



The Discovery Based Development Approach:

A Process Aberration or a Better Way
to Develop Complex Applications?

J. Shupp

Discovery Based Development



- If you don't know where you're going, any road will do

Chinese proverb

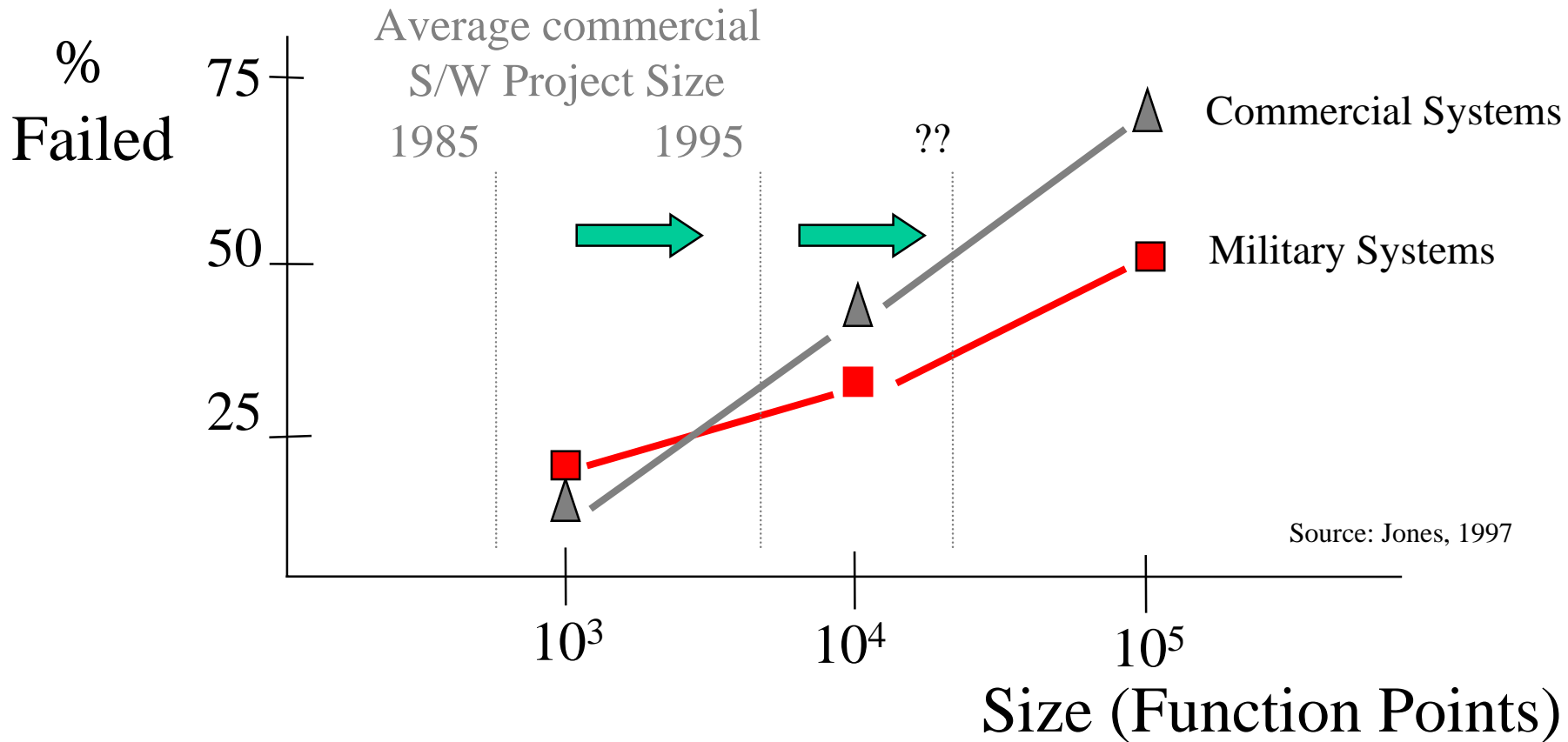
- If you don't know where you are, a map won't help

Watts Humphreys

- If you don't know the terrain ahead, trade in the mules; buy a compass

Jeff Shupp

A Disturbing Trend



Larger Scale Commercial Systems are
More likely to Fail

The role of complexity



- Under-estimate level of effort:
 - Complexity adds up to 75% more functionality to a system's F.P. size

source: Jones, 1997

- Misinterpret the processing semantics
 - Dominant factor in complexity is control

source: McCabe, 1976

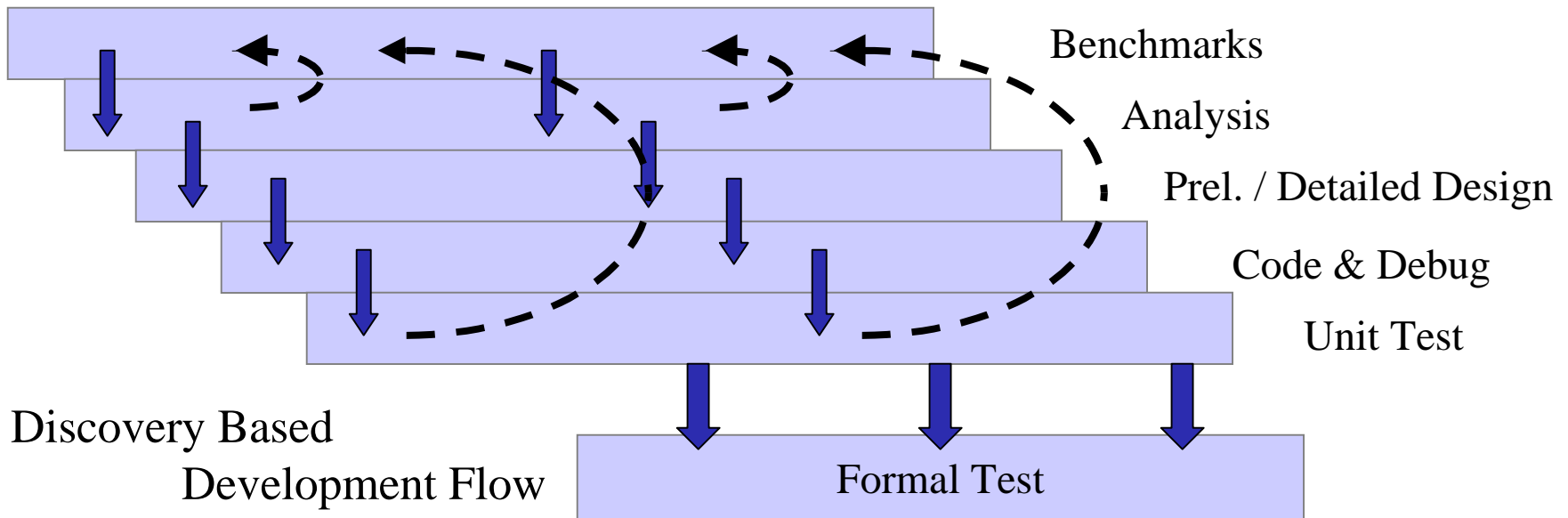
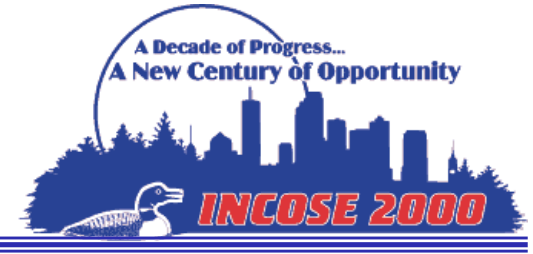
- Whole is greater than sum of the parts
 - Behaviors drive complexity when they exist as an emergent property

Discovery Based Development



- Requires close collaboration between:
 - Systems Engineering
 - Software Development
 - Management
- Centers around systems engineering activities for guidance
 - Benchmark Definition
 - Analysis of Results
 - Determining Success Criteria for expected behavior
- Expects Failures

The Process



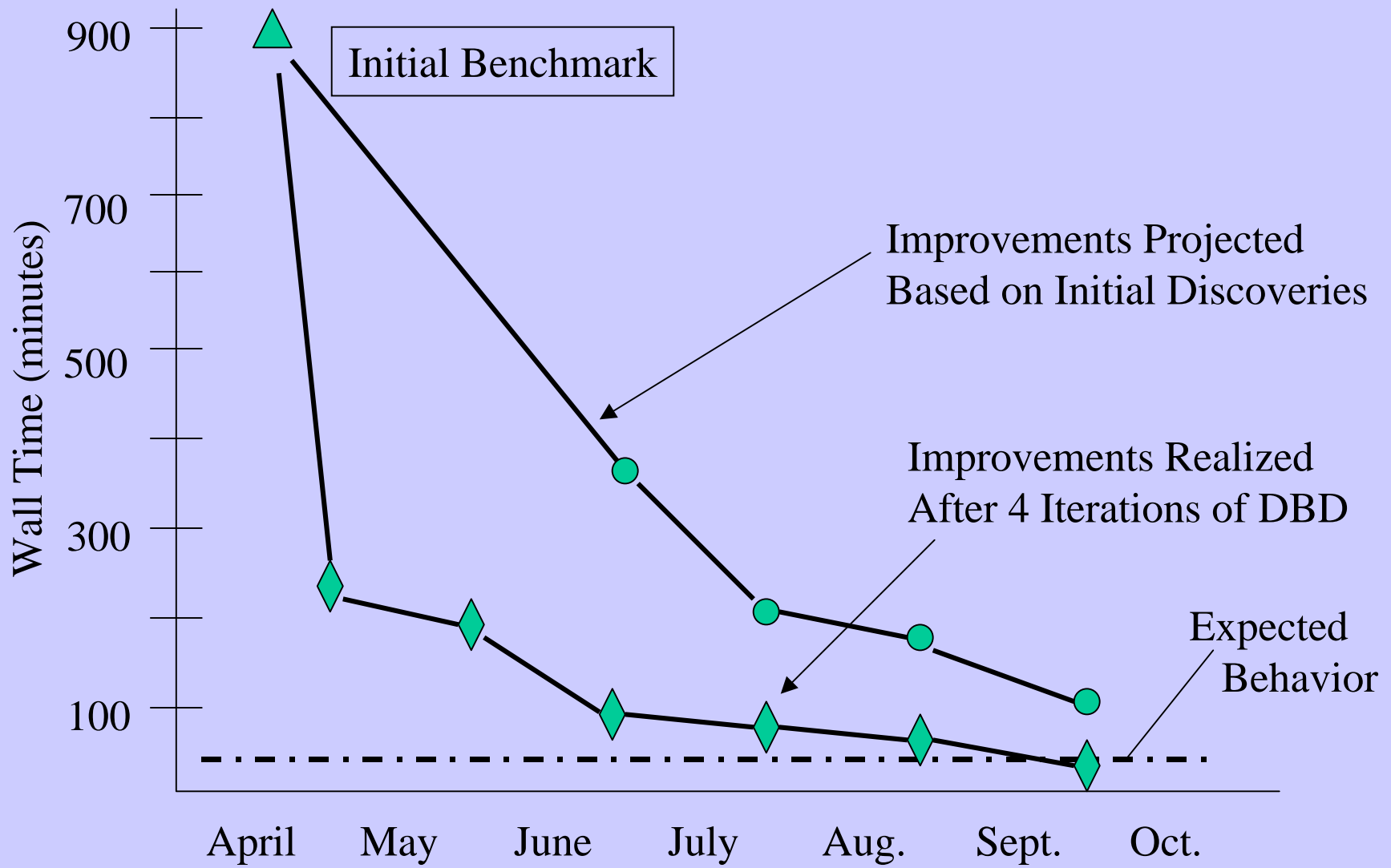
DBD allows for Mid-Course Corrections
due to Discoveries

Discovery Based Development

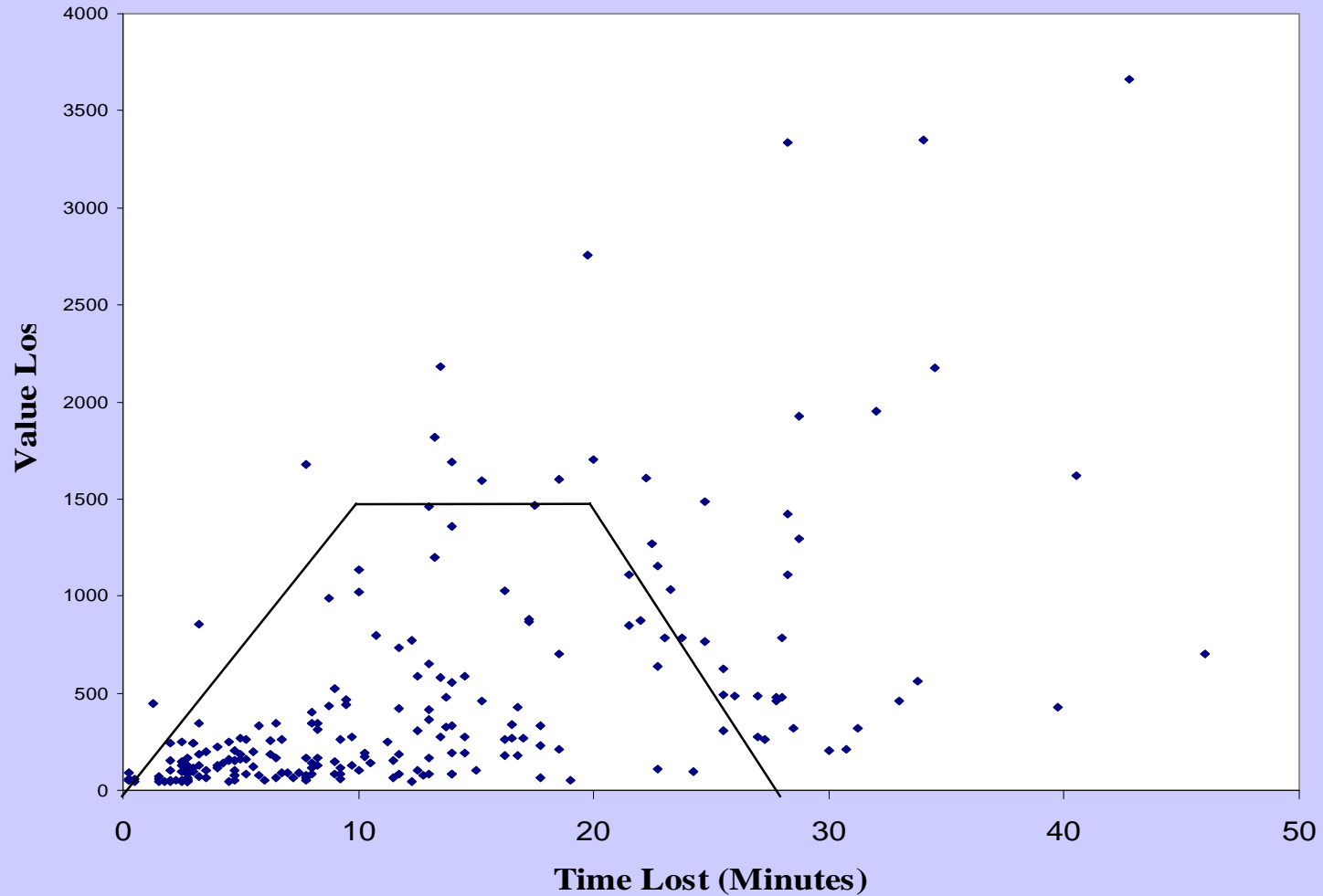


- Identify the problem (Expected Behavior)
- Determine success criteria
- Analyze the benchmark results to identify points of attack
- Prioritize options by greatest payoff
- Determine the time to implement
- Develop the next increment
- Determine the incremental improvement in performance

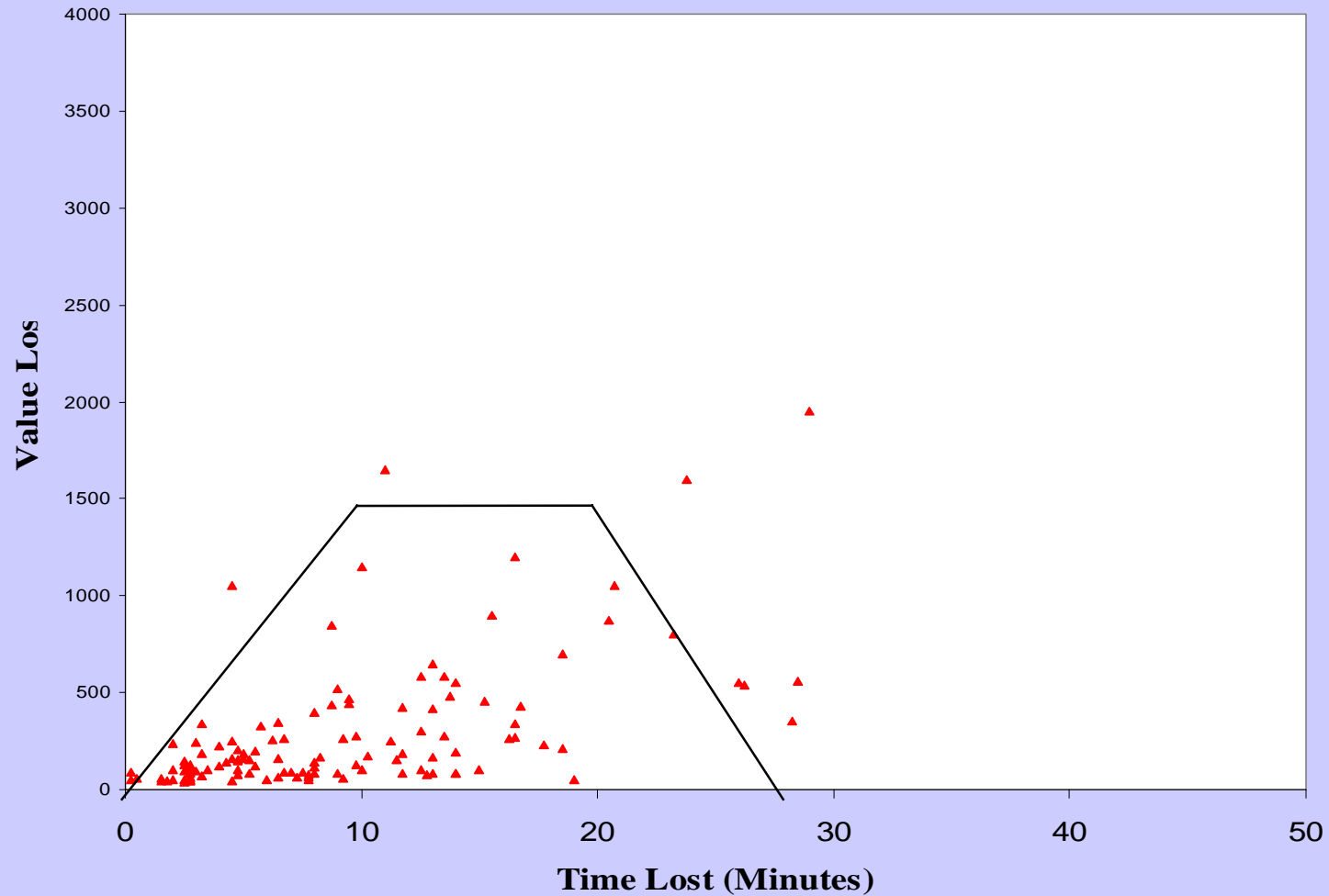
DBD in Action: Timeline



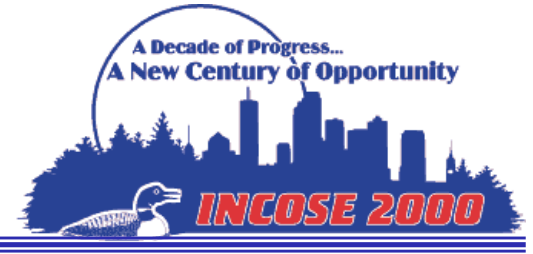
DBD in Action: Before



DBD in Action: After



DBD Sounds Like:



- Technical Performance Measurement (TPM)
 - “A plan of expected technical achievement to which the actual progress is compared using periodic measurement or tests”

source: Kulick, 2000
 - If TPM goes outside range, work it as a program risk

source: Martin, 1996

DBD is the method for eliminating
Performance Risks

DBD Sounds Like:



- Evolutionary Prototyping/Delivery Approach
 - Good for developing functional capabilities
 - Building block or distinct features
 - Don't optimize for performance as you go
 - Miss global contributors

source: McConnel, 1993, 1996

DBD strategies kick in [to address behavior]
as system completes enough functionality to
expose its emergent properties

Conclusion



- If you don't know where you're going...
 - Define an expected behavior (a TPM)
- If you don't know where you are...
 - Perform a benchmark as soon as the system is complete enough to detect its emergent behavior,
 - Repeat the benchmark as improvements are made
- If you don't know the terrain ahead...
 - Analyze benchmark to understand next point of attack, prioritize changes, implement