Strategies for Rapid Development in Internet Time

William A. Cunningham
December 5, 2000
NYOUG
New York, NY
Productivity differences can be 10 to 1 in developing systems in the same industry!

2/3rds of all projects significantly miss time and cost estimates

Size of project slippage increases with the size of the project

Rapid development issues not limited to the Internet
Purpose

- Introduce some of the concepts of Rapid Development
- Provide ideas to help you get your software projects under control
- Provide information on proven concepts that have helped other organizations succeed.
• Define what Rapid Development Is
• Define and discuss the rules that make rapid development effective
• Discuss which flavor of rapid development works best for your organization
• Define 4 major lifecycle methods
• Discuss the role of Oracle Designer in rapid development
Audience

• Everyone - as they are affected by slow development of systems
• Especially for
  – Managers
  – Tech Leads
  – DBAs and Data Modelers
What is Rapid Development?

• Different Viewpoints
  – Some Developers
    • particular software, hardware tool or method that produces an application
  – Dot.Com Developer
    • 22 hour workdays and a cot under the workstation
  – Data architect
    • CASE methodologies, JAD sessions, tight schedules
  – Manager
    • latest methodology reviewed in Computerworld
Rapid Development

• Definition
  – Rapid Development is the effective and efficient creation of application systems within a full-fledged strategy
  – Or, put simply, it is developing something more quickly and efficiently than we normally do.
Rapid Development - Minimizing Timeframe

- Each organization has their own ‘hot’ button for development:
  - Speed-oriented - those that improve development speed, producing faster code
  - Risk-avoidance - those that address technology and business risk, avoiding severe schedule overruns
  - Visibility-oriented - those that show progress, avoiding the illusion of slow development
Rules of Rapid Development

Rapid Development Strategies are quite simple and can be stated in 3 basic rules:

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

- Rules ARE Simple
- But Rules are Not Easy to Follow
- Successful rapid development requires ALL of the rules to be in place
- If any rule is ignored, the project schedule is probably in jeopardy.
1. Apply Proven Methodologies and Development Fundamentals

- Methodology defined
  - A methodology is a body of practices, procedures, and rules used by those who work in a particular field or specialty

- Some proven methodologies are:
  - CASE*Method - Richard Barker - Oracle
  - The Zackman Framework - John Zachman
  - Rational Unified Process - Booch, Jacobson and Rumbaugh
Oracle Designer is NOT a Methodology

- A methodology is a disciplined way of approaching a system
- Oracle Designer is a toolkit
- Oracle Designer implements the CASE*Method methodology of Richard Barker
- Designer helps document and engineer your work; it does not do the design for you!
2. Identify and Manage Your Technology and Business Risks

- What style risk management do you use:
  - Crisis management - fire fighting, after they have become a problem
  - Fix on Failure - Detect and react to risks after they have happened
  - Risk mitigation - Have resources ready to handle risks if they occur
  - Prevention - Implement plan to identify and prevent risks from being a problem
  - Elimination of root cause - eliminate risks before they can exist
Most Common Schedule Risks

- Feature Creep
- Shortchanging quality
- Overly optimistic schedules
- Inadequate design
- Silver-bullet syndrome
- Research-oriented development
- Mismatched technical skills
- Communication issues between developers and clients
Case Study -
What went wrong here?

- Large multinational organization wanted to be a ‘key player’ in the travel portal business within 6 months.
- CIO selected a software vendor with a software product that was new and untested in the commercial environment.
- The software product was not Oracle DBMS based, the target platform.
Case Study - What went wrong here?

- A quick ‘fit-gap’ analysis showed significant gaps that were documented.
- However, CIO sold the project to top management and a delivery date was already set before the gap analysis was complete.
- First release of software did not include required ‘gap’ functionality.
Case Study -
What went wrong here?

- The software was not ‘frozen’ until the end of the project
- Unit testing had not begun by the time full integration tests were scheduled to start.
- The QA work became one of ‘heroic effort’
- The development team worked harder and harder but bugs kept on cropping up.
- Daily builds sometimes had bugs resurface
Case Study - What went wrong here?

- Deadline came (and three others afterward) before software was stable enough for release
- System was released as a subset of the original design
What Rules Were Ignored?

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

- Feature Creep
- Shortchanging quality
- Overly optimistic schedules
- Inadequate design
- Silver-bullet syndrome
- Research-oriented development
- Mismatched technical skills
- Communication issues between developers and customers
Which Rapid Development Strategy Works For You?

- There is no ‘one size fits all’ development strategy
- There are no shortcuts to the methodology side of successful rapid development
- Need to devote the proper resources to the classic lifecycle phases of a project
Which Rapid Development Strategy Works For You?

<table>
<thead>
<tr>
<th>Activity</th>
<th>Small Project (2.5K lines code)</th>
<th>Large Project (500K lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategy &amp; Analysis</td>
<td>10%</td>
<td>30%</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>Code/Debug</td>
<td>25%</td>
<td>10%</td>
</tr>
<tr>
<td>Unit Test</td>
<td>20%</td>
<td>5%</td>
</tr>
<tr>
<td>Integration</td>
<td>15%</td>
<td>20%</td>
</tr>
<tr>
<td>System Test</td>
<td>10%</td>
<td>15%</td>
</tr>
</tbody>
</table>
Which Rapid Development Strategy Works For You?

• Depends on your specific business, corporate culture and current systems

• Need to know if your project has:
  – A very strong project schedule constraint, such as a regulatory deadline or startup funding constraint
  – A top management or user request for ‘rapid development’ that translates into a desire for lower cost or less risk
  – Do you have any limitation or weakness that would prevent a rapid development success?
Rapid Development - A Lifecycle Approach

- The Pure Waterfall - the granddad of other, more effective lifecycle models
- The Code and Fix - a common, but not rapid, development model
- Spiral Development - breaks the project into manageable submodels, a RUP approach
- Timebox Prototyping - defining the specifications as the system is prototyped
The Waterfall Model

- Strategy
- Analysis
- Design
- Prototype
- Build
- Test
- Deploy
<table>
<thead>
<tr>
<th>Feature</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works with poorly defined requirements</td>
<td>Poor</td>
</tr>
<tr>
<td>Works with poorly understood architecture</td>
<td>Poor</td>
</tr>
<tr>
<td>Produces highly reliable system</td>
<td>Excellent</td>
</tr>
<tr>
<td>Produces systems with large growth envelope</td>
<td>Excellent</td>
</tr>
<tr>
<td>Manages Risk</td>
<td>Poor</td>
</tr>
<tr>
<td>Can be constrained to a predefined schedule</td>
<td>Fair</td>
</tr>
<tr>
<td>Has low overhead</td>
<td>Poor</td>
</tr>
</tbody>
</table>
The Waterfall Model - Summary

- Allows for midcourse corrections: Poor
- Provides customer with progress visibility: Poor
- Provides management with progress visibility: Fair
- Requires little manager or developer sophistication: Fair
If you have not selected a methodology, you are probably using this model.

CLASSIC MISTAKE
- ‘I do not have time for strategy and analysis; I need to start coding right away’

Code ‘like Hell’ technique

22 Hour days and Crisis management
Code & Fix

Fuzzy Specs

Code & Fix

System (Perhaps??)
Code and Fix - Summary

- Works with poorly defined requirements: Poor
- Works with poorly understood architecture: Poor
- Produces highly reliable system: Poor
- Produces systems with large growth envelope: Poor to Fair
- Manages Risk: Poor
- Can be constrained to a predefined schedule: Poor
- Has low overhead: Excellent
Code and Fix - Summary

- Allows for midcourse corrections: Unknown
- Provides customer with progress visibility: Fair
- Provides management with progress visibility: Poor
- Requires little manager or developer sophistication: Excellent
Spiral Development

- Breaks project into sub projects
- Handles high risk areas first
  - poorly understood requirements
  - poorly understood architecture
  - potential performance issues
- Then the model finishes as a classic waterfall
Spiral Development
Spiral Development - Summary

- Works with poorly defined requirements: Excellent
- Works with poorly understood architecture: Excellent
- Produces highly reliable system: Excellent
- Produces systems with large growth envelope: Excellent
- Manages Risk: Excellent
- Can be constrained to a predefined schedule: Fair
- Has low overhead: Fair
Spiral Development - Summary

- Allows for midcourse corrections: Fair
- Provides customer with progress visibility: Excellent
- Provides management with progress visibility: Excellent
- Requires little manager or developer sophistication: Poor
Timebox Prototyping

- Define specifications as you are coding the system
- Starts as ‘fuzzy’ specification of client requirements
- Produces a prototype within a specified timeframe
- Process repeated until client is satisfied
Timebox Prototyping

Initial Idea

Design and Implement Prototype

Acceptable?

Yes

Complete and Release Prototype

No
Timebox Prototyping - Summary

- Works with poorly defined requirements: Excellent
- Works with poorly understood architecture: Poor to Fair
- Produces highly reliable system: Fair
- Produces systems with large growth envelope: Excellent
- Manages Risk: Fair
- Can be constrained to a predefined schedule: Poor
- Has low overhead: Fair
Timebox Prototyping - Summary

- Allows for midcourse corrections: Excellent
- Provides customer with progress visibility: Excellent
- Provides management with progress visibility: Fair
- Requires little manager or developer sophistication: Poor
Role of Oracle Designer in Rapid Development

- Oracle Designer is a tool
- Oracle Designer is 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.
  - Designer will not make a bad model good, but it will make the modeler more efficient
Role of Oracle Designer in Rapid Development

- Designer is a ‘repository’ based system
  - allows for business and data information gathered from activities such as JAD sessions and prototyping
  - information gathered in Upper CASE can be used for physical table design and as input into Developer for the next stages of development (generation)
Role of Oracle Designer in Rapid Development
Where does Designer Fit into the Lifecycle Approach?

• Waterfall or Spiral Lifecycle Models
  – Strategy Phase
    • Business Functions
      – Function Diagrammer
      – Process Modeler
      – Dataflow Diagrammer
      – Repository Object Navigator (RON)
  • Entity Information
    – Entity Relationship Diagrammer
    – Repository Object Navigator (RON)
Data and Functions - Cross Checking

- Cross checking work is one of the fundamental principles in software engineering.
- Designer provides the *Matrix Diagrammer* to help cross reference each entity to a function or functions and visa versa.
  - Black holes
  - Novas
Where does Designer Fit into the Lifecycle Approach?

- Timebox Prototype Approach
- There may be a need to bypass strategy and analysis to quickly demonstrate the proposed system to the client
- Skip creation of entity and functions and go directly into first-cut database design using RON and the Data Diagrammer
Other Thoughts and Avoiding Classic Mistakes

• Which strategy works depends on your business and your current systems
  – dot.com
  – legacy system conversion
  – new system, established business
• Any methodology is better than none
• Have frequent milestones - know where you are and where you want to be
Final Thoughts and Avoiding Classic Mistakes

- Do not rush into coding before you know what you need to code
- Document, document, document and document
- Use a repository to have a ‘living’ strategy and design
- Need to have a target to hit
Final Thoughts and Avoiding Classic Mistakes

- 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
Thanks - and any Questions?

William A. Cunningham
203.888.0649
Bill@CunninghamSystems.Com