

Verus Approach to Flexible Software
Architecture Design in Next
Generation Test Data
Acquisition/Data Management
Systems

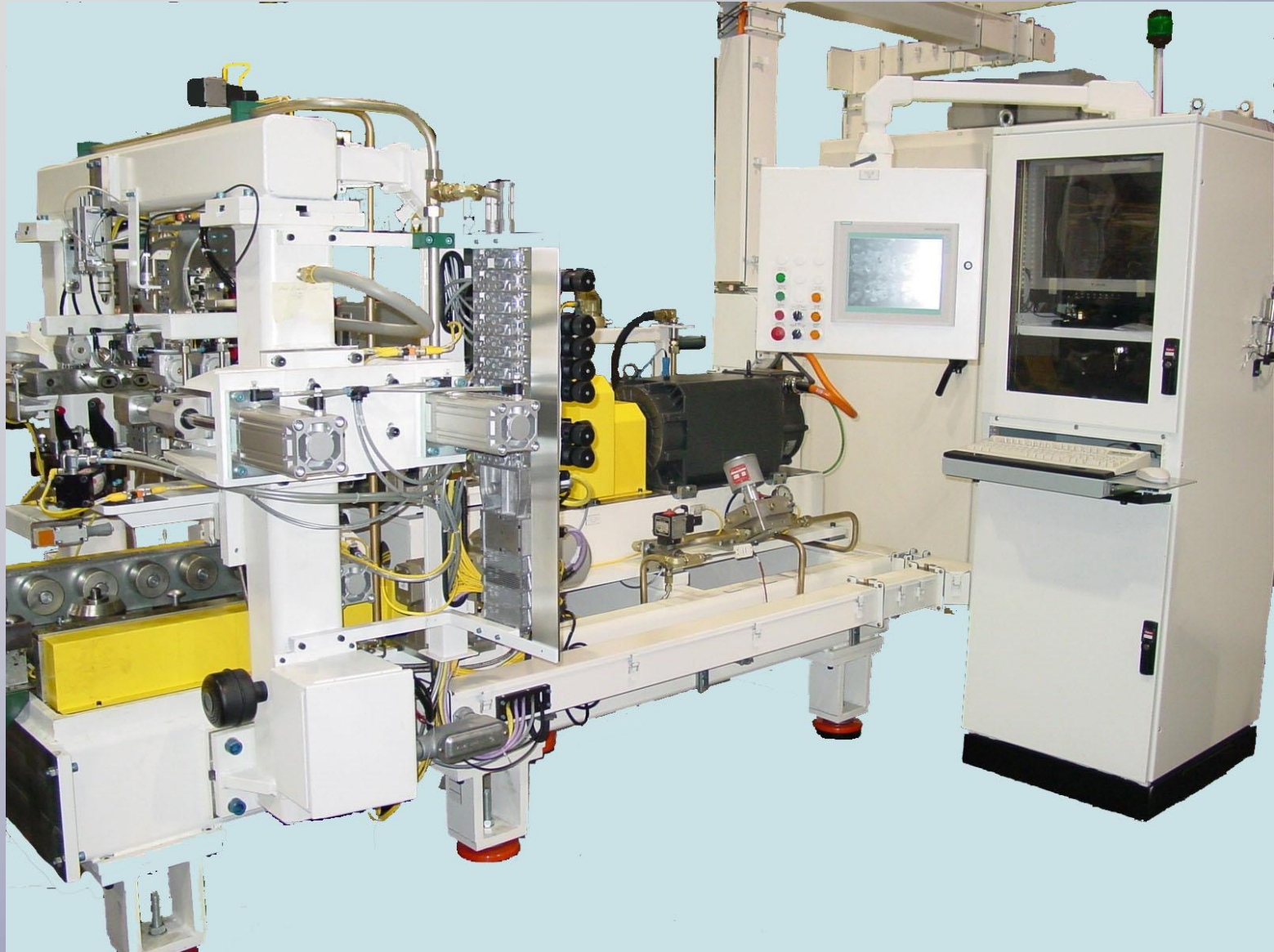
Dale Karolak, Ph.D.



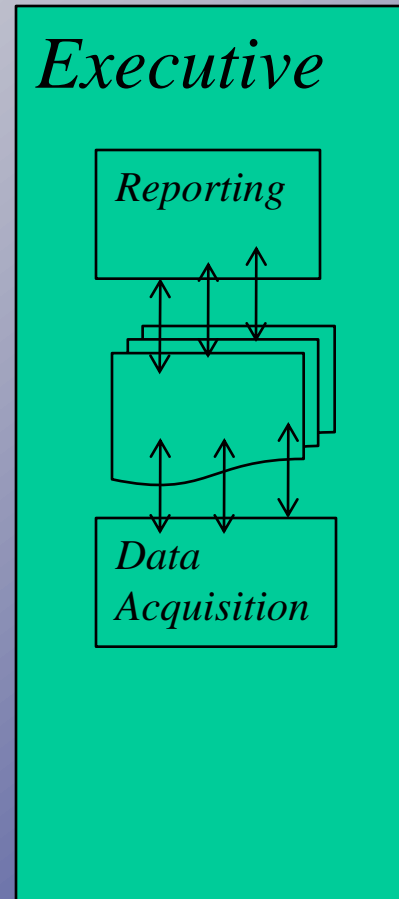
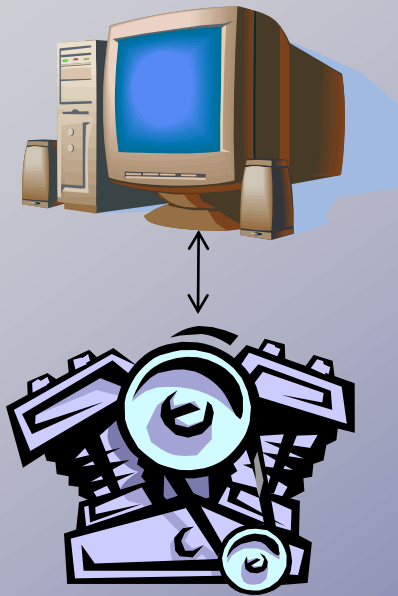
Software Evolution

- Many test systems software starts out as software developed for a project that then get reused again and again
- Over time these test systems software grow out of it's architecture
- Soon the cost of updating, revalidating and maintaining the software outweighs the benefits of its reuse

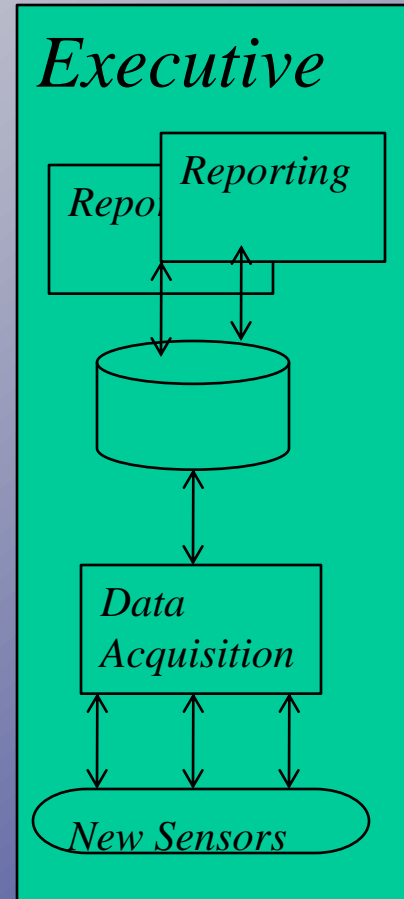
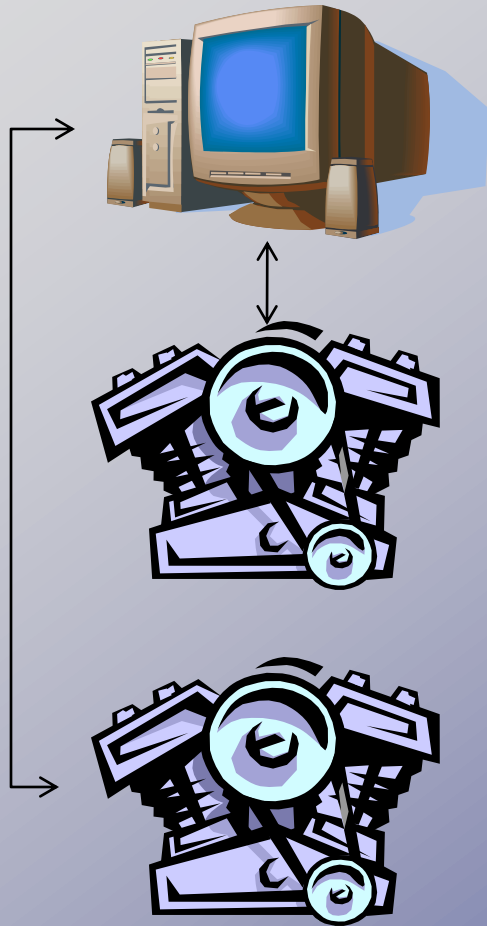
Example – Engine Test Stand



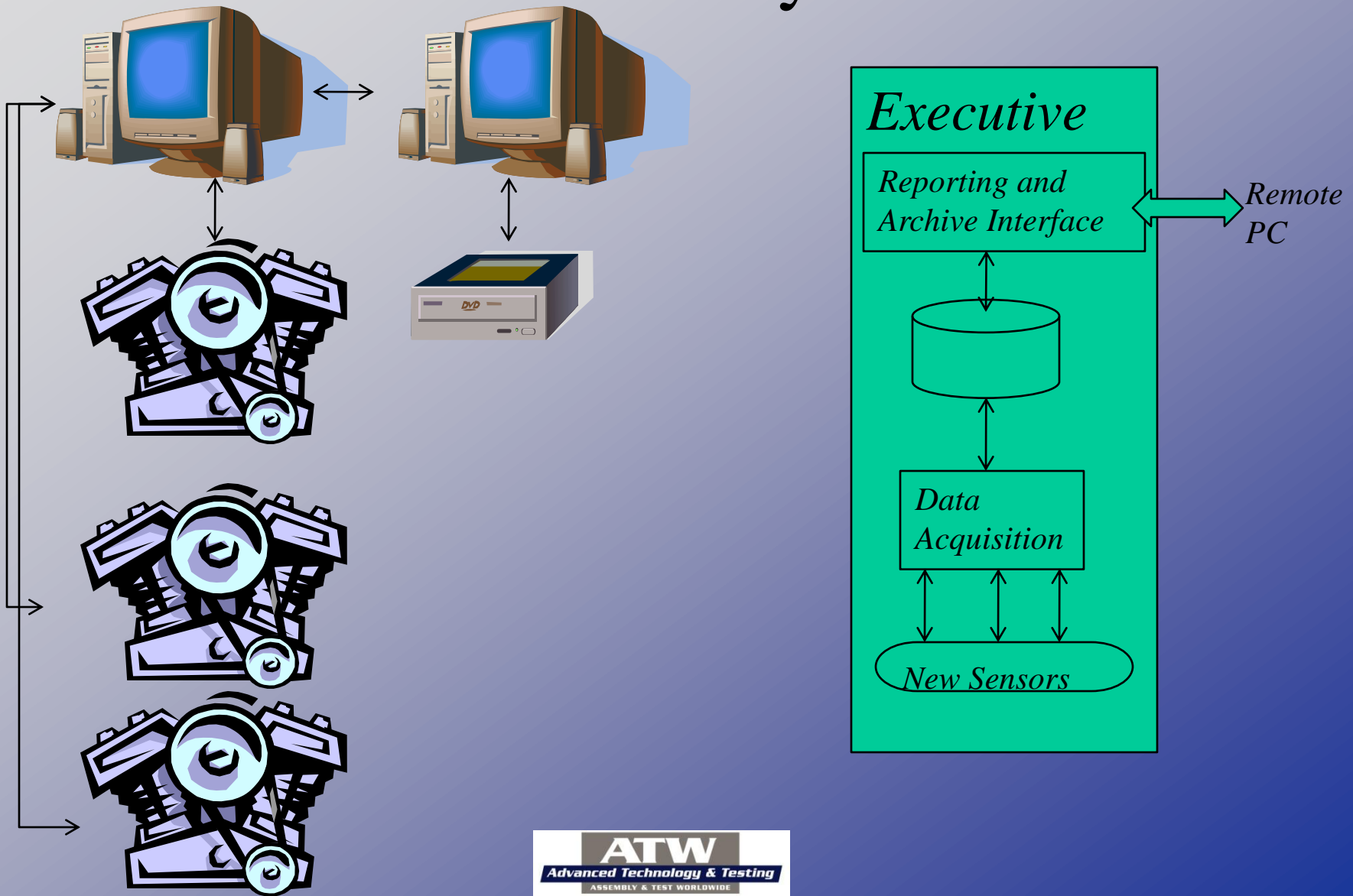
Example of Product Evolution – Stand Alone Systems



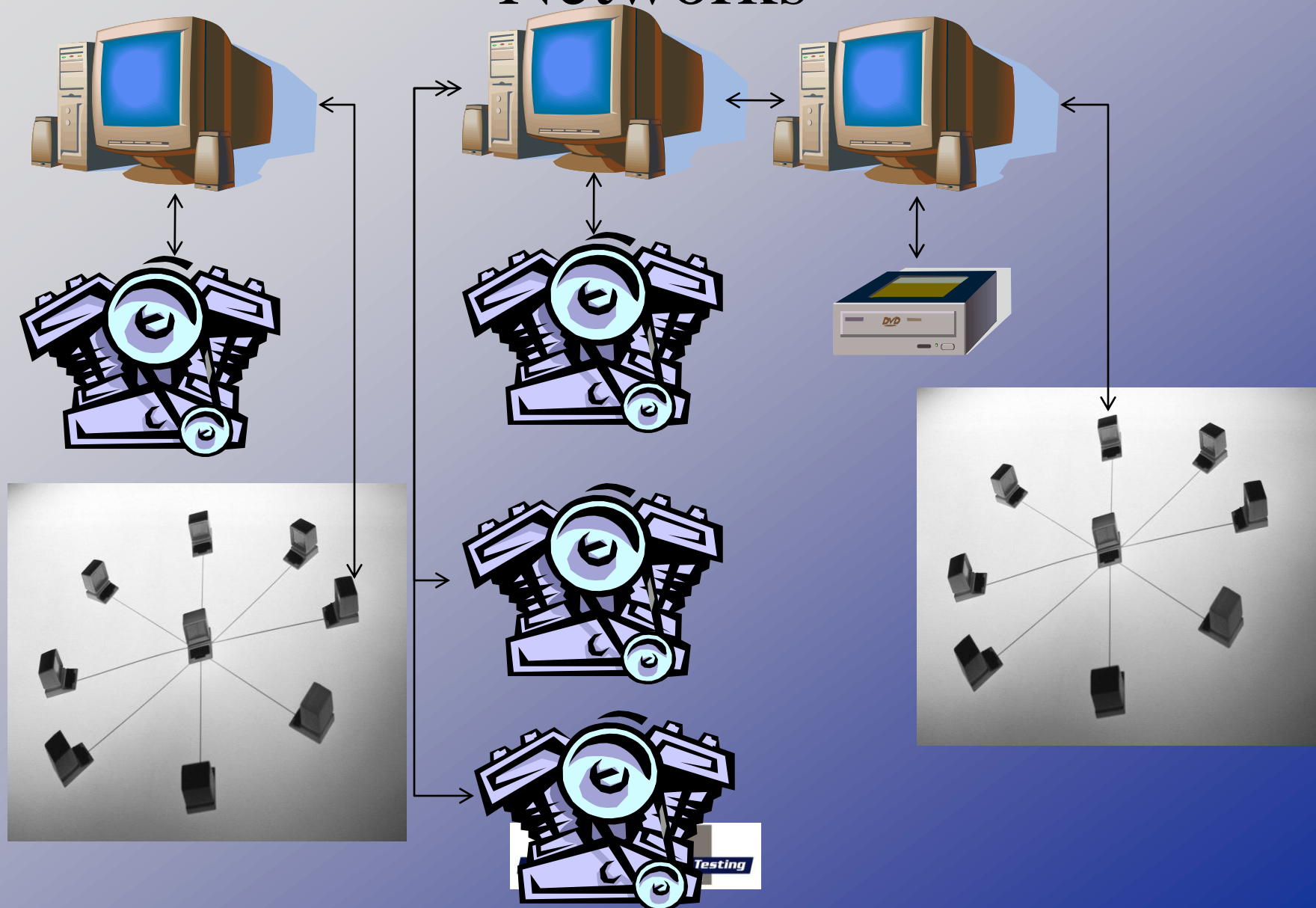
Example of Product Evolution – Stand Alone Systems



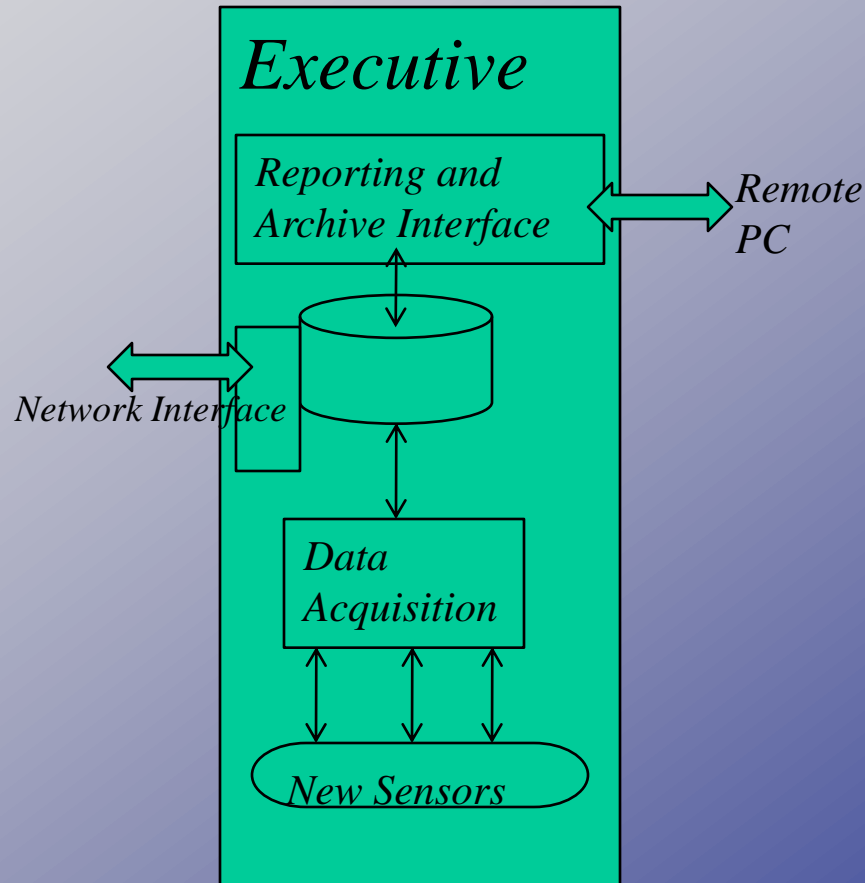
Example of Product Evolution – Stand Alone Systems



Example of Product Evolution - Networks



Software Architecture - Networks



Over Time These Test Systems Software Grow Out of It's Architecture

- This can be measured by Architecture Complexity-Stability Metric (*Acs*)

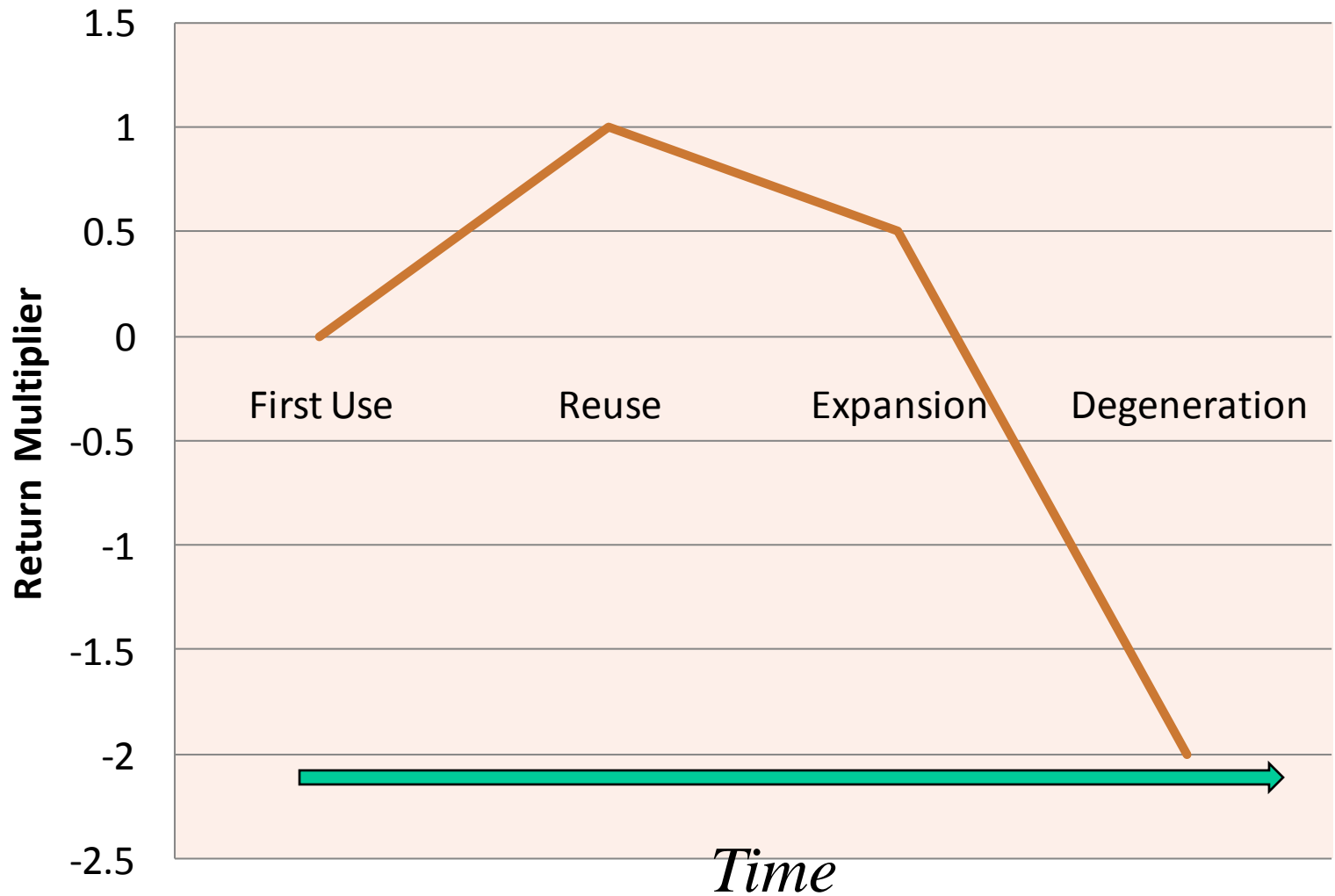
- $Acs = ((\sum \Delta DS) / DS_{St}) * (1 / \Delta T)$

ΔDS = Data structures that changed

DS_{St} = Total number of data structures

T = time interval

Cost vs. Benefit Over Time



What is Verus?

- ATW's next generation test software for data acquisition and management is called "Verus"
- Verus is Latin for truth - capable of withstanding a test or trial
- The following explains key concepts in the architecture design of Verus

Verus Architecture

- Considerations/Benefits/Goals
 - Support multiple product types
 - Able to be elastic for future changes
 - Reduce time for changes
 - Reduce time for verification
 - Fast execution

Verus Architecture

- Not A Consideration/Benefit/Goal
 - Shortest development time for initial software
 - Lowest cost solution for initial software
 - Reduced feature content
- This is an investment with long term returns

Verus Architecture – Design Process

- Identify feature list
- Identify components with properties high in cohesion and low in coupling
- Place elasticity has a high priority

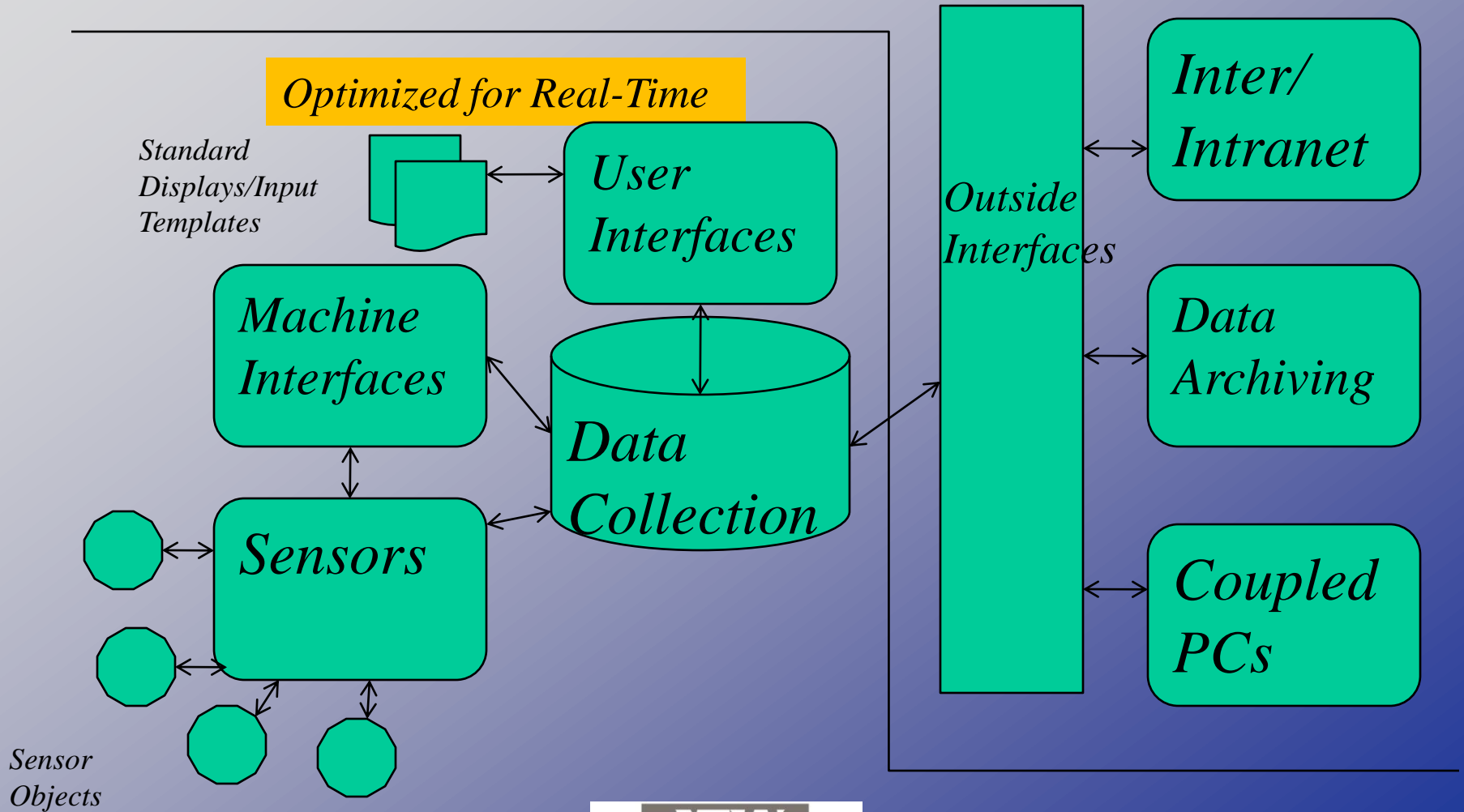
Verus Architecture – Design Process

- Identify design tradeoffs
- Highly coupled data storage
- Tradeoff between commonization and customization
- Design around templates

Verus Architecture – Design Process

- Isolate data handling/processing from data collection and interfaces
- Identify critical real-time areas
- Highlight background tasks that are not real-time

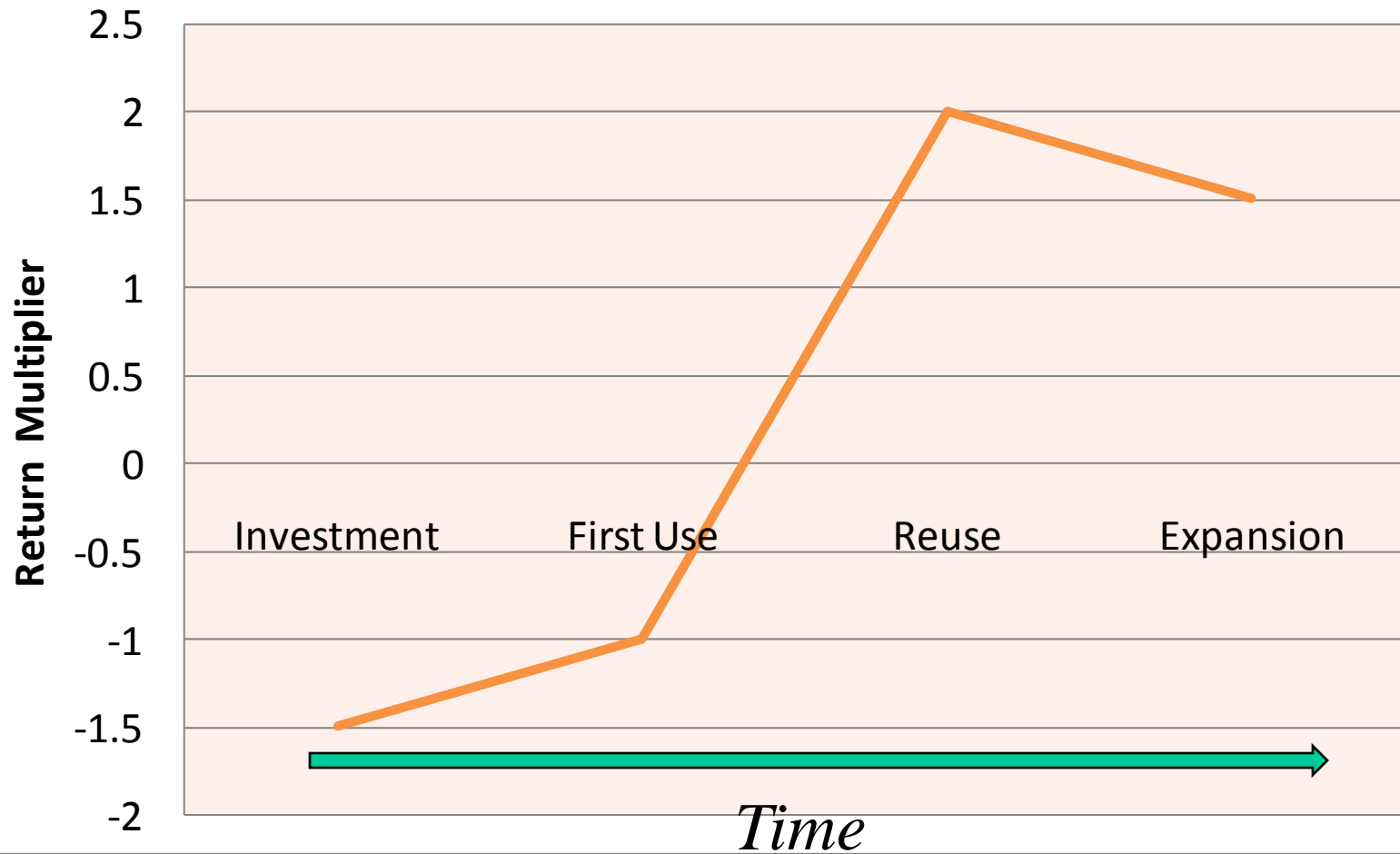
Verus Architecture Diagram



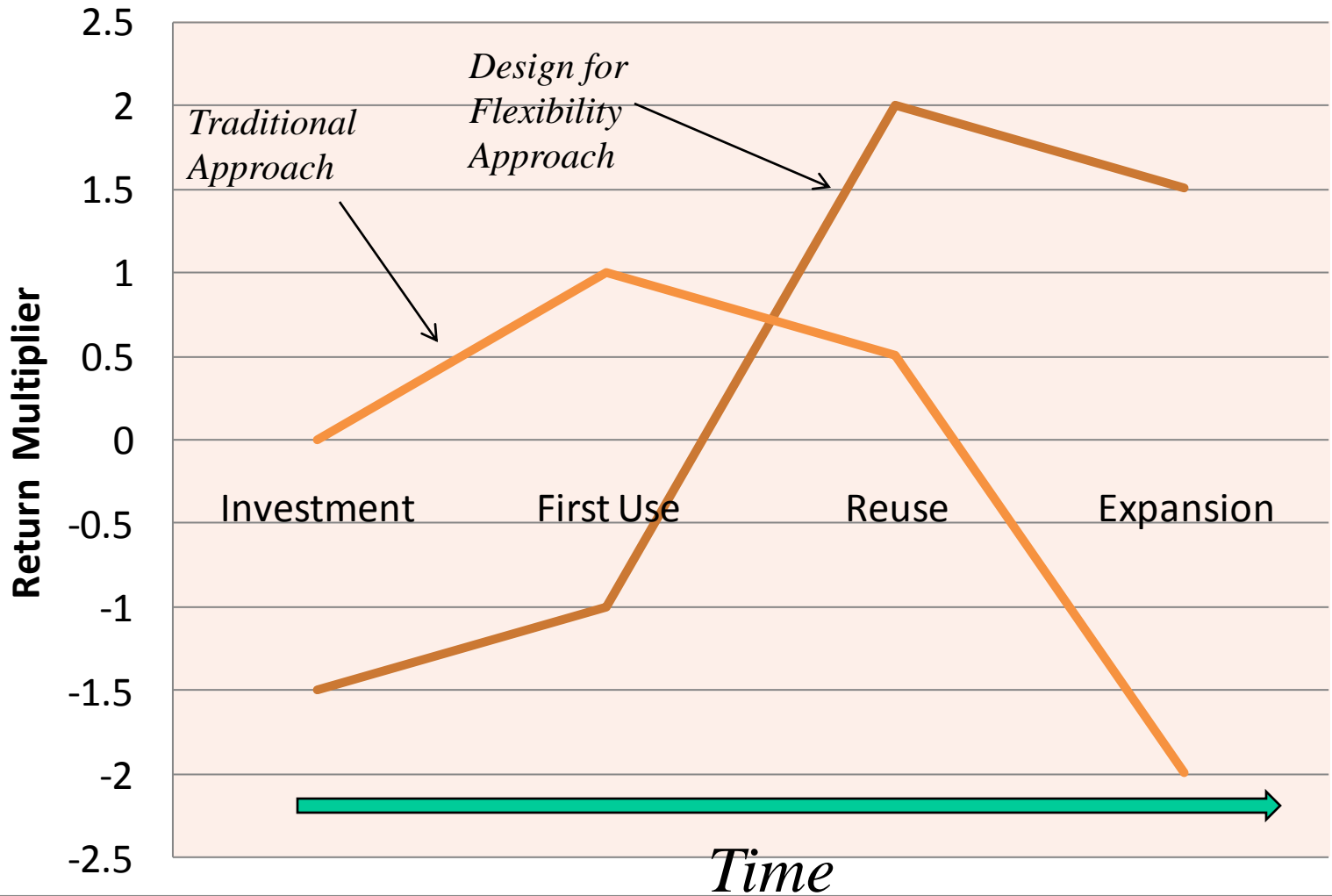
Verus Architecture - Results

- Most changes to software are isolated from other parts requiring less changes and shorter validation time
- Real-time performance not impacted by changes to non-real-time software updates
- Common templates/objects reduce time to change software that usually change
- Data Management is now a major system emphasis

Cost vs. Benefit Over Time



Cost vs. Benefit Over Time



Verus Architecture - Summary

- Architecture design is key to less costly and faster development of test systems
- Emphasis needs to be placed on elasticity for reuse
- Cohesion and coupling are key consideration of design decisions
- Investment in design is required for long-term payback