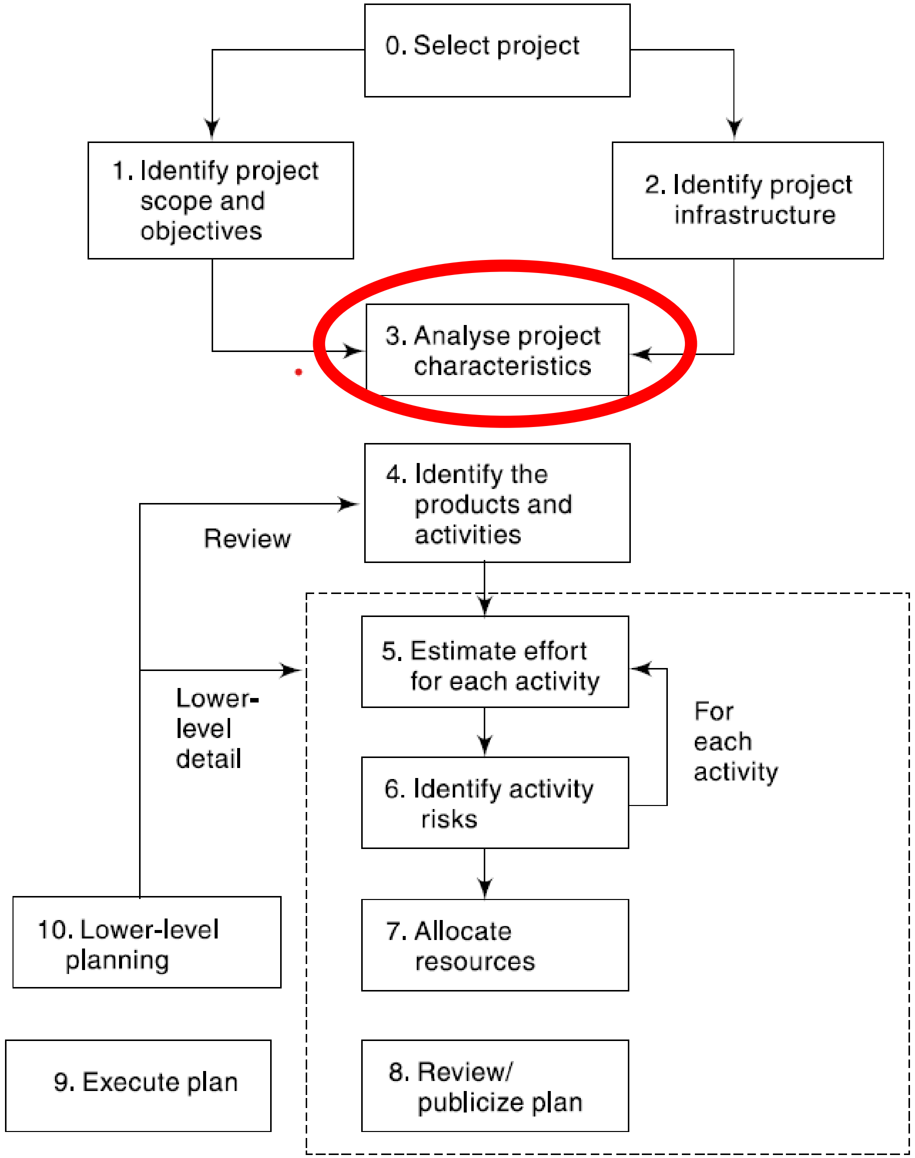
Software Project Management

Lecture 07

Selection of an Appropriate Project Approach

Step wise- Overview



0. Select project
1. Identify project scope and objectives
2. Identify project infrastructure
3. Analyse project characteristics
4. Identify the products and activities
Review
Lower-level detail
5. Estimate effort for each activity
6. Identify activity risks
For each activity
7. Allocate resources
8. Review/ publicize plan
9. Execute plan
10. Lower-level planning

Step 3: Analyze Project Characteristics

- Step 3.1 Distinguish the project as either objective-driven or product-driven
  - Object v/s Product
  - Is there more than one way of achieving success?
  - Mostly in earlier stage projects are objective and in later stage it become product driven

Step 3: Analyze Project Characteristics

- Step 3.2 Analyze other project characteristics (including quality-based ones)
    - What is different about this project?
    - Whether it is a information system or embedded system?
    - What are the critical issues?
    - Malfunctioning threatened human life?

Step 3: Analyze Project Characteristics

- Step 3.3 Identify high level project risks
    - What could go wrong?’
    - What can we do to stop it?’

- Step 3.4 Take into account user requirements concerning implementation
    - For example customer want the product in java

Step 3: Analyze Project Characteristics

- Step 3.5 Select general lifecycle approach in the light of the above
  - Waterfall? Increments? Prototypes?
  - Generally Company use the same method or SDLC as they were using in past
  - It also depends on project need.

Analyze Project Characteristics

- There is two way to develop a software
    - In-house
    - Software houses
- Either it is in-house development or by software houses it is required to review methodologies and techniques for each project
- This process of reviewing is known as technical planning or project analysis

Steps for Analyze Project

- Identify project as either objective driven or product driven
- Analyze other project characteristics
  - Is a data-oriented or process-oriented
    - Data oriented- Information systems
    - Process oriented- embedded control system
  - Will the software to be produce is a general tool or application specific?
  - Are there tool available for implementing the particular type of application?
  - Is the system to be critical?
  - What is the nature of hardware and software environment?

Steps for Analyze Project

- Identify high level project risk
  - Some of the risks are:
    - Product uncertainty
    - Process uncertainty
    - Resource uncertainty
  - Take into account user requirement concerning implementation.
  - Select general life-cycle approach

Technical Plan Content List

- Output of project analysis is technical plan.
- It contains
  - Introduction and summary of constraints:
    - Character of the system to be developed
    - Risk and uncertainties
    - User requirement concerning implementation
  - Recommended approach
    - Select methodology or process model
    - Development methods
    - Required s/w tools
    - Target hardware/software environment

Technical Plan Content List

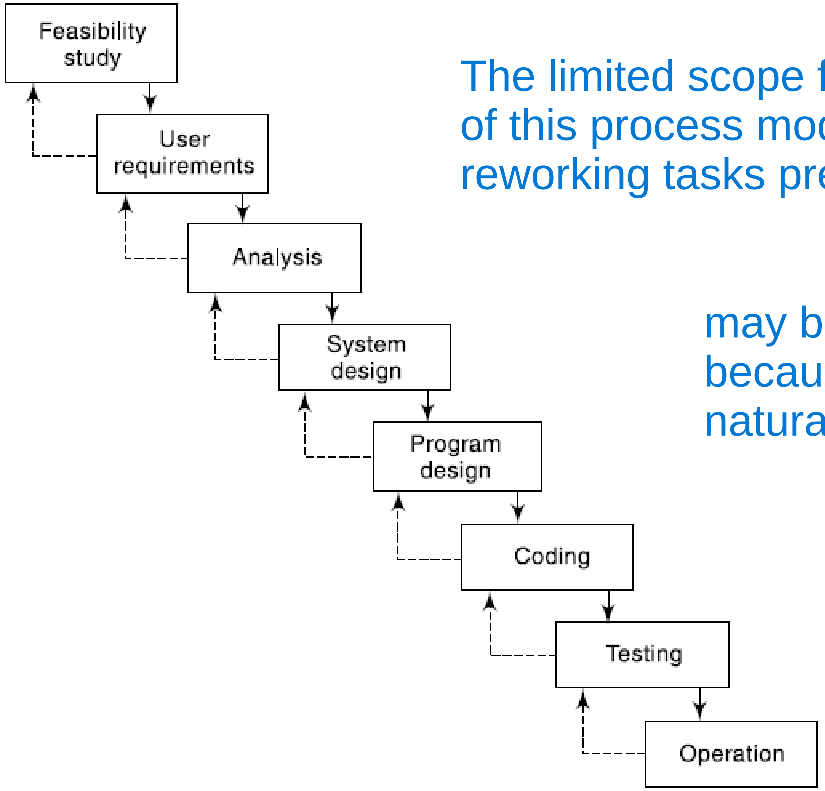
- Development needs
  - Required development environment
  - Required maintenance environment
  - Required training
- Implementations
  - Project products and activities
  - Financial- this report will be used to produce costing

Software Processes and Process Models

also known as the one-shot or once-through

Waterfall

a later stage may reveal the need for some extra work at an earlier stage, but this should definitely be the exception rather than the rule



The limited scope for iteration is in fact one of the strengths of this process model. With a large project you want to avoid reworking tasks previously thought to be completed.

may be favoured by some managements because it creates natural milestones at the end of each phase

Waterfall

- The ‘classical’ model
- Every stage needs to be checked and signed off
- Ideal for:
  - Where requirement are well defined
  - Development methods are well understood

BUT

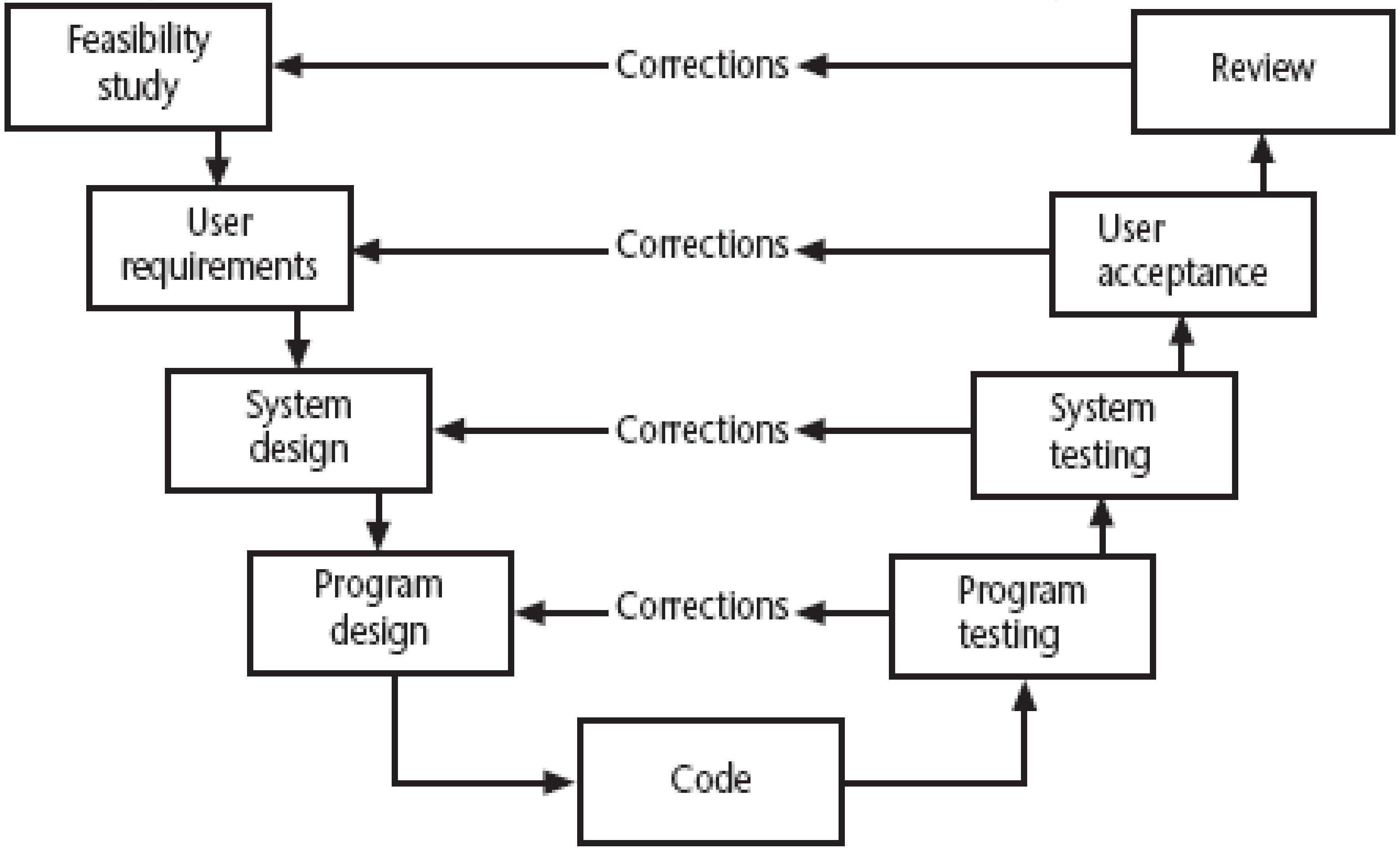
  - Limited scope for iteration

where there is uncertainty

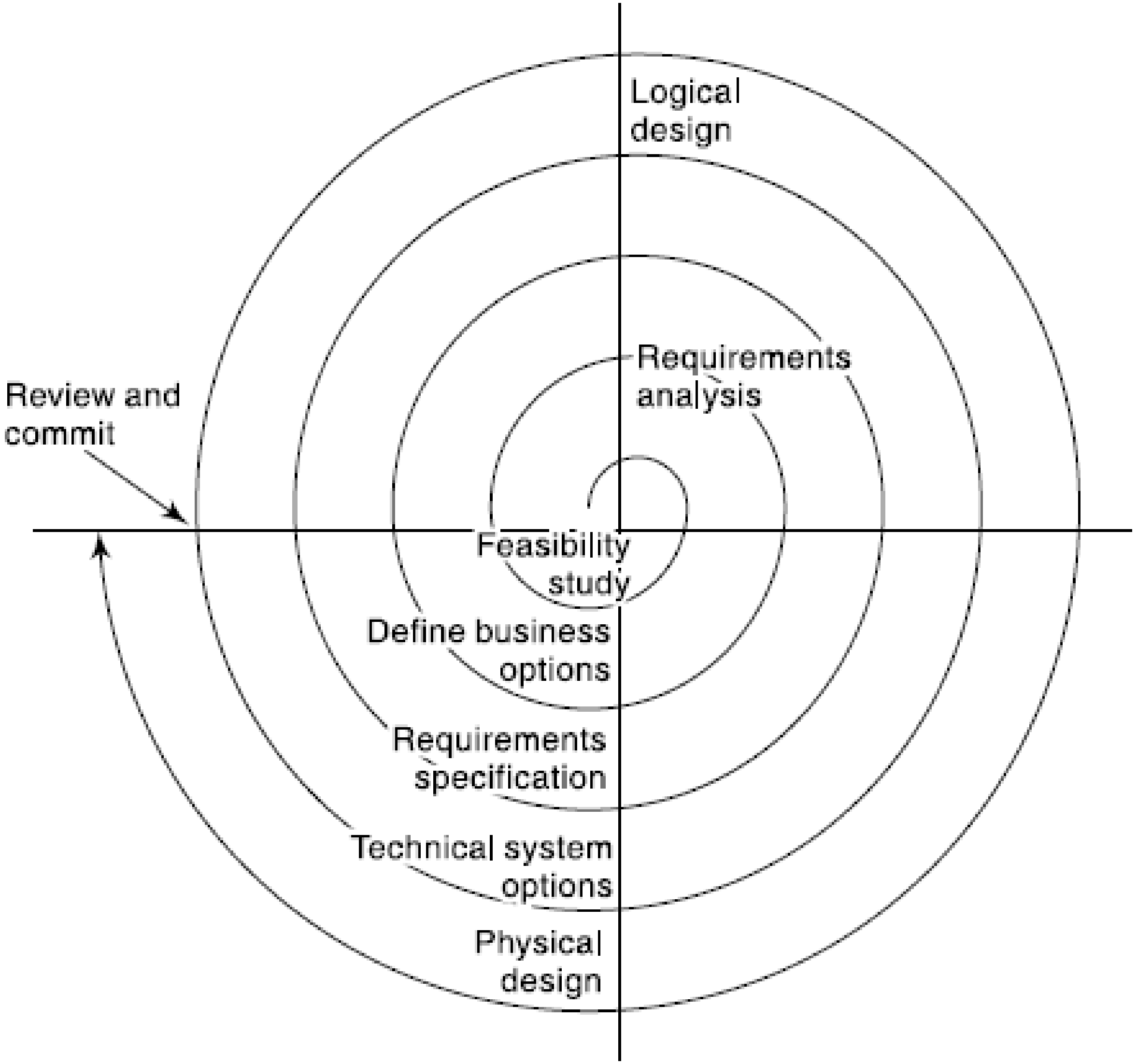
The V-process model can be seen as expanding the activity box ‘testing’ in the waterfall mod

V-Process Model

Each step has a matching validation process which can, where defects are found, cause a loop back



Spiral Model



- The distinguishing characteristic features of the spiral model are the incremental style of development and the ability to handle various types of risks.
- Each loop of the spiral is called a phase of this software process. In each phase, one or more features of the product are implemented after resolving any associated risks through prototyping.
- The exact number of loops of the spiral is not fixed and varies from one project to another.
- Note that the number of loops shown in Figure 4.4 is just an example illustrating how the spiral model can subsume SSADM. Each loop of the spiral is divided into four quadrants, indicating four stages in each phase.
- In the first stage of a phase, one or more features of the product are analysed and the risks in implementing those features are identified and resolved through prototyping
- In the third stage, the identified features are implemented using the waterfall model.
- In the fourth and final stage, the developed increment is reviewed by the customer along with the development team and the features to be implemented next are identified. Note that the spiral model provides much more flexibility compared to the other models

Prototyping (Evolutionary Delivery)

‘An iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions’

- Main types:
  - ‘throw away’ prototypes

    tests out some ideas and is then discarded when dev starts
  - evolutionary prototypes

    prototype is developed and modified until it is finally in a state where it can become the operational system
- What is being prototyped?
  - human-computer interface
  - functionality

Why Prototyping?

- Learning by doing look back & see where we have made mistakes
- Improved communication
- Improved user involvement users can be more actively involved in design decisions
- A feedback loop is established
- Reduces the need for documentation Because a working prototype can be examined
- Reduces maintenance costs i.e. changes after the application goes live
- Prototype can be used for producing expected results

Dangers of Prototyping?

- Users may misunderstand the role of the prototype
- Lack of project control and standards possible
- Additional expense of building prototype
- Focus on user-friendly interface could be at expense of machine efficiency

Incremental Delivery Model

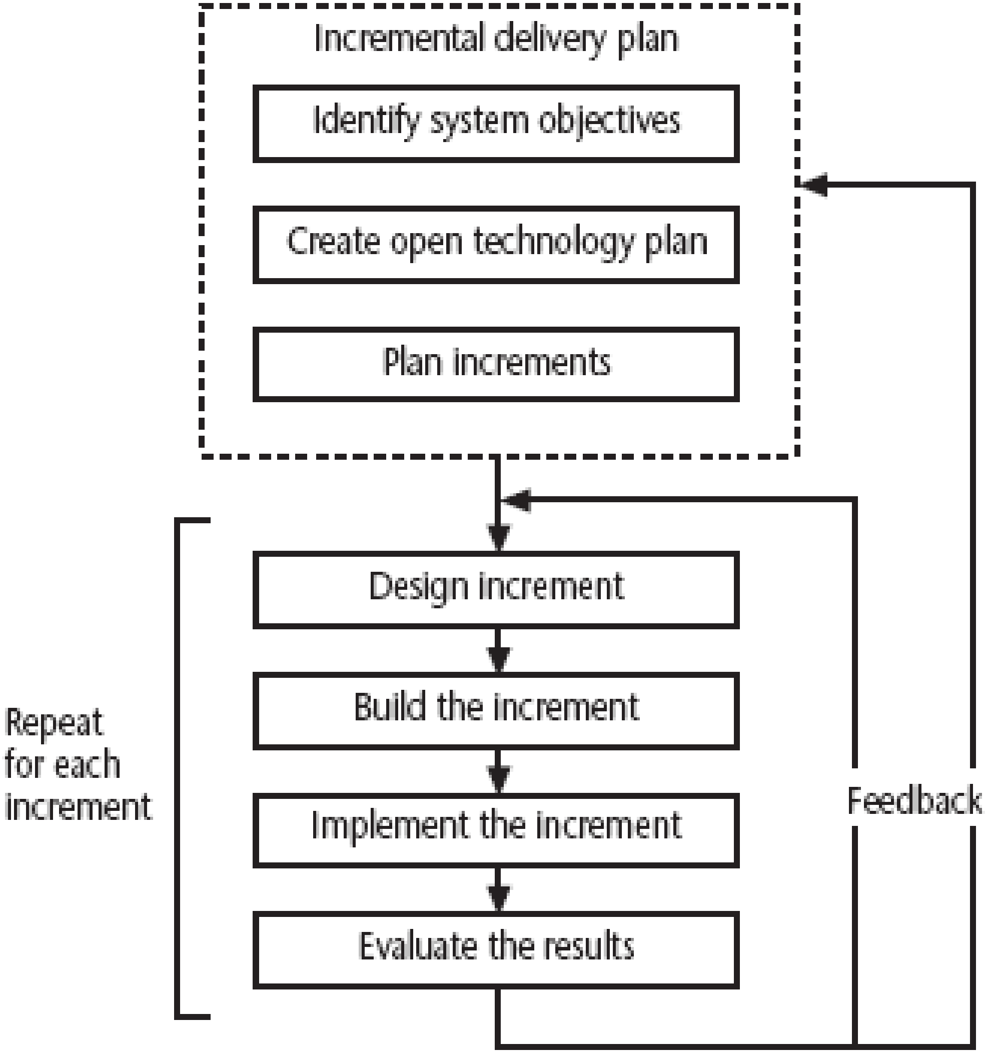
- In this model we break the application down into small components
- These components are then implemented and delivered in sequence

Each component delivered must give some benefit to the user

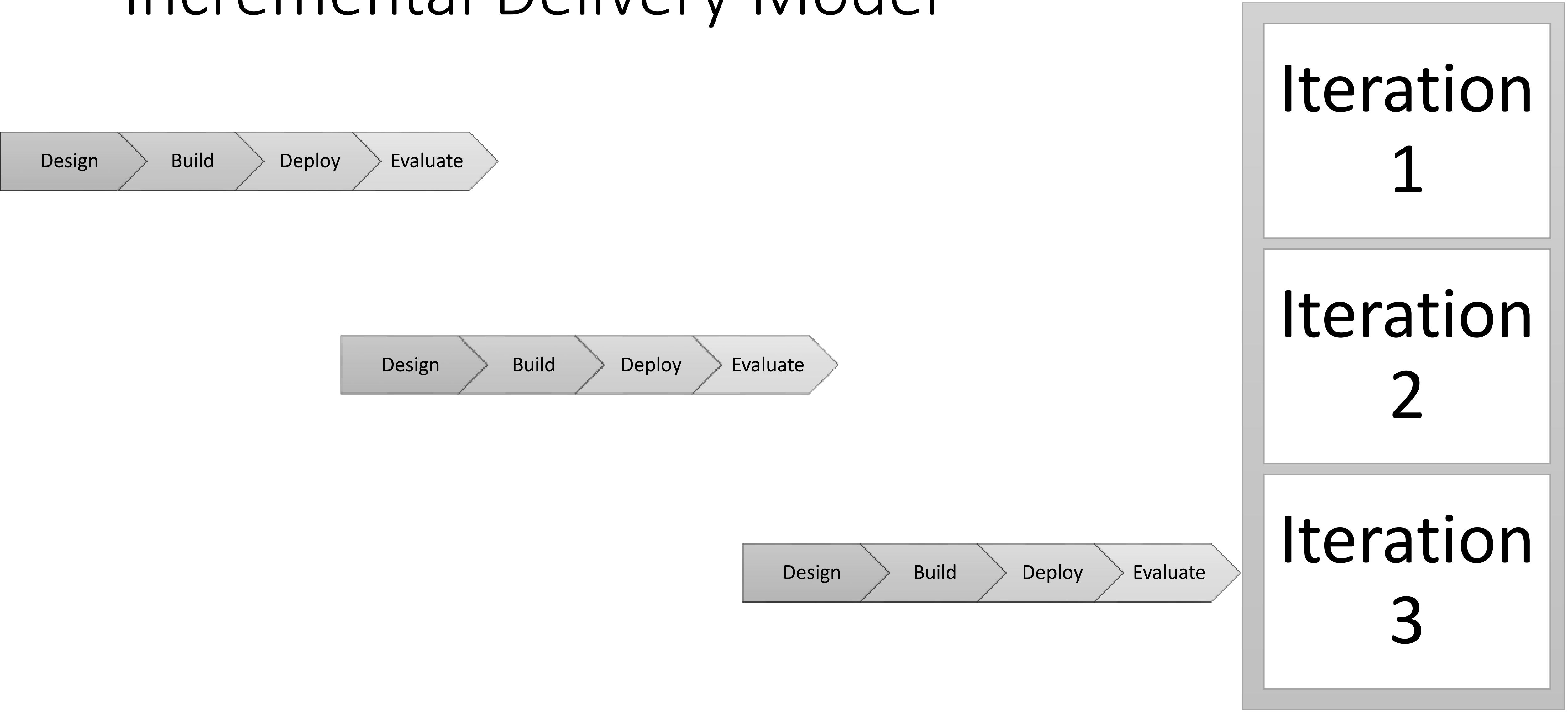
Time-boxing is often associated with an incremental approach. Here the scope of deliverables for an increment is rigidly constrained by an agreed deadline.

This deadline has to be met, even at the expense of dropping
some of the planned functionality. Omitted features can be transferred to later increments.

Incremental Delivery Model


Incremental delivery plan
Identify system objectives
Create open technology plan
Plan increments
Design increment
Build the increment
Implement the increment
Evaluate the results
Repeat for each increment
Feedback

Incremental Delivery Model



Design Build Deploy Evaluate

Design Build Deploy Evaluate

Design Build Deploy Evaluate

Iteration 1

Iteration 2

Iteration 3

Benefits of Incremental Delivery Model

- Feedback from early stages used in developing latter stages
- User gets some benefits earlier
- Easy to control and manage
- Project may be put aside temporarily

Gold-plating, that is, the requesting of features that are unnecessary and not in fact used, is less as users know that if a feature is not in the current increment then it can be included in the next

BUT there are some possible disadvantages

- Loss of economy of scale
    - More productivity at larger scale
    - Software developers may be more productive working on one large system than on a series of smaller ones
- 'Software breakage': Later increments might require modifications to earlier increments. This is known as software breakage

Incremental Delivery Plan

- The nature and order of each increment to be delivered to the user have to be planned.
- Elements of increment plan are:
  - System objective
  - Incremental plan
  - Open technology plan

IDP – System Objectives

- Project planner ideally wants well defined objective but as much freedom as possible about how to be met.
- Objectives
  - Function goal
    - Objective
    - Jobs the system is to do
    - Computer/non-compute function to achieve them
  - Quality Goal measurable quality characteristics
    - Reliability , response and security

IDP – Open Technology Plan

- If it is required to add new components continually to the system the system should be extendible, portable and maintainable
- So it requires:
  - A standard high level language
  - A standard operating system
  - Small modules
  - A standard DBMS

IDP – Incremental Plan

the next stage is to plan the increments using the following guidelines:

- Steps ideally 1% to 5% of the total project
- Ideal if a step takes one month or less:
  - not more than three months
- Each step should deliver some benefit to the user
- Some steps will be physically dependent on others

Rapid Application Development

tries to overcome this problem by inviting and incorporating customer feedback on successively developed prototypes. In the RAD model, absence of long-term and detailed planning gives the flexibility to accommodate requirements change requests solicited from the customer during project execution.

- Also is referred as Rapid Prototyping (incremental +prototyping)
- To decrease the time taken and the cost incurred
- To limit the cost of accommodating change request by incorporating them as early as possible
- Development is done in a series of short cycles called iterations
- Time for each iteration is called time box
- Each iteration enhances the feature/functionality a little
- During each iteration, a quick and dirty prototype for some functionality is developed. The customer evaluates the prototype and gives feedback
- RAD prototype is not released to the customer for regular use

Agile Methodologies

- Designed to avoid the disadvantages of traditional methodologies
- No long term plan is made
- Work is done in iterations
- Each iteration lasts for couple of weeks
- Each iteration plans, develops and then deploys at customer’s site
- It is a group of development processes and not a single model

Agile Methodologies

- Emphasizes face-to-face communication over written documents
- Team size is kept small deliberately
- Customer representatives are part of the team
- Usually apply pair programming

Agile Methodologies

- Multiple Agile approaches
  - Crystal Technologies
  - Atern (Formerly DSDM)
  - Feature-driven Development
  - Extreme Programming (XP)
  - Scrum

ATERN:

Focus on Business need
Deliver on Time
Collaborate
Never Compromise on Quality
Develop Iteratively
Build Incrementally
Communicate Continuously
Demonstrate Control
Main Lifecycle phases:

1. Feasibility/foundation. derivation of a business case + system architecture

2. Exploration cycle. e business requirements.

3. Engineering cycle. takes the design generated in the exploration cycle and converts it into usable
components

4. Deployment. This gets the application created in the engineering cycle into actual operational use.

Extreme Programming

- Taking commonsense to extreme levels
- It presents for core values
  - Communication & Feedback Formal documentation is avoided. face-to-face communication
  - Simplicity simplest design that implements the users’ requirements
  - Responsibility The developers are the ones who are ultimately responsible for the quality
  - Courage the courage to throw away work in which you have already invested a lot of effort, and to start with a fresh design if that is what is called for. It is also the courage to try out new ideas

Extreme Programming

- The Planning XP refers to increments as releases. Within these releases code is developed in iterations, periods of one to four weeks'. features in a release are decided.
- Small Releases e time between releases of functionality to users should be as short. 1 month or 2
- Metaphor The system to be built will be software code that reflects things that exist and happen in the real world
- Simple Design practical implementation of the value of simplicity
- Testing Testing is done at the same time as coding. The test inputs and expected results should be scripted so that the testing can be done using automated testing tools
- Refactoring

Extreme Programming

- Pair Programming one actually doing the typing and the other observing,
- Collective Ownership The team as a whole takes collective responsibility
- Continuous Integration
- 40 hour work weeks
- On-site Customer
- Coding standard

Limitation of Extreme Programming

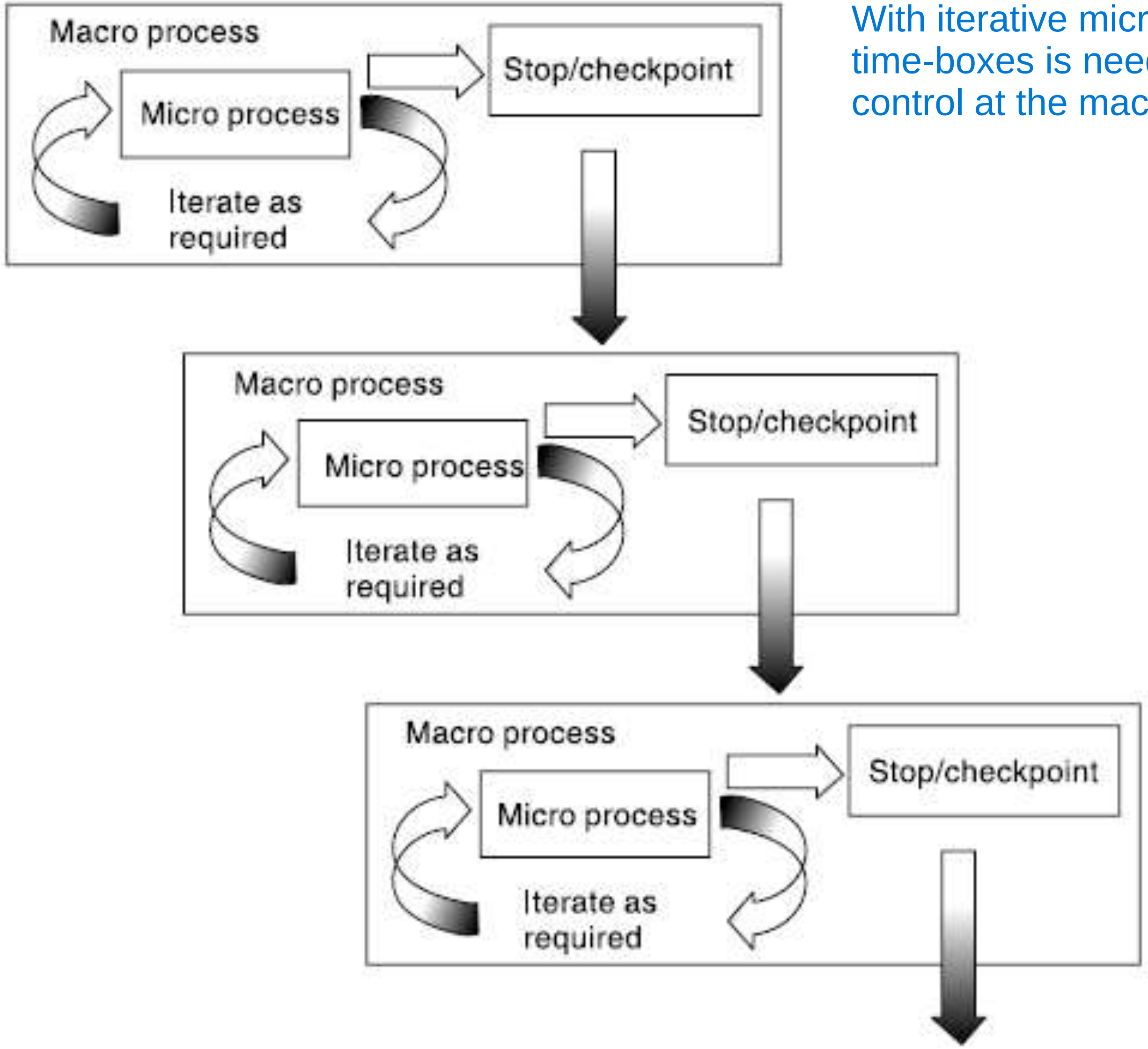
- ● There must be **easy access to users, or at least a customer representative** who is a domain expert. This may be difficult where developers and users belong to different organizations.
- ● Development staff need to be physically located in the same office. ● As users find out about how the system will work only by being presented with working versions of the code, there may be communication problems if the application does not have a visual interface.
- ● For work to be sequenced into small iterations of work, it must be **possible to break the system** functionality into relatively small and self-contained components.
- ● Large, complex systems may initially need significant architectural effort. This might preclude the use of XP.

Scrum

- Projects are divided into small parts of work that can be incrementally developed and delivered in *sprints*
- Product gets developed over a series of manageable chunks
- Each sprint takes couple of weeks
- At the end of each sprint progress is evaluated
- It assumes 2 riles
  - Product Owner
  - Scrum Master
  - Team Member

The macro process is closely related to the waterfall process model. At this level, a range of activities carried out by a variety of specialist groups has to be coordinated. Within this macro process there will be micro process activities which might involve iterative working.

Managing Iterative Process



With iterative micro processes, the use of time-boxes is needed to control at the macro level

Selecting the Most Appropriate Process Model

IF uncertainty is high
THEN use evolutionary approach

IF complexity is high but uncertainty is not
THEN use incremental approach

IF uncertainty and complexity both low
THEN use one-shot

IF schedule is tight
THEN use evolutionary or incremental

Selecting the Most Appropriate Process Model

Use combination of approaches

		Installation		
		One Shot	Incremental	Evolutionary
Construction	One Shot	Yes	Yes	No
	Incremental	Yes	Yes	No
	Evoltionary	Yes	Yes	Yes

Reference

- Software Project Management 5th Edition by Mike Cotterell, Bob Hughes, Rajib Mall

Selection of Appropriate Project Approach - Chapter 4