

6. Models of the Design Process

17 September 2008

Bob Glushko

Plan for ISSD Lecture #6

Meta-Methodology: Sequential, Iterative, Work Product, and Portfolio Approaches

"User-Centered Design at IBM Consulting"

"An Agile Customer-Centered Method"

"Enterprise Transforming Projects that Don't Kill the Enterprise"

Methodologies – Disciplines for Design

When we design something we follow – implicitly or explicitly – some steps or techniques for scoping, analysis, idea generation, and implementation

This DESIGN METHODOLOGY makes assumptions about which design questions can be separately answered, the priorities and dependencies, and who can best answer them

Methodologies can be formal, prescriptive, step-by-step, documented and auditable; they can be the opposite: informal, ad hoc, "seat of the pants" with no trace other than the design artifact itself; or they can be anywhere in between

The choice of methodology sometimes reflects a management and business philosophy, and sometimes reflects a personal or ideological one -- in either case, it can be highly contentious in a project

"Sequential" or "Waterfall" Methodologies

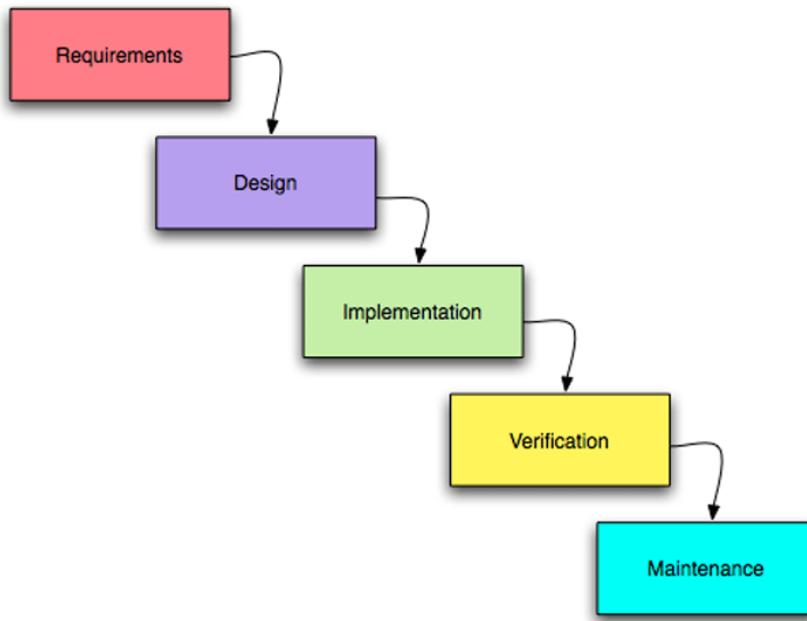
A methodology's process describes the work to be done and the order in which it is to be done

The simplest methodologies consist of a set of sequential activities in which the outputs of each step are the inputs to the next (the "waterfall")

This approach makes sense for small, well-defined problems in a stable context

Its goal is to "get everything right before progressing to next step" to minimize rework, which assumes complete and clear requirements that can be validated at each step

A Waterfall Methodology



Iterative Methodologies

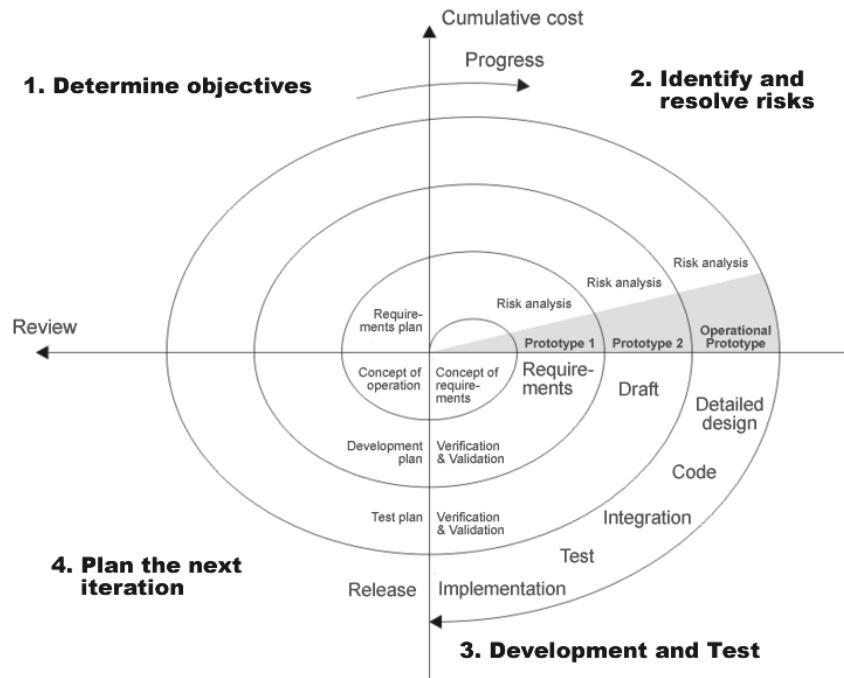
Other methodologies are more iterative or recursive, and assume that rework is inevitable and desirable

An underlying assumption is that requirements and the problem and solution contexts can only be understood over time, so it isn't worth investing too much effort to "pin them down" early in the design process

The goal of iterative methods is "get enough right at each step to know which step to take next"

Prototyping is essential; products emerge throughout the process and quality steadily improves

Spiral Methodology (Boehm, 1988)



"Agile" Methodologies

"Agile" or "extreme programming" methods for software development have become very popular methods in the last decade and are a specialized form of iterative methods used by small design teams

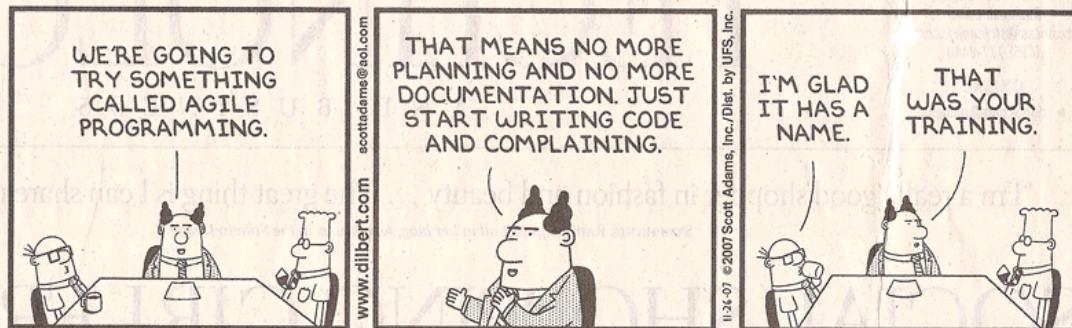
These methods deprecate up front investment in scoping and requirements specification, and rely on very rapid coding and testing cycles to incrementally develop software

The ["Agile Manifesto"](#) (agilemanifesto.org) advocates:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Not Quite Agile Methodology

DILBERT Scott Adams



"The basic assumption by organizations adopting agile is that there is no hope for improving the requirements process so they must jump into coding prototypes and getting user feedback quickly"

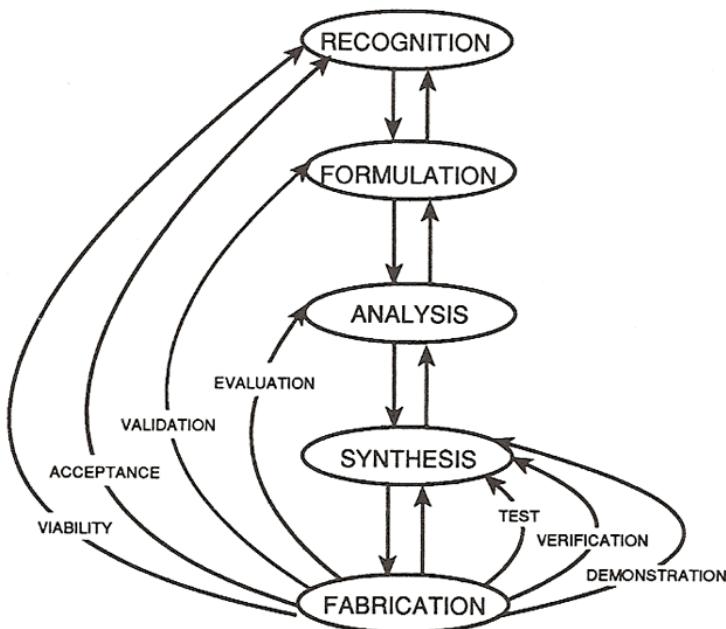
"This institutionalizes rework due to bad requirements and assumptions" (Tom King, Ravenflow)

Most Methodologies Are Hybrids

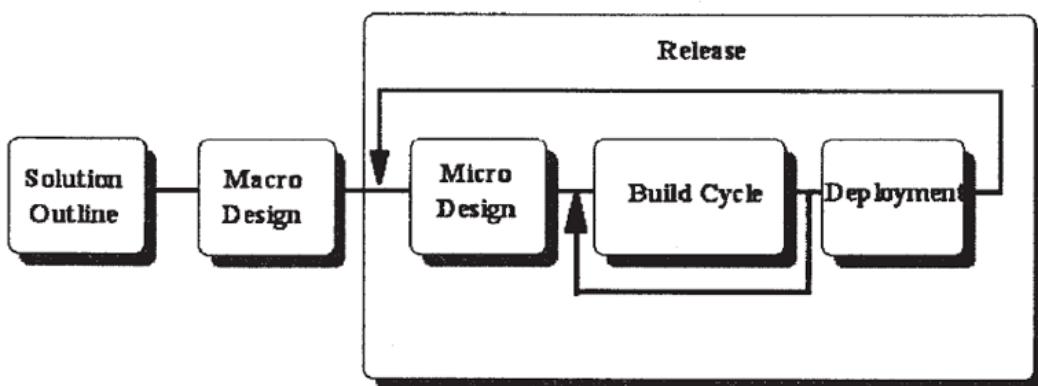
Sequential methodologies are often presented as a "straw man" to be rejected, but their appropriateness depends on granularity -- every methodology has sequential characteristics

Similarly, iterative methods are often presented as a radical departure from sequential methodologies, but every iterative methodology has some sequential characteristics when viewed from a "coarsed-grained" perspective

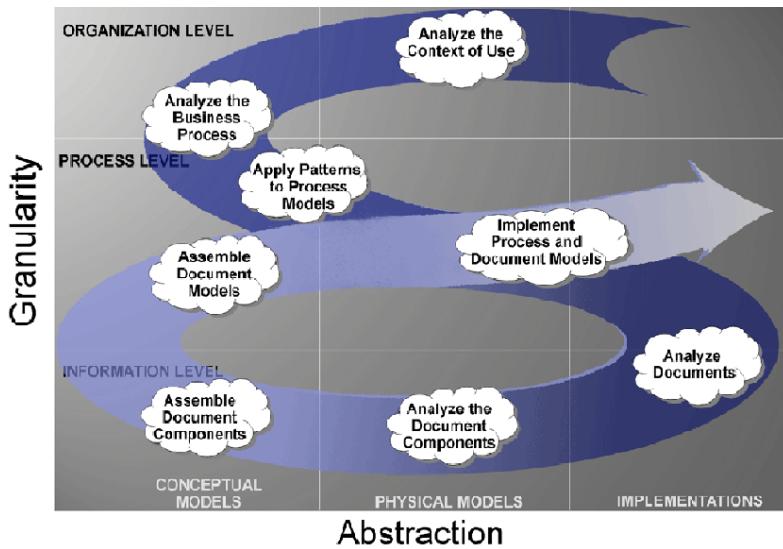
"Design for Success" Methodology



"IBM Global Services" Methodology



"Document Engineering" Methodology



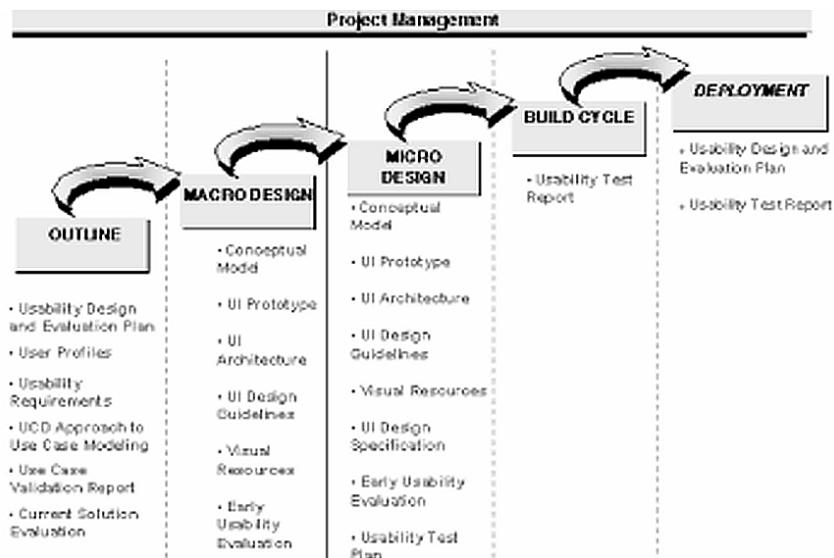
Artifact- or Work Product-Centered Methodologies

Artifact or work product-centered design methods specify activities, but put more emphasis on the artifacts or work products that result from them

These methodologies are less prescriptive about how the work is done, but might be very prescriptive about how it is documented

This approach is appropriate in complex projects with a distributed design team, and especially when designing in a "business ecosystem" of "service interfaces"

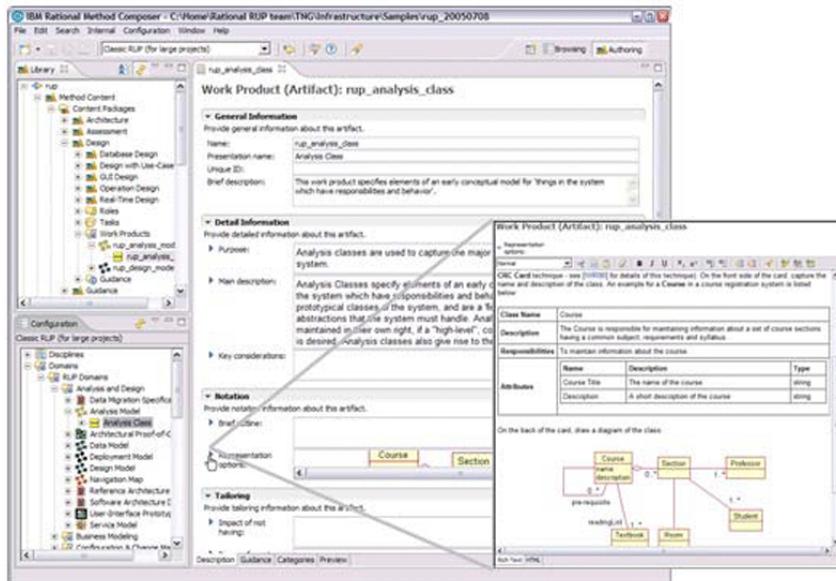
IBM GS UCD Work Products



Some "Document Engineering" Artifacts

Phase	Artifact
Analyzing the Context	UML use case diagrams
Analyzing/Designing Business Processes	Business Domain View Worksheet
	UML use case diagrams
Analyzing/Designing Business Collaborations	Business Process Area Worksheet
	UML activity diagrams
Analyzing/Designing Business Transactions	Business Transaction View Worksheet
	UML sequence diagrams
Applying Patterns to Business Processes	Document checklist

IBM "Rational Method Composer" WP Editor



"Portfolio of Techniques" Approaches

Unless the design projects taken on by an organization or team are always for the same context, with similar scope and requirements, they will not follow the same design methodology on every project

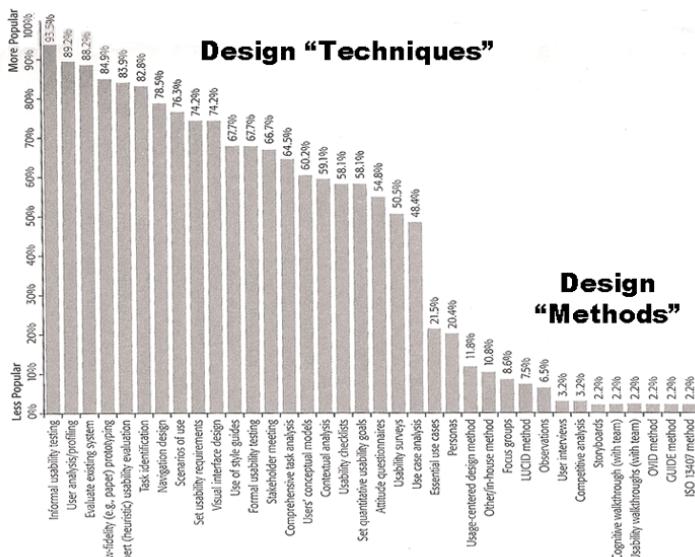
There will be always be a need to adapt the methodology in some way, emphasizing some activities more or less than usual because of schedule, resource, or stakeholder considerations

After all, the goal should never be to follow a methodology -- the goal is to understand a design problem so that a solution can be developed and deployed

So in practice, the "methodology" employed in any given project is likely to be a set of design techniques selected and adapted for it

IBM consulting calls this the "Engagement Model"

Design "Techniques" vs "Methods"



"User-Centered Design at IBM Consulting"

Why did IBM need a UCD methodology?

How was the methodology developed?

How is it a "work-product" methodology?

How is the methodology used in engagements?

The Need for an IBM UCD Methodology

The IBM consulting approach uses "matrixed" project teams assembled from "best available" people wherever they are

But UCD consultants varied in their training, techniques, process, and deliverables

Difficult to write proposals, manage customer expectations, and perform effectively with such heterogeneity

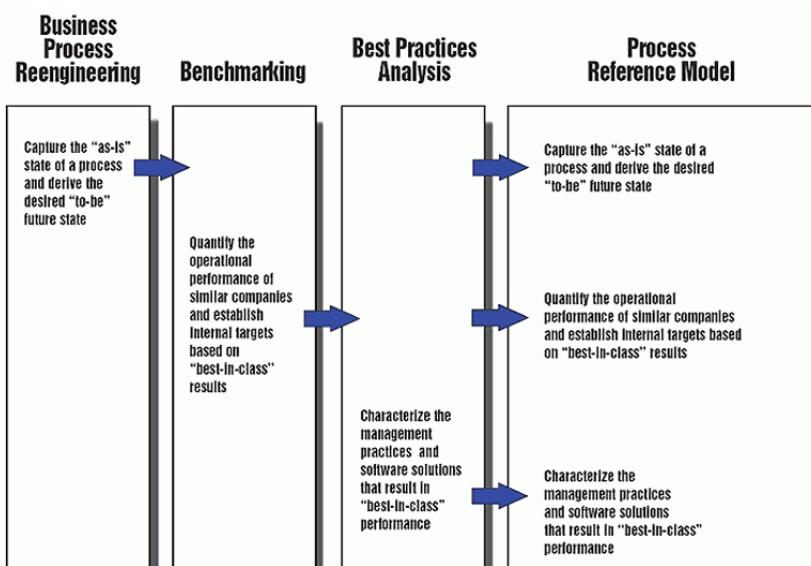
Developing the IBM UCD Methodology

"A benchmark of known existing methodologies was performed, and best practices identified"

"Benchmarking" the practices in some industry is often done by a consulting firm or industry association because companies are not likely to tell competitors about their methods and capabilities.

For a company as big as IBM, where there are lots of more or less competing consultants or consultant practice groups, internal benchmarking serves much the same purpose as industry benchmarking

Benchmarking, Best Practices, and Reference Models



A "Work Product" and Reuse-Centered Methodology

The customer complaint of "too many talkers and not enough do-ers" suggested an artifact- or work product-centered methodology that emphasizes the deliverables in an engagement, not the technique or process of delivery

PUT ANOTHER WAY - get paid for results, not for effort

The goal that "no work product should ever be created from scratch" implies that all work products should follow detailed content specifications and that previous ones should be managed in a "knowledge base" to facilitate their reuse

This is hard for technical and "organization culture" reasons

The UCD Work Products

The complete IBM GS methodology has 100 work products, out of which 15 relate to UCD activities

- **UI Planning**

- Usability Design and Evaluation Plan
- User Profiles
- Usability Requirements
- UCD Approach to Use Case Modeling
- Use Case Validation Report

- **UI Design**

- UI Conceptual Model
- UI Prototype
- UI Architecture
- UI Design Guidelines
- Visual Resources
- UI Design Specification

- **Usability Evaluation**

- Current Solution Evaluation
- Early Usability Evaluation
- Usability Test Plan
- Usability Test Report

The Engagement Model

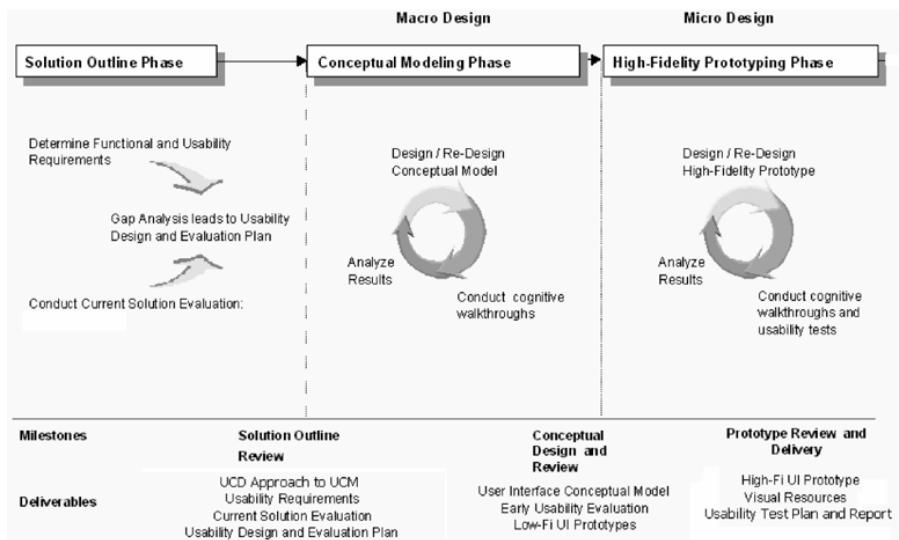
All 15 of the UCD WPs aren't necessary or justified in every design context

For example, an incremental functional upgrade to a legacy system with a known and small set of users needs relatively few of them

There will be other recurring design contexts that will need different subset configurations of the WPs

The "Rapid Prototyping Engagement" is one of the most common

IBM "Rapid Prototyping Engagement"



Compare to UCD "Top Ten" Techniques

1. Informal usability testing (93.5%)
2. User analysis/profiling (89.2%)
3. Evaluate existing system (88.2%)
4. Low-fidelity (e.g., paper) prototyping (84.9%)
5. Expert (heuristic) usability evaluation (83.9%)
6. Task identification (82.8%)
7. Navigation design (78.5%)
8. Scenarios of use (76.3%)
9. Set usability requirements (74.2%)
10. Visual interface design (74.2%)

from "Object Modeling and User Interface Design," Mark Van Harmelen (Ed.), 2001

"An Agile Customer-Centered Method"

Can the "customer-centered" but "upfront" methodology of "contextual design" fit into an "agile" methodology?

Classic (Non-agile) Contextual Design

Contextual inquiry -- workplace "following around" and interviews with customers

Interpretation sessions and work modeling; Consolidation and affinity building -- synthesize the inquiries with different users to create an as-is work model and composite of the target population

Visioning -- high level design of to-be system or work

Storyboarding; User Environment Design

Paper Prototyping

Rapid Contextual Design

Instead of starting by putting a customer or customer surrogate into the software team, carry out a contextual inquiry and bring the completed artifacts to the team

Organize storyboards and User Environment into prioritized "coherent sets of function" that are compatible with agile development

Design UIs for each of these, and test with users on paper and in interviews

Agile developers implement the (revised) UIs and functional packages using their usual methods

"Enterprise Transforming Projects that Don't Kill the Enterprise"

A paper full of wisdom from a very experienced software developer (and software development theorist)

What kinds of projects are "enterprise transforming" or potentially so?

How does the "agile manifesto" work when you take on big enterprise projects with legacy and technology considerations?

The Lessons Learned [1]

Deliver Frequently

- Actual benefits from deployed software NOW always beat hoped-for future ones
- Project sponsors need justification for continued investment
- Early deliveries can focus on exploring new technology capabilities to be exploited in later ones

Expect Surprises

- Changes to requirements should be seen as inevitable and desirable, reflecting better understanding, not as negative impacts on a plan
- Planning should be a continuous process

The Lessons Learned [2]

Get High Level Executive Support -- if you can't, don't take on the project

Treat Business and SW Development as Partners

- Fixed price contracts are desirable... when requirements can be well understood
- But if requirements change, parties will argue about what they agreed to

Choose Technology for the Future

- You need to predict which technology will be the right one when the project is deployed
- You can hedge your bets by thinking "integration" rather than "new platform"

The Lessons Learned [3]

People are the Critical Success Factor

- A key role of management is to ensure that the right people -- the people with the knowledge to make them -- make the right decisions
- The other key role is to facilitate communication between all the stakeholders

Keep Learning

- People change their requirements when they see prototypes and early versions

Readings for 22 September

Robert J. Glushko & Tim McGrath, Document Engineering, Chapter 8,
"Analyzing the context of use," 2005.

Tom King, "Rapid requirements definition," <http://ravenflow.com/>