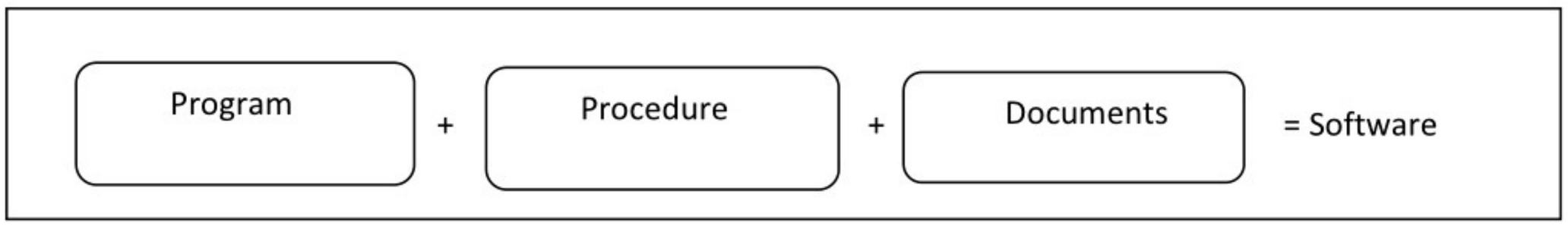
Introduction of software

- સોફ્ટવેર એ પ્રોગ્રામ નો સમૂહ છે, જે well defined કાર્ય કરવા માટે design કરવામાં આવ્યો છે.
- Software મા કોમ્પ્યુટર પ્રોગ્રામર દ્વારા લખવામાં આવતા કોડ નો સમાવેશ હોય છે.



Characteristics of software

1. **Software is engineered, not manufactured like hardware**
   - આ case મા સારી design સારો software બતાવે છે.hardware ને બનાવવામાં quality problem થાય છે.જે software માં થતો નથી.
   - જો એકવાર hardware બની જાય પછી તેને સુધારવો, તેને બદલવો શક્ય નથી. પરંતુ software મા સહેલાઇથી થઈ શકે છે.
   - Hardware ના કિસ્સામાં તેની અલગ અલગ copy બનાવવી costly પડે છે તેમાં raw material વપરાય છે તેના લીધે કોપી બનાવવી costly પડે છે પરંતુ software ના કિસ્સામાં તમે વધુ copies બનાવી શકો છો.
2. **Software does not wear out**
   - Fig. 1 માં દર્શાવ્યા પ્રમાણેના bathtub curve મા કાર્ય કરવાના time એ hardware fail થાય છે.
   - પહેલા phase મા failure rate વધુ હોય છે.પરંતુ તેની ચકાસણી અને ભૂલો સુધાર્યા પછી failure નો દર ઘટે છે અને અમુક time પર સ્થિર થઈ જાય છે.
   - Second phase hardware ની life નો ઉપયોગી phase હોય છે. જ્યાં failure નો દર ઓછો અને સ્થિર હોય છે.અમુક time પસાર થાય પછી હાર્ડવેર component, dust, ધ્રૂજારી, misuse અને તાપમાન અને અમુક ઘણા બીજા પર્યાવરની અસર માથી પસાર થાય છે. તેથી failure rate વધે છે.

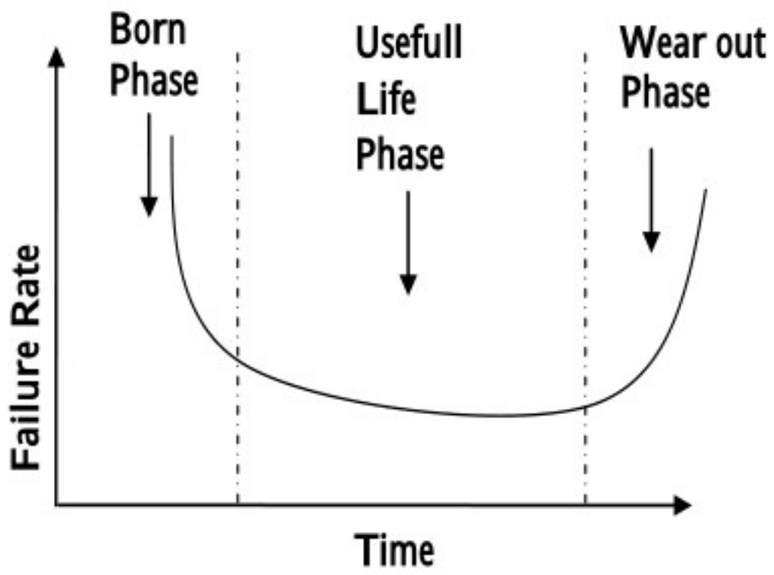


Figure 1

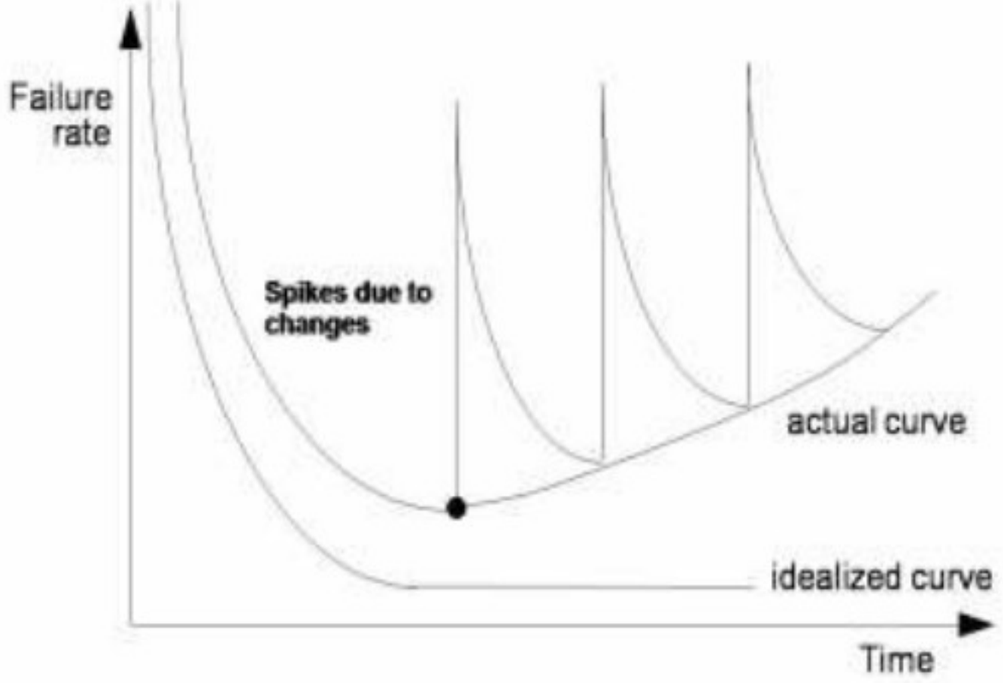


Figure 2

- શોધાયેલી ના હોય એવી ભૂલો failure ના દરને પ્રોગ્રામ ની life માં વધારે છે. જ્યારે તે સુધરે છે. ત્યારે curve flat થઈ જાય છે. તેને idealized curve કહેવાય છે. જે figure 2 માં બતાવેલું છે.

- Software ના જીવન દરમિયાન તે બદલે છે. તે સુધરીને અમુક નવી ભૂલો ઓળખાય છે. તેથી failure rate નો curve spike થઈ જાય છે. જે actual curve figure 2માં દર્શાવેલો છે.
- જ્યારે hardware component ખરાબ થાય છે. ત્યારે તેને spare part થી બદલાવવામાં આવે છે. જ્યારે software માં કોઈ spare part થી બદલી શકતા નથી. તેમાં જે પ્રોડક્ટ હોય તે પ્રોડક્ટ ની અંદર જ બદલવામાં આવે છે.

3. **Software gives component-based construction, it gives reusability of components**
   - Software એ component design કરીને બનાવવામાં આવ્યા હોય છે. જેથી તેનો ઉપયોગ ફરીથી ઘણા program મા થાય છે.
   - Graphical user interface (GUI) એ Reusable component થી બનેલ છે.તેથી તે ગ્રાફિકલ window અને animated menu બનાવે છે.

Attributes of good software

- **Usability:** Users સિસ્ટમ ને easily સમજી શકવા જોઈ અને ઉપયોગ કરી શકવા જોઈ.
- **Efficiency:** Software પ્રોસેસર, મેમરી અને ડિસ્ક સ્પેસનો ઉપયોગ ઓછો કરવો જોઈએ. જેમ less resource use એમ સોફ્ટવર better.
- **Reliability:** એક સિસ્ટમ કે જે function stable રીતે કરી શકે તે important છે.Failure બની સકે તેટલું rare હોવું જોઈએ.
- **Integrity:** Security આપવી જોઈએ account મા. અને સિસ્ટમ માં attackers access unauthorized resources આપવા જોઈએ નહિ.અને data validation છે તે important હોવું જોઈ એટલે bed data system માં સેવ થઈ સકે નહિ.
- **Adaptability:** એક એવી સિસ્ટમ કે જેનો ઉપયોગ વિવિધ પરિસ્થિતિઓમાં ફેરફાર કર્યા વિના કરી શકાય છે.
- **Accuracy:** સિસ્ટમ નું આઉટપુટની accuracy good હોવી જોઈ. Accuracy માપે છે કે સોફ્ટવેર user's માટે proper output આપે છે.
- **Maintainability:** Existing system ની અંદર changes ખુબ important છે. આપડે જેટલા easy changes લાવી શકીએ તે better કેહેવાય.
- **Portability:** એવી સિસ્ટમ કે જે બધા environments ની અંદર work કરી સકે. It's originally designed makes it good.
- **Reusability**: સોફ્ટવર ના piece નો જેમ reuses કરી શકીએ તેમ સોફ્ટવર good. Reuses કરવાનો અર્થ એ છે કે આપણે તેનો શરૂઆતથી ફરીથી ઉપયોગ કરવાની જરૂર નથી.
- **Readability:** કોડ read કરવામાં easy અને જેમ કોડ સમજવામાં easy તેમ કોડ માં changes કરવામાં પણ easy રહે છે અને less error આવે છે.
- **Testability:** સિસ્ટમ ને testable બનાવી તે critical છે.જેમ આપણો કોડ easy તેમ unit tests તે good.
- **Understandability:** global view બાજુ થી અને કોડે લેવલ બંને બાજુ થી સિસ્ટમને સમજવાની important છે.

Introduction of software engineering

- સોફ્ટવેર એન્જિનિયર એ કોમ્પ્યુટર સાયન્સમાં એક શાખા છે. જે developing application સાથે deal કરે છે. તે designing implementing અને modifying મારફતે system બનાવવા માટે ના ટેકનીકલ ભાગને કવર કરે છે.
- તે આવા પ્રોગ્રામિંગ ટીમ અને બજેટિંગ જેવા સોફ્ટવેર મેનેજમેન્ટના મુદ્દાઓને કવર કરે છે.
- સોફ્ટવેર એન્જિનિયરિંગનાં વ્યવસ્થિત ડિઝાઈન અને સોફ્ટવેર પ્રોડક્ટ ડેવલોપમેન્ટ ને સોફ્ટવેર મેનેજમેન્ટ તરીકે define કરી શકાય છે.

Need of software engineering

- સોફ્ટવેર એન્જિનિયરીંગ મેથોડોલોજી framework દ્વારા સોફ્ટવેર developing માટે engineer ને guide કરે છે.આ framework સોફ્ટવેર development ના અલગ અલગ phase જેવા કે requirement analysis, designing, implementation, testing અને maintenance ને દર્શાવે છે.
- Software engineering કેવી રીતે સોફ્ટવેર બનેલો છે તેમની શું જરૂરિયાતો છે અને શું અલગ અલગ પ્રકારના phases સોફ્ટવેર બનાવવા જરૂરી છે તે દર્શાવે છે.
- Software engineering નો મુખ્ય ધ્યેય process models આપવાનો કે છે સારી રીતે documented software બનાવે છે
- Process model નો ઉપયોગ કરીને આપણે અગાઉથી કેટલો ટાઈમ, ખર્ચ અને પ્રયત્નનો final પ્રોડક્ટ બનાવવા માટે જરૂરી છે તે નક્કી કરે છે.
- Software development life cycle (SDLC) એ એક framework જે સોફ્ટવેર ના વિકાસ ના દરેક કાર્ય ને વ્યાખ્યાયિત કરે છે.
- દરેક process model વિવિધ સ્થિતિ અને પરિસ્થિતિ માં વપરાય છે.આ process model software ને ક્યા stage માં ચલાવવો જોઈએ એ નક્કી કરે છે.

Software engineering: a layered technology

- Software engineering એ layered technology છે.

A Quality Focus

- Software engineering નો મુખ્ય ધ્યેય quality વાળો software બનાવવા માટે હોય છે.
- product ની quality તે engineer એ બનાવેલી product ની લાક્ષણિકતા ને દર્શાવે છે
- Software development માં output કાર્યો અને લક્ષણો માટે ના જરૂરિયાત model સ્પષ્ટ મળે છે.

Process

- base layer એ process layer છે
- Software process એ activities નો set છે. અને તે ઈચ્છિત પરિણામ ઉત્પન્ન કરવા માટે perform કરવામાં આવે છે.
- આ layer નો મુખ્ય ઉદ્દેશ software ને time પર પહોચાડવાનો છે.

Method

- Process layer પછીનું layer method છે. તે software product કેવી રીતે બનાવવી તેનું વર્ણન કરે છે.
- Method માં અલગ અલગ કાર્યો જેવા કે user Communication, તેની જરૂરિયાતો, design, coding, testing, અને Maintenance થાય છે.
- Method એ software બનાવવા માટેનો ચોક્કસ way આપે છે.
- Method એ યોગ્ય રીતે process ચલાવવા માટેની એક રીત છે

Tools

- Software engineering ના સાધનો process અને method માટે સ્વચાલિત અને અર્ધ સ્વયંચાલિત support પૂરો પાડે છે. જ્યારે સાધનો integrated હોય ત્યારે એક સાધન દ્વારા બનાવવામાં આવેલી માહિતી દ્વારા વાપરી શકાય છે.
- computer સહાયક software engineering, software, hardware અને software engineering database જે software engineering પર્યાવરણ બનાવવા માટે જોડાયેલું છે.

Various Myths Associated with Software

- અમુક time એ સોફ્ટવેર માટે મેનેજમેન્ટ ના મનમાં,user ના મન માં તથા ડેવલોપર ના મનમાં મૂંઝવણ અને માન્યતાઓ હોય છે.

Management Myths

Myths	Reality
✓ સંસ્થાના સભ્યો તમામ પ્રક્રિયાઓ સિદ્ધાંતો અને standard જાણે છે.	✓ ડેવલોપર standard વિશે જાણતા નથી. ✓ Standard અધૂરા અને જૂના છે. ✓ Developer standard ને follow કરતું નથી.
✓ જો પ્રોજેક્ટ તેના શેડ્યુલ કરતાં પાછળ હશે તો પ્રોગ્રામર ની સંખ્યા વધારીને તે સમયગાળો ઘટાડી શકો છો.	✓ નવા પ્રોગ્રામર ને લાંબો સમય તે પ્રોજેક્ટને જાણવા માટે લાગે છે. તેથી તે વધુ delay થાય છે.

✓ જો પ્રોજેક્ટ કોઈ third party ને આપી દેવામાં આવે તો મેનેજમેન્ટ relax રહશે કારણ કે તે થર્ડપાર્ટી સોફ્ટવેર ને બનાવશે.	✓ જો તે organization સોફ્ટવેર પ્રોજેક્ટ કેવી રીતે નિયંત્રિત કરવા એ સમજી શકે નહીં તો તે જોખમમાં રહેશે જ્યારે એ outsource ને આપવામાં આવશે.

User Myths

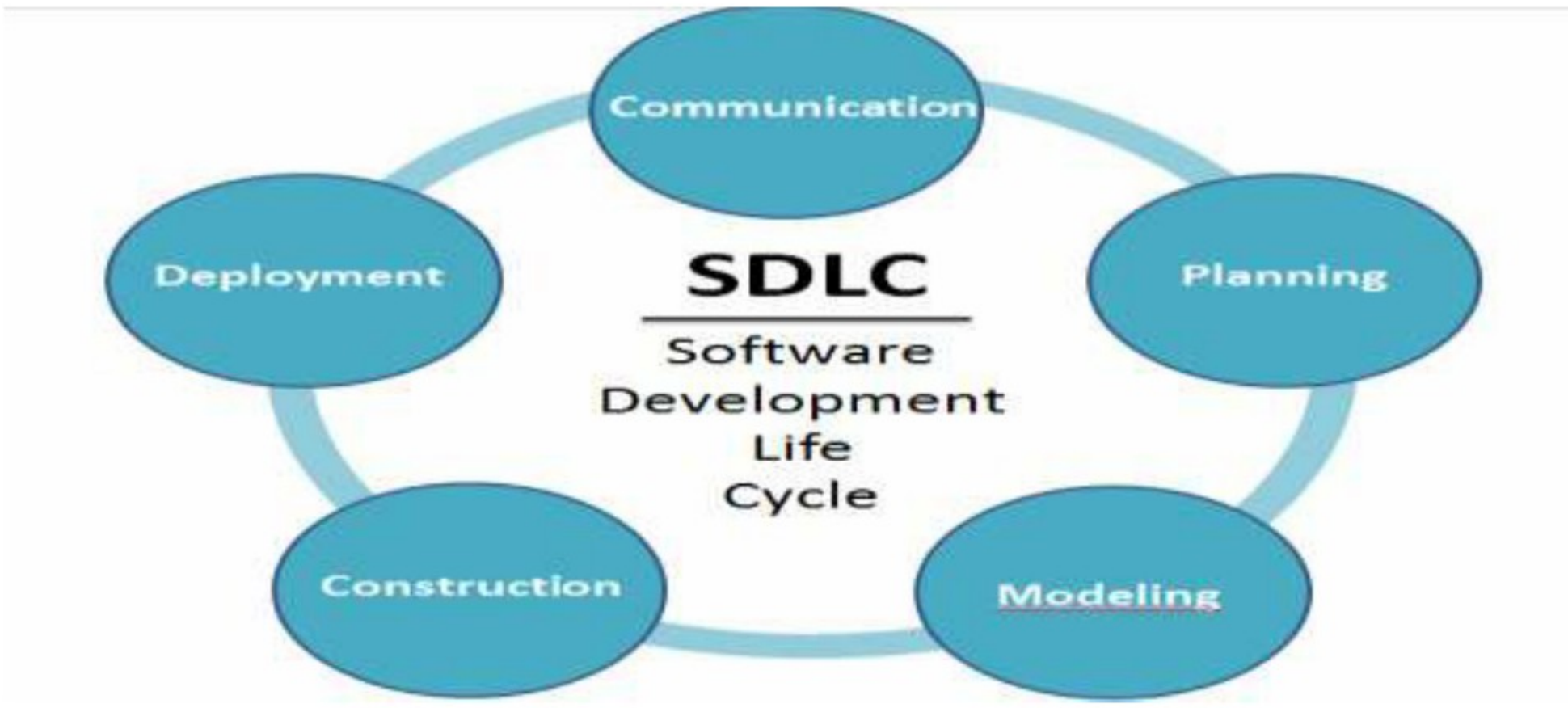
Myths	Reality
✓ Brief requirement તે શરૂ કરવા માટે પૂરતી છે. પછી ની વિગતવાર જરૂરિયાતો પાછળથી ઉમેરીશું.	✓ અપૂર્ણ જરૂરિયાતો થી સોફ્ટવેર fail થઈ શકે છે. ✓ જો પાછળ થી જરૂરિયાતો ઉમેરીએ તો development process ફરી થી કરવી પડે છે.
✓ સોફ્ટવેર ને સરળતાથી બદલી શકાય છે કારણ કે તે flexible છે.	✓ જો પાછળથી ફેરફાર કરવામાં આવે તો તેનું ફરીથી designing અને વધારાના સાધનો જરૂરી છે.

Developer Myths

Myths	Reality
✓ જો એકવાર પ્રોગ્રામ લખાય જાય, તો સોફ્ટવેરને complete કહી શકીએ.	✓ સૉફ્ટવેર user ને software complete બની ગયા પછી પણ તેમાં 60% કરતાં વધારે પ્રયાસો જરૂરી છે.
✓ પ્રોગ્રામ execute થાય પછી તેની ગુણવત્તા માપી શકાય છે.	✓ સોફ્ટવેર quality assurance ની મદદથી સોફ્ટવેરની ગુણવત્તા વિકાસ પ્રક્રિયા દરમિયાન કોઈપણ તબક્કા માં માપવામાં આવે છે.
✓ બીનજરૂરી documentation પ્રોજેકટ ને સ્લો કરે છે.	✓ યોગ્ય documentation ગુણવત્તામાં વધારો કરે છે કે જે પુનઃ કાર્ય ને ઘટાડવા માટે જરૂરી છે.

SDLC life cycle phases

- software life cycle model (also called process model)
- Life cycle model software product બનાવવા માટે ના બધા કાર્યો ને દર્શાવે છે.



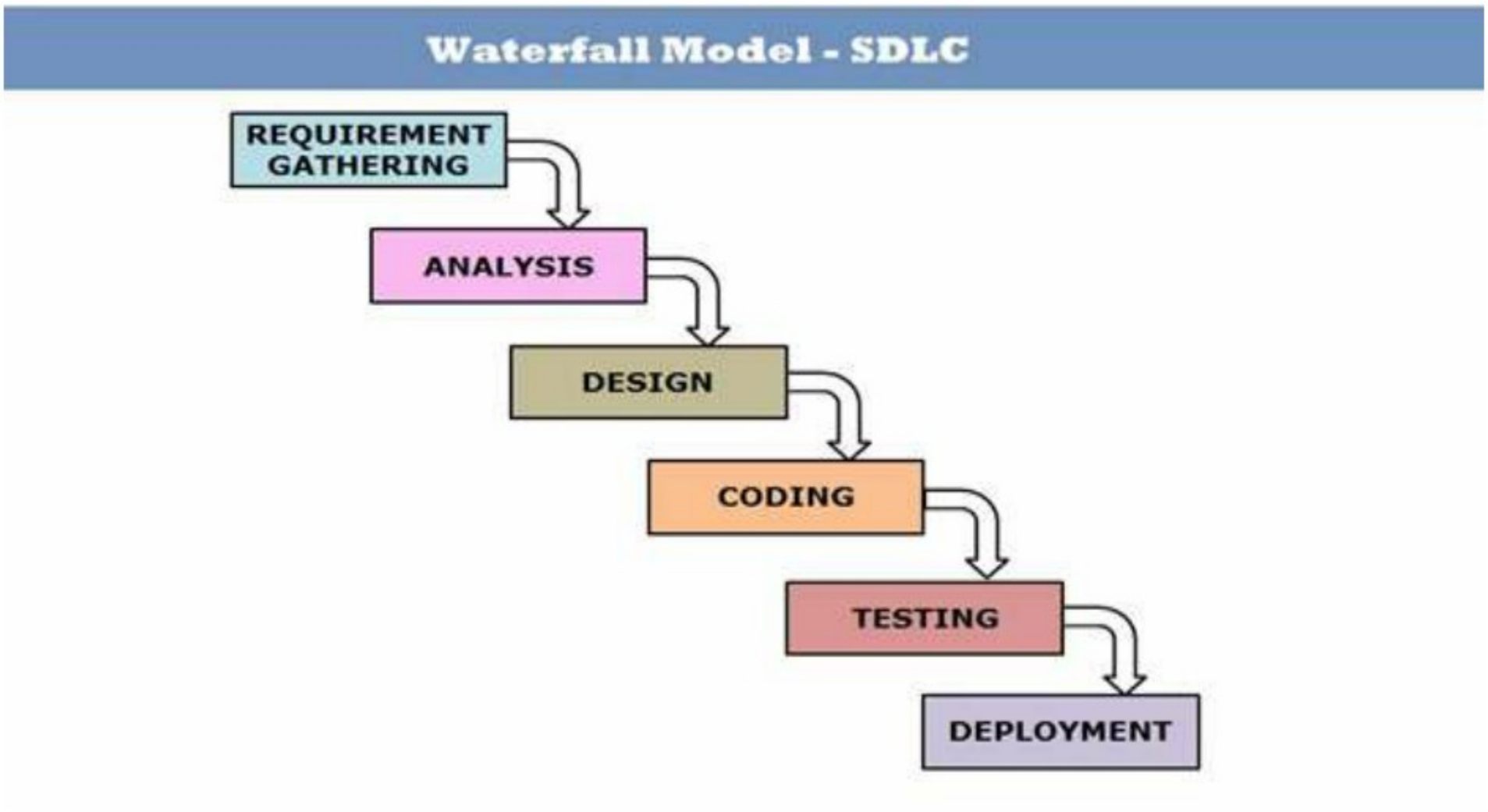
- **Communication:** આ activity customer સાથે વાતચીત કરીને તેની જરૂરિયાતો અને અન્ય બીજી પ્રવૃતિઓ ભેગી કરે છે.
- **Planning:** તે સાધનો,timelineતથા ટેકનીકલ અને management risk ને દર્શાવે છે
- **Modeling:** modelસારી જરૂરિયાતો ને સમજવા તથા તે જરૂરિયાતોને મેળવવા માટે બનાવવામાં આવે છે.
- **Construction:** અહીં codeબને છે.અને તેનું testingથાય છે.
- **Deployment:** અહીં સોફ્ટવેર નુ complete અને અર્ધપૂર્ણ versionગ્રાહક માટે રજૂ કરવામાં આવશે. અને તેનું customer feedback આપે છે.
- **Deployment:** અહીં સોફ્ટવેર નુ complete અને અર્ધપૂર્ણ versionગ્રાહક માટે રજૂ કરવામાં આવશે. અને તેનું customer feedback આપે છે.

List out Software Process models

- Waterfall model (linear sequential model)
- Incremental process model
- Prototyping model
- The spiral model
- Rapid application development model (RAD model)

Classical waterfall model (linear sequential model)

- જ્યારે problem ની requirements સારી રીતે સમજી હોય ત્યારે waterfall modelનો ઉપયોગ થાય છે.



- જ્યારે problem ની requirements સારી રીતે સમજી હોય ત્યારે waterfall modelનો ઉપયોગ થાય છે.
- તેનો work flow communication થી deployment સુધી linearહોય છે.
- આ મોડેલ ને ક્લાસિક લાઈફ સાઈકલ અથવા linear sequential model કહેવામાં આવે છે.

When to use:

- જ્યારે રિક્વાયરમેન્ટ બધી સારી રીતે સમજી હોય અને clear હોય ત્યારે ઉપયોગ થાય છે.
- Product definitions સ્ટેબલ હોય છે.
- ટેકનોલોજી ખબર હોય છે
- રિસોર્સ પૂરતા હોય અને એક્સપર્ટાઈઝ પૂરતા હોય.
- પ્રોજેક્ટ નાનો હોય.

Advantages:

- સરળ રીતે બની શકે અને મેનેજ કરી શકાય.

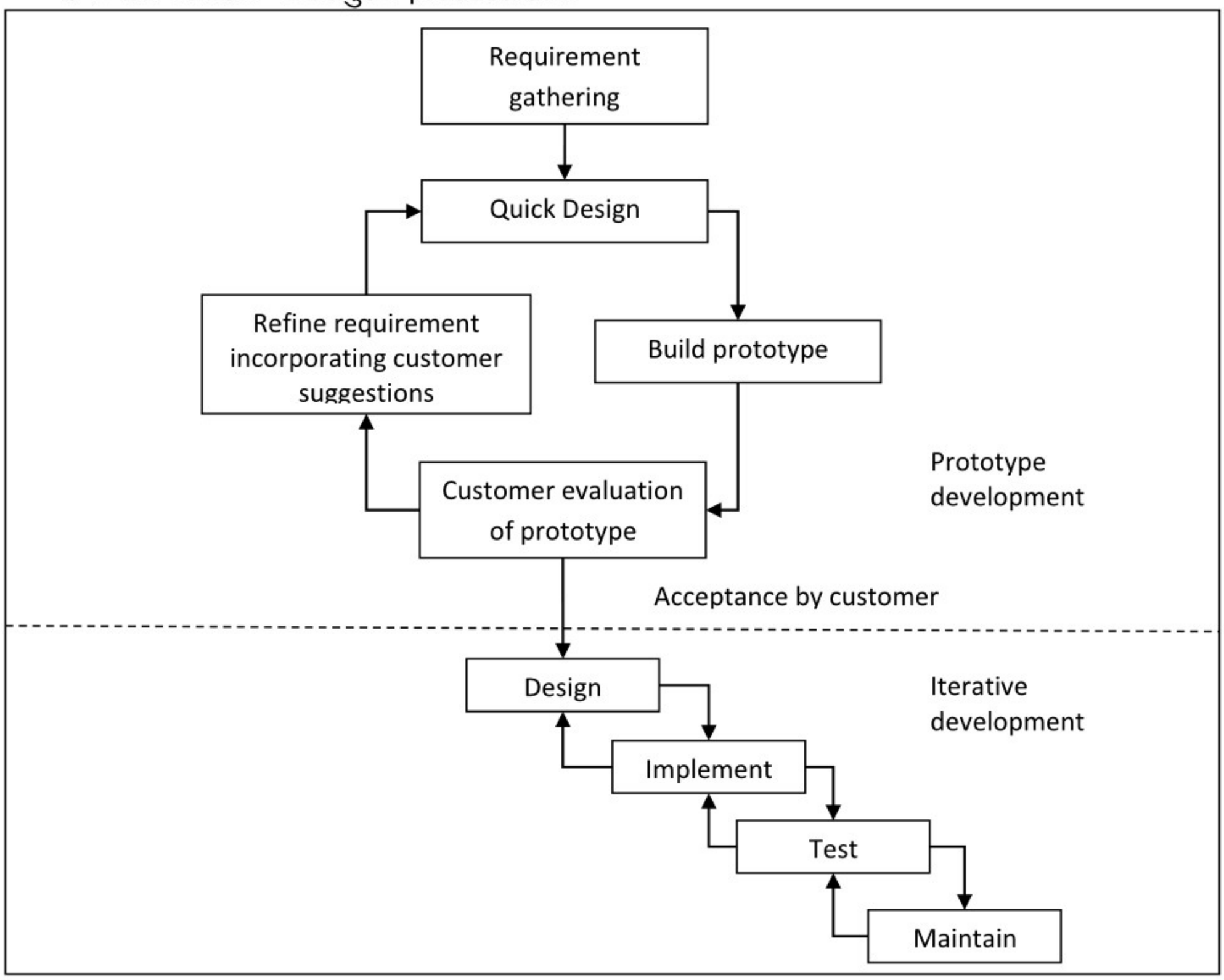
Disadvantages:

- પાછળથી કઈ ફેરફારો કરી શકાય નહી, જે મોટાભાગના cases માંજરૂરી છે.
- ડેવલોપમેન્ટ દરમિયાન વર્કિંગ version હાજરમાં હોતું નથી. તેથી તે મેજર ભુલો વડે ડેવલપ થાય છે.
- મોટા પ્રોજેક્ટ માટે યોગ્ય નથી

Prototype model

- Prototype model real system ઈમ્પ્લીમેન્ટેશન માટેનું સેમ્પલ છે.
- Prototype model મા લિમિટેડ functionality, low reliability એ actual સોફ્ટવેર કરતા ઓછી હોય છે.

- આ મોડલનો ઉપયોગ customerની રિક્વાયરમેન્ટ ને સમજવા માટે થાય છે:
  - ✓ સ્ક્રીન કેવી રીતે જોઈ શકાય.
  - ✓ કેવી રીતે યુઝર ઇન્ટરફેસ behave કરશે
  - ✓ કેવી રીતે સિસ્ટમ આઉટપુટને produce કરશે



- જ્યારે ડેવલોપમેન્ટ ટીમ પાસે રિક્વાયરમેન્ટ ક્લિયર ના હોય ત્યારે prototype model નો ઉપયોગ થાય છે
- Figure માં બતાવ્યા પ્રમાણે પહેલા ફેઈસ માં Prototype develop થાય છે. તે iterative ડેવલોપમેન્ટ સાઈકલ ને ફોલો કરે છે.
- આ મોડલમાં પ્રોટોટાઈપ, રિક્વાયરમેન્ટ gathering phases થી સ્ટાર્ટ કરે છે. તે પછી quick ડિઝાઈન કરીને prototype બનાવવામાં આવે છે.
- ડેવલોપર તે prototype ને કસ્ટમરને ચેક કરવા માટે આપે છે. કસ્ટમરના ફીડબેકના આધારે રિક્વાયરમેન્ટ ને સુધારીને prototype ને સુધારવામાં કરવામાં આવે છે.
- આ customer ફીડબેક અને મોડીફાઇ ની સાઇકલ ત્યાં સુધી continues રહે છે. જ્યાં સુધી કસ્ટમર prototype ને approve ના કરે ત્યાં સુધી, જ્યારે કસ્ટમર એ prototype ને approve કરે ત્યારે actual system એ iterative waterfall ના approach થી ડેવલોપ કરવામાં આવે છે.

Advantages

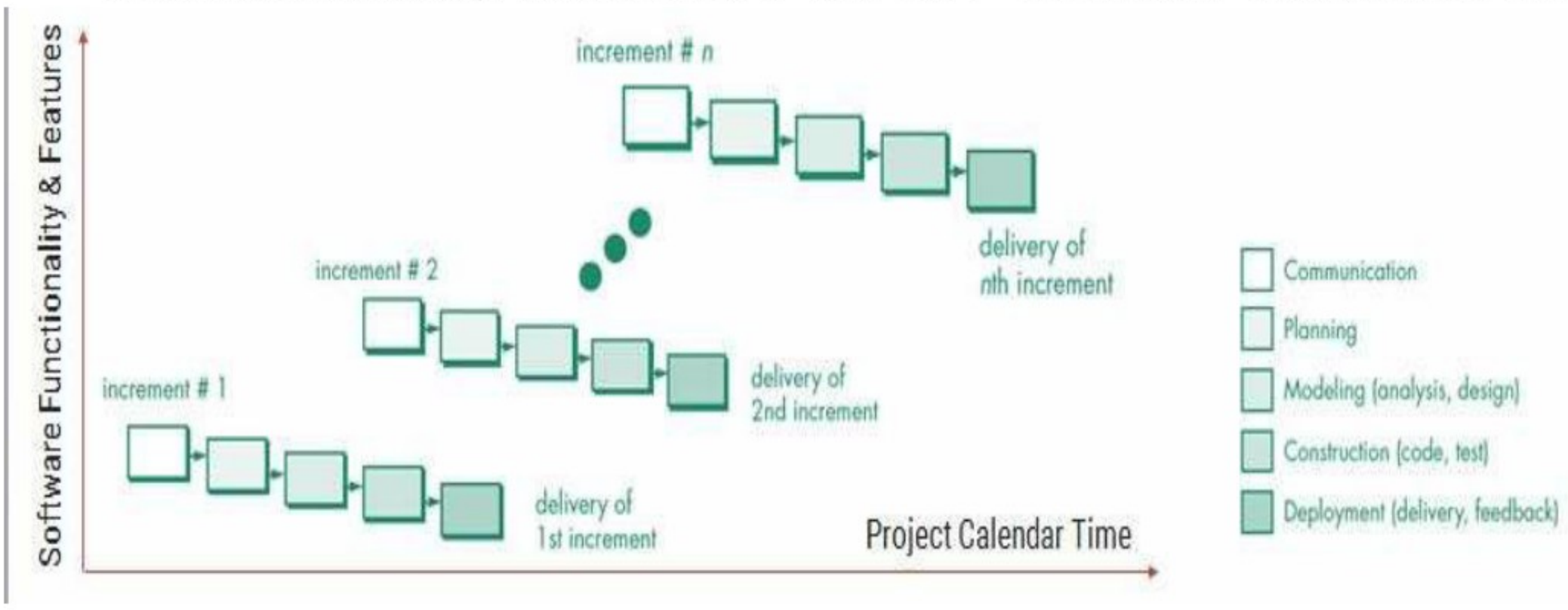
- Error ને વહેલા શોધી શકાય છે.
- યુઝર સિસ્ટમ સાથે એક્ટિવ હોય છે જેથી વધુ ચોક્કસ રિક્વાયરમેન્ટ મળી રહે છે.
- આ પદ્ધતિમાં સિસ્ટમનું વર્કિંગ મોડેલ આપવામાં આવે છે, તેથી યુઝરજે સિસ્ટમ ડેવલપ થઇ છે તેને સારી રીતે સમજી શકે છે.

Disadvantages

- Prototype model મા કસ્ટમરનુ involvement જરૂરી છે. જે હંમેશા માટે શક્ય હોતું નથી..

Increment Model

- ઇન્ક્રિમેન્ટલ મોડલ માં સોફ્ટવેર રિક્વાયરમેન્ટ ને પહેલા અલગ–અલગ models માંdivideકરવામાંથાય છે.



- ડેવલોપમેન્ટ team પહેલા સિસ્ટમ નુ કોર મોડલ ડેવલપ કરે છે.
- તે core model ને બીજા મોડલની સર્વિસની જરૂર નથી.
- Noncore model ને core model ની સર્વિસની જરૂર છે.
- દરેક evolutionary version develop iterative waterfall modelથી કરવામાં આવે છે.
- Productના દરેક successive વર્ઝન સોફ્ટવેર પહેલાના વર્ઝન કરતા વધુ વર્ક કરવા માટે capable હોય છે.
- For example જો આપણે word processing સોફ્ટવેર ઇન્ક્રિમેન્ટલ મોડલથી બનાવીએ તો,તે તેના પહેલા ઇન્ક્રીમેન્ટ મા બેઝિક file મેનેજમેન્ટ,એડિટિંગ અને ડોક્યુમેન્ટ production function આપે છે.
- સેકન્ડ increment મા થોડુ વધુ સારું આપે છે.
- ત્રીજા increment માં સ્પેલિંગ અને ગ્રામર ચેકિંગ આપે છે.
- Forth increment માં એડવાન્સ page layout ની ક્ષમતા આપે છે.

When to use?

- જ્યારે complete systemની બધી રિક્વાયરમેન્ટ સારી રીતે આપેલી અથવા તો સમજી હોય પરંતુ બિઝનેસ ડેડલાઈનમાં સોફ્ટવેર બનાવવા માટે available હોય નહીં ત્યારે incremental model ઉપયોગ થાય છે.

Advantages

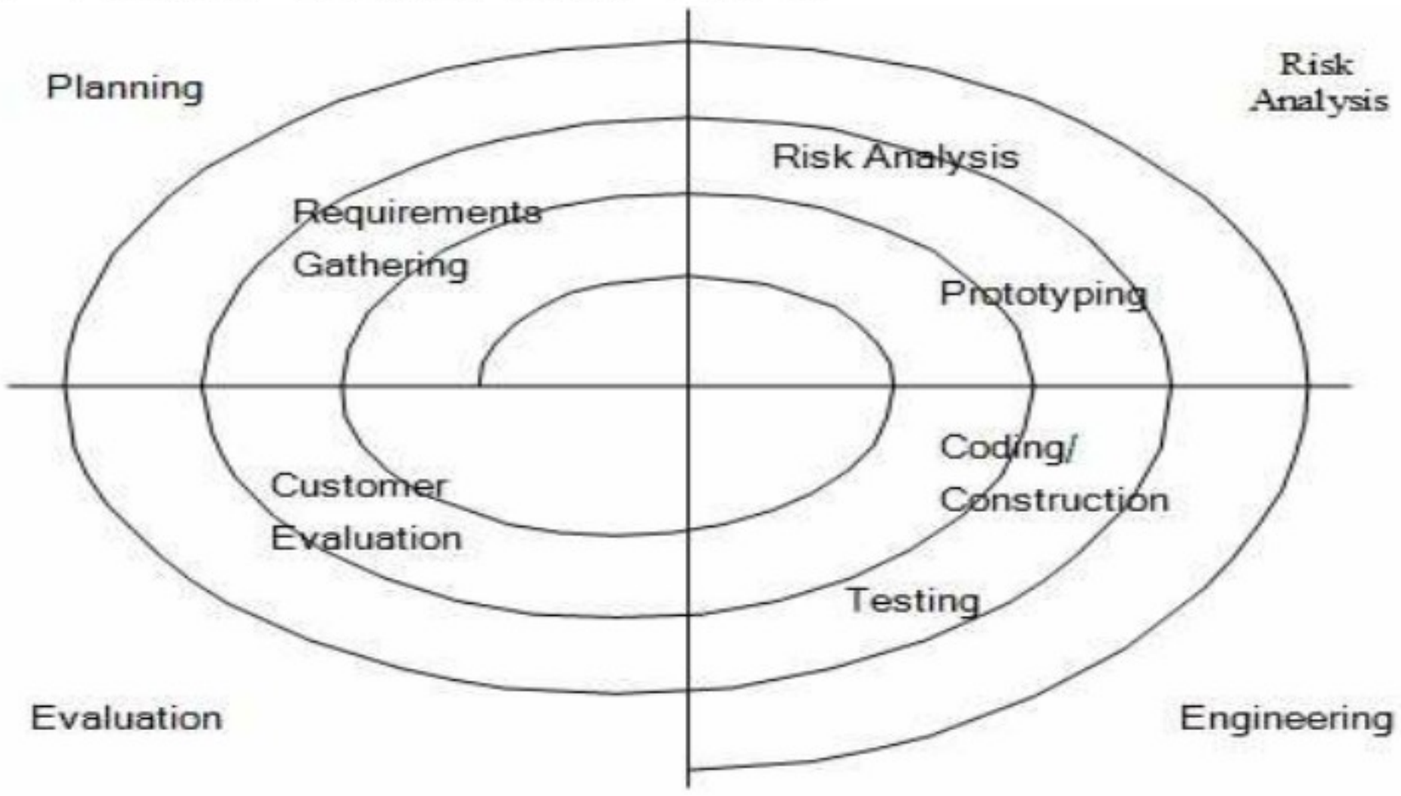
- Software life cycle દરમિયાન વર્કિંગ સોફ્ટવેર ઝડપથી અને વેહેલા બની જાય છે.
- તે સરળ રીતે ટેસ્ટ કરી શકાય છે.
- Riskને easily મેનેજ કરી શકાય છે કારણ કે risk ને ઓળખીને iteration દરમિયાન જ handle કરી શકાય છે.
- પહેલા ઈન્ક્રીમેન્ટ મા કોર પ્રોડક્ટ ને ડેવલોપ કરવા માટે ઓછા costઅને ટાઈમ ની જરૂર પડે છે.

Disadvantages

- બધા ઈન્ક્રીમેન્ટ દરમિયાન resulting cost અને schedule ને મેનેજ કરવા અઘરા થાય છે.

Spiral model

- Spiral model figure માં દર્શાવ્યા પ્રમાણે છે.
- Spiral model નું diagrammatic representationઘણી loop સાથે spiralજેવું દેખાય છે.
- Spiral model મા loop fix હોતી નથી. Spiral modelની દરેક લુપ સોફ્ટવેર પ્રોસેસના phase ને represent કરે છે.
- For example સૌથી અંદરની loop feasibility study સાથે સંકળાયેલી હોય. તેના પછીની loop requirement specification તેના પછી ની ડિઝાઈન અને soon.
- આ મોડેલમાં બધા phase ચાર સેક્ટરમાં split થયેલા છે.



Planning

- આ phaseની અંદર સોફ્ટવેર analyst કસ્ટમર ની સાથે કોમ્યુનિકેશન કરીને જરૂરિયાત પ્રમાણની રિક્વાયરમેન્ટ ને ભેગી કરે છે.
- પ્રોજેક્ટના alternatives અને objective નક્કી થયેલા હોય છે.

Risk Analysis

- બધા possible alternatives,જે કોસ્ટ ઈફેક્ટિવ પ્રોજેક્ટ ડેવલોપ કરવામાં મદદ કરી શકે છે તેનું એનાલિસિસ

કરવામાં આવે છે.

- આ phase મા તમામ possible risk ને ઓળખીને તેને resolve કરવામાં આવે છે.

Engineering

- આ phase માં સોફ્ટવેર બનાવવા માટે જે કોડિંગ જરૂરી છે તે બધું થાય છે.અને testing પણ આ phaseમાં થાય છે.
- પ્રોજેક્ટનું actual development એન્જિનિયરિંગ phase મા કરવામાં આવે છે.

Evaluation

- આ phase માં ડેવલોપ થયેલ સોફ્ટવેર ને કસ્ટમર ચેક કરે છે.
- Evolution થયા પછી જો કસ્ટમર વધુ features ને add કરવા ઈચ્છતો હોય તો iteration પ્રોસેસ ચાલુ કરે છે.

Why we use spiral model?

- જ્યારે high risk પ્રોજેક્ટ ડેવલપ કરવું હોય ત્યારે.
- Cost અને risk evolution important હોય.
- યુઝરને તેમની જરૂરિયાતો પૂરતી ખબર ના હોય.
- રિક્વાયરમેન્ટ અધરી હોય.

Advantages

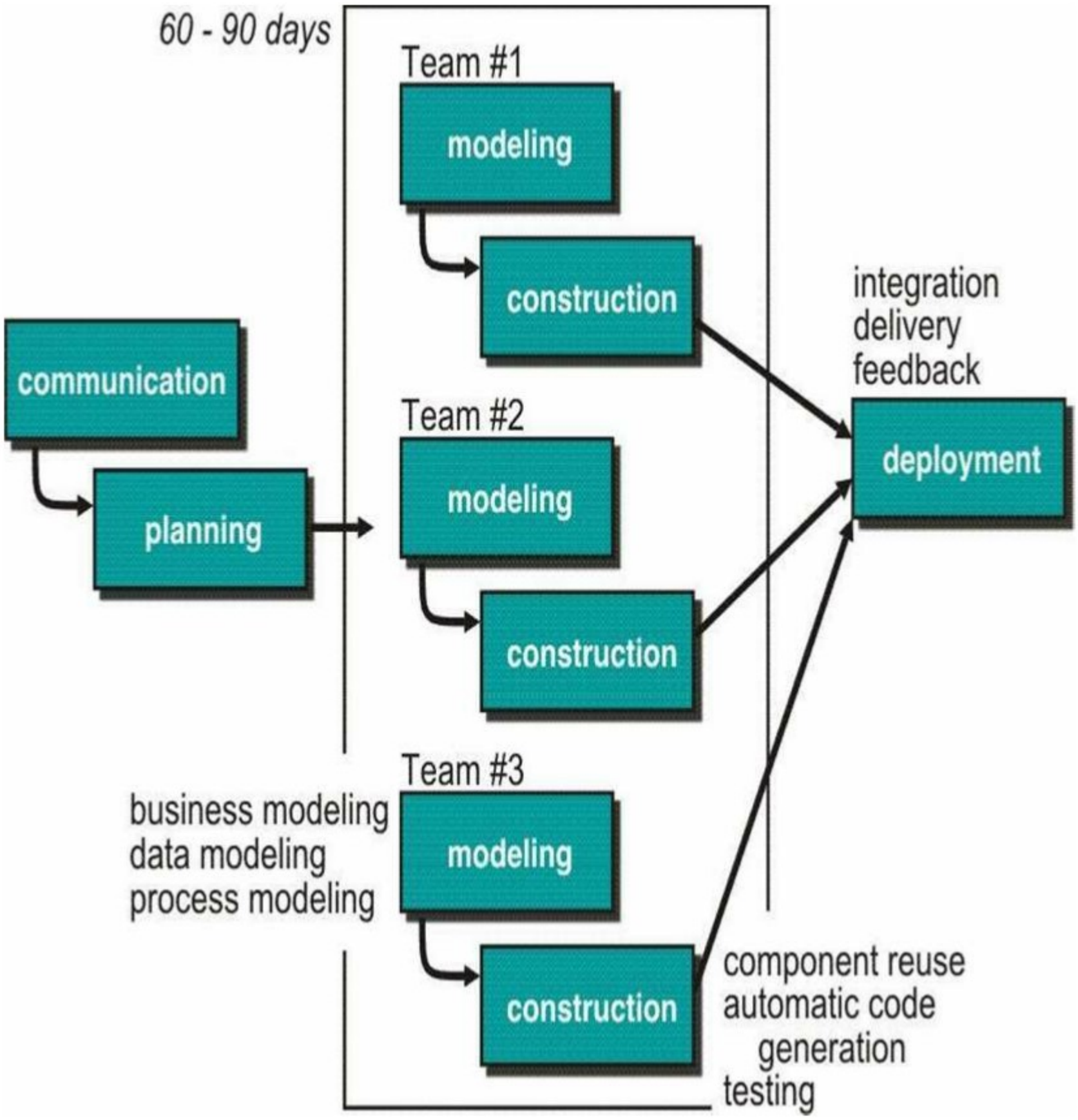
- Strong approval અને ડોક્યુમેન્ટેશન control
- વધારાની functionality પાછળ થી ઉમેરી શકાય છે.
- સોફ્ટવેર લાઈફ સાઈકલ માં વહેલો પ્રોડ્યૂસ થાય છે

Disadvantages

- Use કરવા માટે costly model પડે છે.
- પ્રોજેકટ ની સફળતા રિસ્ક એનાલિસિસ phase પર વધુ dependent હોય છે.
- નાના પ્રોજેકટ માટે સારી રીતે વર્ક કરતું નથી.

Rad model (rapid application development)

- Rapid Application Development (RAD) એ incremental software process model કહેવાય છે.
- તે software components નો ફરી થી ઉપયોગ કરે છે.
- જો રિક્વાયરમેન્ટ સારી રીતે સમજેલી હોય તો RAD process થી ડેવલોપમેન્ટ team, fully functional system ઓછા ટાઈમમાં બનાવી શકે છે. (E.g. 60 to 90 days).
- Communication એ એક્ટીવીટી છે. જેમાં કસ્ટમરcommunication કરીને રિક્વાયરમેન્ટ ને ભેગી કરશે.
- Planning જરૂરી છે કારણકે ઘણી સોફ્ટવેર ટીમ અલગ અલગ સિસ્ટમ માં કરે parallel work કરે છે.
- Modeling includes three major phases -
  1. Business modeling
  2. Data modeling
  3. Process modeling



- **Business Modeling:** બિઝનેસ મોડલ માં information નો flows ની ઓળખાણઅલગ અલગ બિઝનેસ function વચ્ચે થાય છે.
- **Data Modeling:** ડેટા મોડેલિંગ, બિઝનેસ મોડેલિંગ પાસેથી information ભેગી કરીને તેનો ઉપયોગ બિઝનેસ માટે ઉપયોગ થતા ડેટા object ને દર્શાવવા કરવા માટે થાય છે.
- **Process Modeling:** જો ડેટા object માં કોઈ change હોય તો તે આ phases મા દર્શાવાય થાય છે.Data object ને એડ કરવા ડીલીટ કરવા અથવા મોડીફાઇ કરવા માટે નું process description આપે છે.data object મા કઈ ફેરફાર હોય તો તે આ phase મા દર્શાવાય છે.
- **Construction:** જે સોફ્ટવેર કમ્પોનન્ટ existing હોય તેનો ઉપયોગ અને ઓટોમેટિક કોડ જનરેટ કરે છે. Deployment માં સોફ્ટવેર ની ડીલેવરી કરીને કસ્ટમર પાસેથી ફીડબેક મેળવે છે.

Program vs. Software

Program	Software
પ્રોગ્રામ એ લોજિકલ સ્ટેટમેન્ટ છે. જે કોઈ ચોક્કસ કાર્ય માટે લખવામાં આવે છે.	સોફ્ટવેર પ્રોગ્રામ નો સમૂહ છે જે કોઈ સારું ફંક્શન કરવા માટે બનાવવામાં આવે છે.
પ્રોગ્રામ તેમના પર્સનલ ઉપયોગ માટે એક સિંગલ યુઝર દ્વારા વિકસાવવામાં આવે છે.	સોફ્ટવેર એવી એપ્લિકેશન છે જે એક અથવા વધુ સોફ્ટવેર ડેવલપર દ્વારા ડિઝાઇન કરવામાં આવે છે
પ્રોગ્રામ સાઈઝમા નાના હોય છે અને તેમા મર્યાદિત કાર્યક્ષમતા છે	સોફ્ટવેર અત્યંત મોટા હોય છે અને તેમાં એક કરતા વધુ યુઝર હોય છે.
પ્રોગ્રામનો author પોતે જ પ્રોગ્રામ ને વાપરે છે તેથી અન્ય ઇન્ટરફેસ અને ડોક્યુમેન્ટ્સની જરૂર નથી	સોફ્ટવેર ને ઘણા વાપરનાર હોય છે તેથી તેમાં સારું ઇન્ટરફેસ અને સારું ડોક્યુમેન્ટેશન હોવું જોઈ
પ્રોગ્રામમાં માત્ર પ્રોગ્રામ કોડ હોય છે.	સોફ્ટવેર માં પ્રોગ્રામ કોડ સાથે ડિઝાઇન ડોક્યુમેન્ટ , ટેસ્ટ ડોક્યુમેન્ટ અને યુઝર મેન્યુઅલ હોય છે.
એક પ્રોગ્રામ ડેવલોપ પ્રોગ્રામર ની પોતાની શૈલી અનુસાર કરી શકાય છે.	સોફ્ટવેરનું ડેવલોપમેન્ટ સોફ્ટવેર એન્જિનિયરિંગનાં સિદ્ધાંતો વાપરીને કરવામાં આવે છે