Application Lifecycle Model (ALM) of Development

Most application developments fail
- Requirements not met
- Performance not met
- Need disappears before application finished

ALM is just a model, in reality steps overlap, are repeated, not followed at all, etc.

“Flowers4u” example – want service for ordering flowers

1) CONCEPTUALIZATION
   What is the vision?
   Customers submit orders through web-interface
   Orders are forwarded to local flower shops.
   Customer pays flowers4u with a credit card.
   Local flower shop is sent a check.
   Customer is presented with pictures of different arrangements?
   Are arrangements standardized or location dependent?
   If location dependent, how do we get pictures of all these things?
   In the conceptualization phase we are sketching
   Requirements for application
   The business processes that the application will support.
   An opportunity to reengineer an existing business process.
   We may also be creating a business case
   What is the value added?
   Is the investment to make this worthwhile?
   In this phase we can build simple prototypes
   A website of the customer interface

2) ANALYSIS
   What will the application do?
   Not detailed implantation, but enough specification to allow stakeholders to review idea.
   Define Scenarios in which it is used
   Customer – “I want to have flowers delivered to funeral home in Chicago”
   Web interface presents flower arrangements available in Chicago
   Customer picks one, enters credit card information, clicks buy
   Order forwarded to florist in Chicago. (Call? Email?)
   Flowers 4u verifies fulfillment
   Check mailed to florist at end of month for all orders that florist fulfilled.
   What are the performance requirements?
   How many orders per day to handle?
   How many people browsing our site?
3) ARCHITECTURE DESIGN
   Decompose the application into subsystems
      - hardware
      - software
   Decompose to take advantage of as much commercial off-the-shelf subsystems as possible.
      - Requires keeping up with constantly changing portfolio of available COTS

1) Three tier client server architecture
   Customer logic (decompose into app-server and web-server?)
      Apache web server software?
      Build on IBM Websphere app server?
      J2EE?
   Fulfillment logic
      (Websphere? Oracle e-business suite, custom designed applications?)
   Databases for accounts, products, and orders
      (Buy DBMS from Oracle, IBM, or mySQL)

Define the functionality, interaction, and interfaces.
Make subsystems as independent as possible so tasks can be given to different groups.
Crucial considerations:
   Scalability – ease of adding capacity later
   Extensibility – ease of adding new capabilities later
   Administration

4) DEVELOPMENT EVOLUTION
   Programming to fill in the details
   Incremental
      Add detailed capabilities with successive refinement
      Start with simplest implementation possible and get it working
      Later, add more features

5) TESTING
   A must.
   If architected well, subsystems can be tested independently.
   Alpha test – off line test of prototype
   Beta test – tested in intended environment with cooperative users.

6) DEPLOYMENT
   Conversion from previous processes
   (All at once CISCO ERP, or incremental)
   User training
Data importation

7) OPERATIONS, MAINTENANCE and UPGRADE
   Maintain security
   Repair problems
   Correct performance shortcomings (CISCO ERP)
   Upgrades to add features

Buy vs. Make vs. Outsource (Outsource – hire another company to do development)
   Options for whole application, or for subsystems
   -If whole application is bought, ALM might be simplified.
     (But often choosing configuration options and adjustments can be a complicated ALM project as well – CISCO ERP.)
   -If major subsystems are outsourced architecture must clearly specify requirements of outsourced portion.

ALM more rigorously followed for large complex projects in large organizations. Another approach – rapid prototyping.