Connecting Enterprise Applications to Metric Driven Verification

Author: Matt Graham – Cadence Design Systems, Inc.
Advisor: Gergely Sass – NXP Semiconductor
Introduction

• With the advent of UVM, Metric Driven Verification is now a codified process for verification of IP designs
• There are still several challenges at this level, not the least of which is a standardized way of integrating information that has traditionally been outside of the verification domain
• This paper describes a natural extension of MDV to traditionally disconnected enterprise applications
• Specifically - as example, a very common bug tracking example will be used as a proof of concept
Openness Enables Key Decisions

Coverage Closure?

Declining Bug Rate?

Risk Factors?

Verification Plan Results?

The Key is Coordinated Information!

Are all factors agreeing?

- Yes → Tape Out
- No → Keep working...

Verification Plan (vPlan)
- Team A
  - Feature F: 50%
  - Feature C: 75%
  - Feature E: 90%
- Team B
  - Feature A: 0%
  - Feature D: 40%
  - Feature E: 60%
- Team C
  - Feature A: 40%
  - Feature B: 50%

Team A
Feature F: 50%
Feature C: 75%
Feature E: 90%

Team B
Feature A: 0%
Feature D: 40%
Feature E: 60%

Team C
Feature A: 40%
Feature B: 50%
NXP Integrated Flow

- **DOORS**: Management, Requirements (automated)
- **CollabNet**: Source & Bug Tracking Portion
- **ePlanner**: New RTL tag created by designer, Verif. team manually tags verif. environment on top of it
- **eManager**: Issue found, results

Activities:
- Create problem report with regression tag (manual)
- Issue found
- Coverage information (automated)
- RTL fix

Flow:
1. Designer creates RTL fix.
2. Verification (Verif.) team manually tags verif. environment on top of the RTL fix.
3. Regressions: Flow automated, Regressions are run on manually created regression tag (contains RTL + verif. env.).
4. Requirements (automated) are managed by various components.
Triage Based Bug Trigger

Simple Bug Tracking Synchronization Flow
Triage Process Will Trigger Bug ID being assigned
Integrating Bug Tracking to vManager

• After failures confirmed within vManager, user click user Bugzilla user defined button to create a bug ID
• Bug ID, description, status - all synchronized with vManager using REST API
Integration Code – Python / TCL

Bugzilla Python Code for REST

```python
bz = bugzilla.Bugzilla("http://localhost/bugzilla/", "magraham@cadence.com", "webmaster")

if args.bug_action == 'create':
    newbug = bz.Bug(args, {
        "product": args.product,
        "component": args.component,
        "summary": args.desc,
        "version": args.version,
        "op_sys": args.op_sys,
        "platform": args.platform
    })
    id = newbug.create()
    print id
    if args.comment !="" :
        newbug.add_comment(args.comment)
```
What is REST?
The Communication Architecture of the Web

- REST - Representational State Transfer
- Defacto Industry Standard – Surpassed SOAP in 2008
- Syntax is URL style: http://example.org/news
- Methods: GET, PUT, DELETE, HEAD, POST
- Representations: HTML Hypertext Document
- Many different formats: JSON, HTML or XML
JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

{"for-more-info": "http://www.json.org/"}
REST for Enterprise Integrations

- REST is the defacto integration standard for 3rd party solutions
- Extending the MDV environment to be REST compatible enables better verification decision making
- Enterprise Integrations will improve information access and improve the overall verification process
Questions