AEgis TECHNOLOGIES

Capabilities-based Lifecycle Management -or- "What Artifacts do I need"

David A. Cook, Ph.D.

The AEgis Technologies Group, Inc.

Purpose of This Talk

- **Define artifacts**
- **Discuss Lifecycles**
- **Discuss what artifacts you need**

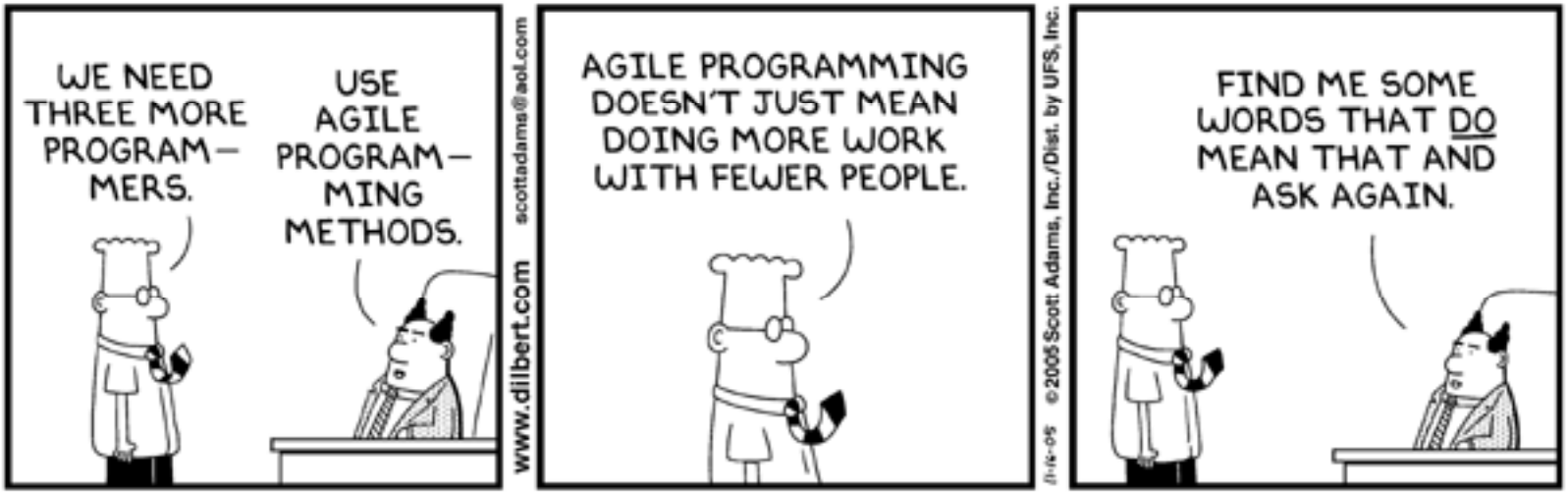
Lifecycles

Capability	Pure Waterfall	Code & Fix	Spiral	Waterfall Modified	Prototype	Agile
Poorly understood rqmts	Poor	Poor	Excellent	Fair to Excellent	Excellent	Excellent
Poor Architecture	Poor	Poor	Excellent	Fair to Excellent	Poor to Fair	Poor to Fair
Highly Reliable System	Excellent	Poor	Excellent	Excellent	Fair	Poor
System Growth Built in	Excellent	Poor to Fair	Excellent	Excellent	Excellent	Excellent
Risk Management	Poor	Poor	Excellent	Fair	Fair	Fair
Predefined Schedule	Fair	Poor	Fair	Fair	Poor	Poor to Fair
Midcourse Correction	Poor	Unknown	Fair	Fair	Excellent	Excellent
Customer Visibility	Poor	Fair	Excellent	Fair	Excellent	Excellent
Management Visibility	Fair	Poor	Excellent	Fair to Excellent	Fair	Fair
Low Management and developer skill level	Fair	Excellent	Poor	Poor to Fair		Poor
Low Overhead	Poor	Excellent	Fair	Excellent	Fair	Fair

Rapid Development (McConnell, 96), mods by Cook (2008)

Lifecycles

- **ALL lifecycles (except the very small projects) have some combination of**
  - **Requirements**
  - **Design**
  - **Code / Implementation / Build**
  - **Test**
  - **Maintain**



Agile?

- **Agile lifecycles are NOT an excuse to return to the old "code – debug and fix – code some more" school of development.**

- **Agile is NOT an excuse to skip requirements and design. Instead, it is an approach that changes the way you perform requirements and design. However, requirements and design artifacts are still needed.**

Before Requirements

- Concept of Operations (CONOPS)

- Purpose – high level description of
  - What the system will do
    - Training system?
    - Fielded use? In what capacity?
  - What the system will not do

  - What are the limits of the proposed system?

CONOPS Philosophy[1]

- Users Perspective
- Holistic (operational) view of system
  - vs. individual pieces or functions or interfaces
- First step of a development
  - identify user types and primary modes
- Focus is on prioritizing user requirements at a VERY high level
- Iterative
- Results of Conceptual Analysis

[1] See "The Concept of Operations: The Bridge from Operational Requirements to Technical Specifications", Fairley & Thayer, 1996.

Essential Elements

- Description of current system or situation
- Description of needs that motivate new system
- Modes of operation
- User classes / characteristics
- Operational features of the new system

Essential Elements

- **Priorities among proposed operational features**
- **Operations scenarios for each operational mode and class of user**
- **Limitations of proposed (new) system**
- **Impact analysis**

When do you build a CONOPS?

- **Before the decision is made to develop a system (to support decision)**
- **Before RFP**
- **After the award of contract, to help developer better understand user needs**

CONOPS Outline

- Scope
- References
- Current system
- Justification / nature of proposed changes
- Concept of operations
- Operational scenarios
- Summary of impacts
- Analysis of proposed system

Living Document

- **CONOPS should be updated and maintained during the entire lifecycle.**
- **Update to reflect new users, system design, operational policies, user needs.**
- **Must be under configuration management**

- **The goal of the CONOPS is “How can I justify and fund this project”**

Requirements

- **Goal of requirements artifact collection is tracability**
  - **From requirements to customer**
  - **From requirements to design and code**

- **Need SRS or equivalent**
- **Focus on “Why” not “How”**

Design

- **Need five different types of design artifacts**

- **#1 – Architectural artifacts**
  - **The conceptual design of the system**
  - **Compartmentalized image**
    - **Ex – a car has “brakes”, “suspension”, “steering” “drivetrain”**

Design

- **#2 – need interfaces into and out of the system and subsystems**
- **Shows how the system will be used, and how to use it**
- **Show data flow, pre & post conditions, format of data, references to database schemas, etc.**

Design

- #3 – Module design
  - Not as important as other design artifacts – can be simple description of the function
  - Remember – the coder will need some latitude

Design

- **#4 – Interfaces to/from the modules**
- **A much lower level set of interfaces, showing parameters, globals used, side effects to the database**
- **Needs to show globals USED and MODIFIED**

Design

- **#5 – Data artifacts**
- **Need the equivalent of a “data dictionary”. A centralized listing of all data that is not declared inside of a module (depending upon the language).**
- **Need source, format, attributes, default values, who owns it, who uses it.**
- **Database schemas**

Code artifacts

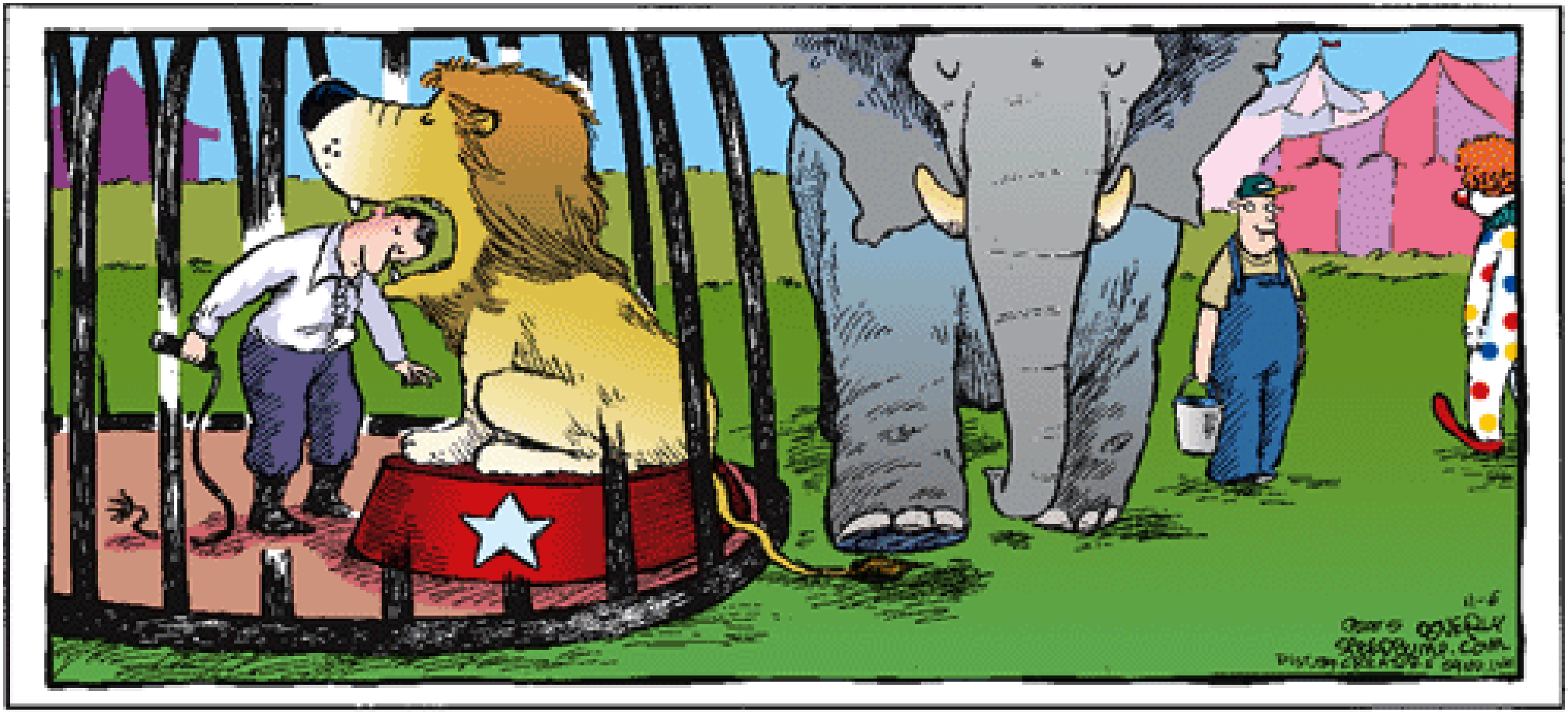
- First of all, need to define a standard for code writing
- As a minimum
  - Parameters used (and directionality)
  - Globals used and modified
  - Description of module
  - Author, and when written
  - Requirements it satisfies
  - Database use and modifications
  - “Unique” things

“Unique Things”

- **Real-time constraints**
- **Database constraints (exclusive access)**
- **Parallelism constraints**
- **Network constraints**
- **Anything that a maintenance programmer will need or want to know**

Unique things

- **Show any unexpected (or unusual) side effects**



Code artifacts

- History of module in terms of testing
- Review status (when, what type of review)
- Possibly testing information

COTS/GOTS

- How it was acquired
- Purpose
- Limitations
- “Glue” needed
- Run-time conditions
  - Libraries needed
  - OS
  - Supporting files

Testing artifacts

- **Baselined test cases**
  - **Inputs**
  - **Outputs**
  - **Expected values**
  - **Dates of test**

Maintenance artifacts

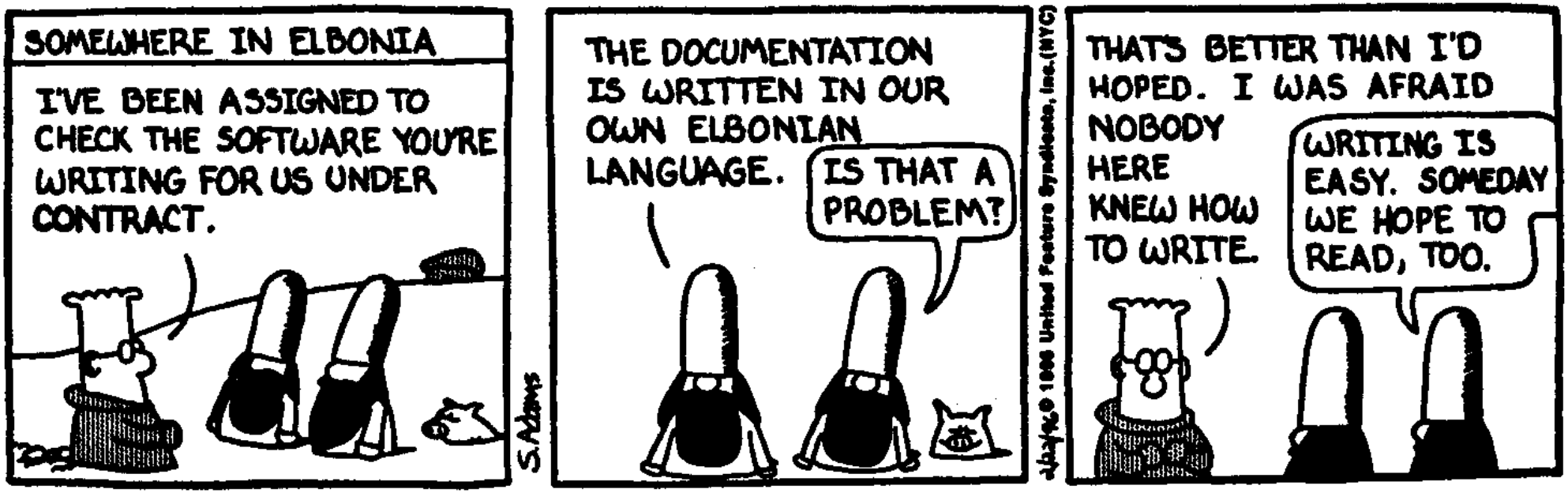
- CM artifacts
  - History of changes
    - Who
    - When
    - Why
    - “Old code”
    - Clearly identified “new” code

Maintenance

- Updates to other artifacts
  - Changed requirements
  - Changed design
  - Changed code
  - Changed documentation
  - Changes to testing results

Documentation

- **Users manual – depending on size and complexity of program**
- **Installation instructions**
- **What else?**



Three last things

- **What happens when you are contracting our all/most/part of the software?**
- **Do you have a list of deliverables that give you control and ownership of the artifacts you need?**
- **How do you control artifacts? Do you have a good CM program in place?**

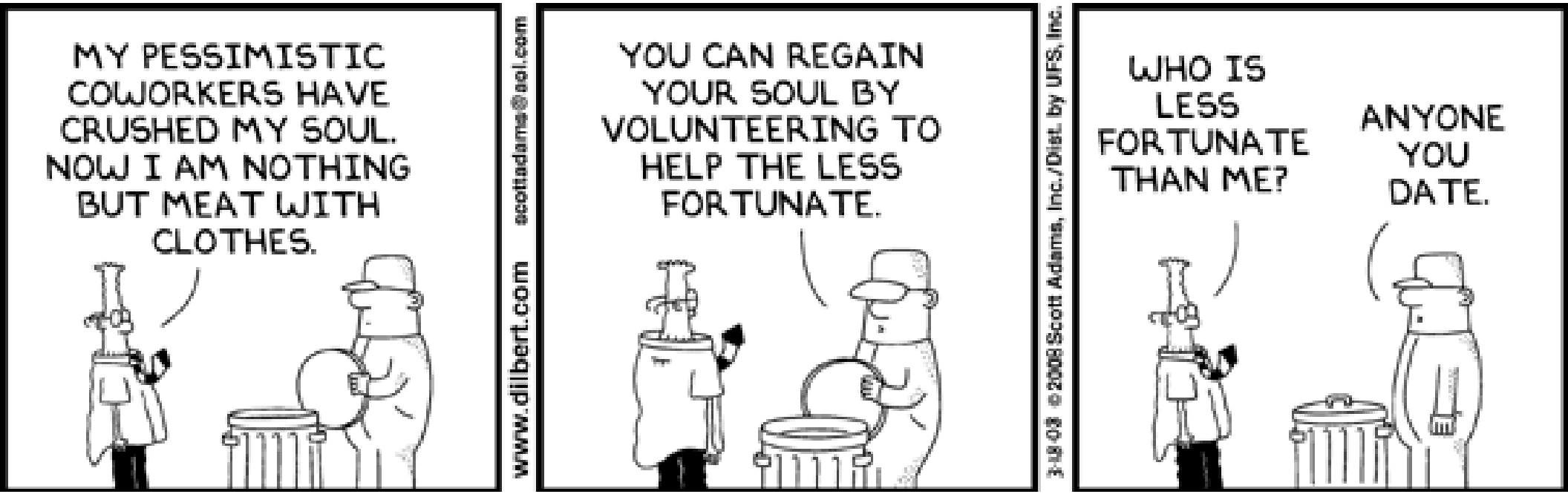
Summary

- **You need enough artifacts to "tell the story" for future generations of maintenance**
- **The artifacts will help you get maintain quality, prevent errors, speed future maintenance**

Once upon a time there were 3 little pigs, so I ate them. End of story. Now go to sleep!

Purpose of this talk

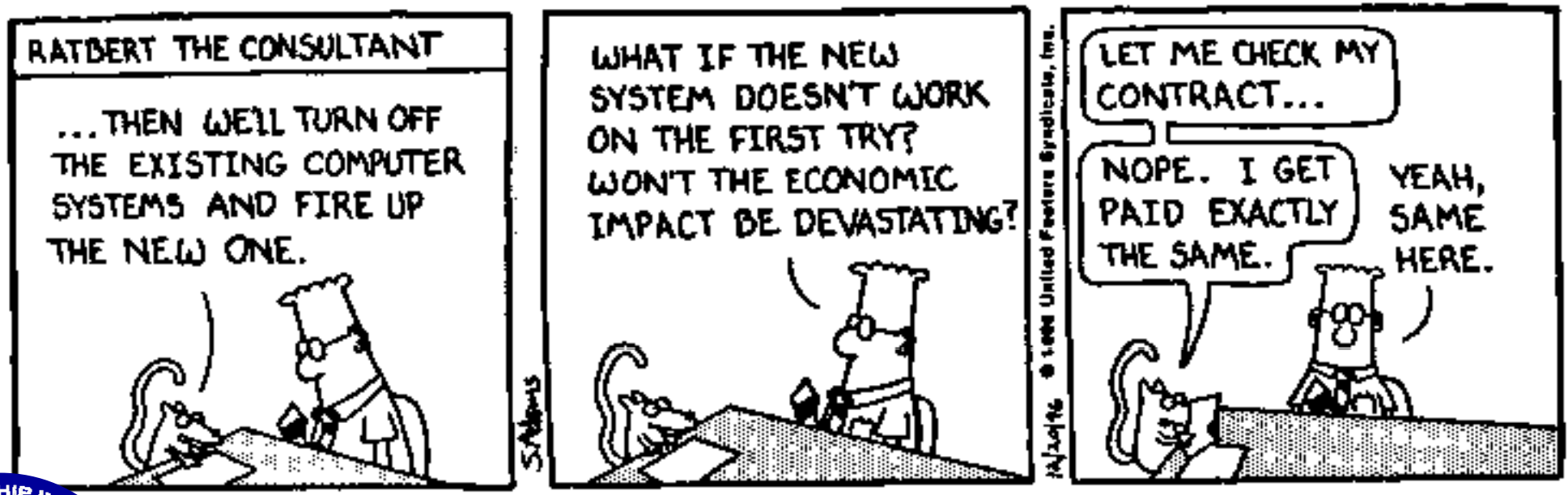
- This talk was “Food for thought”
- I would like to make this a Crosstalk article – and I would like your inputs, criticisms, suggestions, etc. I need your help.



For additional information…

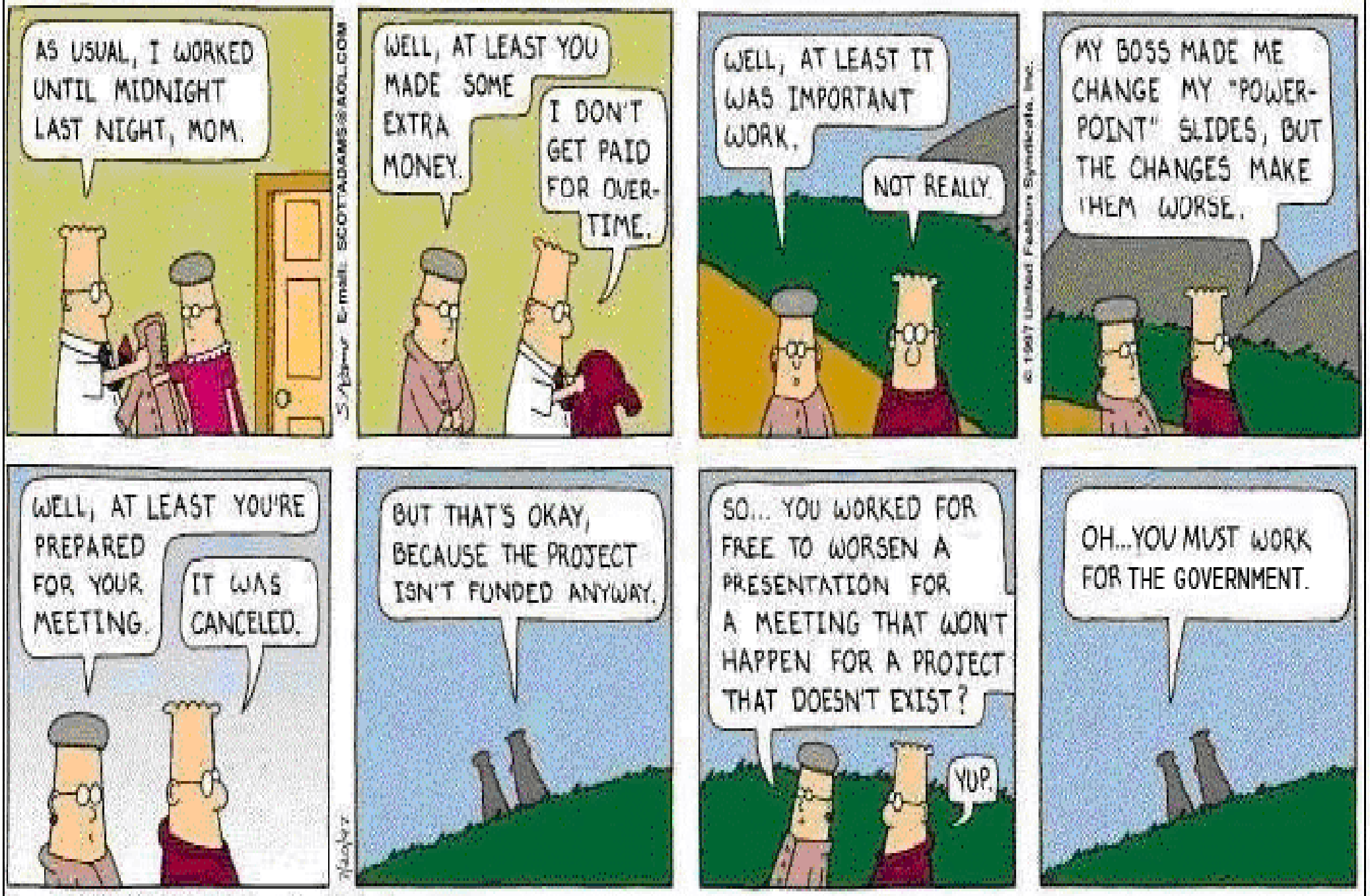
David A. Cook, Ph.D.

dcook@aegistg.com



The AEgis Technologies Group, Inc.
6565 Americas Parkway, Suite 825
Albuquerque, NM 87110
(505) 881-1003

DILBERT BY SCOTT ADAMS



Copyright © 1997 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited.