Implementing organizational learning cycles for improving software V&V

Forrest Shull, Manuel Mastrofini, Madeline Diep
Fraunhofer Center for Experimental Software Engineering

Supported by NASA's
Software Assurance Research Program (SARP)

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Problem

- Problem statement
  - V&V is of undisputed importance to the development of large software systems
    - 30-50% of total development cost [1]
    - NIST estimates that inadequate software testing infrastructure costs $59.5B annually [2]
  - Cost-effective V&V is mandatory for cost-effective software development
  - While V&V *cost* is easy to measure its *effectiveness* often is not.

[1] Value-Based Management of Software Testing - Ramler R., Biffl S., Grünbacher P. (2006) - Springer - Verlag
[2] The Economic Impacts of Inadequate Infrastructure for Software Testing - (2002) - http://www.nist.gov

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Our Solution

- A framework for characterizing, assessing and improving V&V activities
- Goal of the framework:
  - Evidence-based, iterative improvement of V&V
  - Tailored to the specific context
  - Providing outputs useful for decision support

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

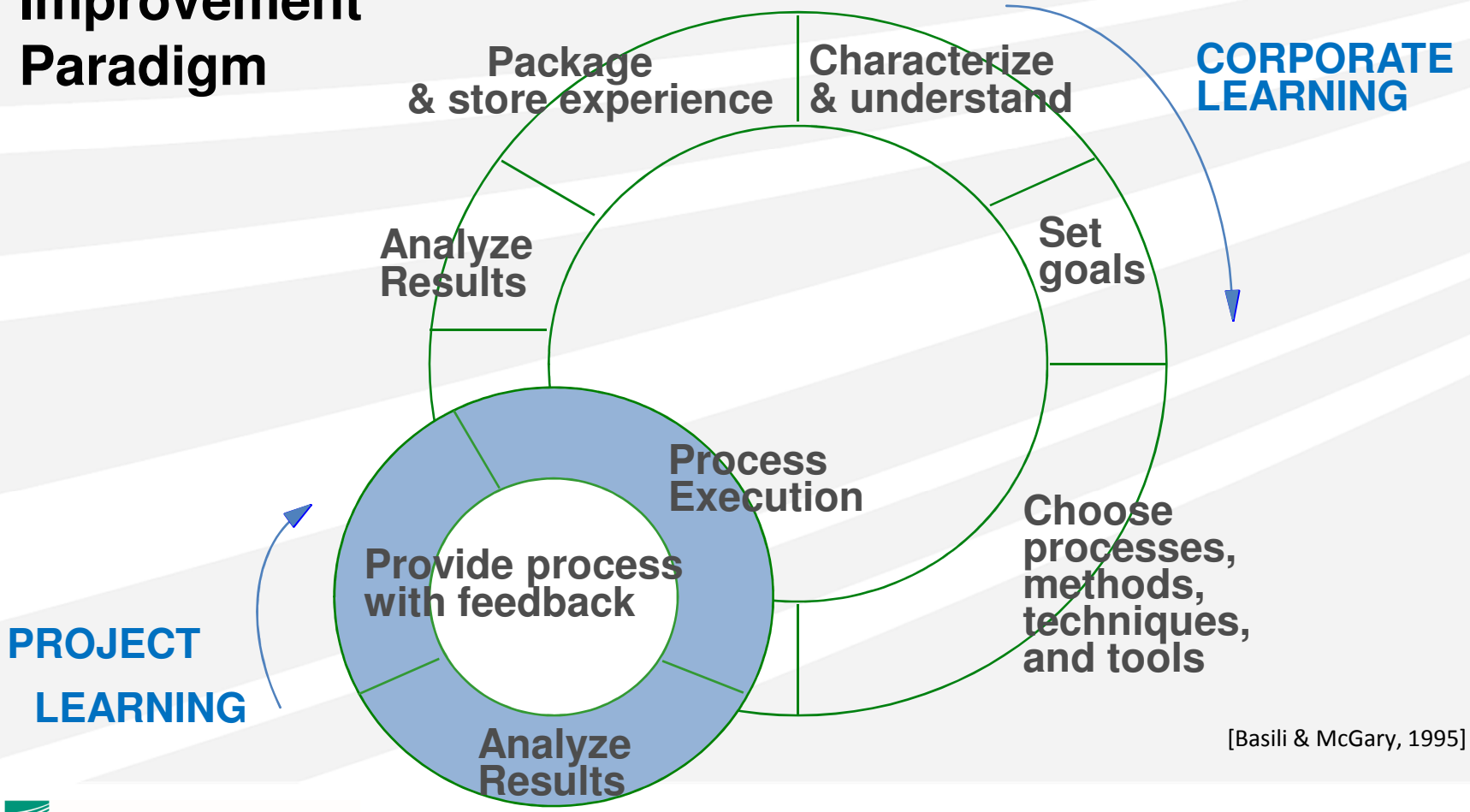
Outline

- Foundation: the Quality Improvement Paradigm (QIP)
- Our implementation of QIP
  - Example application: Software inspections & reviews
  - Resulting heuristics and decision support
- Conclusions

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Foundation: QIP

QIP: Quality Improvement Paradigm



[Basili & McGary, 1995]

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Why Focus on Inspections?

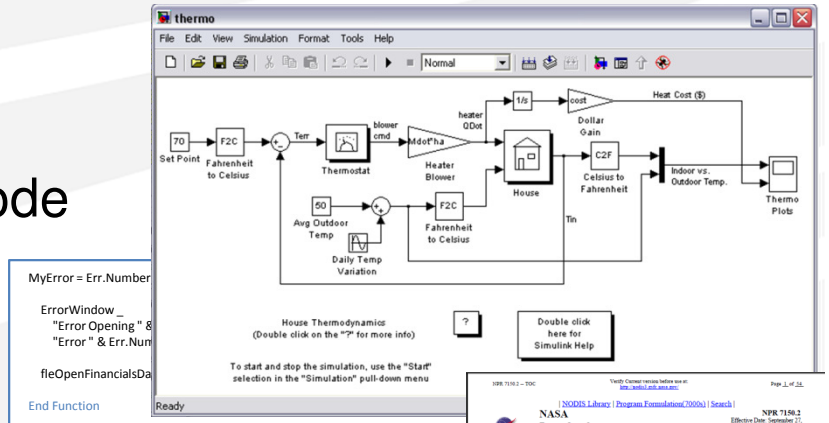
- Inspections / reviews are:
  - Part of the system and software engineering culture within DOD, NASA, and other organizations developing safety- and mission-critical software
  - Still among the most effective V&V practices, capable of removing 60-90% of extant defects if performed with appropriate rigor [1]

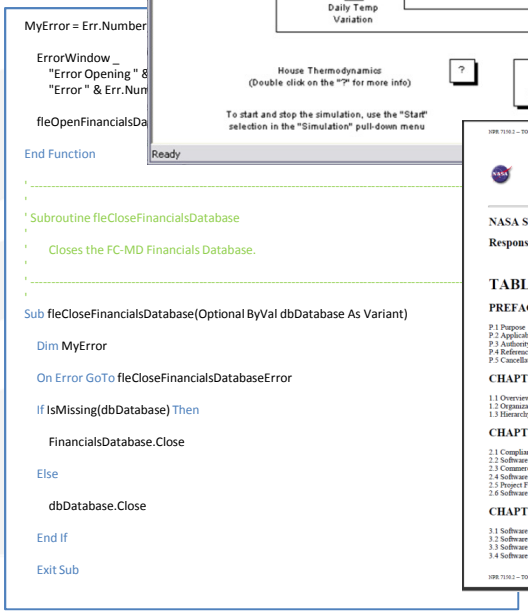
[1] Shull, F., Basili, V. R., Boehm, B., Brown, A. W., Costa, P., Lindvall, M., Port, D., Rus, I., Tesoriero, R., and Zelkowitz, M. V., "What We Have Learned About Fighting Defects," Proc. IEEE International Symposium on Software Metrics (METRICS02), pp. 249-258. Ottawa, Canada, June 2002.

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

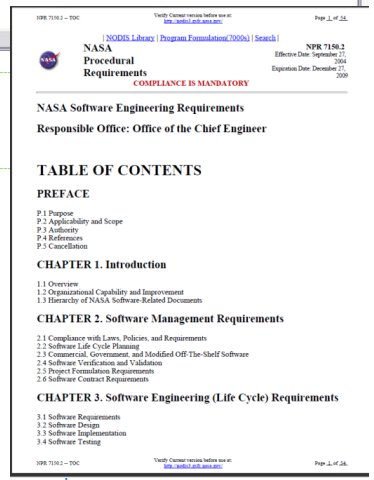
Inspections for Different Artifacts…

- Recent experiences working with NASA teams have included application of inspections to:
  - Software development artifacts (requirements, design, code, test plans…)
  - Complex electronics / logic diagrams
  - Formal and semi-formal models for code generation
  - Safety-critical code segments
  - Standards
- The context of these experiences:
  - US Government agency
  - Safety- and mission-critical software





```
MyError = Err.Number

   ErrorWindow _
      "Error Opening " &
      "Error " & Err.Num

   fleOpenFinancialsDa

End Function

'
' ---------------------------------------------------
'
' Subroutine fleCloseFinancialsDatabase
'
'     Closes the FC-MD Financials Database.
'
' ---------------------------------------------------
'
Sub fleCloseFinancialsDatabase(Optional ByVal dbDatabase As Variant)

   Dim MyError

   On Error GoTo fleCloseFinancialsDatabaseError

   If IsMissing(dbDatabase) Then

      FinancialsDatabase.Close

   Else

      dbDatabase.Close

   End If

   Exit Sub
```

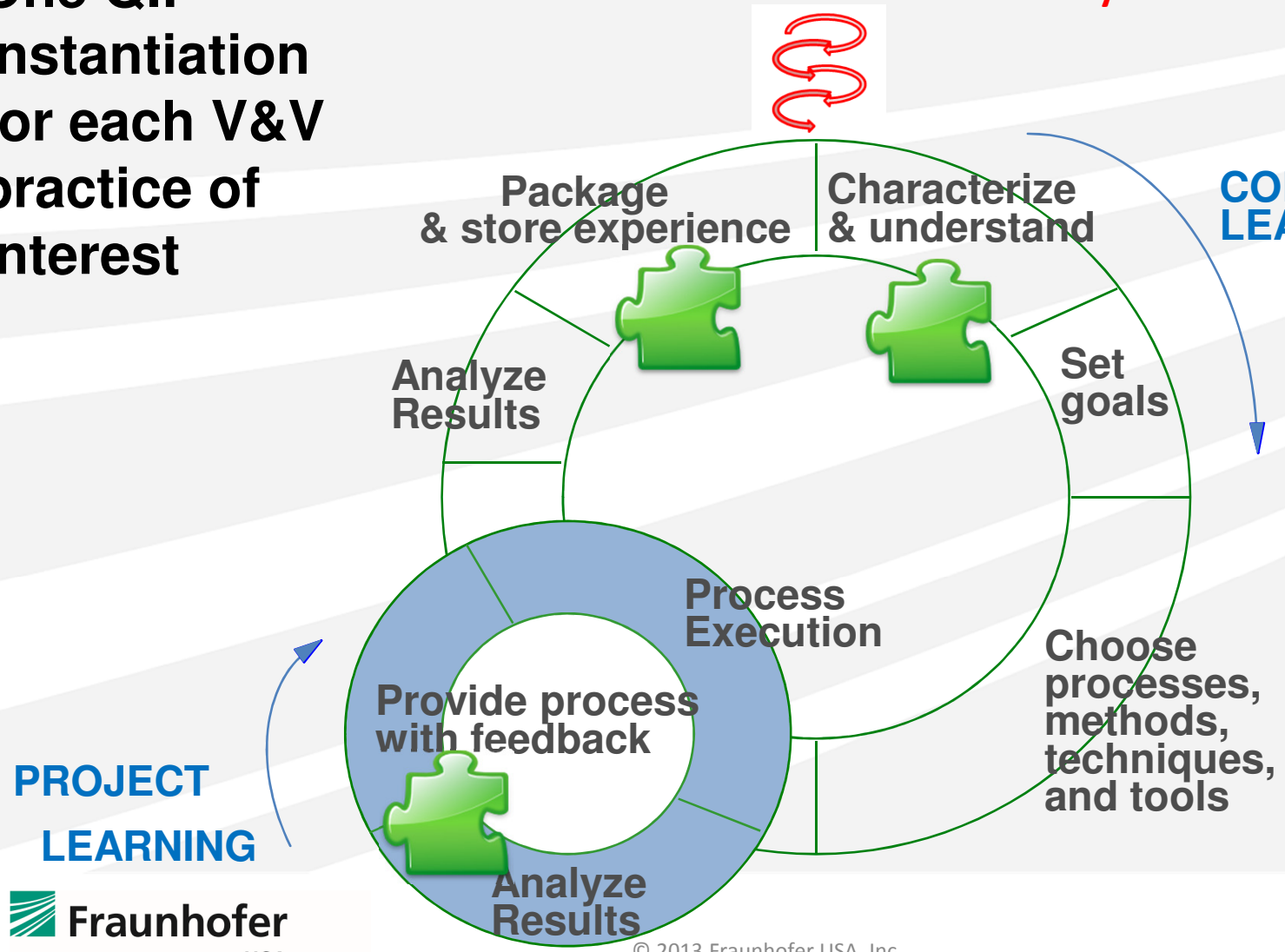

NASA Procedural Requirements
COMPLIANCE IS MANDATORY
NPR 7150.2
NASA Software Engineering Requirements
Responsible Office: Office of the Chief Engineer
TABLE OF CONTENTS
PREFACE
CHAPTER 1. Introduction
CHAPTER 2. Software Management Requirements
CHAPTER 3. Software Engineering (Life Cycle) Requirements

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.

Our Implementation of QIP

One QIP instantiation for each V&V practice of interest



Fraunhofer USA
Center for Experimental Software Engineering

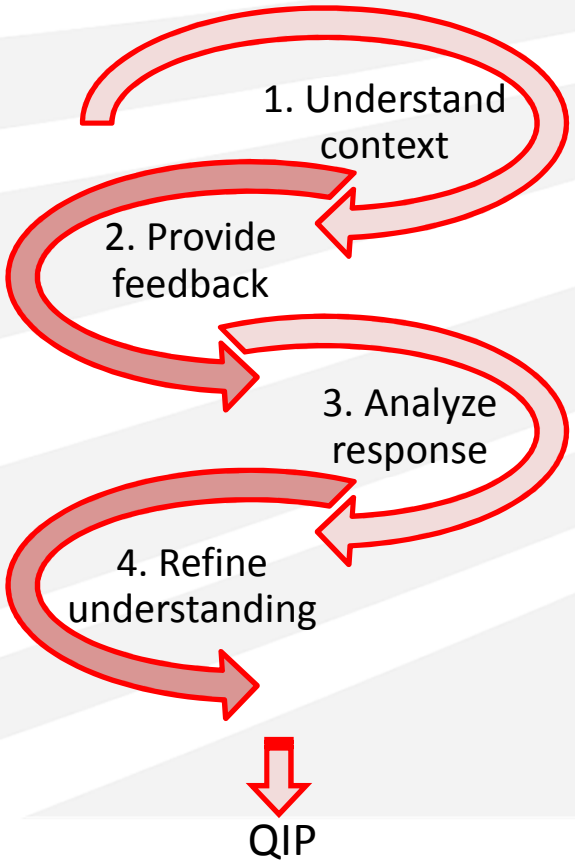
© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Preliminary Assessment

We augment the traditional QIP cycle with an initial process assessment.

Understand context; formulate best practice guidance as a “Health Check”

Assess actual processes against Health Check; provide **preliminary, qualitative and fast** feedback



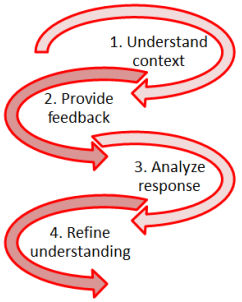
Analyze the response of decision makers to the provided feedback

Refine context understanding and Health Check

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Preliminary Assessment



Example: Health Check for Inspection Process Improvement (Context: Safety-critical hardware interlocks)

1. Systematically gathered context info
   - From industry and local standards, guidebooks, and other material.
   - 4 key areas: Processes, quality attributes, skills, artifacts
   - Formulate questions and expected answers for each area, e.g.:

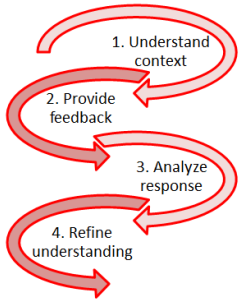
     Q: Which members of the team are generally required during a review?

     A: One system requirement engineer (requirements/user perspective), one specialized engineer for each subsystem/team involved in the artifact under examination, etc...

   - Full health check available at http://fc-md.umd.edu/eb

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Preliminary Assessment



Example: Health Check for Inspection Process Improvement

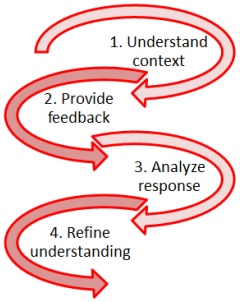
2. Assess practices
   - Analysis of project documents and process documentation
   - Telcons to further elicit info and clarifications
     - We detected 4 potential mismatches and proposed 4 fixes, e.g.:

       Mismatch: "*The Requirements Document is listed as an input to the Critical Design Review (CDR), but no formal change request documents or other forms of feedback are shown as possible outputs.*"

       Fix: "*The team should be open to reporting requirements problems and generating formal change requests during CDR. If the team is not allowed to modify documents from previous phases, it should be noted explicitly.*"

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

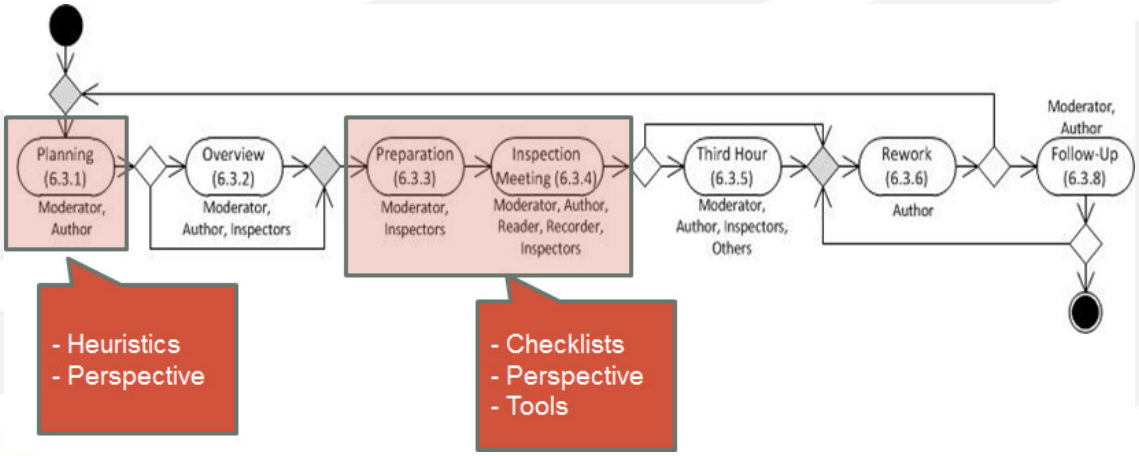
Preliminary Assessment



Example: Health Check for Inspection Process Improvement

3. We scored ourselves against the team's response
   - 2 fixes accepted (which led to standard updates)
   - 1 fix rejected (update to the health check and adjusted understanding of the context)
   - 1 fix forwarded to another team

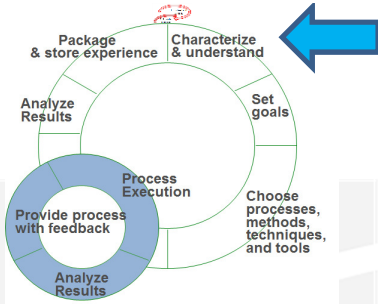
4. We understood teams would have benefited from:
   - Use of checklists during inspections
   - Recommendations for planning inspections
   - Implementation of Perspective Based Inspection



Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

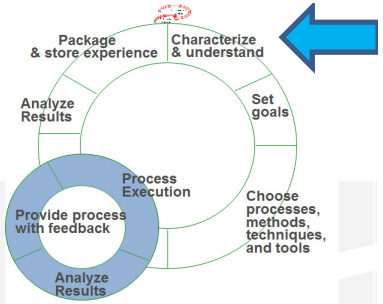
Modeling Understanding



- While executing the QIP cycle, capture findings quantitatively in a mathematical model
  - Impact Vectors: “vectors” describing the expected “impact” in context
    - Each vector represents one V&V practice
  - Dependent variables: performance of the V&V practice
    - Depends on quality priorities of the project
    - E.g. effectiveness, efficiency…
  - Independent variables: factors that decision makers manipulate in order to control the dependent variables
    - Consider factors where we are likely to find enough data
  - Synthetizes results over multiple QIP iterations
  - Enables a structured way of planning, enacting, evaluating and packaging V&V improvements

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Modeling Understanding



- Without a project-specific baseline, consider starting from an organization-wide model
  - Our research team compiled a database from 2500+ inspections across the organization over multiple years
  - Hypothesis-testing and data exploration to identify important factors, formulate heuristics
- Relevant variables include:
  - Type of artifact being inspected (requirements, design, code, test)
  - Inspection team size
  - Meeting length
  - Amount of material inspected per hour

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Initial Heuristics: Method

- Begin with factors that planners can control, and hypothesize regarding limits:

Team size:	**Page rate:**
Too small – miss important expertise Too large – drive up costs, dampen discussion => Hypothesize optimal range: 4 to 6	Too small – miss interrelations Too large – thorough review impossible => Hypothesized optimal range: 10 to 30 requirement-pages.

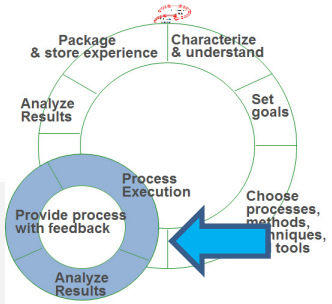
- Use data to statistically test those limits:

Team size: Avg results for all projects: If followed: **14** defects detected If not: **7** defects detected *Significant, p<0.0005*	**Page rate:** Avg results for all projects: If followed: **14** defects detected If not: **6.5** defects detected *Significant, p<0.0005*

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

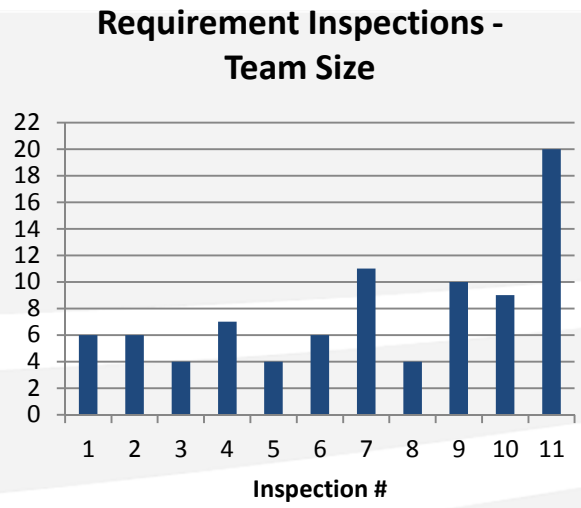
Executing the Practice

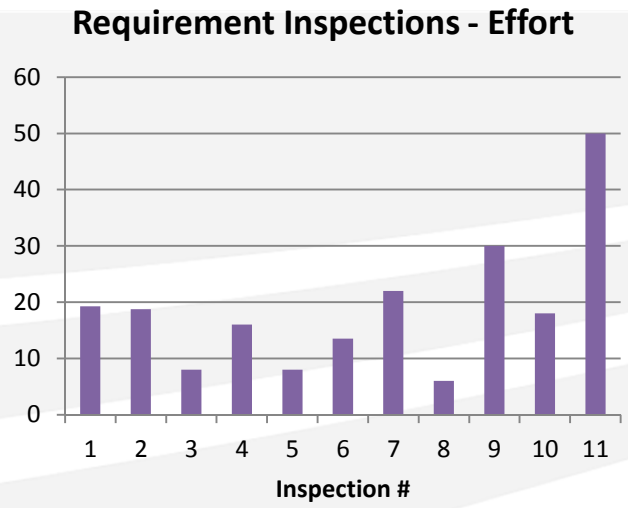


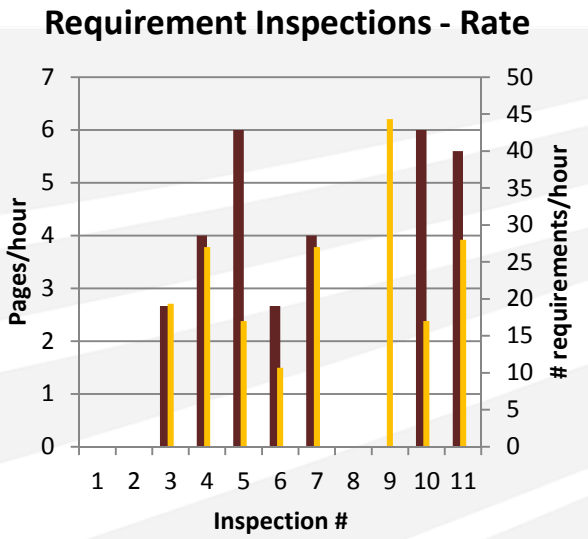
- As V&V practices are applied on the project, the team can build up a context-specific baseline against which the heuristics can be tested.
- In case of anomalies, investigate whether project practice or heuristics need to be updated.

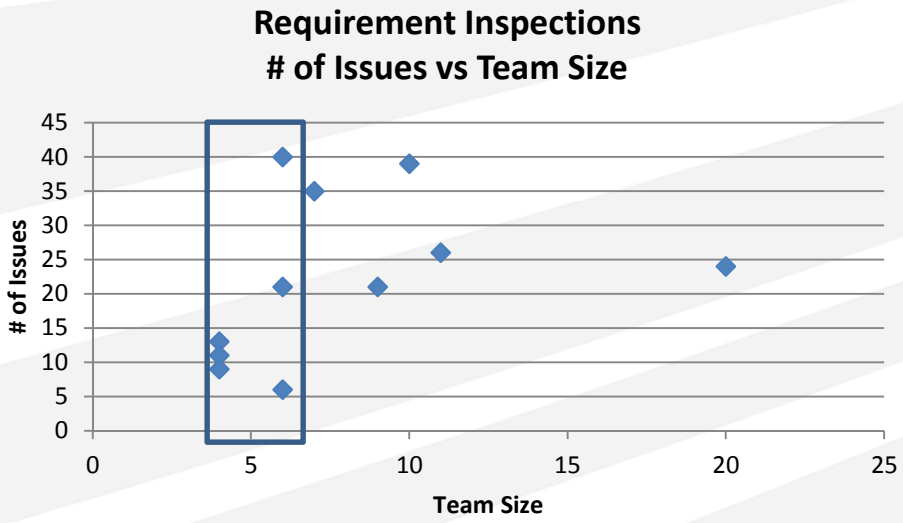
© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

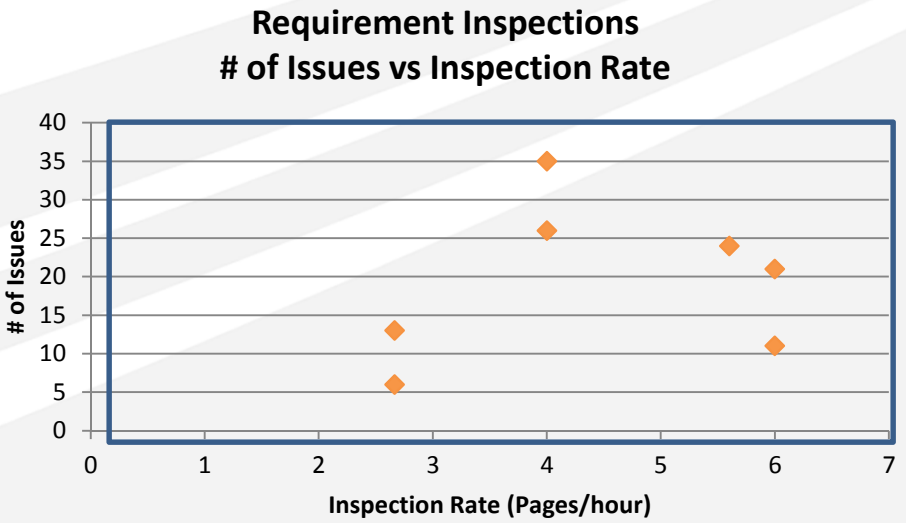
Executing the Practice







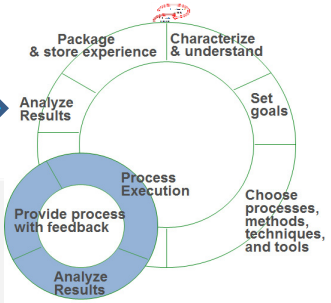




Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Analysis & Packaging



- As project data accumulates, continue to test project findings against heuristics and update as needed.
- Also, use project retrospectives and other opportunities to analyze the organizational data periodically.

- Outcome:
  - Update heuristics (based on larger dataset) and/or
  - Provide support for dissemination and decision support

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Updating Recommendations

- Organization-wide databases also enable testing for trends over time.

Team size:	**Page rate:**
Too small – miss important expertise Too large – drive up costs, dampen discussion => Heuristic = 4 to 6	Too small – miss interrelations Too large – thorough review impossible => Heuristic = 10 to 30 pgs for reqts, 20 to 40 pages for test plans, etc.

- Our database confirms that the heuristics are still good predictors of effective inspections.

Team size: Avg results for all projects:	**Page rate:** Avg results for all projects:
If followed: **14** defects detected If not: **7** defects detected *Significant, p<0.0005*	If followed: **14** defects detected If not: **6.5** defects detected *Significant, p<0.0005*

- Yet, fewer projects are able to follow them:

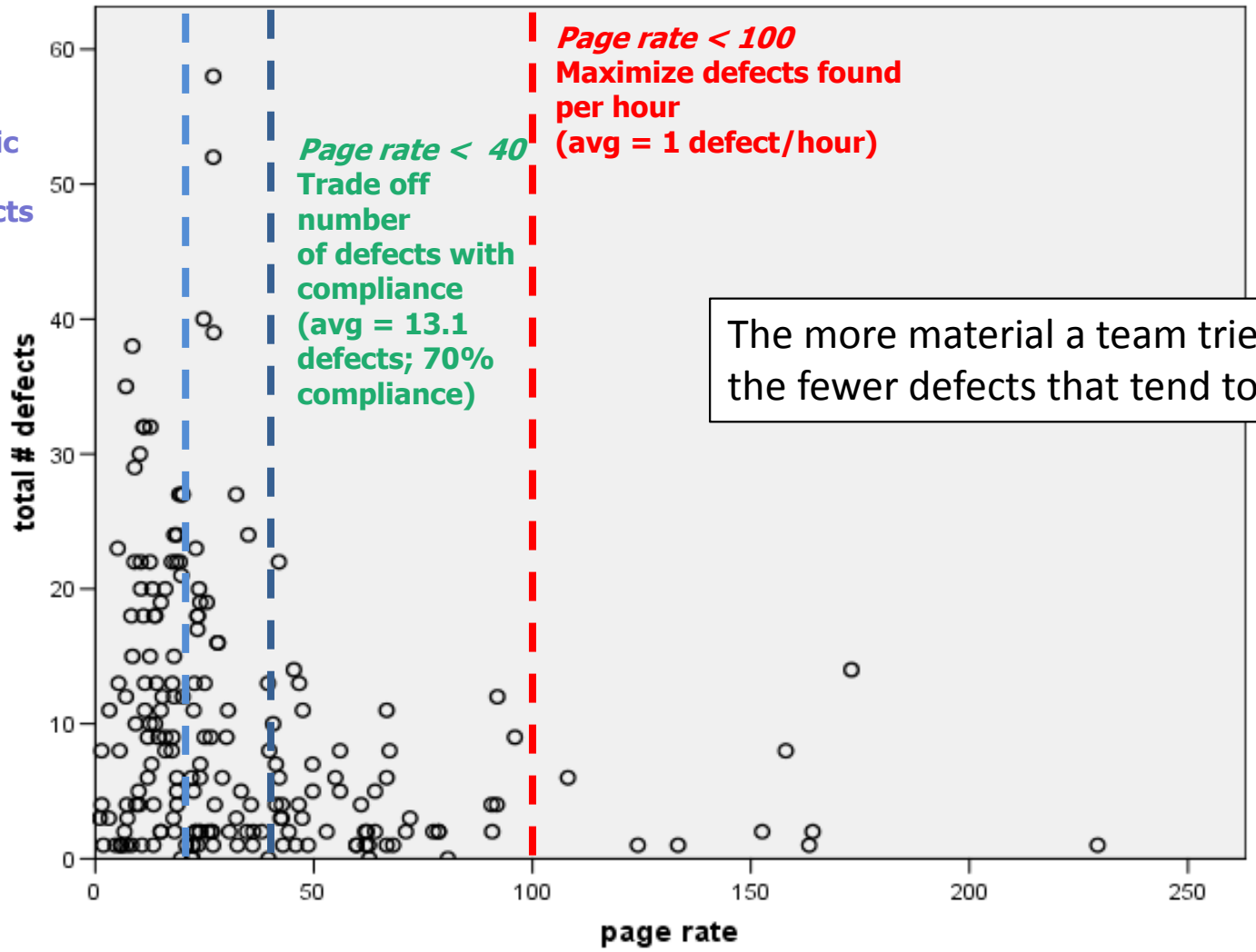
Team size: 10% of contemporary projects followed	**Page rate: 15%** of contemporary projects followed

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Refining Heuristics

Page rate < 20
Original heuristic – maximize number of defects (avg = 15.4)



The more material a team tries to cover, the fewer defects that tend to be found.

Data from design inspections

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Latest Heuristics: Results

For maximizing *effectiveness*:

- Team size:
  - 3-6 persons
  - More likely to need larger teams earlier in lifecycle, but limit to 10
- Page rate:
  - For project plans: 15 pages / hour
  - For requirements: 15 pages / hour
  - For architecture and design: 20 pages / hour
  - For source code: 10 pages / hour
  - For test plans & test procedures: 20 pages / hour
- Meeting length: 2 hours or less

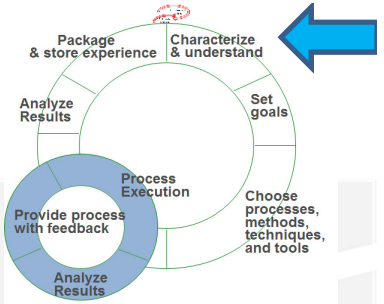
© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Dissemination and Decision Support

- Organization-level heuristics are useful for inclusion in organization-wide standards and guidebooks.
  - Able to provide the *rationales* as well as the requirements.
- We are currently working with NASA HQ to incorporate our findings into an update to the Agency-wide standard
- Don't lose the idea that projects still need to perform their own tailoring. E.g.,
  - Team size:
    - The moderator **shall** identify key stakeholders in the work product, as a basis for the selection of inspectors.
    - The inspection team **shall** consist of a minimum of three inspectors.
    - It is recommended that team size not exceed six members…
  - Page rate:
    - The moderator **should** limit the amount of work product to be inspected in order to maintain an acceptable inspection rate.
    - Prior data and experiences suggest a starting rate for this type of inspection of at most 15 pages per hour.

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Iterating…



- Planning V&V for future projects now can begin from an even larger organizational experience base
- Different and more accurate analysis techniques can be created and used
  - Data clustering, filtration and reduction to match the context of new projects
  - Some data mining techniques can provide more precise heuristics (e.g. combination of Weka algorithms)
  - Mathematical optimization to refine the heuristics

	Team size	Meeting length	Page rate
Requirements	[5;8]	[1;2.5]	[10;30]
Design	[4;7]	[1;2.5]	[20;45]
Code	[3;6]	[1;2.5]	[13.3;20]
Test	[3;6]	[1;2.5]	[20;45]

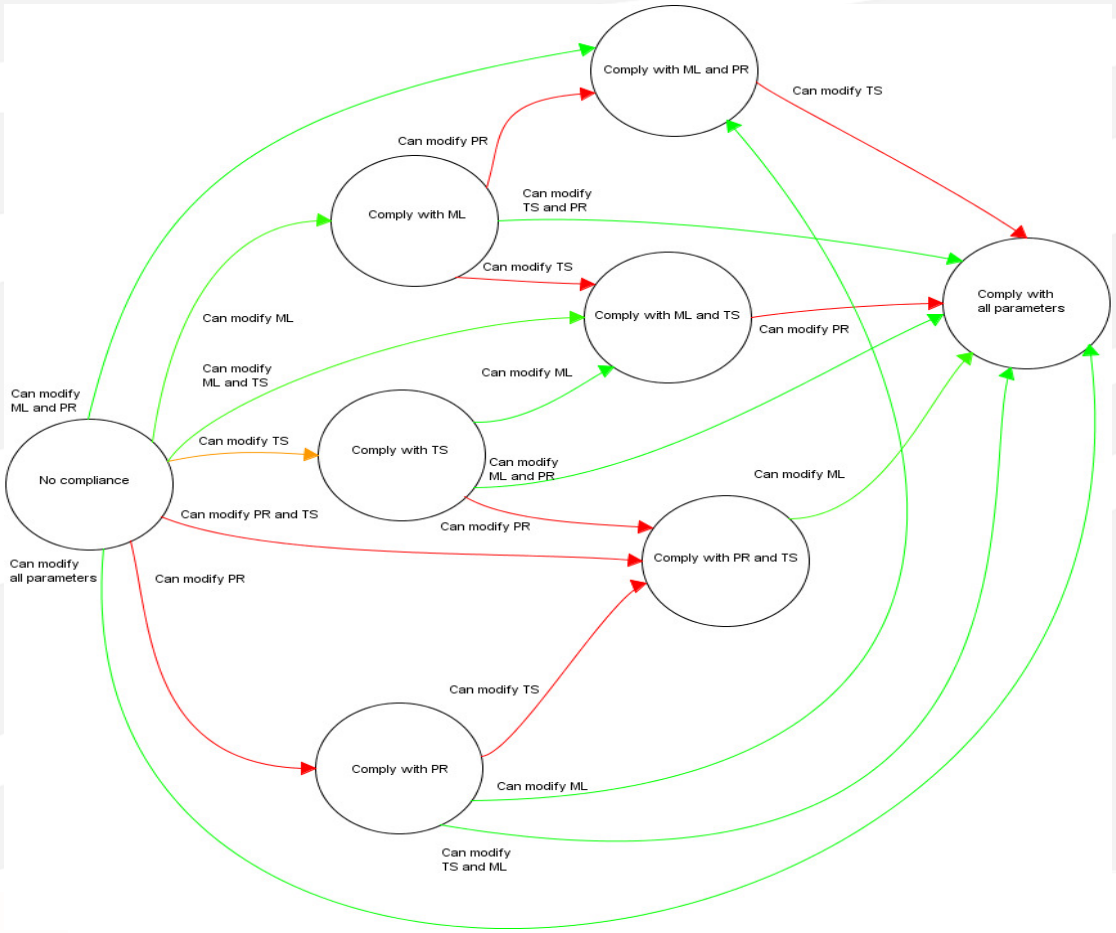
	Team size	Meeting Length	Page Rate
Requirements	=	=	=
Design	[3;7]	[1;2.25]	[5;25]
Code	[2-5]	[0.75;2]	[10;30]
Test	=	=	=

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Additional Decision Support

- More advanced models to support decision making can be created



Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

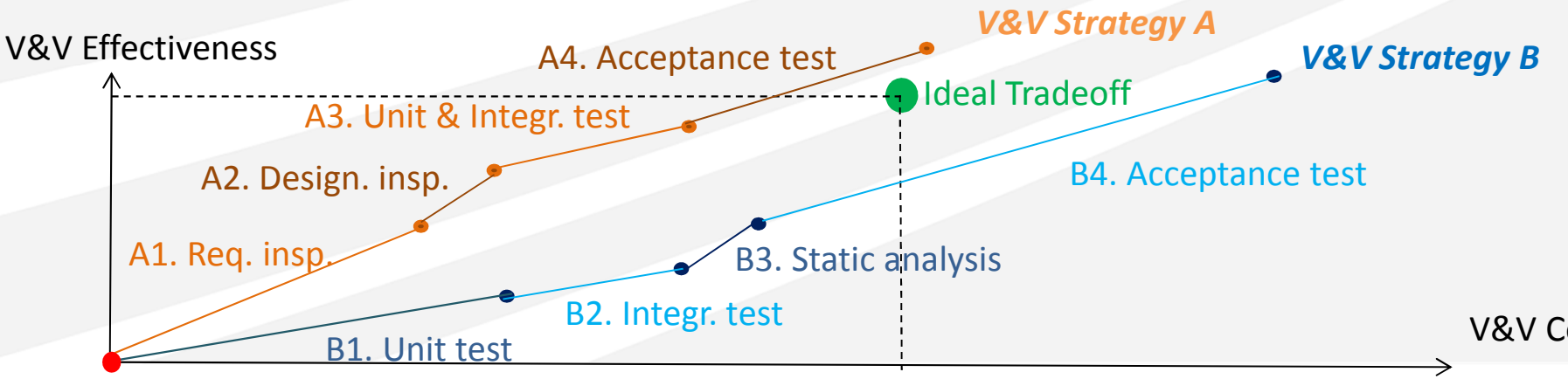
Other Experiences

- Applicability of the method is not just limited to this environment.
- **Context: Small IT company**
  - Multiple iterations through QIP – Led to achieving CMMI Level 5 (and still improving!)
  - Goal: V&V cost reduction without lowering quality
  - Formulating predictive models of defect density – during planning phase for each iteration. Includes:
    - Project factors that influence defect density,
    - Impact vectors that influence defect removal
- **Context: Large international defense company**
  - Goal: reduce project time at reasonable cost
  - Employing QIP
    - Defined a new model-based development process model
    - Defined a measurement strategy for the new process
    - Experimenting with process activities in a confined environment
    - Tuning process activities and experimenting with them in production

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Conclusions

- This talk has demonstrated an approach to evidence-based & iterative improvement of V&V.
  - Useful for decision support in planning
  - Useful for assessments of prior V&V activities
- Our inspection heuristics are testable in other environments.
- Our future work:
  - Formalizing impact vectors for other V&V technologies, e.g. different testing approaches, static analysis techniques
  - Investigating *composability* within a given context
  - Experimenting with impact vectors as the raw materials for assurance cases



Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering

Questions?

Feel free to contact:

Forrest Shull
fshull@fc-md.umd.edu
240-487-2904

Fraunhofer USA
Center for Experimental Software Engineering

© 2013 Fraunhofer USA, Inc.
Center for Experimental Software Engineering