
PERFORCE
ON TOUR
2018

PERFORCE

Static Analysis Considerations for Medical Device Software

Presenter

Gerald Rigdon
gerald.rigdon@bsci.com

Fellow, Software Engineering
Boston Scientific

Today’s Agenda

System Components With Software

Static Analysis vs. Dynamic Analysis

Regulatory Requirements and Static Analysis

Why Use Static Analysis?

How We Use Static Analysis Tools

System Components With Software

Definitions

- Dynamic: executing programs
- Static: not executing programs

"Static code analysis is the analysis of computer software that is performed without actually executing programs built from that software (analysis performed on executing programs is known as dynamic analysis)."

— Wikipedia

"Static analysis evaluates the software without executing it. Instead, it examines a representation of the software. In some ways, static analysis is more complete than dynamic analysis, since general conclusions can be drawn and not just conclusions limited to the particular test cases that were selected."

— Nancy Leveson, Safeware, Addison-Wesley, 1995

"Static testing plays an important role in establishing the characteristics of the system over its entire operating range – a function that cannot be performed during dynamic methods because of the infinite number of tests that would be required."

—Neil Storey, Safety-Critical Computer Systems, Addison-Wesley, 1996

Regulatory Perspective

"We're hoping that by quietly talking about static analysis tools, by encouraging static tool vendors to contact medical device manufacturers, and by medical device manufacturers staying on top of their technology, that we can introduce this up-to-date vision that we have."

— Brian Fitzgerald, FDA

— Chloe Taft, CDRH Software Forensics Lab: Applying Rocket Science To Device Analysis.

— Published in "Medical Devices Today", October 15, 2007.

"We use static analysis for the post-market review of commercial medical devices. The aim of our analysis is to determine all possible potential causes for failure in the software that can be identified with the SA tools, and to assess compliance to established software and quality control standards."

— Rick Chapman, FDA, "Symposium on Static Code Analysis and Complex Medical Devices", University of Minnesota, 2009

Why Static Analysis Tools?

- The Dynamic Testing Space
  - Unit testing (white box based on implemented software components)
  - Integration testing (grey box based on software design)
  - Verification testing (black box based on software requirements)
  - Validation testing (black box based on system/user requirements)
- The Problem Space – Candidates for SA Tools
  - Well-defined set of properties that can be checked by a machine.
  - High likelihood of missing a defect in dynamic testing and code reviews.
  - Difficult to perform manually or using simple search techniques.
  - High impact (safety risk, efficacy) if a defect escapes and leads to system malfunction.

Quantitative Factors

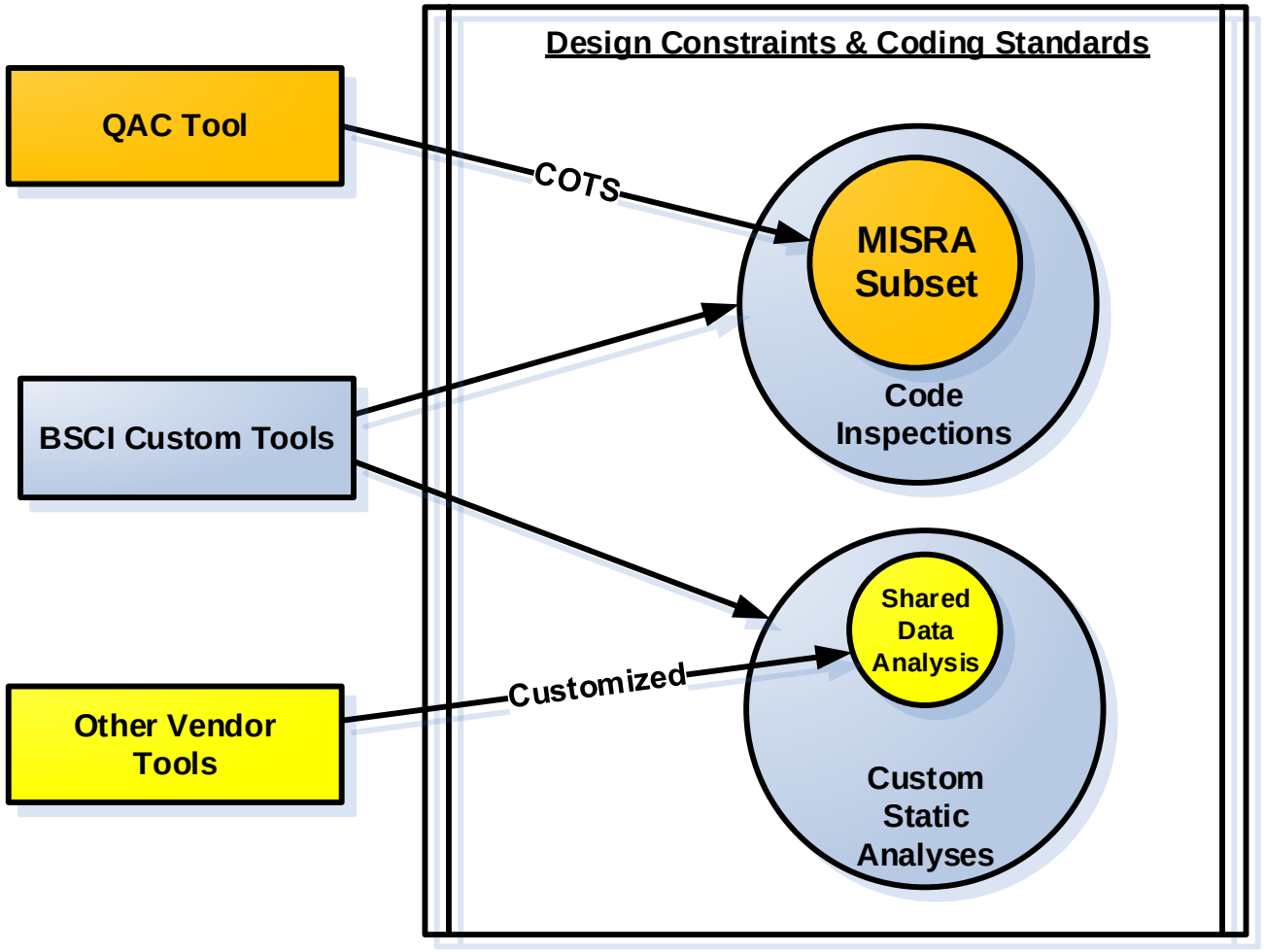
- TP = True Positive
  (defect found was real)
- FP = False Positive
  (defect not real, a false alarm)
- FN = False Negative
  (defect not found)

- Recall = TP / (TP + FN)
- Precision = TP / (TP + FP)

Static Analysis Tool Challenges

- State space of software is large!
  - Infinite at a practical level
  - Properties are “undecidable”
    - Leads to “approximations”
- What is the conclusion?
  - Answer: You Can’t Have it All!
    - Over-approximate (False Positives)
    - Under-approximate (False Negatives)

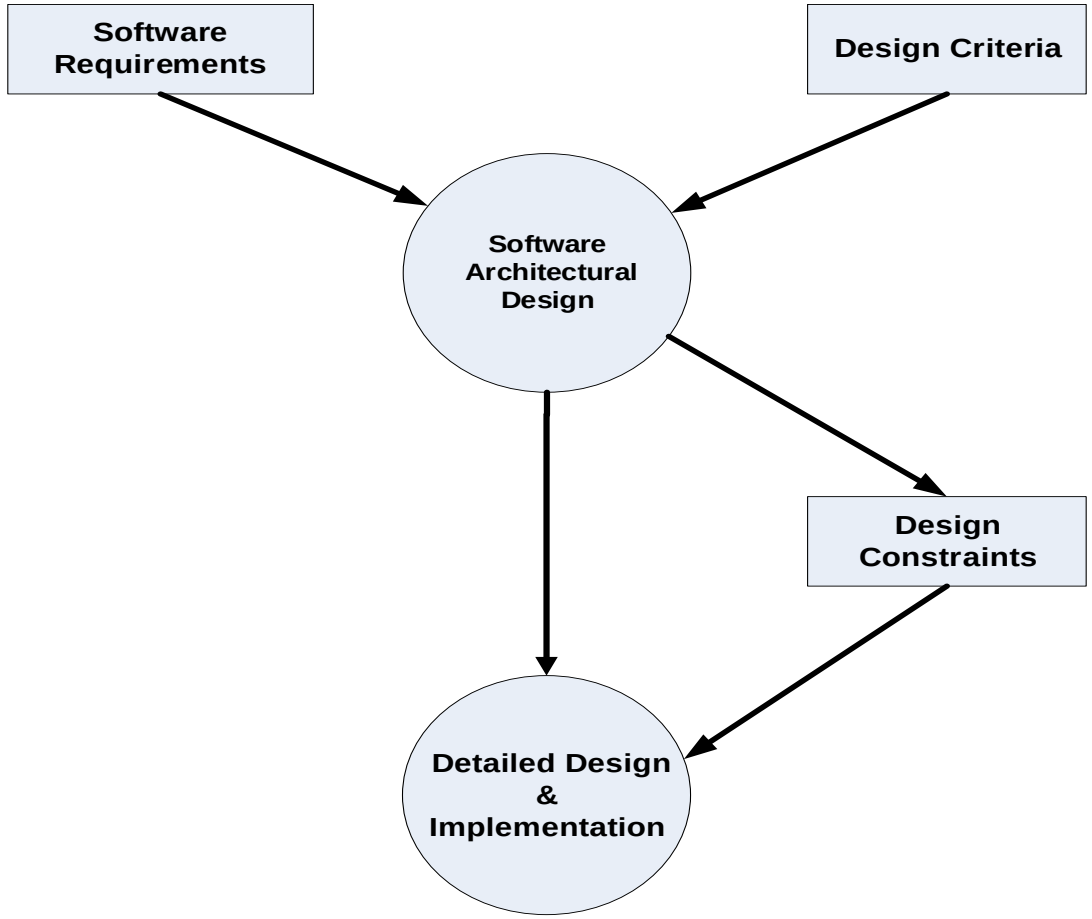
Our Static Analysis Framework



Use of Design Constraints

- Design constraints are mid-level design rules based on design choices made in the software architecture and platform
- Design constraints are an integral part of software risk management:
  - Define the implementation specifics for risk control measures.
  - Provide a systematic way to capture the design rules that uphold the assumptions and limitations of the underlying architecture.
  - Can dictate how something may NOT be done.
  - Provide a means to force an analysis.
    - Use of static analysis for compliance, analysis traceable to constraint.

Design Constraints Visual Context



- Criteria Examples:
  - Maintainability
  - Robustness
  - Reliability
  - Testability

Example of Design Constraints

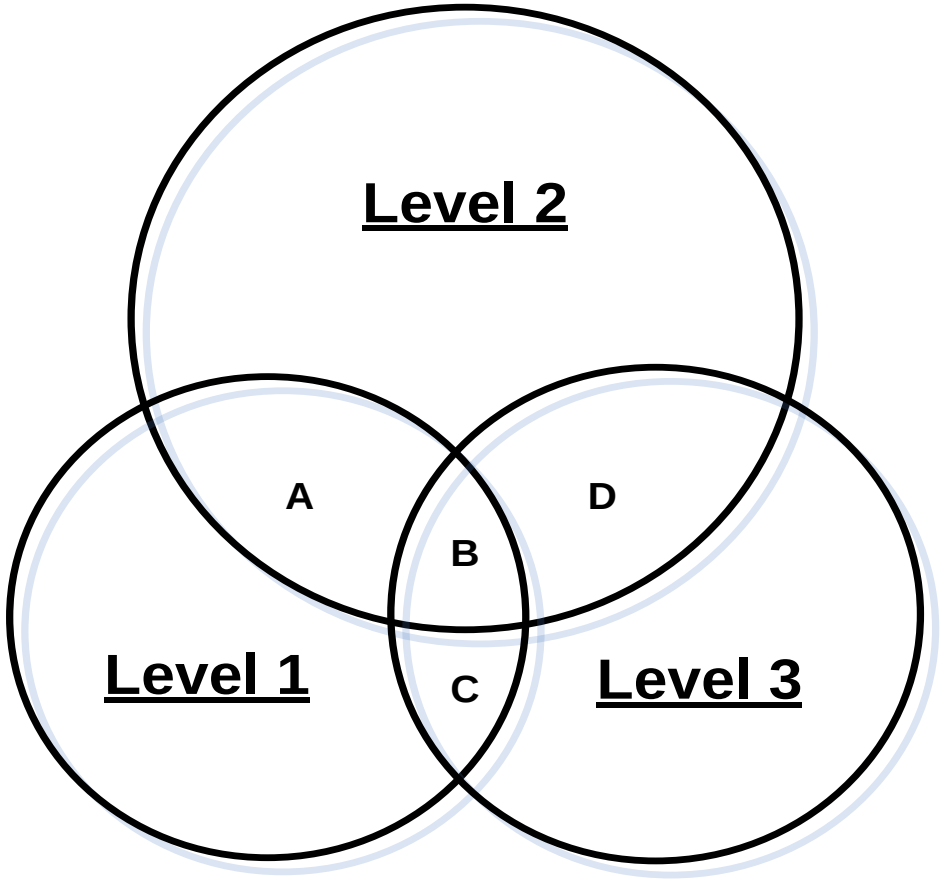
- Design constraint:
  - “All data instances that are accessed from multiple priority levels of threads must be protected with adequate guard constructs.”
- How is this constraint enforced?
  - Stay tuned…

Shared Data Analysis

- Design constraint:
  "All data instances that are accessed from multiple priority levels of threads must be protected with adequate guard constructs."

- Rationale:
  Race conditions result in making assumptions contingent on the sequence or timing of other processes, which leads to unpredictable behavior.

A, B, C, D represent "Shared Data" between different priority levels



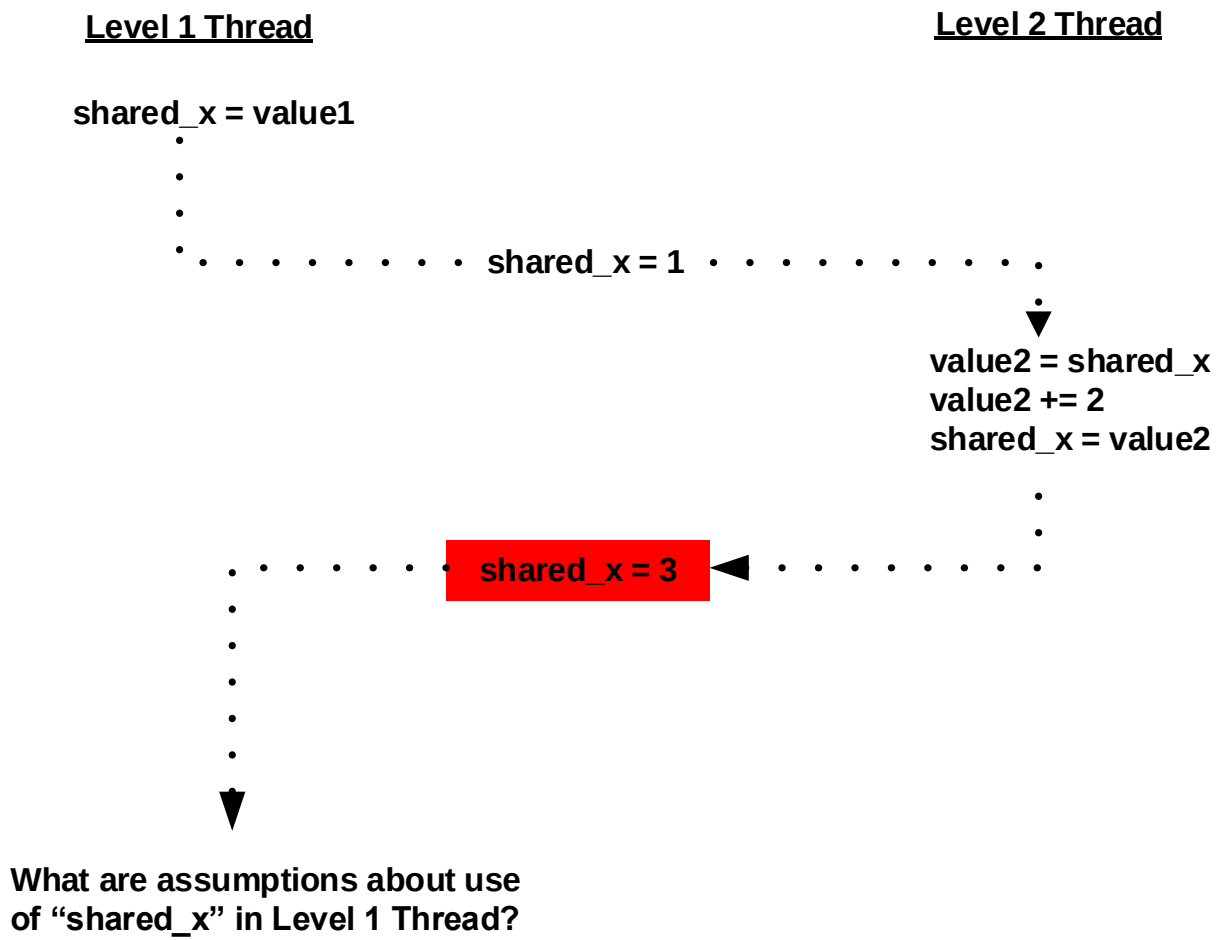
Shared Data Analysis

- A good example where static analysis is better than dynamic testing:
  - Static analysis is more complete than dynamic analysis, since:
    - “general conclusions can be drawn and not just conclusions limited to the particular dynamic test cases that are selected”

      — Nancy Leveson, Safeware, Addison-Wesley, 1995
  - Characterizing one’s software system by static analysis is:
    - “a function that cannot be performed during dynamic methods because of the infinite number of tests that would be required”

      — Neil Storey, Safety-Critical Computer Systems, Addison-Wesley, 1996

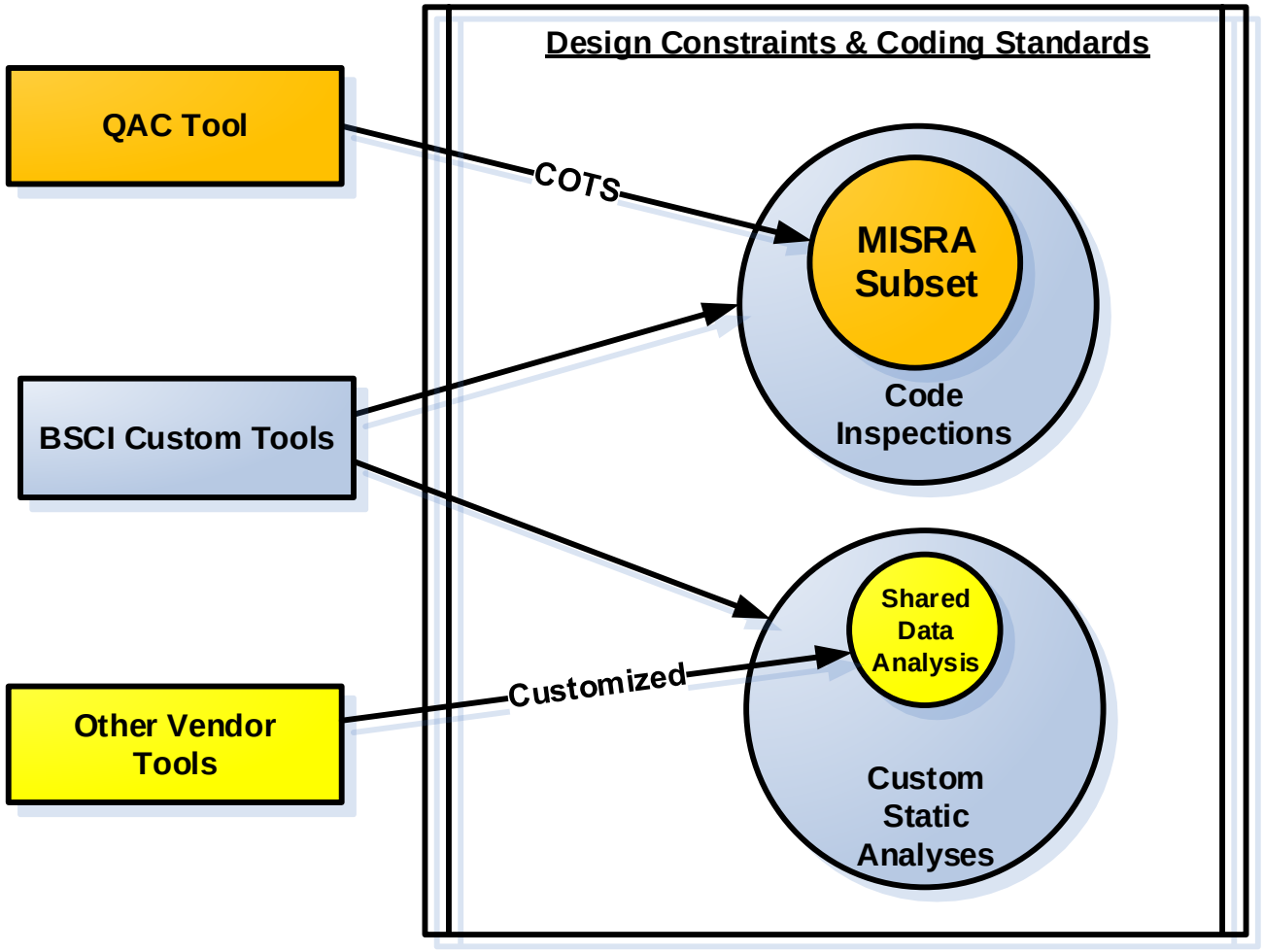
Shared Data Example: Static Analysis Is Best Choice



Why Focus on Design For Static Analysis?

- What are the real issues in Medical Device?
  - Missing / incomplete requirements.
  - Requirements not understood during design.
  - Complexity — interactions between components in a design.
  - Unconstrained, brittle design, maintenance issues.
- For mature software development environments (regulated) static analysis targets the areas of most concern (the domain!)
  - Implementation driven by design constraints — not just language-driven
- So….where do language driven standards like MISRA fit in?

Our Static Analysis Framework — Revisited



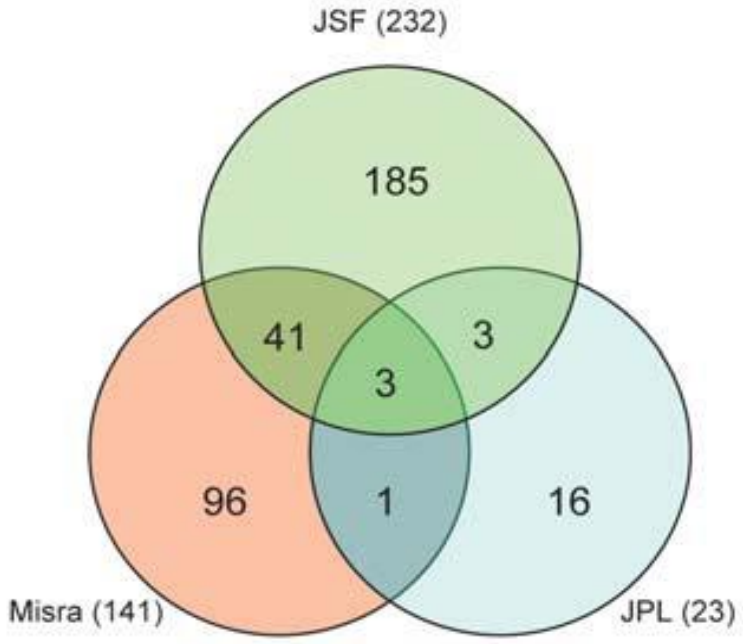
Coding Standards / Rule Sets

- **MISRA:** Motor Industry Software Reliability Association
  - C language rules
  - MISRA C:2004
    - 121 mandatory rules

- **Should all these rules be used?**
  - “it is likely that adherence to the **MISRA** standard as a whole would have made the software less reliable. This observation is consistent with Hatton’s earlier assessment of the **MISRA** C 2004 standard [10]”.

— Cathal Boogerd and Leon Moonen, “Assessing the Value of Coding Standards: An Empirical Study” Report TUD-SERG-2008-017

Intersections of Coding Standards



— Paul Anderson, "How to develop a coding standard for an embedded project", Grammatech, June, 2008

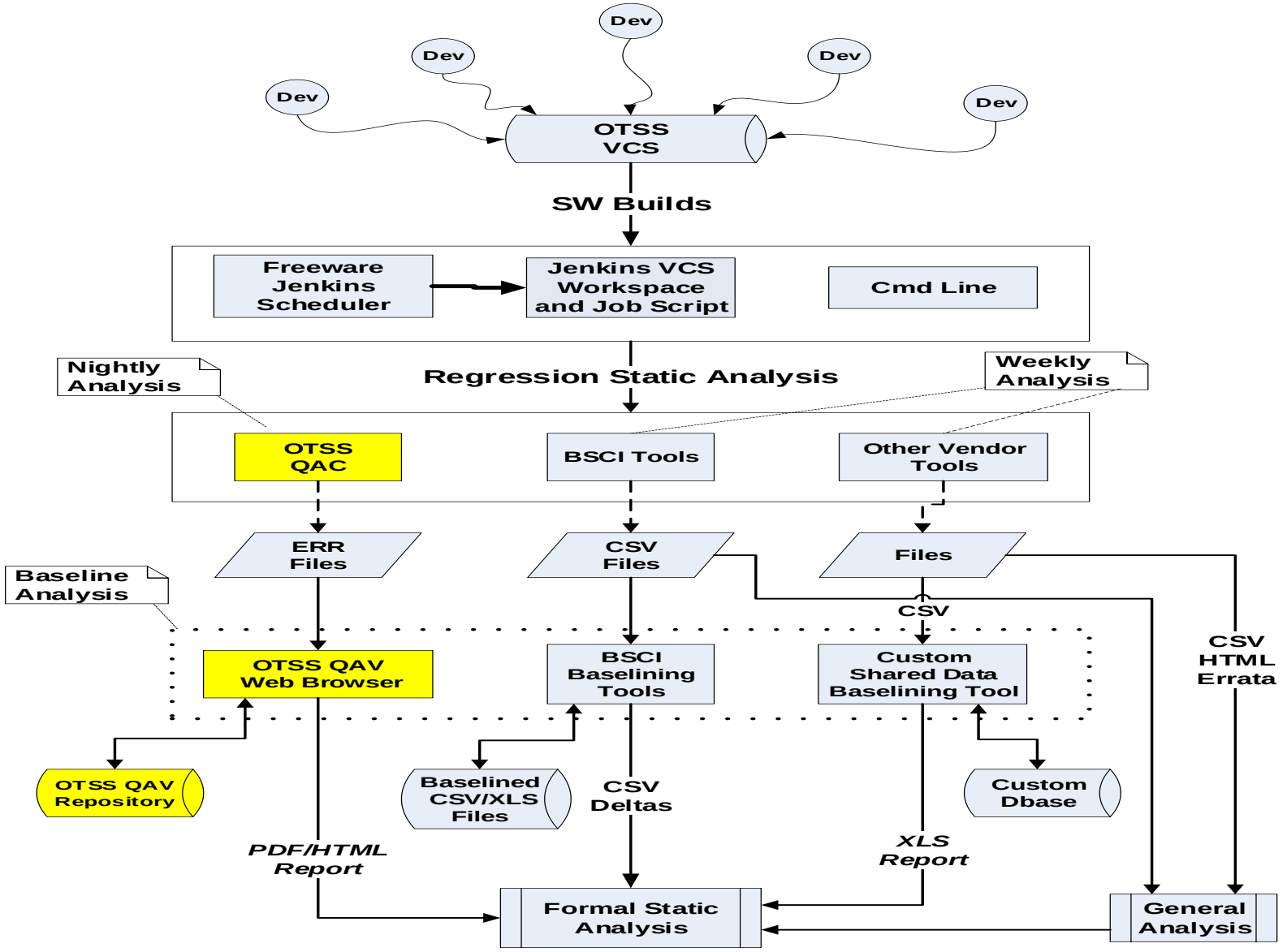
Practical Use of Rule Sets

- More effective when subsets are used that make sense for your domain
  - “The subset should be compact and cover most faults in cited surveys…MISRA is welcomed but with caution. When signal to noise ratios are low….it becomes a weapon…only the tool vendors flourish”

    — Les Hatton, “A measurement based safer subset of ISO C suitable for embedded system development”, Computing Laboratory, University of Kent, 2003

  - “selection of rules that are most likely to contribute to an increase in reliability maximizes the benefit of adherence while decreasing the necessary effort.”

    — Cathal Boogerd and Leon Moonen, “Assessing the Value of Coding Standards: An Empirical Study”, Report TUD-SERG-2008-017.


Dev
Dev
Dev
Dev
Dev
OTSS
VCS
SW Builds
Freeware
Jenkins
Scheduler
Jenkins VCS
Workspace
and Job Script
Cmd Line
Nightly
Analysis
Regression Static Analysis
Weekly
Analysis
OTSS
QAC
BSCI Tools
Other Vendor
Tools
ERR
Files
CSV
Files
Files
Baseline
Analysis
CSV
CSV
HTML
Errata
OTSS QAV
Web Browser
BSCI
Baselining
Tools
Custom
Shared Data
Baselining Tool
OTSS QAV
Repository
Baselined
CSV/XLS
Files
CSV
Deltas
Custom
Dbase
PDF/HTML
Report
XLS
Report
Formal Static
Analysis
General
Analysis

Using Helix QAC for MISRA C

- Verifies compliance with MISRA C coding rules.
- Monitors overall code quality.