Improved Software Testing for Open Architecture

Valdis Berzins

Professor of Computer Science, Naval Postgraduate School

Agenda

- **Research Context**
  - U.S. Navy Open Architecture
  - Problem Addressed & Proposed Solution
- **Profile-Based Automated Software Testing**
  - Automated Testing Process
  - HFPM Functional Concept
  - Acquisition Community HFPM Employment
  - Deriving HFPs from Historical Data
  - Validating HFPs
  - Deriving Stress-Testing HFPs from Historical Models
- **Conclusions**

U.S. Navy Open Architecture

- **A multi-faceted strategy for developing joint interoperable systems that adapt and exploit open system design principles and architectures**

- **OA Principles, processes, and best practices:**
  - Provide more opportunities for completion and innovation
  - Rapidly field affordable, interoperable systems
  - Minimize total ownership cost
  - Maximize total system performance
  - Field systems that are easily developed and upgradable
  - Achieve component software reuse

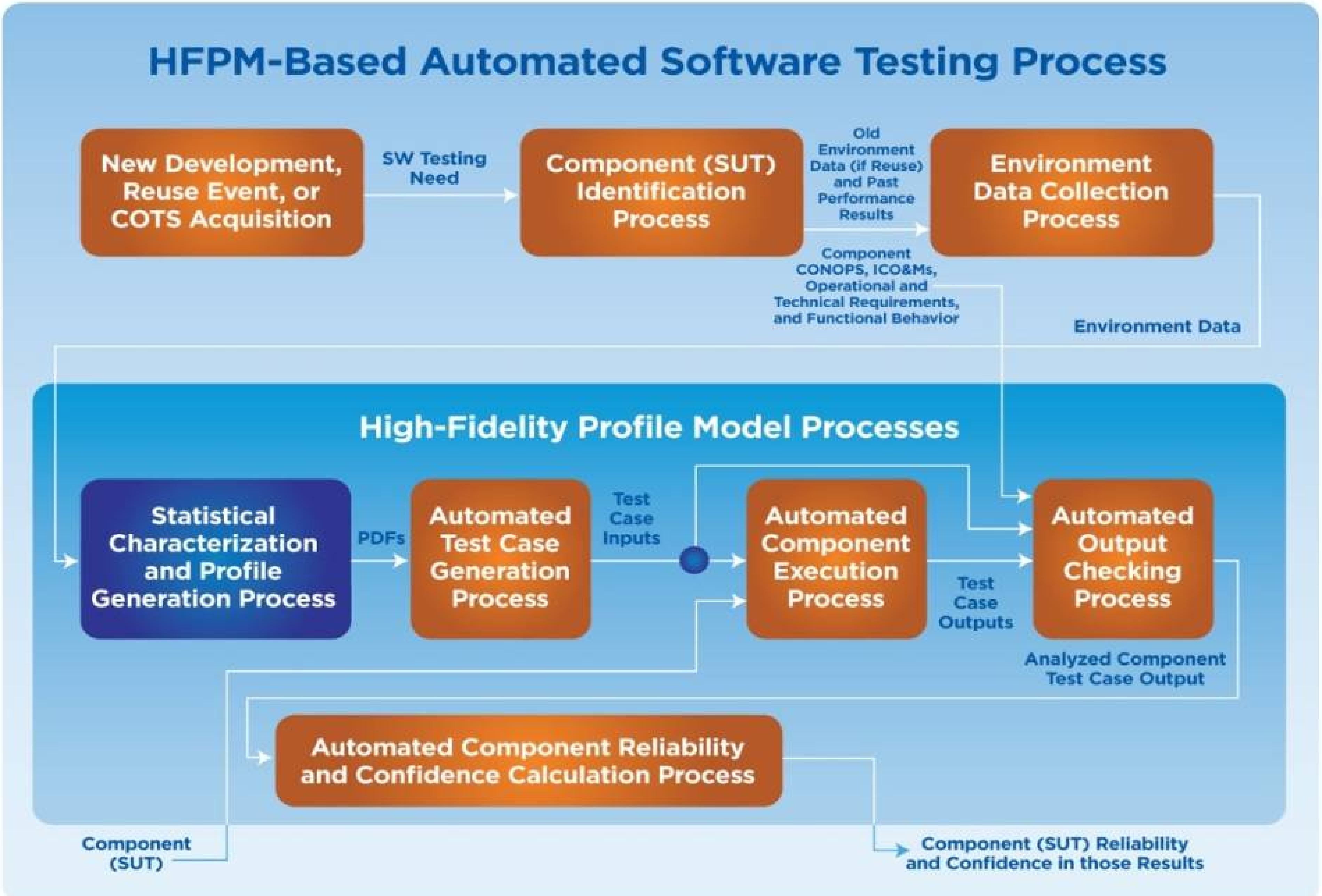
Problem and Proposed Solution

- **Traditional U.S. Navy Software T&E practices will limit many benefits of OA**
  - It will be virtually impossible to field frequent and rapid configuration changes
- **New Testing Technologies, Processes & Policies are Needed**
  - Determine how to Safely Reduce Amount of Testing Required (Berzins, 2009)
  - **Transition from Manual Testing to Profile-Based Automated Statistical Testing**

HFPM-Based Automated Software Testing Process

- Software's requirements, CONOPS, architecture standards, and interfaces used to establish boundaries for component testing
- Component's external environment analyzed and characterized
- Environment statistical model (HFPM) used to automatically generate test cases, execute test cases and check component outputs

Effective for new development efforts, reuse, or COTS acquisition

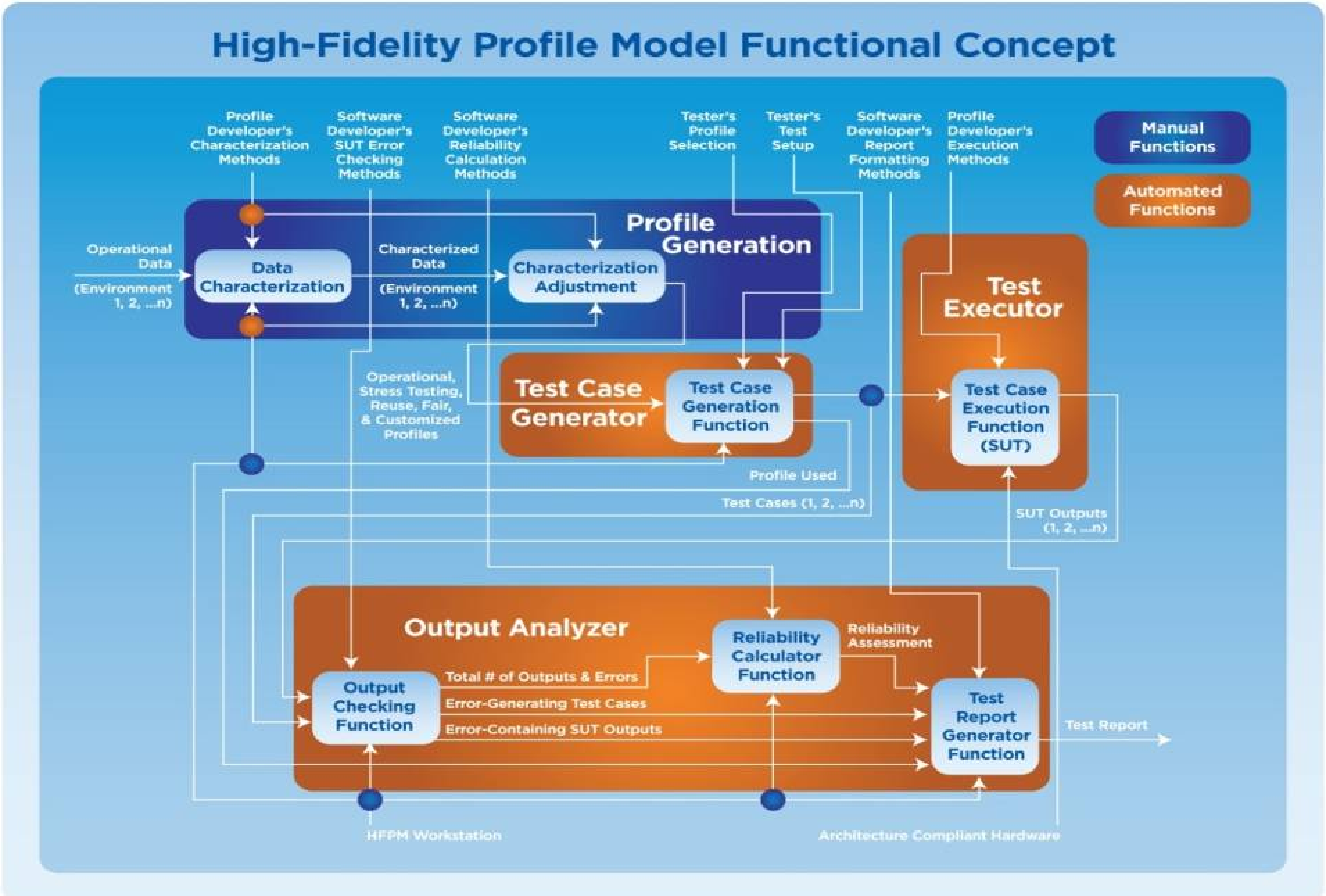

HFPM-Based Automated Software Testing Process
New Development, Reuse Event, or COTS Acquisition
SW Testing Need
Component (SUT) Identification Process
Old Environment Data (if Reuse) and Past Performance Results
Environment Data Collection Process
Component CONOPS, ICO&Ms, Operational and Technical Requirements, and Functional Behavior
Environment Data
High-Fidelity Profile Model Processes
Statistical Characterization and Profile Generation Process
PDFs
Automated Test Case Generation Process
Test Case Inputs
Automated Component Execution Process
Test Case Outputs
Automated Output Checking Process
Analyzed Component Test Case Output
Automated Component Reliability and Confidence Calculation Process
Component (SUT)
Component (SUT) Reliability and Confidence in those Results

High-Fidelity Profile Model (HFPM)

- **HFPM utilizes statistical environment characterizations to automatically test software**
  - Profile-Based => Optimized test case coverage
  - Automated => High #s of test cases => High confidence in results
  - Concept is scalable from component to system level

- **Model is reusable, following component throughout life-cycle**
  - Profiles can be modified to check component behavior in multiple environments and at different stress-levels
  - Model can be used to check multiple component configurations during iterative development
  - Model architecture is reusable

- **HFPM developed to accompany each component during testing**
  - Initial investment up front enables long-term benefits including reduction in testing time and more effective & efficient testing

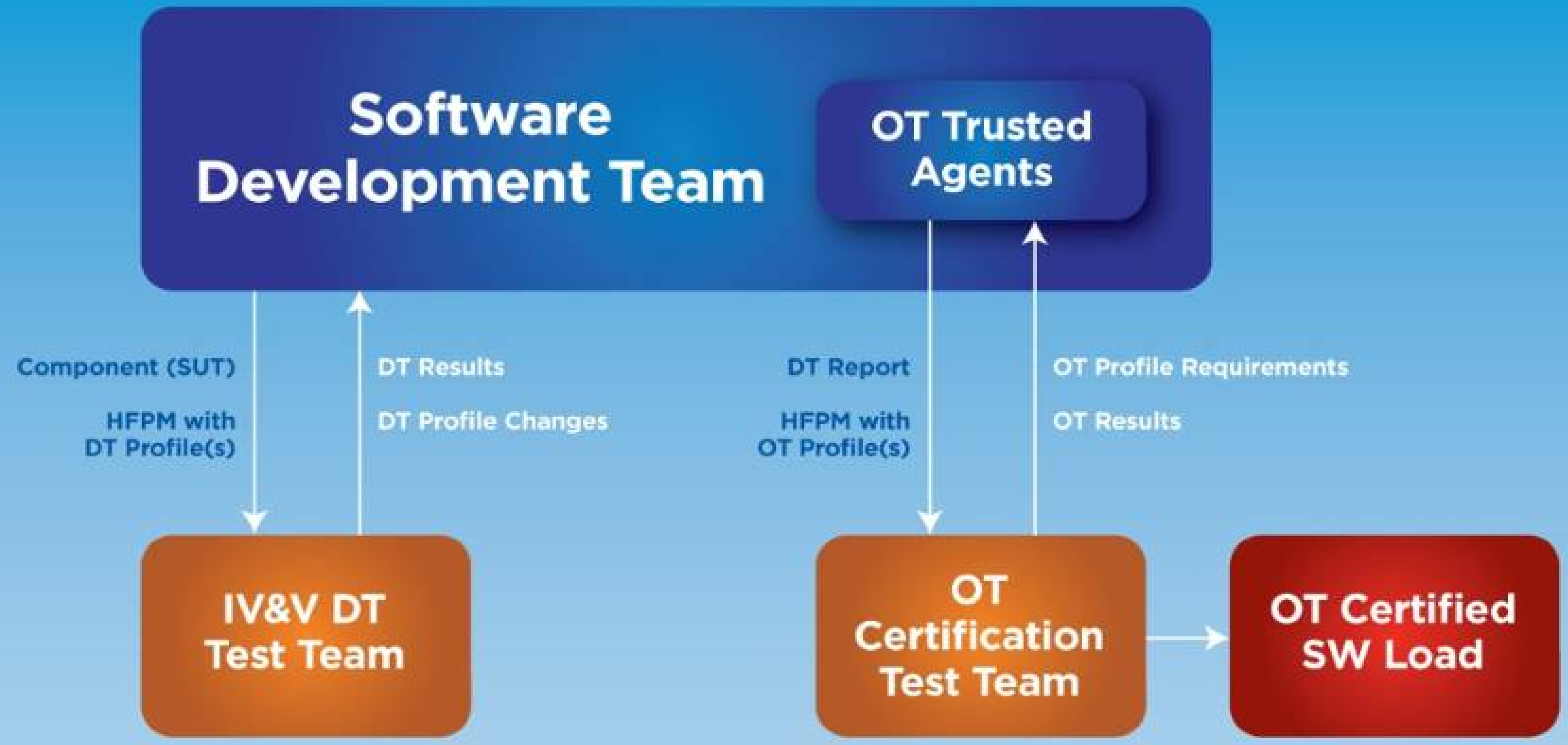


High-Fidelity Profile Model Functional Concept
Profile Developer's Characterization Methods
Software Developer's SUT Error Checking Methods
Software Developer's Reliability Calculation Methods
Tester's Profile Selection
Tester's Test Setup
Software Developer's Report Formatting Methods
Profile Developer's Execution Methods
Manual Functions
Automated Functions
Profile Generation
Operational Data
(Environment 1, 2, ...n)
Data Characterization
Characterized Data
(Environment 1, 2, ...n)
Characterization Adjustment
Test Executor
Test Case Generator
Test Case Generation Function
Operational, Stress Testing, Reuse, Fair, & Customized Profiles
Test Case Execution Function (SUT)
Profile Used
Test Cases (1, 2, ...n)
SUT Outputs (1, 2, ...n)
Output Analyzer
Reliability Calculator Function
Reliability Assessment
Output Checking Function
Total # of Outputs & Errors
Error-Generating Test Cases
Error-Containing SUT Outputs
Test Report Generator Function
Test Report
HFPM Workstation
Architecture Compliant Hardware

HFPM-Based Testing Employment for U.S. Navy Acquisition

- **HFPM developed and used during new software development, COTS acquisition, or reuse event by R&D team**
  - R&D DT profiles include stress-testing profiles
- **Component, HFPM & profiles passed to IV&V for developmental testing (DT)**
  - IV&V team can use R&D profiles or modify if desired
  - R&D / IV&V DT loop continues until software is mature
- **Mature component, HFPM & DT report passed to certification team for operational testing (OT)**
  - Certification team defines OT requirements
  - R&D OT trusted agents develop operational profiles
  - Cert team conducts/witnesses OT and certifies component or sends back for more development & DT

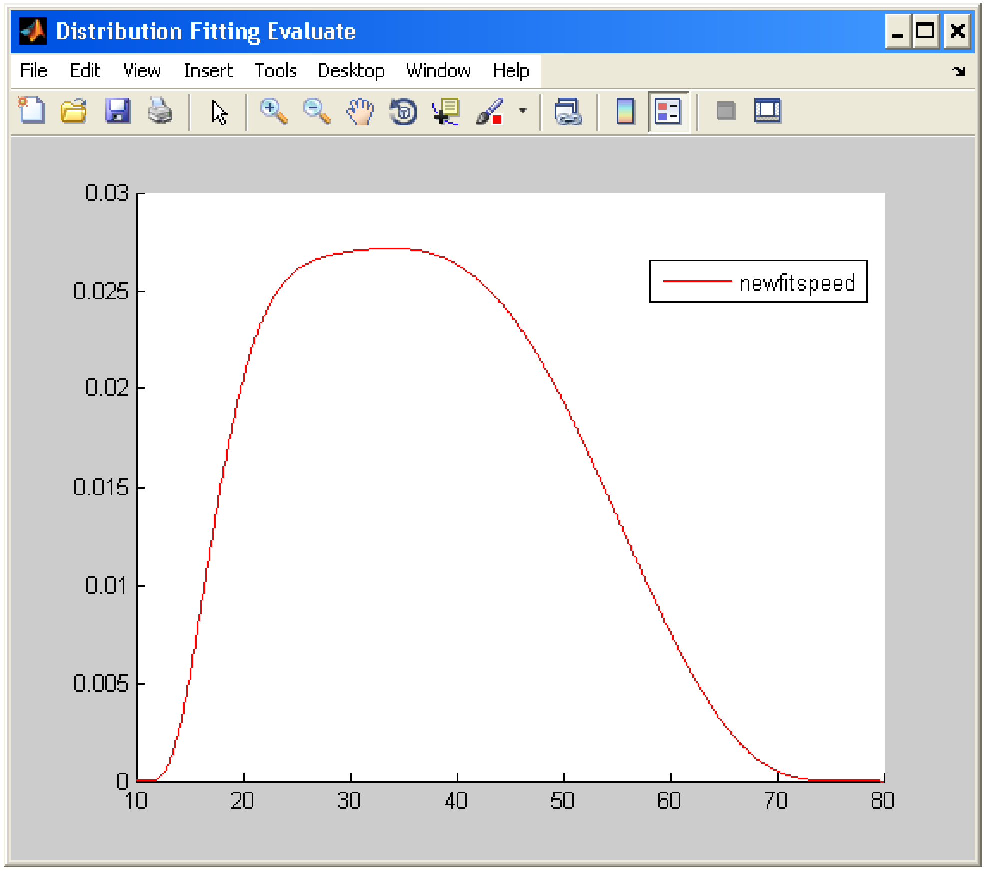
HFPM-Based Automated Software Testing Process Employment Scheme



Deriving HFPs from Historical Data

- Collecting historical data
  - – Lots of real data is best
  - – Else can approximate using known constraints
- Characterizing historical data
  - – Maximum Likelihood parameter estimation
  - – Maximum *A Posteriori* probability estimation
  - – Kernel density Estimation
  - – Parzen Neural Network

Example: Maritime tracks



Notional Small Boat Maximum Velocity PDF (Knots) (Dailey, 2010)

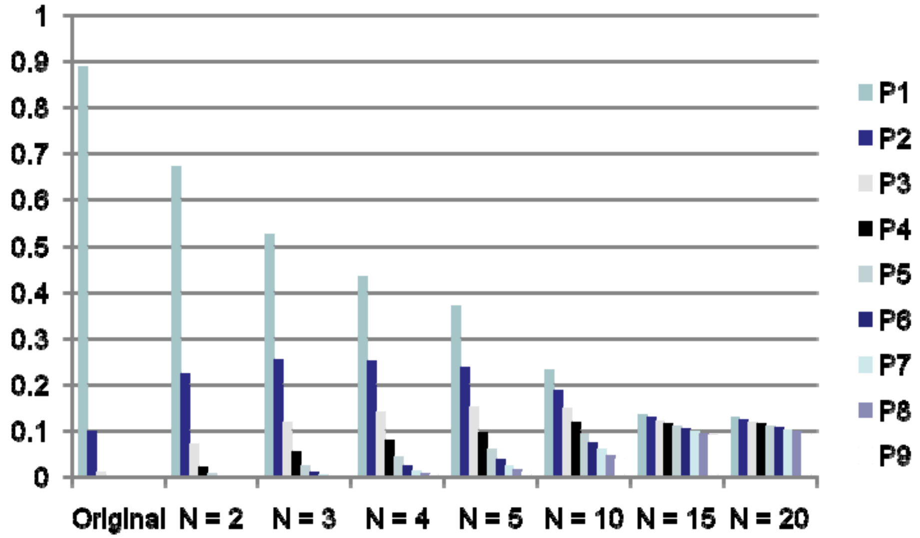
Validating HFPs

- Bayesian Information Criterion
  - Minimize (K ln N – 2 ln L)
    - K: number of free parameters to be estimated
    - N: number of data points
    - L: maximum of the likelihood function for the estimated model
- Goodness of Fit Tests
  - Minimize sum of squared error
- Confidence calculation based on amount of historical data

Deriving Stress-Testing HFPs from Historical Models

- Standard deviation-based methods
- Scale-expanding transformations
  - $P(x-m) \rightarrow P((x-m)/s)$, $s \in \{10, 100, 1000, \ldots\}$
  - Work for numerical and vector types
- Probability scaling transformations
  - $P(x) \rightarrow P(x)^{1/n}$, $n \in \{2, 3, \ldots, 20\}$
  - Work for arbitrary data types
- Utilization of dominating test cases
- Defining coverage criteria

Example: Probability Scaling Transformation



	Original	N = 2	N = 3	N = 4	N = 5	N = 10	N = 15	N = 20
P1	0.88888889	0.670925	0.526601	0.432891	0.369481	0.233181	0.134859	0.128692
P2	0.1	0.225035	0.254214	0.250707	0.238684	0.187417	0.128468	0.12409
P3	0.01	0.071162	0.117996	0.140983	0.150599	0.14887	0.12206	0.119418
P4	0.001	0.022504	0.054769	0.079281	0.095022	0.118252	0.115971	0.114922
P5	0.0001	0.007116	0.025421	0.044583	0.059955	0.093931	0.110186	0.110595
P6	0.00001	0.00225	0.0118	0.025071	0.037829	0.074612	0.10469	0.106432
P7	0.000001	0.000712	0.005477	0.014098	0.023868	0.059266	0.099468	0.102425
P8	0.0000001	0.000225	0.002542	0.007928	0.01506	0.047077	0.094506	0.098568
P9	0.00000001	7.12E-05	0.00118	0.004458	0.009502	0.037395	0.089792	0.094857

Dominating Cases for Stress Testing

Error Type	Heuristics for choosing stress test cases
Numeric Overflow	Largest and smallest representable numbers
Buffer Overflow	Very long input string
Free Storage Overflow	Create many new objects
Wrong Conditional Logic	Data values close to the both sides of an interval boundary
Unprotected Pointers	Null pointer
Unprotected Division	Zero

Conclusions

- **Effective and cost-efficient testing can be achieved by a mixture of automation methods**
  - Determine which tests can be safely eliminated
  - Determine which test cases will most-likely expose errors
- **This research defines a statistically-based automated testing process that can be executed using historical environment data**
  - Reduces testing time while increasing coverage
  - Model-driven process is reusable, scalable
  - Process should enable benefits brought on by OA