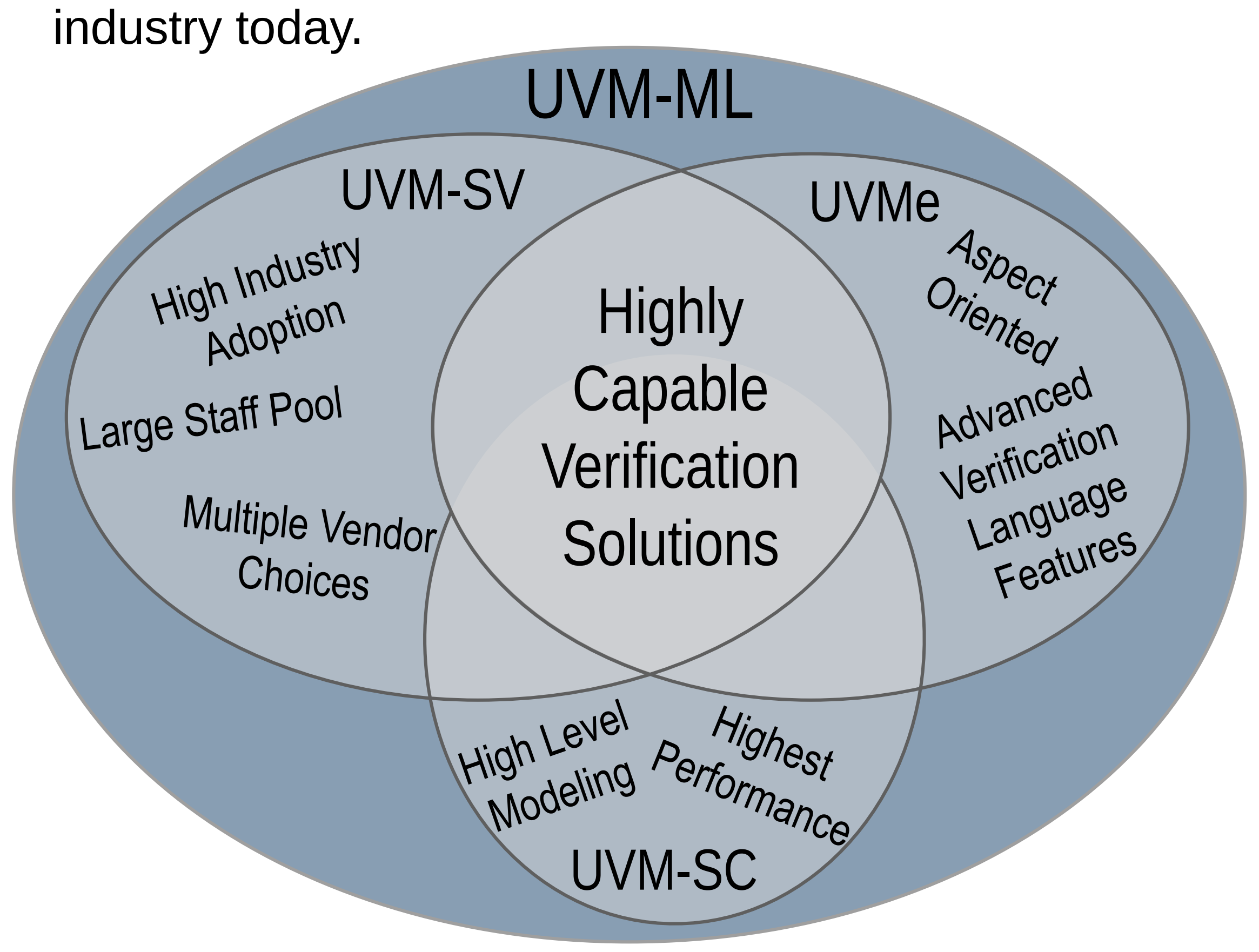
# Is Specman Still Relevant? Using UVM-ML to Take Advantage of Multiple Verification Languages

Hewlett Packard Enterprise

**Timothy Pertuit, David Lacey, Doug Gibson**

## Why UVM-ML

From SystemVerilog, to Specman/e, to SystemC companies need the flexibility to choose the right language for the right problem. They need to reuse existing code and tests from a variety of sources from a long history of verification. UVM-ML provides a mechanism for companies to utilize the best of all technologies, deploying them based on the needs of the program, the cost of the tools, the target audience, and the many other factors that drive our industry today.



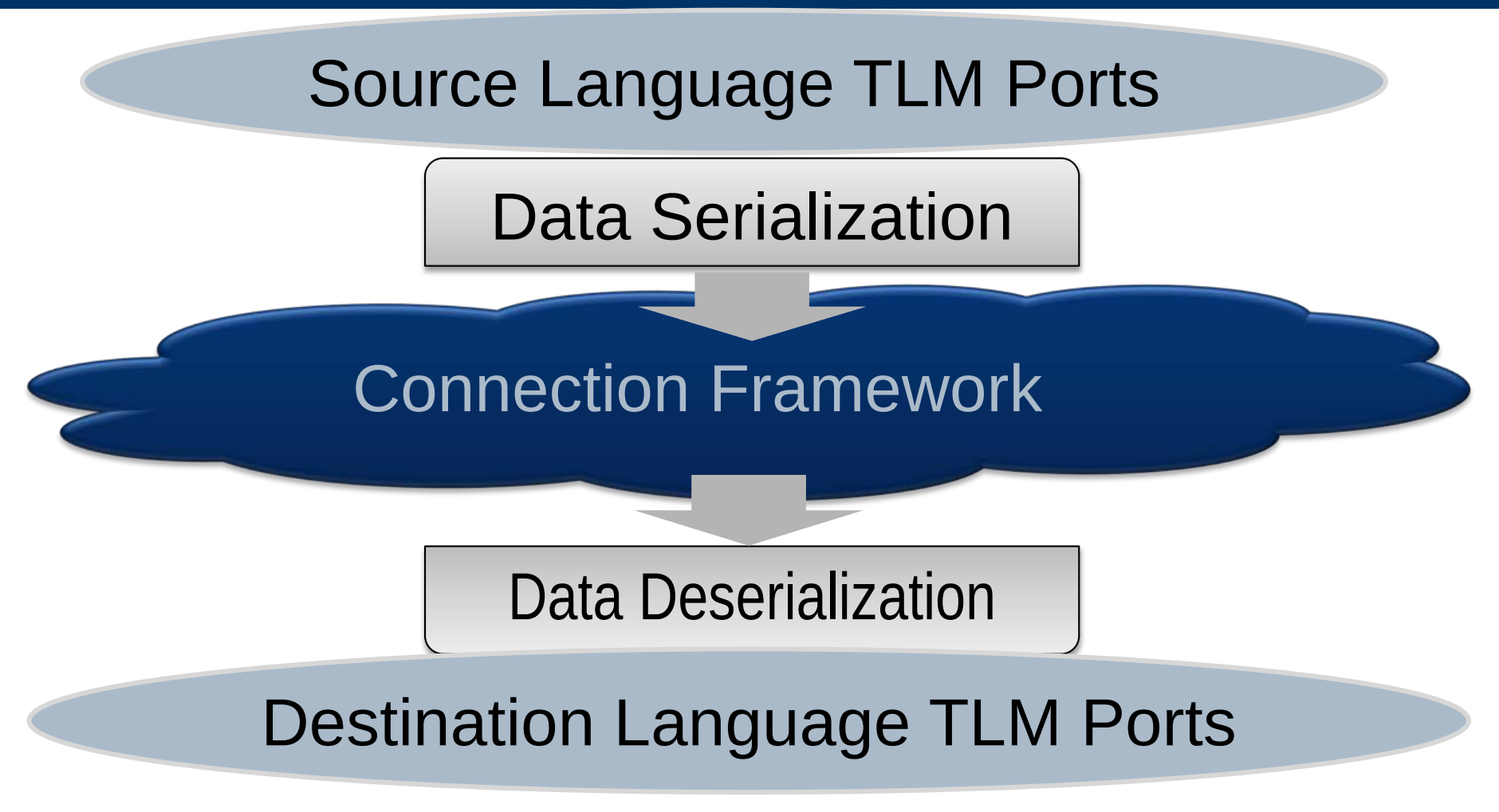
SystemVerilog and UVM-SV have nearly become the de facto standard for the Verification Industry. It offers a wide variety of benefits due to it's widespread use. Specman/e has a long history going back through eRM that helped to shape UVM. SystemC can offer best in class performance and excels at high level modeling. Each of these languages can be used to build highly capable verification solutions across a wide variety of use cases and types of designs.

UVM itself is a standard that aims to improve the interoperability of VIP along with reducing the cost of repurchasing or rewriting IP for each new project. With multiple languages in use an additional framework is needed to reach the interoperability goal. UVM-ML is an open source framework that aims to bridge these languages to provide an even greater level of reuse.

## UVM-ML Basics

The foundation of the UVM-ML framework is in providing a mechanism for creating communication channels between various languages. This is accomplished primarily through the use of TLM ports. UVM-ML provides the backend components necessary for creating connections between TLM ports of different languages

## UVM-ML Basics (cont)

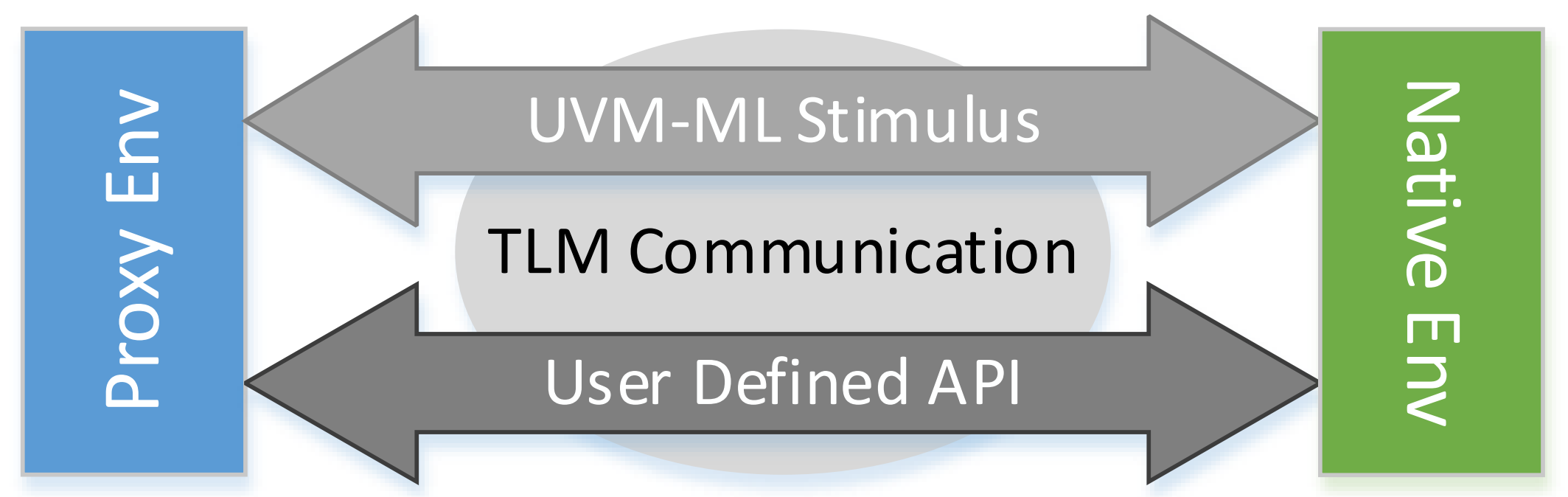


UVM-ML supports TLM 1.0 and 2.0 to enable support for a wide variety of applications and use cases

## UVM-ML Stimulus

Integrating stimulus mechanisms across multiple languages is required in order to provide a full featured UVM-ML environment. UVM-ML provides a set of TLM ports and implementations to facilitate the passing of data between the sequencers of different languages. These ports help to define a "proxy" sequencer that can be used in the foreign language and interact with the native language sequencer of the UVM component being used to drive stimulus.

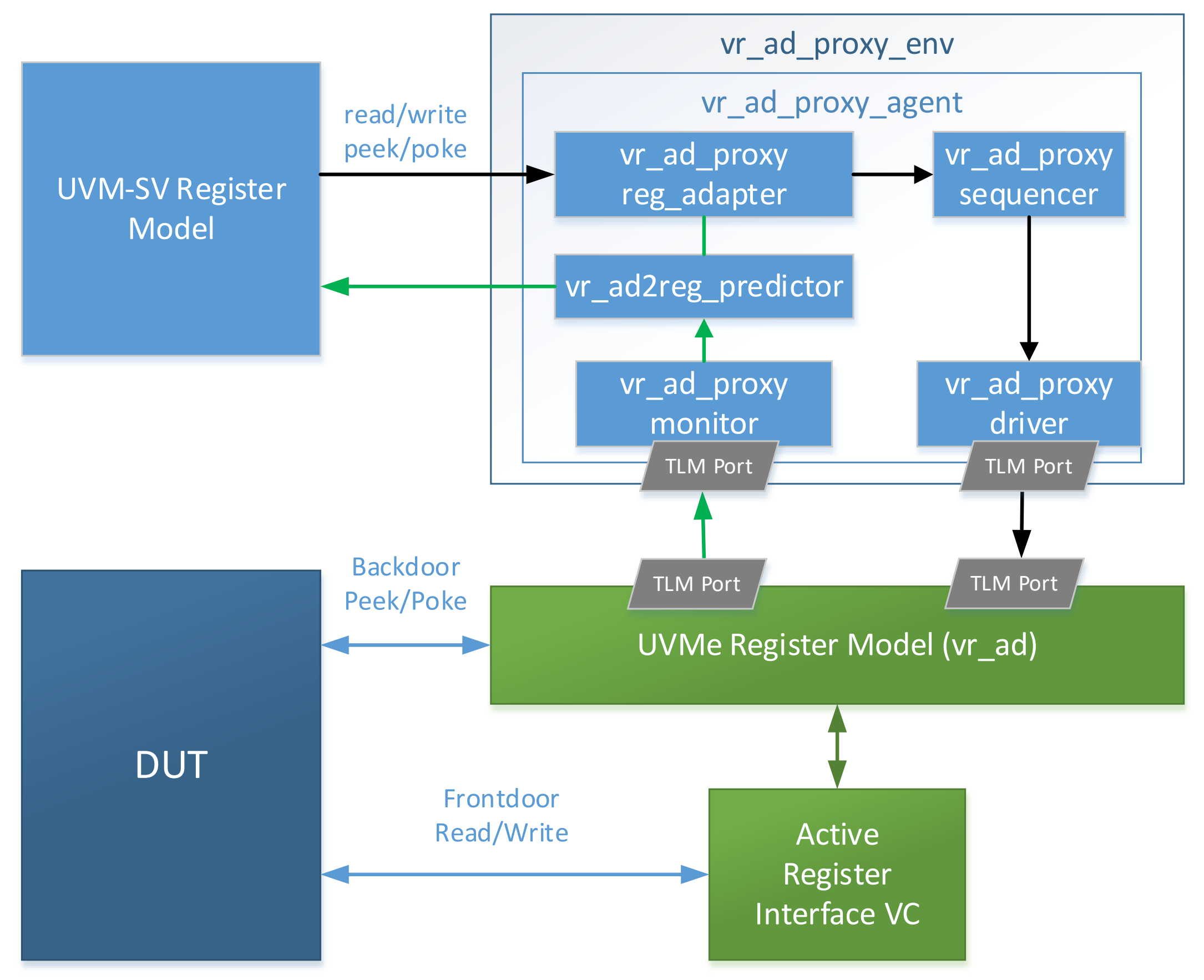
In sophisticated UVM components, it is not uncommon for the sequencer to provide functionality to its sequences to aid in creating interesting and complex stimulus. With UVM-ML, the test writer may not have direct access to the native sequencer that the sequences will be run on. It takes some thought and care to develop an API that will be exported to the foreign language for use by the remote sequences.



While UVM-ML provides the basics for supporting stimulus across the language, one area that it did not natively support was the handling of deferred responses. Many protocols support a deferred response capability where many requests may be issued at a time and responses come back much later in time, potentially out of order with respect to the original requests. The paper details the implementation of a mechanism for proxying the responses from a UVM-SV's "get_response()" API through back to a UVMe sequencer to enable more complex sequence and test writing.

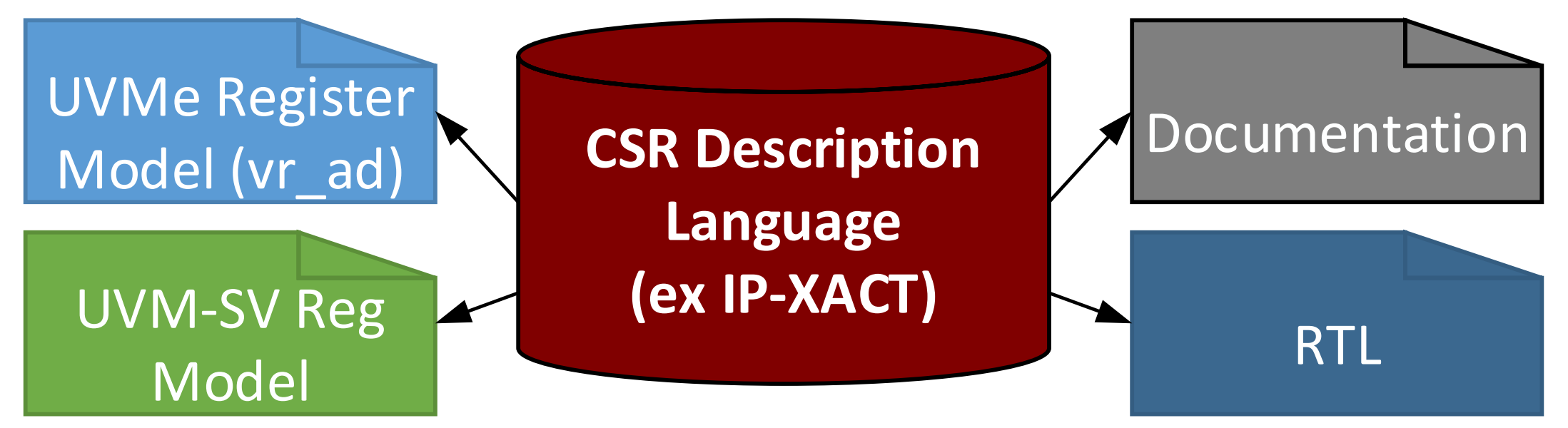
## UVM-ML Register Model

Both UVM-SV and UVM-e include register models as part of their methodology, but the two implementations are distinct and separate. UVM-e uses a model called "vr_ad" whereas UVM-SV has the UVM Register Layer implementation. The base UVM-ML OA implementation includes a TLM interface providing access to a vr_ad based register model from UVM-SV. This consists of a basic interfaces for reading, writing, peeking, and poking registers from UVM-SV through the vr_ad model, however it does not include a mode that provides the same seamless type of integration as the stimulus aspects do. To make the model more usable from both languages, SDL implemented a proxy mechanism to bridge between the UVM-SV register layer model and the UVM-e vr_ad model.



The implementation depicted in the above figure consists of a proxy UVM-SV VC and adapters for interfacing with the UVM Register Layer. Like the stimulus implementation, all accesses (frontdoor and backdoor) are proxied to vr_ad for interacting with the DUT. Register changes observed by vr_ad are also send back through as if monitored natively in UVM-SV. This allows sequences and checkers to be written in either language while keeping all the register states in sync.

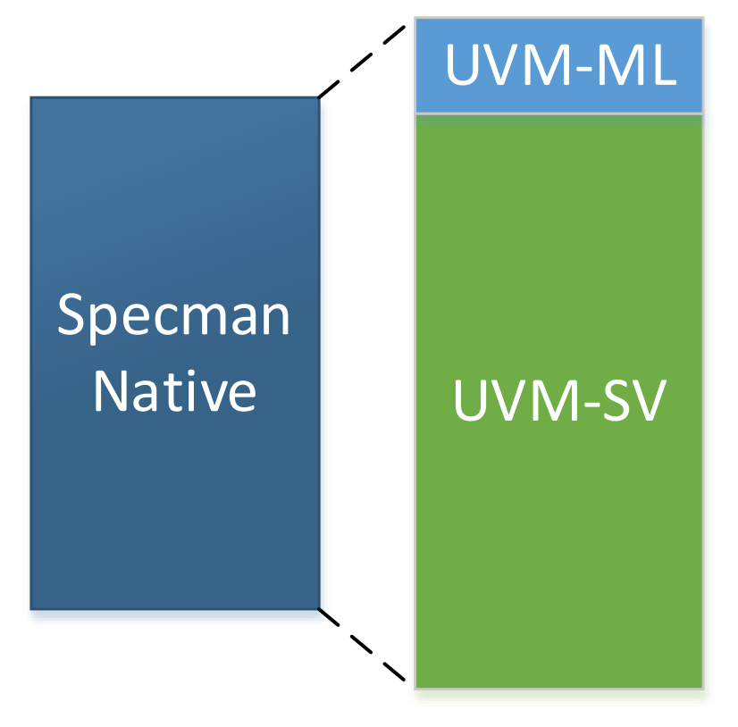
This type of setup requires duplicate register models in UVM-SV and Specman. To solve this SDL extended its existing CSR automation infrastructure to support both vr_ad and UVM-SV register model generation from a common CSR source description.


UVMe Register Model (vr_ad)
UVM-SV Reg Model
CSR Description Language (ex IP-XACT)
Documentation
RTL

## Porting UVMe to UVM-ML

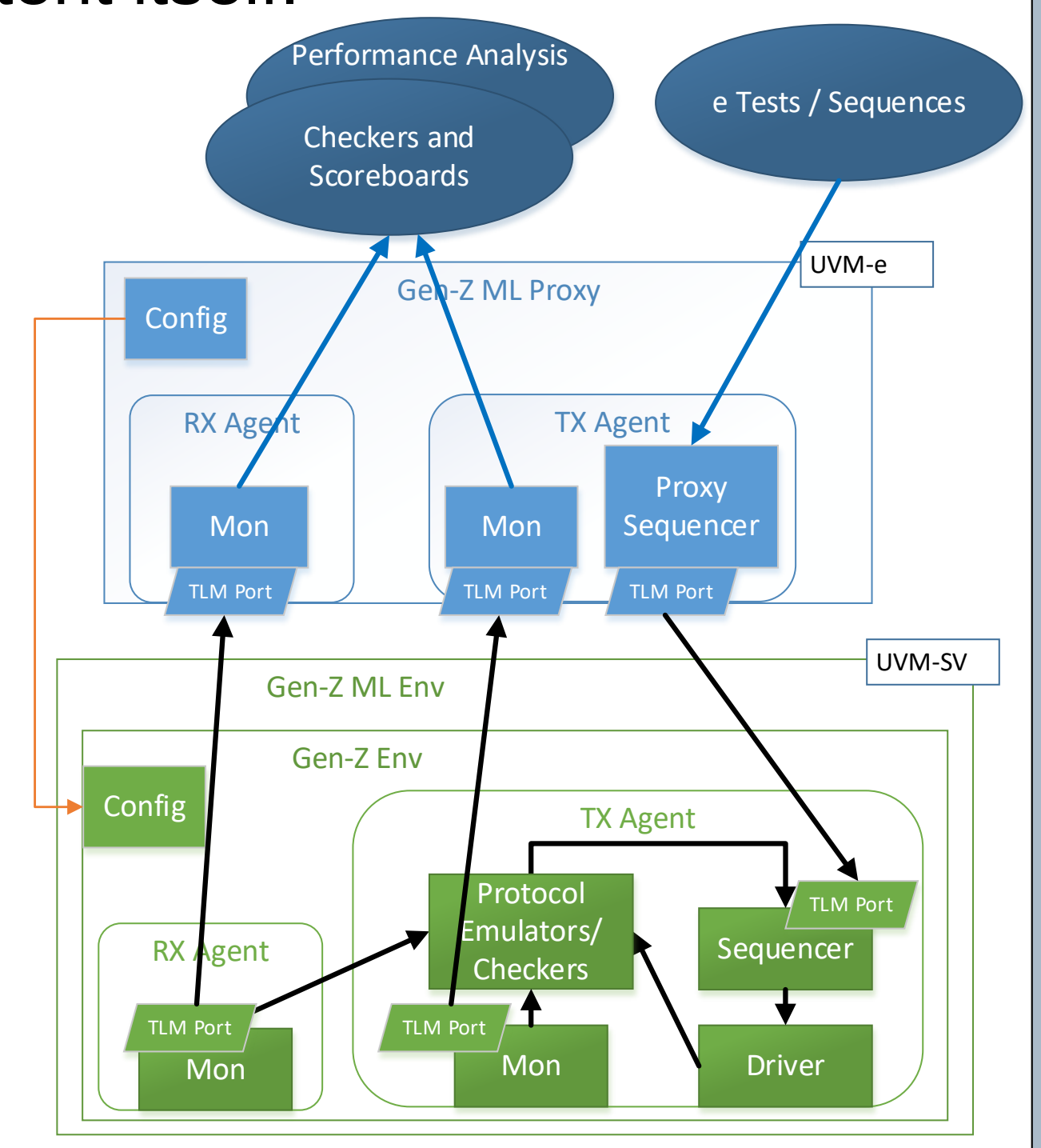
The paper discusses two interface VCs that were converted from UVM-e to UVM-SV including support for UVM-ML. These two VCs were ported because the usage of them needed to shift from internal use only VIP to ones that would be shared across multiple companies. UVM-ML made it possible to transition these interface VCs to UVM-SV with a minimum of disruption to existing test benches and tests spanning four IP block environments and three chip environments.

The Gen-Z VC pictured below was directly ported, nearly line by line, from UVM-e to UVM-SV. In most cases the UVM-e/aspect oriented functionality was able to be refactored to fit within the limits of SystemVerilog. The UVM-e implementation of the Gen-Z VC was approximately 4000 lines while the UVM-SV content was 4600 lines. The UVM-ML (e and SV) content accounted for about 600 additional lines. Through this conversion, we grew the total lines of code by about 25% over all, but half of that growth was in the UVM-ML content itself.



25% Total Code Growth

½ of Total Code Growth in UVM-ML Content



## Conclusion

SDL has been using UVM-ML functionality for multiple years across numerous projects and designs. Through multiple projects, SDL has been able to take advantage of many of the advanced testing features available in Specman/e while utilizing a variety of UVM-SV content from internally developed VCs to externally purchased Verification Ips. Through the capabilities already present in UVM-ML as well as those added by SDL itself, the larger integration and test teams have been insulated from most of the multi-language aspects.
So is Specman still relevant? We believe yes! It is still relevant to our industry as it has many advanced features that are not available in other languages. However UVM-ML is a necessary component of the solution as it is clear that no one language has yet filled all of the needs of the verification industry.