Software Evaluation and Costing

Contents

- Project Evaluation: Strategic Assessment, Technical Assessment, Cost-benefit analysis, Cashflow forecasting, Cost –benefit evaluation techniques, Risk Evaluation.
- Selection of Appropriate report
- Project approach: Choosing technologies, choice of process models, Structured methods

Project Evaluation

- A high level assessment of the project
  - to see whether it is worthwhile to proceed with the project.
  - to see whether the project will fit in the strategic planning of the whole organization.
- Projects must be evaluated based on strategic, technical and economic grounds.
- The strategic objectives combined with the outcomes of the project.

Project Evaluation - Why

- Want to decide whether a project can proceed before it is too late
- Want to decide which of the several alternative projects has a better success rate, a higher turnover, a higher ...
- Is it desirable to carry out the development and operation of the software system?

Project Evaluation - Who

- Senior management
- Project manager/coordinator
- Team leader

Project Evaluation - When

- Usually at the beginning of the project
- e.g. Step 0 of Step Wise Framework

Project Evaluation - What

- Strategic assessment
- Technical assessment
- Economic assessment

Project Evaluation - How

- Cost-benefit analysis
- Cash flow forecasting
- Cost-benefit evaluation techniques
- Risk analysis

Strategic Assessment (SA)

- Used to assess whether a project fits in the *long-term goal* of the organization
- Usually carried out by senior management
- Needs a strategic plan that clearly defines the objectives of the organization
- Evaluates individual projects against the strategic plan or the overall business objectives
- **Programme management** : suitable for projects developed for use in the organization
- **Portfolio management :** suitable for project developed for other companies by software houses

- A programme management is where a portfolio of projects all contribute common objective.

Eg:-Maintenance work of clients

- Customer's experience of the organization might be very variable and inconsistent.
- A business objective might be consistent and uniform.

SA – Programme Management

- Individual projects as components of a programme within the organization
- *Programme as "a group of projects that are managed in a coordinated way to gain benefits that would not be possible were the projects to be managed independently"*

SA – Programme Management Issues

Objectives

- How does the project contribute to the long-term goal of the organization?
- Will the product increase the market share? By how much?

IS plan

- Does the product fit into the overall IS plan?
- How does the product relate to other existing systems?

Organization structure

- How does the product affect the existing organizational structure? the existing workflow? the overall business model?

MIS

- What information does the product provide?
- To whom is the information provided? □ How does the product relate to other existing MISs?

Personnel

- What are the staff implications?
- What are the impacts on the overall policy on staff development?

Image

- How does the product affect the image of the organization?

SA – Portfolio Management

- Suitable for product developed by a software company for an organization
- May need to assess the product for the client organization
    - Programme management issues apply
- Need to carry out strategic assessment for the providing software company

SA – Portfolio Management Issues

- Long-term goal of the software company
- The effects of the project on the portfolio of the company (synergies and conflicts)
- Any added-value to the overall portfolio of the company

Technical Assessment

- Technical Assessment of a proposed system consists of evaluating the required functionality against the h/w and s/w available.
- The strategic IS plan of the organization
- any constraints imposed by the IS plan

Limitations

-nature of solutions produced by strategic information systems plan

-cost of solution ,hence undergoes cost-benefit analysis.

Economic Assessment

Why?

- Consider whether the project is the best among other options
- Prioritise the projects so that the resources can be allocated effectively if several projects are underway

How?

- Cost-benefit analysis
- Cash flow forecasting
- Various cost-benefit evaluation techniques
    - NPV and IRR

Cost-benefit Analysis

- An economic assessment of a proposed information system or software product is used to compare the expected costs of development and operation of the system.
- Assessment focuses on whether the estimated income and other benefits exceed the estimated costs.
- **Two steps,**
- Identifying and estimating all of the costs and benefits of carrying out the project and operating the delivered application.
- Expressing these costs and benefits in common units.
    - Estimate costs include the developmental, setup and operational costs involved in the process of   the new system.

Categories of Costs

- Costs can broadly classified into two groups,

1) Based on system
2) Based on benefits

- Many costs are easy to identify and measure in monetary terms.
- **Costs categories are,**
    - **Development costs:** salaries, employee benefit costs and all associated costs (PF, health Insurance, car allowance)
    - **Setup costs:** (h/w and s/w infrastructure, Recruitment/staff training, Installation, data taken on/data conversion)
    - **Training cost:**(Cost of trainer, equipment's and other needs of training)
    - **Operational costs:** (Help desk, Hosting costs , Licensing costs, Maintenance costs, Backup costs)

Cost Benefits Analysis-Benefits Types

- **Direct benefits:** Directly accountable to new system

– Cost savings (e.g., less staff, less paper, quicker turnaround)

– Money generation (e.g., new revenue stream, new markets) Measurable after system is operational

- **Indirect benefits:** Secondary benefits of new system

- Examples: better work flow, increased flexibility

- Somewhat quantifiable after the system is operational

- have to be estimated for cost/benefit analysis

- **Intangible benefits:** Positive side effects for new system

- External system (e.g., increase branding, entry to new markets)

- Internal system (increased interest in job for users, enabler for other systems)

- Often very specific to a project; not measurable even after a system is operational Part of strategic decision rather than cost/benefit analysis

Cash Flow Forecasting

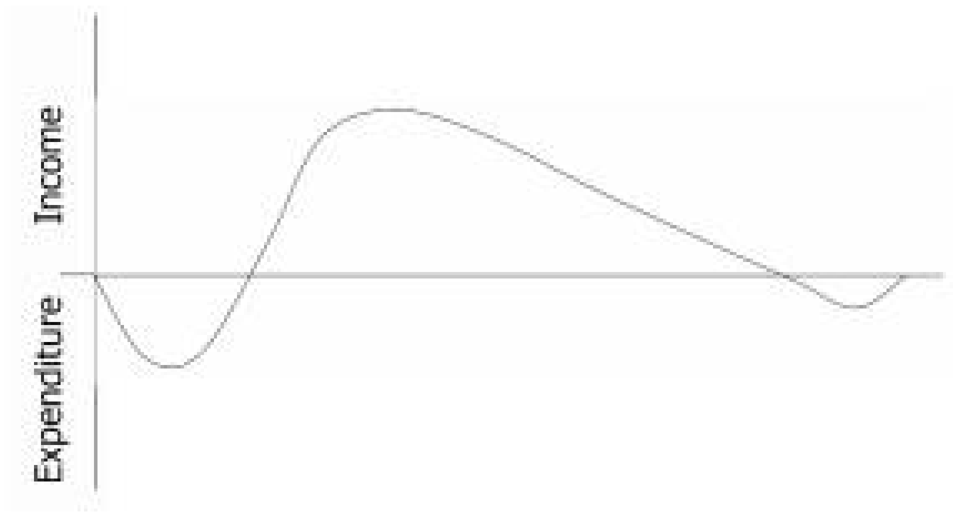
- Estimating the overall costs and benefits of a project is the forecasting.
- A cash flow forecast will indicate when expenditure and income will take place.

What?

- Estimation of the cash flow over time

Why?

- An excess of estimated benefits over the estimated costs is not sufficient
- Need detailed estimation of benefits and costs versus time



- Need to forecast the expenditure and the income
- Accurate cash flow forecasting is not easy.
- When estimating future cash flows, it is usual to ignore the effects of inflation.
- Forecasts of inflation rates tend to be uncertain.
- If expenditure is increased due to inflation it is likely that income will increase proportionately.

Cost Benefit Evaluation Techniques

- Cost-benefit analysis is a set of practical procedures for guiding public expenditure decisions.
- Project evaluation usually requires comparing costs and benefits from different time periods.
- Cash flow forecasting can be compared for different projects based on same general methods defined,
  1. **Net profit** =Total income-Total costs
  2. **Payback period**=Time taken to break even or pay back the initial investment
  3. **Return on Investment (ROI)/Average rate of return (ARR),**

  ROI=(average annual profit/total investment(Net investment in project))*100

  Example**:-** Calculating the ROI for project 1, the net profit is £50,000 and the total investment is £100,000. The return on investment is therefore calculated as

  ROI = (average annual profit /total investment)*100=((50,000/5)/100,000)*100 = 10%

4. Net Present Value (NPV),

- for example, invest the £100 in a bank today and have £100 plus the interest in a year’s time. If we say that the present value of £100 in a year’s time is £91, we mean that £100 in a year’s time is the equivalent of £91 now.
- The present value of any future cash flow may be obtained by applying the following formula

  Present value= value in year $t/(1+r)^t$

Where, **‘r’** – denotes the discount rate expressed as a decimal value.

‘t’ – represents the number of years of future cash flows.

- Net present value can also be calculated by multiplying the cash flow by the appropriate discount factor.
- NPV for project is obtained by summing the discounted values and discounting each cash flows.
- The NPV for project is obtained by discounting each cash flow and summing the discounted values.

5. Internal Rate of return:

- The IRR is calculated as that percentage discount rate that would produce an NPV of zero.
- A spreadsheet or a small computer program can be used to calculate the IRR is a convenient and useful measure of value of a project.
- The limitation of IRR is that it does not indicate the absolute size of the return value.

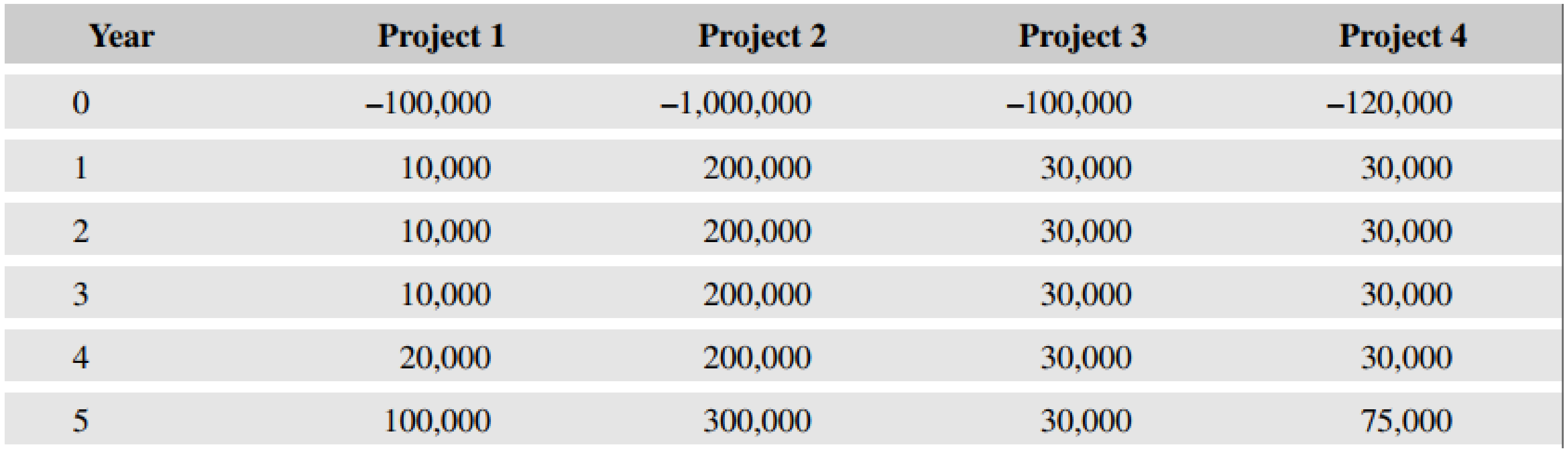
Example:

Year	Project 1	Project 2	Project 3	Project 4
0	–100,000	–1,000,000	–100,000	–120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000

Net profit

Year	Project 1	Project 2	Project 3	Project 4
0	–100,000	–1,000,000	–100,000	–120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net profit	50,000	100,000	50,000	75,000

Calculate Payback period



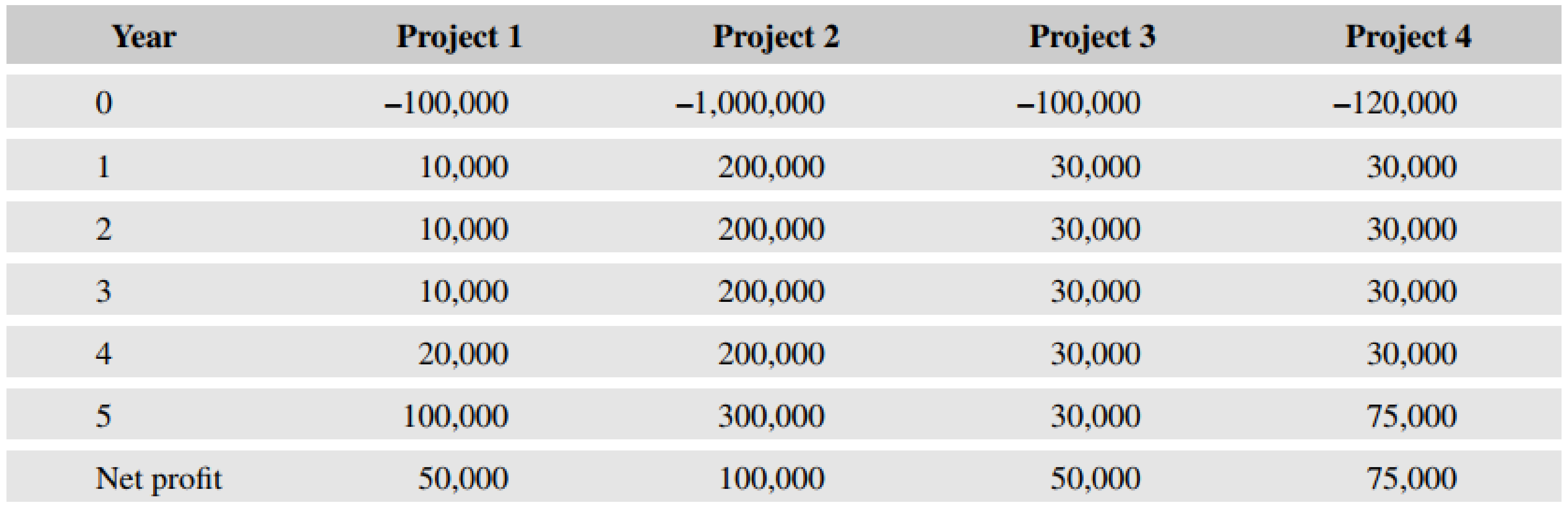
Year	Project 1	Project 2	Project 3	Project 4
0	–100,000	–1,000,000	–100,000	–120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000

Payback period

Project 1=10,000+ 10,000+10,000+20,000+1,00,00=1,50,000

Project 2=2,00,000+ 2,00,000+2,00,000+20,000+3,00,00=11,000,00

Calculate ROI



Year	Project 1	Project 2	Project 3	Project 4
0	–100,000	–1,000,000	–100,000	–120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net profit	50,000	100,000	50,000	75,000

ROI=(average annual profit/total investment)*100

ROI (project 1)=(50,000/5)/100,000*100=10%

ROI (project 2)=(100,000/5)/1,000,000*100=2%

ROI (project 3)=(50,000/5)/1000,000*100=1%

ROI (project 4)=(75,000/5)/120,000*100=12.5 %

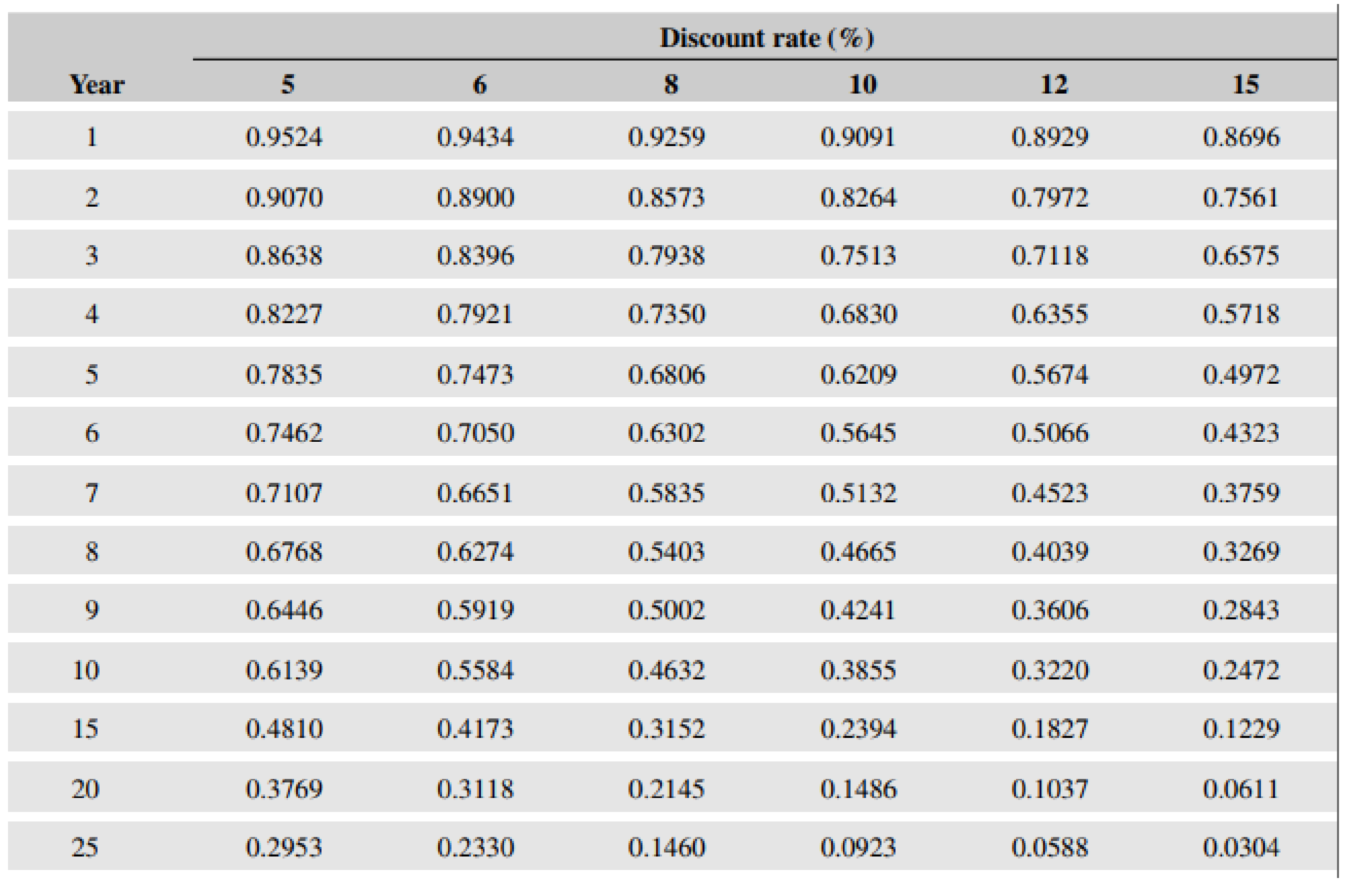
Calculate Net Present Value (NPV)

Present value= value in year t/(1+r)^t

Discount factor=1/(1+r)^t

Hence,

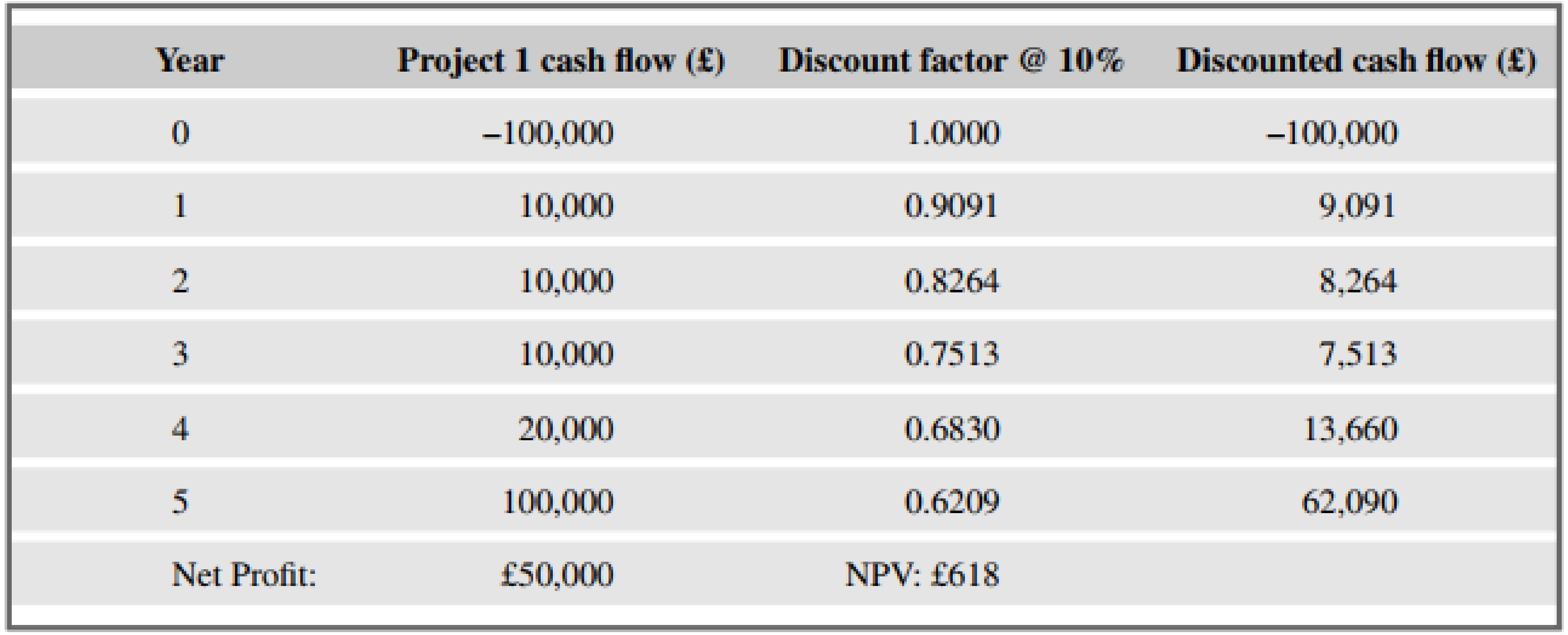
NVP=discount factor *value in year t



Year	Discount rate (%)					
	5	6	8	10	12	15
1	0.9524	0.9434	0.9259	0.9091	0.8929	0.8696
2	0.9070	0.8900	0.8573	0.8264	0.7972	0.7561
3	0.8638	0.8396	0.7938	0.7513	0.7118	0.6575
4	0.8227	0.7921	0.7350	0.6830	0.6355	0.5718
5	0.7835	0.7473	0.6806	0.6209	0.5674	0.4972
6	0.7462	0.7050	0.6302	0.5645	0.5066	0.4323
7	0.7107	0.6651	0.5835	0.5132	0.4523	0.3759
8	0.6768	0.6274	0.5403	0.4665	0.4039	0.3269
9	0.6446	0.5919	0.5002	0.4241	0.3606	0.2843
10	0.6139	0.5584	0.4632	0.3855	0.3220	0.2472
15	0.4810	0.4173	0.3152	0.2394	0.1827	0.1229
20	0.3769	0.3118	0.2145	0.1486	0.1037	0.0611
25	0.2953	0.2330	0.1460	0.0923	0.0588	0.0304

- Assuming a 10% discount rate, the NPV for project 1 would be calculated as below table.
- For discount factor of 0 year $=1/(1+0.10)^0$ similarly, for all years.
- The net present value for project 1, using a 10% discount rate, is therefore £618.
- Using a 10% discount rate, calculate the net present values for projects 2, 3 and 4 and decide which, on the basis of this, is the most beneficial to pursue.

Applying the discount factors to project 1



Year	Project 1 cash flow (£)	Discount factor @ 10%	Discounted cash flow (£)
0	–100,000	1.0000	–100,000
1	10,000	0.9091	9,091
2	10,000	0.8264	8,264
3	10,000	0.7513	7,513
4	20,000	0.6830	13,660
5	100,000	0.6209	62,090
Net Profit:	£50,000	NPV: £618	

Q. Calculate the internal rate of an investment of $1,36,000 which yields the following cash inflows

year	**Cash Inflows**
1	30,000
2	40,000
3	60,000
4	30,000
5	20,000

Solution:

Year	**Cash inflows**	**DF@10%**	**PV**	**DF@12%**	**PV**
0	(136,000)	1	(1,36,000)	1	(1,36,000)
1	30,000	0.909	27,270	0.893	26,790
2	40,000	0.826	33,040	0.797	31,880
3	60,000	0.751	45,060	0.712	42,720
4	30,000	0.683	20,490	0.636	19,080
5	20,000	0.621	12,420	0.567	11,340
NPV			2280	NPV	-4190

$IRR=L+[(N_l /N_l – N_h)*(H-L)]$

l=lower rate,h=higher rate

N_l=NPV of lower rate

N_h=NPV of higher rate

Calculate NPV

If negative, calculate NPV with lower rate you will gate +NPV

If positive, calculate NPV with higher rate you will gate -NPV

So,

IRR=10+[(2280/(2280+4190)*(12-10)]

=10+0.70

=10.70%

Risk Evaluation

- A risk is any condition or event whose currently not certain but if were to occur it would have a negative impact on the outcome of project.
- Every project involves risk. We have already noted that *project* risks, which prevent the project from being completed successfully, are different from the *business* risk that the delivered products are not profitable. Here we will discuss business risk.
- Risk evaluation process
  - Risk identification and ranking
  - Risk quantification

Risk Categorization

Project risks

– They threaten the project plan

– If they become real, it is likely that the project schedule will slip and that costs will increase

Technical risks

– They threaten the quality and timeliness of the software to be produced

– If they become real, implementation may become difficult or impossible

Business risks

– They threaten the viability of the software to be built

– If they become real, they jeopardize the project or the product

Sub-categories of Business risks

– Market risk – building an excellent product or system that no one really wants

– Strategic risk – building a product that no longer fits into the overall business strategy for the company

– Sales risk – building a product that the sales force doesn't understand how to sell

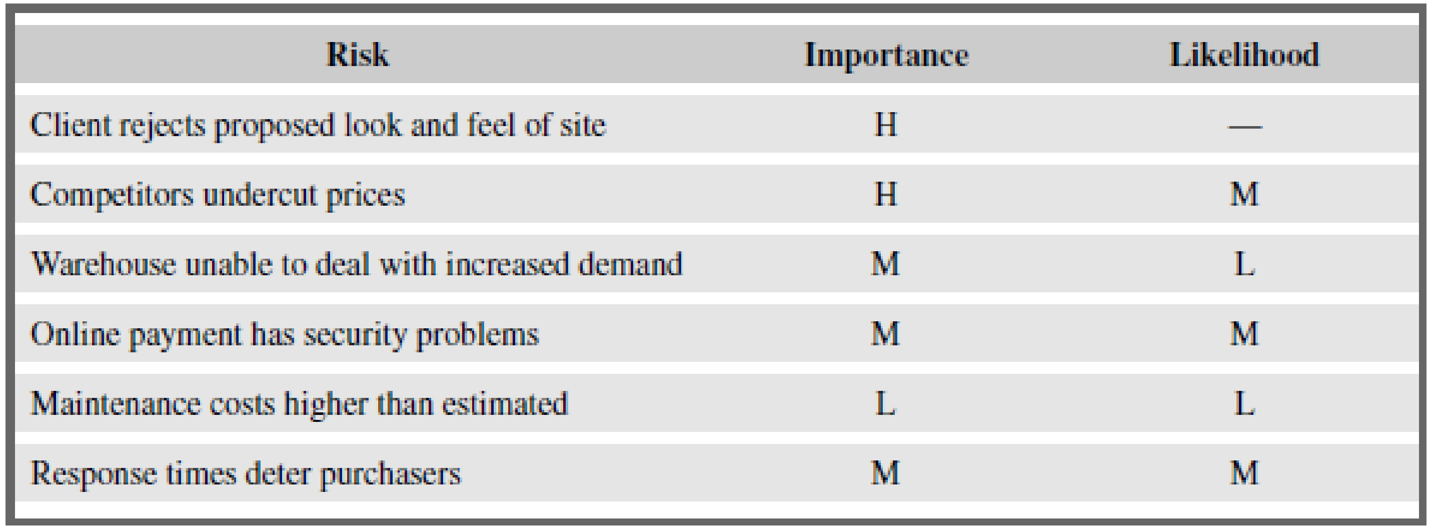
– Management risk – losing the support of senior management due to a change in focus or a change in people

– Budget risk – losing budgetary or personnel commitment

Risk identification and ranking:

- In any project evaluation we should identify the risks and quantify their effects.
- One approach is to construct a project risk matrix utilizing a checklist of possible risks and classifying risks according to their relative importance and likelihood.
- Following Table illustrates a basic project risk matrix listing some of the business risks for a project, with their importance and likelihood classified as high(H),medium(M),low(L)or exceedingly unlikely (—). Example1.

Table: A fragment of a basic project/business risk matrix for an e-commerce application



Risk	Importance	Likelihood
Client rejects proposed look and feel of site	H	—
Competitors undercut prices	H	M
Warehouse unable to deal with increased demand	M	L
Online payment has security problems	M	M
Maintenance costs higher than estimated	L	L
Response times deter purchasers	M	M

Example 2 :

Risk	**Details**
1	Unavailability of a specific domain experts.
2	Unavailability of key resources from customer side.
3	Risk of a delay in delivery of a specific hardware /software/tool.
4	Project completion within the given budget.
5	Demands of the over enthused customer with respect to performance.

- Rank risk to assess its likelihood of occurrence

✓ Scale of zero to 5

Example:

Unavailability of a specific domain experts.	Ranking 1
Unavailability of a specific domain experts	Importance - High, likelihood-Low

Risk Quantification:

– NPV

– Probable average expected value (leading to decision tree)

NPV method

– Based on single case flow statement

– Give numerical value, easy for comparing 2 projects

– Risk value of 2%, 4%, 6% (discount factor)

– What is required is the ‘relative judgment’ and not an ‘absolute estimate’

Decision tree

- Decision trees is a tool which provides evaluation of project’s expected outcomes and choosing between the alternative strategies.
- The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point.

- The expected value of each path is determined by the sum of the value of each possible outcome multiplied by its probability of occurrence.
- Advantage: It will give a precise idea of modeling and analyzing the problems in the project.

- As an example, say a successful company is considering when to replace its sales order processing system. The decision largely rests upon the rate at which its business expands – if its market share significantly increases the existing system might need to be replaced within two years.

- It is calculated that extending the existing system will have an NPV of £75,000, although if the market expands significantly, this will be turned into a loss with an NPV of –£1 00,000 due to lost revenue. If the market does expand, replacing the system now has an NPV of £250,000 due to the benefits of being able to handle increased sales and other benefits such as improved management information. If sales do not increase, however, the benefits will be severely reduced and the project will suffer a loss with an NPV of –£50,000.
- The company estimate the likelihood of the market increasing significantly at 20% and, hence, the probability that it will not increase as 80%. This scenario can be represented as a tree structure as shown in Figure below.

- The analysis of a decision tree consists of evaluating the expected benefit of taking each path from a decision point (denoted by D in Figure below).
- The expected value of each path is the sum of the value of each possible outcome multiplied by its probability of occurrence.
- The expected value of extending the system is therefore £40,000 (75,000 *0.8 – 100,000 *0.2) and the expected value of replacing the system £10,000 (250,000*0.2 –50,000*0.8). The company should therefore choose the option of extending the existing system.

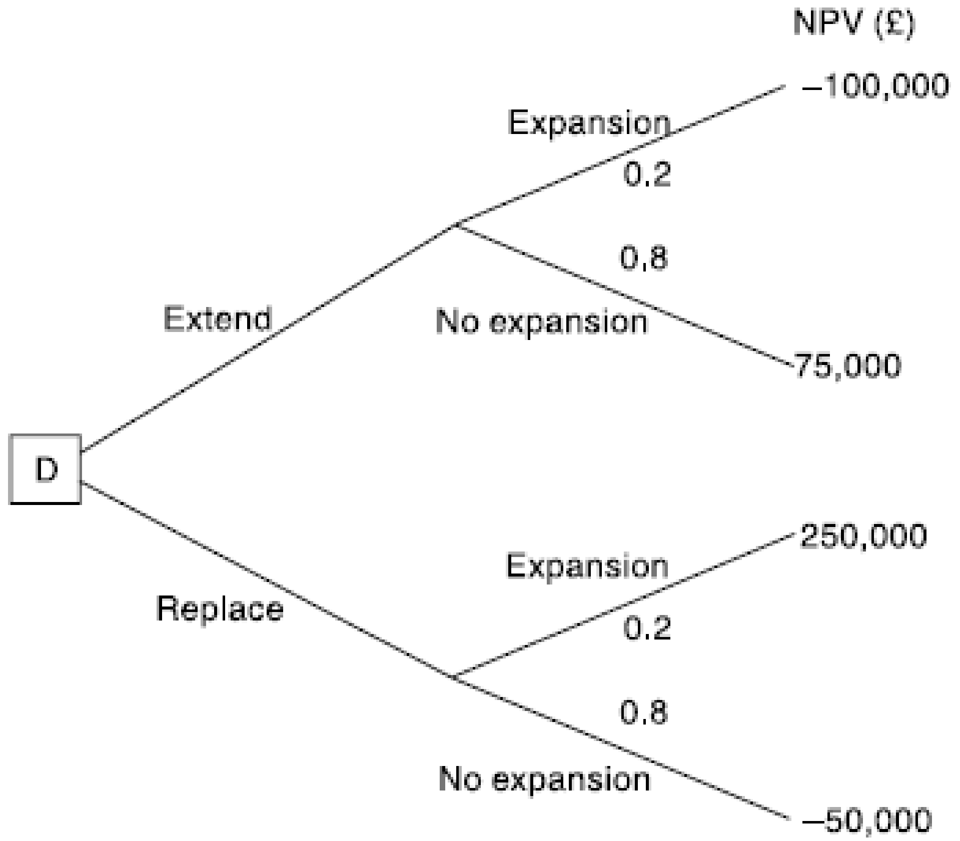


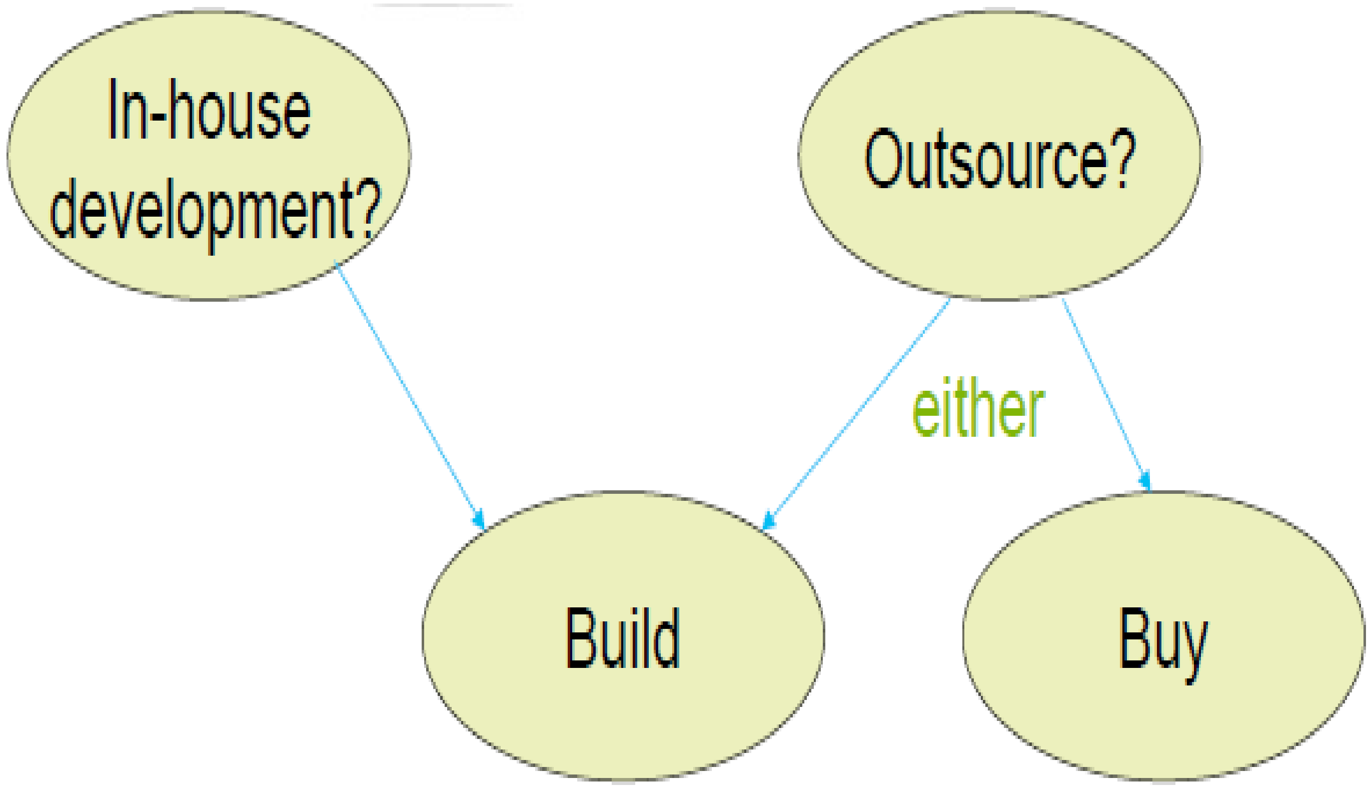
Fig: A decision tree

Selection of an Appropriate Project Approach

Contents

- Build or Buy?
- Methodologies and technologies of analyzing various project characteristics
- Software Process Models
- Managing Iterative Processes
- Selecting the most appropriate model.

Build or Buy?



Build or Buy?

In-house:

- The development of software in-house usually means that:
    - the developers and the users belong to the same organization;
    - the application will slot into a portfolio of existing computer-based systems;
    - the methodologies and technologies are largely dictated by organizational standards and policies, including the existing enterprise architecture.

Outsourced:

- means that the developers and the users of the software are in the different organization.
    - need for tailoring as different customers have different needs.
- **Off-the-shelf:** means a ready-made software product that is purchased.

- **Why Out Sourcing?**
  - Time is needed to develop the software.
  - Would often require the recruitment of new technical staff to do the job.
  - Usually, the new staff won’t be needed after the project is completed.
  - Sometimes due to the novelty of the project there may be lack of executives to lead the effort.

- **Why Buying?**
  - Whether in house or out sourced, software development is still involved.
  - The contracting company will not have completely developed software readily available to the client.
  - Considerable management effort is needed by the client to establish and manage the contract.
- **Advantages of off-the-shelf (OTS) software**
  - the supplier of the application can spread the cost of development over a large number of customers and thus the cost per customer should be reduced;
  - the software already exists and so
    - it can be examined and perhaps even trialed before acquisition,
    - there is no delay while the software is being built;
  - where lots of people have already used the software, most of the bugs are likely to have been reported and removed, leading to more reliable software.

- **Disadvantages of off-the-shelf software**
  - Customer will have same application as everyone else so there is no competitive advantage.
  - Customer may need to change the way they work in order to fit in with OTS application.
  - Customer does not own the code and cannot change it.
  - Danger of over-reliance on a single supplier.

Choosing Methodologies and Technologies

- The chosen methods and technologies will affect:
    - the training requirements for development staff;
    - the types of staff to be recruited;
    - the development environment – both hardware and software;
    - system maintenance arrangements.
- Some of the steps of project analysis;
    - **Identify project as either objective driven and product driven**
    - **Analyze other project characteristics**
    - **Identify high level project risks**
    - **Take into account user requirement concerning implementation**

Objective driven or product driven:

- A product-driven project creates products defined before the start of the project.
- An objective-driven project will often have come first which will have defined the general software solution that is to be implemented.
- Product-driven project:
  - a project will be to create a product.
  - The details of the product is provided by the client.
- Objective-driven project:
  - A project is to meet an objective.
  - The Client may have a problem and asks a specialist to recommend solutions.

Analyze other project characteristics:

- Will we implement a data-oriented or a process oriented system? *Data-oriented* systems generally mean information systems that will have a substantial database. *Process-oriented* systems refer to embedded control systems. It is not uncommon to have systems with elements of both.
- *Will the software to be produced be a general tool or application specific?* An example of a general tool would be a spreadsheet or a word processing package.
- *Are there specific tools available for implementing the particular type of application?*
    - Is the system knowledge-based?
    - Will the system to be produced makes heavy use of computer graphics?
- Is the system to be created safety critical? For instance, could a malfunction in the system endanger human life? If so, among other things, testing would become very important.
- *Is the system designed to carry out predefined services or to be engaging and entertaining?*
- *What is the nature of the hardware/software environment in which the system will operate?*

Identify high-level project risks:

- The more uncertainty in the project the more the risk that the project will be unsuccessful.
- Recognizing the area of uncertainty allows taking steps towards reducing its uncertainty.
- Uncertainty can be associated with the products, processes, or resources of a project.
- ***Product uncertainty,*** How well are the requirements understood? The users themselves could be uncertain about what a proposed information system is to do.
- ***Process uncertainty,*** The project under consideration might be the first where an organization is using an approach like extreme programming (XP) or a new application-building tool.
- ***Resource uncertainty,*** The main area of resource uncertainty is the availability of the staff with the right ability and experience. The larger the number of resources needed or the longer the duration of the project.

Take into account user requirements concerning implementation:

Select general life-cycle approach

- ***Control systems,*** A real-time system will need to be implemented using an appropriate methodology. Real-time systems that employ concurrent processing may have to use techniques such as Petri nets.
- ***Information systems,*** Similarly, an information system will need a methodology, such as SSADM or Information Engineering, that matches that type of environment.
- ***Availability of users,*** Where the software is for the general market rather than application and user specific, then a methodology which assumes that identifiable users exist who can be quizzed about their needs would have to be thought about with caution.
- ***Specialized techniques,*** For example, expert system shells and logic-based programming languages have been invented to expedite the development of *knowledge-based systems*.

- ***Hardware environment,*** The environment in which the system is to operate could put constraints on the way it is to be implemented.
- ***Safety-critical systems,*** Where safety and reliability are essential, this might justify the additional expense of a formal specification using a notation such as OCL.
- ***Imprecise requirements,*** Uncertainties or a *novel hardware/software platform* mean that a prototyping approach should be considered. If the environment in which the system is to be implemented is a rapidly changing one, then serious consideration would need to be given to *incremental delivery*. If the users have *uncertain objectives* in connection with the project, then a *soft systems* approach might be desirable.

Software Process and Process Model

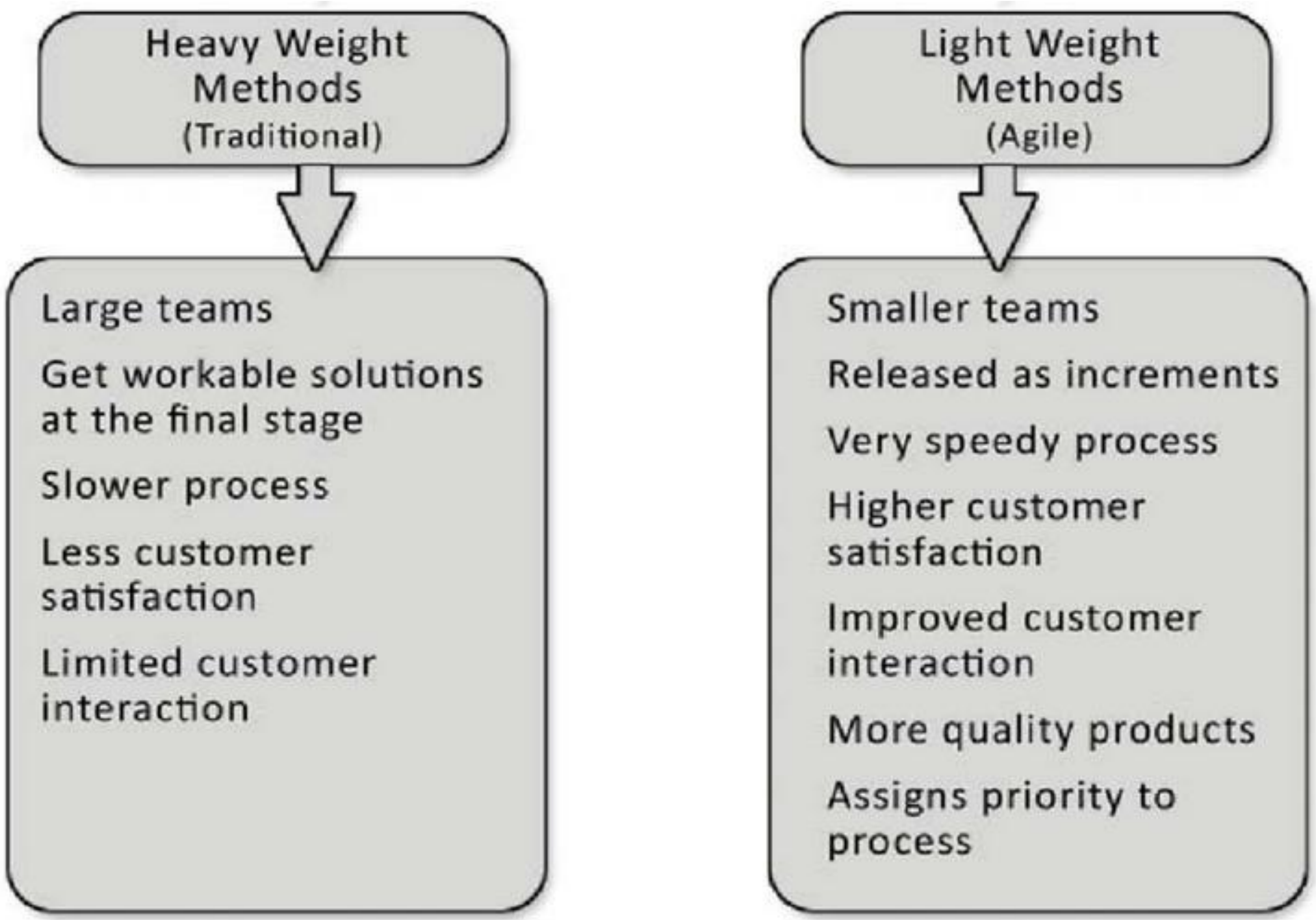
- A software product development process usually starts when a request for the product is received from the customer.
- A number of inter related activities have to be undertaken to create a final product. These activities can be organized in different ways and we call these process models.
- The software life cycle is also commonly referred to as Software Development Life Cycle (SDLC) and software process.
- A life cycle model (also called a process model) of a software product is a graphical or textual representation of its life cycle. Additionally, a process model may describe the details of various types of activities carried out during the different phases and the documents produced.

Structure vs. speed of delivery

Two competing pressures

- One is to make sure that the final product has a robust structure which will be able to meet evolving needs.
- Other is to get the job done as quickly and as cheaply as possible.

- Structured Approach:
    - Also called ***heavyweight*** approaches
    - Step-by-step methods where each step and intermediate product is carefully defined
    - Emphasis on getting quality right first time
    - More time consuming and expensive
    - End result is expected to be a less error prone and more maintainable final system.
- Speed of Delivery:
    - Also called ***lightweight*** approaches
    - Emphasis is on speed of delivery rather than documentation


Heavy Weight Methods (Traditional)
Large teams
Get workable solutions at the final stage
Slower process
Less customer satisfaction
Limited customer interaction
Light Weight Methods (Agile)
Smaller teams
Released as increments
Very speedy process
Higher customer satisfaction
Improved customer interaction
More quality products
Assigns priority to process

- Some process models are:
    - Waterfall model
    - Spiral model
    - Software prototype
    - Incremental delivery
    - Agile methods

Waterfall Model:

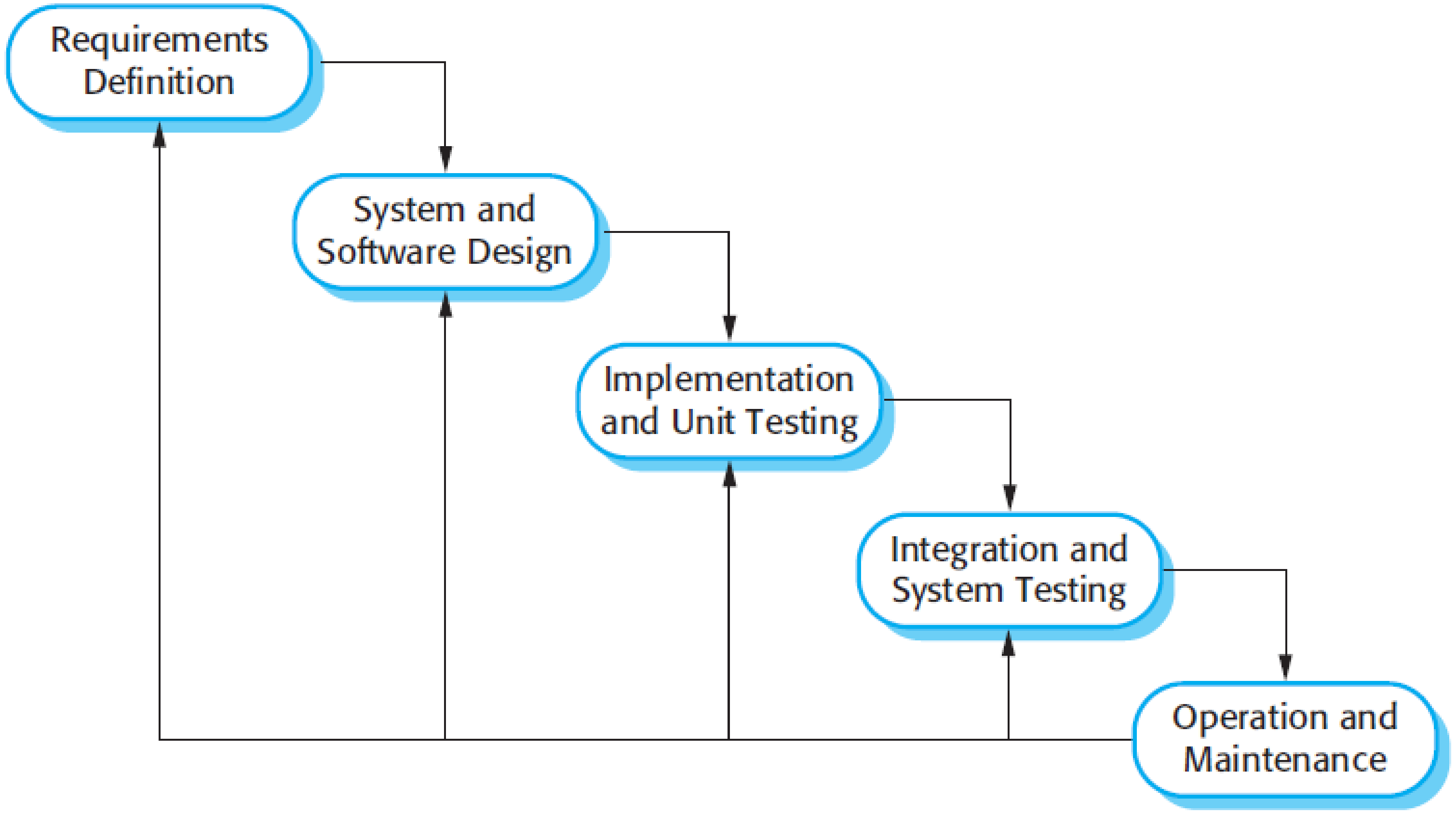

Requirements Definition
System and Software Design
Implementation and Unit Testing
Integration and System Testing
Operation and Maintenance

Fig: Waterfall Model

Waterfall Model:

- It is the Classical model of system development also known as one-shot or once-through model.
- There is a sequence of activities working from top to bottom.
- It has Limited scope of iteration.
- Is this a strength or a limitation??
    - This is a strength for the waterfall-model.
    - Because it is suitable for some projects especially for large projects, we want to avoid reworking tasks that are thought to be completed.
    - Reworking tasks could result in late delivery.

- The waterfall approach creates natural milestones at the end of each phase, where project progress can be reviewed.
- When the requirements are well defined and the development methods are well understood. It allows project completion time to be forecast with some confidence, allowing effective control of the project.
- However, when there is uncertainty, a more flexible and iterative approach is required.

Strengths-

  - Easy to manage due to the rigidity of the model, because each phase is specific
  - Reinforces good habits: define-before-design, design-before-code.

Weakness-

  - Unrealistic to expect accurate requirements so early in project.
  - Software is delivered late in project, delays discovery of serious errors

The Spiral Model:

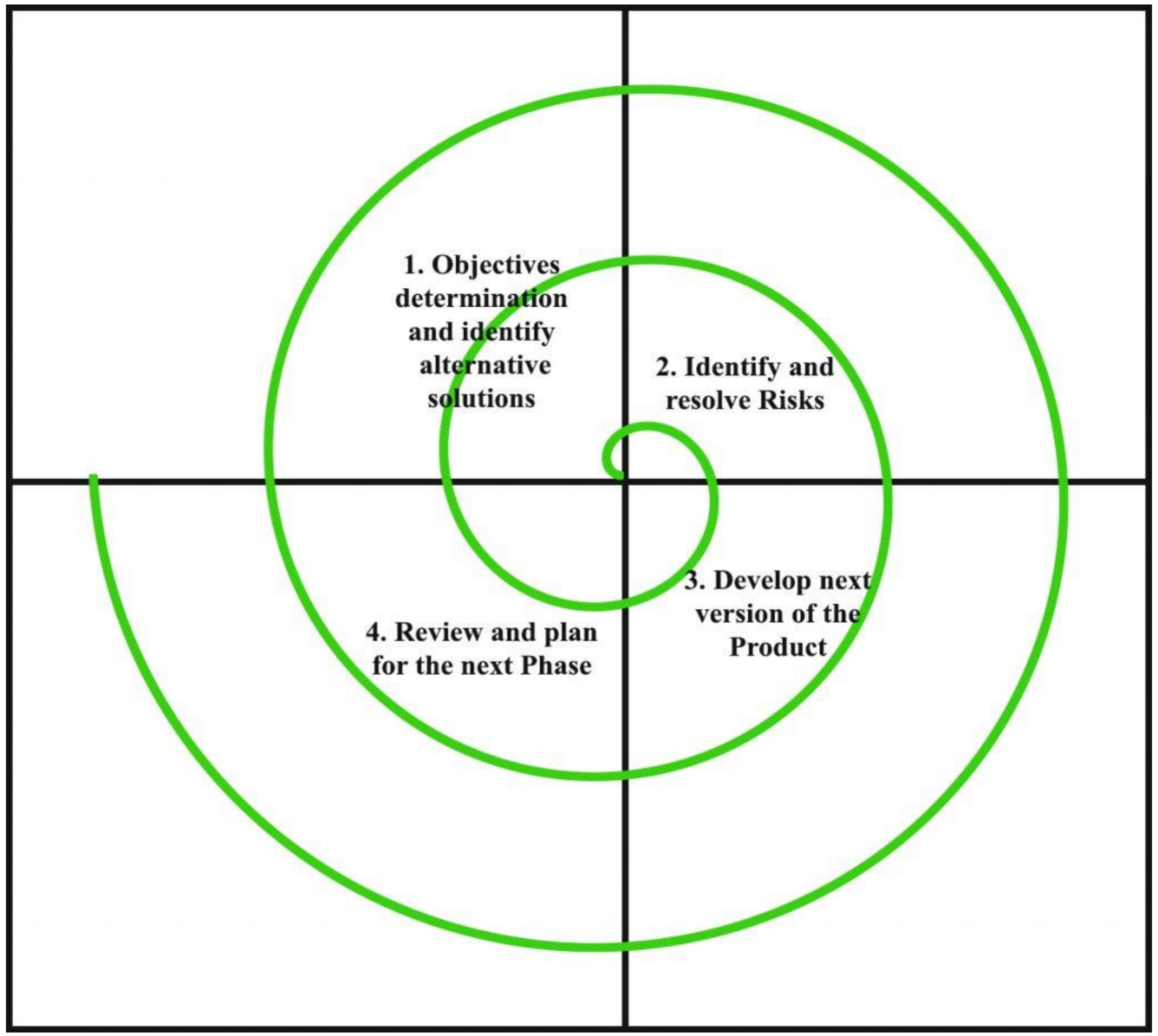

1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

Fig: Spiral Model

- Represented as a loop or a spiral where the system is considered in more detail.
- The distinguishing characteristic features of the spiral model are:
    - Incremental style of development
    - 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 spiral are not fixed, and vary from project to project providing higher flexibility.
- Each loop of the spiral is divided into four quadrants, indicating four stages in each phase
    - In 1st stage, one or more features of the project are analyzed.
    - In 2nd stage, risks in implementing those features are identified and resolved through prototyping.
    - In 3rd stage, the identified features are implemented.

- In 4th stage, the developed feature is reviewed and the features to be implemented in the next phase are identified.

Strengths-

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Software is produced early in the software life cycle.
- The model makes use of techniques like reuse, prototyping.

Weakness-

- The model is not suitable for small projects as cost of risk analysis may exceed the actual cost of the project.
- Different persons involved in the project may find it complex to use

Software Prototyping:

- Prototype is a working model of one or more aspects of the projected system.
- The prototype is constructed and tested, quickly and inexpensively to test assumptions.
- Goals,
  - Gain knowledge
  - Reduce risk and uncertainty
  - Verify a design or implementation approach
- Prototypes can be classified as,
  - **Throwaway:** The prototype tests out some ideas and is then discarded when the true development of the operational system is commenced. The prototype could be developed using a different software or hardware environment. For example, a desktop application builder could be used to evolve an acceptable user interface. A procedural programming language is then used for the final system where machine-efficiency is important.

- **Evolutionary:** The prototype is developed and modified until it is finally in a state where it can become the operational system. In this case the standards that are used to develop the software have to be carefully considered.

Strengths-

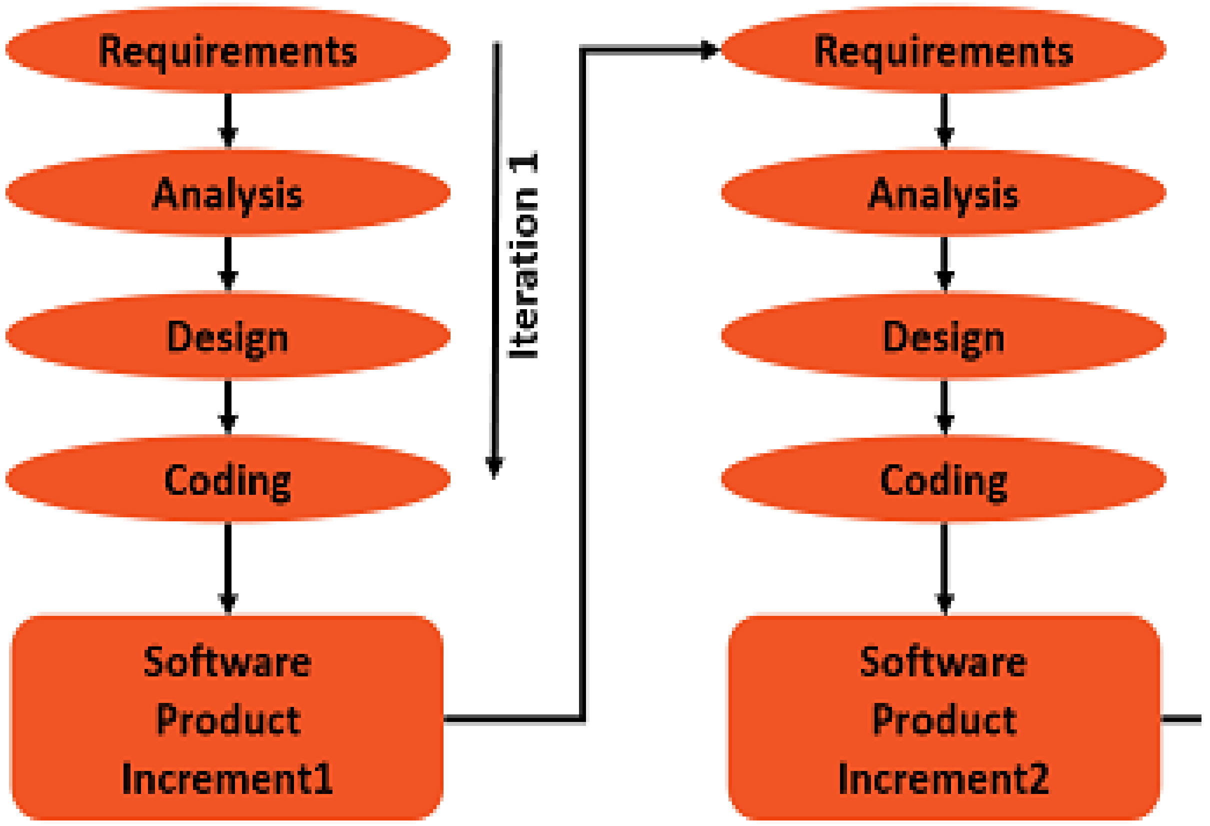
  - Learning by doing.
  - Improved communication.
  - Improved user involvement.
  - Clarification of partially-known requirements.
  - Reduced need for documentation.
  - Feature constraint.

- **Weakness-** Users sometimes misunderstand the role of the prototype.
  - Lack of project standards possible, lack of control
  - Additional expense.
  - Close proximity of developers.

- It would be unusual for the whole application to be prototyped.
- It usually simulates only some aspects of the target application.
- For example there might be
  - Mockups
  - Simulated interaction
  - Partial working model
    - Vertical
    - Horizontal

Incremental Model:

- This approach breaks the system into small components.
- These components are then implemented and delivered in sequence.
- Every delivered component provides some more functionality to the user as compared to previous one.
- The concept of time boxing is attached with an incremental approach.
- Scope of deliverables for an increment is rigidly constrained by an agreed deadline.
- This deadline has to be met, even at the expense dropping some of the planned functionality.
- Omitted features can be transferred to later increments.


Requirements
Analysis
Design
Coding
Software Product Increment1
Iteration 1
Requirements
Analysis
Design
Coding
Software Product Increment2

Strengths-

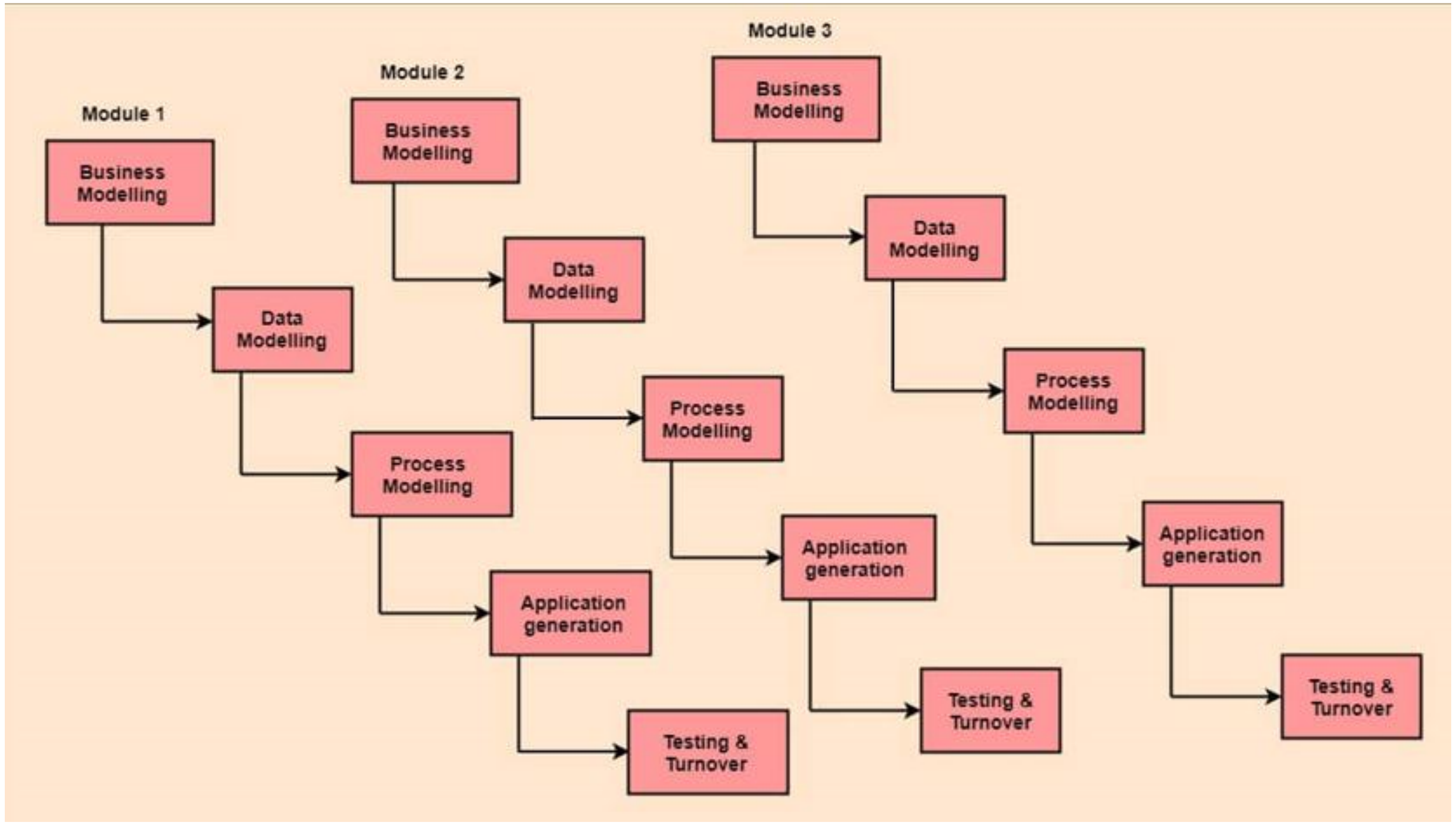
- The feedback from early increments improves the later stages
- Users get benefits earlier than with a conventional approach
- Early delivery of some useful components improves cash flows
- Smaller sub projects are easy to control and manage
- Less scope creep

Weakness-

- Total cost is higher
- Requires heavy documentation
- Software breakage
- Software developers may be more productive working on one large system than on a series of smaller ones.

Rapid Application Development (RAD):

- RAD model is also sometimes referred to as the rapid prototyping(-working model that is functionally equivalent to a component of the product) model.
- RAD model, the functional modules are developed in parallel as prototypes and are integrated to make the complete product for faster product delivery. Since there is no detailed preplanning, it makes it easier to incorporate the changes within the development process.
- This model has the features of both the prototyping and the incremental delivery models.
- The major aims of the RAD model are as follows:
    - to decrease the time taken and the cost incurred to develop software systems
    - to limit the costs of accommodating change requests by incorporating them as early as possible before large investments have been made on development and testing.


Module 1
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 2
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover
Module 3
Business Modelling
Data Modelling
Process Modelling
Application generation
Testing & Turnover

When to use RAD Model?

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Advantage of RAD Model

- This model is flexible for change.
- changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

Atern / Dynamic Systems Development Method:

- In the United Kingdom, SSADM (Structured Systems Analysis and Design Method) has until recently been a predominant methodology, re-badged as Atern.
- Eight core Atern principles have been enunciated.
  - Focus on business need
  - Deliver on time
  - Collaborate
  - Never compromise quality
  - Develop iteratively
  - Build incrementally from firm foundations
  - Communicate continuously
  - Demonstrate control

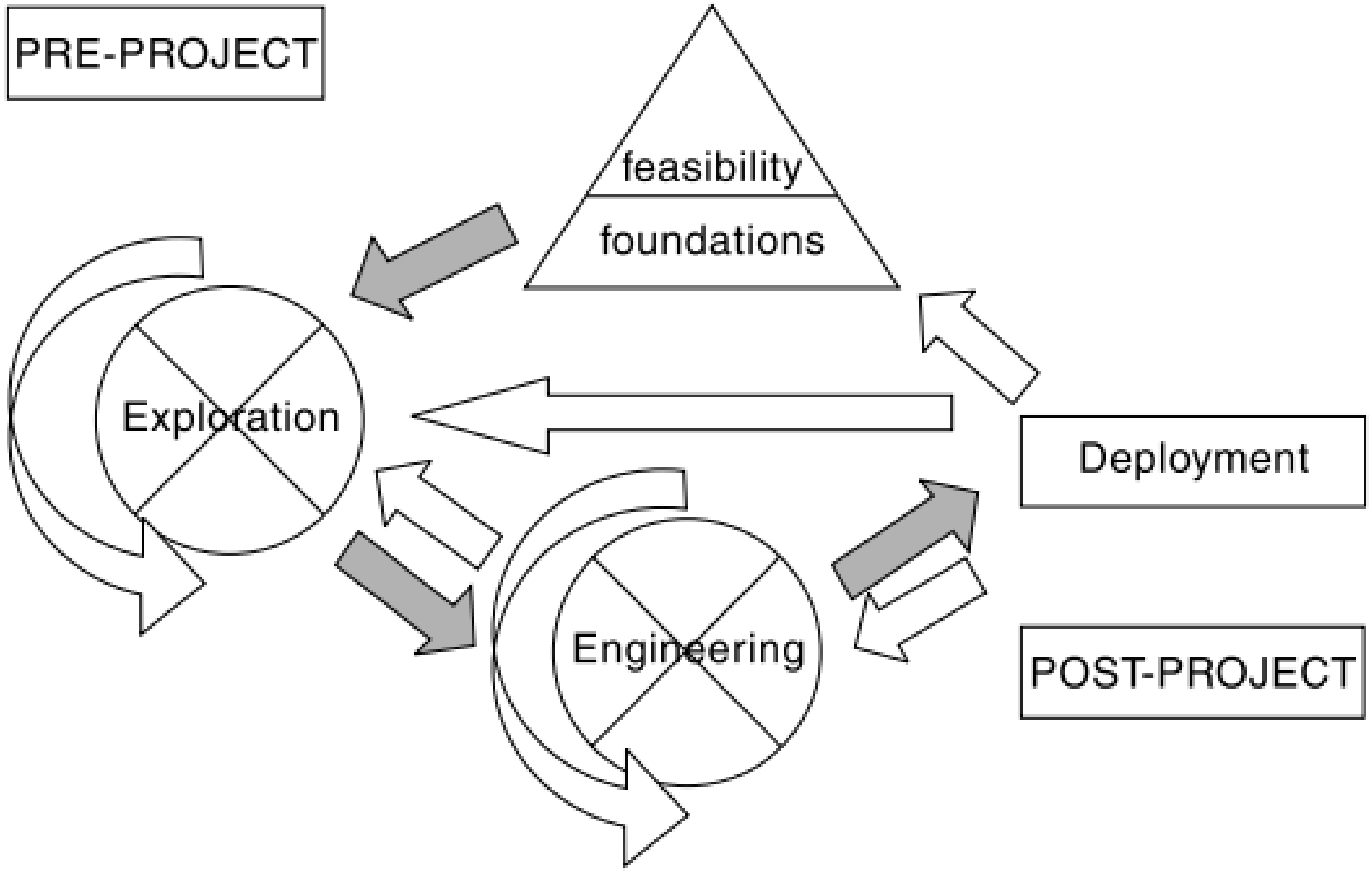


Fig: Atern process model

- ***Feasibility/foundation,*** Among the activities undertaken here is derivation of a business case and general outlines of the proposed architecture of the system to be developed.
- ***Exploration cycle,*** This investigates the business requirements. These requirements are translated into a viable design for the application. This could be an iterative process that could involve the creation of exploratory prototypes. A large project could be decomposed into smaller increments to assist the design process.
- ***Engineering cycle,*** This takes the design generated in the exploration cycle and converts it into usable components of the final system that will be used operationally. Once again this could be done using incremental and evolutionary techniques.
- ***Deployment,*** This gets the application created in the engineering cycle into actual operational use.

Agile Methods:

- Why need Agile Methods?
  - Difficult to accommodate change requests in heavyweight processes.
  - Heavyweight processes are documentation driven and too rigid.
- Agile methods are designed to overcome the disadvantages of the heavy weight development approaches.
- The most glaring changes advocated by agile technique are iterative development and an enhanced interaction with customers.
- The most widely-used Agile methodologies include:
    - Agile Scrum Methodology
    - Lean Software Development
    - Extreme Programming (XP)

    - Kanban
    - Crystal
    - Dynamic Systems Development Method (DSDM)
    - Feature Driven Development (FDD)
- Central principles of agile methods are,
  - Incremental delivery after each time box.
  - Agile model emphasizes face-to-face communication over written documents. Team size is deliberately kept small (5–9 people) to help the team members effectively communicate with each other and collaborate.
  - An agile project usually includes a customer representative in the team.
  - Minimal documentation
  - Agile development projects usually deploy pair programming. In this approach, two programmers work together at one work station.

Extreme Programming (XP):

- Extreme Programming is based on the following values −
  - ***Communication,*** Everyone on a team works jointly at every stage of the project.
  - ***Simplicity,*** Developers strive to write simple code bringing more value to a product, as it saves time and efforts.
  - ***Feedback,*** Team members deliver software frequently, get feedback about it, and improve a product according to the new requirements.
  - ***Respect,*** Every person assigned to a project contributes to a common goal.
  - ***Courage,*** Programmers objectively evaluate their own results without making excuses and are always ready to respond to changes.
- Some core practices of extreme programming are
  - The planning exercise
  - Small releases

- Testing, Refactoring, Pair programming
- Collective ownership
- Forty-hour weeks
- On-site customers

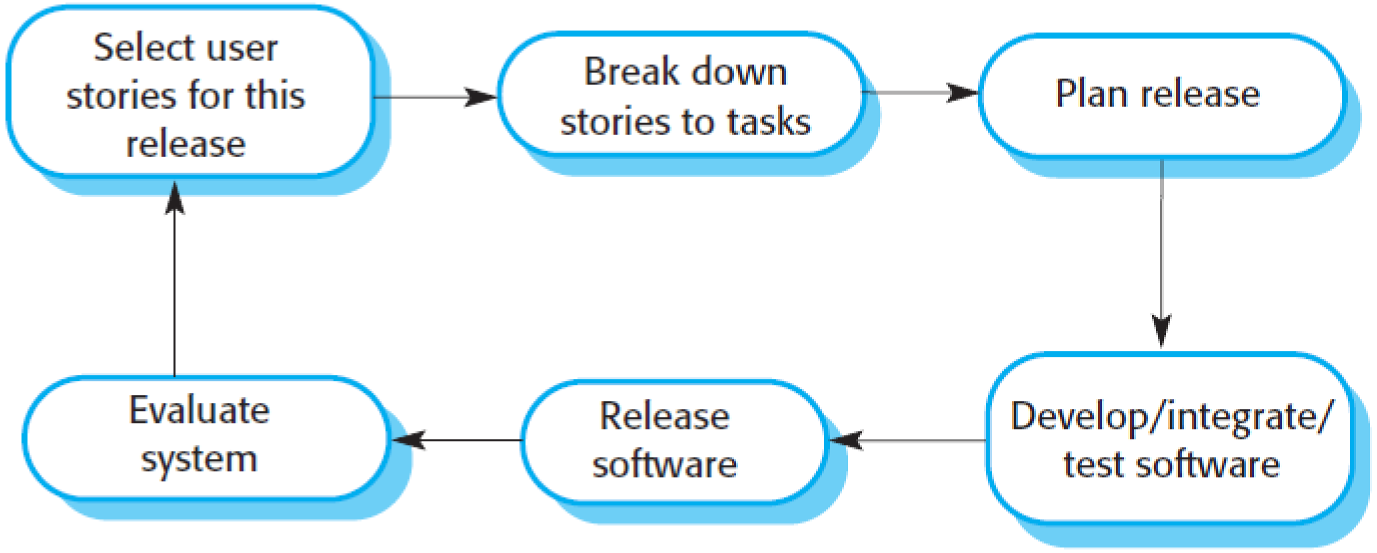

Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Fig: The XP release cycle

Scrum:

- The Scrum approach is a general agile method but its focus is on managing iterative development rather than specific agile practices.
- There are three phases in Scrum:
    - The initial phase is an outline planning phase where you establish the general objectives for the project and design the software architecture.
    - This is followed by a series of **sprint** cycles, where each cycle develops an increment of the system.
    - The project closure phase wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project.
- Sprints are fixed length, normally 2-4 weeks. They correspond to the development of a release of the system in XP.

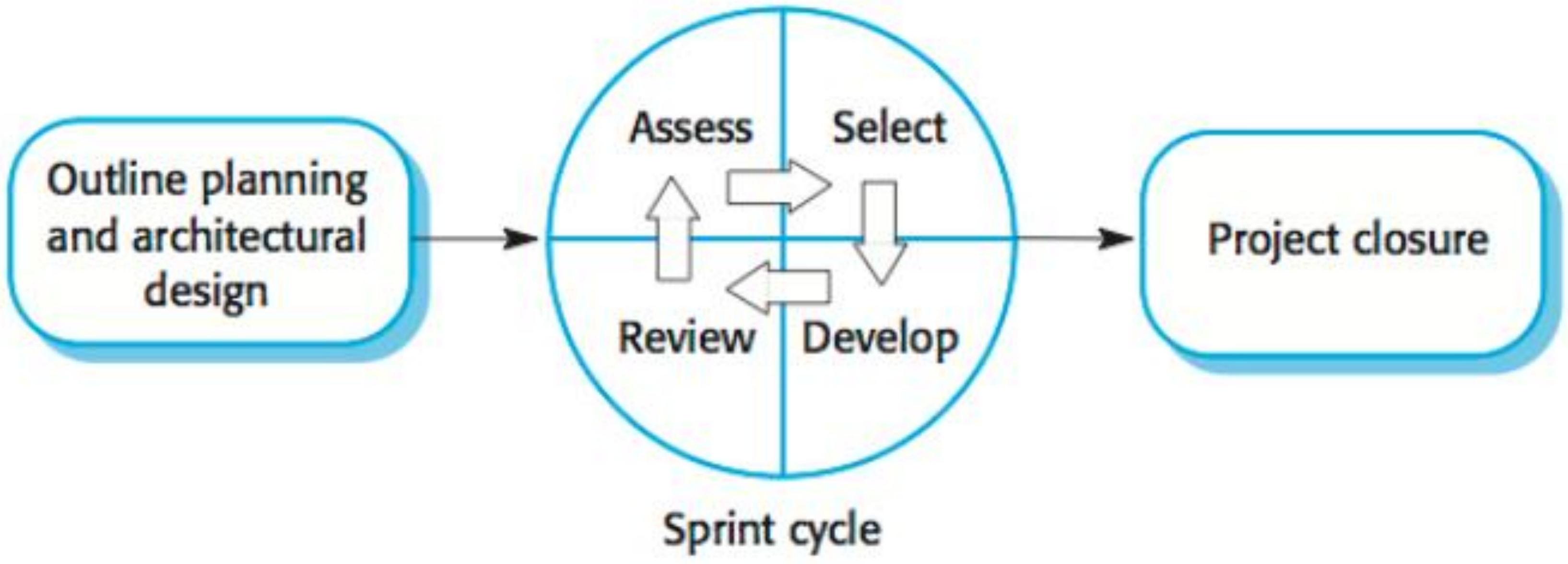

Outline planning and architectural design
Assess
Select
Review
Develop
Sprint cycle
Project closure

Fig: Scrum

The Sprint cycle

- The starting point for planning is the **product backlog**, which is the list of work to be done on the project. The selection phase involves all of the project team who work with the customer (**product owner /idea people**) to select the features and functionality to be developed during the sprint.
- During this stage the team is relatively isolated from the product owner and the organization, with all communications channeled through the **ScrumMaster.**
- The role of the ScrumMaster Master is responsible for setting up the team, sprint meeting and removes obstacles to progress. At the end of the sprint the work done is reviewed and presented to stakeholders (including the product owner).

- **Velocity** is calculated during the **sprint review**; it provides an estimate of how much product backlog the team can cover in a single sprint.
- Team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring and improving performance.
- The next sprint cycle then begins.

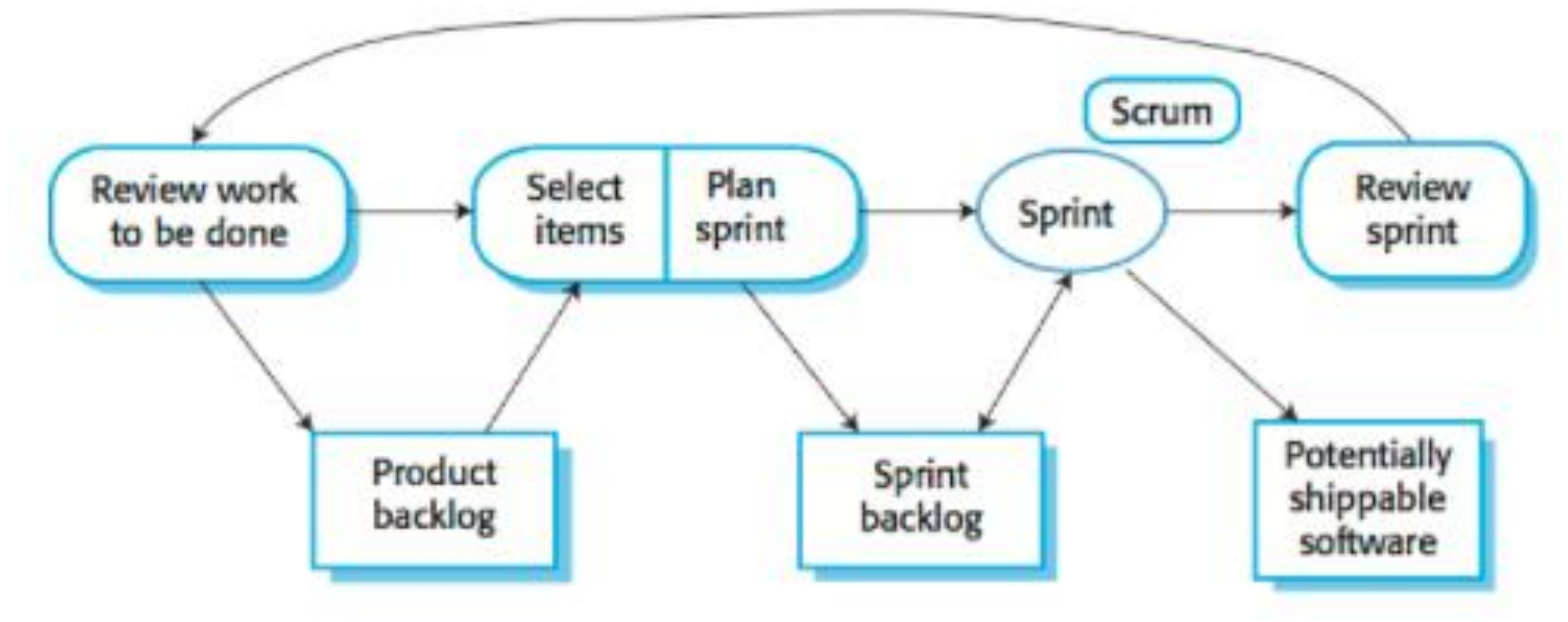


Fig: The Scrum software sprint cycle

- The ScrumMaster is a facilitator who arranges short daily meetings (**daily scrums**), tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with the product owner and management outside of the team.
- The whole team attends daily scrums where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.

Advantages of scrum include:

- The product is broken down into a set of **manageable and understandable chunks**.
- Unstable requirements do not hold up **progress**.
- The whole team have visibility of everything and consequently **team communication** is improved.
- Customers see **on-time delivery** of increments and gain feedback on how the product works.
- **Trust** between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Lean Software Development:

- **Lean Software Development (LSD)** is an agile framework used to streamline and optimize the software development process.
- It focuses on delivering high-quality software in a timely and efficient manner, ensuring that the development process is continually improved.

The 7 Key Lean Software Development Principles

1. Eliminate Waste
2. Amplify Learning
3. Defer Commitment
4. Deliver as fast as possible
5. Build Quality In
6. Respect for People
7. Optimize the whole

Eliminate Waste

- This principle focuses on identifying and eliminating all forms of waste in the software development process.
- Waste refers to any activity or resource that doesn’t add value to the final product.
- Examples,
    - **Over-production (Unnecessary features)**
    - **Transportation (Constant work handoffs between developers)**
    - **Inventory (Partially done user stories)**
    - **Over-processing (Relearning)**
    - **Waiting (Delays due to bugs or other process impediments)**
    - **Motion (Context switching between coding work)**
    - **Defects (Faulty code which requires rework)**

Build in Quality

- Instead of relying solely on testing and bug fixing, lean encourages building quality into your product from the beginning.
- You achieve this quality through practices like ,
    - **Pair programming** - writing code jointly with another developer to catch errors early.
    - **Test-driven development** - producing testing criteria before the actual code.
    - **Incremental development and constant feedback**
    - **Limiting work in progress** - reducing context switching to enhance team's focus.
    - **Automation** - automating testing processes where possible.

Respect People

- This principle emphasizes the value of individuals within the development team.
- It encourages open communication, trust, and mutual respect among team members.

Fast Delivery

- This Lean development principle is related to enhancing the speed to market by releasing **MVP**s (Minimum Viable Products) rather than fully complete solutions. Software teams should release a feature (or an entire product) that's "just good enough" rather than trying to get it perfect from the first shot.
- This approach allows for faster validation of ideas and provides opportunities for continuous improvement based on real-world usage.

Amplify Learning

- Lean software development emphasizes the importance of knowledge sharing and learning within a team.
- This principle encourages cross-training, code reviews and documentation to ensure that knowledge isn’t siloed or isolated within individual team members.

- Sharing knowledge improves collaboration, reduces dependencies, and enhances the team's overall capabilities.

Defer Commitment

- Deferring commitment is about waiting until the last responsible moment to make critical and allowed decisions.
- This principle allows teams to gather more information, reduce uncertainty, and adapt to changing requirements or market conditions before making decisions.
- By deferring commitment, teams can consistently make more informed choices, reducing the risk of costly mistakes.

Optimize the Whole

- Optimizing the entire value stream involves looking at the end-to-end process of software development.
- Instead of focusing on individual components or stages, lean encourages teams to identify bottlenecks, constraints, and areas of improvement across the entire workflow.

Managing Iterative Processes

- Booch suggests that there are two levels of development: the macro process and the micro process.
- **Macro process**
  - The macro process is closely related to the waterfall process model.
  - a range of activities carried out by a variety of specialist groups has to be coordinated.
  - We need to have some dates when we know that major activities will be finished so that we know when we will need to bring in staff to work on subsequent activities.
- **Micro process**
  - Within this macro process there will be micro process activities which might involve iterative working.
  - With iterative micro processes, the use of time-boxes is needed to control at the macro level.

- Following figure illustrates how a sequential macro process can be imposed on a number of iterative sub-processes.

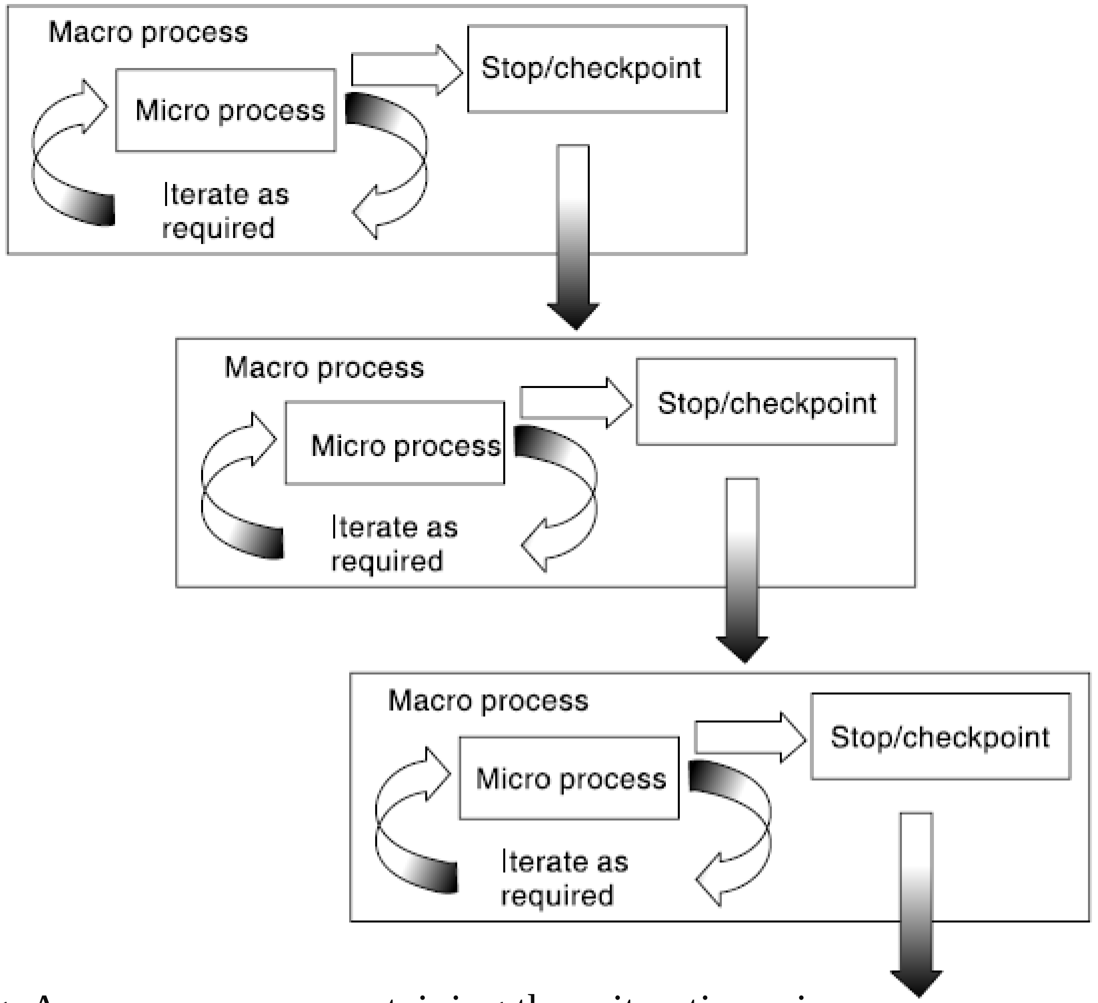


Fig: A macro process containing three iterative micro processes

- There are cases where the macro process itself can be iterative. It might be that a prototype for a complex technical system is produced in two or three successive versions, each taking several months to create and evaluate.

Selecting the most Appropriate process model

- Whenever uncertainty is high, an evolutionary approach needs to be favored e.g. when user requirements are not clear.
- When requirements are relatively certain, but there are many complexities, then an incremental approach is favored.
- Evolutionary and incremental approaches are favored in the case of tight deadlines.
- For development of simple and well-understood applications, waterfall should be preferred.
- If the development team is entirely novice, then even the development of simple applications require an incremental and prototyping approach.
- The spiral model would be appropriate, if the project is large and it is not possible to anticipate the project risks at the start of the project.
- If the customer is unsure about some features of the software to be developed, then an evolutionary or agile model would be the best choice.