

Connecting Enterprise Applications to Metric Driven Verification

Author: Matt Graham – Cadence Design Systems, Inc.

Advisor: Gergely Sass – NXP Semiconductor

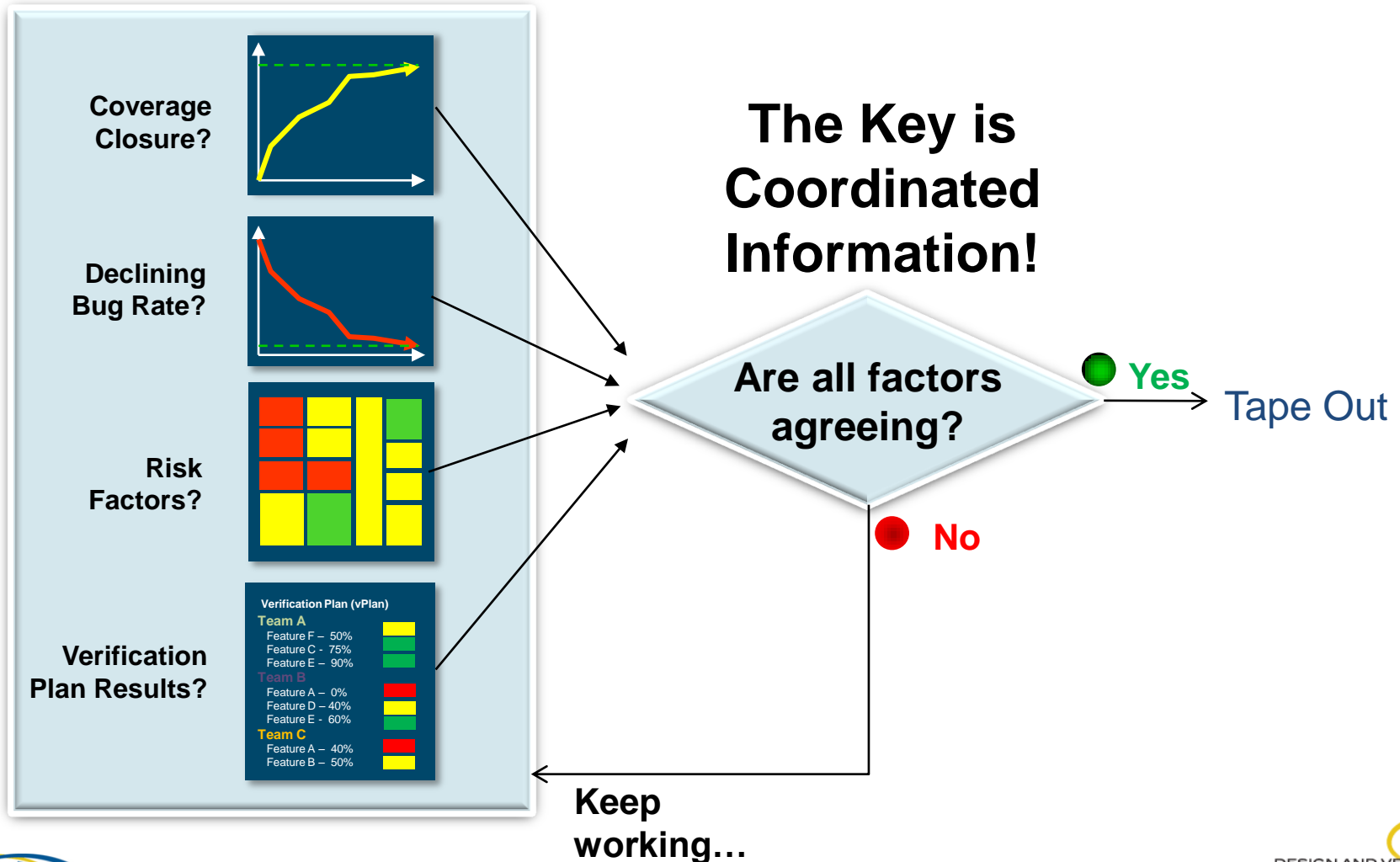
cādence[®] **NXP**



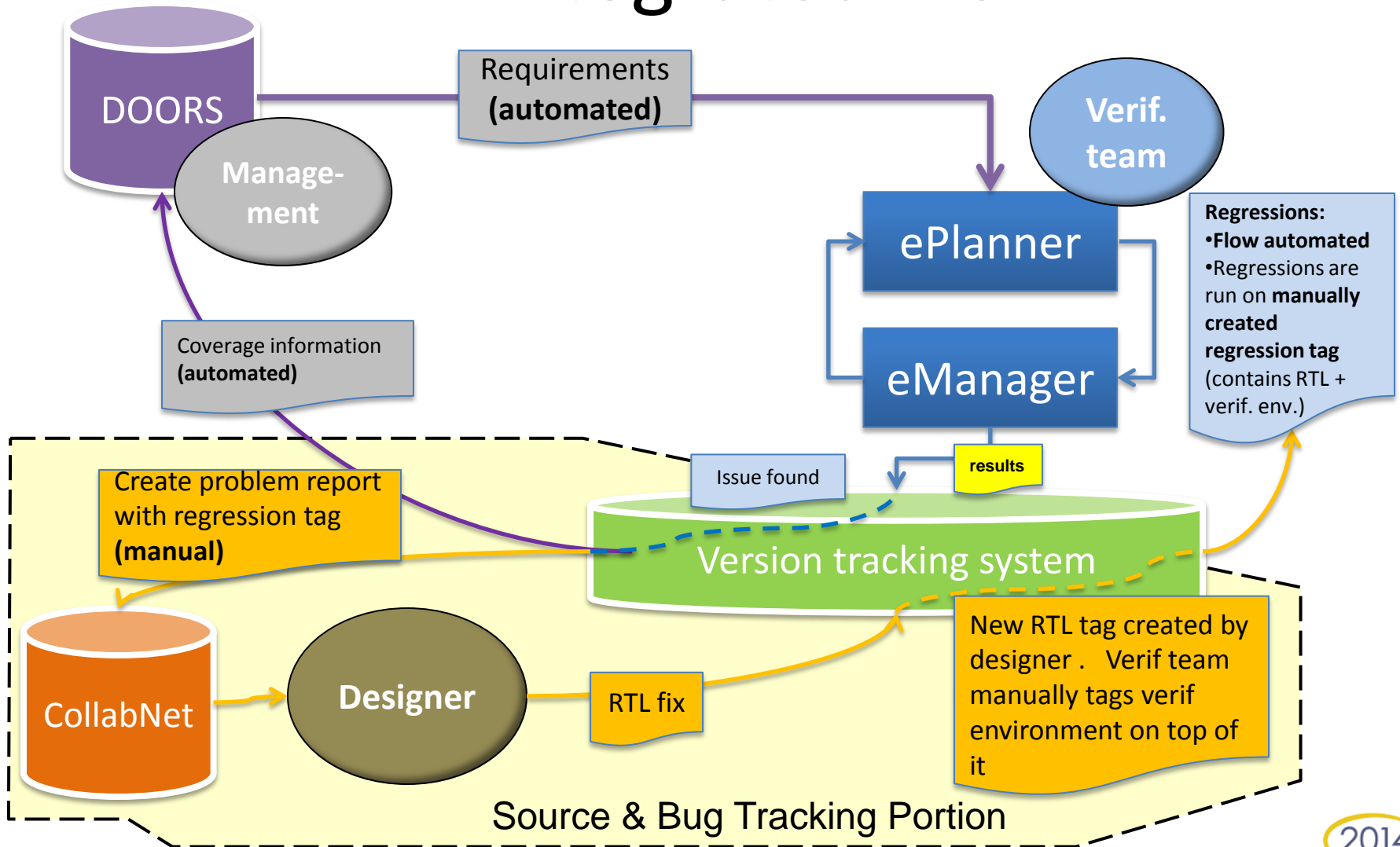
Introduction

- 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

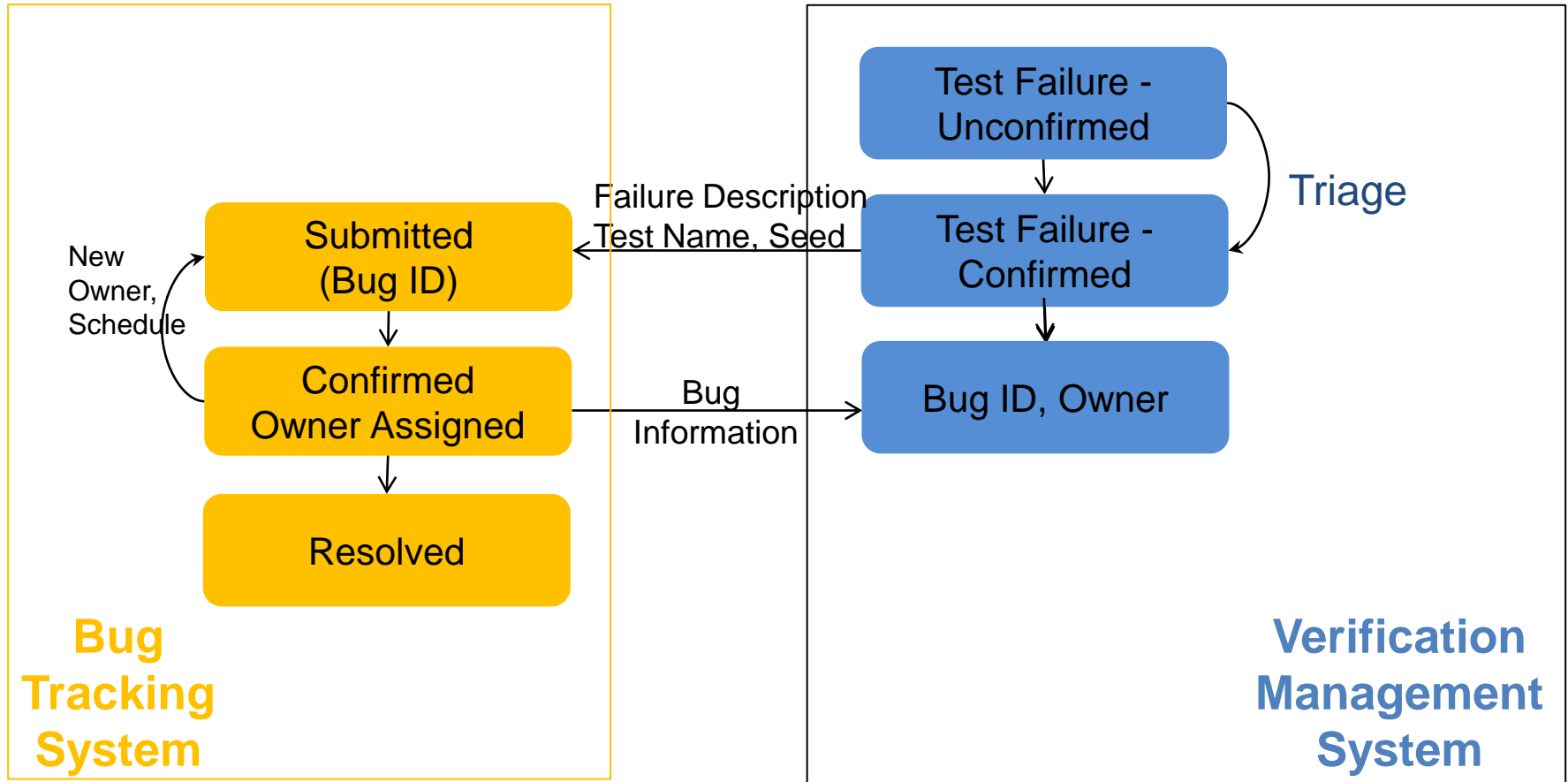
Openness Enables Key Decisions



NXP Integrated Flow

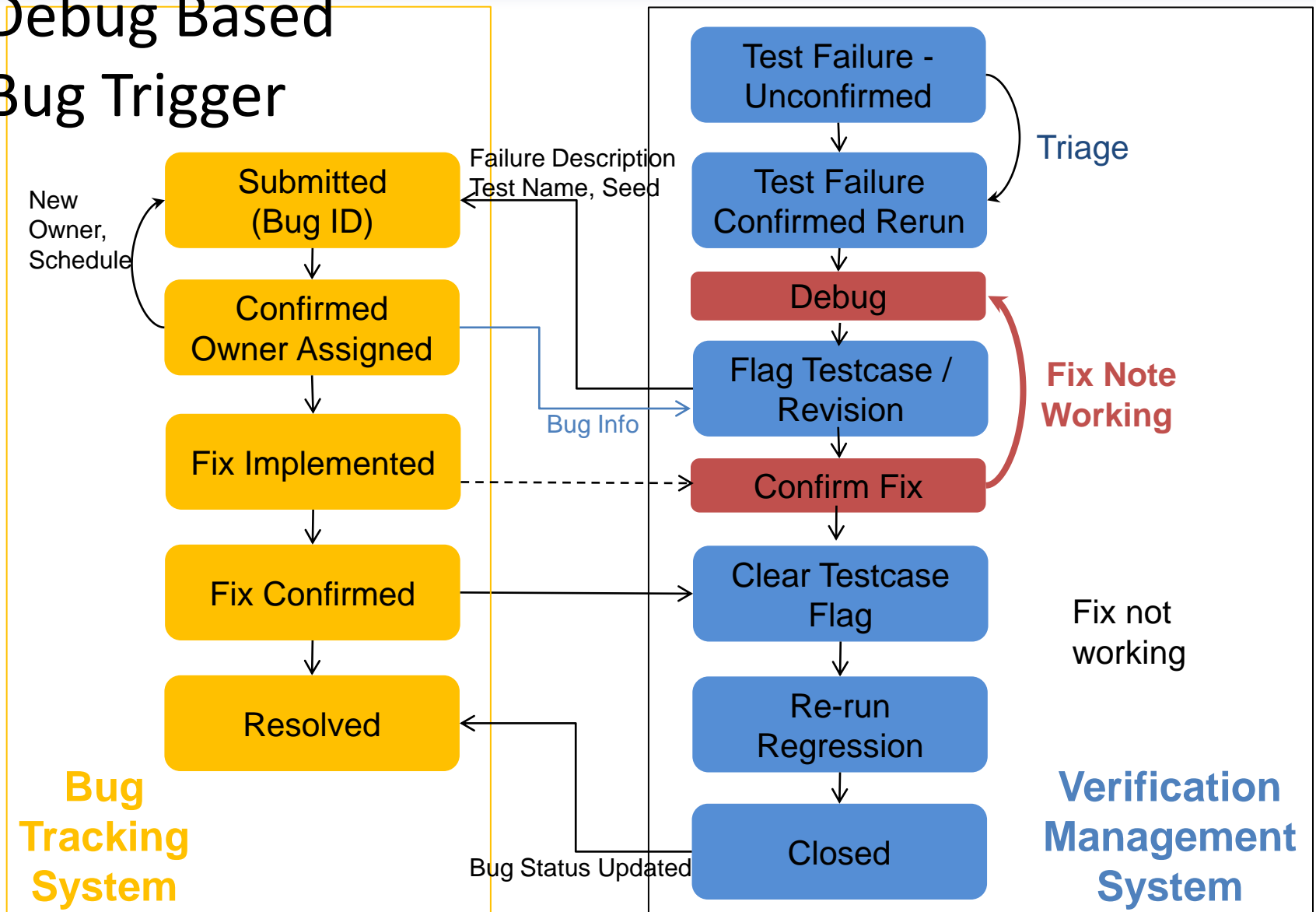


Triage Based Bug Trigger



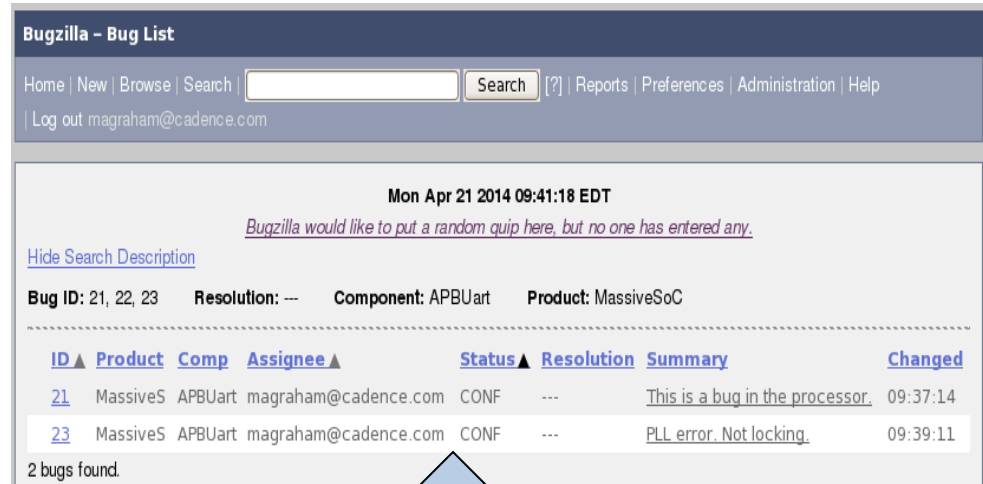
Simple Bug Tracking Synchronization Flow
Triage Process Will Trigger Bug ID being assigned

Debug Based Bug Trigger

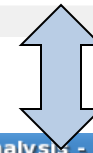


Integrating Bug Tracking to vManager

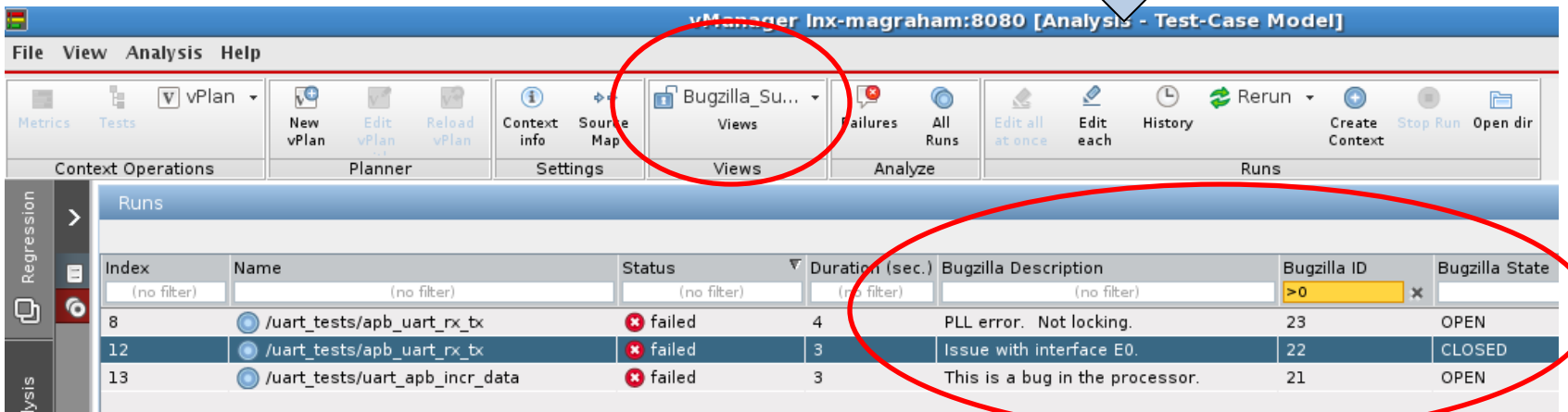
- After failures confirmed within vManager, user click user Bugzilla user defined button to create a bug ID
- Bug ID, description, status - all synchronized with vManager using REST API



The screenshot shows the Bugzilla web interface. At the top, there's a navigation bar with links like Home, New, Browse, Search, and a search input field. Below that, the current date and time are displayed: "Mon Apr 21 2014 09:41:18 EDT". A message states: "Bugzilla would like to put a random quip here, but no one has entered any." There's a "Hide Search Description" link. Below this, summary information is shown: "Bug ID: 21, 22, 23", "Resolution: ---", "Component: APBUart", and "Product: MassiveSoC". A table lists the bugs with columns for ID, Product, Comp, Assignee, Status, Resolution, Summary, and Changed. Two bugs are listed: ID 21 (MassiveS, APBUart, magraham@cadence.com, CONF, ---, "This is a bug in the processor.", 09:37:14) and ID 23 (MassiveS, APBUart, magraham@cadence.com, CONF, ---, "PLL error. Not locking.", 09:39:11). At the bottom, it says "2 bugs found."



REST Interface



The screenshot shows the vManager interface. The title bar reads "vManager Inx-magraham:8080 [Analysis - Test-Case Model]". The menu bar includes File, View, Analysis, and Help. The toolbar has various icons for Metrics, Tests, vPlan, Context info, Settings, Bugzilla_Su..., Failures, All Runs, Edit all at once, Edit each, History, Rerun, Create Context, Stop Run, and Open dir. The main area is divided into sections: Context Operations, Planner, Settings, Views, Analyze, and Runs. The "Views" section is circled in red. The "Runs" section contains a table with columns: Index, Name, Status, Duration (sec.), Bugzilla Description, Bugzilla ID, and Bugzilla State. The table data is as follows:

Index	Name	Status	Duration (sec.)	Bugzilla Description	Bugzilla ID	Bugzilla State
(no filter)	(no filter)	(no filter)	(no filter)	(no filter)	>0	x
8	/uart_tests/apb_uart_rx_tx	failed	4	PLL error. Not locking.	23	OPEN
12	/uart_tests/apb_uart_rx_tx	failed	3	Issue with interface E0.	22	CLOSED
13	/uart_tests/uart_apb_incr_data	failed	3	This is a bug in the processor.	21	OPEN

Integration Code – Python / TCL

Bugzilla Python Code for REST

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

if args.bug_action == 'create' :
    newbug = bug.Bug(bz, {"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)
```

```
if {$attr(bug_description) == ""} {
    set desc_string $attr(first_failure_description);
    set attr(bug_description) $attr(first_failure_description);
} else {
    set desc_string $attr(bug_description);
}

set comment $attr(name);
append comment " $attr(sv_seed)";

set attr(bug_id) [exec vm_bz_button create --desc "$desc_string" --comment "$comment"];
puts $attr(bug_id);

set attr(bug_state) OPEN;
```

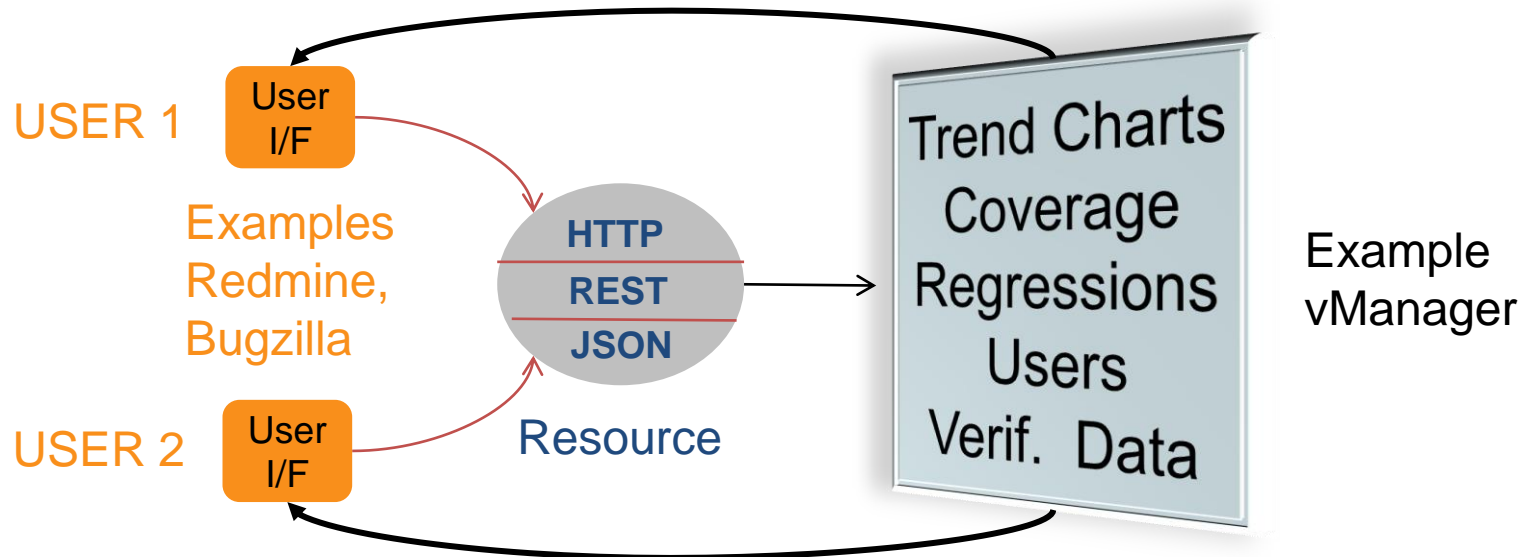
vManager
TCL Code

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



Example: JSON Interchange Format

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.

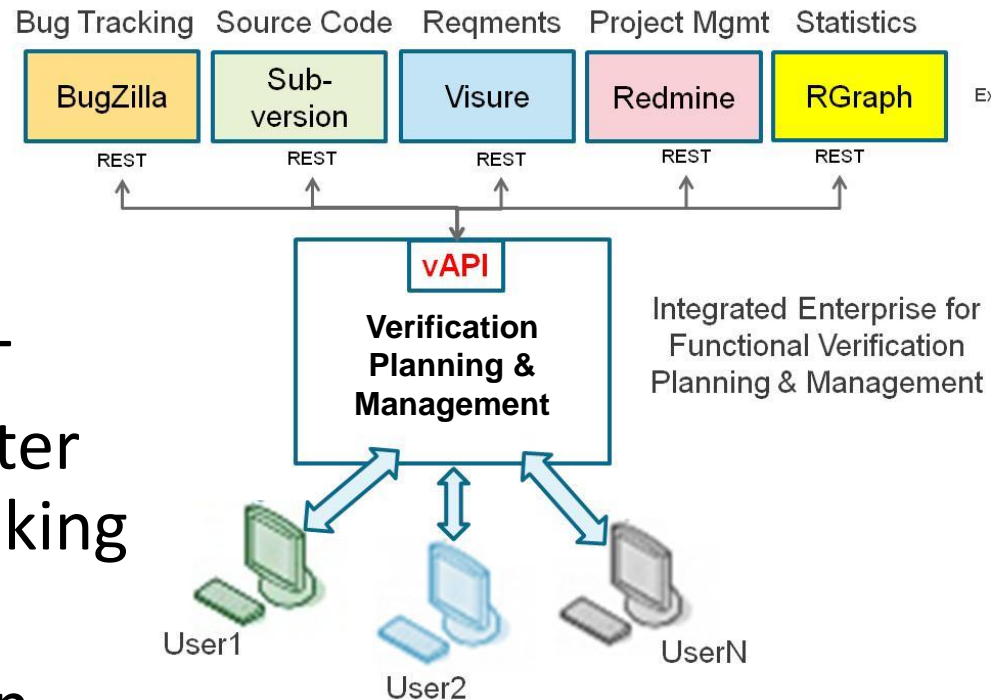
```
{“for-more-info” : “http://www.json.org/”}
```

```
{
  "results": [
    {
      "id": 1,
      "name": "Renegade Internet",
      "username": "renegade",
      "status": true,
      "recycle": false,
      "notes": "",
      "information": {
        "company": "Renegade Internet",
        "name": "Mike Cherichetti",
        "title": "mike@renegadeinternet.com"
      }
    }
  ]
}
```

JSON
JavaScript Object Notation

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