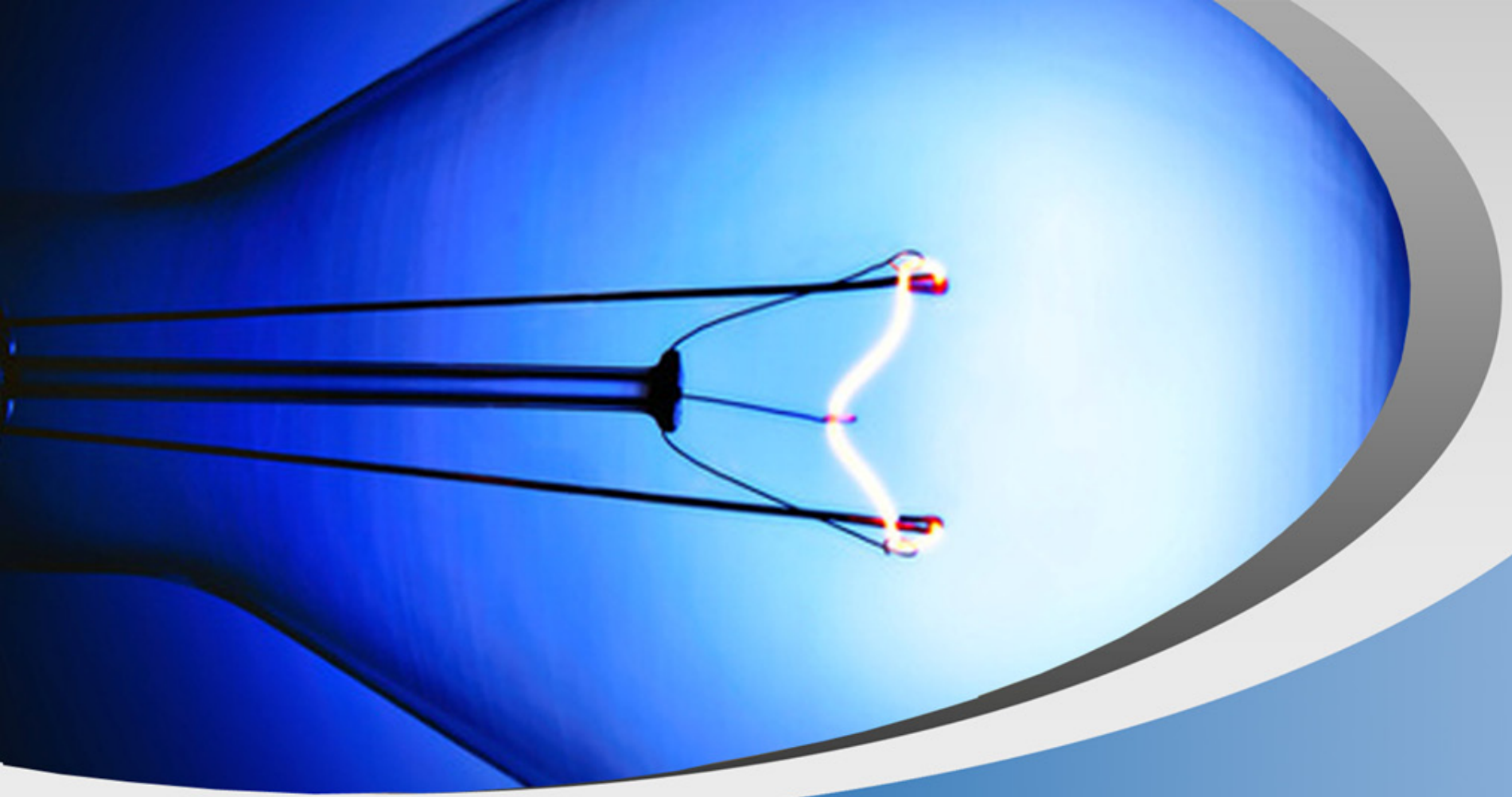


Accellera Systems Initiative MLWG – Introduction

MLWG – Warren Stapleton

Jun 3, 2013

accellera SYSTEMS INITIATIVE

MLWG – Agenda

- Introduction
- Problem Statement
- Example Use Cases
- High Level Architectural Requirements
- Envisioned Architecture
- Timeline

INTRODUCTION AND CHARTER

accellera
SYSTEMS INITIATIVE

MLWG – Introduction (1)

- Chair: Warren Stapleton (AMD)
- Vice Chair: Faris Khundakjie (Intel)
- Initially tasked by the Accellera Board to collect user-level requirements for a mixed-language solution, met weekly since May, 2012.
- Initial user-level requirements list was contributed to by:

Company	Representative
AMD	Bryan Sniderman
Freescale	Hillel Miller
Intel	Faris Khundakjie
NXP	Martin Barnasconi
Qualcomm	Kartik Talsania
ST Micro	Antoine Perrin

MLWG – Introduction (2)

- Sanctioned to become a full-fledged Accellera working group in Apr-2013.
- Now with ~70 interested individuals (about ½ are members of the WG) from 24 different companies. Interest is growing!

AMD	Fraunhofer Institute	Qualcomm
Atrenta, Inc.	Freescale Semi.	Semifore, Inc.
Broadcom	Intel Corp.	ST Microelectronics
Cadence	Jasper	Synopsys
Cisco	Mentor	Texas Instruments
Cypress Semiconductor	NVidia	Verilab
Doulos	NXP	Xilinx, Inc.
Ericsson	OFFIS	+non-member company

MLWG - Charter

The mission of the MLWG is to create a standard and functional reference for interoperability of multi-language verification environments and components.

- The MLWG will review requirements and develop an open source proof-of-concept library for creating a standards-based approach for combining verification environments developed in different languages/frameworks.
- In addition, the group will look at ways to enable the introduction of UVM (Universal Verification Methodology) concepts in other environments and languages that come from legacy projects or developed with other frameworks for beneficial reasons.

MLWG - SCOPE

- This is not an effort to develop yet another verification framework/methodology that attempts to solve all problems, but
  - seeking an improved way to integrate existing and new methodologies
  - help align Accellera standards with each other.

accellera SYSTEMS INITIATIVE

MLWG – Why Accellera?

- Most credible, far-reaching, standards-driven verification body.
- Viewed as an incremental path to transition more of ML user code to UVM SV as users find convenient integration and can afford more time to learn UVM in their environment.
- More predictable approach for achieving ML compliance across vendors and user testbenches.
- MLWG goal is to provide a solution which works for any language but is not requesting proven compliance with anything other than standard languages used: e, SV, SC, and C++

PROBLEM STATEMENT

accellera
SYSTEMS INITIATIVE

MLWG – Problem Statement

- Verification engineers encounter multi-language (ML) integration problems on a regular basis.
- The ML integration problem is not limited to SV⇔SC but can include models and stimulus in other languages such as VHDL, matlab, e, scripts, C/C++.
- Many users share the same use cases and most (re)invest redundant efforts with non-standard internal or vendor-specific solutions.
- The problem grows when mixing technologies from different vendors for their unique benefits.

MLWG – Example Use Cases

- **Firmware stimulus engines**
  - Core or embedded controller with own instruction set where UVM style sequences do not make sense. DUT fetches from initialized caches or other memory devices.
  - Tools (in some cases lots of legacy) developed in languages other than standard SV/SC for test generation and initialization.
- **Reuse for simulation acceleration**
  - SC reference and functional model reuse in hardware accelerated environments where SV testbench is not present.

MLWG – Example Use Cases

- **SC model verification with specialized stimulus**
  - Specialized stimulus developed for reference model verification before RTL ready with stimulus from different tools and languages.
- **Legacy code reuse**
  - Legacy testbench code and utilities written in different languages including but not limited to C++, e, Python, and Perl
  - Desire for reuse is not for lack of desire to move to SV but lack of general purpose and community-developed rich libraries found in areas like C++, Python and Perl

MLWG – Example Problems

- **Incompatibility with wrapped VCs/VIPs**
  - Wrapped VIPs to present UVM-SV API may not work with native UVM-SV VIP. Better to have integration with VIPs in their native form (nor wrappers).
- **AMS Verification with configurable VIP-DUT interfaces**
  - Depending on the type of test, the VIP-DUT interface(s) should be either support electrical, real-value, or logic signal types. Current approaches based on connect modules, collars/gaskets/wrappers or other dedicated “glue logic” are not configurable and often incompatible, and hamper reusability.

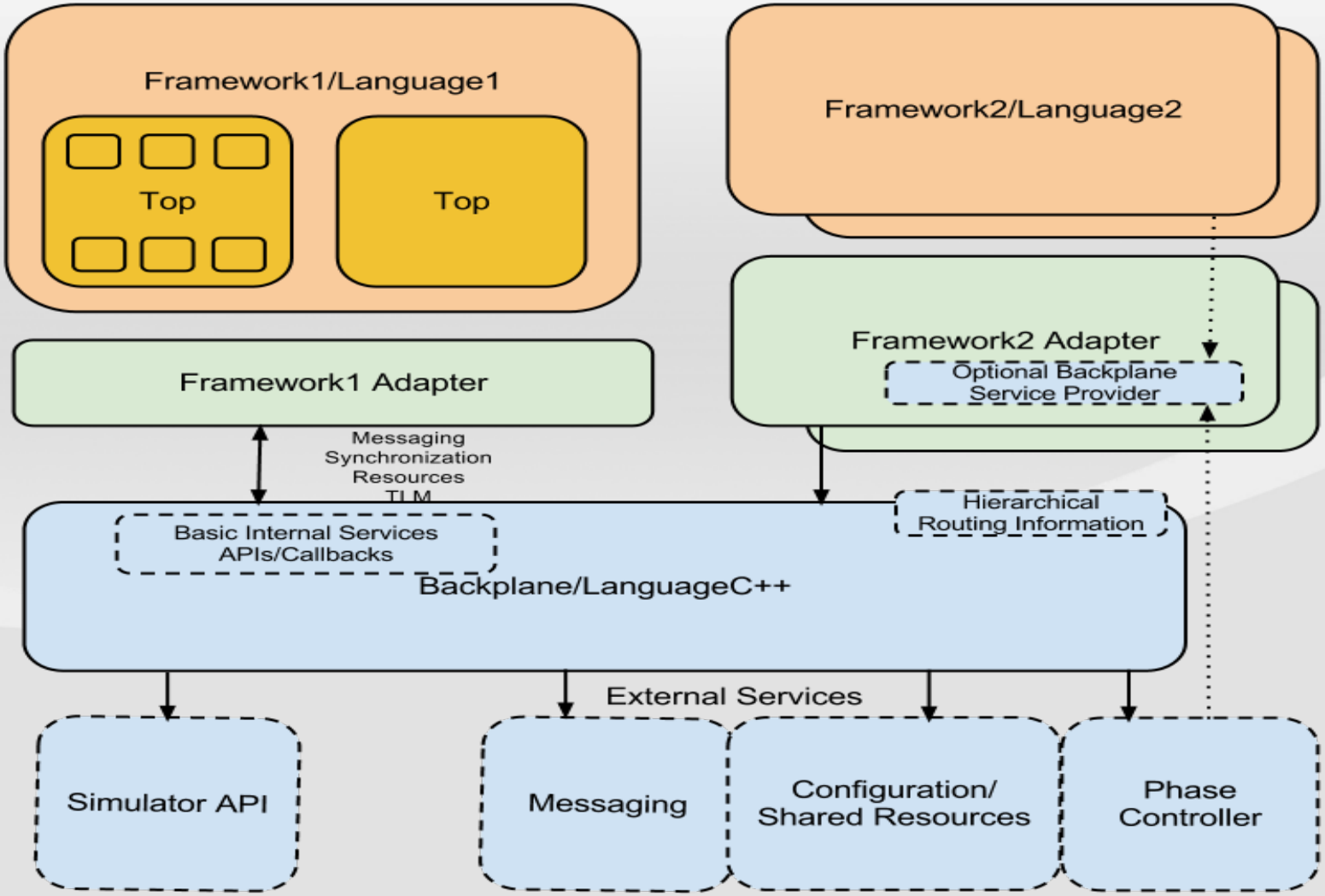
ARCHITECTURE

accellera
SYSTEMS INITIATIVE

MLWG – Architectural Requirements

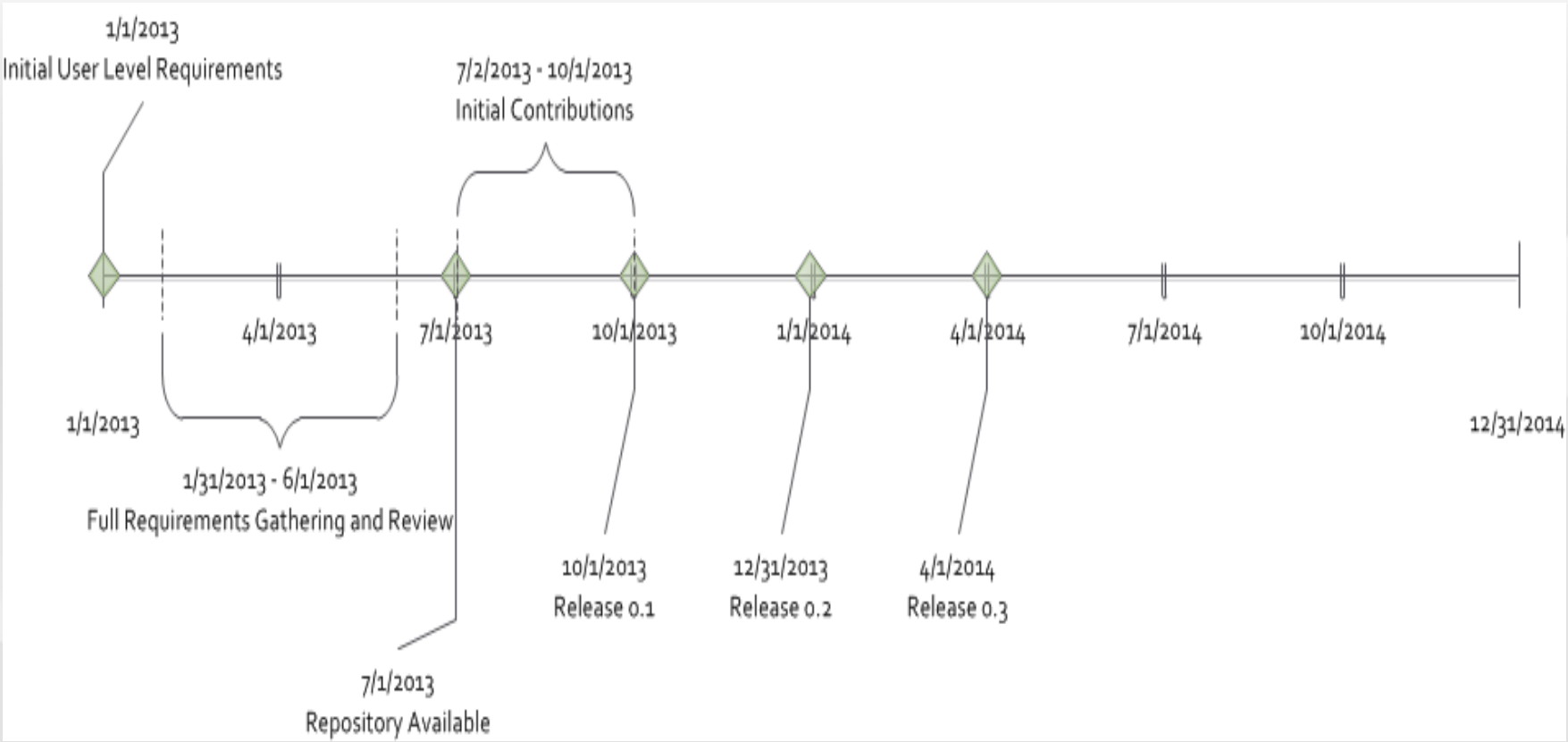
- Goals
  - Allow multiple different verification frameworks to interoperate by providing
    - communication between the frameworks
    - coordination with phasing/synchronization primitives
    - and making common methodology facilities available across the entire system.
  - Support the integration of external VIP
  - Support the integration of external C libraries
- Tenets
  - Deliver API in form natural to the topic/specific requirements the API focuses on
  - Don't impose unnatural use-model for existing frameworks
  - Define ML methodology in terms of UVM user guide and subsequent updates
  - Support a superset of the UVM functionality (i.e. user should be able to provide arguments to the API with the same fidelity as the matching UVM functionality).
    - UVM adapter should be able to provide support for key UVM facilities
  - Deliver debug state and access methods for data that flows through the system.
  - Should encompass emulation and hardware assisted verification environments

MLWG – Envisioned Architecture



accellera SYSTEMS INITIATIVE

Potential Timeline



accellera SYSTEMS INITIATIVE

Conclusion

- We have a solid set of high-level requirements for a system that will simplify the integration of verification elements developed in different languages.
- Confirmation that this is a good problem to solve based on the level of interest in the working group.
- A growing core group of companies willing to contribute to the implementation of the solution.
- Next steps
  - Refine the contribution process
  - Pseudo API development
  - Work towards early concept release