

## How I Learned to Stop Worrying and Love Benchmarking Functional Verification!

**DVCon 2012**

**Mike Bartley, TVS**

## Recognise any of these?

- **Why do we always miss our verification deadlines?**
- **Surely we could have found these bugs earlier?**
- **How comes we seem to have bugs in some basic use case scenarios?**
- **Why do our sites have such different verification capabilities?**
- **How do I integrate this new team in ....?**
- **Why do we seem to make the same mistakes over and over again?**

# Why benchmark?

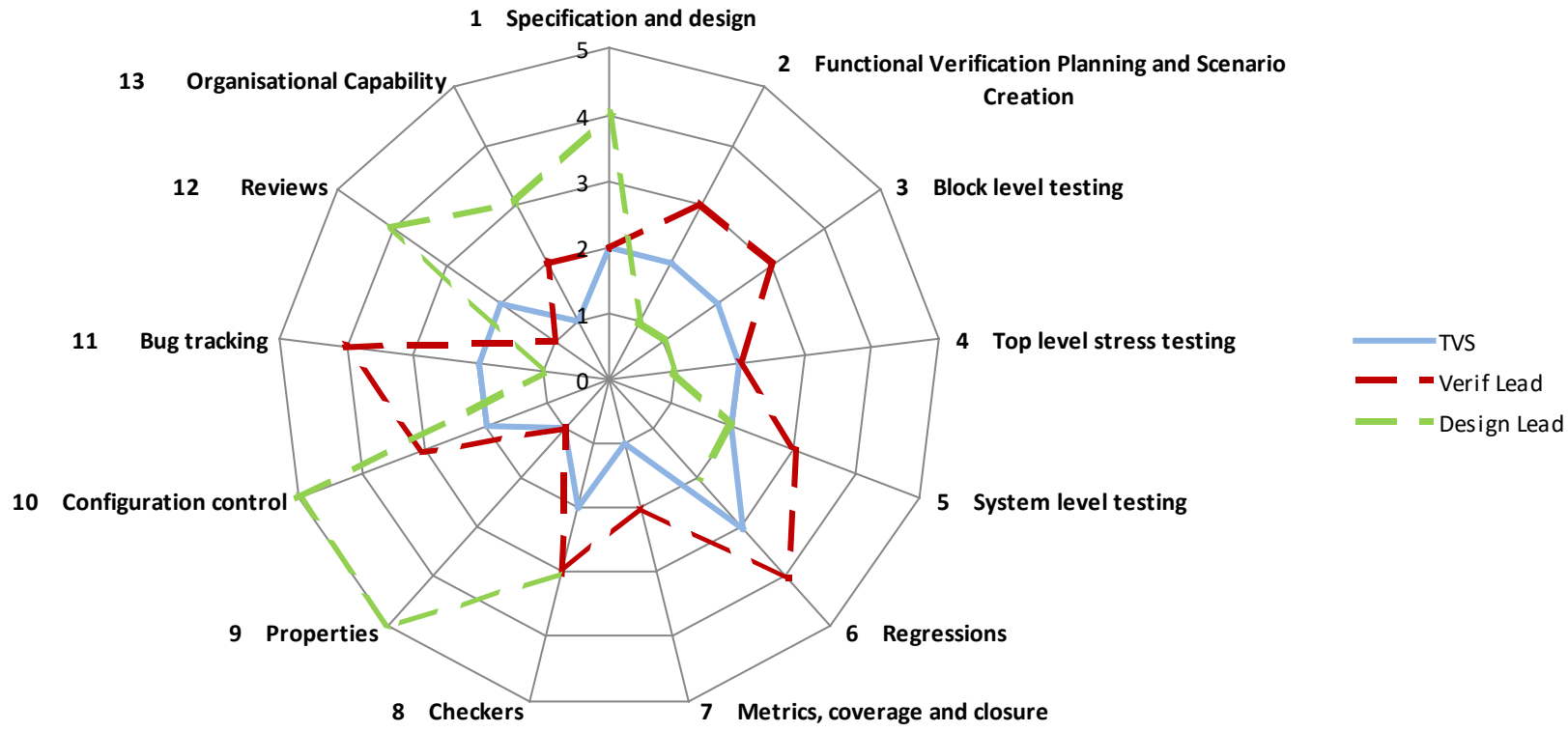
- **To understand current verification capability**
  - and identify improvements
- **Better prepare for tomorrow**
  - Increasing verification complexity
  - Reduced time to market
  - Reducing costs
- **How does benchmarking help with that?**
  - Measure the maturity of functional verification activities
  - Gain an integrated view of the organisation functional verification capability
  - A framework for continuous process improvement
    - Define goals, priorities and actions
    - Regular measurement of progress

# Other benchmarks are available

- **CMMi**
  - General purpose and heavyweight
  - Does not address the specific capabilities relevant to verification
- **Evolving Capabilities Model**
  - Foster and Warner
- **How is FV-CMM different?**
  - View of the whole org from functional verif aspect
  - Objective measure
  - Framework for process improvement
  - Top-down decomposition and bottom-up evaluation
  - 3 key elements: capability, maturity and process

# Different Views of Verification Within a Project

## Self Assessment of Verification Workflow Execution

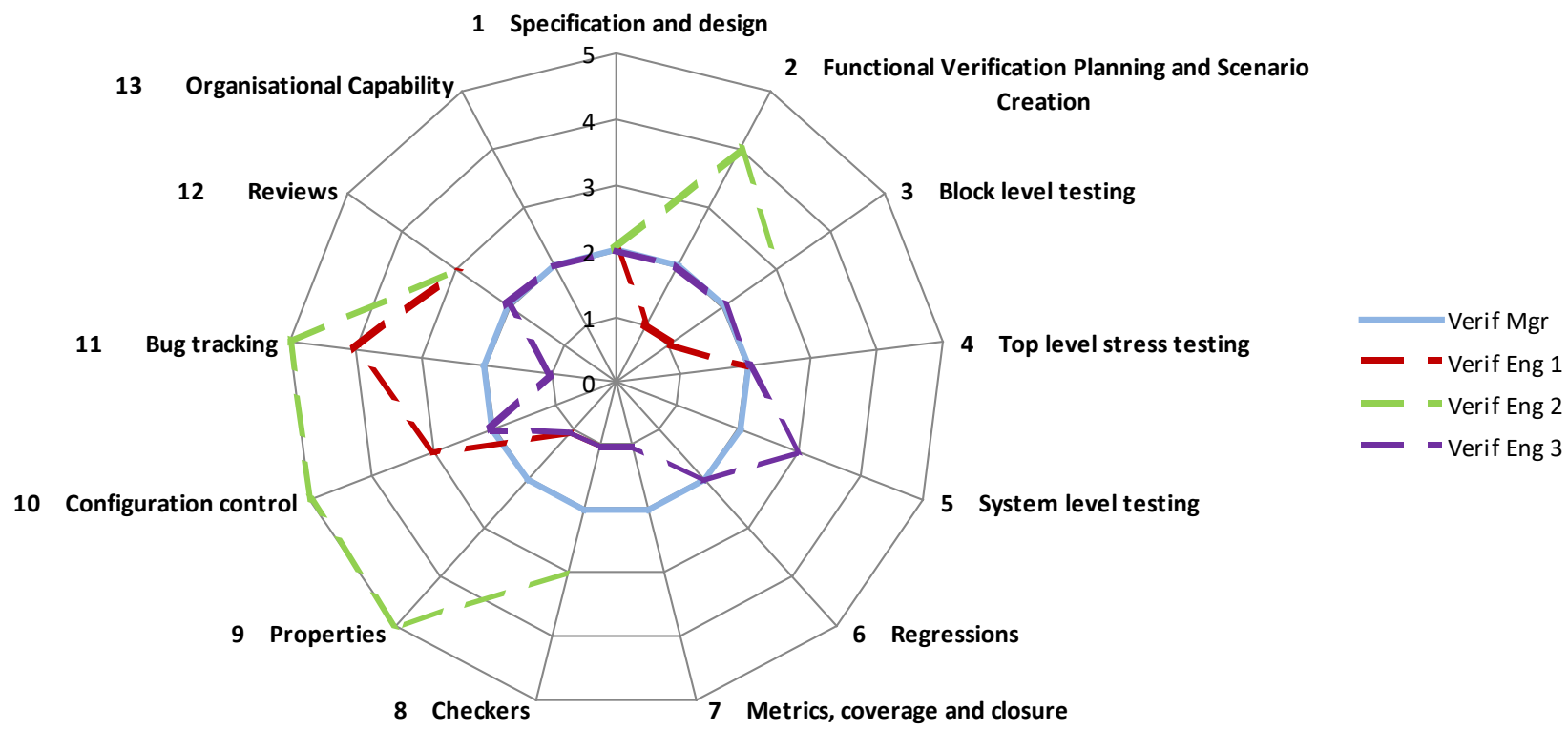


# Process areas

- 1 Specification and design
- 2 Functional Verification Planning and Scenario
- 3 Block level
- 4 Top level stress testing
- 5 System level
- 6 Regressions
- 7 Metrics, coverage and closure
- 8 Checkers and properties
- 9 Configuration control
- 10 Debug
- 11 Bug Tracking
- 12 Reviews
- 13 Organisational Capability

# Verification Teams Can Have Wildly Different Views

## Verif Team Self Assessment of Verification Workflow Visibility



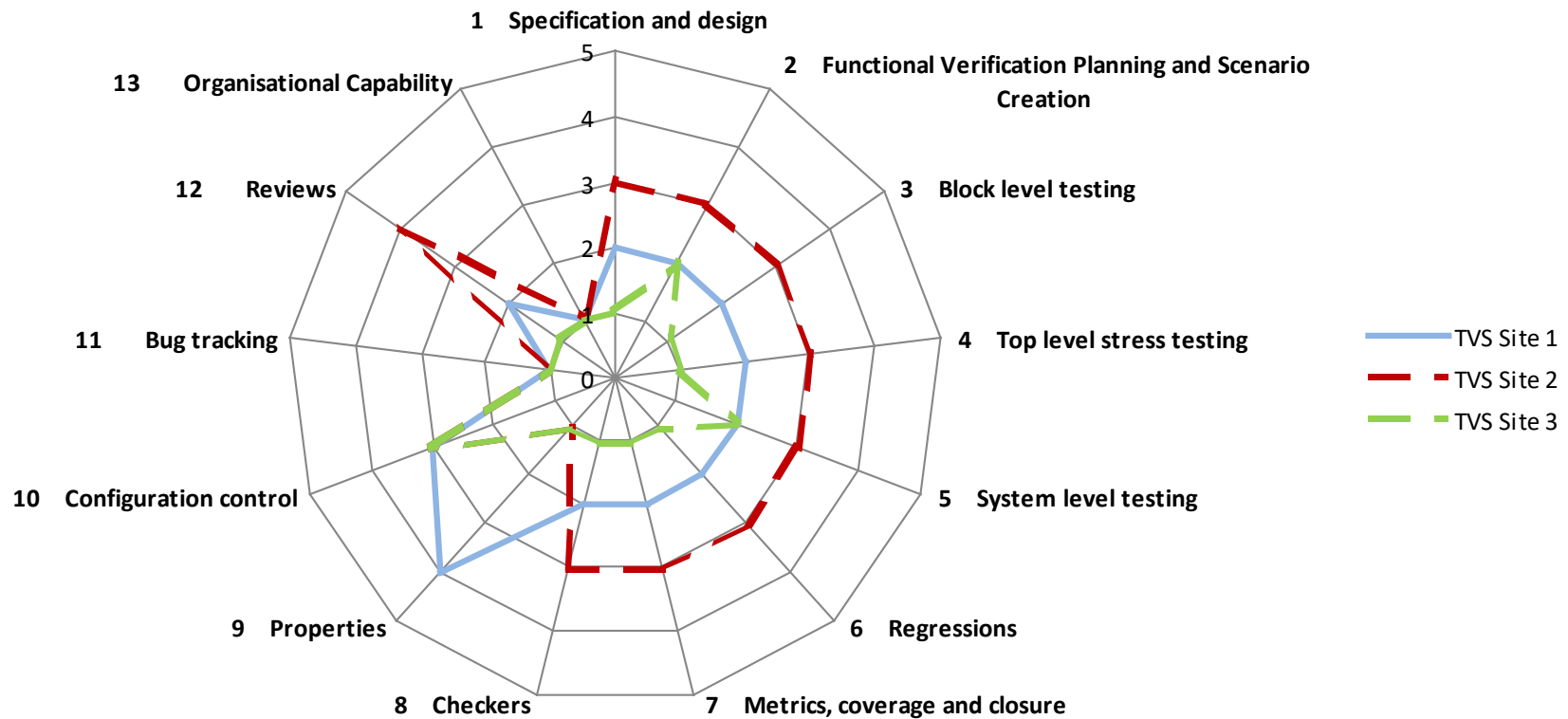
# Evaluation: Axes and levels

	<b>Initial</b>	<b>Managed</b>	<b>Defined</b>	<b>Quantitative</b>	<b>Optimising</b>
<b>Ownership</b>	<b>Individual</b>	<b>Project Team</b>	<b>Project Stakeholders or ad hoc groups of projects</b>	<b>Community</b>	<b>Company wide or institutionalised</b>
<b>Visibility</b>	<b>Not documented No reviews. No metrics.</b>	<b>Documents incomplete or unmaintained. Point reviews. Progress metrics.</b>	<b>Maintained docs. Continuous tracking against quality metrics.</b>	<b>Living docs. Quantified quality metrics.</b>	<b>Data integrated across the organisation.</b>
<b>Execution</b>	<b>Ad hoc</b>	<b>Tasks performed but completion not explicitly checked</b>	<b>Tasks planned and implemented in a systematic fashion. Check completion of planned tasks.</b>	<b>Quantifiable metrics used for coverage closure and release determinism</b>	<b>Quantifiable metrics used to drive continuous improvement.</b>



# Looking at Different Sites Across the Organisation

## TVS Assessment of Different Sites



# The Benchmarking Process

4

FV-CMM process areas	Maturity	Ownership	Visibility	Execution
5 System level testing	3. Defined	2. Project Team	3. Maintained documents and point reviews	3. Tasks planned and implemented in a systematic fashion
5.1 The purpose of each test bench should be clearly identified	3. Defined	2. Project Team	3. Maintained documents and point reviews	3. Tasks planned and implemented in a systematic fashion
5.1.1. The purpose and the scenarios to be reached by each test bench should be clearly identified. The purpose must consider the appropriate level of testing for the various scenarios (e.g. integration with other IP, software debug features, low power features, performance validation via benchmarking)	Environment to run real world software. This is the big thing emulators gives them and it hits things they wouldn't find anywhere else. A mix of what historically available (Symbian, WinCE and Linux), what feels as though it could be useful and the available simulation capacity. Use irritators for OS booting and stress apps. that try to make use of some key system features such as virtualisation and TrustZone. Some reusable software like "crashme", "memcopy". Run this againstr different configs of hardware such as a small L2 cache to increase stress. Can also use Cambridge knowledge from A9 of what cases found bugs.			
5.1.2. Regression testing, using appropriate scenarios and checkers, should be used to validate bug fixes and ensure errors are never reintroduced.				

3

2

1

# So how does benchmarking answer these?

- **Why do we always miss our verification deadlines?**
  - Weakness in particular process areas
- **Surely we could have found these bugs earlier?**
  - Is system verification stronger than block and/or top?
- **How comes we seem to have bugs in some basic use case scenarios?**
  - Weak verification planning and reviews

# So how does benchmarking answer these?

- **Why do our sites have such different verification capabilities?**
  - Weak organisational capabilities do not promote knowledge sharing
- **How do I integrate this new team in ....?**
  - First understand their strengths and areas for improvement
- **Why do we seem to make the same mistakes over and over again?**
  - Are you collecting the right data?
  - Are you doing continuous improvement via benchmarking?

- **Benchmarking helps to**
  - Measure the maturity of functional verification activities
  - Gain an integrated view of the organisation functional verification capability
  - A framework for continuous process improvement
- **FV-CMM is proven lightweight benchmarking process**