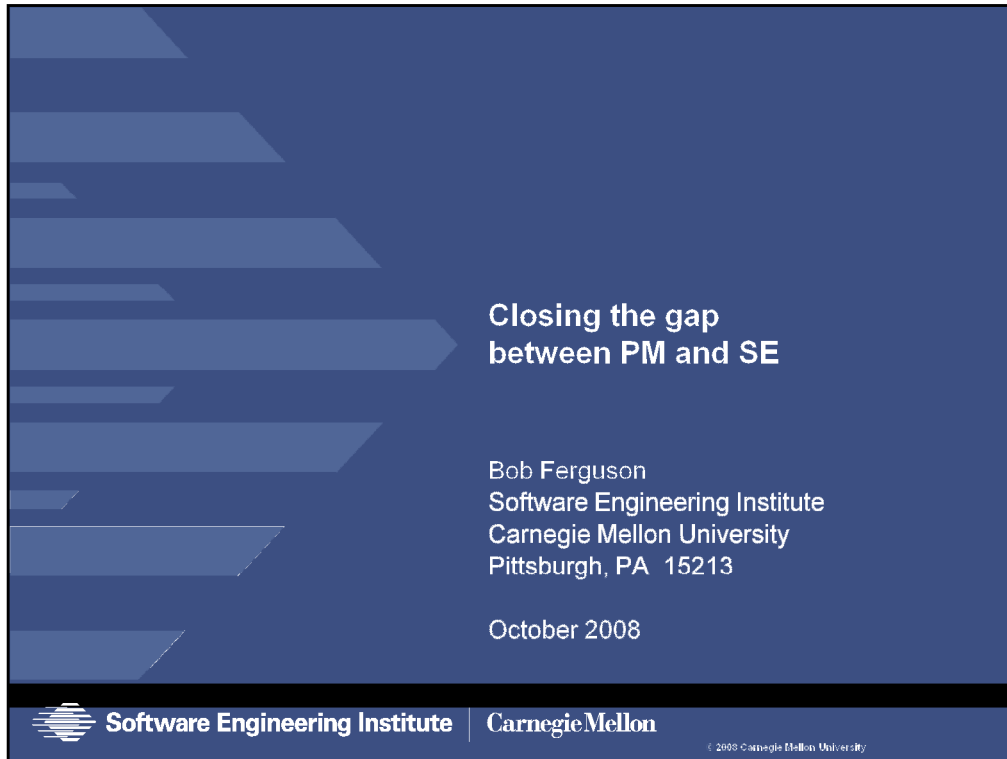



Presentation Title



**Closing the gap
between PM and SE**

Bob Ferguson
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

October 2008

 **Software Engineering Institute** | **CarnegieMellon**

© 2006 Carnegie Mellon University

Today's Topic Description

A conversation

Resolving complexity requires learning

Multiple value systems affect decisions

Innovation causes schedule unpredictability

A Conversation

Manager: How big is this project?

Developer: I don't know. This looks really hard.

Manager: Well we need to know how big it is so we can estimate the work.

Developer: I'll have to figure out how hard it is so I can tell you how long it will take.

*These two are talking about different things.
The developer believes that his estimate of size, will not recognize the uncertainty. He wants to know something about this complexity to adjust duration.*



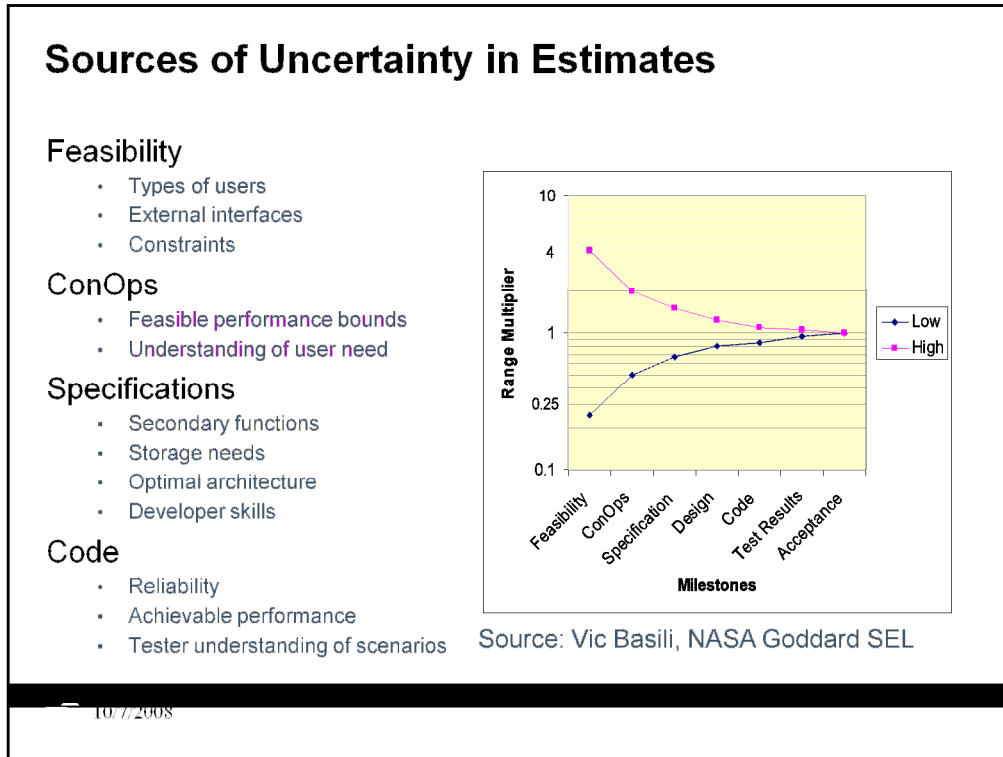
Project Manager's Concern

The PM is concerned with staffing and planning to meet the project's objectives.

The PM may not understand what the engineer means by complexity.

The PM does not know what questions to ask, nor has he thought sufficiently about engaging the SE in project planning.

How do we create a new "conversation"?



The nature of the uncertainty changes as we progress through the project. This means we have to be prepared to ask different questions depending on the deliverables under scrutiny and the actual tasking of the current work.

Uncertainty, Learning and Complexity

We often say “that’s really complicated” when we mean that we don’t know how to do a certain type of work within that specific domain.

- McCabe’s complexity is a means to describe how much has to be learned in order to provide satisfactory (error-free) maintenance.

Complexity often cannot be eliminated but there can be strategies (models) for resolving or reducing the impact of the complexity.

Questions to ask about “Learning”

What has to be learned to develop and deliver the product?

What can we see that provides evidence of this learning?

How can we best measure team performance?

Who can we influence if the right things are not happening?

Also, looking to the future --

- What should we record as “learning”?
- How can we advertise success?

**“Learning” should result in reduced uncertainty.
Perhaps that is measurable?**

10/7/2008



Complexity Types

Big

- Project is going to require partitioning into multiple teams and separable components

Invention and innovation

- Organization does not (yet) have the needed technical capability

Conflicting/Interacting goals

- E.g. power-weight tradeoff

Emergent behavior*

- External systems (including users) present changing stimuli

I don't know if this is an exhaustive classification of types of complexity. These categories seem to be useful to someone in a project management role. The effect of each type described calls for different management response.

We do not have time to talk about "Emergent Behavior" this time. However, I can tell you that discussion is about determining what happens when your system must respond to a new stimulus. You do not want the system to do anything bad or to fail, but often you must accept that new stimuli will arrive. How would you instrument your system to recognize that you had received a new stimulus or that your system was exhibiting a new type of response?

Big: Partition the work

Rationale:

- System or component encompasses too much functional responsibility for a single person or a single team to accomplish in a reasonable period of time.

What problems does our chosen partition introduce?

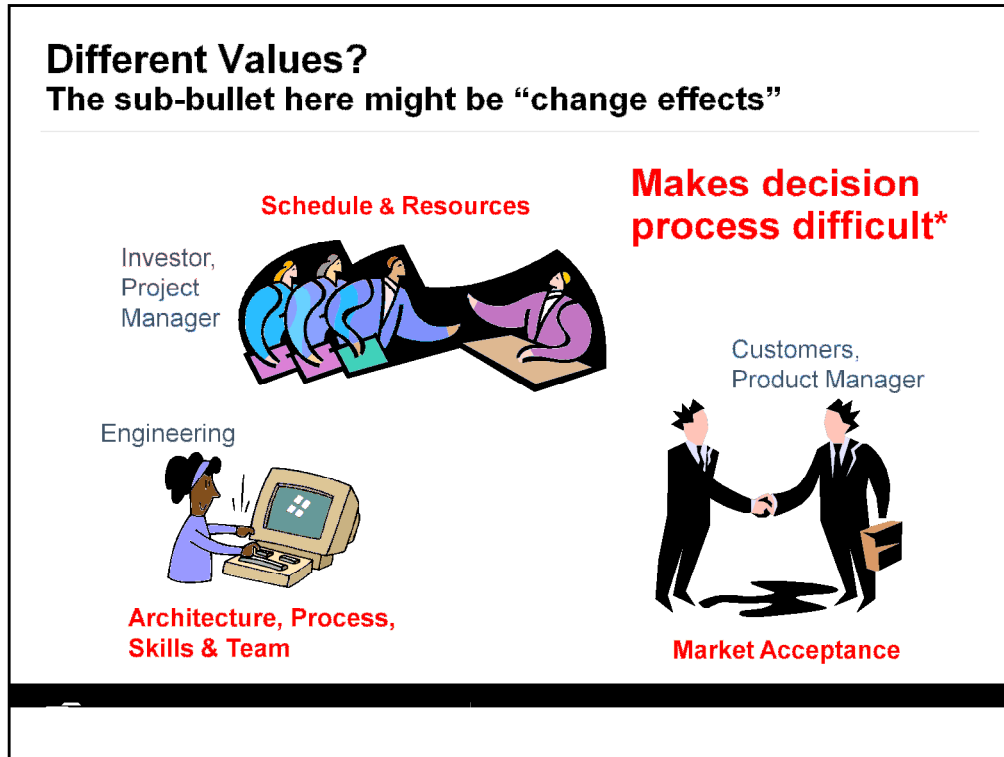
- Coupling creates component “brittleness”
- Interface definition is hard to communicate
- There may be design responsibility questions
- Sequencing the work is more difficult and the schedule is more fragile

Brittle means that a change to a component may break other components.

Interfaces will require IPT structures. The IPT structure generally should be built along product structure rather than organizational structure.

Design responsibility questions arise when two different teams have a debate about who should do what work.

Sequencing (schedule) is more difficult because critical path moves back and forth between teams.



These different value systems are often in conflict with one another. Problems occur when one individual has to be responsible for more than one viewpoint – the resulting behavior is either schizophrenic or ignoring one of the value systems.

Engineering process includes such things as design rules governing how we make design decisions.

This diagram also reflects the problem of change management. The effects of each change ripple to the other value systems. Somehow, the project must bring all three (or more) value systems to the table. This can only be accomplished effectively if each value system is independently represented.

Measures of Big

Component functional responsibility (raw size)

- Calls + I/O + Data-structure-manipulation
(Wayne Zage at www.serc.net)

Coupling

- $(\text{inflows} * \text{outflows}) + (\text{fan-in} * \text{fan-out})$

#Capabilities needed (too many for the team?)

- #Design-skills
- #Organizational functions
- #Process steps (e.g. pharmaceutical industry)
- #Quality checkpoints

It is impossible to create a complete list of size measures. The primary purpose of size is to help with estimation and planning. The second use of size is to normalize defect information and process.

Component functional responsibility affects team size, individual work assignments, IPT size, etc.

Coupling is an indirect (second order) measure of big. We do know, however, that modules with the highest coupling correlate to modules with lowest reliability. Too much coupling is bad for both the developers and the testers. Coupling is a measure applied to a single component.

#Capabilities (skills and processes) is another measure that suggests the need for an IPT structure (team of teams). Mixtures of hardware and software, development and validation, etc. all drive the need for the IPT.



Innovation means “Fix the Process”

Rationale:

- The organization does not yet have the capability (knowledge and skills) to do the work or to utilize this new technology; therefore the people and organization have something to learn.

What problems arise?

- Late technology capability will drive incorrect assumptions.
- Innovation is initiated by a single person or small team.
- Innovators are often not good communicators.

The traditional model of technology as “Technology Readiness Levels” only looks at whether the science exists. It does not address the organization’s ability to use the technology or the customer’s readiness either.

This error in understanding has often generated mistaken assumptions about the nature of technology problems and how they affect implementation. We suggest a measure of “technology adoption” or transition that includes monitoring the creation of design rules, testing procedures, user documentation, logistics, etc.



Innovation: Two Steps for invention and transfer

Ways to acquire the basic knowledge

- Purchase
- Develop
- Hire

Develop the skills for use – new process and product rules

- Design (“Design Rules” by Baldwin and Clark)
- Verification and validation
- Manufacture
- Distribution
- Marketing and customer support

Measures: how many processes, design rules, ...?

Design rules represent how the engineers are supposed to make decisions using the new technology.

Verification and validation should not be new, but validation is often neglected. You simply cannot deliver a successful product based solely on the requirements and specifications.

You can find some interesting reading about how organizations manage development of new skills. It's not really about training – it's about learning. Some of the reading will fall under the heading of IPPD – Integrated Product and Process Development.

“Conflicting Goals” require experiments

Examples

- Increasing power requires additional weight affects product price and margin.
- More security affects usability and performance

Experimental methods

- Prototypes
- Simulation
- Thought exercise
- Formal methods (e.g. rate monotonic analysis)

Executing an experiment is not particularly unpredictable even though the outcome is unpredictable. Therefore the experiment needs to be designed to help with making a choice.

The number of experiments can be unpredictable but there have been some studies that suggest using the Design Structure Matrix method to predict how many experiments might be needed.

The biggest problem occurs when no one realizes that the conflicting goals scenario virtually requires us to perform some number of experiments. The alternatives for finding the sweet spot are all more complicated and more time consuming.



Strategies for project managers and SE's

Identifying the complexity factors

Project planning methods

Multiple dimensions for tracking

Identifying Complexity and Uncertainty

Estimate Functional Responsibility.

- FP, #requirements, use cases

Estimate innovation.

- New technology, additional skills

Identify distinct value systems.

- Internal and external sponsors,
- Distinct users,
- Internal functional leaders

Every estimate needs a range and usually a constraint for an escalated decision.

We must estimate functional responsibility at the component level at some point during the planning process because it drives the partitioning of product into team- and individual-sized chunks. Component size is part of the detailed design planning, though it can be estimated reasonably at the architecture and requirements level.

Estimating innovation means understanding how many technologies will be adopted and estimating how many people, deliverables and processes will utilize the new technology. Each new technology will affect resources, schedule and product development processes. Does your change management allow you to change the process?

Concerns of “Big”: Size the design responsibility

Everyone uses principally one of three strategies:

By organizational function (groups of skills)

- Is easy, but may make too large a team

By function and feature

- Works ok if the architecture and domain are familiar
- May cause design problems because of responsibility conflicts

By product architecture

- May create resource contention and political problems
- This one best represents the “project-ized organization.”

This slide is presented because assigning too much responsibility to any individual or team will increase project risk.

Besides team-to-component size considerations, failure to align responsibility properly can result in other dysfunction.



PM Strategies for Innovation

Avoid multitasking the inventor

Give the inventor an assistant for technology transfer

Never let invention occupy the critical path

- Corollary: never let invention be deferred.
- If the invention is on the critical path, be careful of starting the related design activities and acquiring the resources before doing the needed transition work such as design rules.

It is hard to emphasize how severe a problem multitasking can be. Research literature suggests that people can handle two conceptual streams but not 3 or more. Task switching becomes very expensive.

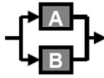

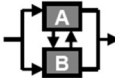
Since lead time for invention is somewhat unpredictable, adding multitasking increases the unpredictability. In other words, you just added to the project risk.



Analyze Modularity Effects

Design Structure Matrix (DSM)

- DSM has proved to be a successful approach to partitioning and analyzing very large systems. www.dsmweb.org

Three Configurations that Characterize a System			
Relationship	Parallel	Sequential	Coupled
Graph Representation			

Three Configurations that Characterize a System																														
Relationship	Parallel	Sequential	Coupled																											
DSM Representation	<table border="1"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td></td></tr> <tr><td>B</td><td></td><td>■</td></tr> </table>		A	B	A	■		B		■	<table border="1"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td></td></tr> <tr><td>B</td><td>X</td><td>■</td></tr> </table>		A	B	A	■		B	X	■	<table border="1"> <tr><td></td><td>A</td><td>B</td></tr> <tr><td>A</td><td>■</td><td>X</td></tr> <tr><td>B</td><td>X</td><td>■</td></tr> </table>		A	B	A	■	X	B	X	■
	A	B																												
A	■																													
B		■																												
	A	B																												
A	■																													
B	X	■																												
	A	B																												
A	■	X																												
B	X	■																												

We mentioned the importance of modularizing the project. Sometimes the modularity results in development “cycles” or coupling where the output of A affects the input of B and the output of B requires the input of A.

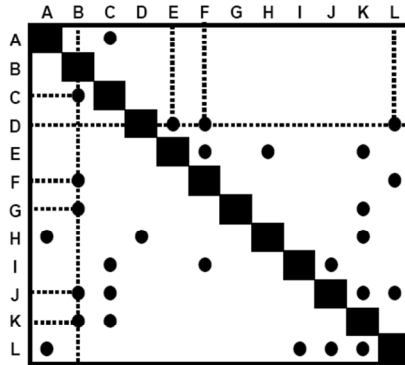
Not every cycle can be broken, however it is important to utilize as few cycles as possible and to keep them close together as possible.

DSM Types and Methods

DSM Data Types	Representation	Application	Analysis Method
Task-based	Task/Activity input/output relationships	Project scheduling, activity sequencing, cycle time reduction	Partitioning, Tearing, Banding, Simulation and Eigenvalue Analysis
Parameter-based	Parameter decision points and necessary precedents	Low level activity sequencing and process construction	Partitioning, Tearing, Banding, Simulation and Eigenvalue Analysis
Team-based	Multi-team interface characteristics	Organizational design, interface management, team integration	Clustering
Component-based	Multi-component relationships	System architecting, engineering and design	Clustering

There are several methods for using Design Structure Matrices. Each provides potential for action.

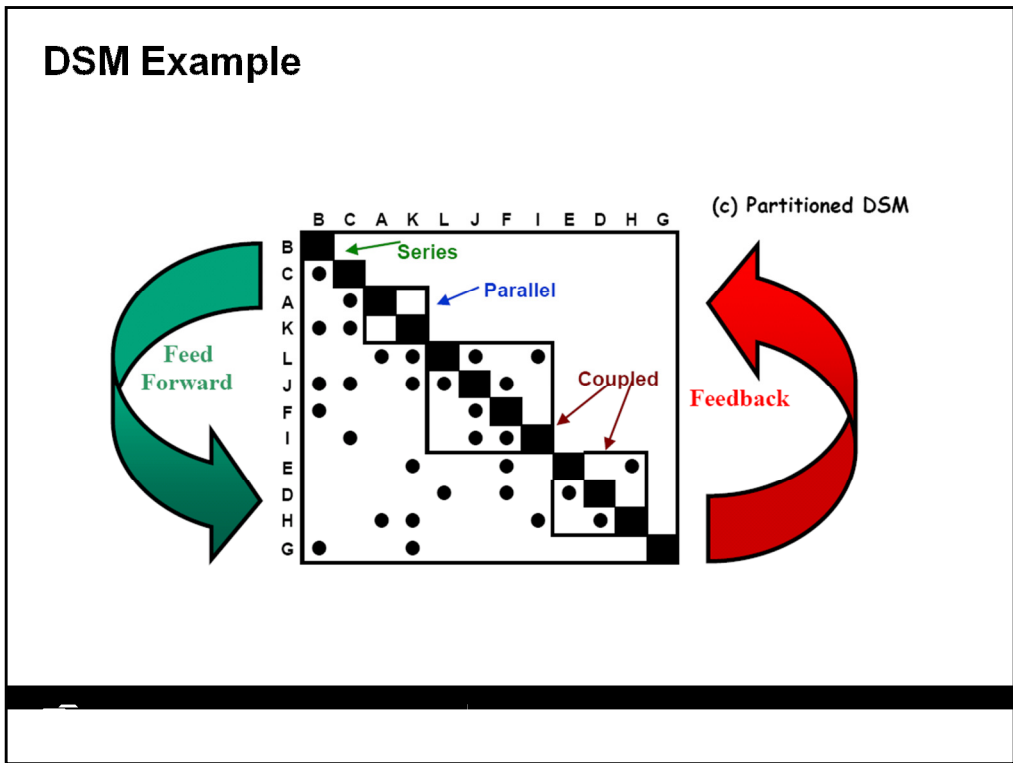
DSM Example



This matrix represents lots of interactions.

Structuring teams to modules is not clear.

By re-ordering the matrix we can achieve a better better modularity of both task and design.



Reordering the matrix yields a significantly better structure. We can use this approach to improve, component relationships, team relationships, and task relationships. The goal is to rearrange the order of rows and columns to reduce the effects of the cycles.

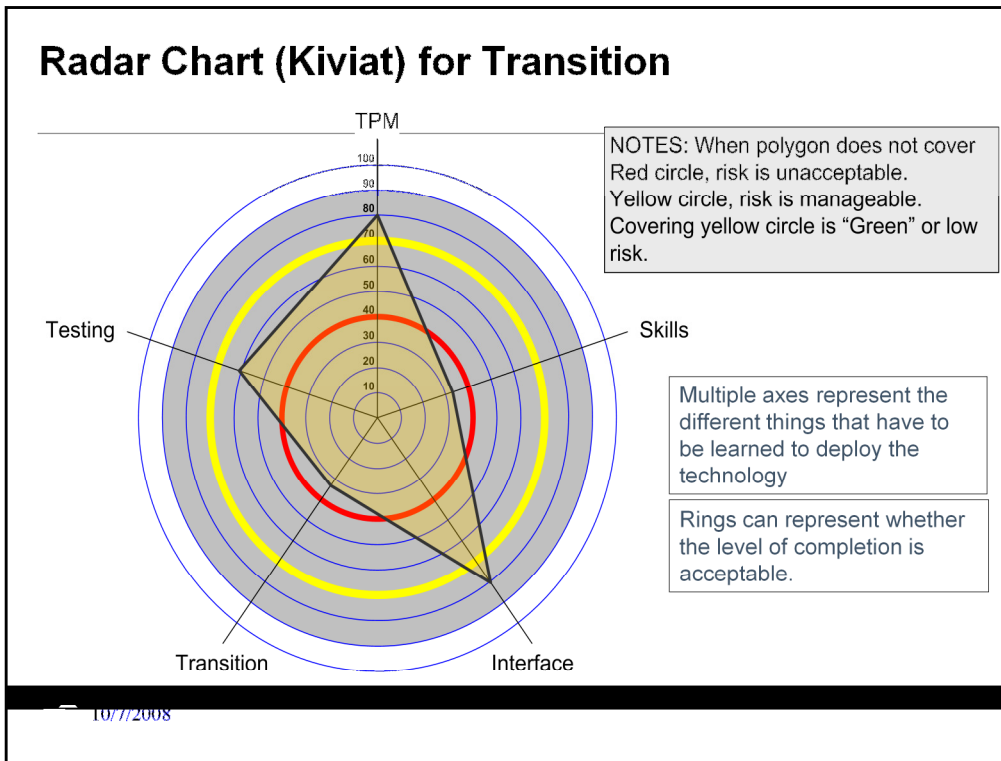
Monitor Invention

Technology Transition Continues to be a problem

- How mature is the technology? Can both technology and processes be acquired? Do we develop them?
- **Examine schedule for possible integration points.**
- What organizational capability is needed for use at this point of program?
- What organizational capacity for this work is needed (skills)?
- Have we identified technical performance measures?
- Create Kiviati diagram showing the various dimensions of technical and learning measures represented
- Review after each integration.
- Progress is evidenced when **all** dimensions are at the desired level.

10/7/2008





Technology transition is about teaching the organization. It is represented on each axis of organizational capability. At each major milestone, the risk can be evaluated. Scores on all axes are used to judge whether the technology is maturing as planned.

Product Technical Progress

How does a PM ascertain the reality of “technical progress”?

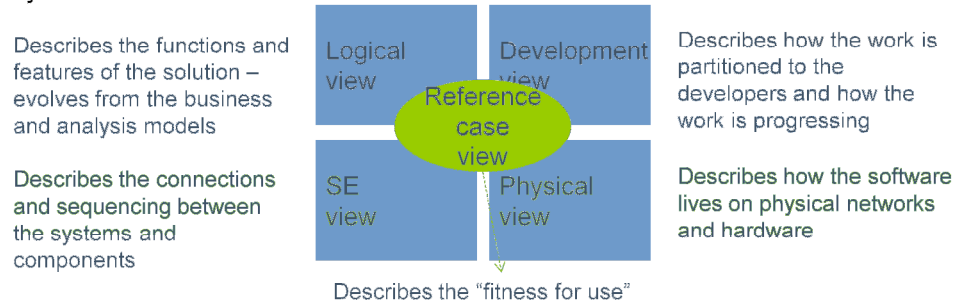
- **Rule 0: progress never counts unless there is a quality check.**
- Verification: the specification (feature function) is checked for completeness.
- Verification: the design meets explicit quality goals and is checked for consistency.
- Validation: the exposed design meets customer expectations
- Validation: the modeling technique in the design is suitable for use by the organization.



Assessing Technical Progress & Quality

We can improve our assessment of technical progress & quality by analyzing these **views** as they represent the knowledge captured in the system

Krutchen's 4+1 view model* is a useful representation for a software system



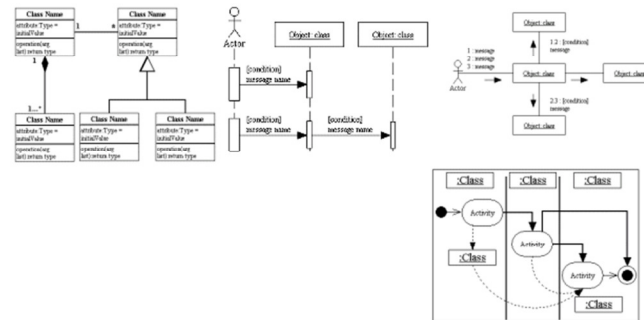
* Reference Krutchen, P "The 4+1 View Model of Architecture", IEEE Software, November 1995

Systems Engineering/Integrator View

SE View

Diagrams:

- class
- timing, sequencing
- object
- Component and connector approach
- System layering
- Design rules



How is component design responsibility established?

Performance objectives of components including throughput and latency.

Component and connector diagrams (wiring, object classes and public methods)

Are all requirements allocation to functional component?

Apply failure effects to prepare integration test cases and system test cases.

Coupling measure

I have a draft of a 12-page guide about applying these concepts. The guide consists of a customizable set of questions related to each of the architectural views. The questions are then applied to a set of design reference cases. (next slide)

Design Reference Cases (DRC)

Design Reference Cases are scenarios.

Use DRCs to probe the design solution. Choose them by identifying high-level capability needs and creating scenarios that stress the system.

- Mission threads
- Maintenance and support threads
- Performance analysis
- Safety analysis

Select specific DRCs by understanding the current technical milestone under review.

A number of DRCs are needed but only a few need to be checked during any specific design review.

Say for example, you are going to test 2 mission threads, a failure case and a performance case.

Failure case: "What happens if we lose a node of the network for 24 hours? How much capability remains?"



Example

A Critical Safety Case:

- Context: There riders on the train
- Stimulus: Equipment malfunction
- Desired response sequence
 - Vehicle stops
 - 3rd rail power shutdown
 - Announcement, manual latch accessible
 - Emergency team dispatched
 - Operator action
 -

“Show me how the design models implement this case”

- What components are affected, layers, etc.



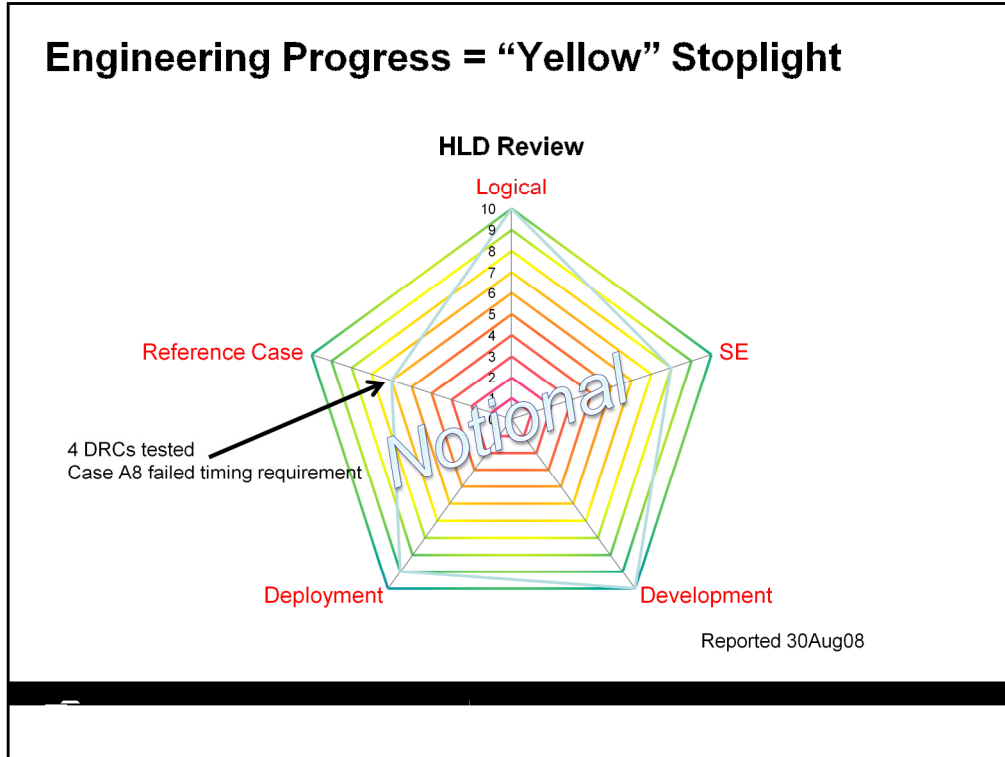
Scoring Procedure

View axes are scored by associating deliverable to PPQA, Verification and Validation score.

- Systems Engineering view questions: g-force, software timing, path through the layers of the system, ...
- Development questions: CM system audit, productivity, process quality, ...
- Deployment questions: prototype delivery, supply chain planning,
- Logical view: what the system looks like from the external business view, training, consistency of specifications, ...

Exception: Reference Case View is scored according to score of lowest rated DRC.





SE's strategies

Tradeoff Analysis

- Methods exist to predict the number of experiments required for convergence.

Innovation

- Lobby for dedicated resources based on schedule risk.
- Lobby for parallel development efforts based on schedule and product risk.



Summary

We described mechanisms for managing large projects.

- Modularity effects on project structure

- Monitoring technical progress

- Monitoring technology insertion

All these are intended to help you plan, monitor and manage responses to the uncertainties of new product development work.



Name
Program Name

Contact Information Slide

Bob Ferguson

Sr. Member of the Tech. Staff
SEMA
Telephone: +1 412-268-9750
Email: rwf@sei.cmu.edu

U.S. mail:

Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

World Wide Web:

www.sei.cmu.edu
www.sei.cmu.edu/contact.html

Customer Relations

Email: customer-relations@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800

