

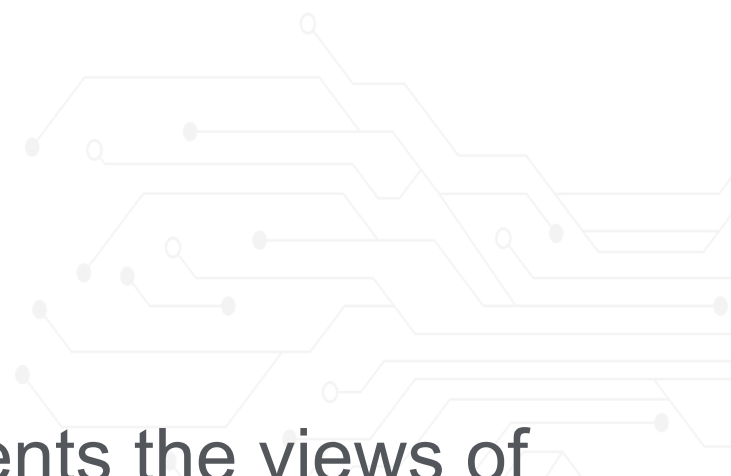


IEEE P2654™ System Test Access Management (STAM) Update


International Test Access, Automation & Adoption (TAAA) Workshop

Bradford G. Van Treuren





“This presentation solely represents the views of this Working Group and does not necessarily represent the position of either the IEEE or the IEEE Standards Association”



Active Members (13)

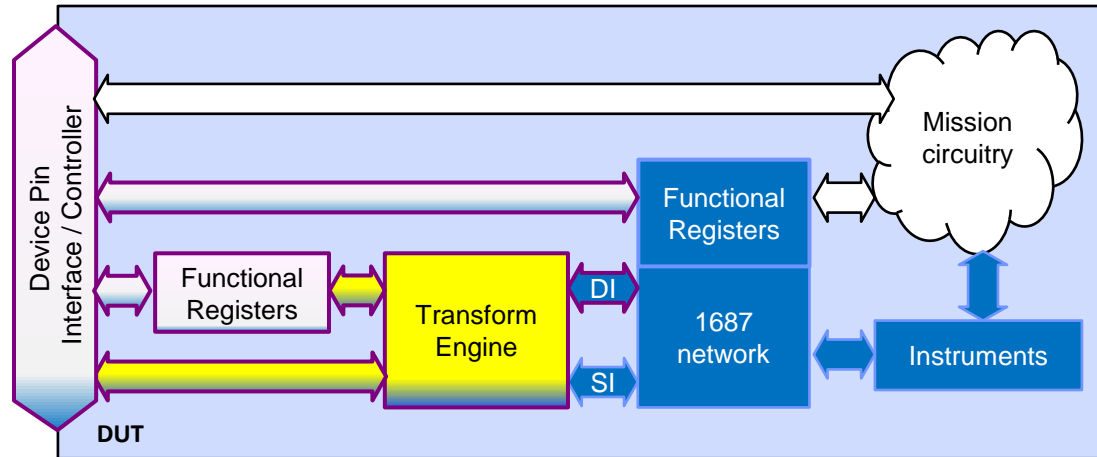
Full roster may be found at: <http://sjtag.org/members>

Member	Office	Affiliation
Ian M. McIntosh	Chair	Leonardo
Brian Erickson	Vice-Chair	JTAG Technologies
Louis Ungar	Secretary	A.T.E. Solutions
Eric Cormack	Editor	DFT Solutions
Terry Duepner		National Instruments
Heiko Ehrenberg		GOEPEL Electronics
Peter Horwood		Digital Development Consultants Ltd.
Bill Huynh		Marvell Inc.
Joel Irby		AMD
Richard Pistor		Curtiss-Wright
Jon Stewart		Dell
Bradford G. Van Treuren		VT Enterprises Consulting Services
Carl Walker		Cisco Systems

We meet weekly on Mondays at 11:00 ET.

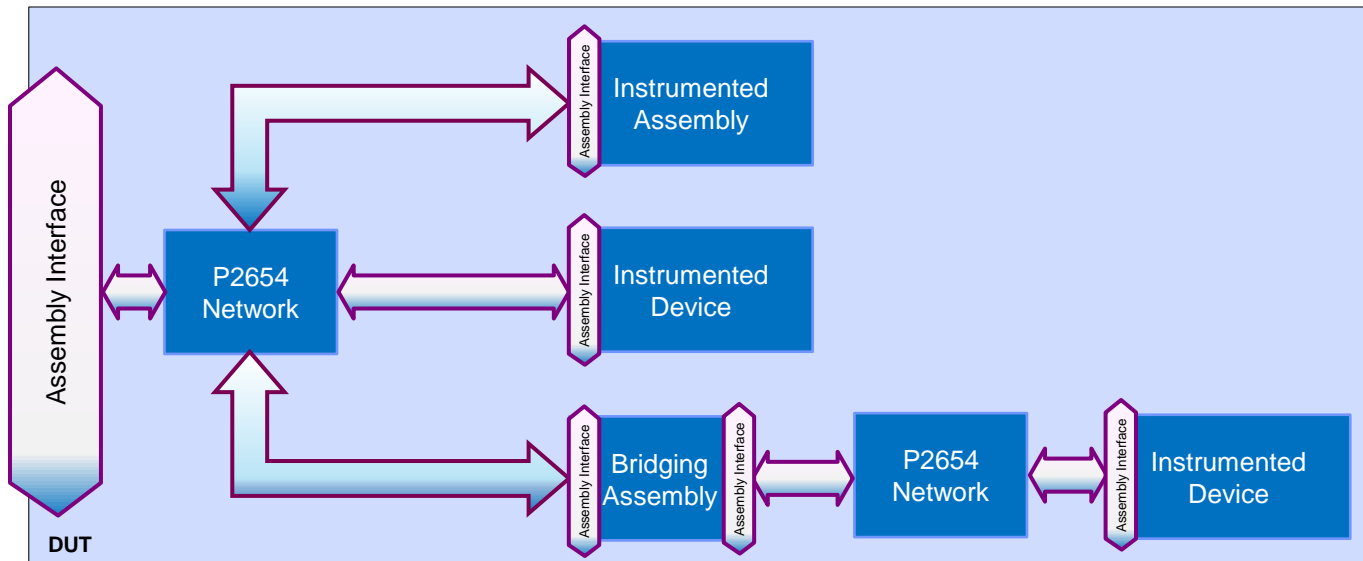
Difference Between P1687.1 and P2654

P1687.1



Specific right hand side Target (IEEE 1687 elements describable with ICL and PDL)

P2654

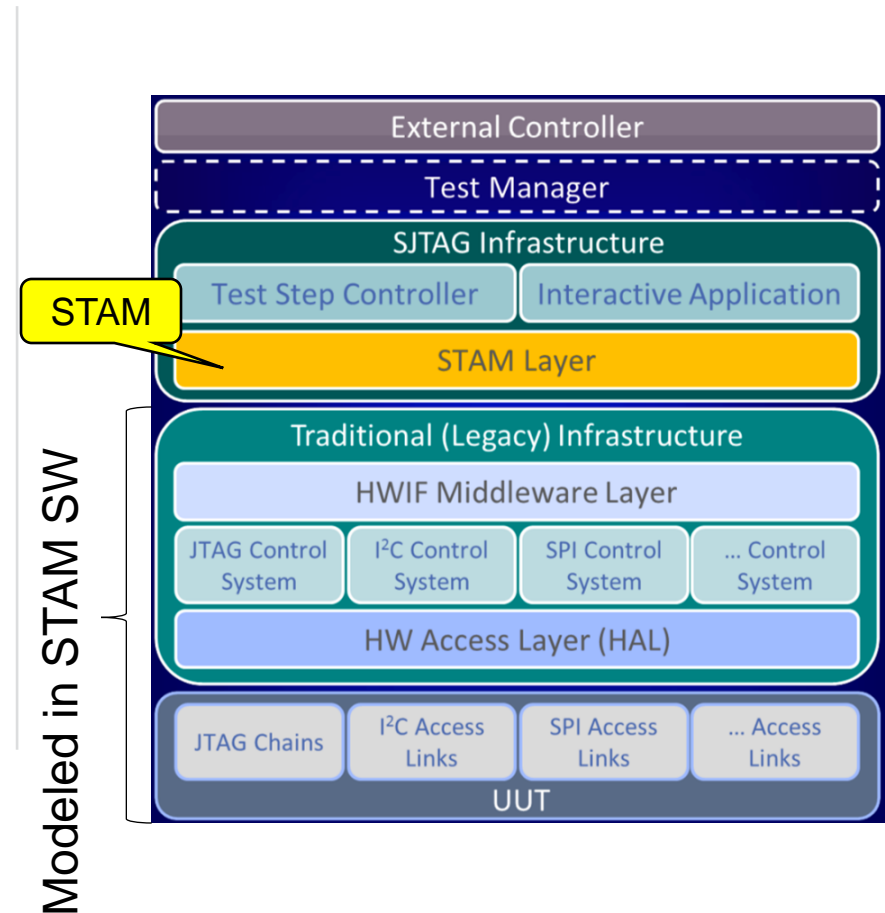


Unlimited right hand side Target varieties

P2654 Role in System JTAG (SJTAG)

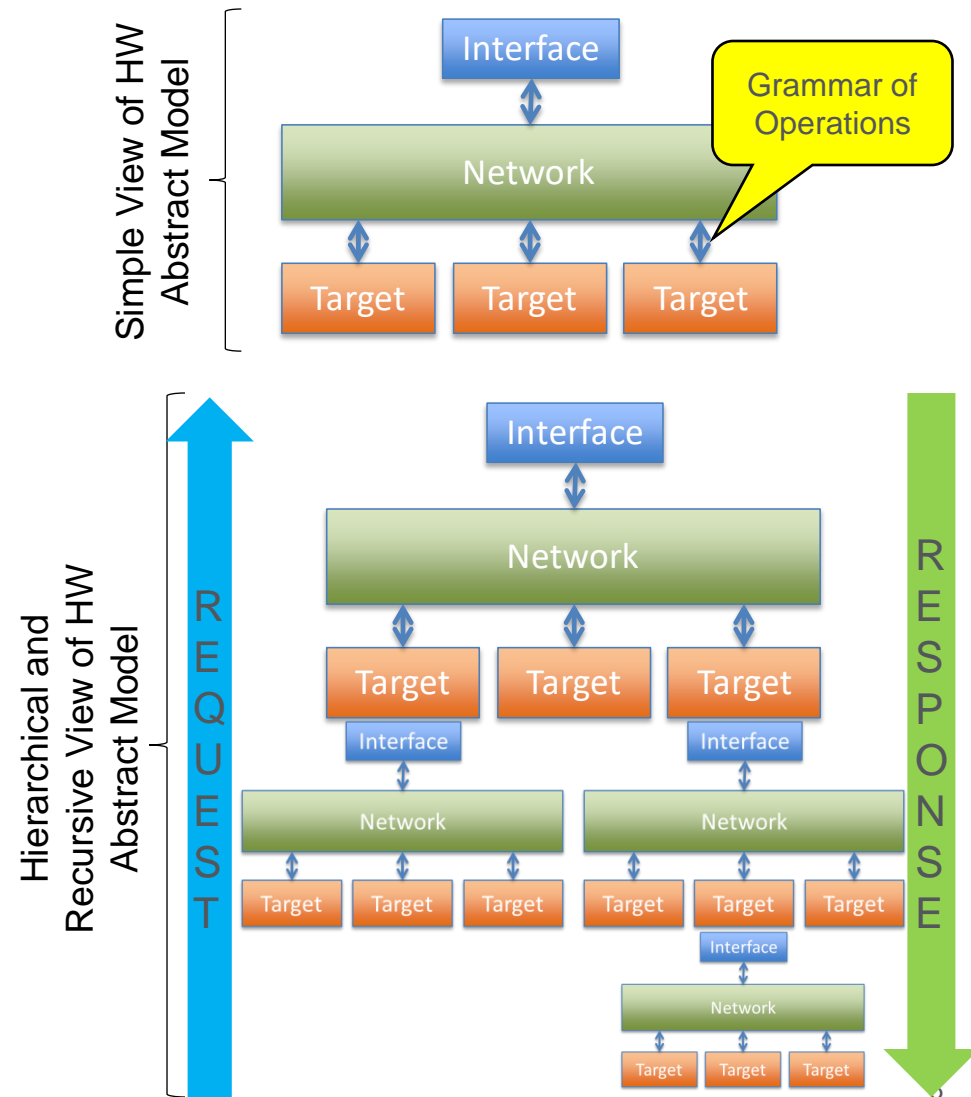
Software model of HW state change behavior at interface

- P2654 Standard for System Test Access Management (STAM) to Enable Use of Sub-System Test Capabilities at Higher Architectural Levels
- **Models the behavior** of the HW system through **hierarchical and recursive transformation** procedures for each level
- STAM SW model implements transform procedures via **Request/Response messages** between HW modeled entities



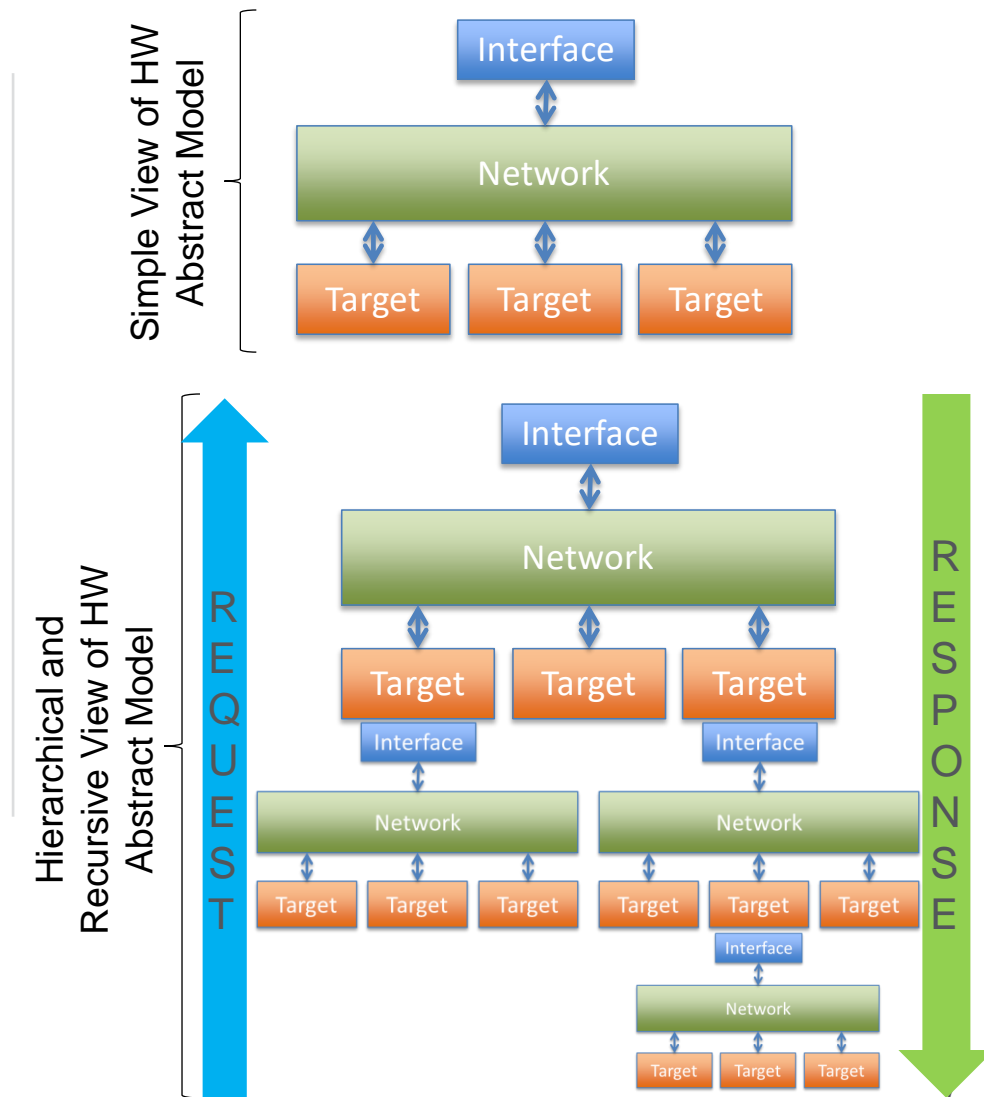
P2654 Abstract Perspective of a System

- **Network** represents the behavior of specific path logic used to communicate with a Target entity
- Each **Target** has a specific grammar of operations which may be performed by the Interface between Target and Network (e.g., I2CWrite, I2CRead)
- **Remaps Target context sensitive grammar** of interfaces through **Request/Response messages** to the higher level Network model until exposed at the Network Interface



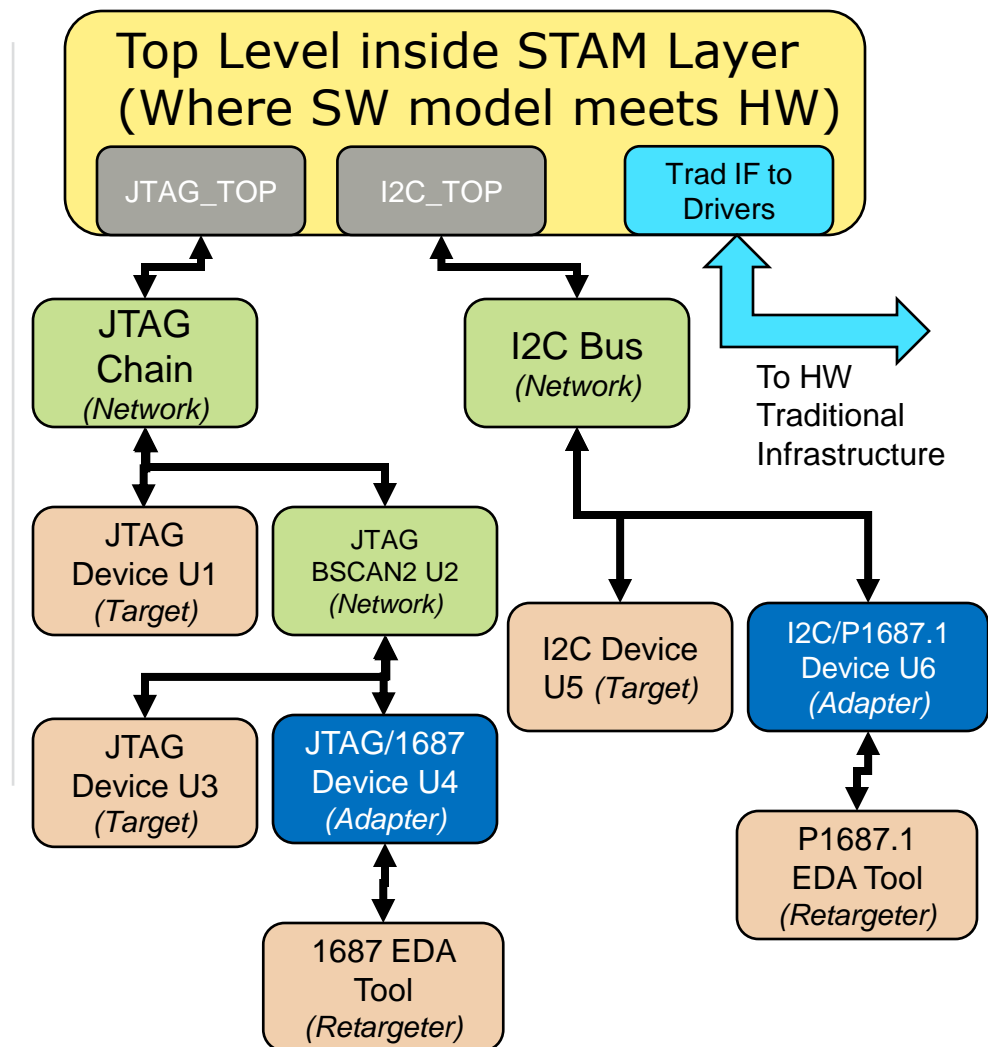
P2654 Abstract Perspective of a System

- **Top Level** Interface makes **Request to STAM Layer** to apply data to the **device driver** of the Traditional Infrastructure for that HW path
- The **data captured** from the driver is packaged up in a **response message** sent back to the Top Level SW Interface
- The message data is then **reverse transformed** into context appropriate messages to the Target tooling for processing/diagnosis



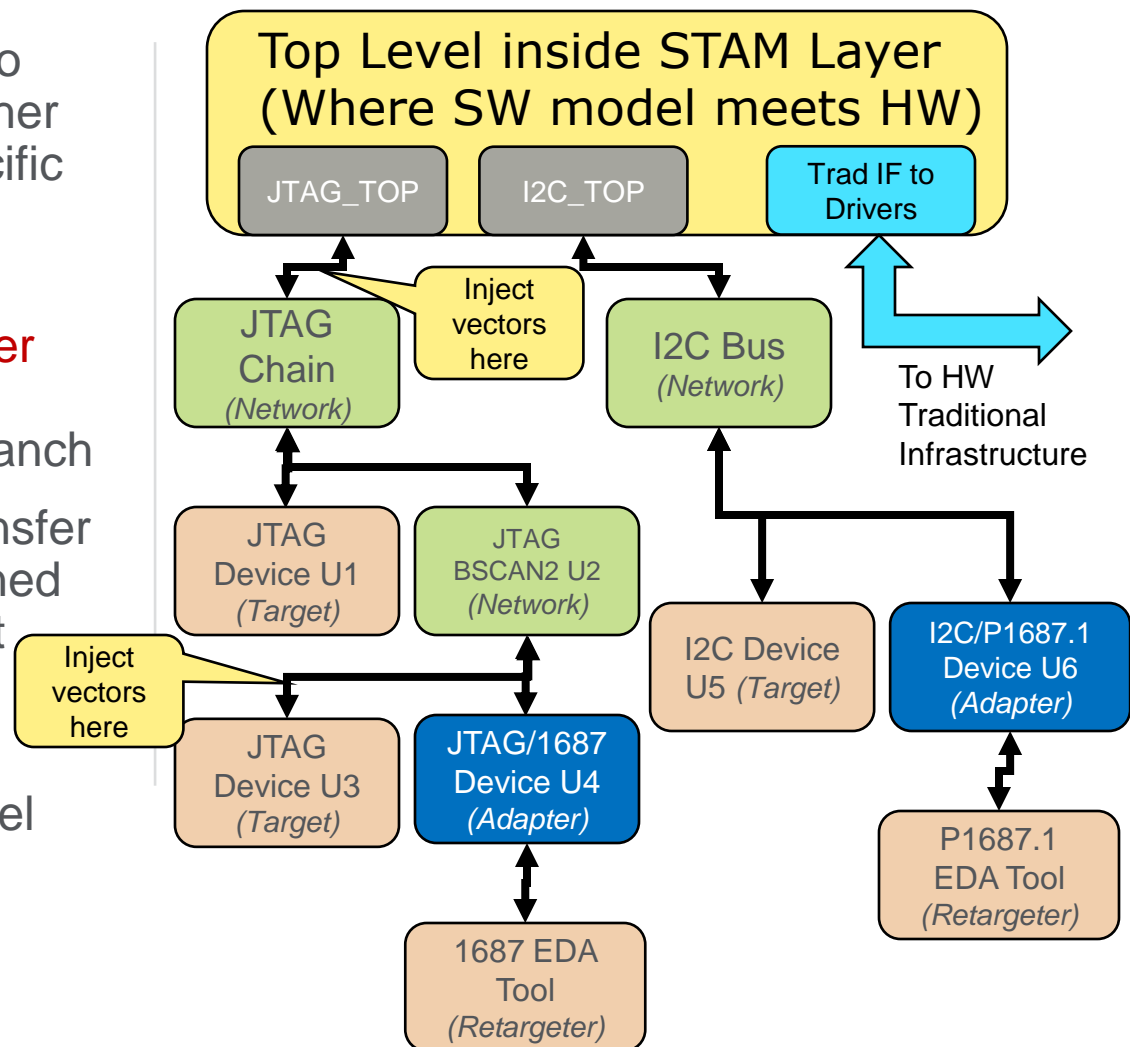
P2654 STAM Layer SW Model

- **HW architecture** is modeled as a **hierarchical tree** with STAM Layer as the root
- **Target** nodes represent Register behaviors with context appropriate interface grammar messages
- **Networks** transforms the message content in requests to a set of grammar messages to the next level returning a response
- **Adapters** transform non-P2654 compliant messages from external sources to P2654 compliant messages



P2654 STAM Layer SW Model

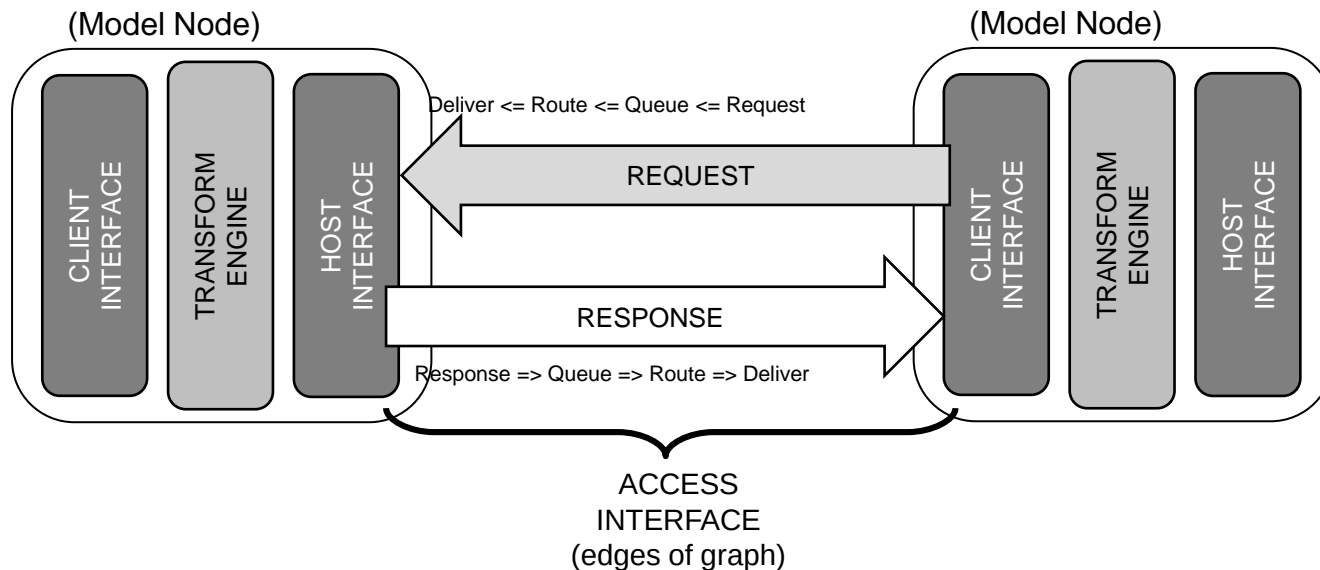
- Application SW may desire to **inject vectors** somewhere other than to a Target or to a specific Target
- Transformations will have to perform a **simplified retargeter** operation by managing path connections through their branch
- A **set of transformations** (transfer procedures) need to be defined for each request/response at each interface
- A **set of injection commands** must be defined for each level that injects vectors/data



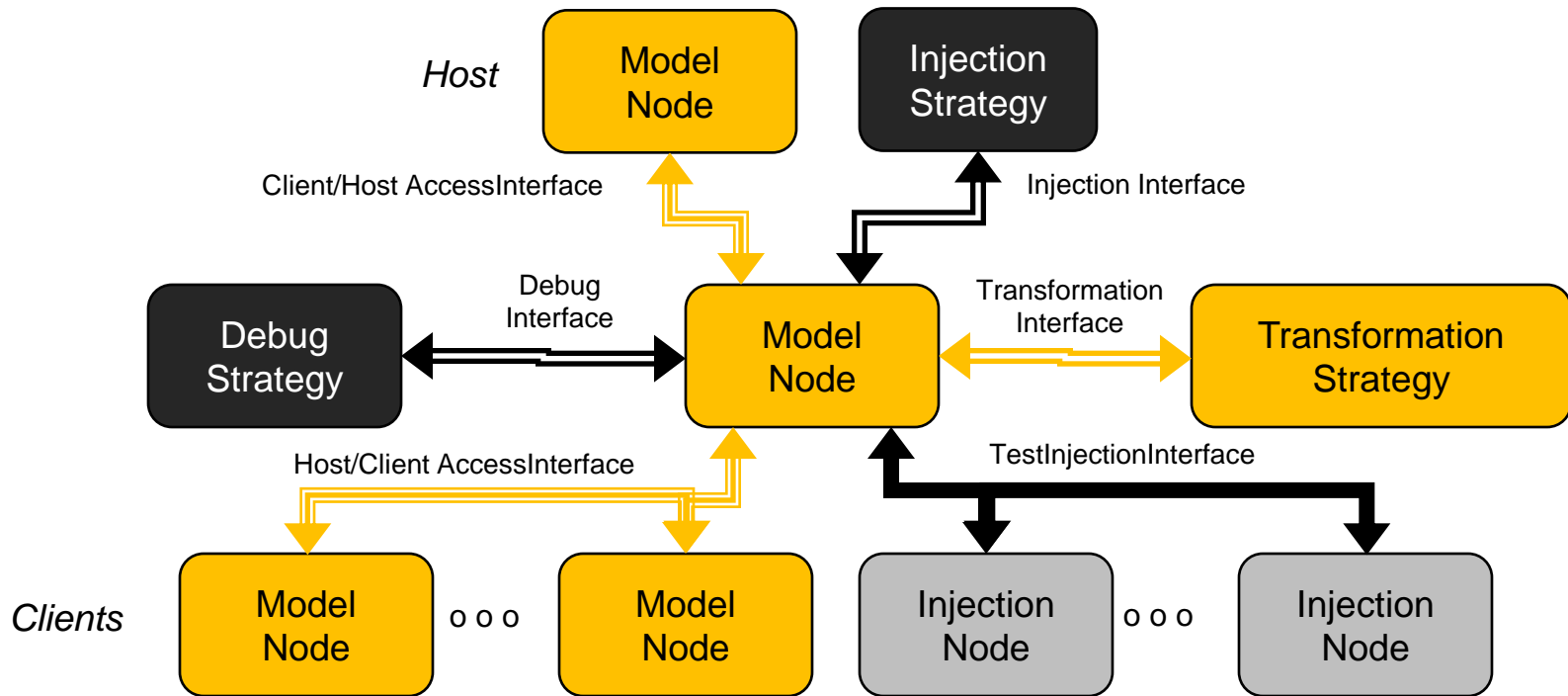
Simplified AccessInterface and Node Diagram

Interface between Model Nodes (AccessInterface)

- **Abstracts** the **connections** between Client/Host
- **Standardizes** the communications **interface**
- **Routes messages** between Client/Host and Host/Client
- Relocatable Vector Format (**RVF**) is agnostic to context
- **Buffer message groups** of same context
- **Not** transformation mechanism
- **Not** router to handler callback (Client/Host Interfaces are)
- **Not** synchronizing agent for model (transform vs. retarget method)

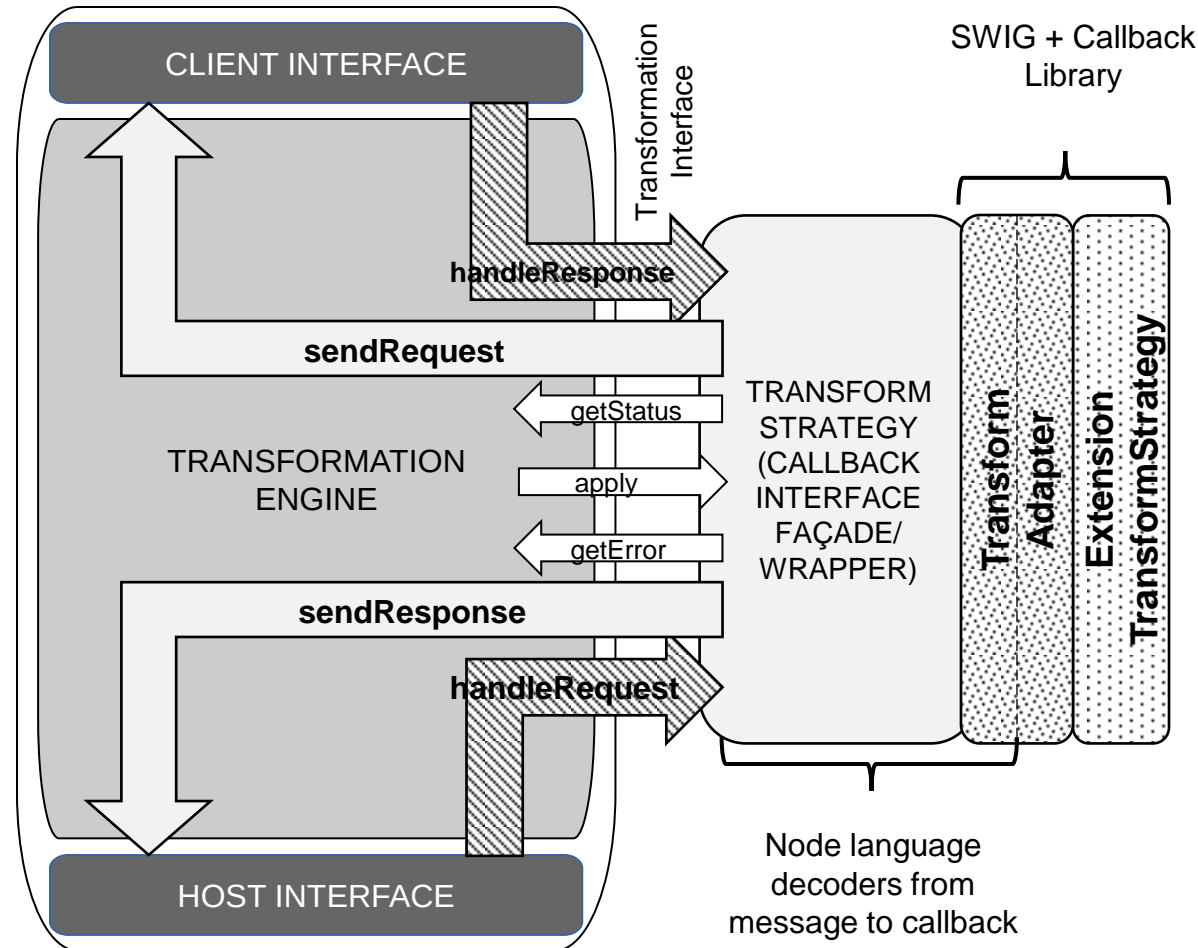


Model Node Interfaces



C/C++ Library Extension Strategy

- Leverages **protobuf** programming language to support model language and C/C++
- Direct call to **callback** functions from TransformStrategy
- Service functions from **protoc** compile in C or C++ code
- **SWIG** generated or hand crafted to adapt model code to C/C++ callbacks



Description Entities

Entity	Category	Description
ROOT	STAM Layer	A model node' that represents the top node of a tree where CONTROLLER nodes are coordinated as a single unit. The children represent a list of dissimilar sub-trees, as CONTROLLERS, coordinated for a test.
CONTROLLER	STAM Layer	A model node' that represents the top node of a tree where event messages are folded into hardware device driver calls. The children represent a list of similar sub-trees coordinated for a test and controlled by the CONTROLLER.
CHAIN	Network	A hierarchical node' that represents a chained hierarchy. This node models a structure of multiple segments as a hierarchy of sibling children nodes of the same order. Cardinality order of the modules in the list is important. Modules are specified with the children keyword.
LINKER	Network	A hierarchical node' that represents a selectable chained hierarchy. This node models a structure of multiple segments as a hierarchy of sibling children nodes that may be present or missing from the path. Order of the modules in the list is important. Modules are specified with the children keyword.
CUSTOM	Network/ Target	A model node' that represents the customizable node of a tree where none of the primitive node types describe the behavior of the node. The children may represent a list of dissimilar sub-trees coordinated for a test.
INSTANCE	Target	A leaf node' that represents the instantiation of a module/branch at the point of insertion (as for any other child nodes).
MODELPOINT	Target	A leaf node' that represents a translator between the software model and external tools or protocol formats. This node references the strategy to be used to bridge the tooling with the model tree.
REGISTER	Target	A leaf node' that represents the 1687 or JTAG TDR register modeling node.

Protobuf RVF

Encapsulation Model Example

RVF.proto

```
syntax = "proto3";
package RVF;

message RVFMessage {
  uint32 UID = 1;
  enum RVFType {
    ERROR = 0;
    STATUS = 1;
    REQUEST = 2;
    RESPONSE = 3;
  }
  RVFType rvf_type = 2;
  string metaname = 3;
  repeated bytes serialized= 4;
}
```

- AccessInterface messages defined as separate proto buffer messages in .proto file for entity
- Proto message name as metaname field used to select proper callback wrapped in RVFMessage wrapper
- Message treated as raw data encapsulated as a “bytes” type
- RVFMessage as interface to Request and Response handlers

Protobuf RVF

Encapsulation Model Example

SCAN.proto

```
syntax = "proto3";  
import "IntBV.proto";  
package RVF;
```

```
Message SU{  
  uint32 UID = 1;  
  IntBV si_vector = 2;  
}
```

```
Message CS{  
  uint32 UID = 1;  
  IntBV safe_vector = 2;  
  IntBV so_vector = 3;  
}
```

```
Message CSU{  
  uint32 UID = 1;  
  IntBV si_vector = 2;  
  IntBV so_vector = 3;  
}
```

- Specialized messages for context around Model Node
- Required fields used by software administration of message routing (e.g., UID, command)
- Context messages serialized/deserialized to pack into carrier message

Q&A