Strategies for Rapid Development in Internet Time

By: William A. Cunningham

December 5, 2000

1. Introduction

1.1. Background

Roughly two-thirds of all projects significantly overrun their cost and time estimates. According to surveys, the average large software project misses its deadline by at least 25 to over 50 percent. The size of slippage increases with the size of the project. There is a perceived slow-development problem that is not limited to the Internet community.

However, some organizations are developing systems rapidly much more quickly than others. The difference in productivity in the same industry can be as high as 10-1!

1.2 Purpose

The purpose of this paper is to introduce some of the concepts of Rapid Development to help you get your software development projects under control and to apply some of the same proven concepts that have helped other organizations dramatically increase productivity.

1.2. Scope

The scope of this paper is to:

- define what Rapid Development is (and what it is not),
- define and discuss the rules that make rapid development effective,
- discuss which flavor of rapid development works best for your organization,
- define four major lifecycle methods that have an impact on rapid development, and
- discuss the role of Oracle Designer in rapid development

1.3. Audience

Everyone is affected by slow development of systems. Software developers, managers, clients and end users should find at least one topic in this paper useful. However, this paper is designed especially for:

Managers
Software development is not solely a technical issue. It is a business issue. Many of the slow software development issues could be helped with the efficient application of proper business and strategy planning procedures to IT solutions

DBAs and Data Modelers
Data management means more than fast transaction speeds. It means having data structures that answer the business needs of today while being flexible enough to handle the eventual needs of tomorrow. DBAs need to stand back and ask serious questions and have the flexibility built into the data structures to handle the ‘we never ever do that’ situation.

2. What Is Rapid Development?

Today, we hear that 'it only takes 15 seconds on the web to lose a customer', that 'we have to get it right the first time' and that the systems have to be developed 'in Internet time'. This can be a real challenge for the software developers of today. Code has to be written faster, better; we turn to 'Rapid Development' to meet those challenges.

But what is Rapid Development anyway?

Rapid Development can mean different things to different people. To some, it is a particular software, hardware tool or method that produces an application. Many in the dot.com ranks see Rapid Development as 22 hour days and a cot under the workstation. To data architects and information engineers it is CASE methodologies, JAD sessions, and tight time frames. To a manager with ever tightening schedules, it is the latest methodology gleamed from the last issue of ComputerWorld or an InFlight magazine.

For the purposes of this paper, **Rapid Development is the effective and efficient creation of application systems within a full-fledged strategy.** Or, simply put, it is developing something that works more quickly and efficiently than we normally do.

Rapid Development involves the use of effective practices and tools that are oriented specifically toward achieving your schedule. We tend to think of this as being obvious. But, if obvious, why do we tend to go with the 'latest' beta or 'hottest' software tool (a.k.a. the silver bullet) that may or may not support our goals. In our case, should we select the just released, new version of Oracle Designer, Developer or DBMS or work with a stable configuration with known limitations?

Since development projects are based on schedules, rapid development must find a way to minimize the overall timeframe. Each organization has their own 'hot' buttons for development but these can be generalized into:

- Speed-oriented techniques – those that improve development speed, producing code faster
- Risk avoidance techniques – those that address technology and business risk, avoiding severe schedule overruns
- Visibility-oriented techniques – those that show progress, avoiding the appearance of slow development (a.k.a. the why are they not coding yet syndrome.)

> "Rapid product development is not a quick fix for getting one product – which is probably already late – to market faster. Instead, it is a strategic capability that must be built from the ground up." *Preston G. Smith and Donald G. Reinertsen, Developing Products in Half the Time*

3. Rules of Rapid Development

Rapid Development strategies are fundamentally quite simple. There are three basic rules:

1. Apply proven methodologies and development fundamentals
2. Identify and manage your technology and your business risks
3. Apply good project management schedule oriented techniques

These rules sound simple. However, they are not easy to follow. A successful rapid development project requires that ALL of the rules be in place. If any one is missing, the project schedule is probably in jeopardy.

Apply Proven Methodologies and Development Fundamentals

A methodology is body of practices, procedures, and rules used by those who work in a specific field or specialty. Some of the proven software development methodologies are, but not limited to,

- CASE*Method – by Richard Barker

- The Zachman Framework – by John A. Zachman
- Strategic Data-Planning Methodologies – by James Martin
- Rational Unified Process(RUP) – by Booch, Jacobson and Rumbaugh

Definition – CASE Computer Aided Systems Engineering is the combination of graphical, dictionary, generator, project management and other software tools to assist computer development staff to engineer and maintain high-quality sytems for their end-users, within the framework of a structured method. *Richard Barker – CASE*METHOD Entity Relationship Modeling*

A methodology is a disciplined way of approaching a problem. A methodology is **not** Oracle Designer nor is it Rational ROSE. These two products are toolkits or aids. Oracle Designer implements the CASE*Method developed by Richard Barker while Rational Rose implements the concepts of RUP. These products help you engineer and document your designs according to the specific methodology; they do not do your design work for you.

Identify and manage your technology and your business risks

Before delving into technology and business risks, which type of risk management do you frequently engage:

- Crisis management – Fire fighting; address risks only after they have become problems
- Fix on failure – Detect and react to risks promptly, but only after they have happened
- Risk mitigation – Plan ahead of time to provide resources to cover risks if they occur, but do nothing to eliminate them in the first place
- Prevention – Implement and execute a plan as part of the software project to identify risks and prevent them from becoming problems
- Elimination of root cause – Identify and eliminate factors that make it possible for risks to exist at all.

There are a number of potential risks that can affect your software development project. However, the most common schedule risks are:

- Feature Creep
- Shortchanged quality
- Overly optimistic schedules
- Inadequate design
- Silver-bullet syndrome
- Research-oriented development
- Mismatched technical skills
- Communication issues between developers and clients

Case Study

A large multinational corporation decided that they wanted to be a ‘key player’ in the travel portal business within six months. The CIO selected a software vendor who had created a software product that was new and untested in the commercial marketplace and was not Oracle DBMS (the target platform) based. A superficial ‘fit-gap’ analysis was performed showing some significant gaps. However, the CIO sold the project to top management and the delivery date was already agreed upon before the technical details were known. Required and identified ‘gap’ functionality was missing when the first software release was delivered. The software was not frozen until the end of the project. Unit testing had not begun by the scheduled time for full integration tests. The Quality Assurance effort became one of ‘heroic effort’ rather than systematic planning. The development teams worked harder and harder but bugs were discovered faster than they could be fixed. New software was released on a daily basis and some fixed bugs resurfaced. The deadline came and went as did three other deadlines before the software was stable enough to be released.

What went wrong?

Rule 1: **Apply proven methodologies and development fundamentals**

The proven methodologies and development fundamentals were not followed. Although the 'fit-gap' analysis was complete, the development team did not address the 'gap' requirements until too late in the software development. When the software development timeframe overran the test requirement phase, it was decided to eliminate most of the testing time to catch up.

- Inadequate design
- Shortchanged quality

4. Rule 2: **Identify and manage your technology and your business risks**

Business and technical risks were underestimated. The technology risk was severely underestimated that the software vendor had not developed this application for an Oracle port nor for an organization as large as this. The business risk was also underplayed of trying to become a major player in the web portal business within six months.

- Shortchanged quality
- Inadequate design
- Silver-bullet syndrome

Rule 3: **Apply good project management schedule oriented techniques**

The key problem was 'wishful thinking'. The top down direction of a six month deliverable as the keystone of this project sealed its fate. There were no other considerations other than meeting the deliverable date, whatever it took.

- Overly optimistic schedules

Sadly, this Case Study is not an isolated case. Many corporations have the same experiences and do not appear to learn from their mistakes. As a pilot, we refer to this tendency as "Take Off-itus". That is, once the decision is made to take-off or go, it is difficult, if not impossible to stop the forward motion and abort a system that you know will not fly right.

4. Which Rapid Development Strategy Works For You?

Rapid Development does not mean that critical stages in the project can be skipped. There is no 'one size fits all' development strategy and there are no shortcuts to the methodology side of successful rapid development. It is even more important in Internet time that we devote the proper resources to the following classic stages of the project:

Activity	**Small Project (2.5 K lines code)**	**Large Project (500k lines code)**
Strategy and Analysis	10%	30%
Detailed Design	20%	20%
Code/Debug	25%	10%
Unit Test	20%	5%
Integration	15%	20%
System Test	10%	15%

Classic Stages of a Project – *Steve McConnell, Code Complete*

However, which rapid development style fits you depends on your specific business, corporate culture and your current systems. To determine which rapid development strategy works for you, you need to know if your project has:

- A very strong project schedule constraint, such as having a regulatory deadline, having a system in place for the Christmas season or meeting a funding constraint as a startup dot.com
- A top management or user requirement for 'rapid development' that really translates into a desire for lower cost or less risk instead

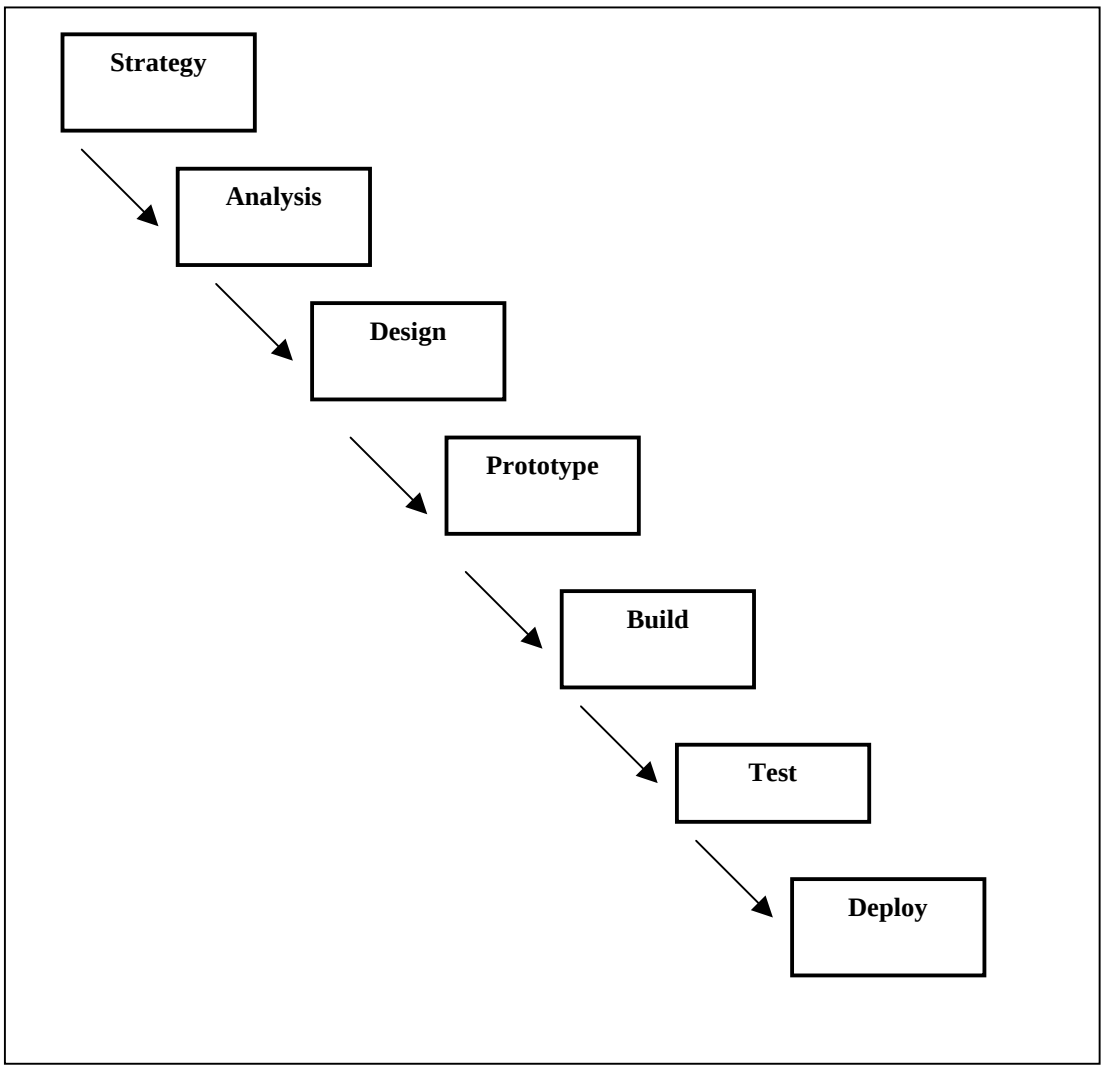
In either case does your project have any limitation or weakness that would prevent a rapid development success?

5. Rapid Development – A Lifecycle Approach

Over the years, a number of approaches have been used to define a master plan and to improve the project success ratio.

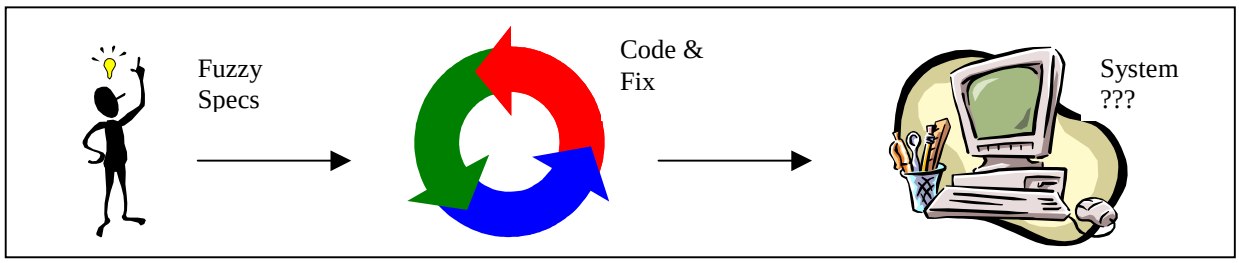
5.1 Pure Waterfall

The original lifecycle model has become the basis for other, often more effective, lifecycle models. Each phase of the project is completed and stable before the next phase starts. The waterfall model uses documentation as the deliverable product and the phases do not overlap.

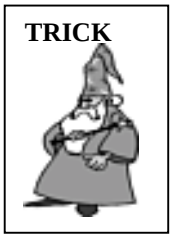


5.1 Code and Fix

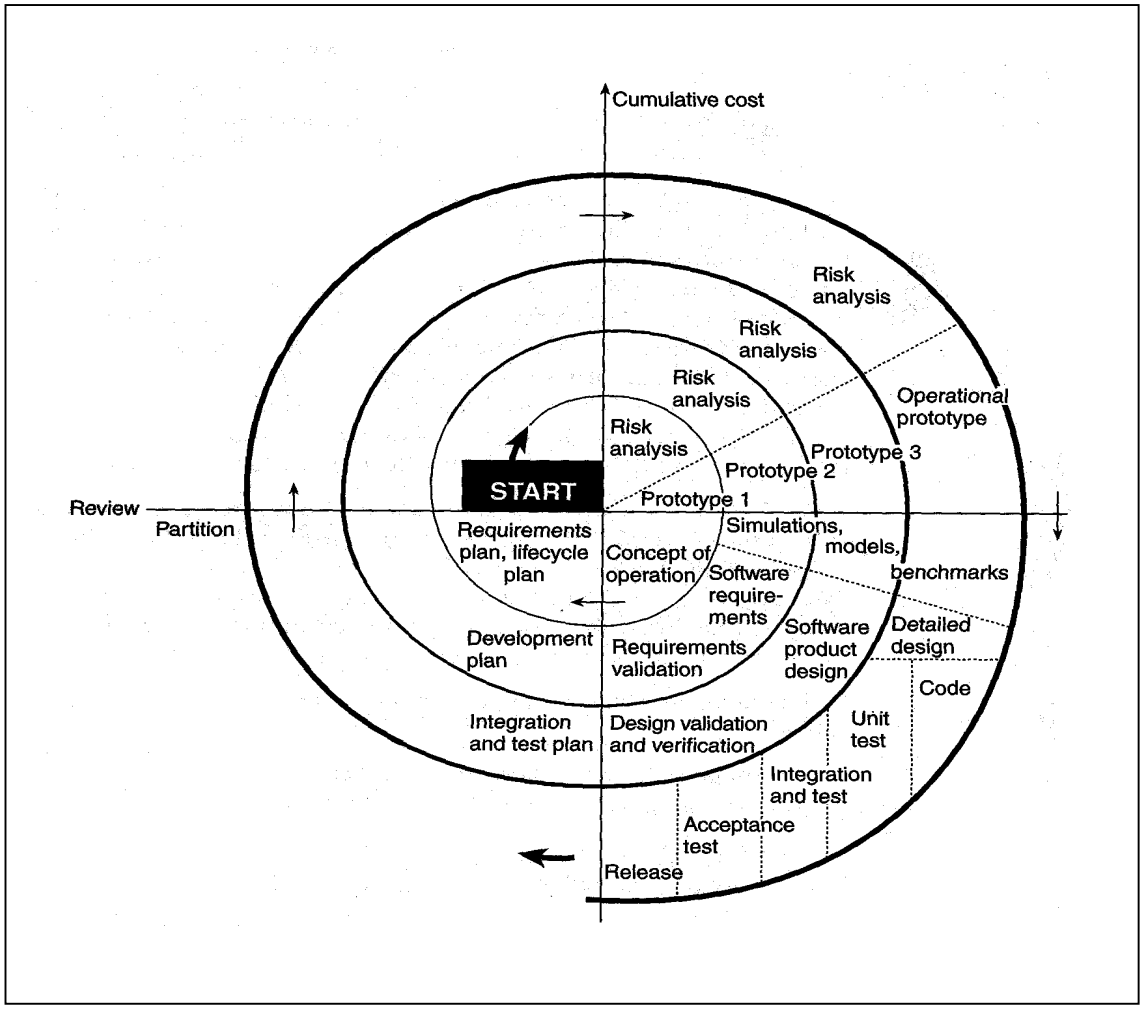
This is a common, but not rapid, development methodology. If you have not already selected another proven methodology, this is the one that you will probably be using. This is the classic mistake of 'I don't have time for strategy and analysis; I need to start coding right now to meet our deadline'. Unfortunately, this is a 'you can pay me now or you can pay me later' approach to the development project. As you might expect, this falls into the 'code like hell' category and any development that comes close to making its schedule is due to heroic efforts and many 22 hour nights.



5.2 Spiral Development



The Spiral Development is the exact opposite of the Code and Fix model. The spiral model breaks the project into manageable sub-projects, each of which addresses topics such as poorly understood requirements, poorly understood architecture, potential performance problems. When all of the major sub-projects are complete, the model is complete as if it were a classic waterfall model.



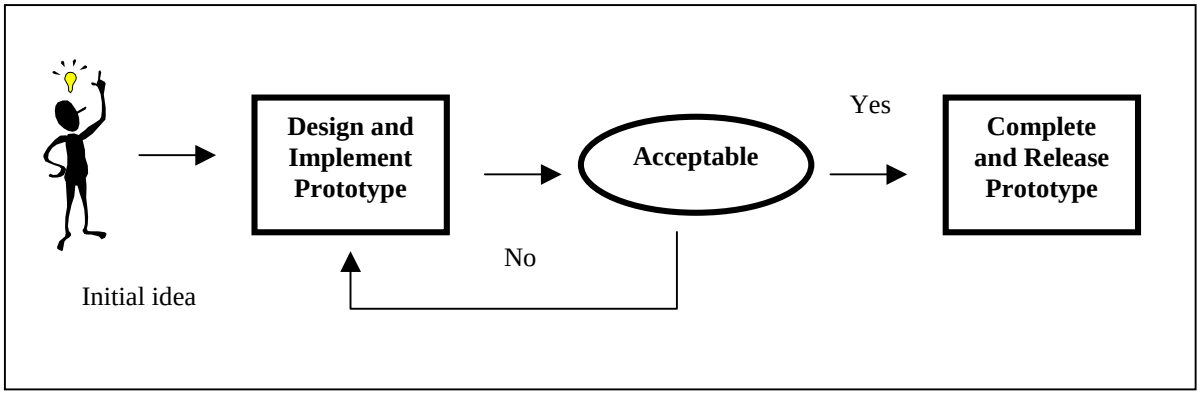
TRICK

5.3 Timebox Prototyping

Timebox prototyping is a lifecycle model where you define the specifications as you are coding the system. The model begins with a 'fuzzy' specification of the client requirements and a prototype of what you think the system is supposed to perform within a specified timeframe. This requires that the client provide feedback on the work in progress of the prototype in a timely manner and that any requirements that can not be met within the agreed timeframe are held for a future cycle. This is an iterative procedure in which the prototype is refined until the client finds it acceptable for production.

This is a very useful lifecycle method when the client is reluctant to commit to a set of specifications, the requirements are changing rapidly or where you or your client do not understand the application area well. This technique produces steady, visible signs of progress which are important when there is a strong demand for development speed.

The downside it that you may not know with any certainty how long it will take to get an acceptable product. This uncertainty can be mitigated with the client having a number of opportunities to review the prototype in detail and to determine when or why the development should stop. The project could stop because the prototype meets the client specification or because the client finds that the proposed solution may not work.



5.4 Summary of Lifecycle Model Strengths and Weaknesses

As mentioned earlier, there is no 'one size fits all' rapid development methodology. However, the table below outlines some of the key strengths and weaknesses of each lifecycle methodology. Which one to use is based on your company, your corporate culture and your current applications.

Lifecycle Model/ Capability	Pure Waterfall	Code and Fix	Spiral	Timebox
Works with poorly defined requirements	Poor	Poor	Excellent	Excellent
Works with poorly understood architecture	Poor	Poor	Excellent	Poor to Fair
Produces highly reliable system	Excellent	Poor	Excellent	Fair
Produces systems with large growth envelope	Excellent	Poor to Fair	Excellent	Excellent
Manages risk	Poor	Poor	Excellent	Fair
Can be constrained to a predefined schedule	Fair	Poor	Fair	Poor
Has low overhead	Poor	Excellent	Fair	Fair
Allows for midcourse corrections	Poor	?	Fair	Excellent

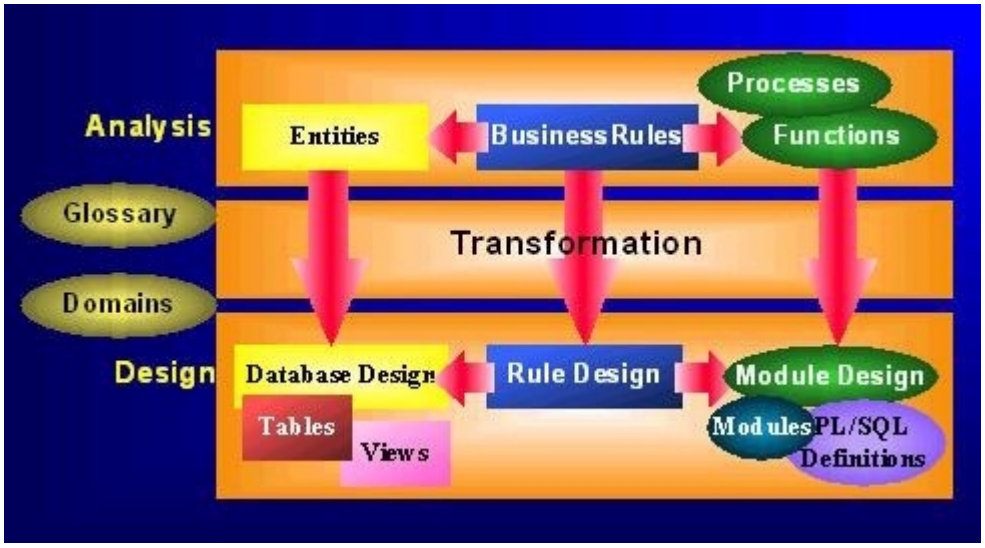
Provides customer with progress visibility	Poor	Fair	Excellent	Excellent
Provides management with progress visibility	Fair	Poor	Excellent	Fair
Requires little manager or developer sophistication	Fair	Excellent	Poor	Poor

6. Role of Oracle Designer in Rapid Development

Oracle Designer is a tool, not a 'silver bullet'. As a tool, it helps a data architect, data modeler or business analyst in the same way a word processor helps a writer. A word processor does not make a writer a Pulitzer Prize winner but it will make them more efficient and more grammatically correct. Oracle Designer will not make a bad data model good, but it will make the modeler more efficient.

Oracle Designer is a 'repository' based system that allows you to record business and data information gathered from such activities as JAD (Joint Application Development) sessions and prototyping. This information can be used as input to Oracle Developer for the next phase of development.

Where does Oracle Designer help in rapid development as a tool?



(Oracle Corporation)

- **Waterfall or Spiral Lifecycle Approach**

Strategy Phase

The Strategy portion of the lifecycle determines what information is needed by the business (a.k.a. entities) and what business functions need to be performed. The entity definitions become the building blocks for the table definitions that later are transformed into physical database tables. The business functions become the building blocks for the program specifications that are later transformed into programming modules.

- Oracle Designer has a number of ways to document and view business functions using the *Function Diagrammer*, the *Dataflow Diagrammer*, the *Process Modeler*, and the *Repository Object Navigator* (RON) interface.

- Entities may be documented and viewed in the *Entity Relationship Diagrammer* and the *Repository Object Navigator* (RON) interface.

TRICK

Cross Checking

The concept of cross checking your work is one of the fundamental principles in software engineering. Oracle Designer has included the *Matrix Diagrammer* to cross check that every entity has an appropriate function and that the functions have appropriate entities. This is one step, although tedious, that should not be overlooked.

Case Study

During an enterprise wide strategy study for an international commodity trading company, the IT design team interviewed and documented all the functional areas within the corporation. At the end of the strategy phase both a function hierarchy and a full entity diagram were produced. The technical team immediately wanted to go into the analysis and design phases of the project. The IT Director insisted on having a function to entity cross-reference to confirm that all the entities and all the functions were in agreement. The programming staff insisted that they could not have missed anything and thought the exercise a complete waste of time.

The Director led the actual cross-reference mapping exercise that took about a week to complete. During the sessions, the team discovered that they had missed an entire subject area in the entity model. This required a rework of the enterprise entity model to address the newly discovered requirements.

• The Timebox Prototyping Approach

There may be a need to bypass all strategy and analysis activities to quickly demonstrate what the proposed system might look like to the client. In this lifecycle approach, you would skip the creation of the entity and the functions and go directly to the first-cut database design using the *Repository Object Navigator* (RON) and the *Data Diagrammer*.

TRICK

Other Thoughts on Avoiding Classic Mistakes

- Which strategy works depends on your business and current systems
  - Dot.com
  - Legacy system conversion
  - New system, established business
- Any methodology is better than none
- Have frequent milestones – are you going where and when you want
- Do not rush into coding before you know what you need to code
- Document, document, document and then document
- Using a repository to have a ‘living’ strategy and design
- Need a target to hit
- Develop key players and teamwork
- Information hiding is good but distorting data structures as ‘shortcuts’ is bad
- Involve the business and
- Remember, there are no Silver Bullets

Bill Cunningham founded Cunningham Systems Corporation, a Connecticut based consulting firm, in 1998 with the mission of helping organizations 'get it right the first time' and specializes in all phases of rapid development strategy and design. Bill is highly knowledgeable in data management, data warehouse systems and business modeling, having 10 years of professional experience in developing strategy studies in preparation for installing ERP (Enterprise Research Planning) systems, and business re-engineering to e-commerce applications. He is also highly knowledgeable and sensitive to Information Technology management needs and business requirements, having 15 years of experience as CIO or Director of Information Technology.

Bill may be reached at bill@cunninghamsystems.com