Assembly-line Driven Development

David P. Brown (sqa@compumonk.com)

More With Less

- Feeling squeezed?
  - Market rapidly changing
  - Business forced to do more with less
  - Current solutions not effective
  - Is testing really valued, or a necessary evil?

More With Less

- During the 1950s, US went through a manufacturing boom
- Europe's capacity for manufacturing was decimated by WW II
- US could make anything, and the world was buying
- As the decade ended, Japan came out of nowhere with innovative products, lower cost and better quality

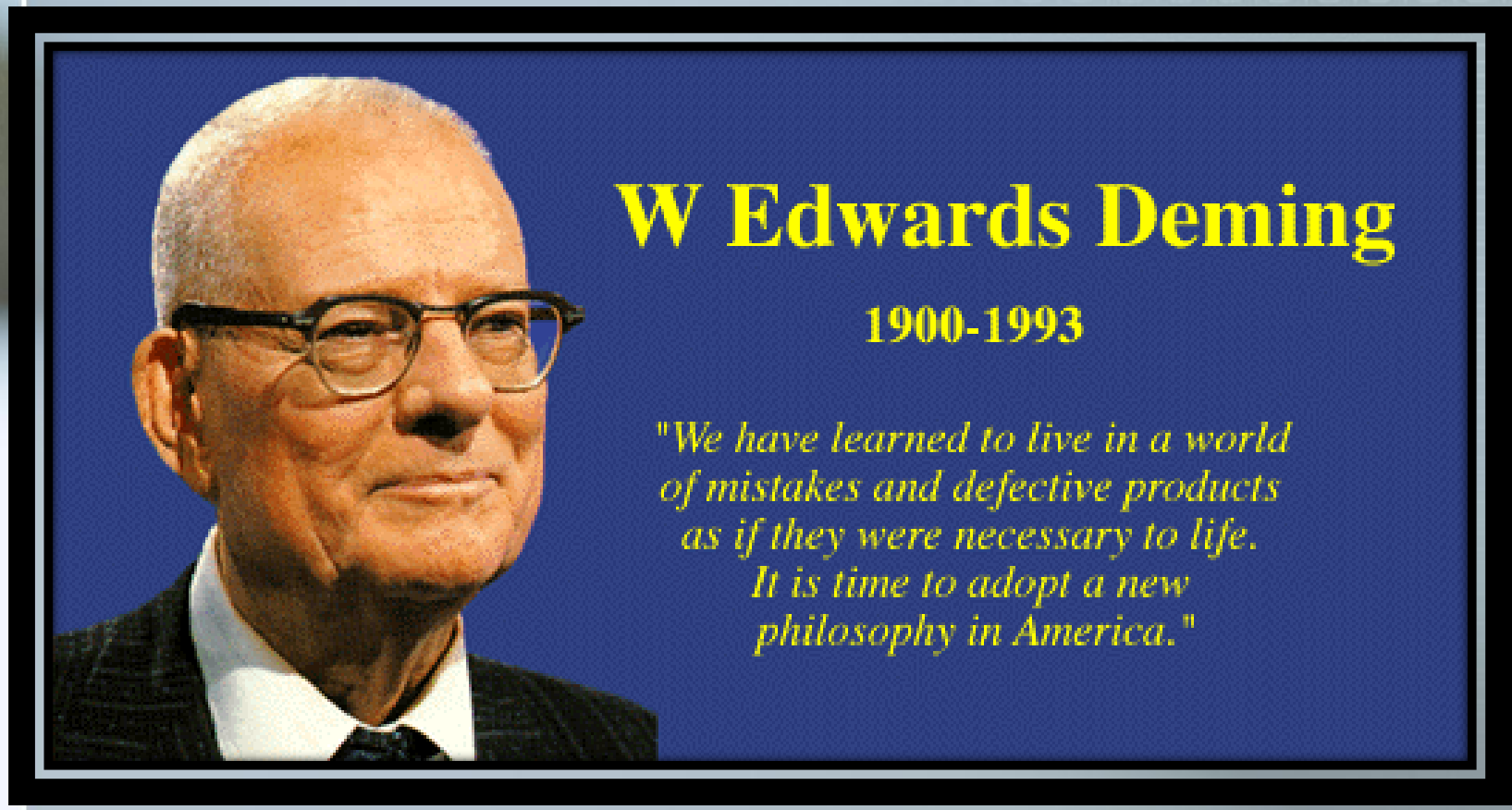
More With Less

- What happened?

More With Less

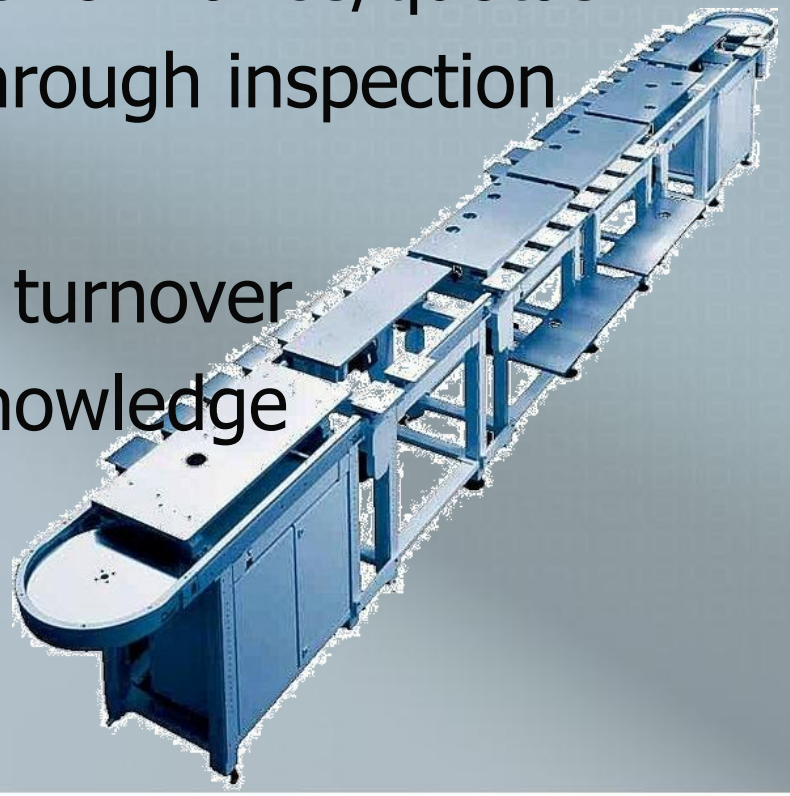
- US producing products with little innovation and poor quality
- Quality efforts focused on inspections with some estimates at 38% rejection rate
- Costs inflated to compensate
- Japan changed, the US was behind the curve
- What was the catalyst of this change?

More With Less



More With Less

- Key Industry Issues
  - Dependence on requirements
  - Management by performance/quotas
  - Focus on quality through inspection
  - Competition within
  - High management turnover
  - Little transfer of knowledge



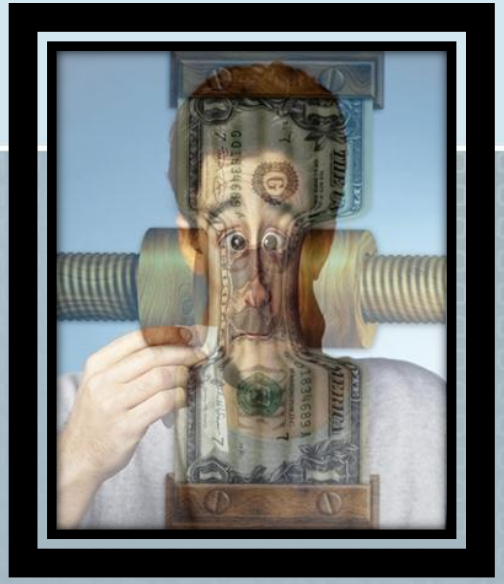
More With Less

- Software Development Weaknesses
  - Dependency on requirements
  - Management by performance/quotas
  - Focus on quality through testing
  - Competition within
  - High management/talent turnover
  - Little transfer of knowledge

More With Less

- The manufacturing industry changed quickly and radically
- Those that did not are in the dustbin of history
- The software development discipline must change as well

More With Less

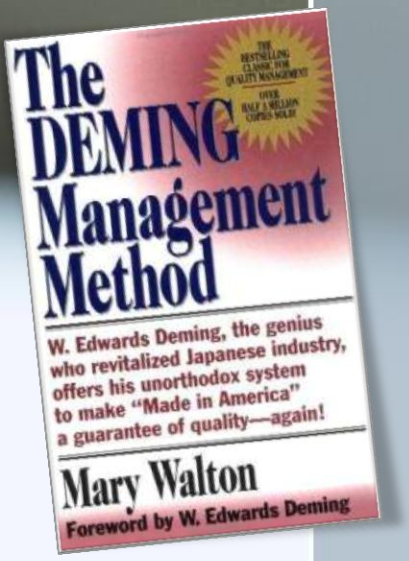


- Bending the cost curve
  - Iterative Development
  - RAD
  - RUP
  - Agile/SCRUM
  - Strict cost control/gate processes
  - Offshoring
- Have any of these really been effective?

Key Facts

- Software development is still an artistic venture, a craft
- You cannot test quality into software
- The more time spent on requirements, the more successful the project
- Quality is the responsibility of everyone, not just test

Change



- Deming Management Method
  - Identify and reduce variability
  - Acquire domain knowledge
  - Document and consistently adhere to process
  - Jettison competition
  - Eliminate necessity of requirements
  - Do away with reliance on individual effort
  - Reduce dependence on testing

Competition

- Competition is a great thing in the marketplace
- Works well between companies, but not ***within*** companies
- Must break down anything that puts barriers between stakeholders and development
- There cannot be one defined "channel" of stakeholder communications

Knowledge

- All key workers must have a thorough understanding of the company's domain, business and priorities
- Requires investment a training regimen managed by the business
- Necessitates less reliance on contractors, especially in leadership roles

Process

- People work best under a defined process
- A process must be simple, easy to understand and adaptable
- Complex processes, especially those with fixed and immovable “gates” breed non-compliance
- Processes must be thoroughly documented and training provided

Reduce Variability

- Environments (IT)
  - Rigidly controlled production environment
  - Well managed test environments
  - Configured as close as possible to production
  - Follow same processes as pushes to production
  - Refreshable

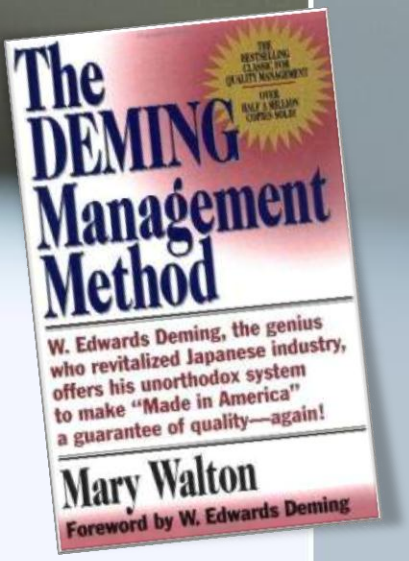
Reduce Variability

- Environments (COTS)
  - Target consumer representative
  - Refreshable
  - Same systems available to all
  - Virtualization is very effective

Reduce Variability

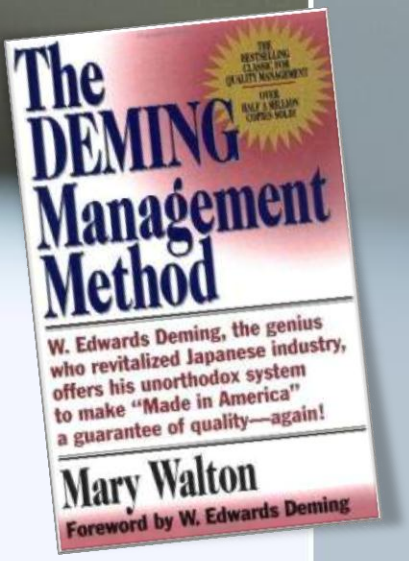
- Development systems
  - Centralized "push a button" build from source system
  - For COTS, include installers
  - Rigid check-in processes
  - Any change can be tracked to a work item/requirement or defect repair
  - All systems can be accessed by test for verification/auditing
  - Tool automation (reduce admin)

Recap



- Deming Management Method
  - Identify and reduce variability
  - Acquire domain knowledge
  - Document and consistently adhere to process
  - Jettison competition
  - Eliminate necessity of requirements
  - Do away with reliance on individual effort
  - Reduce dependence on testing

Recap



- Deming Management Method
  - Identify and reduce variability
  - Acquire domain knowledge
  - Document and consistently adhere to process
  - Jettison competition
  - **Eliminate necessity of requirements**
  - **Do away with reliance on individual effort**
  - **Reduce dependence on testing**

Assembly-line Driven Dev

- Assembly-line Driven Development
  - Quality through design instead of formal inspection (testing)
  - Heavy involvement of all players through the entire lifecycle
  - Change accounted for and controlled
  - Utilizes a risk-based model
  - Minimal end-stage testing
  - Concepts we are already familiar with
  - **It is not "A.D.D."** (although I am)

Assembly-line Driven Dev

- High-level framework
- First cut, needs refinement
- First presentation, ditto above
- Everything presented here has been used effectively in practice
- Concepts modeled after 40 years of successful implementation in manufacturing
- WARNING: Don’t try this at home!

Inception

- "Back of the Box" requirements
  - Short, one paragraph, statement of the intention of the project
  - What problem are we trying to solve?
  - What need are we trying to fill?
  - Timeline for delivery
  - High-level bulleted list of features
  - Ranked as "Must", "Should" and "Nice"
  - Posted on everyone's wall!

Inception

- Personas
  - Develop one or personas depicting the target consumer or user
  - For a COTS product, might be one well crafted persona
  - For a IT product, will probably be several
  - Posted on everyone’s wall
  - Brought to every meeting
  - Give them names and personalities!

Inception

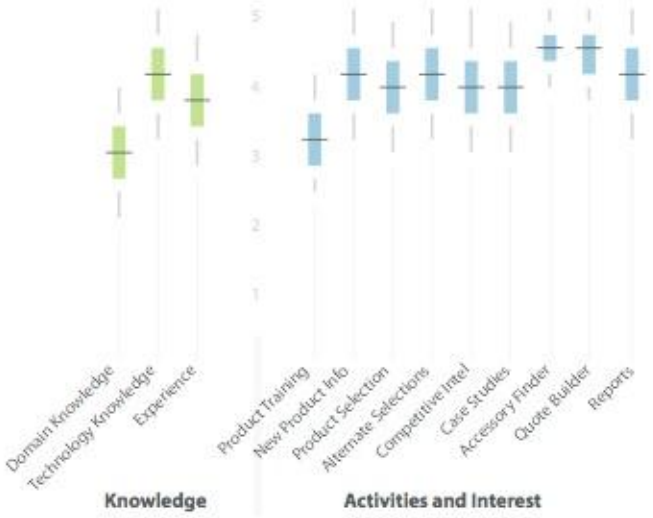
The Moderately Seasoned Professional

Michael

"I'd like to see a good, better, best."

MIChAEL HAS BEEN IN SALES for YEARS AND HAS BEEN SELLING AND OTHER products for most of them. he's pretty comfortable with the Symbol products and isn't that interested in basic product info, but finds himself wondering if there's a better alternative than the product he's suggesting. It's a challenge keeping up to date on all the product info from Symbol and the other lines he sells. he'd love to see something that recommends a good, better, and best option when they're available as well as showing him where the particular model stacks up against other competitive products.

The right tool for Michael helps him pick the best product while recommending other alternatives. It also has case studies with examples of how other more seasoned salespeople have been able to upsell in similar environments and applications. Accessories and add-on services for Symbol scanners are a must. And if it is a frontend for Solution Builder that would be a huge bonus.



Age: 42

Occupation: r egional Sales

Primary Use
- Case studies
- Alternate product selections (good, better, best)
- Accessory finder
- Product info for new products or product refresh
- As a frontend to Solution Builder

Goals
- Pick the right product and find better alternatives.
- Keep up-to-date on competitive intelligence.
- Increase sales volume.
- Increase accessory and add-on sales.
- Close more deals faster.

Influencers
- Easy-to-use
- Speed
- New product info
- Ability to run reports on open vs. closed quotes
- Ability to generate quotes

Frustrations & Pain Points
- having multiple usernames and passwords. he already has several for the different applications he uses at work and email – he doesn't need another one.
- A tool that leaves him in the dark

Source: Todd Warfel "Data Driven Personas": http://www.slideshare.net/toddwarfel/data-driven-personas

Inception

- Initial Narratives
  - Short stories developed in partnership or even by the sponsors
  - Tied to the "Back of the Box" bulleted requirements
  - Brings the system to life
  - Used as input into the design

Design

- UI Mockups
  - Develop UI mockups in Visio, PowerPoint or even by hand of each interface
  - Revise iteratively with the sponsors
- Functional UI Prototypes
  - Bring the mockups to life
  - Continue to revise iteratively with the sponsors
- Architecture
  - High-level design supporting framework

Design

- Narratives
  - Continue to develop and refine narratives per the evolving UI
- Functional Design
  - Screenshots of the UI
  - Field descriptions
  - Critical business rules
  - Map narratives to document sections
  - This document comes under change control

Design

- Benefits
  - Compliments the fact that people are visual by nature
  - Sponsors and the development team both gain insight and understanding of what can and cannot be accomplished
  - Some of the framework can be developed early when the understanding is clear

Construction

- Work Breakdown
  - Features and functions broken down into manageable and assignable chunks
  - Work prioritized to complete “Must” items first

Construction

- Test Cases
  - Test cases are written using the functional design and narratives as a basis
  - Organized and mapped to the work breakdown
  - Written in priority order
  - Reviewed initially by development
  - Reviewed then by the sponsors
  - Priority must be placed on reviews!

Construction

- Framework
  - While initial set of test cases are being written, development will be building up the plumbing
- Functionality
  - As areas of the work breakdown have completed test cases, development begins work
  - Coding uses the test cases as the basis

Construction

- Unit Testing
  - Coders use the test cases as their unit testing
  - Testers will fully test those high priority/high risk items
  - Testers will spot-check those lower priority items
  - Sponsors have the ability to spot check as well

Construction

- Reporting
  - Daily reports (preferably web-based) produced showing the status of each work breakdown item
  - Demonstrate estimated work, actual work or work to date and estimated completion

Construction

- Change
  - Change happens!
  - Coders determine that the envisioned function is not possible
  - Sponsors determine that functions need revision or expansion (or even removal)
  - Changes negotiated between development and sponsors
  - Changes reflected in test cases, narratives and function design

Construction

- Change
  - For changes affecting scope, ability to quickly determine what uncompleted can fall off
  - Changes affecting other work items, those test cases can be brought into this work item
  - Changes affecting already coded functions, those test cases can be brought into this work item for regression

Construction

- End-game
  - As deadlines approach, if the project is behind, easy to determine what will/will not be completed (instead of last minute)
  - Sponsors know continually what is and is not complete
  - Sponsors determine if more time or resources are needed
  - Sponsors determine when we are done

Testing

- Testing is built into the entire lifecycle
- Once construction is deemed complete, team moves into a formal System Test cycle
- Test cycle is short and targeted on the high priority items through test cases and narratives
- Sponsors determine when test cycle is complete

Testing

- System Test
  - Reports generated and available “live” (real-time)
  - Review board composed of development and stakeholders classify and determine if/when reported issues are resolved
  - Changes will only be considered if absence would invalidate deliverable

Team Structure

- Team Leadership
  - Sponsors
  - Project Manager (for large efforts)
  - Business Analyst
  - Development Manager
  - Test Manager
- Team Members
  - Developers
  - Testers

Team Work Assignments

- Inception
  - Team leadership and members are working on inception deliverables
- Design
  - Team leadership and members are working on design deliverables under the direction of the Business Analyst
  - Some developers dedicated to prototyping
  - Some developers dedicated to architecture

Team Work Assignments

- Construction
  - Leadership managing resources/work
  - Test Manager publishes tactical metrics & status
  - Developers working on framework and functionality as test cases come available
  - Testers producing test cases, responding to audit updates and changes
  - Developers maybe re-tasked to help develop test cases

Team Work Assignments

- System Test
  - Leadership managing resources/work
  - Test Manager publishes tactical metrics & status
  - Testers and developers executing test cases & narratives
  - Some developers working approved defects/changes or unfinished work

Risk Evaluation

- Impact of failure
  - Feature importance (“Must”, “Should”, “Nice”)
- Likelihood of failure
  - Process complexity
  - Functional complexity
  - Continually evaluated through the lifecycle
- Feeds into the determination of testing level

Conclusion

- Lifecycle
  - All team members fully engaged through the entire lifecycle
  - Sponsors involved and driving the lifecycle (they have control)
  - Changes reacted to quickly, efficiently and with proper documentation

Conclusion

- Requirements
  - Requirements gathering spread throughout the lifecycle
  - “Visual” design concept

Research

- The Deming Management Method
- Cooper Design Methodology
- Agile/SCRUM

Books

- The Deming Management Method (Mary Walton and Edwards Deming)
- Cooper Design Methodology

  The Inmates Are Running the Asylum (Alan Cooper)
- Test As Design

  Specification by Example: How Successful Teams Deliver the Right Software (Gojko Adzic)

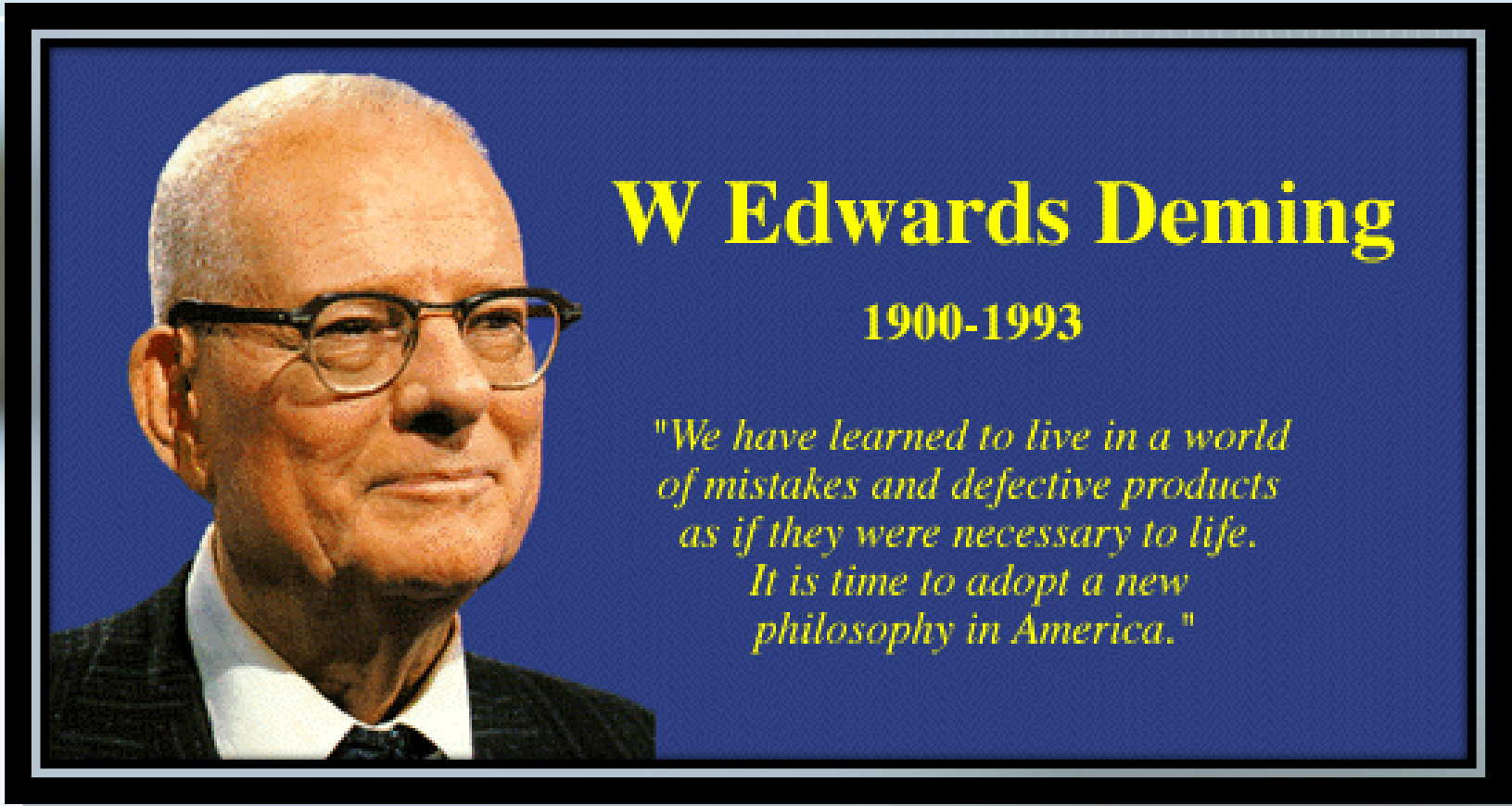
Knowledge

- Be ahead of the curve!
  - Must acquire knowledge and skillsets
  - Less focused on craft
  - More focused on design, business need and risk evaluation
  - Be able to speak to developers and the business in their own language
  - Be able to read code (don’t panic, it is not that hard)

Disclaimers

- This model works well for a variety of verticals
- There are some exceptions
  - Life safety (automotive, medical devices)
  - Rockets
  - Financial
- Risk model can still be applied, but there will always be a high testing cost

Questions?



David P. Brown (sqa@compumonk.com)