2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

Novel Method To Speed-Up UVM Testbench Development

Nimay Shah, Prashant Ravindra,
Barry Briscoe, Miguel Castillo

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

accellera
SYSTEMS INITIATIVE

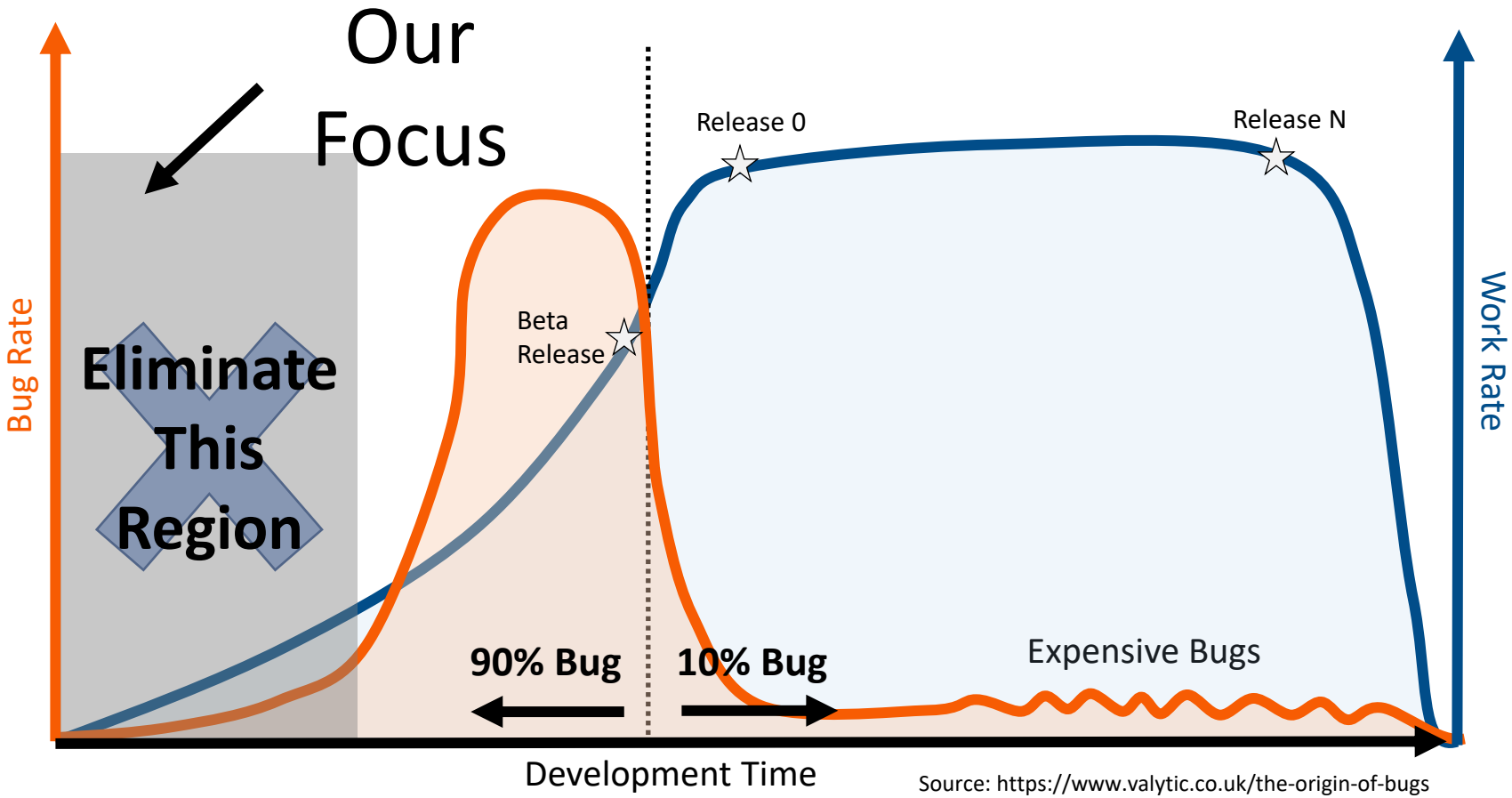
Agenda

- Verification Complexity
- Development challenges
- UVM TB Development Flow
  - Typical Flow
  - Existing Flow
  - New Flow (Metadata)
- Results & Summary

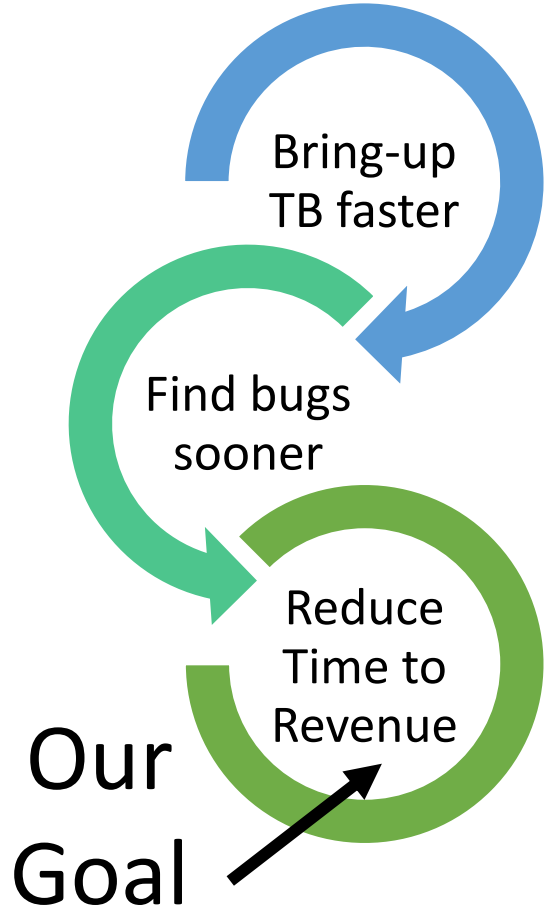
Verification Complexity

- Sophisticated designs are the need of the hour!
  - First-pass bug-free silicon is crucial to meet stringent TTR
- Demand for “Effective” and “Efficient” verification methods to signoff (near) bug-free spec-compliant designs!
  - This calls for highly complex, yet configurable UVM Testbenches

Catching Bugs Early….

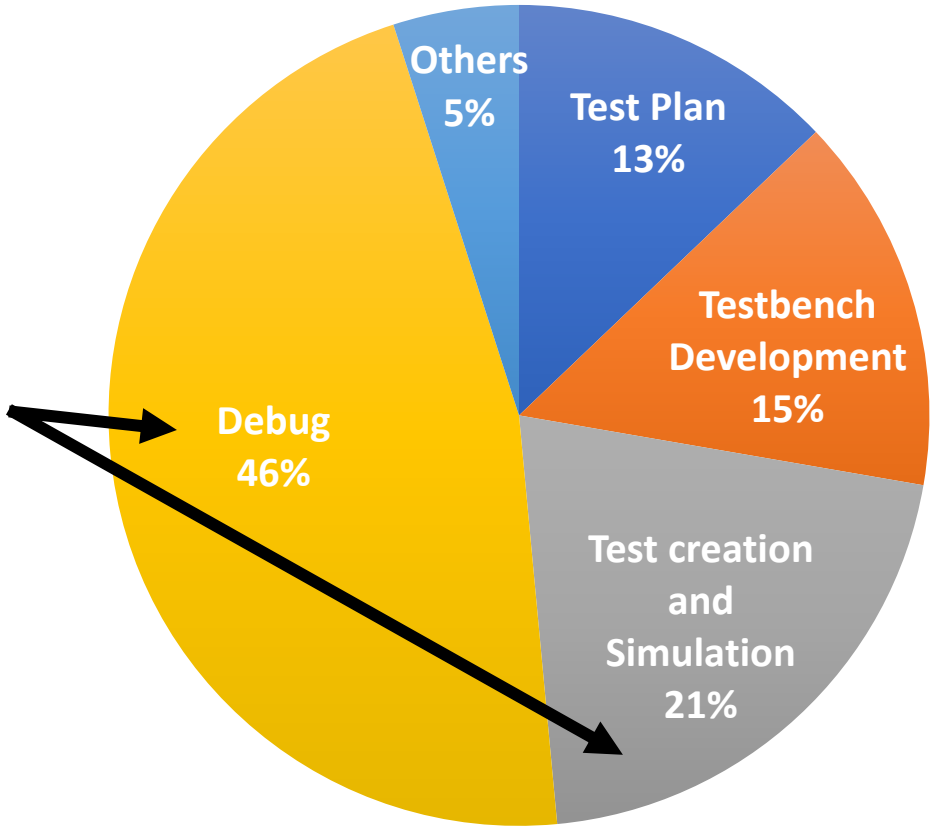


Source: https://www.valytic.co.uk/the-origin-of-bugs



Time Spent by Verification Engineers

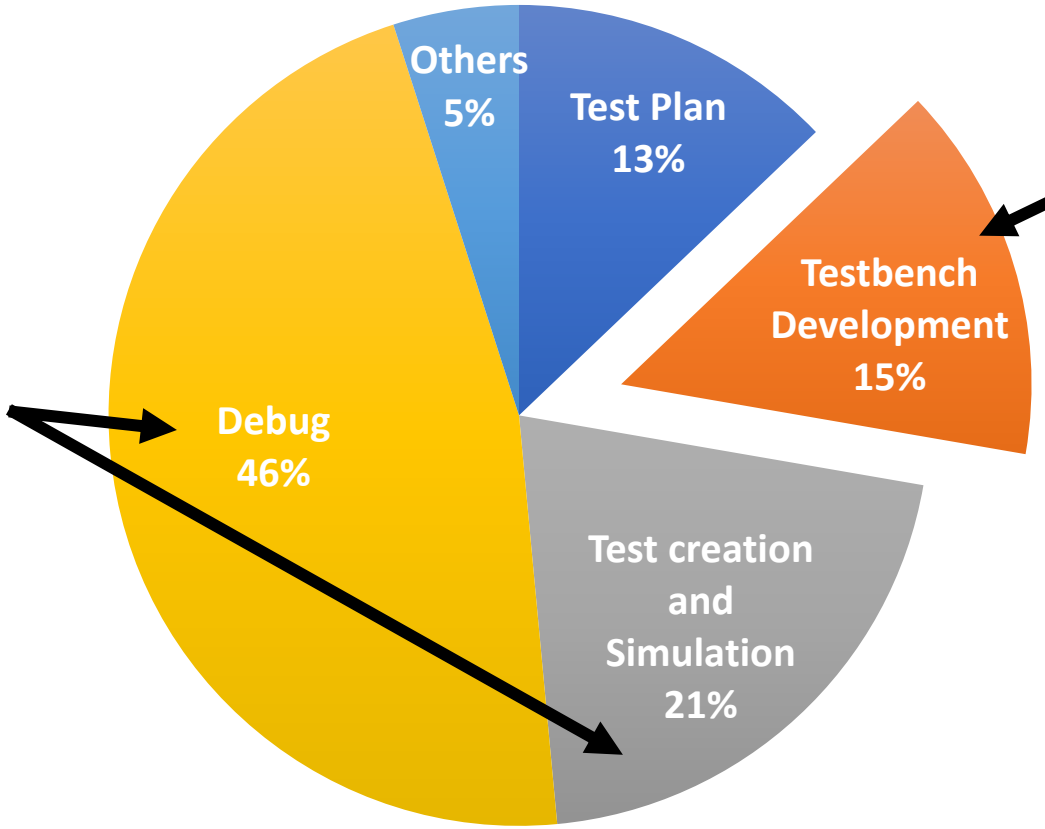
EDA Focus Areas



Source: Wilson Research Group Functional Verification Study, 2022

Time Spent by Verification Engineers

EDA Focus Areas



What about this?

Source: Wilson Research Group Functional Verification Study, 2022

EDA Offerings for TB Development

Verification IP

Building blocks of UVM TB

Complex integration

Testbench Generator

Generates UVM TB framework

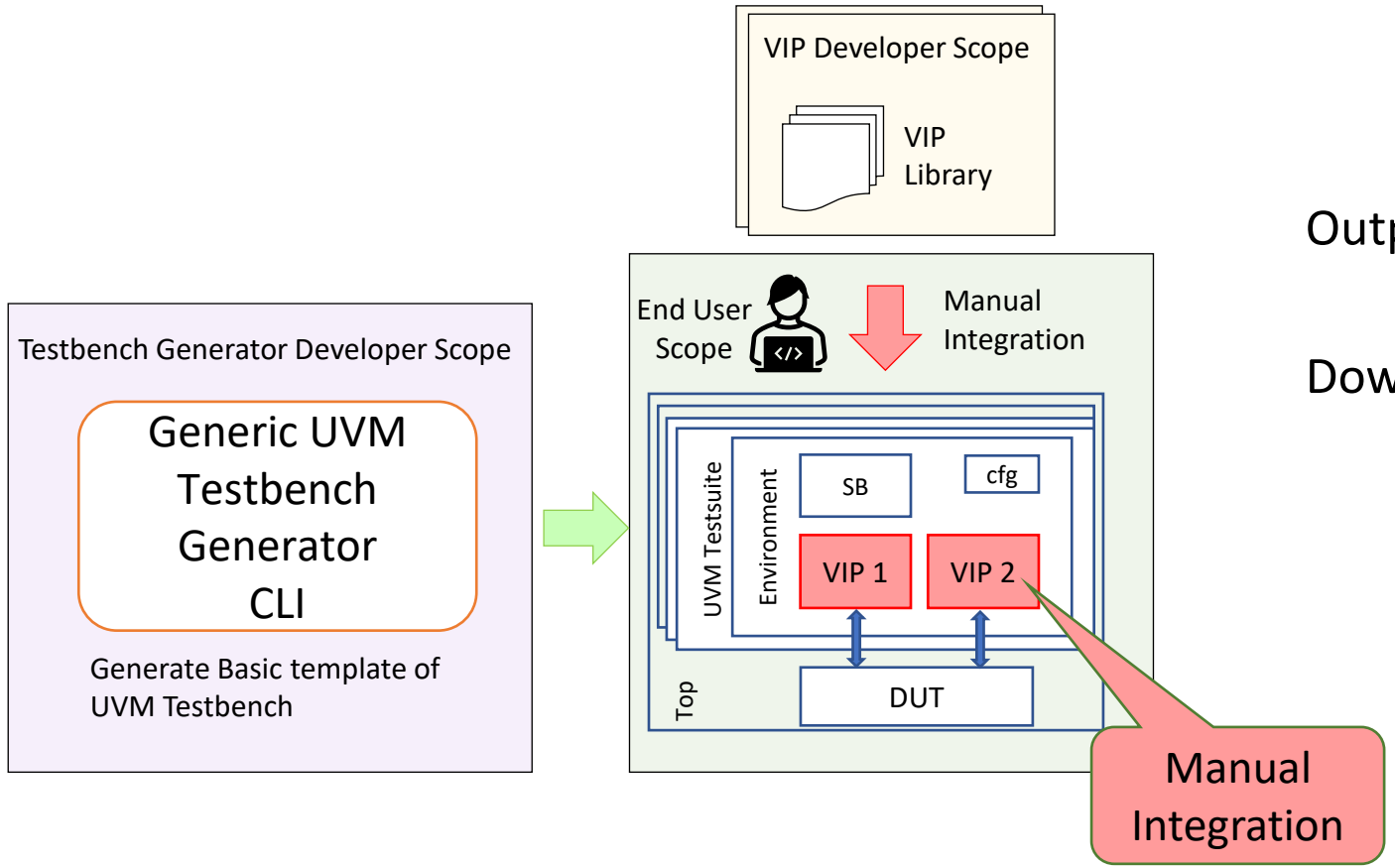
Closed ecosystem

Integrated Development Environment

Faster TB coding - Autocomplete & Debug

LRM support

UVM TB Development - Typical Flow



Output:

- Basic UVM TB template

Downside:

- Manual integration of DUT & VIPs
- Not fine-tuned for end-user's DV ecosystem

TB Development Challenges

Complex Protocols

- *Not straight forward to port VIP example cases to user's TB*

Learning curve

- *VIP structure, configs, sequences, coverage, checkers, interfaces, etc.*

Integration

- *Find VIP packages, class names, generate libraries, extract models, etc.*

Manual Work

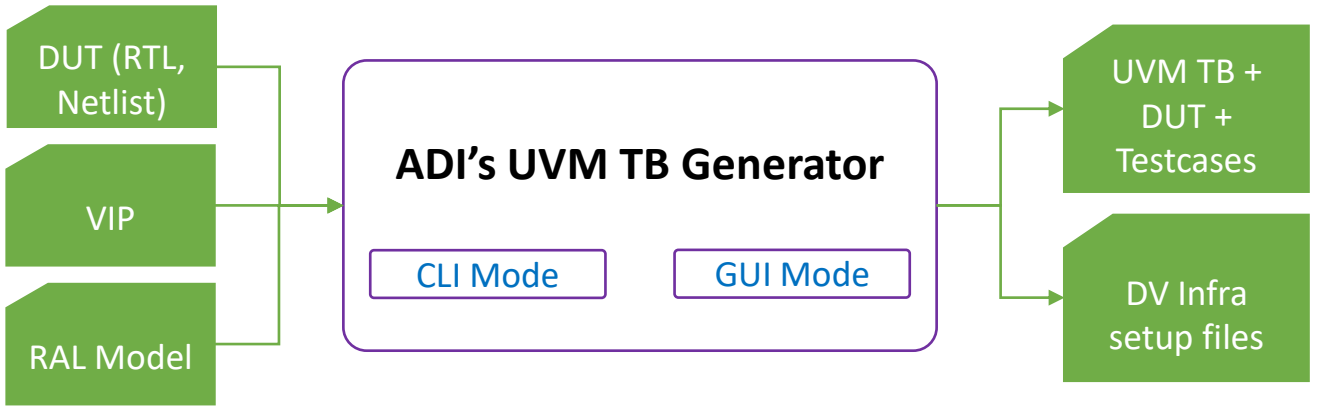
- *Leads to issues which are difficult and time-consuming to root cause*

Duplicate Effort

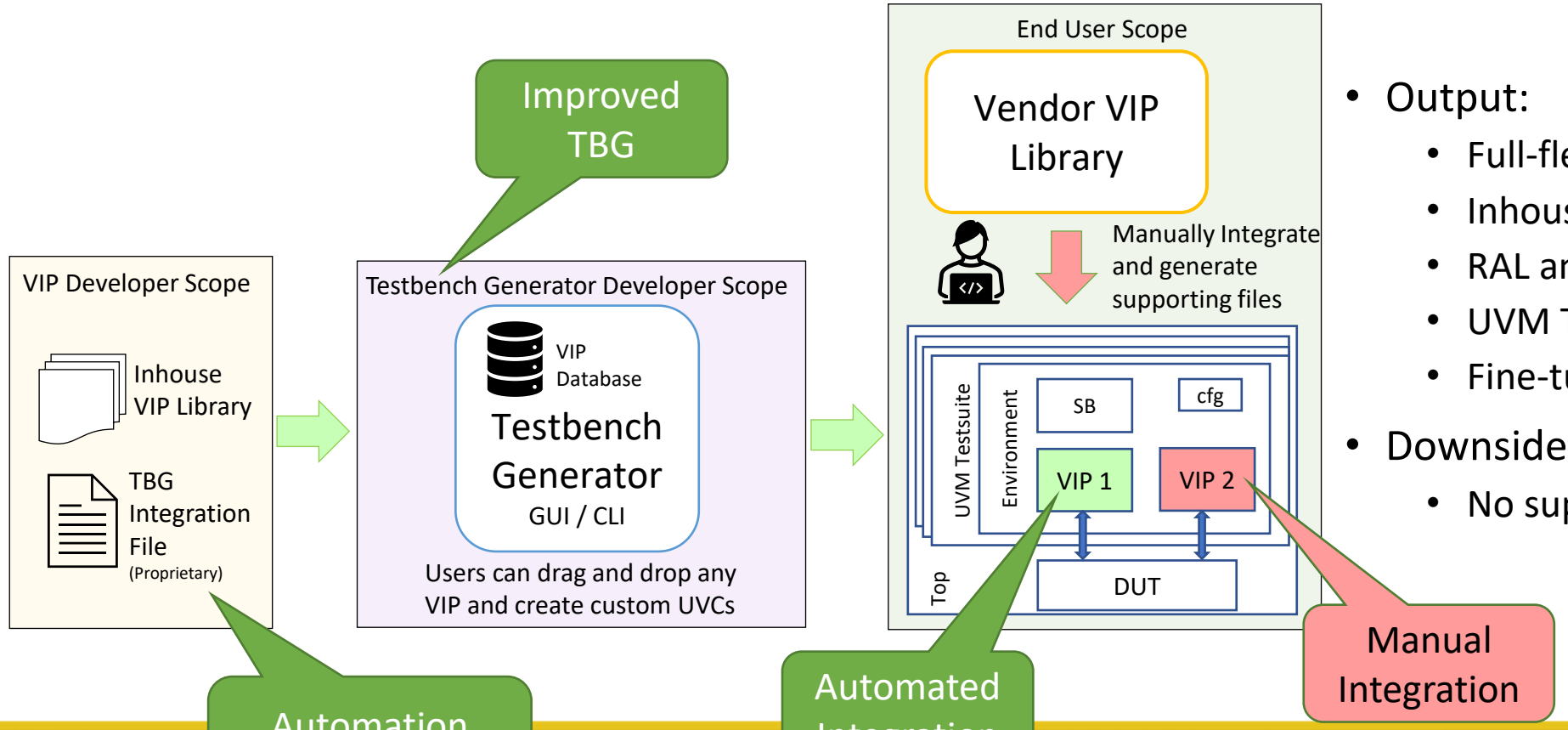
- *Design/TB architecture updates lead to re-do of manual work*

ADI's TB Generator

- ADI internally developed UVM Testbench Generator that can generate unified testbench for Digital, DMS, AMS and Analog DV



TB Development – Existing Flow (ADI)

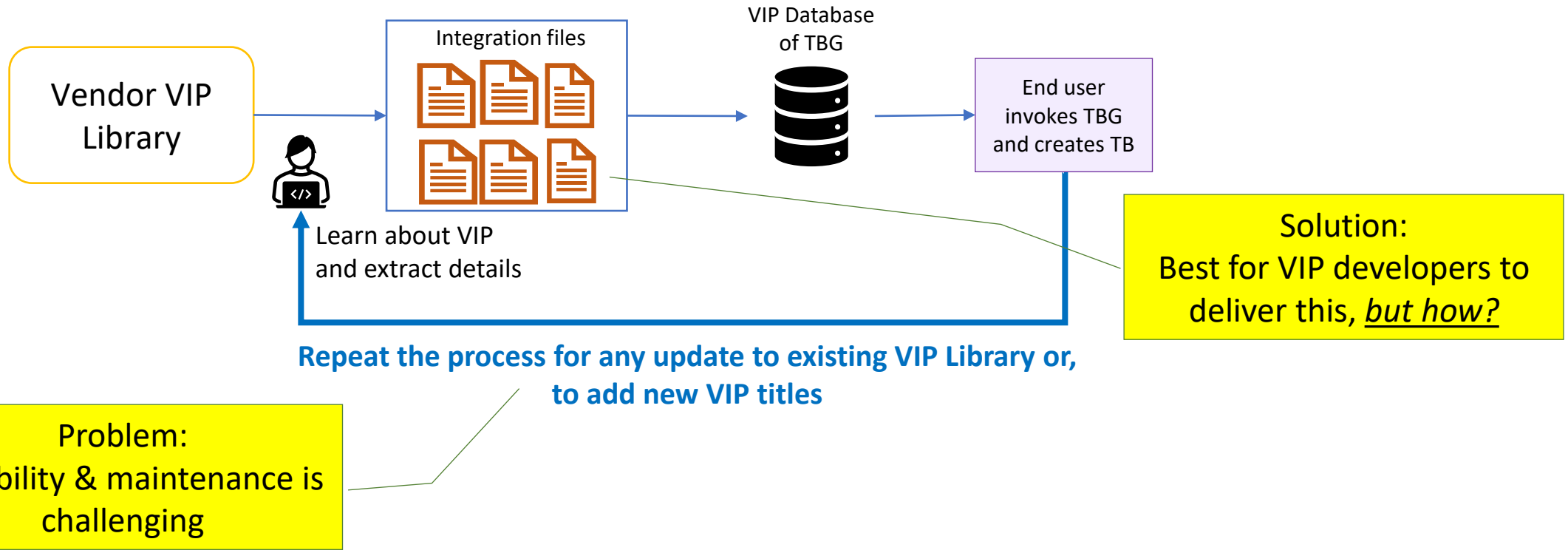


- Output:
  - Full-fledged UVM TB
  - Inhouse VIP+DUT integrated
  - RAL and SB integrated
  - UVM Tests
  - Fine-tuned for ADI DV ecosystem
- Downside:
  - No support for Vendor VIPs

If we can automate integration of inhouse VIPs,
then **what stops us** from doing the same for vendor VIPs?

accellera
SYSTEMS INITIATIVE

Challenges in Vendor VIP Integration

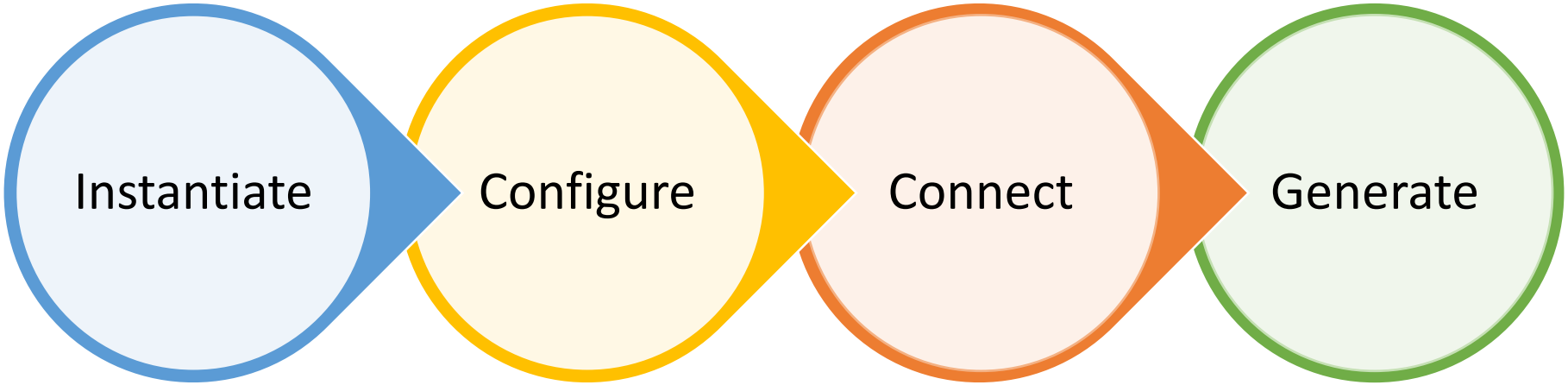


Need for a non-proprietary format to capture the VIP Metadata (integration details)

accellera
SYSTEMS INITIATIVE

VIP Integration Process

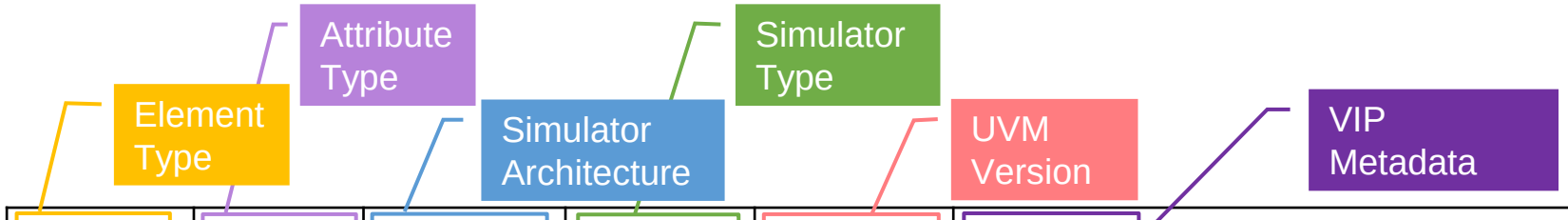
- Four step (universal) process



Metadata Required for VIP Integration

Installation path	File Includes	Packages imports	Parameter, env vars / compile options
Agent / Env class	Interface	Protocol and VIP config	Transaction, Sequence lib, Sequencer
	Scoreboard, coverage	RAL integration	

VIP Metadata Template



Element	Attribute	Sim_Arch	Simulator	UVM_ver	Metadata
setup	inc_dir	32	XLM	UVM12	<Path to VIP 32bit dir>
setup	inc_dir	64	VCS	UVMIEEE	<Path to VIP 64bit dir>
…					
common	param_list				<Parameter Name>=<Value>
…					
source	if_type				<interface name>
…					
sink	agent_type				<agent/env name>
…					

vendor1_common	**vendor1_dp**	vendor1_smbus	vendor1_pmbus	vendor1_axi4	vendor1_ace	…

Vendor specific – Applicable to all VIPs from same vendor

Title specific – Applicable to only to a specific VIP title

Element - Attribute

- “Element” supports the following types:
  - Setup
  - <Agent> - variable
  - Common

- Each element supports multiple attributes
  - Only valid Element-Attribute pairs are supported

Element: “setup”

Attribute	Description	Reference
vendor_name	Name to identify the vendor in TB generator	<vendor name>
vip_name	Name to identify VIP in TB generator	<protocol>:<version>
env_var	Set required environment variables	setenv VIP_LIB_PATH ${TB_ROOT}/agents/<vendor>/<vip>
src_path	Path to the location of the user-editable VIP source code that needs to be copied to the generated TB	${VIP_ROOT}/…./
pre_comp	Perform required operation/ execute script before compilation	source $VIP_LIB_PATH/vip_comp.csh
pre_sim	Perform required operation/ execute script before simulation	source $VIP_LIB_PATH/vip_sim.csh
pre_comp_sim	Perform required operation/ execute script before compilation and simulation	source $VIP_LIB_PATH/vip_all.csh
comp_opt	Compilation options	-define VENDOR_PROTOCOL
comp_file	Files to be compiled	${TB_ROOT}/agents/protocol/x/y/z
inc_dir	Directories required for compilation	${TB_ROOT}/agents/protocol/x/y/z
sim_opt	Run time options for simulator	-pli ${VIP_ROOT}/somefile.so

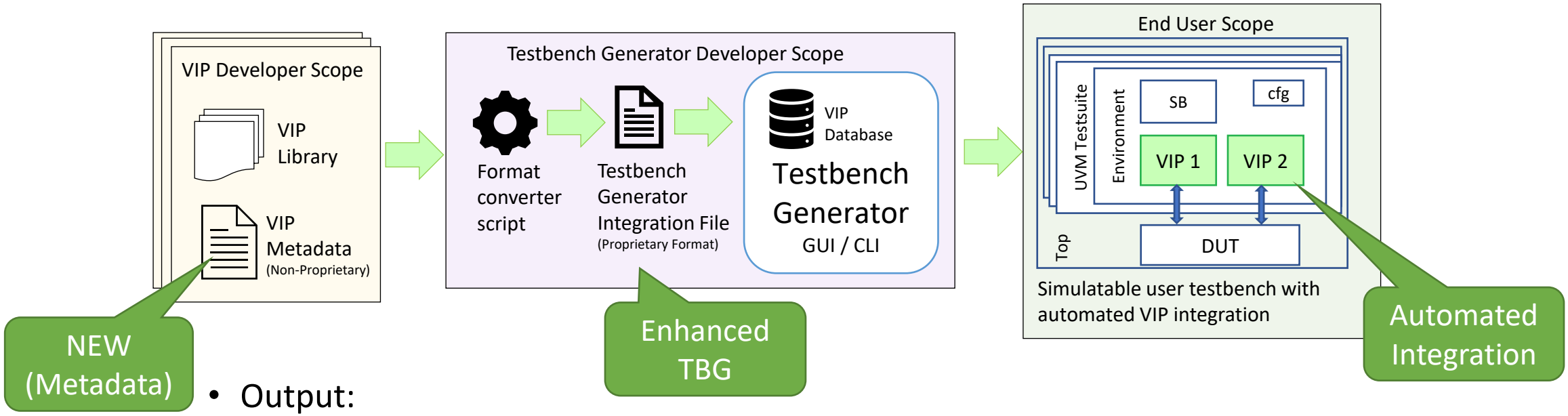
Element: “common” or “<Agent>”

Attribute	Description	Reference
pkg_import	VIP env/agent package to be imported in TB env scope Scope: TB env package	vendor_protocol_pkg
inc_file	Files to be included in TB environment package Scope: TB env package	vip_protocol_file.sv
add_tbpkg_code	Custom code required for VIP compilation This code will be added in the TB environment package scope prior to importing other packages Typically used for type or vendor specific forward-declaration/parameters Note: If used in <vendor>_common sheet, its vendor-specific else it is type-specific	typedef class vip_example_class;
param_list	List of parameters used in the TB env scope Comma separated list should have the parameter name and its value Suggestion: Recommended to use only one parameter per line Scope: TB Env package	ADDRESS_WIDTH=32, DATA_WIDTH=32

Element: “common” or “<Agent>”

Attribute	Description	Reference
agent_type	Agent type to instantiate and create the VIP instance in the TB environment	vip_protocol_agent
if_type	Interface type	vip_protocol_interface
sig_list	List of VIP interface signals available for connection with DUT ports	input sigA, output [1:0] sigB, inout sigC
cfg_type	VIP configuration class type	cfg_type
cfg_vars	VIP configuration class variables to be available in generator GUI Syntax: <field> = <#value1,value2,value3#> <field> = <[value1:value10]> <field> = <value> Note: Relative to top VIP config instance	vip_protocol_kind = <#xkind,ykind#>;
tr_type	Transaction class type	vip_xtn
sb_port	Used to Connect monitor analysis ports to scoreboard implementation ports	vip_protocol_monitor.x_dir_prt

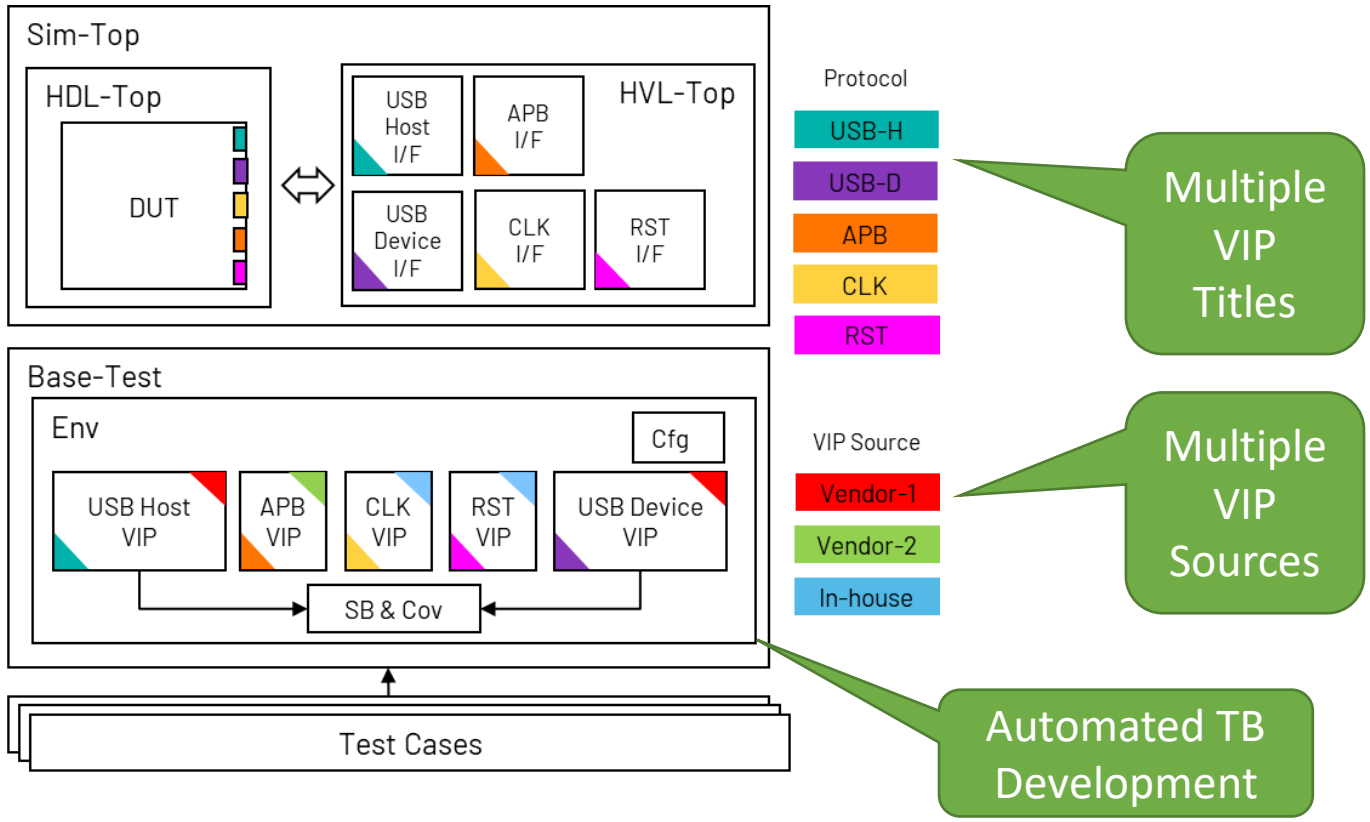
TB Development – New Flow

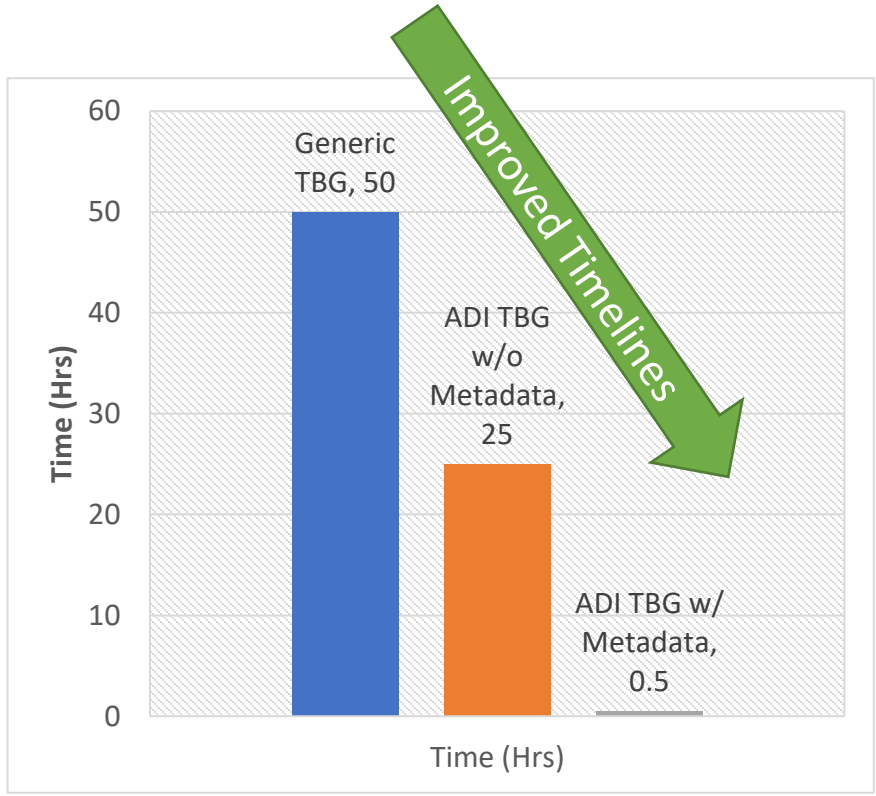


- Output:
  - Full-fledged UVM TB
  - VIP+DUT
  - RAL and SB

- UVM Tests
- Fine-tuned for ADI DV ecosystem
- **Native support for vendor VIPs**

Example Usecase: Development of USB 3.2 TB

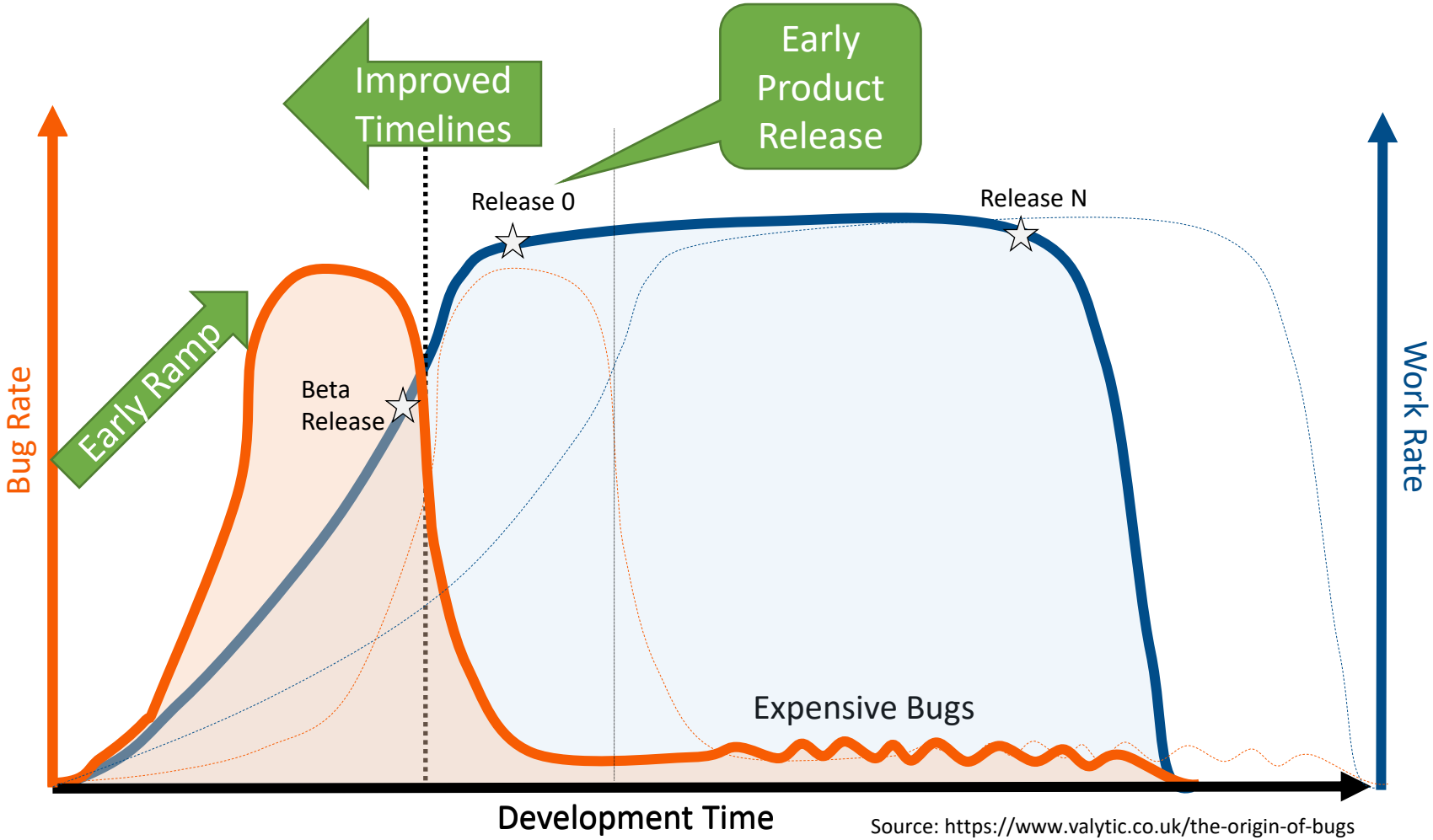




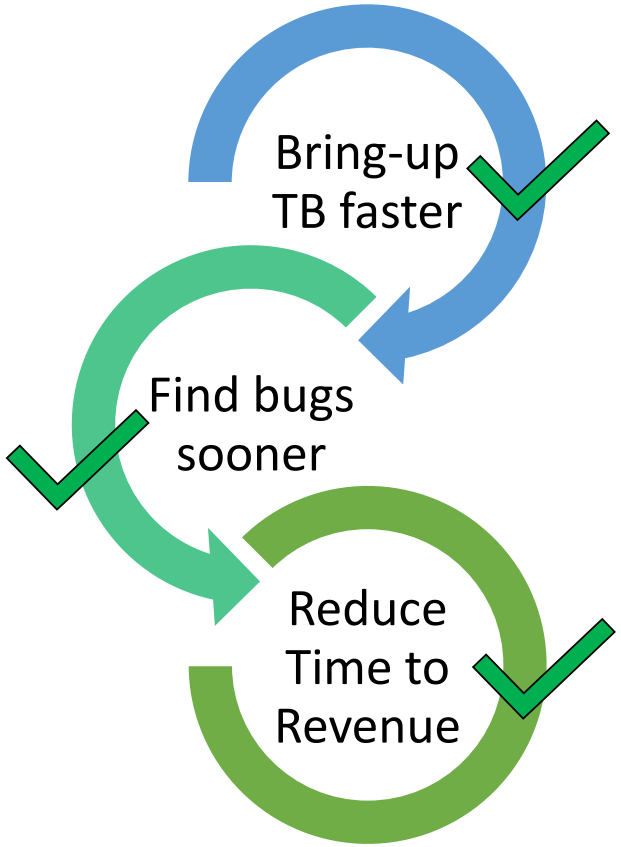
Results

- TB development reduced from weeks to minutes
- Over 20 vendor VIPs are supported
  - IEEE Ethernet, USB 3.2, VESA DP, MIPI CSI-2, MIPI I3C, AMBA etc
- Deployed in production environment
  - Deployed in four projects; many more in pipeline
  - Demand for new title addition

What Did We Achieve?



Source: https://www.valytic.co.uk/the-origin-of-bugs



Summary

- Non-proprietary “Metadata” format can enable automated integration of vendor VIPs
  - Shrink TB development time & improve Time To Revenue
  - Lower the entry-bar for designers, Analog/MS DV experts who aren’t UVM savvy
  - Increased adoption of MDV & UVM VIPs
- Scalable and non-invasive solution ensuring liberty to developers
  - Win-win solution for vendor and end-users

Call for Action!

- Give this a try!
  - If you like it, ask your TBG and VIP vendors to support Metadata
  - Refer to the paper for the complete list of supported elements & attributes
- Connect with authors to contribute to development of Metadata

Thank You!

accellera
SYSTEMS INITIATIVE