Choosing a Model

Why do you want to categorize and use a specific process model?

- To apply the best approach to the project at hand
- To standardize development for your organization which is a CMMI level 3 requirement
- Historical data or metrics can best be applied to prediction of cost and schedule in the context of past performance characteristics
- Process models and methodologies can be fine-tuned for each category
- Automated environments can be developed to fit the characteristics of each category

Examine and determine the characteristics of the project at hand. Combinations of these factors will determine the category and therefore best approach:

- What is the level of knowledge and specificity of requirements?
    - Are requirements known and stable?
    - Are user needs unclear and vague?
    - Is the software large or complex?
    - Is the software small or limited in functionality?
    - Is the software highly interactive with users?
    - Does the software involve client/server function?
    - Must the full system be implemented?
    - Must the system be responsive to user needs?
    - Must progress be demonstrated early?
    - Is user feedback needed?
- What are the budget and schedule constraints?
    - Must initial cost and schedule estimates be followed?
    - Must the costs of fixes and corrections be reduced?
- To what degree is software being created, modified, or deleted?
    - Is there new software?
    - Is there a COTS or 4GL product solution available?
    - Is reusable software available?
    - Does this effort involve converting from one language/operating system to another?
- What is the extent to which this effort is completely new or is this an enhancement to an existing system?
    - Must this system interface with other systems?
    - Will this system need to integrate with new or future technology?
    - Must this system minimize impact on current operations?
- What is the degree of technological risk?
    - Are there significant risks that need to be addressed?
- What is the degree of freedom allowed by the customer?
    - Is an early initial operational capability needed?
    - Is early functionality needed to refine requirements for subsequent deliveries?
- What is the availability of resources including people and equipment?
    - Is detailed documentation necessary?
    - Can the number of required people be reduced?
    - Can the management of the project be simpler?

This table shows the characteristics of the different process models for comparison.

Category	Characteristics	
Code & Fix	• Code used for analysis/problem solving • Small amount of code	• One developer • Thrown away after results obtained
Waterfall	• Requirements well defined • Predictable development • Known environment • Experienced development team	• Enhancing/upgrading existing system • No technological risks • Reasonable budget/schedule • Interfaces established/unchanging
Evolutionary (or Increment Build)	• Most requirements well defined • Budget/schedule constraints • Some technology risks • Integration with other systems/subsystems	• Some evolving requirements • Interfaces not well defined • New system, never done before • COTS and/or reusable software not available
Spiral	• Very large software system • Long schedule (5 to 10 years)	• Many risk areas • Emphasis on system reliability
COTS Integration	• Requirements can be satisfied by COTS products • Numerous functional capabilities in commercial	• Government furnished components • Intended to be inexpensive solution • Often assumed “simple” solution
Agile Development	• Requirements unstable • Unpredictable development • Unknown environment • Minimal planning	• Anticipates Requirements Volatility • People-oriented • Highly Collaborative • Focus on Construction

This table shows the general capabilities.

Lifecycle Model Capability	Code and Fix	Waterfall	Evolutionary	Spiral	COTS Integration	Agile Development
Works with ambiguous requirements	Poor	Poor	Fair to Excellent	Excellent	Excellent	Excellent
Works with ambiguous architecture	Poor	Poor	Poor	Excellent	Poor to Excellent	Excellent
Produces highly reliable system	Poor	Excellent	Fair to Excellent	Excellent	Poor to Excellent	Poor to Excellent
Produces system with large growth envelop	Poor	Excellent	Excellent	Excellent	N/A	Fair
Manages risks	Poor	Poor	Fair	Excellent	N/A	Fair
Can be constrained to a predefined schedule	Poor	Fair	Fair	Fair	Excellent	Poor
Minimal management & tech oversight to use the model	Excellent	Poor	Fair	Fair	Excellent	Fair
Allows for midcourse corrections	Poor to Excellent	Poor	Fair to Excellent	Fair	Poor	Excellent
Provides customer with progress visibility	Fair	Poor	Excellent	Excellent	N/A	Excellent
Provides management with progress visibility	Poor	Fair	Excellent	Excellent	N/A	Excellent
Requires little manager or developer sophistication	Excellent	Fair	Fair	Poor	Fair	Fair