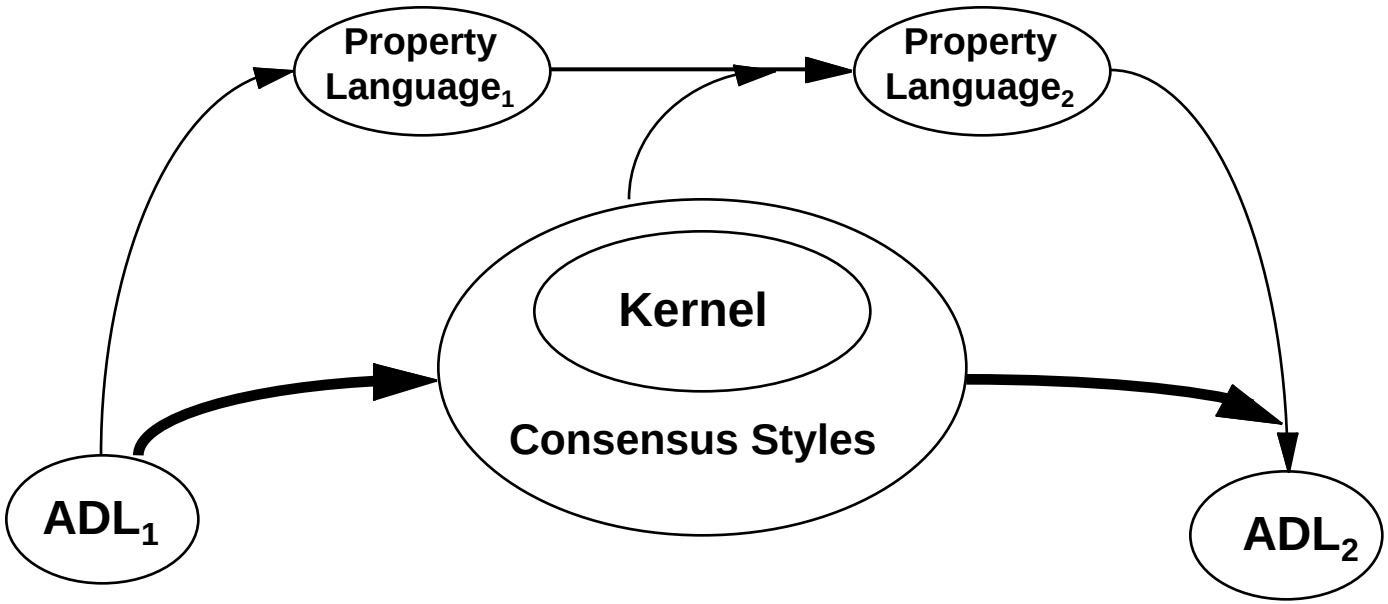


ACME Success Criterion:
Narrow arrow between property languages.

Additional Kernel Concepts

- **Need to extend Kernel to as large a language as is acceptable by the community**
- **Types**
  - **predicates, available for components, ports, connectors and roles**
  - **extendible**
- **Refinement**
  - **substructure specification**
  - **bindings of external interfaces to internal**

Introduce Community Consensus Styles



Narrows the arrow between property languages.

ACME Extensions to Kernel

- **Templates**
  - **typed macros**
  - **with typed arguments**
- **Families: styles and other constrained aggregates**
  - **specification as a set of templates and types**
  - **declaration of restriction to family enforces template usage**

Status

- **Exchange architectural information between a diverse set of architectural development and analysis tools**

Interchange Experience

- **Wright -> Rapide translation**
  - **Initial translation technology developed**
  - **One-way translation (not round trip)**
- **Aesop <-> ACME <-> UniCon**
  - **Aesop <-> ACME 1.0 works**
  - **Aesop <-> ACME 3.0 underway**
  - **UniCon <-> ACME 3.0 underway**
  - **Aesop <-> ACME <-> Unicon eventually**

Interchange Observations

- **One-way translation easier than round-trip**
- **Subtle semantic differences still a concern**
- **Expected properties problems not such a big deal**
- **ACME-based analysis tools perhaps more promising than ADL round-trip translation**

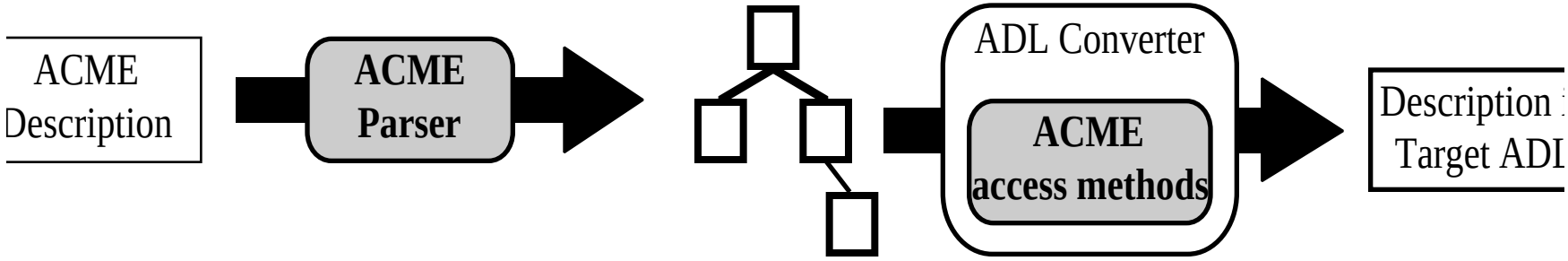
Future Interchange Directions

- **Translation into ACME**
- **ACME-based analysis, animation, simulation**
  - **UniCon "super make" system, providing automatic architecture compilation.**
  - **Rapide POSet analysis providing event-based deadlock, starvation, etc., analysis.**
  - **Rapide animator, given an architecture specification and event trace, providing flow visualization.**
  - **ACME translation to skeletal versions of Unicon, Aesop, Rapide, Wright, etc., as higher buy-in becomes warranted.**

Status

- **Web access to architectural descriptions**
- **Baseline tools for**
  - **manipulation,**
  - **analysis,**
  - **change-impact analysis of architectural structures**
- **that can be universally and transparently invoked from existing ADL platforms.**

Infrastructure



- ACME-Lib infrastructure
  - Extensible ACME parsers and unparsers
  - Extensible ACME translation tools
  - Native-ADL embeddable support
  - Support for design traversal, manipulation, and type-checking in ACME-native tools.

Ongoing Work

- **Prototypes for several ACME tools to be provided to the Architecture and Generation EDCS Cluster:**
  - **an ACME description repository,**
  - **various analyzers for connectedness and completeness,**
  - **and a translator from ACME into a predicate calculus - based semantics.**
- **Prototypes for tools that allow others to provide domain - specific analyzers, such as a**
  - **code - walker**
  - **ACME elaborator---a tool that translates extended, style - based ACME descriptions into the kernel language.**
- **Promised**
  - **ACME type checker**
  - **Tool to visualize ACME specifications graphically**

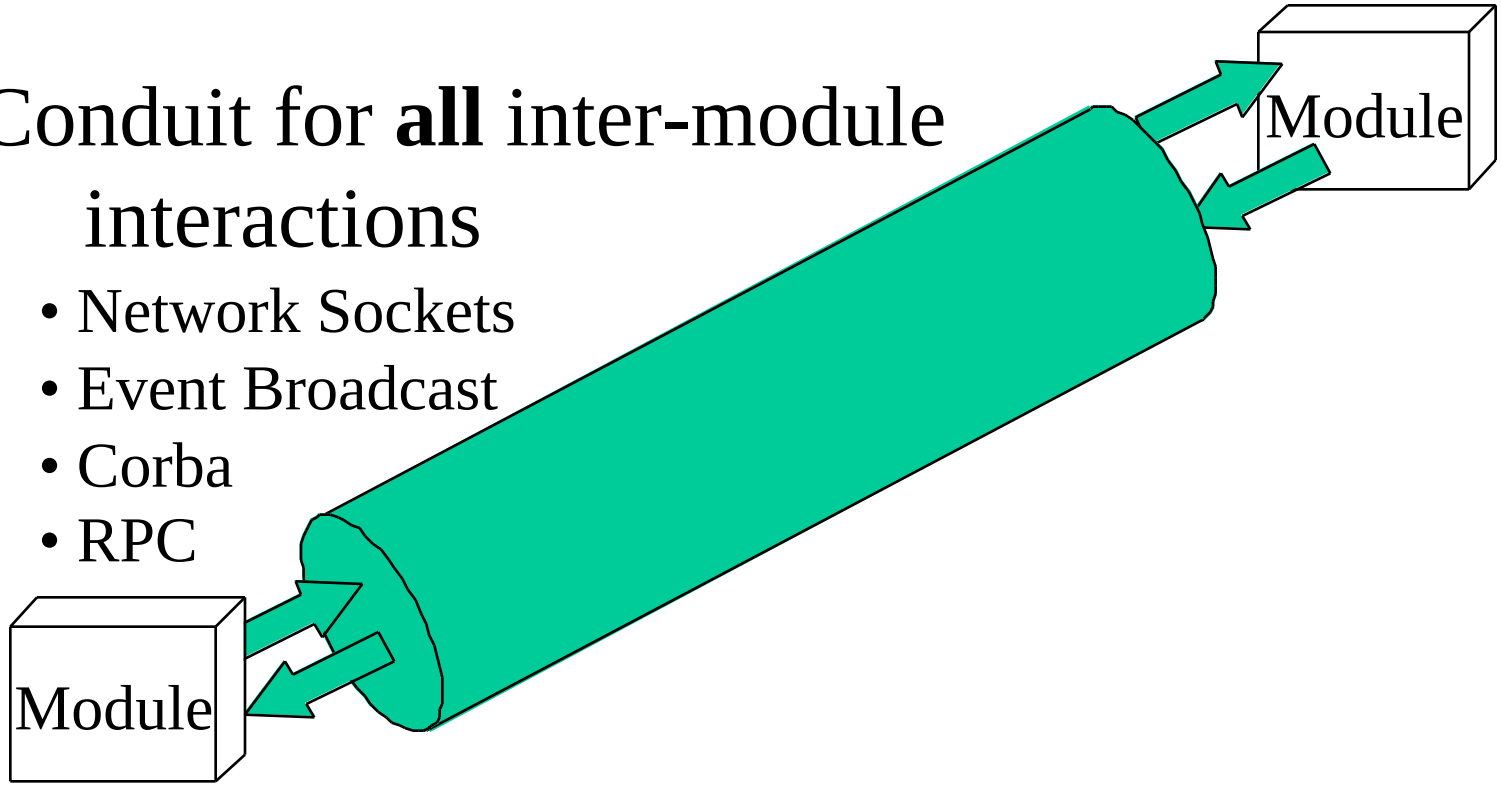
Instrumented Connectors

Robert Balzer

Architecture Connectors

Conduit for **all** inter-module interactions

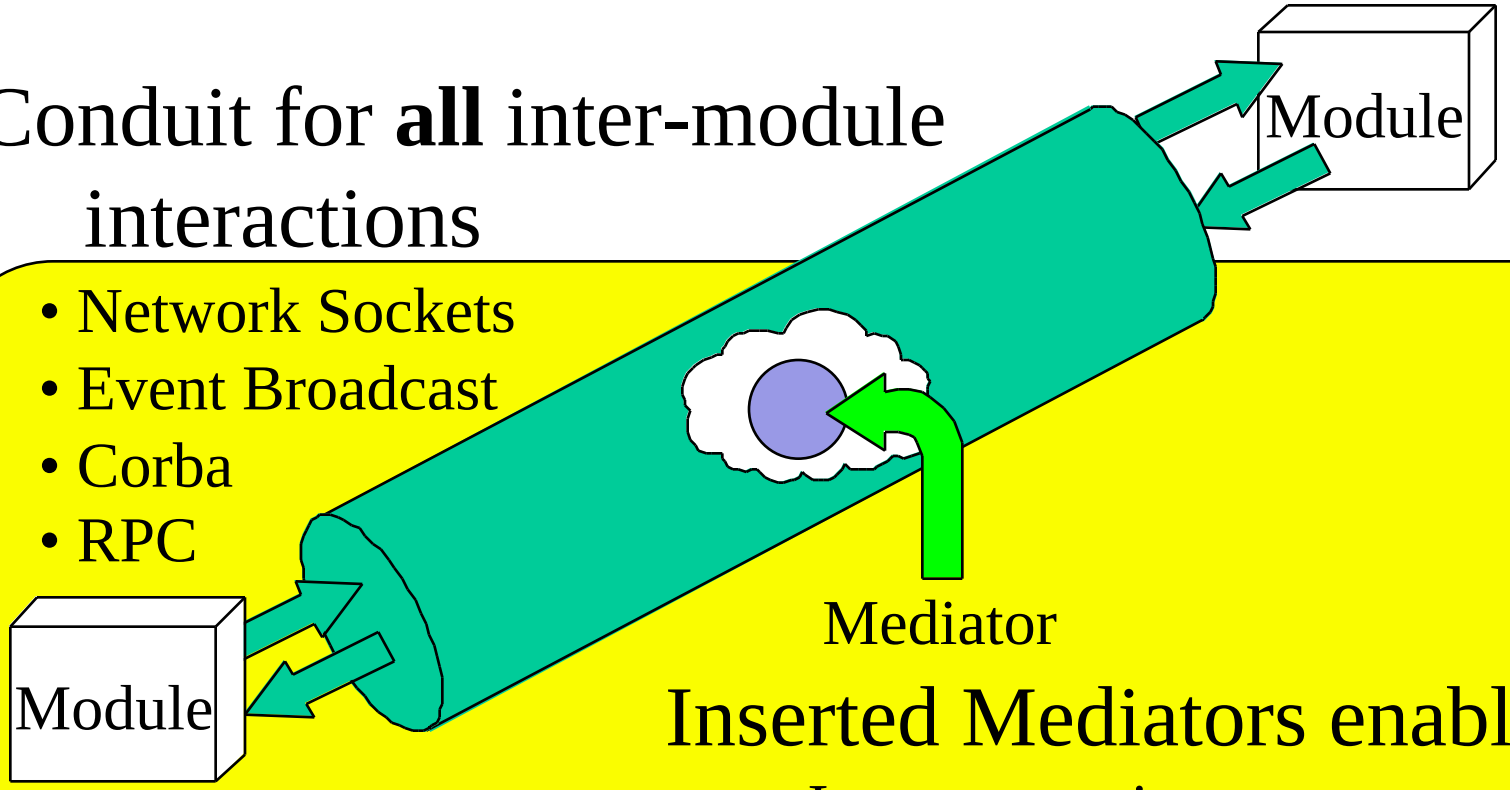
- Network Sockets
- Event Broadcast
- Corba
- RPC



Architecture Connectors

Conduit for **all** inter-module interactions

- Network Sockets
- Event Broadcast
- Corba
- RPC



Inserted Mediators enable

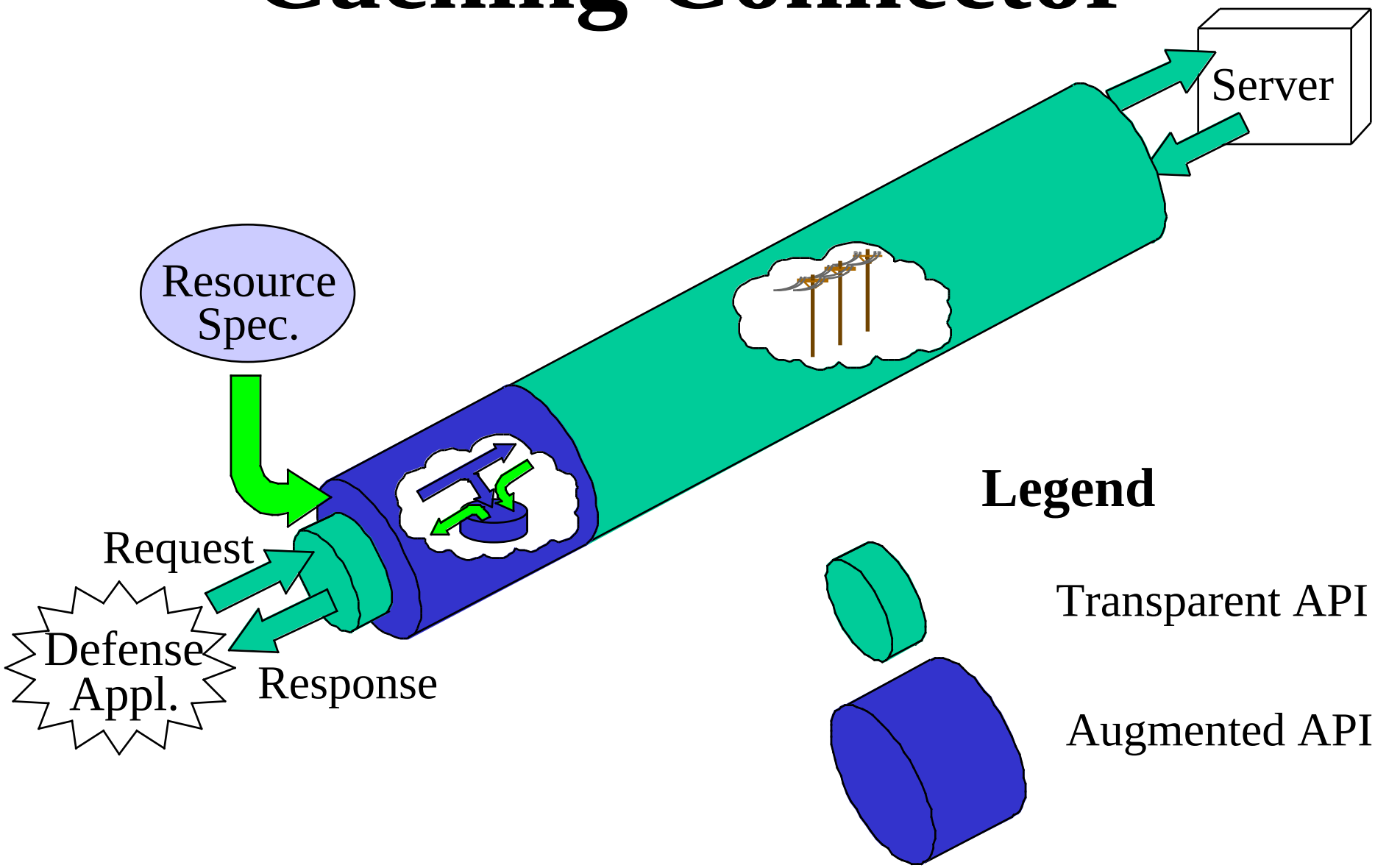
- Instrumentation
- Interface adaptation
- Filtering
- Value Added Infrastructure

Uniform Mediator Interface Spanning Integration Frameworks

Semi-Transparent Interfaces

- **Transparent API**
  - **Useable by Unmodified Applications**
- **Augmented API**
  - **Provides Value-Added Capabilities**
  - **Sources of Control**
    - **Static “Resource” Configuration Files**
    - **Dynamic Third-Party Controller**

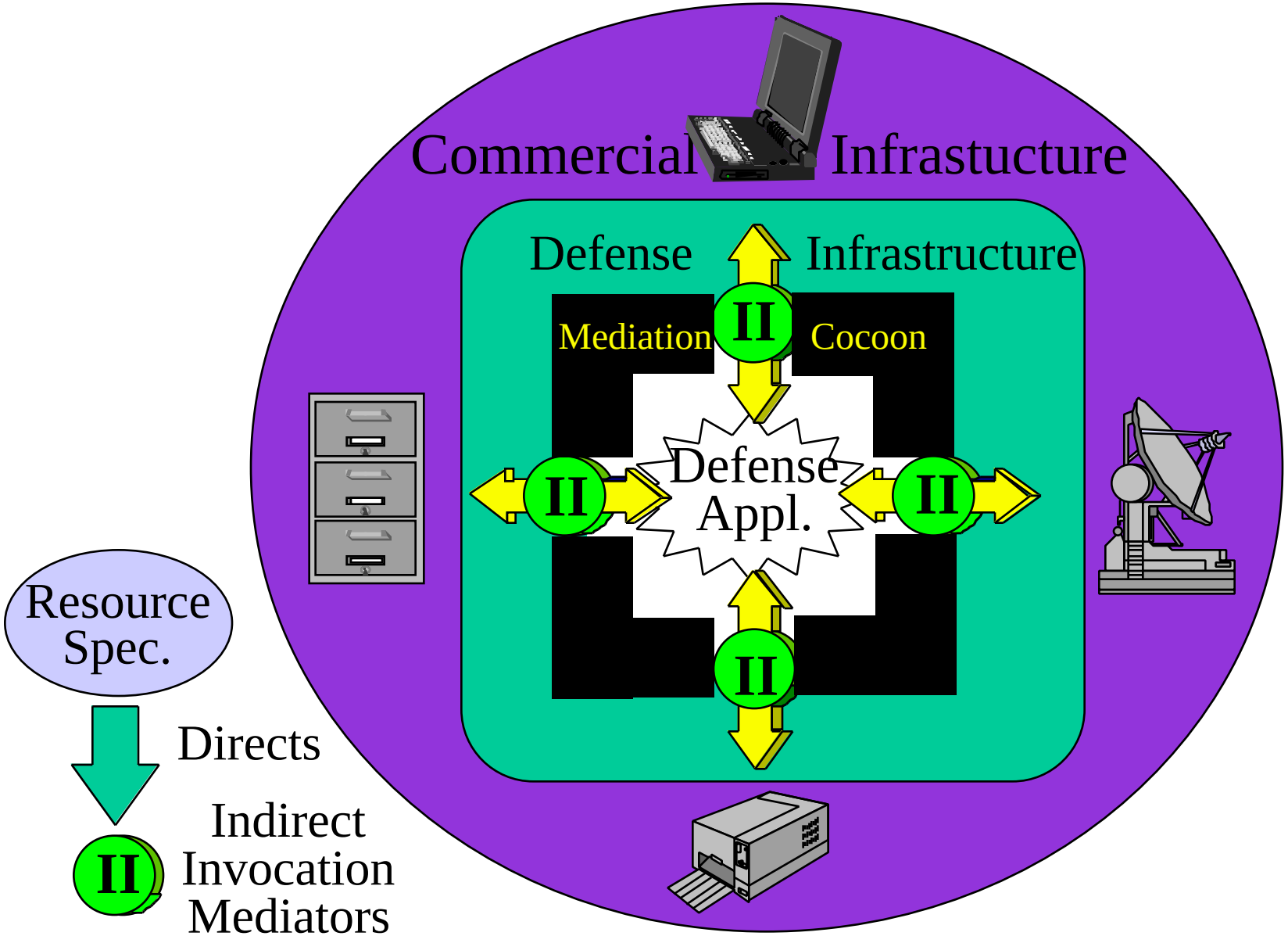
Caching Connector



Caching Connector
Augmented API

- **What to Cache**
  - **Matching Client Request with Server Response**
- **Cache Integrity**
  - **Is a Cached object still valid**
    - **Always**
    - **Time Duration or Observable Event**
    - **Query to Determine**
- **Cache Retention Policy**
  - **Flush Least Recently Used**
  - **Retain/Flush Selector**
- **Cache Allocation**
  - **Initial size**
  - **Dynamic Growth**

Augmented Infrastructure Through Indirect Invocation



Architecture Infrastructure for Inter-Module Interaction

- **Provide infrastructure for managing and manipulating inter-module connectors**
  - **(All inter-module interactions occur through these connectors)**
  - **dynamic probes - instrument & monitor behavior**
  - **redirect or alter messages, spawn reactive processes**
  - **move events from one integration space to another**
- **Allow others to provide middle-ware services based on this infrastructure**

Integration Paradigms

- **Old Style - CoResident**
  - **Shared Memory (including global variables)**
  - **Direct Subroutine Calls**
- **New Style - Distributed & Autonomous**
  - **Object Oriented (CORBA, OLE2)**
  - **Event Based (Broadcast Message Server)**
  - **RPC**
  - **Client/Server**
  - **Protocol Stacks**

Provide Integration Across This Class