# A bestiary of performance diagnosis methodologies

B. Robert Helm

**CIS-TR-93-24**
November 1993

## Abstract

The goal of this research is to formalize and automatically support parallel *performance diagnosis*. Performance diagnosis is the task of identifying the principal flaws in a parallel program that hinder good performance.

An initial step toward this goal is to identify the performance diagnosis *methodologies* that are supporteed by current performance evaluation tools. This talk presents preliminary results from a survey of ten such tools.

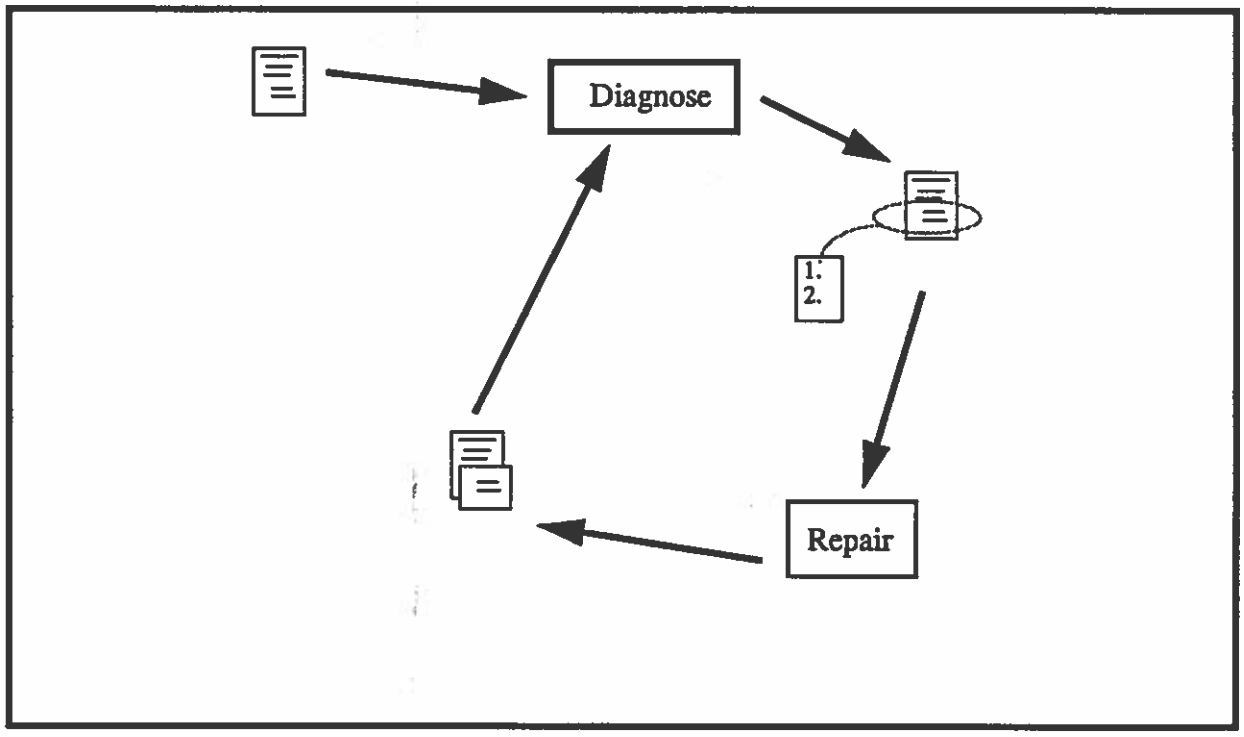
Department of Computer and Information Science
University of Oregon
Anonymous FTP: ftp.cs.uoregon.edu:pub/malony/Papers/CIS-TR-93-24.ps.Z

# What is performance diagnosis

Performance diagnosis is identifying the principal performance problems in a program and their causes.

This information is used to guide repair of the program.

Supports a hillclimbing tuning strategy



# How is performance diagnosed?

Numerous papers give hypothetical performance diagnosis scenarios.
Scenarios were intended to highlight an analysis tool.
Most (not all) written by researchers rather than programmers.

**Scenario summary**

| Tool | Targets | Programs | Papers |
|---|---|---|---|
| Predicates | Shared memory. | Parallel search | [1] |
| (PTOPP) | Clustered shared memory. | (6) Perfect Benchmarks | [2],[3] |
| $W^3$ | Shared memory. | (2) Splash Benchmarks | [4] |
| IPS-2 | Shared and dist. memory. | Simplex | [5] |
| Chitra | Shared and dist (?) mem. | Parallel DEVS | [6] |
| MTOOL | Shared memory. | Econometric model | [7] |
| (SPT) | Shared, dist. memory. | Quicksort | [8],[9] |
| PIE | Shared memory | OS thread scheduler | [10] |
| ChaosMon | Shared and dist. memory | Quicksort | [11] |
| PEPP | Shared and dist. memory | Navier-Stokes solver | [12] |

# Supporting Performance Diagnosis

Based on scenarios, the performance diagnosis process:

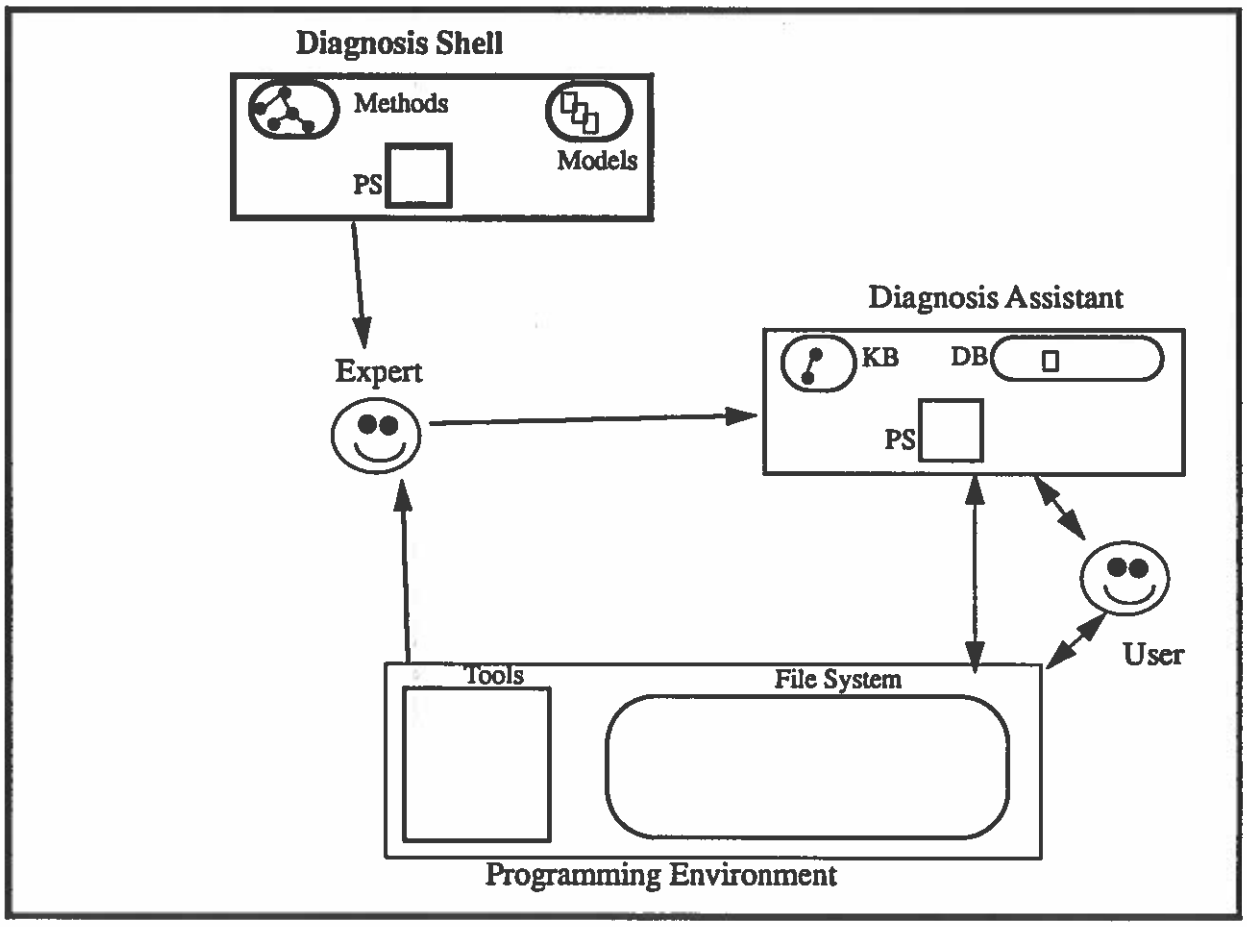
1. Is hypothesis- and data-directed.
2. Generates many program versions and associated data.

# Supporting Performance Diagnosis

Thesis: A machine- and program-independent diagnosis shell that integrates:

1. An reactive, blackboard-like problem-solver.
2. A library of problem-solving goals, methods, and models.
3. An experiment and results management system.

can automate significant aspects of current performance diagnosis methods.

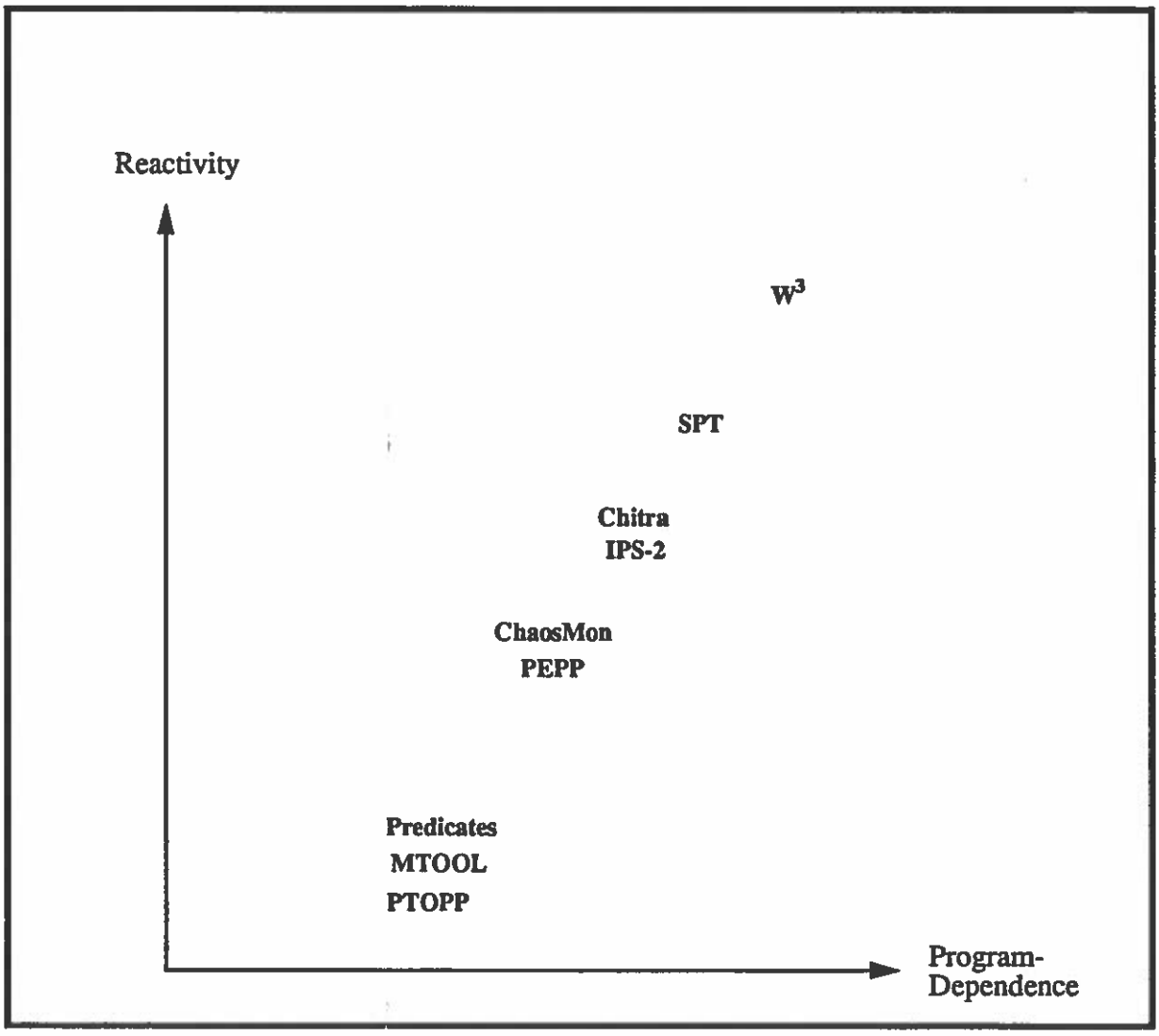


# A Problem-Solving Model for Diagnosis

Claim 1: Performance diagnosis is best characterized as refinement of a set of hypotheses that explain negative aspects of performance.

Refinement strategies range from planned to reactive.

Differing amounts of program- and architecture-specific knowledge are used to generate and order hypotheses.

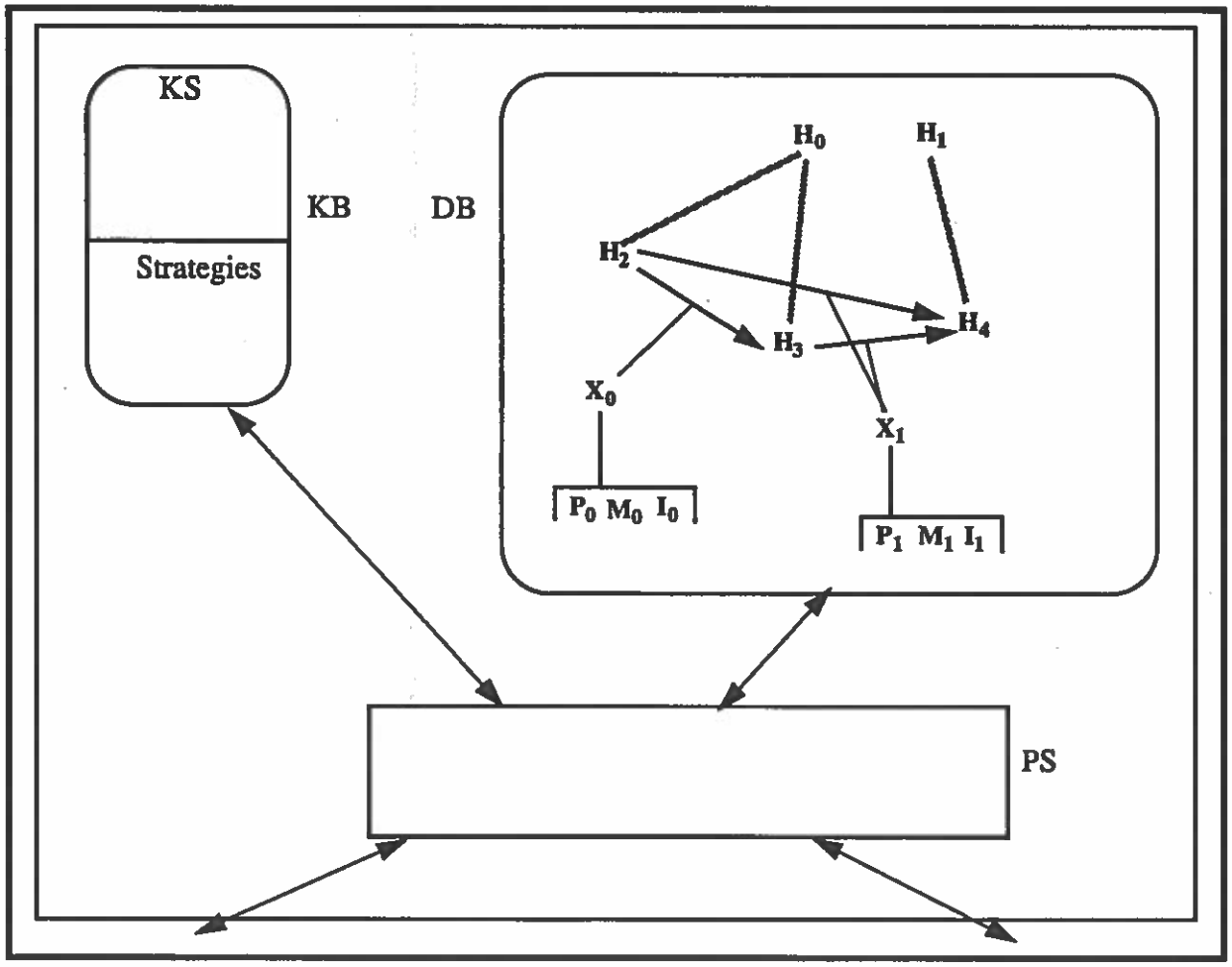


# Problem-Solving Model: Automated Support

Support the model with an expert system shell providing extensible facilities for representing:

- Diagnosis state (hypotheses, experiments, and results).
- Diagnosis actions (rules for experimentation and updating hypotheses).
- Diagnosis strategies.

Blackboard framework will serve as a basis.



# Experiment Management for Performance Diagnosis

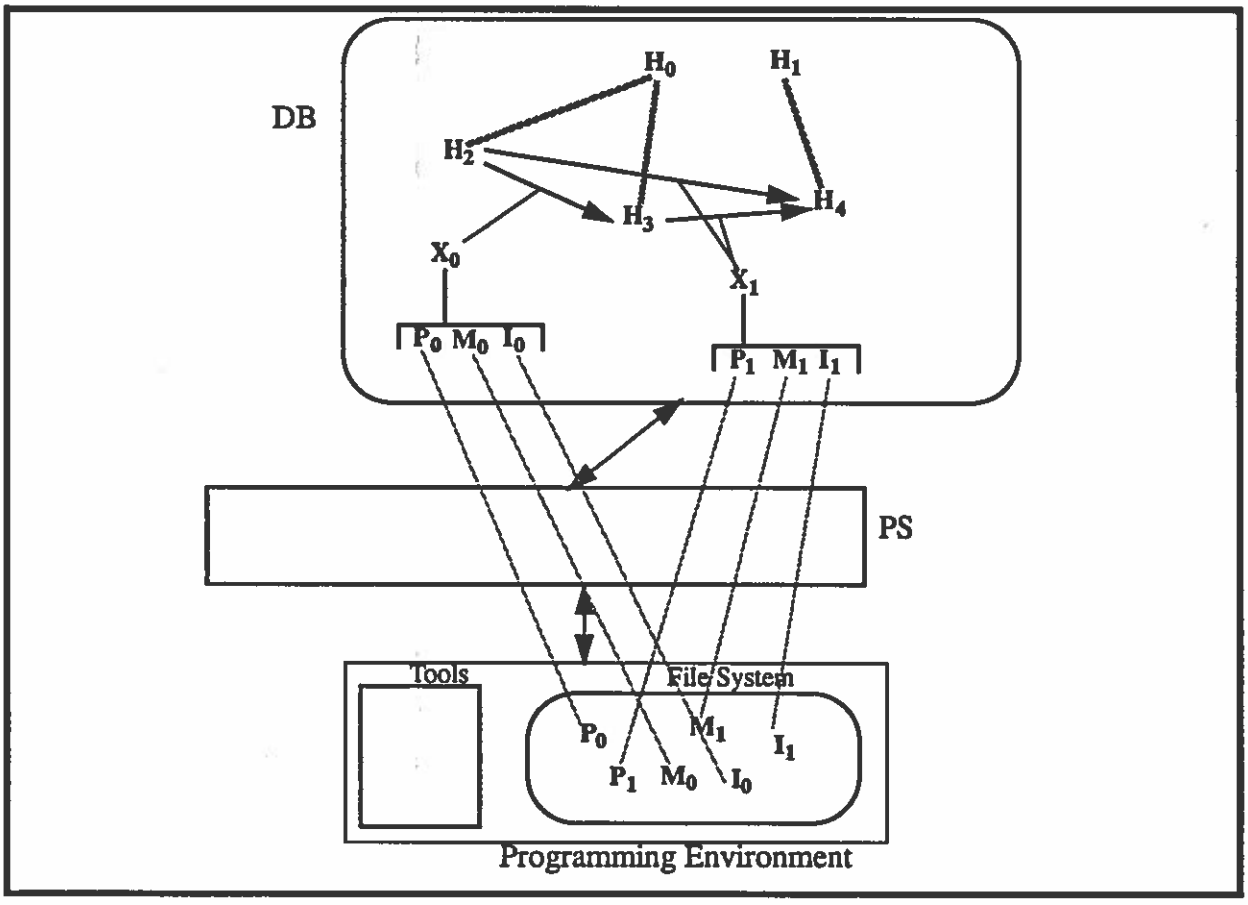
Claim 2: Tracking source/object code and input/output data is a significant, difficult subtask of performance diagnosis.

- Construction of correct *experiments* out of consistent program versions, machine configurations, and input data sets is difficult.
- Experiments and experimental results should be reused within and across sessions where possible.

# Experiment Management: Automated Support

Support experiment management by integrating the diagnosis shell with facilities for:

- Linking program versions, data sets, and machine configurations into experiments.
- Running experiments and incorporating results into the diagnosis database.
- Retrieving past experiments and results guided by the current problem-solving state



# Related Work

Performance analysis.

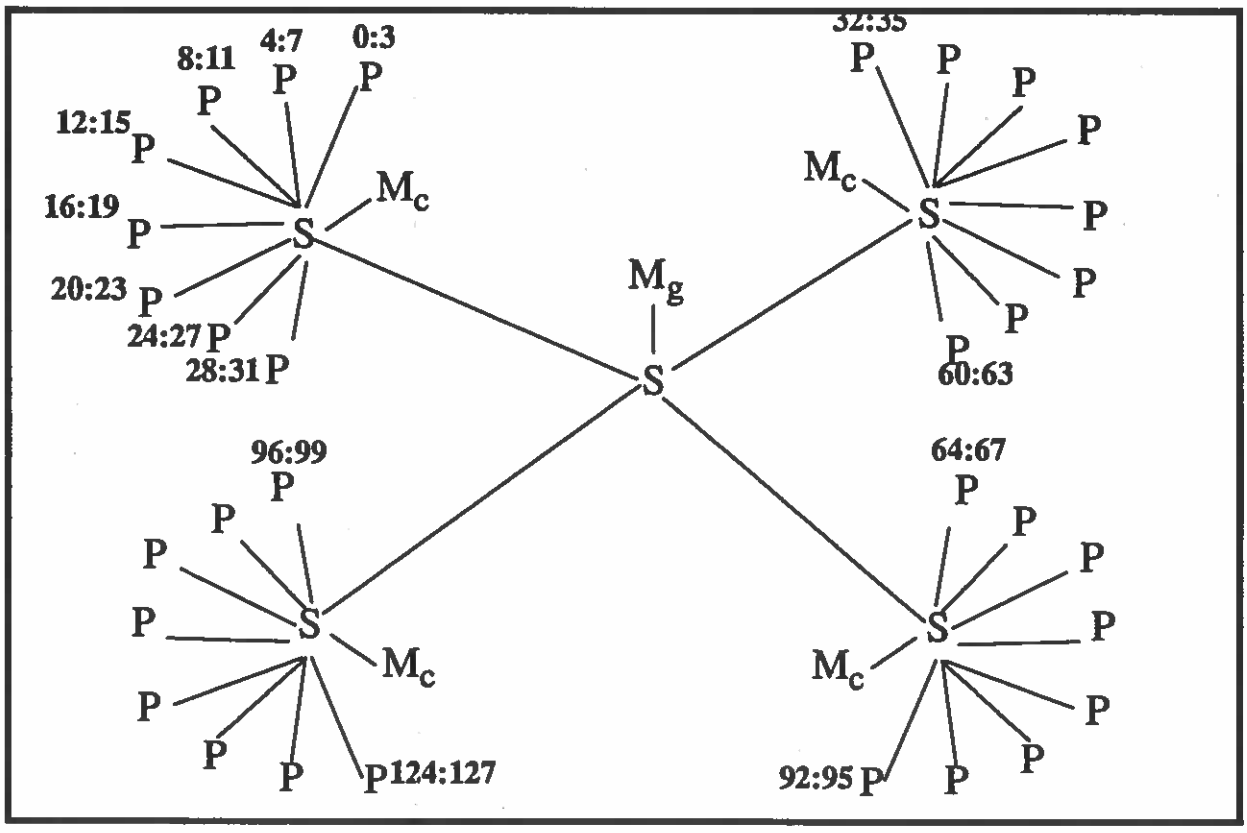
Diagnostic expert system shells.

Rule-based process programming environments.
- Marvel.
- Law-governed systems.

# PTOPP Methodology

The hypothesis space, knowledge sources, and strategy are based on:

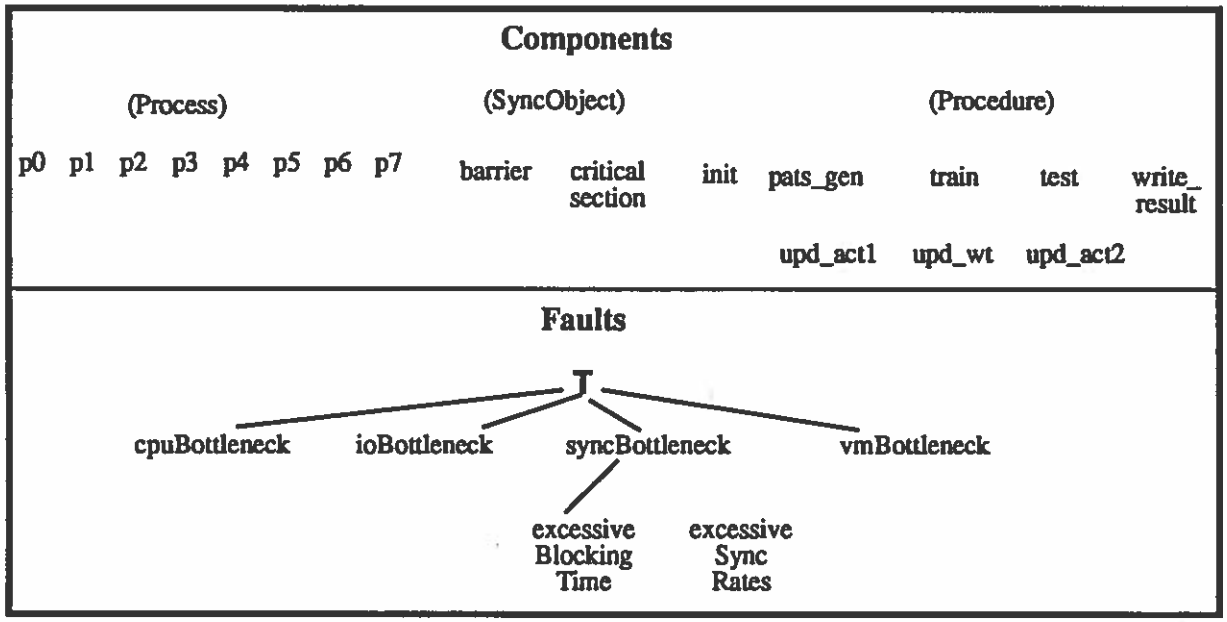
- The Cedar architecture and machine.
- The type of parallelism supported (DO-loop spreading).
- The programs being diagnosed (Perfect benchmarks).



# W3 - Neural Net Hypothesis Space

The hypothesis space is the cross-product of:

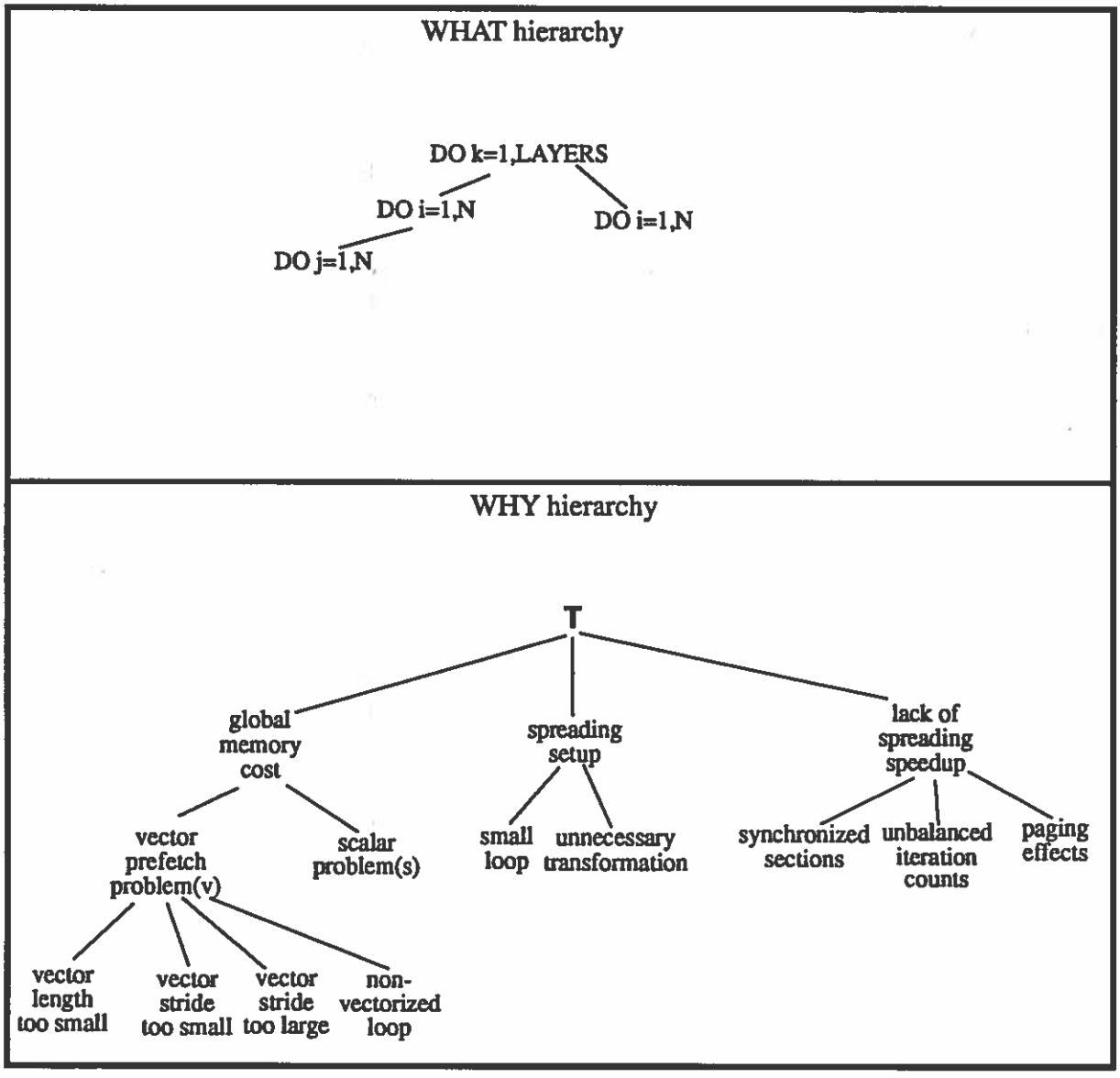
- Where (component)
- Why (fault).



# PTOPP Hypothesis Space

The hypothesis space is the cross-product of:

- Where (which loops have problems)
- Why (what problem each loop has).



# PTOPP Strategy

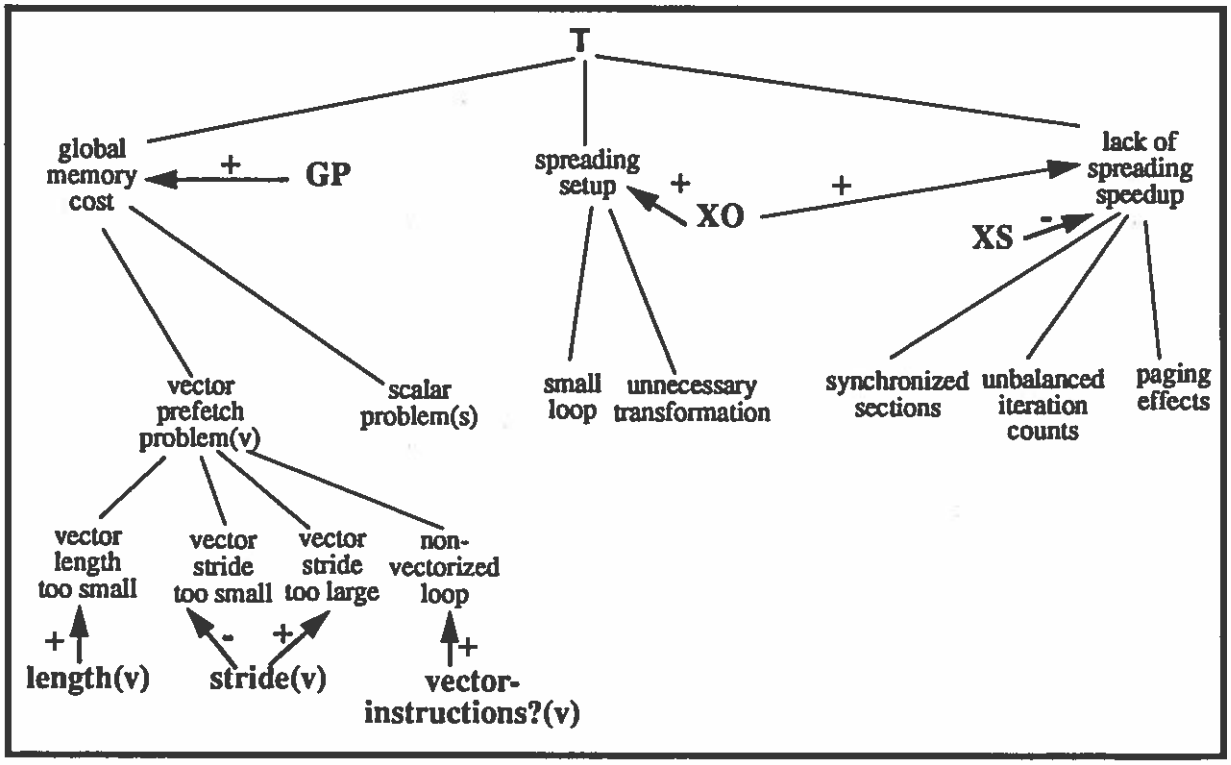
1. Order loops by significance.
   - Profile serial execution time, broken down by loops.
   - Order hypothesis space along "where" dimension based on results.
2. Compute basic metrics for program.
3. Refine lead hypothesis.
   - Generate hypotheses for longest loop.
   - Order and prune using rules and metric results.
4. -> Repair based on diagnoses.
5. Go to 1.

# PTOPP Hypothesis refinement

Order and/or prune possible performance problems using metrics:

- GP (O1g/O1 - 1) indicates memory problems.
- XO (C1/O1g - 1) indicates cluster spreading overhead.
- XS (C1/C4) indicates effectiveness of cluster spreading.

Final refinements selected principally by static analysis of parallel code.



# PTOPP File Tracking

Source files:

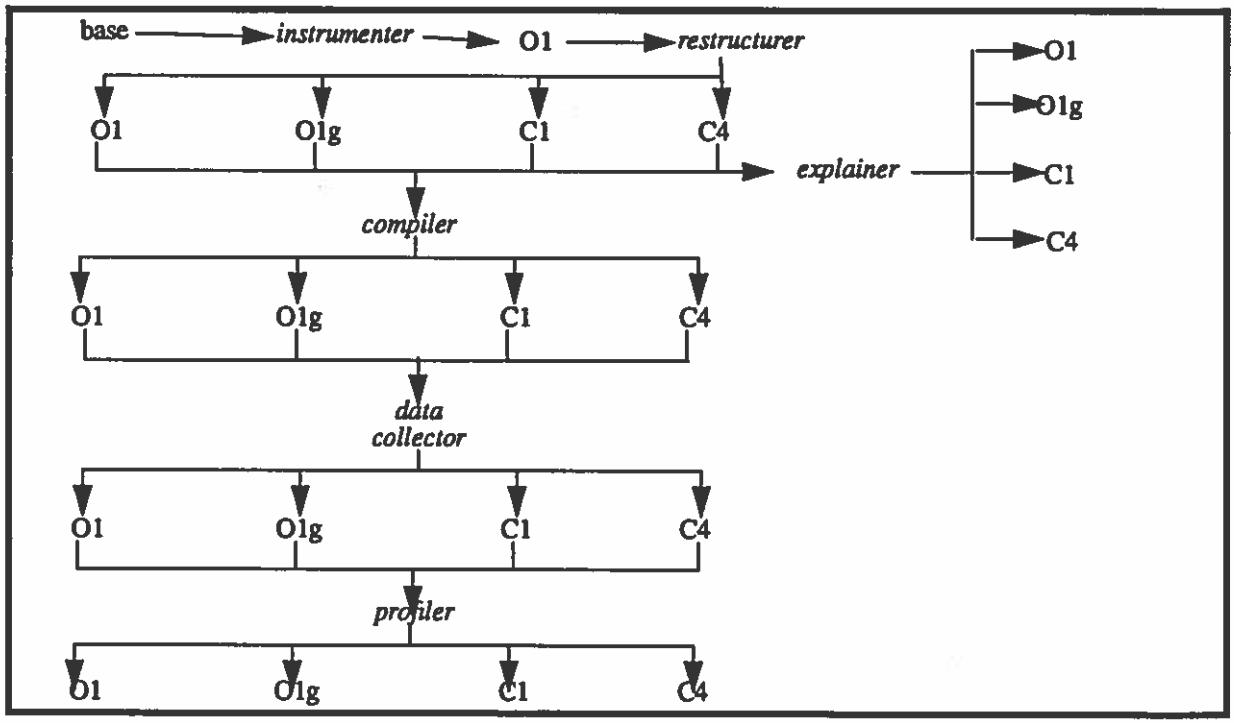
- Sequential base version
- Sequential version instrumented for loop-level profiling (O1).

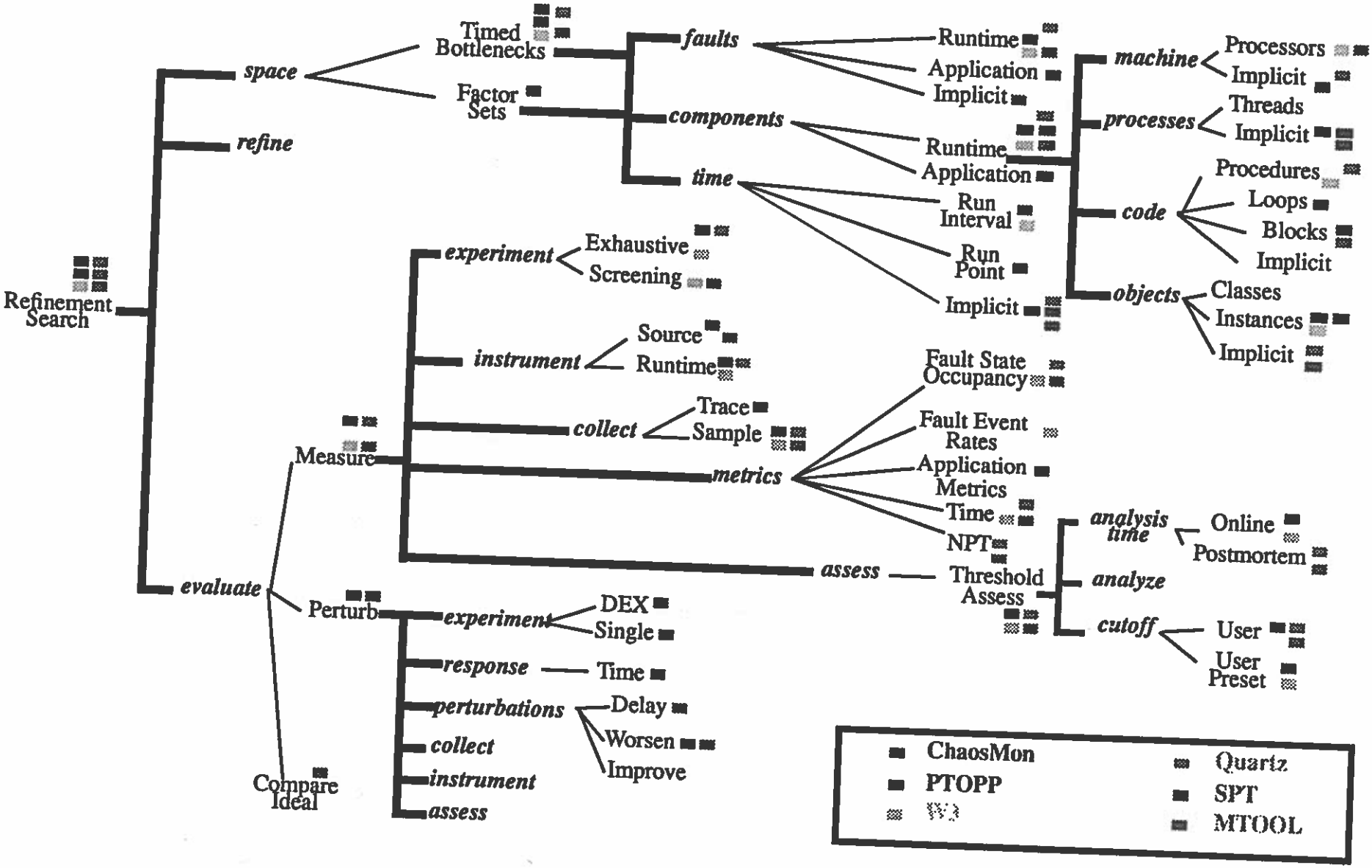
Compiler outputs:

- Restructured source code for O1, O1g, C1, C4.
- Restructurer static analysis report.
- Object codes.

Experiment results

- O1, O1g, C1, C4 loop-level profiles.




Refinement Search
space
refine
evaluate
Timed Bottlenecks
Factor Sets
faults
components
time
Runtime
Application
Implicit
Runtime
Application
Run Interval
Run Point
Implicit
machine
processes
code
objects
Processors
Implicit
Threads
Implicit
Procedures
Loops
Blocks
Implicit
Classes
Instances
Implicit
Measure
experiment
Exhaustive
Screening
instrument
Source
Runtime
collect
Trace
Sample
metrics
assess
Fault State Occupancy
Fault Event Rates
Application Metrics
Time
NPT
Threshold Assess
analysis time
analyze
cutoff
Online
Postmortem
User
User Preset
Perturb
experiment
DEX
Single
response
Time
perturbations
Delay
Worsen
Improve
collect
instrument
assess
Compare Ideal
ChaosMon
PTOPP
Quartz
SPT
MTOOL