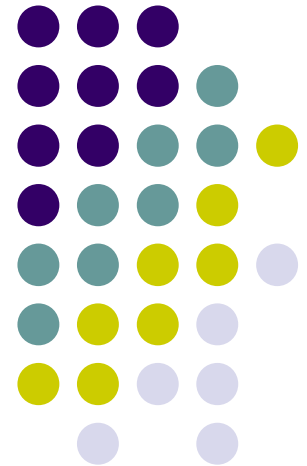


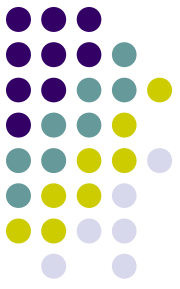
# Incrementally Migrating Semiconductor Equipment to a New Software Architecture

Wayne A. Lobb  
Engineering Director  
Foliage Software Systems, Inc.  
March 2003

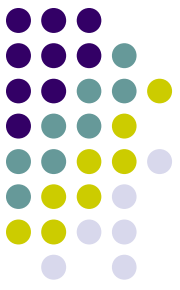


# Overview

---



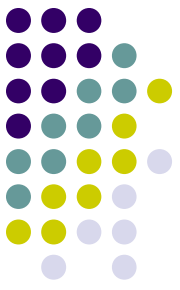
- Moving semi equipment to new software
  - Should you wait, start over now, or migrate now?
  - Massive implications for risk and ROI
  - No simple answers
- Migrating could be the right approach
  - Evaluate current software architecture (SAAM)
    - Envision and evaluate the new architecture (ATAM)
  - Estimate effort and risks of a migration approach
  - If it's the right approach, then plan and execute



# This Presentation

---

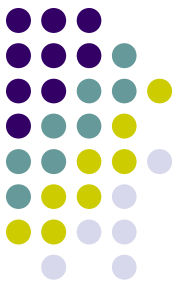
- Complements Foliage White Paper
  - Paper is at [www.foliage.com/whitepapers/#semi](http://www.foliage.com/whitepapers/#semi)
  - References are listed in next-to-last slide
- Focus here is on feasibility of migration
  - Rapid, low-cost approach to evaluating (SAAM)
  - Paper introduces SAAM; this presentation has details
  - How much a feasibility evaluation should cost
  - Step by step process for SAAM evaluation
  - Estimating return on investment (ROI) for migrating



# Existing Control System

---

- 500,000 – 2,000,000 lines of C/C++ code
- Monolithic design from 5-15 years ago
- Scores of programmers have worked on it
- Runs several tools solidly and dependably
- *But*
  - It's crufty
  - Slow and expensive to maintain and enhance
  - And sooner or later it'll have to be replaced



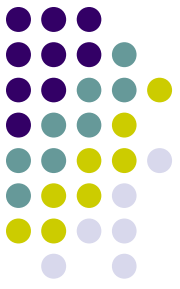
# Three Possible Approaches

---

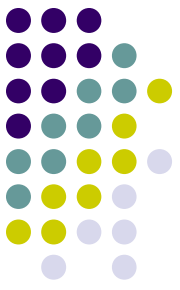
- Go on enhancing
  - Make no real changes, live with it for now
  - Risk: losing competitive ground, diminishing ROI
- Or start over
  - Start from a clean slate and rebuild everything
  - Risk: Big Bang projects often explode...
- Or migrate the architecture piecewise
  - Could save very significant expenses
  - Risk: might not be feasible

# Terminology

---



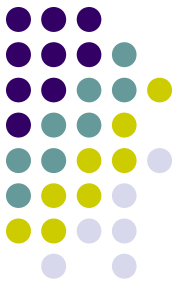
- Architecture
  - Components + their interfaces/APIs + interactions
- Enhancing
  - Working on software *within* its current architecture
  - Example: adding a new measurement algorithm to a metrology tool
- Migrating
  - Rebuilding component by component, not all at once
  - Example: reengineering existing measurement algorithms to operate in a new software architecture



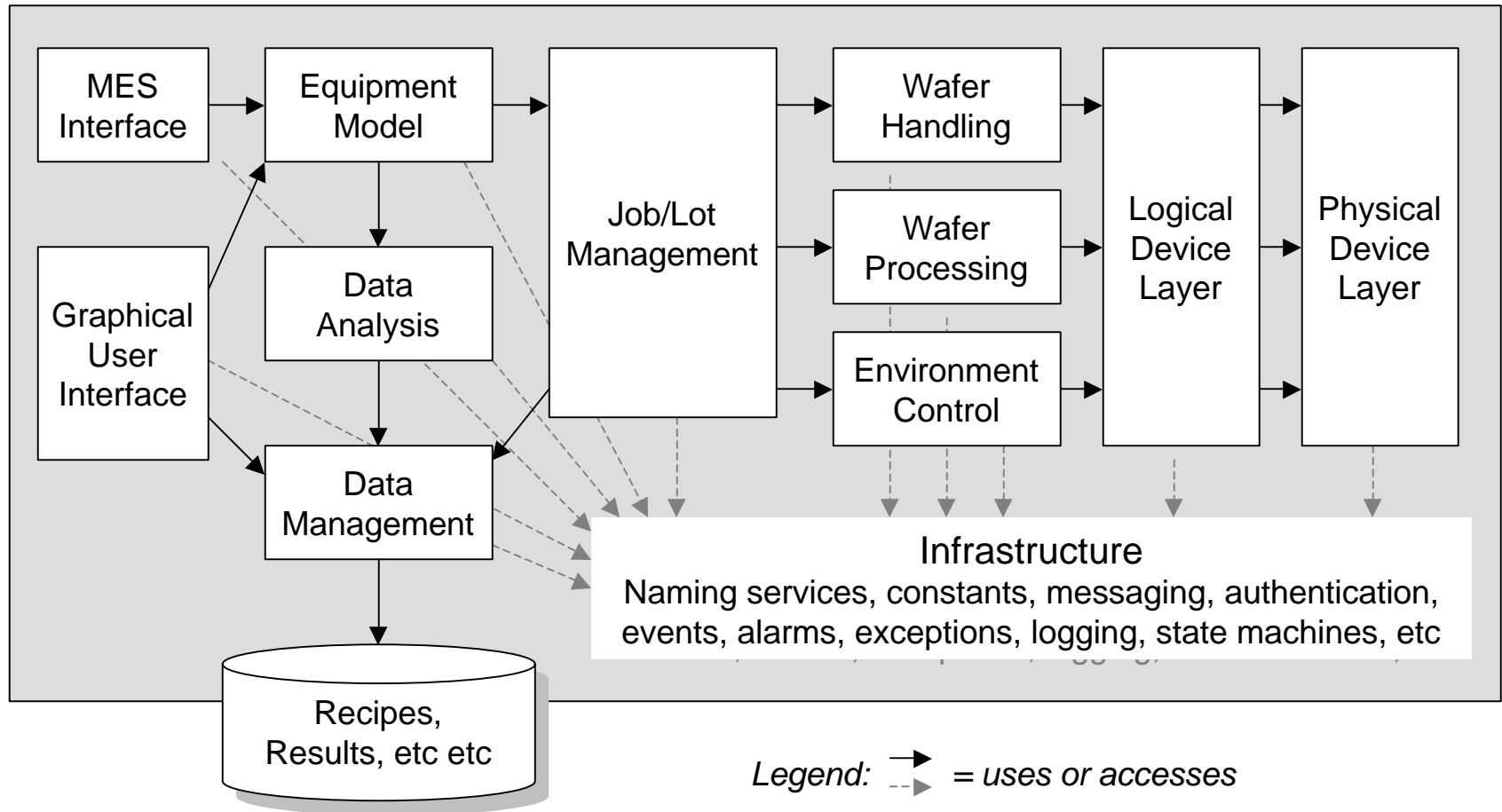
# Migration Feasibility: Integrity

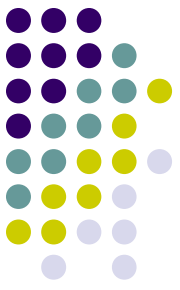
---

- Functional component integrity
  - Does source separation match functional separation?
  - Related to encapsulation
- The less integrity, the less feasible
  - Intertwined components live or die together
  - Replacing one forces rework/replacement of other
  - Soon you're rebuilding everything from scratch...
- To evaluate feasibility, use SAAM



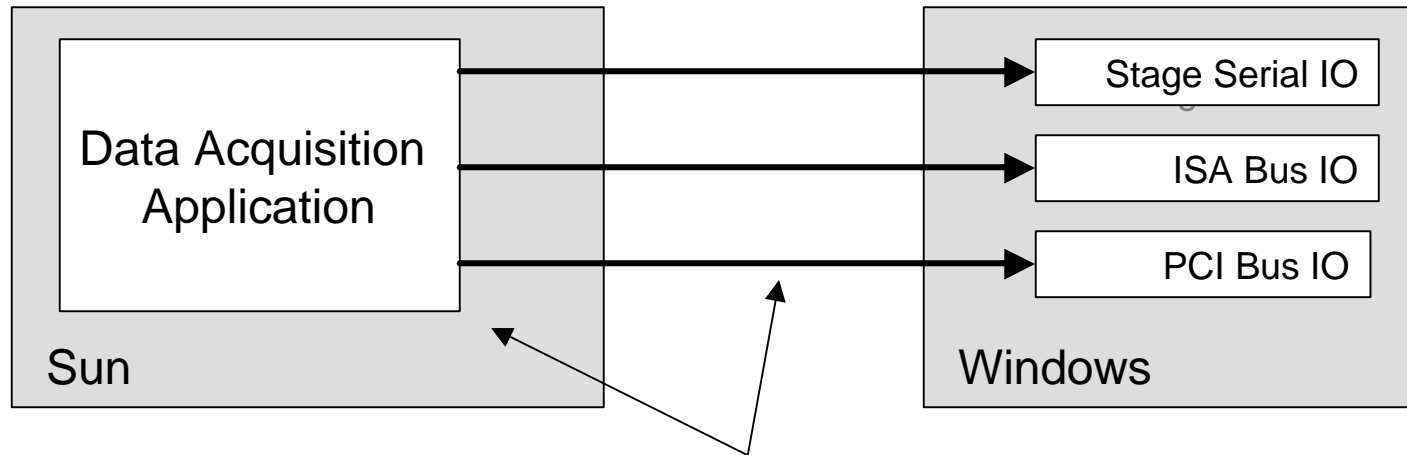
# Typical Functional Components



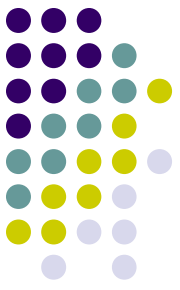


# Absence of Component Integrity

- Migrating data acquisition from Sun to PC, adding one device and replacing another

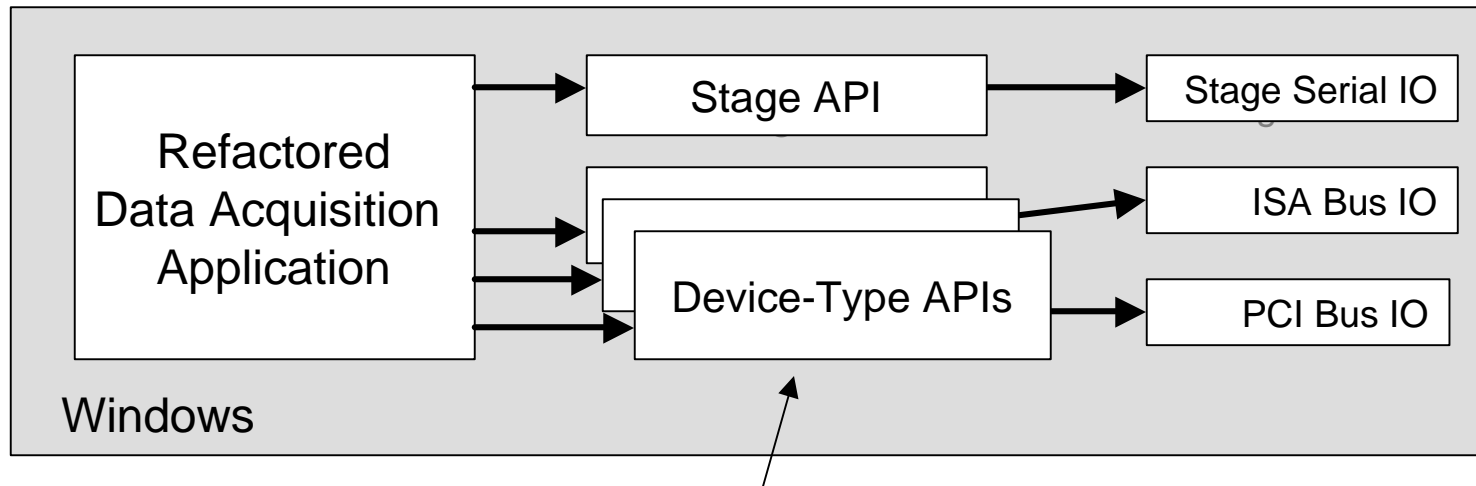


*Direct formatted-text physical commands, specific to individual devices, are scattered throughout the Data Acquisition Application and passed to the physical device layer. This forces a rewrite of the app. Cost: \$\$\$\$.*



# Creating Integrity via APIs

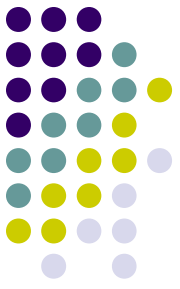
- Abstract the internal specifics of each device type into a *stable* logical API layer



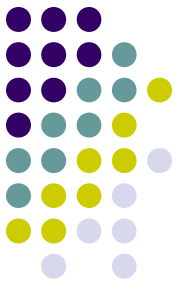
*Logical commands are mapped into physical commands only in the device abstraction API layer. Changing or adding a device mostly affects just one component, the logical API. Cost: \$ or \$\$, not \$\$\$\$.*

# SAAM

---

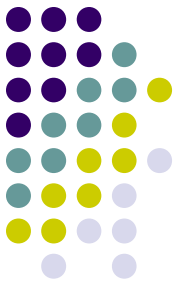


- “Software Architecture Analysis Method”
- Focuses on the *modifiability* of system
- From Software Engineering Institute (SEI)
  - Carnegie Mellon University, early 1990s
- Precursor of ATAM
  - ATAM: “Architecture Tradeoff Analysis Method”
  - Superset of SAAM, much broader and deeper
  - *Fully* evaluate current or envisioned architecture

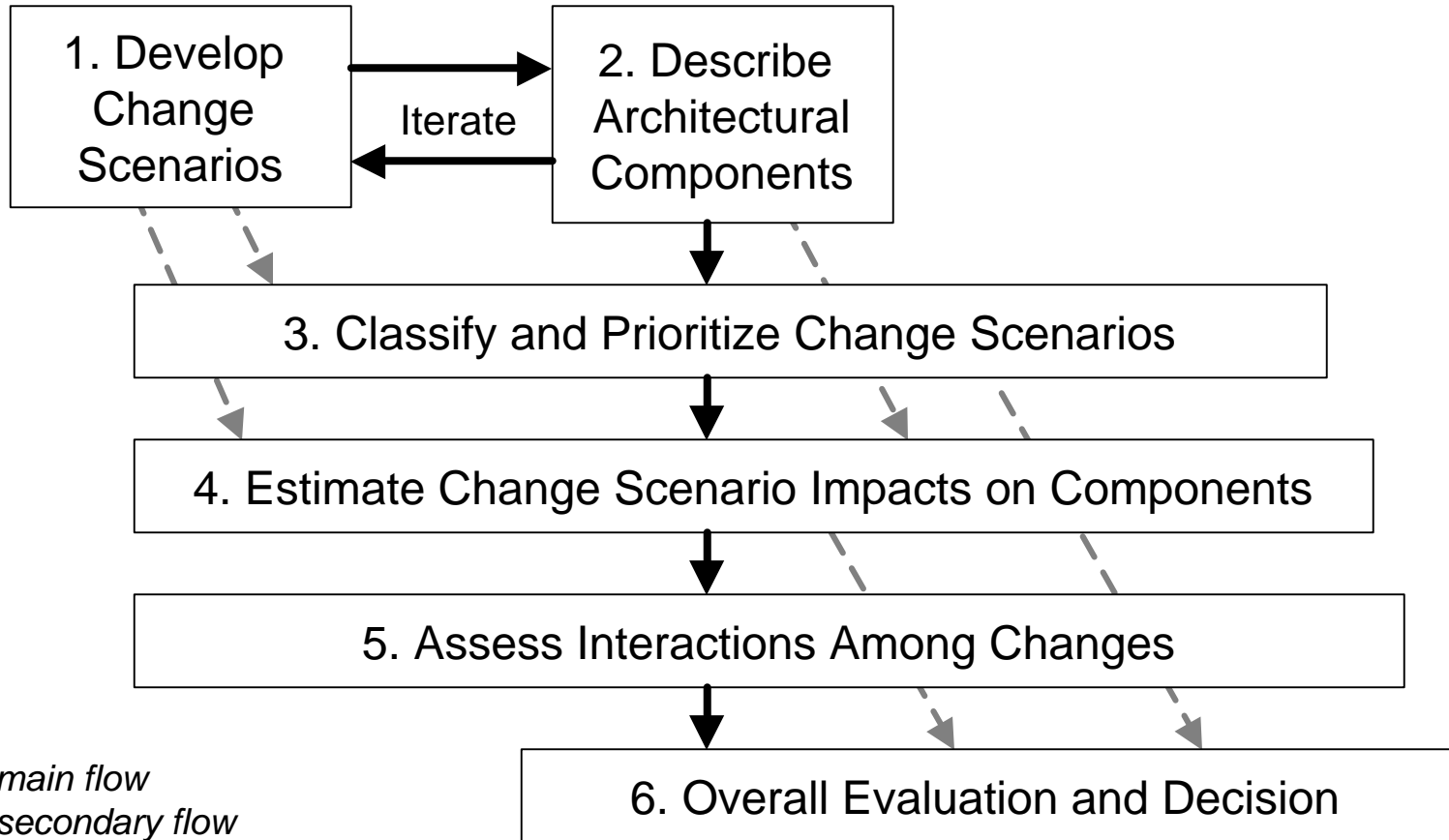


# SAAM-ATAM Comparisons

	SAAM	ATAM
Focus	Feasibility of modifying an existing system and its architecture	All aspects of an existing or envisioned software architecture
Quality Attributes Addressed	Modifiability, portability and “subsetability” (ability to deliver partial change efficiently), all via modification scenarios	Performance, reliability, availability, security, modifiability, portability, functionality, variability, “subsetability”, conceptual integrity
Evaluation Method	Estimated effort to carry out expected modification scenarios	Precise characterizations of all quality attributes via hierarchical use cases
Effort/Cost	5-10 stakeholders 2 days	10-12 stakeholders 2 weeks
Results	Estimation of how well system will support expected modifications: risk areas and non-risks	Estimation of how well system achieves <i>all</i> quality attributes: risks, non-risks, sensitivity points, tradeoffs



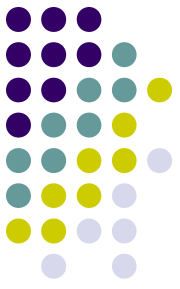
# SAAM for Migration Feasibility



## Legend

- ➔ = main flow
- = secondary flow or influence

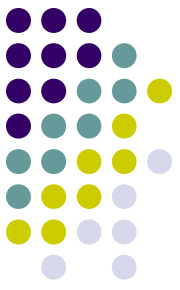
# SAAM 1: Develop Scenarios



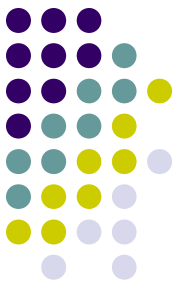
- Stakeholders develop change scenarios
  - Brainstorming, criticism-free environment
  - Record change/modification scenarios on the spot
- Must capture *all* major uses, now & future
  - Tasks per users: integrators, end users, developers, test, maintenance, quality control, sales...
  - Ascribe quality attributes: how fast, how long, at what expense, how frequently, with what training...
  - “Enable login over WAN to query and chart selected variables from the hundreds of available ones”

# SAAM 2: Describe Architecture

---



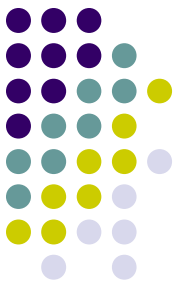
- Ideally, already fully documented
  - Often not so: bring in your technical experts
  - Don't trust memory on important items – do research
- Scenarios probe the architecture
  - “What has to change to enable logins over WAN?”
- Architecture suggests change scenarios
  - “These two subsystems use separate log files? How do you figure out the overall sequence of events if something goes wrong? Should we change this?”



# SAAM 3: Classify and Prioritize

---

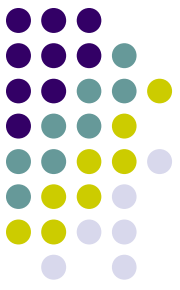
- “Direct” scenarios: already supported
  - Either the system already does this
  - Or the architecture already supports it
- “Indirect” scenarios: modification required
  - Not supported by the current architecture
- Vote on most important indirect scenarios
  - If N items, each voter gets  $\sim 0.3 \cdot N$  total votes
  - Voters can use votes in any way: 1 here, 4 there, ...
  - Vote *only* by importance, not by perceived cost



# SAAM 4: Assess Components

---

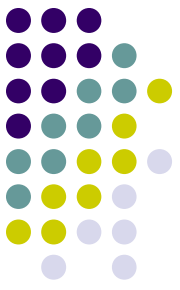
- Map indirect scenarios to components
  - Start with the most important indirect scenarios
  - Map scenarios to components that will have to change
  - Estimate *total* effort to implement each scenario
  - Estimates must include design, coding, test, doc etc
- What if it's too hard to map or estimate?
  - Need more information about the architecture?
  - Or need more information about the scenario?
  - If so, iterate portions of Steps 1 and 2...



# SAAM 5: Assess Interactions

---

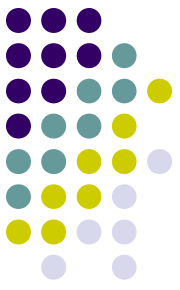
- This is a key step
  - Its value depends on accuracy & depth of prior steps
- Interaction: Two or more change scenarios require changes to the same component
  - Examine components involved in lots of interactions
  - If fairly unrelated scenarios, possible architectural flaw
  - Reconstitute components to remove interactions?
  - *Changes in components mean changed architecture*
  - *Interactions represent risk and cost when migrating*



# SAAM 6: Evaluate

---

- Seek a low-interaction migration route
  - Prioritized scenarios imply incremental migration plan
  - Can implement non-interacting scenarios in parallel
  - Might implement high-interaction scenarios in series
- Too much interaction implies infeasible
  - “Everything seems to depend on everything else”
  - Too little component integrity at the start
  - No route has low-interaction releases along the way
  - Don’t try to migrate: hold the course, or start over

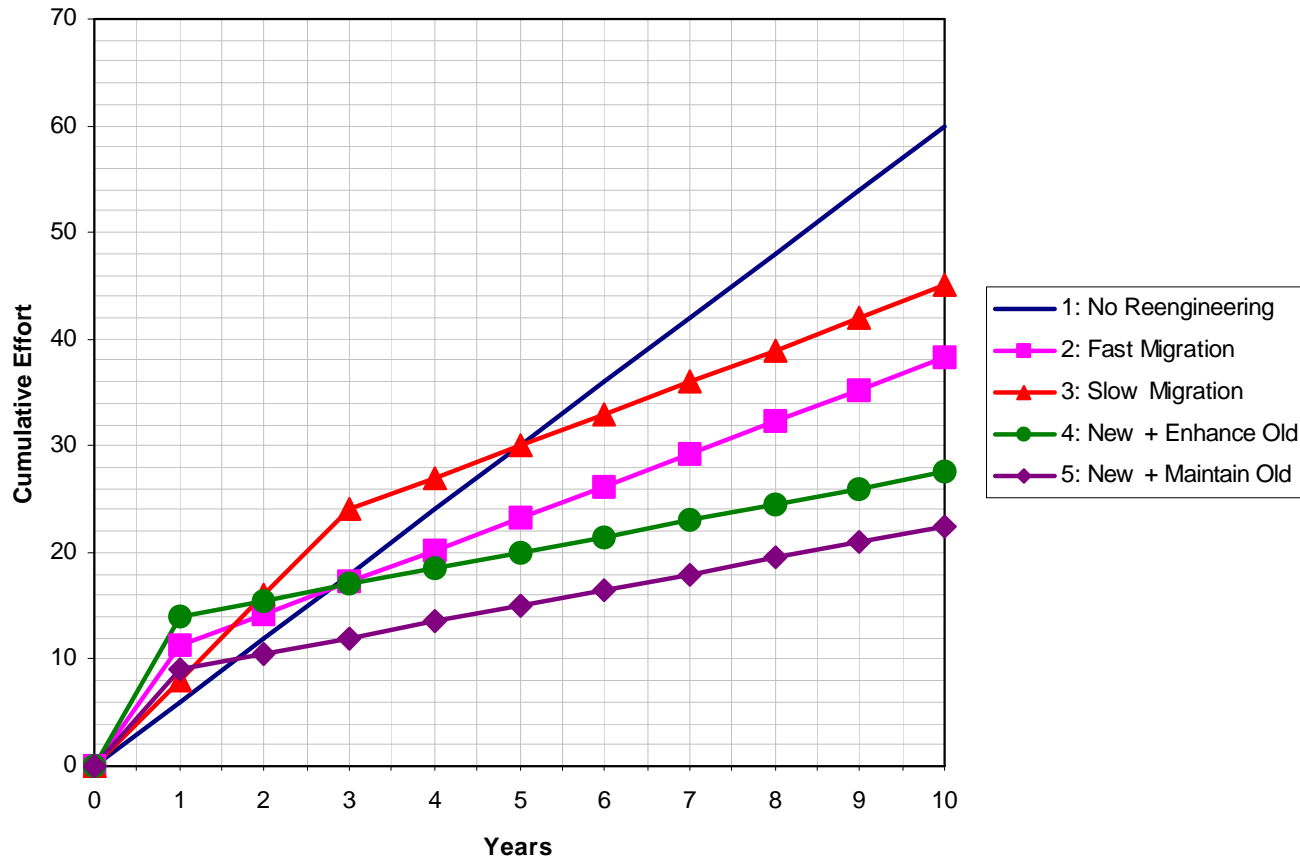
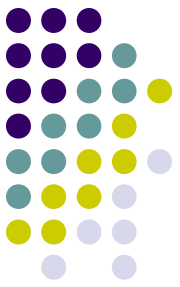


# Moving to a New Architecture

---

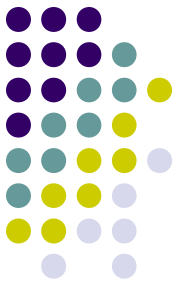
- Migration culminates in a new architecture
- Will the new architecture actually work?
  - SAAM helps determine feasibility of migration
  - Use ATAM on new architecture to control risk
- Cost of migration versus starting over
  - Consider comparative cost of starting over
  - Back-of-envelope often implies cheaper to start over
  - Examine such estimates *extremely* carefully...

# Example Migration ROI Analysis

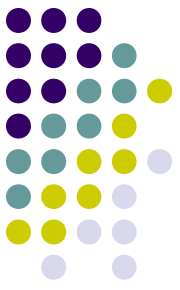


# Foliage Experiences

---



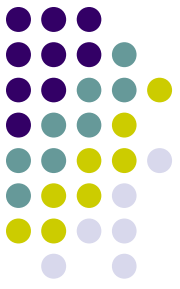
- **Successful: Sun to PC, adding CORBA**
  - Moved infrastructure from Sun to Windows
  - Replaced messaging system with CORBA
  - Under a year to the modified architecture
- **Mixed results: Migrate only GUI and DB**
  - Existing architecture had just a few huge components
  - Migrating only two of them was ultimately problematic
  - Too much interaction with the unchanged ones
  - Mismatches, latent bugs, performance problems



# Some Lessons Learned

---

- Get full buy-in from the stakeholders
  - Especially the technical staff: they'll make or break it
- Undocumented systems are nightmares
- Breaking up huge components is risky
  - Usually better off to rewrite them from scratch
- Pay special attention to infrastructure
- Take one incremental step at a time
  - Always have a complete, solid, running system



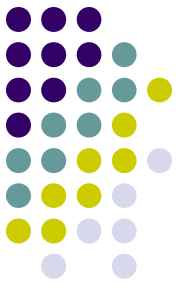
# References

---

- Alfred, C., “Using Architecture Challenges to Formulate Software Architecture”, Foliage White Paper, [www.foliage.com/whitepapers/index.shtml#technology](http://www.foliage.com/whitepapers/index.shtml#technology), 2002
- Clements, P., Kazman, R. and Klein, M., *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley, Boston 2002
- Delisle, N. et al., “Advanced Control System Architectures for Semiconductors”, Foliage White Paper, [www.foliage.com/whitepapers/index.shtml#semi](http://www.foliage.com/whitepapers/index.shtml#semi), 2002
- Delisle, N. et al., “Incrementally Migrating Semiconductor Equipment to a New Software Architecture”, Foliage White Paper, [www.foliage.com/whitepapers/index.shtml#semi](http://www.foliage.com/whitepapers/index.shtml#semi), 2002

# About

---



Foliage Software Systems, Inc., delivers custom software, integration services and technology consulting services. Since being founded in 1991, Foliage has completed more than 150 projects in the areas of semiconductor, medical, financial services, aviation and e-business. Engineers at Foliage average over 18 years of experience. Most have recent semiconductor experience. Employee retention has always been very high, which translates to consistency and tight-knit teamwork. The company is private and self-funded with annual revenues of over \$25 million, and has been named to the Deloitte and Touche Fast 50 as well as *Software Magazine's* Software 500. Foliage is headquartered in Burlington MA and maintains a presence in Silicon Valley. Visit [www.foliage.com](http://www.foliage.com).

Wayne A. Lobb, PhD, Engineering Director at Foliage, has 25 years of experience in various kinds of software engineering from computer-aided design to manufacturing automation to enterprise systems. During the 1980s and 90s he worked at FASTech Integration, now a part of Brooks-PRI Automation, on cell controllers and statistical process control. At Foliage he has been working on semiconductor equipment control systems. His doctorate is in mathematics from the University of Illinois.